Learning the Kernel Matrix with Low-Rank Multiplicative Shaping

Tomer Levinboim and **Fei Sha**
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
{*tomer.levinboim, feisha*}@*usc.edu*

Abstract

Selecting the optimal kernel is an important and difficult challenge in applying kernel methods to pattern recognition. To address this challenge, multiple kernel learning (MKL) aims to learn a kernel from a combination of base kernel functions that perform optimally on the task. In this paper, we propose a novel MKL-themed approach to combine base kernels that are multiplicatively shaped with low-rank positive semidefinitve matrices. The proposed approach generalizes several popular MKL methods and thus provides more flexibility in modeling data. Computationally, we show how these low-rank matrices can be learned efficiently from data using convex quadratic programming. Empirical studies on several standard benchmark datasets for MKL show that the new approach often improves prediction accuracy statistically significantly over very competitive single kernel and other MKL methods.

1 Introduction

Kernel methods have been successfully applied to many classification and recognition tasks (Scholköpf and Smola 2001). However, selecting the optimal kernel for a given task so far remains a challenging open problem. For most users, choosing the right kernel is largely based on trial and error. A more appealing approach is to learn a single composite kernel from a fixed set of base kernels. This general framework is known as "multiple kernel learning" (MKL) (Lanckriet et al. 2004; Chapelle et al. 2002; Ong, Smola, and Williamson 2005; Argyriou, Micchelli, and Pontil 2005).

Early research has focused on learning convex combinations of the base kernels (Lanckriet et al. 2004; Bach, Lanckriet, and Jordan 2004; Kloft et al. 2011; Rakotomamonjy et al. 2008). This formulation is appealing because the combination coefficients can be efficiently computed with convex optimization. Despite the seemingly restrictive formulation, several empirical studies have demonstrated that learning convex combinations can significantly improve classification accuracy (Gehler and Nowozin 2009; Lampert 2009; Vedaldi et al. 2009; Longworth and Gales 2008).

Other forms of combinations have also been explored (Lewis, Jebara, and Noble 2006; Gönen and Alpaydin 2008; Xu et al. 2010; Yang et al. 2009; Cortes, Mohri, and Rostamizadeh 2009). In particular, Cortes, Mohri, and Rostamizadeh (2009) proposed to combine kernels polynomially with the Hadamard product between base kernels. These types of nonlinear combinations give rise to higher-order interactions among base kernels and are thus more flexible than ordinary convex combination. This flexibility appears to be beneficial in practice: extensive studies have shown that nonlinear combinations attain lower classification errors than convex combination (Gönen and Alpaydin 2011). However, nonlinear MKL formulations often lead to non-convex optimization problems where a global optimum is not guaranteed.

Can we harvest the benefits of both worlds? To this end, we propose a novel formulation of combining base kernels. The key insight of our approach is to identify previous schemes as two opposing extremes. In convex combination, the elements of a base kernel matrix are *homogeneously* weighted by a single scalar (the combination coefficient). In nonlinear combinations with Hadamard products, the base kernel matrix is weighted element-wisely by another kernel matrix whose elements are in general not constant.

Our approach imposes additional and special structures on weighting base kernel matrices, thus interpolating between these two extremes. Concretely, our weighting matrices are positive semidefinite and composed of block (sub)matrices. Within each block, the elements are the same. This structure offers more flexibility than convex combination, which essentially weights a kernel using a single block. At the same time, the structure is more restrictive than the generic kernel matrices used in nonlinear combinations with Hadamard products, as our weighting matrices are low-rank and thus have fewer degrees of freedom.

Using low-rank matrices to shape base kernels provides a desirable tradeoff between flexibility and computational tractability. These matrices multiplicatively (thus nonlinearly) interact with the base kernels, yet they can be learned efficiently with convex optimization. By decreasing the degrees of freedom, our method has less risk of overfitting than generic fully-ranked kernel matrices do.

We present empirical studies on several standard benchmark datasets that demonstrate that our method often attains statistically significant improvement in classification accuracy over other MKL methods.

The paper is organized as follows. In section 2 we briefly review MKL and related work. In section 3 we describe our approach for MKL in details. Section 4 presents the results from our empirical studies. We conclude and discuss future work in section 5.

2 Background

We briefly review existing approaches for multiple kernel learning (MKL). We start by describing how kernels are used in support vector machines (SVMs) for classification. We then describe two popular MKL methods which inspired our own method.

Note that using SVMs as a case example is not overly restrictive — ideas developed for learning kernels in SVMs can often be adapted to other types of kernelized methods for classification and regression.

2.1 Kernels in support vector machines

For binary classification, an SVM computes a hyperplane as its decision boundary. The hyperplane is uniquely defined as it maximizes the margin — the minimum distance of the data points to the hyperplane. Formally, given N labeled training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{\mathsf{N}}$, the normal vector of the hyperplane is determined by the solution to the following convex optimization problem, referred as the *dual formulation*,

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{e} - \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{y} = 0, \quad \alpha_n \geq 0, \ \forall \ n \end{aligned} \tag{1}$$

where $\boldsymbol{\alpha}$ is the N-dimensional vector of dual variables, one per training sample. $\boldsymbol{y}$ is the vector of all labels $(y_1, \ldots, y_{\mathsf{N}})^{\mathrm{T}}$. $\boldsymbol{Y}$ denotes the diagonal matrix with $\boldsymbol{y}$ as diagonal elements and $\boldsymbol{e}$ denotes an all-one vector. For simplicity, we have assumed that the training data is separable.

The kernel matrix $\boldsymbol{K} \in \mathbb{R}^{\mathsf{N}\times\mathsf{N}}$ plays a pivotal role in constructing the classifier. Its elements are defined as $K_{ij} = \langle\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle$, the inner product between nonlinearly transformed features $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$. By the "kernel trick", the inner product can be equivalently computed using a kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ without explicitly specifying $\phi(\cdot)$. Common choices for kernel functions include the polynomial kernel $K(\boldsymbol{x}, \boldsymbol{z}) = (1 + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{z})^d$, or the Gaussian RBF kernel $K(\boldsymbol{x}, \boldsymbol{z}) = \exp\{-\|\boldsymbol{x} - \boldsymbol{z}\|_2^2/\sigma^2\}$), and others.

By choosing a specific kernel function, we commit to a particular feature transformation or representation of the data. For a given task however, identifying the optimal kernel — either the right type of kernel or by selecting the right parameters for a given kernel type (such as the degree d or bandwidth σ^2 in the above functions) — is a challenging and unsolved problem. To overcome this challenge, MKL aims to directly learn an optimal kernel from the data.

2.2 Convex combination for MKL

The main idea of MKL is to identify the desired kernel by combining a fixed set of base kernel functions $\{K_m(\cdot,\cdot)\}_m^{\mathsf{M}}$. Popular choices for base kernels are Gaussian RBF kernels using only subsets of features or varying over several values of bandwidths. One major constraint is that the composited kernel should be positive semidefinite.

A simple way to maintain this constraint is to use a convex combination of the base kernels. The desired kernel function $K(\cdot,\cdot)$ is parameterized as (Lanckriet et al. 2004),

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{m=1}^{\mathsf{M}} \beta_m K_m(\boldsymbol{x}, \boldsymbol{z}) \tag{2}$$

where the combination coefficients are constrained to be nonnegative and sum to one. To identify $\{\beta_m\}_{m=1}^{\mathsf{M}}$, we substitute $K(\cdot,\cdot)$ of eq. (2) into the objective function of eq. (1) and arrive at:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \max_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{e} - \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{Y}\left(\sum_m \beta_m \boldsymbol{K}_m\right)\boldsymbol{Y}\boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{y} = 0, \ \alpha_n \geq 0, \ \forall \ n \\ & \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{e} = 1, \ \beta_m \geq 0, \ \forall \ m \end{aligned} \tag{3}$$

Eq. (3) is a convex optimization program where the objective function is convex in $\boldsymbol{\beta}$ and concave in $\boldsymbol{\alpha}$. The program can be solved efficiently (we defer the discussion on optimization techniques to the appendix).

2.3 Nonlinear combination for MKL

Arguably, convex combination is just one of infinitely many possibilities of composing kernels. To incorporate nonlinear interactions among base kernels, Cortes, Mohri, and Rostamizadeh (2009) proposed the following parametric form,

$$K(\boldsymbol{x}, \boldsymbol{z}) = \sum_{m=1}^{\mathsf{M}}\sum_{l=1}^{\mathsf{M}} \beta_m\beta_l K_m(\boldsymbol{x}, \boldsymbol{z}) K_l(\boldsymbol{x}, \boldsymbol{z}) \tag{4}$$

In a slightly more compact form, the final kernel matrix is a linear combination of Hadamard products between base kernel matrices $\boldsymbol{K} = \sum_{ml} \beta_m\beta_l \boldsymbol{K}_m \circ \boldsymbol{K}_l$. Since Hadamard products preserve positive semidefiniteness, the resulting kernel is valid as long as the coefficients β_m are nonnegative. The quadratic interaction in eq. (4) can also be extended to high-order polynomials, though we focus on the former.

Substituting the kernel function in eq. (4) into the SVM dual problem of eq. (1) leads to a non-convex program[1].

A recent study has shown that nonlinear combination such as eq. (4) generally outperform convex combination (Gönen and Alpaydin 2011), though at the cost of identifying solutions that are only locally optimal.

How can we harvest benefits of both convex and nonlinear combinations? In the following, we describe our approach.

3 Low-rank Multiplicative Shaping

A key insight to our approach is to intuitively identify the convex and nonlinear MKL approaches as two opposing extremes. We begin by elaborating on this intuition. We then describe our formulation and approach (numerical optimization details are deferred to the appendix). We conclude the section with a brief discussion on related work.

[1]Carefully restricting β_m seems to lead to convexity, cf. Prop. 4 in (Cortes, Mohri, and Rostamizadeh 2009), though it is not clear whether that has been used in practice.

3.1 Intuition

Kernels measure similarities between samples in their feature space. Intuitively, we desire a kernel function that outputs high values on pairs of samples that belong to the same class, and low values on pairs that do not. Thus, MKL can be viewed as adjusting each base kernel's elements until the resulting kernel satisfies our desideratum.

For convex combination, each base kernel $\boldsymbol{K}_m$ is adjusted "coarsely" — the value of $K_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is multiplied by a scalar β_m irrespective of the relation between samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Thus, we cannot selectively adjust the kernel entries. For example, suppose $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same class while $\boldsymbol{x}_k$ is from the other. If both $K_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $K_m(\boldsymbol{x}_i, \boldsymbol{x}_k)$ are high valued (as the base kernel is not "perfect"), we would ideally want to increase the former and decrease the later. This is impossible with convex combination.

On the other end, nonlinear combinations with Hadamard products offer a very "fine" shaping. The value of $K_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is multiplied by another kernel $K_l(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Thus, even if the former does not meet our requirement, hopefully the latter will correct. Obviously, this kind of shaping is very detailed — the change to the base kernel $\boldsymbol{K}_m$ is not homogeneous as in the case of convex combination. With the extra flexibility comes the cost of computation: the optimization is no longer convex.

Can we find a middle-ground shaping mechanism? Our approach explores this notion. In particular, we argue that it suffices to have shapings at the "cluster" level. Each cluster contains a subset of data points, all sharing the same shaping factors — to be made more precise soon.

Choosing the word "cluster" is not incidental. In fact, our approach uses each base kernel to generate partitions of the data points (by running kernelized K-means algorithms). Thus, cluster membership assignments with their associated shaping factors can be viewed as a summary of the base kernel.

To ensure the shaping is maximally adaptive to the data, the cluster-associated shaping factors are optimized from the data directly — we only use the base kernel to generate the cluster structure and ignore its kernel function values afterwards. Because of this decoupling, the optimization is convex and can be efficiently solved.

As opposed to the typically fully ranked matrices used in the nonlinear combination approach (cf. section 2.3), our shaping matrices are low-ranked, due to the use of a small number of clusters. In the following, we formalize these intuitions and present our approach of **low-rank multiplicative shaping (LMS)** for MKL.

3.2 Formulation

Low-rank Multiplicative Shaping (LMS) Assume we are given a set of base kernel functions $\{K_m(\cdot,\cdot)\}_{m=1}^{\mathsf{M}}$. As describe above, we use the base kernels to generate partitions of the data into disjoint clusters. We denote the partitions with $\boldsymbol{S}_p \in \{0,1\}^{\mathsf{N}\times\mathsf{C}}$ where the ik-th element is a binary indicator of whether $\boldsymbol{x}_i$ is assigned to the k-th cluster. For simplicity, we assume the number of clusters is fixed at C for all partitions.

We use a different subscript p to imply that the number of used partitions, denoted P, is independent of M. Indeed, several partitions can be generated using the same base kernel and a selection process can be applied (In our experiments, we used kernelized K-means with different initializations and kept partitions with high normalized mutual information between cluster membership and classification labels).

For each kernel $\boldsymbol{K}_m$ and partition $\boldsymbol{S}_p$ pair, we follow the intuition described above and define C^2 cluster-level shaping factors as follows. If $\boldsymbol{x}_i$ is assigned to cluster s and $\boldsymbol{x}_j$ is assigned to cluster t, then the base kernel value is multiplicatively shaped by the factor

$$\theta_{st} \cdot K_m(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{5}$$

where all θ_{st} form a matrix $\boldsymbol{\Theta}_{m,p} \in \mathbb{R}^{\mathsf{C}\times\mathsf{C}}$. For reasons soon to be clear, we impose the constraint $\boldsymbol{\Theta}_{m,p} \succeq 0$, i.e., $\boldsymbol{\Theta}_{m,p}$ is positive semidefinite (PSD).

We define the following *low-rank multiplicative shaping matrix*, which is of key importance to our approach,

$$\boldsymbol{\Omega}_{m,p} = \boldsymbol{S}_p \boldsymbol{\Theta}_{m,p} \boldsymbol{S}_p^{\mathrm{T}} \tag{6}$$

It is easy to verify that the shaped kernel value in eq. (5) is the ij-th element of the matrix $\boldsymbol{\Omega}_{m,p} \circ \boldsymbol{K}_m$, namely, the Hadamard product between the base kernel and the low-rank shaping matrix. The proof of the following propositions is omitted due to its simplicity.

Proposition 3.1. *rank*$(\boldsymbol{\Omega}_{m,p}) \le$ *rank*$(\boldsymbol{\Theta}_{m,p}) \le \mathsf{C} \le \mathsf{N}$

Proposition 3.2. $\boldsymbol{\Theta}_{m,p} \succeq 0 \implies \boldsymbol{\Omega}_{m,p} \succeq 0 \implies \boldsymbol{\Omega}_{m,p} \circ \boldsymbol{K}_m \succeq 0$

Having defined the low-rank shaping matrices $\boldsymbol{\Omega}_{m,p}$, we are now ready to present our MKL approach.

MKL with LMS We parameterize our kernel as

$$\boldsymbol{K} = \sum_{m,p} \boldsymbol{\Omega}_{m,p} \circ \boldsymbol{K}_m \tag{7}$$

Note that the kernel is PSD due to Proposition 3.2. The shaping matrices $\boldsymbol{\Theta}_{m,p}$ are optimized directly from data, by solving the following convex program:

$$\begin{aligned} \min_{\{\boldsymbol{\Theta}_{m,p}\}} \max_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^T e - \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{Y} (\boldsymbol{K} + \lambda \boldsymbol{I}) \boldsymbol{Y} \boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{\alpha}^T \boldsymbol{y} = 0, \ \alpha_n \ge 0 \ \ \forall \ n \\ & \sum_{m,p} \mathrm{Trace}(\boldsymbol{\Theta}_{m,p}) = \mathsf{C} \\ & \boldsymbol{\Theta}_{m,p} \succeq 0 \ \ \forall \, m, p \end{aligned} \tag{8}$$

where the inclusion of the identity matrix $\boldsymbol{I}$ scaled by a small positive constant λ ensures the strict concavity of the objective function in $\boldsymbol{\alpha}$. The constraint on the trace ensures that when the number of the clusters C is one, the formulation degenerates to the convex combination (eq. (2)).

Task Name	Source	N	D
`Advert`	UCI	2359	1554
`Australian`	UCI	690	13
`Splice`	UCI	1527	240
`TIS2007`	UCSD	1980	200
`Spambase`	UCI	4601	57
`German`	UCI	1000	24

Table 1: Datasets used in classification tasks. N: the number of samples. D: the number of feature dimensions. The Source column denotes the origins of the datasets.

Extensions to soft-clustering Our formulation can also use soft cluster assignments, as Proposition 3.2 is valid for any choice of $\boldsymbol{S}_p$. One particular assignment matrix that we used in our empirical studies is given by

$$\boldsymbol{S}_p = \gamma \boldsymbol{S}_p^{\text{HARD}} + (1-\gamma)\boldsymbol{e}\boldsymbol{e}^{\text{T}}/\mathsf{C} \tag{9}$$

which is a convex combination between a hard assignment matrix and a matrix that assigns each data point to the C clusters with equal probability. Note that, when $\gamma = 0$, the above formulation "backs off" to the approach of convex combination eq. (2).

Numerical optimization Eq. (8) is a convex optimization program as both the objective function and the constraints are convex in $\boldsymbol{\Theta}_{m,p}$. We describe our numerical optimization technique for learning shaping matrices in the appendix.

3.3 Related work

Other forms of shaping matrices have also been proposed (Lewis, Jebara, and Noble 2006; Gönen and Alpaydin 2008; Xu et al. 2010; Yang et al. 2009). In those methods, the base kernel $K_m(\boldsymbol{x}, \boldsymbol{z})$ is multiplied by $\eta_m(\boldsymbol{x})\eta_m(\boldsymbol{z})$. This is equivalent to learning a rank-one PSD matrix $\boldsymbol{\eta}$ which is composed with the kernel as $\boldsymbol{\eta} \circ \boldsymbol{K}_m$. However, learning either $\eta_m(\boldsymbol{x})$ explicitly or $\boldsymbol{\eta}$ results in non-convex optimization. In contrast, the weights we learn are specified in pairwise and form block structures according to the clustering. This style of shaping should not be considered as *localized* and leads to ***convex*** optimization.

Close in spirit to our approach is the recent work in (Mu and Zhou 2011). Their method also pre-computes a clustering structure from the training data, but requires solving a non-convex graph embedding problem to learn kernels.

4 Experimental Results

We validate our approach empirically by comparing it to the methods described in section 2 and other common baselines classifiers. The results demonstrate the advantages of our method in improving classification accuracy as well as in using fewer support vectors.

4.1 Setup

Datasets Table 1 lists the 6 datasets we have used in our experiments. Samples with missing values were filtered out. For the biological classification tasks `Splice` and `TIS2007`, a 1-to-4 encoding was used to represent amino-acids. The datasets were obtained from the UCI Machine Learning Repository (Frank and Asuncion 2010) and the UCSD MKL Repository (Lanckriet et al. 2009)

Baselines We compare our method to 4 other methods: i) BestSingle: the best single kernel among all base kernels; ii) Uniform: the uniform (equal weight) combination of base kernels; iii) SimpleMKL: the convex combination of base kernels, with the combination coefficients computed using the SimpleMKL algorithm proposed in (Rakotomamonjy et al. 2008); iv) Quadratic: the non-linear combination eq. (4) proposed in (Cortes, Mohri, and Rostamizadeh 2009), after being adapted to classification tasks from its original regression setting. A recent study shows that Quadratic generally outperforms the aforementioned methods (Gönen and Alpaydin 2011), as well as Localized MKL (Gönen and Alpaydin 2008).

Base kernels We investigate two different settings for selecting base kernels, as in (Gönen and Alpaydin 2011) and in the instructions at the UCSD MKL Repository. We restrict our base kernels to a set of either linear or Gaussian RBF kernels (with bandwidth normalized by the median distance) generated from non-overlapping attributes of the training data. For the Gaussian RBF base kernels, we added one additional kernel that uses all attributes. We typically use $\mathsf{M} = 5$ or 6 base kernels.

Our method For our method of Low-rank Multiplicative Shaping (LMS), we use what is described in section 3 with the following extension: i) each partitions could have a different number of clusters $\mathsf{C}_p \in \{4, 16, 32, 48, 56, 64\}$. The trace constraint is changed from C to $\max_p \mathsf{C}_p$. ii) soft clustering assignment is used by choosing a γ from 3 possible values 0.05, 0.1, and 0.15.

For each of the M base kernels and for each C_p, a partition of the training data is computed using kernelized K-means on the training data. We chose $\mathsf{P} \leq \mathsf{M}$ partitions, whose normalized mutual information between cluster membership assignment and the ground-truth labels are the highest.

Parameter tuning and cross-validation For all methods, ours or the baselines, we calculated results over 15 random splits of the data (40/30/30% for training/validation/test respectively). The SVM regularization parameter C (for inseparable datasets) was cross-validated by selecting from $\{10^2, \ldots, 10^6\}$.

4.2 Classification accuracy

Tables 2 reports the averaged classification errors on test data over 15 random splits, as well as the standard errors, computed as the standard deviation in classification errors divided by $\sqrt{15}$.

When linear base kernels are used, our method (LMS) generally outperforms all other 4 methods, attaining the lowest classification errors. In 3 out of 6 datasets, the improvement in accuracy (over the second best method) is significant, beyond one standard error.

When Gaussian RBF base kernels are used, while our

Dataset	P	γ	LMS	SimpleMKL	Quadratic	Uniform	BestSingle
Advert	4	0.15	3.78 (0.14)	3.89 (0.19)	5.33 (0.22)	4.24 (0.20)	4.51 (0.24)
Australian	6	0.10	13.88 (0.58)	14.17 (0.56)	20 (0.99)	14.27 (0.56)	14.24 (0.58)
Splice	6	0.15	**2.98 (0.19)**	3.4 (0.16)	5.42 (0.23)	3.41 (0.13)	5.23 (0.24)
tis2007	2	0.10	**9.57 (0.20)**	10.17 (0.29)	19.38 (0.31)	10.89 (0.17)	16.32 (0.38)
Spambase	6	0.15	**6.78 (0.17)**	7.31 (0.17)	17.45 (0.23)	7.33 (0.20)	14.86 (0.17)
German	4	0.15	24.73 (0.28)	24.87 (0.38)	29.84 (0.25)	24.47 (0.33)	29.8 (0.14)

(a) Linear base kernels

Dataset	P	γ	LMS	SimpleMKL	Quadratic	Uniform	BestSingle
Advert	6	0.05	3.56 (0.18)	3.66 (0.20)	3.55 (0.19)	3.71 (0.17)	3.61 (0.18)
Australian	2	0.15	13.91 (0.62)	14.46 (0.66)	14.17 (0.58)	14.17 (0.64)	14.59 (0.47)
Splice	2	0.10	**2.41 (0.15)**	3.27 (0.20)	2.79 (0.17)	3.17 (0.16)	3.38 (0.15)
tis2007	6	0.05	**9.18 (0.25)**	9.66 (0.21)	9.84 (0.23)	10.4 (0.23)	10.32 (0.16)
Spambase	2	0.05	6.22 (0.10)	6.28 (0.10)	6.19 (0.09)	6.37 (0.11)	7.02 (0.17)
German	4	0.05	23.4 (0.42)	23.07 (0.29)	23.27 (0.53)	23.38 (0.34)	24.82 (0.58)

(b) Gaussian RBF base kernels

Table 2: Averaged classification error rates (in %) by different methods on 6 standard benchmark datasets. For our method, the number of partitions P and the soft assignment regularizer γ are also shown. Entries are highlighted if they are statistically significant better than all other methods. See the text for details.

method still outperforms others in general, the improvements are only significant in 2 out of 6 datasets.

Our results are also consistent with a well-known fact: well-tuned best single kernel (cf. BestSingle) and non-adaptive combination of kernels (cf. Uniform) are very competitive baselines and are often very hard to be improved over. In fact, in several datasets, either of the two methods is very close to the second best performing one.

Note that, when $\gamma = 0$, our method reduces to SimpleMKL. Our empirical results from Table 2 show, however, that the optimal γ is strictly positive, illustrating the benefits of using low-rank shaping matrices that nonlinearly interact with base kernels through Hadamard products.

On the other hand, we find that using fully-ranked matrices (such as base kernels themselves) to shape the base kernels, as in Quadratic, does not yield consistent benefits. In particular, in the case of linear base kernels, for the datasets Splice, tis2007 and Spambase, our method does significantly better than Quadratic which is significantly worse than SimpleMKL. There are two possible reasons for the poor performance: the algorithm could be trapped in a local optimum or the method is overfiting due to the use of full-rank matrices.

4.3 The number of support vectors

The number of support vectors is often used as another metric for evaluating MKL approaches. Intuitively, a "well learned" composite kernel might be able to cover the feature space with several base kernels, thus resulting in an overall reduction in the number of support vectors.

Fig. 1 contrasts the averaged percentage of support vectors (of the total number of data points) by our methods and baselines. We can see that our method attains similar low percentages as SimpleMKL, while other methods tend to be higher. Quadratic attains the highest percentage.

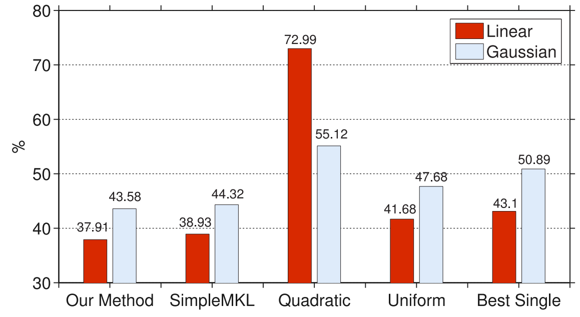

Figure 1: Averaged percentage of support vectors by different methods. See the text for details.

Moreover, except Quadratic, all other methods use higher percentage of support vectors when Gaussian RBF base kernels are used. This is likely due to the fact that RBF kernels have stronger localized properties where the kernel function decays exponentially fast with respect to the distances among data points.

5 Conclusions

Various kernel composition schemes have been studied in the literature. Ideally, one would want the composition to be flexible (spanning a large subspace of all possible kernels), yet remain computationally tractable.

The basic convex combination scheme results in a convex optimization program for learning the combination coefficients. Many numerical optimization techniques have been developed to solve this problem efficiently. Despite its success on several large-scale problems in terms of improving classification accuracy, it does not consistently outperform a well-tuned single kernel or a simple rule-based combination schemes such as trivially averaging all kernels (Zhuang, Tsang, and Hoi 2011; Gönen and Alpaydin 2011). Nonlinear combination of base kernels, such as those using Hadamard products to combine kernels, has been shown to be superior to convex combination (Gönen and Alpaydin 2011). However, these methods are generally not based on convex optimization and thus a global optimum is not guaranteed. Furthermore, they often use more support vectors than simpler schemes, including single kernels (cf. Fig. 1).

This paper proposes a new approach that has the benefits of both worlds. Our low-rank multiplicative shaping approach incorporates nonlinear interactions between kernels by learning low-rank positive semidefinite matrices, and combining them with the base kernels using the Hadamard product. At the same time, learning is cast as a convex optimization, retaining the computational attractiveness of simple schemes. Empirical studies on standard benchmarks clearly demonstrate the advantages of our method: our method often outperforms other methods statistically significantly. Moreover, our method uses fewer support vectors than other approaches.

For future work, we plan to study the utility of our approach on large-scale problems, as well as the possibility of using other MKL learning criteria in lieu of the SVM loss function used in this paper (Cortes, Mohri, and Rostamizadeh 2010).

Acknowledgements

This work is partially supported by NSF IIS#1065243 (F. Sha) and a USC Annenberg Graduate Fellowship (T. Levinboim).

A Numerical Optimization

In this appendix, we provide the details of our numerical method for solving eq. (8).

A.1 General procedure

We employ the "SVM wrapper" approach (Rakotomamonjy et al. 2008; Xu et al. 2010). The approach is an iterative procedure. The outmost iteration minimizes the objective function by updating $\mathbf{\Theta}_{m,p}$, using the method of projected gradient descent. Inside the loop, we solve a SVM in $\boldsymbol{\alpha}$ assuming fixed $\mathbf{\Theta}_{m,p}$ parameters.

Let $\mathbf{\Theta}$ denote the block diagonal matrix composed of all factor matrices $\mathbf{\Theta}_{m,p}$. Program (8) can then be posed as:

$$\begin{aligned} \min_{\mathbf{\Theta}} \quad & g(\mathbf{\Theta}) \\ \text{subject to} \quad & \mathbf{\Theta} \succeq 0,\ \mathsf{Trace}(\mathbf{\Theta}) = \mathsf{C} \end{aligned} \tag{10}$$

where, for a fixed $\mathbf{\Theta}$, $g(\mathbf{\Theta})$ is the standard SVM optimization eq. (1) with the kernel computed using eq. (7):

$$\begin{aligned} g(\mathbf{\Theta}) = \max_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^T \boldsymbol{e} - \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{Y}(\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{Y}\boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{\alpha}^T \boldsymbol{y} = 0,\ \boldsymbol{\alpha}_n \geq 0,\ \forall\ n \end{aligned} \tag{11}$$

The inclusion of $\lambda \boldsymbol{I}$ in $\boldsymbol{K}$ ensures that the objective function eq. (8) is strictly concave in $\boldsymbol{\alpha}$ and therefore has a unique maximizer. Consequently, $g(\mathbf{\Theta})$ is differentiable in $\mathbf{\Theta}$, due to Danskin's theorem ((Bonnans and Shapiro 1998), Theorem 4.1). Specifically, the gradient $\nabla g(\mathbf{\Theta})$ exists and is given by

$$\frac{\partial g(\mathbf{\Theta})}{\partial \mathbf{\Theta}_{m,p}} = -\frac{1}{2}\boldsymbol{S}_p^{\mathrm{T}}(\boldsymbol{K}_m \circ [\boldsymbol{Y}\boldsymbol{\alpha}^* \boldsymbol{\alpha}^{*\mathrm{T}} \boldsymbol{Y}])\boldsymbol{S}_p \tag{12}$$

where $\boldsymbol{\alpha}^*$ is the unique solution to eq. (11) at $\mathbf{\Theta}$.

```
  Input: ${\boldsymbol{K}_m}_{m=1}^{\mathsf{M}}$, ${\boldsymbol{S}_p}_{p=1}^{\mathsf{P}}$, $\mathbf{\Theta}^0$
  Output: $\mathbf{\Theta}^t$
1 $t \leftarrow 0$;
  repeat
2     $\boldsymbol{\alpha}^t := \mathrm{SVMSOLVER}(\mathbf{\Theta}^t)$;
3     compute $\nabla g(\mathbf{\Theta}^t)$ at $\boldsymbol{\alpha}^t$;
4     line search for a suitable step size $\eta_t$ and update
      $\mathbf{\Theta}^{t+1} := \mathrm{PROX}_{\mathcal{C}}(\mathbf{\Theta}^t - \eta_t \cdot \nabla g(\mathbf{\Theta}^t))$;
5     $t \leftarrow t + 1$;
  until stopping condition ;
```

Algorithm 1: Main steps in our numerical optimization

We apply the proximal method, a popular method for constrained optimization (Rockafellar 1976). Iteratively, we use gradient descent to update $\mathbf{\Theta}$ from its current value, which might violate the constraints, and then project it back to the feasible region defined by the constraints. The pseudocode in Algorithm 1 sketches the main steps of our optimization procedure.

Concretely, Step 2 of the Algorithm solves the SVM of eq. (11). Step 3 computes the gradient according to eq. (12). Step 4 performs a line search to find a suitable step size and project $\mathbf{\Theta}$ back to the feasible region, denoted by $\mathcal{C}$. The projection is computed using the prox-operator.

The step size η_t is chosen with Armijo's rule. Starting from its maximal value η^{max} (tunable in practice), we successively decrease η_t by a factor of $\beta < 1$ until $\mathbf{\Theta}^{t+1}$ yields a sufficient decrease (with respect to the chosen step size) in the objective function.

The stopping condition is simply $||\mathbf{\Theta}^{t+1} - \mathbf{\Theta}^t||_2 \leq \epsilon$ for some $\epsilon > 0$.

A.2 Computing $\mathrm{PROX}_{\mathcal{C}}(\hat{\mathbf{\Theta}})$

The prox-operator computes the projection of $\hat{\mathbf{\Theta}}$ onto the convex feasible region $\mathcal{C}$:

$$\begin{aligned} \min_{\mathbf{\Theta}} \quad & \|\mathbf{\Theta} - \hat{\mathbf{\Theta}}\|_F^2 \\ \text{subject to} \quad & \mathbf{\Theta} \in \mathcal{C} \end{aligned} \tag{13}$$

We show this problem can be reformulated as a quadratic program (QP) despite the semidefinite constraint on $\mathbf{\Theta}$.

Since $\hat{\mathbf{\Theta}}$ is symmetric, it can be diagonalized: $\hat{\mathbf{\Theta}} = \boldsymbol{V}\hat{\boldsymbol{D}}\boldsymbol{V}^{\mathrm{T}}$, where $\boldsymbol{V}$ is a unitary matrix and $\hat{\boldsymbol{D}}$ is diagonal. Therefore, the objective function can be rewritten as:

$$\|\mathbf{\Theta} - \hat{\mathbf{\Theta}}\|_F^2 = \|\boldsymbol{V}^{\mathrm{T}}\mathbf{\Theta}\boldsymbol{V} - \hat{\boldsymbol{D}}\|_F^2 \tag{14}$$

Let $\boldsymbol{A}$ denote $\boldsymbol{V}^{\mathrm{T}}\mathbf{\Theta}\boldsymbol{V}$. We observe that the minimizer $\mathbf{\Theta}$ of eq. (13) necessarily makes $\boldsymbol{A}$ diagonal, since $\hat{\boldsymbol{D}}$ is.

In other words, the minimizer $\mathbf{\Theta}$ is in the form of $\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^{\mathrm{T}}$, where $\boldsymbol{A}$ is the solution to

$$\begin{aligned} \min_{\boldsymbol{A}} \quad & \|\boldsymbol{A} - \hat{\boldsymbol{D}}\|_F^2 \\ \text{subject to} \quad & \boldsymbol{A} \succeq 0,\ \mathsf{Trace}(\boldsymbol{A}) = \mathsf{C} \end{aligned} \tag{15}$$

where we have rewritten the feasible region $\mathcal{C}$ in terms of the variable $\boldsymbol{A}$. To transform (15) into QP form, we simply consider only the diagonal terms of the (diagonal) matrices $\boldsymbol{A}$ and $\hat{\boldsymbol{D}}$.

Since $\boldsymbol{A}$ is diagonal, eq. (15) is therefore an instance of quadratic programming and can be solved efficiently. Note that the size of $\boldsymbol{A}$ does *not* depend on the number of data points, therefore, the algorithm is scalable to large-scale problems.

References

Argyriou, A.; Micchelli, C. A.; and Pontil, M. 2005. Learning convex combinations of continuously parameterized basic kernels. In *Proc. of the 18th Annual Conference on Learning Theory (COLT)*, 338–352. Springer-Verlag.

Bach, F. R.; Lanckriet, G. R. G.; and Jordan, M. I. 2004. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proc. of the 21st International Conference on Machine learning (ICML)*, 41–48. ACM.

Bonnans, J. F., and Shapiro, A. 1998. Optimization problems with perturbations: A guided tour. *SIAM Review* 40(2):pp. 228–264.

Chapelle, O.; Vapnik, V.; Bousquet, O.; and Mukherjee, S. 2002. Choosing multiple parameters for support vector machines. *Journal of Machine Learning Research* 46:131–159.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2009. Learning non-linear combinations of kernels. In Bengio, Y.; Schuurmans, D.; Lafferty, J. D.; Williams, C. K. I.; and Culotta, A., eds., *23rd Annual Conference on Neural Information Processing Systems (NIPS)*. MIT Press. 396–404.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2010. Two-stage learning kernel algorithms. In Fürnkranz, J., and Joachims, T., eds., *Proc. of the 27th International Conference on Machine Learning (ICML)*, 239–246. Omnipress.

Frank, A., and Asuncion, A. 2010. UCI machine learning repository. http://archive.ics.uci.edu/ml.

Gehler, P. V., and Nowozin, S. 2009. On feature combination for multiclass object classification. In *Proc. of 12th International Conference on Computer Vision (ICCV)*, 221–228.

Gönen, M., and Alpaydin, E. 2008. Localized multiple kernel learning. In *Proc. of the 25th International Conference on Machine Learning (ICML)*, 352–359.

Gönen, M., and Alpaydin, E. 2011. Multiple kernel learning algorithms. *Journal of Machine Learning Research* 12:2211–2268.

Kloft, M.; Brefeld, U.; Sonnenburg, S.; and Zien, A. 2011. Lp-norm multiple kernel learning. *Journal of Machine Learning Research* 12:953–997.

Lampert, C. H. 2009. *Kernel Methods in Computer Vision*. Now Publishers Inc.

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

Lanckriet, G.; Bach, F.; Srebro, N.; and McFee, B. 2009. UCSD multiple kernel learning repository. http://mkl.ucsd.edu.

Lewis, D. P.; Jebara, T.; and Noble, W. S. 2006. Nonstationary kernel combination. In *Proc. of the 23rd International Conference on Machine Learning (ICML)*, 553–560.

Longworth, C., and Gales, M. J. F. 2008. Multiple kernel learning for speaker verification. In *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1581–1584.

Mu, Y., and Zhou, B. 2011. Non-uniform multiple kernel learning with cluster-based gating functions. *Neurocomputing* 74(7):1095–1101.

Ong, C. S.; Smola, A. J.; and Williamson, R. C. 2005. Learning the kernel with hyperkernels. *Journal of Machine Learning Research* 6:1043–1071.

Rakotomamonjy, A.; Bach, F.; Canu, S.; and Grandvalet, Y. 2008. SimpleMKL. *Journal of Machine Learning Research* 9:2491–2521.

Rockafellar, R. T. 1976. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization* 14(5).

Scholköpf, B., and Smola, A. J. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press.

Vedaldi, A.; Gulshan, V.; Varma, M.; and Zisserman, A. 2009. Multiple kernels for object detection. In *Proc. of 12th International Conference on Computer Vision (ICCV)*, 606–613.

Xu, Z.; Jin, R.; Yang, H.; King, I.; and Lyu, M. R. 2010. Simple and efficient multiple kernel learning by group Lasso. In *Proc. of the 27th International Conference on Machine Learning (ICML)*, 1175–1182.

Yang, J.; Li, Y.; Tian, Y.; Duan, L.; and Gao, W. 2009. Group-sensitive multiple kernel learning for object categorization. In *Proc. of 12th International Conference on Computer Vision (ICCV)*, 436–443.

Zhuang, J.; Tsang, I. W.; and Hoi, S. C. H. 2011. Two-layer multiple kernel learning. *JMLR W&CP* 15:909–917.