Margin and radius based multiple Kernel Learning

Do, Thi Thanh Huyen; Kalousis, Alexandros; Woznica, Adam; Hilario, Mélanie

How to cite

DO, Thi Thanh Huyen et al. Margin and radius based multiple Kernel Learning. In: Machine Learning and Knowledge Discovery in Databases. Buntine, Wray, Grobelnik, Marko, Mladeni, Dunja & Shawe-Taylor, John (Ed.). Bled (Slovenia). Berlin, Heidelberg : Springer, 2009. p. 330–343. (Lecture Notes in Computer Science) doi: 10.1007/978-3-642-04180-8_39

This publication URL: https://archive-ouverte.unige.ch/unige:6892
Publication DOI: 10.1007/978-3-642-04180-8_39

Margin and Radius Based Multiple Kernel Learning

Huyen Do, Alexandros Kalousis, Adam Woznica, and Melanie Hilario

University of Geneva, Computer Science Department
7, route de Drize, Battelle batiment A, 1227 Carouge, Switzerland
{Huyen.Do,Alexandros.Kalousis,Adam.Woznica,Melanie.Hilario}@unige.ch

Abstract. A serious drawback of kernel methods, and Support Vector Machines (SVM) in particular, is the difficulty in choosing a suitable kernel function for a given dataset. One of the approaches proposed to address this problem is Multiple Kernel Learning (MKL) in which several kernels are combined adaptively for a given dataset. Many of the existing MKL methods use the SVM objective function and try to find a linear combination of basic kernels such that the separating margin between the classes is maximized. However, these methods ignore the fact that the theoretical error bound depends not only on the margin, but also on the radius of the smallest sphere that contains all the training instances. We present a novel MKL algorithm that optimizes the error bound taking account of both the margin and the radius. The empirical results show that the proposed method compares favorably with other state-of-the-art MKL methods.

Keywords: Learning Kernel Combination, Support Vector Machines, convex optimization.

1 Introduction

Over the last few years kernel methods [1,2], such as Support Vector Machines (SVM), have proved to be efficient machine learning tools. They work in a feature space implicitly defined by a positive semi-definite kernel function, which allows the computation of inner products in feature spaces using only the objects in the input space.

The main limitation of kernel methods stems from the fact that in general it is difficult to select a kernel function, and hence a feature mapping, that is suitable for a given problem. To address this problem several several attempts have been recently made to learn kernel operators directly from the data [3,4,5,6,7,8,9,10,11,12]. The proposed methods differ in the objective functions (e.g. CV risk, margin based, alignment, etc.) as well as in the classes of kernels that they consider (e.g. combination of finite or infinite set of basic kernels).

The most popular approach in the context of kernel learning considers a finite set of predefined *basic kernels* which are combined so that the margin-based

W. Buntine et al. (Eds.): ECML PKDD 2009, Part I, LNAI 5781, pp. 330–343, 2009.

objective function of SVM is optimized. The learned kernel K is a linear combination of basic kernels K_i, i.e. $K(\boldsymbol{x}, \boldsymbol{x'}) = \sum_{i=1}^{M} \mu_i K_i(\boldsymbol{x}, \boldsymbol{x'}),\ \mu_i \geq 0$, where M is the number of basic kernels, and $\boldsymbol{x}$ and $\boldsymbol{x'}$ are input objects. The weights μ_i of the kernels are included in the margin-based objective function. This setting is commonly referred to as the Multiple Kernel Learning (MKL).

The MKL formulation has been introduced in [3] as a semi-definite programming problem, which scaled well only for small problems. [7] extended that work and proposed a faster method based on the conic duality of MKL and solved the problem using Sequential Minimal Optimization (SMO). [5] reformulated the MKL problem as semi-infinite linear problem. In [6] the authors proposed an adjustment in the cost function of [5] to improve predictive performance. Although the MKL approach to kernel learning has some limitations (e.g. one has to choose the basic kernels), it is widely used because of its simplicity, interpretability and good performance.

The MKL methods that use the SVM objective function do not exploit the fact that the error bound of SVM depends not only on the separating margin, but also on the radius of the smallest sphere that encloses the data. In fact even the standard SVM algorithms do not exploit the latter, because for a given feature space the radius is fixed. However in the context of MKL the radius is not fixed but is a function of the weights of the basic kernels.

In this paper we propose a novel MKL method that takes account of both radius and margin to optimize the error bound. Following a number of transformations, these problems are cast in a form that can be solved by the two step optimization algorithm given in [6].

The paper is organized as follows. In Section 2 we introduce the general MKL framework. Next, in Section 3 we discuss the various error bounds that motivate the use of the radius. The main contribution of the work is presented in Section 4 where we propose a new method for multiple kernel learning that aims to optimize the margin- and radius-dependent error bound. In Section 5 we present the empirical results on several benchmark datasets. Finally, we conclude with Section 6 where we also present pointers to future work.

2 Multiple Kernel Learning Problem

Consider a mapping of instances $\mathbf{x} \in \mathcal{X}_i$, to a new feature space $\mathcal{H}_i$

$$\mathbf{x} \rightarrow \mathbf{\Phi}_i(\mathbf{x}) \in \mathcal{H}_i \tag{1}$$

This mapping can be performed by a kernel function $K_i(\mathbf{x}, \mathbf{x}')$ which is defined as the inner product of the images of two instances $\mathbf{x}$ and $\mathbf{x}'$ in $\mathcal{H}_i$, i.e. $K_i(\mathbf{x}, \mathbf{x}') = \langle \mathbf{\Phi}_i(\mathbf{x}), \mathbf{\Phi}_i(\mathbf{x}') \rangle$; $\mathcal{H}_i$ may have even infinite dimensionality. Typically, the computation of the inner product in $\mathcal{H}_i$ is done implicitly, i.e. without having to compute explicitly the images $\mathbf{\Phi}_i(\mathbf{x})$ and $\mathbf{\Phi}_i(\mathbf{x}')$.

2.1 Original Problem Formulation of MKL

Given a set of training examples $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l)\}$ and a set of basic kernel functions, $\boldsymbol{Z} = \{K_i(\mathbf{x}, \mathbf{x}') | i := 1, \ldots M\}$, the goal of MKL is to optimize

a cost function $Q(f(\boldsymbol{Z}, \boldsymbol{\mu})(\mathbf{x}, \mathbf{x}'), S)$ where $f(\boldsymbol{Z}, \boldsymbol{\mu})(\mathbf{x}, \mathbf{x}')$ is some positive semi-definite function of the set of the basis kernels, parametrized by $\boldsymbol{\mu}$; most often a linear combination of the form:

$$f(\boldsymbol{Z}, \boldsymbol{\mu})(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{M} \mu_i K_i(\mathbf{x}, \mathbf{x}'), \mu_i \geq 0, \sum_{i}^{M} \mu_i = 1 \tag{2}$$

To simplify notation we will denote $f(\boldsymbol{Z}, \boldsymbol{\mu})$ by $K_{\boldsymbol{\mu}}$. In the remaining part of this work we will only focus on the normalized versions of K_i, defined as:

$$K_i(\mathbf{x}, \mathbf{x}') := \frac{K_i(\mathbf{x}, \mathbf{x}')}{\sqrt{K_i(\mathbf{x}, \mathbf{x}) \cdot K_i(\mathbf{x}', \mathbf{x}')}}. \tag{3}$$

If $K_{\boldsymbol{\mu}}$ is a linear combination of kernels then its feature space $\mathcal{H}_{\boldsymbol{\mu}}$ is given by the mapping:

$$\mathbf{x} \rightarrow \boldsymbol{\Phi}_{\boldsymbol{\mu}}(\mathbf{x}) = (\sqrt{\mu_1}\boldsymbol{\Phi}_1(\mathbf{x}), ..., \sqrt{\mu_M}\boldsymbol{\Phi}_M(\mathbf{x}))^T \in \mathcal{H}_{\boldsymbol{\mu}} \tag{4}$$

where $\boldsymbol{\Phi}_i(\mathbf{x})$ is the mapping to the $\mathcal{H}_i$ feature space associated with the K_i kernel, as this was given in Formula 1.

In previous work within the MKL context the cost function, Q, has taken different forms such as the Kernel Target Alignment, which measures the "goodness" of a kernel for a given learning task [9], or the typical SVM cost function combining classification error and the margin [3,5,6], or as in [4] any of the above with an added regularization term for the complexity of the combined kernel.

3 Margin and Radius Based Error Bounds

There are a number of theorems in statistical learning that bound the expected classification error of the thresholded linear classifier, that corresponds to the maximum margin hyperplane, by quantities that are related to the margin and the radius of the smallest sphere that encloses the data. Below we give two of them that are applicable on linearly separable and non-separable training sets, respectively.

Theorem 1. *[10], Given a training set* $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l)\}$ *of size* l*, a feature space* $\mathcal{H}$ *and a hyperplane* $(\mathbf{w}, b)$*, the margin* $\gamma(\mathbf{w}, b, S)$ *and the radius* $R(S)$ *are defined by*

$$\gamma(\mathbf{w}, b, S) = \min_{(\mathbf{x}_i, y_i) \in S} \frac{y_i(\langle \mathbf{w}, \boldsymbol{\Phi}(\mathbf{x}_i) \rangle + b)}{\|\mathbf{w}\|}$$
$$R(S) = \min_{\mathbf{a}} \max_{i} \|\boldsymbol{\Phi}(\mathbf{x}_i) - \mathbf{a}\|$$

The maximum margin algorithm $L_l : (\mathcal{X} \times \mathcal{Y})^l \rightarrow \mathcal{H} \times \mathbb{R}$ *takes as input a training set of size* l *and returns a hyperplane in feature space such that the margin* $\gamma(\mathbf{w}, b, S)$ *is maximized. Note that assuming the training set is separable means*

that $\gamma > 0$. Under this assumption, for all probability measures P underlying the data S, the expectation of the misclassification probability

$$p_{err}(\mathbf{w}, b) = P(sign(\langle \mathbf{w}, \mathbf{\Phi}(\mathbf{X}) \rangle + b) \neq Y)$$

has the bound

$$E\{p_{err}(L_{l-1}(Z))\} \leq \frac{1}{l} E\left\{ \frac{R^2(Z)}{\gamma^2(L_l(Z), Z)} \right\}$$

The expectation is taken over the random draw of a training set Z of size $l-1$ for the left hand side and l for the right hand side.

The following theorem gives a similar result for the error bound of the linearly non-separable case.

Theorem 2. *[13], Consider thresholding real-valued linear functions $\mathcal{L}$ with unit weight vectors on an inner product space $\mathcal{H}$ and fix $\gamma \in \mathcal{R}^+$. There is a constant c, such that for any probability distribution $\mathcal{D}$ on $\mathcal{H} \times \{-\infty, \infty\}$ with support in a ball of radius R around the origin, with probability $1-\delta$ over l random examples S, any hypothesis $f \in \mathcal{L}$ has error no more than:*

$$err(f)_{\mathcal{D}} \leq \frac{c}{l} \left(\frac{R^2 + \|\boldsymbol{\xi}\|_2^2}{\gamma^2} log^2 l + log \frac{1}{\delta} \right), \tag{5}$$

where $\boldsymbol{\xi} = \boldsymbol{\xi}(f, S, \gamma)$ is the margin slack vector with respect to f and γ.

It is clear from both theorems that the bound on the expected error depends not only on the margin but also on the radius of the data, being a function of the R^2/γ^2 ratio. Nevertheless standard SVM algorithms can ignore the dependency of the error bound on the radius because for a fixed feature space the radius is constant and can be simply ignored in the optimization procedure. However in the MKL scenario where the $\mathcal{H}_{\boldsymbol{\mu}}$ feature space is not fixed but depends on the parameter vector $\boldsymbol{\mu}$ the radius is no longer fixed but it is a function of $\boldsymbol{\mu}$ and thus should not be ignored in the optimization procedure. The radius of the smallest sphere that contains all instances in the $\mathcal{H}$ feature space defined by the $\mathbf{\Phi}(\mathbf{x})$ mapping is computed by the following formula [14]:

$$\begin{aligned} \min_{R, \Phi(\mathbf{x}_0)} \quad & R^2 \\ s.t. \quad & \|\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_0)\|^2 \leq R^2, \forall i \end{aligned} \tag{6}$$

It can be shown that if $K_{\boldsymbol{\mu}}$ is a linear combination of kernels, of the form given in Formula 2, then for the $R_{\boldsymbol{\mu}}$ radius of its $\mathcal{H}_{\boldsymbol{\mu}}$ feature space the following inequalities hold:

$$\max_i(\mu_i R_i^2) \leq R_{\boldsymbol{\mu}}^2 \leq \sum_{i=1}^{M} \mu_i R_i^2 \leq \max_i(R_i^2), \tag{7}$$

$$s.t. \sum_i^M \mu_i = 1$$

where R_i is the radius of the component feature space $\mathcal{H}_i$ associated with the K_i kernel. The proof of the above statement is given in the appendix.

4 MKL with Margin and Radius Optimization

In the next sections we will show how we can make direct use of the dependency of the error bound both on the margin and the radius in the context of the MKL problem in an effort to decrease even more the error bound than what is possible by optimizing only over the margin.

4.1 Soft Margin MKL

The standard l_2-soft margin SVM is based on theorem 2 and learns maximal margin hyperplanes while controlling for the l_2 norm of the slack vector in an effort to optimize the error bound given in equation 5; as already mentioned previously the radius although it appears in the error bound is not considered in the optimization problem due to the fact that for a given feature space it is fixed. The exact optimization problem solved by the l_2-soft margin SVM is [13]:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \ & \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + \frac{C}{2}\sum_{i=1}^{l}\xi_i^2 \\ s.t. \ & y_i(\langle \mathbf{w}, \boldsymbol{\Phi}(\mathbf{x}_i)\rangle + b) \geq 1-\xi_i, \forall i \end{aligned} \tag{8}$$

The solution hyperplane $(\mathbf{w}^*, b^*)$ of this problem realizes the maximum margin classifier with geometric margin $\gamma = \frac{1}{\|\mathbf{w}^*\|}$.

When instead of a single kernel we learn with a combination of kernels $K_{\boldsymbol{\mu}}$ then the radius of the resulting feature space $\mathcal{H}_{\boldsymbol{\mu}}$ depends on the parameters $\boldsymbol{\mu}$ which are also learned. We can profit from this additional dependency and optimize not only for the margin but also for the radius, as Theorems 1 and 2 suggest, in the hope of reducing even more the error bounds than what would be possible by just focusing on the margin.

A straightforward way to do so is to alter the cost function of the above optimization problem so that it also includes the radius. Thus we define the primal form of soft margin MKL optimization problem as follows:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\mu}} \ & \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle R_{\boldsymbol{\mu}}^2 + \frac{C}{2}\sum_{i=1}^{l}\xi_i^2 \\ s.t. \ & y_i(\langle \mathbf{w}, \boldsymbol{\Phi}(\mathbf{x}_i)\rangle + b) \geq 1-\xi_i, \forall i \end{aligned} \tag{9}$$

Accounting for the form $\boldsymbol{\Phi}_{\boldsymbol{\mu}}$ of the feature space $\mathcal{H}_{\boldsymbol{\mu}}$, as it is given in equation 4, this optimization problem can be rewritten as:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\mu}} \ & \frac{1}{2}\sum_{k}^{M}\langle \mathbf{w}_k,\mathbf{w}_k\rangle R_{\boldsymbol{\mu}}^2 + \frac{C}{2}\sum_{i=1}^{l}\xi_i^2 \\ s.t. \ & y_i(\sum_{k}^{M}\langle \mathbf{w}_k, \sqrt{\mu_k}\boldsymbol{\Phi}_k(\mathbf{x}_i)\rangle + b) \geq 1-\xi_i, \forall i \end{aligned} \tag{10}$$

where the $\mathbf{w}$ is the same as that of Formula 9 and equal to $(\mathbf{w}_1, \ldots, \mathbf{w}_M)$, $R^2_{\boldsymbol{\mu}}$ can be computed by equation 6. By letting $\mathbf{w} := \sqrt{\boldsymbol{\mu}}.\mathbf{w}$ then $\mathbf{w}_k := \sqrt{\mu_k}\mathbf{w}_k$ we can rewrite equation 10 as[1]:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\mu}} \quad & \frac{1}{2}\sum_{k}^{M}\frac{\langle \mathbf{w}_k, \mathbf{w}_k\rangle}{\mu_k}R^2_{\boldsymbol{\mu}} + \frac{C}{2}\sum_{i}^{l}\xi_i^2 \\ s.t. \quad & y_i(\sum_{k}^{M}\langle \mathbf{w}_k, \boldsymbol{\Phi}_k(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i, \\ & \sum_{k=1}^{M}\mu_k = 1, \mu_k \geq 0, \forall k \end{aligned} \tag{11}$$

The non-negativity of $\boldsymbol{\mu}$ is required to guarantee that the kernel combination is a valid kernel function; the constraint $\sum_{k=1}^{M}\mu_k = 1$ is added to make the solution interpretable (kernel with bigger weight can be interpreted as more important one) and to get a specific solution (note that if $\boldsymbol{\mu}$ is solution of 11 (without the constraint $\sum_{k=1}^{M}\mu_k = 1$), then $\lambda\boldsymbol{\mu}, \lambda \in \mathbb{R}^+$ is also its solution).

We will denote the cost function of equation 11 by $F(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\mu})$. F is not a convex function, this is probably the main reason why in current MKL algorithms the radius is simply removed from the original cost function, therefore they do not really optimize the generalization error bound.

From the set of inequalities given in equation 7 we have $R^2_{\boldsymbol{\mu}} \leq \sum_k^M \mu_k R_k^2$ and from this we can get:

$$\begin{aligned} F(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\mu}) = \quad & \tfrac{1}{2}\textstyle\sum_k^M \frac{\langle \mathbf{w}_k, \mathbf{w}_k\rangle}{\mu_k} R^2_{\boldsymbol{\mu}} + \frac{C}{2}\sum_i^l \xi_i^2 && \leq \\ & \tfrac{1}{2}\textstyle\sum_k^M \frac{\langle \mathbf{w}_k, \mathbf{w}_k\rangle}{\mu_k} \sum_k^M \mu_k R_k^2 + \frac{C}{2}\sum_i^l \xi_i^2 && = \\ & (\tfrac{1}{2}\textstyle\sum_k^M \frac{\langle \mathbf{w}_k, \mathbf{w}_k\rangle}{\mu_k} + \frac{C}{2\sum_k^M \mu_k R_k^2}\sum_i^l \xi_i^2)\sum_k^M \mu_k R_k^2 && \leq \\ & (\tfrac{1}{2}\textstyle\sum_k^M \frac{\langle \mathbf{w}_k, \mathbf{w}_k\rangle}{\mu_k} + \frac{C}{2\sum_k^M \mu_k R_k^2}\sum_i^l \xi_i^2) && = \widetilde{F}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\mu}) \end{aligned} \tag{12}$$

The last inequality holds because in the context we examine we have $\sum_k^M \mu_k R_k^2 \leq 1$. This is a result of the fact that we work with the normalized feature spaces, using the normalized kernels as these were defined in equation 3, thus we have $R_k^2 \leq 1$ and since $\sum_k^M \mu_k = 1$ it holds that: $\sum_k^M \mu_k R_k^2 \leq 1$. Since $\widetilde{F}$ is an upper bound of F and moreover it is convex[2] we are going to use it as our objective function. As a result, we propose to solve, instead of the original soft margin optimization problem given in equation 11, the following upper bounding convex optimization problem:

[1] Note that if $\mu_k = 0$ then from the dual form we have $\mathbf{w}_k = 0$. In this case, we use the convention that $\frac{0}{0} = 0$.

[2] The convexity of this new function can be easily proved by showing that the Hessian matrix is positive semi-definite.

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\mu}} \ & \frac{1}{2}\sum_{k}^{M}\frac{\langle \mathbf{w}_k, \mathbf{w}_k\rangle}{\mu_k} + \frac{C}{2\sum_k^M \mu_k R_k^2}\sum_{i}^{l}\xi_i^2 \\ s.t. \ & y_i(\sum_{k}^{M}\langle \mathbf{w}_k, \boldsymbol{\Phi}_k(\mathbf{x}_i)\rangle + b) \geq 1-\xi_i, \\ & \sum_{k=1}^{M}\mu_k = 1, \mu_k \geq 0, \forall k \end{aligned} \tag{13}$$

The dual function of this optimization problem is:

$$\begin{aligned} W_s(\boldsymbol{\alpha},\boldsymbol{\mu}) &= -\frac{1}{2}\sum_{ij}^{l}\alpha_i\alpha_j y_i y_j \sum_{k}^{M}\mu_k K_k(\mathbf{x}_i,\mathbf{x}_j) \\ &\quad + \sum_{i}^{l}\alpha_i - \frac{\sum_k^M \mu_k R_k^2}{2C}\langle \boldsymbol{\alpha},\boldsymbol{\alpha}\rangle \\ &= -\frac{1}{2}\sum_{ij}^{l}\alpha_i\alpha_j y_i y_j (\sum_{k}^{M}\mu_k K_k(\mathbf{x}_i,\mathbf{x}_j) + \frac{\sum_k^M \mu_k R_k^2}{C}\delta_{ij}) + \sum_{i}^{l}\alpha_i \end{aligned} \tag{14}$$

where δ_{ij} is the Kronecker δ defined to be 1 if $i = j$ and 0 otherwise. The dual optimization problem is given as:

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\mu}} \ & W_s(\boldsymbol{\alpha},\boldsymbol{\mu}) \\ s.t. \ & \sum_{i}\alpha_i y_i = 0, \\ & \alpha_i \geq 0, \forall i \\ & \sum_{ij}^{l}\alpha_i\alpha_j y_i y_j K_k(\mathbf{x}_i,\mathbf{x}_j) = \frac{R_k^2 C\sum_i \xi_i^2}{(\sum_k \mu_k R_k^2)^2}, \forall k \end{aligned} \tag{15}$$

In the next section we will show how we can solve this new optimization problem.

4.2 Algorithm

The dual function 14 is quadratic with respect to $\boldsymbol{\alpha}$ and linear with respect to $\boldsymbol{\mu}$. One way to solve the optimization problem 13 is to use a two step iterative algorithm such as the ones described in [6], [10]. Following such a two step approach, in the first step we will solve a quadratic problem that optimizes over $(\mathbf{w}, b)$, while keeping $\boldsymbol{\mu}$ fixed; as a consequence the resulting dual function is a simple quadratic function of $\boldsymbol{\alpha}$ which can be optimized easily. In the second step we will solve a linear problem that optimizes over $\boldsymbol{\mu}$.

More precisely, the formulation of the optimization problem with the two-step approach takes the following form:

$$\begin{aligned} \min_{\boldsymbol{\mu}} \ & J(\boldsymbol{\mu}) \\ s.t. \ & \sum_{k=1}^{M} \mu_k = 1, \mu_k \geq 0, \forall k \end{aligned} \tag{16}$$

where

$$J(\boldsymbol{\mu}) = \begin{cases} \min_{\mathbf{w},b} \frac{1}{2} \sum_k^M \frac{\langle \mathbf{w}_k, \mathbf{w}_k \rangle}{\mu_k} + \frac{C}{2 \sum_k^M \mu_k R_k^2} \sum_i^l \xi_i^2 \\ s.t. \quad y_i (\sum_k^M \langle \mathbf{w}_k, \boldsymbol{\Phi}_k(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i \end{cases} \tag{17}$$

To solve the outer optimization problem, i.e. $\min_{\boldsymbol{\mu}} J(\boldsymbol{\mu})$, we use gradient descent method. At each iteration, we fix $\boldsymbol{\mu}$, compute the value of $J(\boldsymbol{\mu})$ and then compute the gradient of $J(\boldsymbol{\mu})$ with respect to $\boldsymbol{\mu}$. The dual function of Formula 17 is the $W_s(\boldsymbol{\alpha}, \boldsymbol{\mu})$ function already given in Formula 14. Since $\boldsymbol{\mu}$ is fixed we now optimize only over $\boldsymbol{\alpha}$ (the resulting dual optimization problem is much simpler compared to the original soft margin dual optimization problem given in Formula 15):

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \ & W_s(\boldsymbol{\alpha}, \boldsymbol{\mu}) \\ s.t. \ & \sum_i \alpha_i y_i = 0, \alpha_i \geq 0, \forall i \end{aligned}$$

which has the same form as the SVM quadratic optimization problem, the only difference is that the C parameter here is equal to $\frac{C}{\sum_k^M \mu_k R_k^2}$.

For the strong duality, at the optimal solution $\boldsymbol{\alpha}^*$, the values of dual cost function and primal cost function are equal. Thus the value of $W_s(\boldsymbol{\alpha}, \boldsymbol{\mu})$, and the $J(\boldsymbol{\mu})$ value, is given by:

$$\begin{aligned} W_s(\boldsymbol{\alpha}^*, \boldsymbol{\mu}) = -\frac{1}{2} \sum_{ij}^{l} \alpha_i^* \alpha_j^* y_i y_j (\sum_k^M \mu_k K_k(\mathbf{x}_i, \mathbf{x}_j) \\ + \frac{\sum_k^M \mu_k R_k^2}{C} \delta_{ij}) + \sum_i^l \alpha_i^* \end{aligned}$$

The last step of the algorithm is to compute the gradient of the $J(\boldsymbol{\mu})$ function, Formula 17, with respect to $\boldsymbol{\mu}$. As [6] have pointed out, we can use the theorem of Bonnans and Shapiro [15] to compute gradients of such functions. Hence, the gradient is in the following form:

$$\frac{\partial J(\boldsymbol{\mu})}{\partial \mu_k} = -\frac{1}{2} \sum_{ij}^{l} \alpha_i^* \alpha_j^* y_i y_j (K_k(\mathbf{x}_i, \mathbf{x}_j) + \frac{R_k^2}{C} \delta_{ij})$$

To compute the optimal step in the gradient descent we used line search. The complete two-step procedure is given in Algorithm 1.

Algorithm 1. R-MKL

Initialize $\mu_k^1 = \frac{1}{M}$ for $k = 1, ..., M$
repeat
  Set $R^2_{\boldsymbol{\mu}} = \sum_k^M \mu_k^t R_k^2$
  compute $J(\boldsymbol{\mu}^t)$ as the solution of a quadratic optimization problem with $K := \sum_k^M \mu_k^t K_k$
  compute $\frac{\partial J}{\partial \mu_k}$ for $k = 1, ..., M$
  compute optimal step γ_t
  $\boldsymbol{\mu}^{t+1} \leftarrow \boldsymbol{\mu}^t + \gamma_t \frac{\partial J(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}}$
until *stopCriteria* is *true*

4.3 Computational Complexity

At each step of the iteration we have to compute the solution of a standard SVM, with kernel $K = \sum_{k=1}^M \mu_k K_k$, and C equal to $\frac{C}{\sum_k^M \mu_k R_k^2}$, which is a quadratic programming problem with a complexity of $O(n^3)$ where n is number of instances. Moreover, when $\boldsymbol{\mu}$ is updated we have to recompute the approximation of $R^2_{\boldsymbol{\mu}}$; the complexity of this procedure is linear in the number of kernels, $O(M)$.

5 Experiments

We experimented with ten different datasets. Six of them were taken from the UCI repository (*Ionosphere, Liver, Sonar, Wdbc, Wpbc, Musk1*), while four come from the domain of genomics and proteomics (*ColonCancer, CentralNervousSystem, FemaleVsMale, Leukemia*) [16]; these four are characterized by small sample and high dimensionality morphology. A short description of the datasets is given in Table 1. We experimented with two different types of basic kernels, i.e. polynomials and Gaussians, and performed two sets of experiments. In the first set of experiments we used both types of kernels and in the second one we focused only on Gaussians kernels. For each set of experiments the total number of basic kernels was 20; for the first set we used polynomial kernels of degree one, two, and three and 17 Gaussians with bandwidth δ that ranged from 1 to 17 with a step of one; for the second set of experiments we only used Gaussian kernels with bandwidth δ that ranged from 1 to 20 with a step of one.

We compared our MKL algorithm (denoted as R-MKL) with two state-of-the-art MKL algorithms: Support Kernel Machine (SKM) [7], and SimpleMKL [6]. We estimate the classification error using 10-fold cross validation. For comparison purposes we also provide the performances of the best single kernel (BK) and the majority classifier (MC); the latter always predicts the majority class. The performance of the BK is that of the best single kernel estimated also by 10-fold cross validation, since it is the best result after seeing the performance of all individual kernels on the available data it is optimistically biased. We tuned the parameter C in an inner-loop 10-fold cross-validation choosing the values from the set $\{0.1, 1, 10, 100\}$. All algorithms terminate when the duality gap is smaller than 0.01. All input kernel matrices are normalized by equation 3.

Table 1. Short description of the classification datasets used

Dataset	#Inst	#Attr	#Class1	#Class2
Ionosphere	351	34	126	225
Liver	345	6	145	200
Sonar	208	60	97	111
Wdbc	569	32	357	212
Wpbc	198	34	151	47
Musk1	476	166	269	207
ColonCancer	62	2000	40	22
CentralNervous	60	7129	21	39
FemaleVSMale	134	1524	67	67
Leukemia	72	7128	25	47

We compared the significance level of the performance differences of the algorithms with McNemar's test [17], where the level of significance is set to 0.05. We also established a ranking schema of the examined MKL algorithms based on the results of the pairwise comparisons [18]. More precisely, if an algorithm is significantly better than another it is credited with one point; if there is no significant difference between two algorithms then they are credited with 0.5 points; finally, if an algorithm is significantly worse than another it is credited with zero points. Thus, if an algorithm is significantly better than all the others for a given dataset it has a score of two. We give the full results in Tables 2, 3. For each algorithm we report a triplet in which the first element is the estimated classification error, the second is the number of selected kernels, and the last is the above described rank.

Our kernel combination algorithm does remarkably well in the first set of experiments, Table 2, in which it is significantly better than both other algorithms in four datasets and significantly worse in two; for the four remaining datasets there are no significant differences. Note that in the cases of *Wpbc* and

Table 2. Results for the first experiments, where both polynomial and Gaussian kernels are used. Each triplet x,y,z gives respectively the classification error, the number of selected kernels, and the number of significance point that the algorithm scores for the given experiment set and dataset. Columns BK and MC give the errors of the best single kernel and the majority classifier, respectively.

D. Set	SKM	Simple	*R*-MKL	BK	MC
Ionos.	04.00,02,1.5	03.71,02,1.5	04.86,02,0	05.71	36.00
Liver	33.82,05,1.5	33.53,13,1.5	36.18,03,0	30.29	42.06
Sonar	15.50,01,1.0	15.50,01,1.0	15.50,01,1	17.50	46.00
Wdbc	11.25,03,1.0	13.04,18,0.0	03.75,18,2	08.57	37.32
Wpbc	23.68,17,1.0	23.68,01,1.0	23.68,01,1	23.68	23.68
Musk1	11.70,07,1.0	13.40,18,0.0	06.60,01,2	04.47	43.83
Colon.	18.33,18,1.0	18.33,18,1.0	16.67,18,1	11.67	35.00
CentNe.	35.00,17,1.0	35.00,17,1.0	35.00,17,1	31.67	35.00
Female.	33.85,20,1.0	38.92,18,0.0	20.00,18,2	22.31	60.00
Leuke.	07.14,18,0.5	07.14,18,0.5	02.86,18,2	02.86	34.29

Table 3. Results for the second set of experiments, where only Gaussian kernels are used. The table contains the same information as the previous one.

D. Set	SKM	Simple	R-MKL	BK	MC
Ionos.	04.86,02,1.0	05.43,04,1.0	05.14,03,1.0	05.14	36.00
Liver	33.53,03,1.0	33.53,20,1.0	33.53,16,1.0	34.71	42.06
Sonar	15.50,01,1.0	15.50,01,1.0	15.50,01,1.0	17.00	46.00
Wdbc	37.32,03,1.0	37.32,19,1.0	37.32,20,1.0	37.32	37.32
Wpbc	23.68,20,1.0	23.68,19,1.0	23.68,20,1.0	23.68	23.68
Musk1	43.83,14,1.0	43.83,19,1.0	43.83,20,1.0	43.83	43.83
Colon.	NA	35.00,20,1.5	35.00,20,1.5	35.00	35.00
CentNe.	35.00,20,1.0	35.00,20,1.0	35.00,20,1.0	35.00	35.00
Female.	60.00,20,1.0	60.00,20,1.0	60.00,20,1.0	60.00	60.00
Leuke.	34.29,20,1.0	34.29,20,1.0	34.29,20,1.0	34.29	34.29

CentralNervousSystem all algorithms have a performance that is similar to that of the majority classifier, i.e. the learned models do not have any discriminatory power. By examining the classification performances of the individual kernels on these datasets we see that none of them had a performance that was better than that of the majority classifier; this could explain the bad behavior of the different kernel combination schemata. Overall, for this set of experiments R-MKL gets 12 significance points over the different datasets, SKM 10.5, and SimpleMKL 7.5. The performance improvements of R-MKL over the two other methods are are quite impressive on those datasets on which R-MKL performs well; more precisely its classification error is around 30%, 50%, and 40%, of that of the other algorithms for *Wdbc*, *Musk1*, and *Leukemia* datasets respectively.

In the second set of experiments,Table 3, all methods perform very poorly in seven out of ten datasets; their classification performance is similar to that of the majority classifier. In the remaining datasets, with the exception of *ColonCancer* for which SKM failed (we used the implementation provided by the authors of the algorithm and it returns with some errors), there is no significant difference between the three algorithms. The collectively bad performance in the last seven databases is explained by the fact that none of the basic kernels had a classification error that was better than that of the majority classifier. Overall, for this set of experiments SKM scores 9 points, and Simple MKL and R-MKL score 10.5 points each.

Comparing the number of selected kernels by the different kernel combination methods using a paired t-test (significance level of 0.05) revealed no statistically significant differences between the three algorithms on both sets of experiments.

In an effort to get an empirical estimation of the quality of the approximation of the radius that we used to make the optimization problems convex, we computed the approximation error defined as $\frac{\sum_k^M \mu_k R_k^2 - R_{\boldsymbol{\mu}}^2}{R_{\boldsymbol{\mu}}^2}$. We computed this error over the different folds of the ten-fold cross-validation for each dataset. The average approximation error over the different datasets was 0.0056. We also computed this error over 1000 random values of $\boldsymbol{\mu}$ for each dataset and the

average error was 0.0104. Thus, the empirical evidence seems to indicate that the $R^2_{\boldsymbol{\mu}} \leq \sum_k^M \mu_k R_k^2$ bound is relatively tight, at least for the datasets we examined.

6 Conclusion and Future Work

In this paper we presented a new kernel combination method that incorporates in its cost function not only the margin but also the radius of the smallest sphere that encloses the data. This idea is a direct implementation of well known error bounds from statistical learning theory. To the best of our knowledge this is the first work in which the radius is used together with the margin in an effort to minimize the generalization error. Even though the resulting optimization problems were non-convex and we had to use an upper bound on the radius to get convex forms, the empirical results were quite encouraging. In particular, our method competed with other state-of-the-art methods for kernel combination, thus demonstrating the benefit and the potential of the proposed technique. Finally, we mention that it is still a challenging research direction to fully exploit the examined generalization bound.

In future work we would like to examine optimization techniques for directly solving the non-convex optimization problem presented in Formula 11. In particular, we will examine whether it is possible to decompose the cost function as a sum convex and concave functions, or to represent it as *d.m functions* (difference of two monotonic functions) [19,20]. Additionally, we plan to analyze the bound $R^2_{\boldsymbol{\mu}} \leq \sum_k^M \mu_k R_k^2$ and see how it relates with the real optimal value.

Acknowledgments. The work reported in this paper was partially funded by the European Commission through EU projects DropTop (FP6-037739), DebugIT (FP7-217139) and e-LICO (FP7-231519). The support of the Swiss NSF (Grant 200021-122283/1) is also gratefully acknowledged.

References

1. Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge University Press, Cambridge (2004)
2. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
3. Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L.E.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research 5, 27–72 (2004)
4. Ong, C.S., Smola, A.J., Williamson, R.C.: Learning the kernel with hyperkernels. Journal of Machine Learning Research 6, 1043–1071 (2005)
5. Sonnenburg, S., Ratsch, G., Schafer, C.: A general and efficient multiple kernel learning algorithm. Journal of Machine Learning Research 7, 1531–1565 (2006)
6. Bach, F., Rakotomamonjy, A., Canu, S., Grandvalet, Y.: SimpleMKL. Journal of Machine Learning Research (2008)

7. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the smo algorithm. In: ICML 2004: Proceedings of the twenty-first international conference on Machine learning, p. 6. ACM, New York (2004)
8. Lanckriet, G., Bie, T.D., Cristianini, N.: A statistical framework for genomic data fusion. Bioinformatics 20 (2004)
9. Cristianini, N., Shawe-Taylor, J., Elisseeff, A.: On kernel-target alignment. Journal of Machine Learning Research (2002)
10. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing multiple parameters for support vector machines. Machine Learning 46(1-3), 131–159 (2002)
11. Crammer, K., Keshet, J., Singer, Y.: Kernel design using boosting. In: Advances in Neural Information Processing Systems, vol. 14. MIT Press, Cambridge (2002)
12. Bousquet, O., Herrmann, D.: On the complexity of learning the kernel matrix. In: Advances in Neural Information Processing Systems, vol. 14. MIT Press, Cambridge (2003)
13. Cristianini, N., Shawe-Taylor, J.: An introduction to Support Vector Machines. Cambridge University Press, Cambridge (2000)
14. Vapnik, V.: Statistical learning theory. Wiley Interscience, Hoboken (1998)
15. Bonnans, J., Shapiro, A.: Optimization problems with perturbation: A guided tour. SIAM Review 40(2), 202–227 (1998)
16. Kalousis, A., Prados, J., Hilario, M.: Stability of feature selection algorithms: a study on high dimensional spaces. Knowledge and Information Systems 12(1), 95–116 (2007)
17. McNemar, Q.: Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika 12, 153–157 (1947)
18. Kalousis, A., Theoharis, T.: Noemon: Design, implementation and performance results for an intelligent assistant for classifier selection. Intelligent Data Analysis Journal 3, 319–337 (1999)
19. Leo Liberti, N.M. (ed.): Global Optimization - From Theory to Implementation. Springer, Heidelberg (2006)
20. Collobert, R., Weston, J., Bottou, L.: Trading convexity for scalability. In: Proceedings of the 23th Conference on Machine Learning (2006)
21. Stephen Boyd, L.V. (ed.): Convex optimization. Cambridge University Press, Cambridge (2004)

Appendix

Proof of Inequality 7. If $K(\mathbf{x}, \mathbf{x}')$ is the kernel function associated with the $\mathbf{\Phi}(\mathbf{x})$ mapping then the computation of the radius in the dual form is given in [1]:

$$\max_{\beta_i \beta_j} \; R^2 = \sum_i^l \beta_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{ij}^l \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{18}$$

$$s.t. \; \sum_i^l \beta_i = 1, \beta_i \geq 0$$

If $\boldsymbol{\beta}^*$ is the optimal solution of (18) when $K = K_{\boldsymbol{\mu}} = \sum_1^M \mu_k K_k$, and $\hat{\boldsymbol{\beta}^{\boldsymbol{k}}}$ is the optimal solution of (18) when $K = K_k$, i.e. :

$$R^2_{\boldsymbol{\mu}} = \sum_{k=1}^{M} \mu_k (\sum_{i=1}^{l} \beta^*_i K_k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{l} \beta^*_i \beta^*_j K_k(\mathbf{x}_i, \mathbf{x}_j))$$
$$R^2_k = \sum_{i=1}^{l} \hat{\beta^k}_i K_k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{l} \hat{\beta^k}_i \hat{\beta^k}_j K_k(\mathbf{x}_i, \mathbf{x}_j)$$

then

$$\sum_{i=1}^{l} \beta^*_i K_k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{l} \beta^*_i \beta^*_j K_k(\mathbf{x}_i, \mathbf{x}_j) \leq$$
$$\sum_{i=1}^{l} \hat{\beta^k}_i K_k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{l} \hat{\beta^k}_i \hat{\beta^k}_j K_k(\mathbf{x}_i, \mathbf{x}_j)$$

Therefore: $R^2_{\boldsymbol{\mu}} \leq \sum_{k=1}^{M} \mu_k R^2_k$

Proof of convexity of R-MKL (Eq.13). To prove that 13 is convex, it is enough to show that functions $\frac{x^2}{\mu}$, where $x \in \mathbb{R}$, $\mu \in \mathbb{R}^+$, and $\frac{\xi^2}{\sum_k^M \alpha_k \mu_k}$, where $\xi \in \mathbb{R}, \mu_k, \alpha_k \in \mathbb{R}^+$ are convex. The first is quadratic-over-linear function which is convex. The second is convex because its epigraph is a convex set [21].