Efficient multiple scale kernel classifiers

Rocco Langone
KU Leuven, Department of Electrical Engineering
Kasteelpark Arenberg 10, B-3001
Leuven, Belgium
Email: rocco.langone@esat.kuleuven.be

Johan A. K. Suykens
KU Leuven, Department of Electrical Engineering
Kasteelpark Arenberg 10, B-3001
Leuven, Belgium
Email: johan.suykens@esat.kuleuven.be

***Abstract*—While kernel methods using a single Gaussian kernel have proven to be very successful for nonlinear classification, in case of learning problems with a more complex underlying structure it is often desirable to use a linear combination of kernels with different widths. To address this issue, this paper presents a classification algorithm based on a jointly convex constrained optimization formulation. The primal problem is defined as jointly learning a combination of kernel classification models formulated in different feature spaces, which account for various representations or scales. The solution can be found by either solving a system of linear equations in case of equal combination weights or by means of a block coordinate descent scheme. The dual model is represented by a classifier using multiple kernels in the decision function. Furthermore, time and space complexity are reduced by adopting a divide and conquer strategy and through the use of the Nyström approximation of the eigenfunctions. Several experiments show the effectiveness of the proposed algorithms in dealing with datasets containing up to millions of instances.**

***Keywords*-Multiple Kernel Learning; Nyström approximation; Divide and Conquer; Map-Reduce.**

I. Introduction

Research in multiple kernel learning (MKL) [1] has shown how the combination of several base kernels can lead to improvements in prediction accuracy w.r.t. a single classifier. In particular, the convex combination of Gaussian kernels with different widths allows to better tackle learning problems involving multiple representations. More precisely, sparse combination coefficients can be used to understand which representations are important for discrimination [2]. On the other hand, non-sparse combination weights allow for robust kernel mixtures with better generalization capabilities but at the expense of a lower interpretability [3].

In this paper we introduce the efficient Multiple-Scale Kernel Classification algorithm (MSKC), which is based on solving a jointly convex constrained optimization problem. The primal formulation is designed as a joint learning of a combination of kernel classifiers in different feature spaces accounting for various scales of description [4]. The dual representation of the model is nothing but a multiple kernel classification decision function. We propose two frameworks: weighted and unweighted algorithm. In the first framework the single classifiers are given the same weights, and the related optimization problem can be tackled by simply solving a linear system. In the second framework, an alternating convex optimization scheme [5] is adopted in order to optimize w.r.t. to the model parameters and the combination weights. For large-scale datasets the Nyström method is exploited to compute finite dimensional approximation of the feature maps, such that the computational complexity depends mainly on the size of the subset used for the approximation [6]. Moreover, a divide and conquer[1] strategy implemented through a Map-Reduce scheme allows to further increase the speed-up without significantly decreasing the accuracy. In a nutshell, the core contribution of this article is represented by a multiple kernel learning algorithm able to process millions of instances at a desktop PC scale in a few hundred seconds. Even the most efficient methods such as SMO-MKL [7], L_p-MKL [8], [3], [9] and Weight Update MKL [10], are not able to achieve this performance.

The rest of this article is organized as follows. Section 1.1 summarizes some related work. Section 2 gives a short introduction on the base learners used in the proposed multiple-scale classification algorithms, the Nyström method and the divide and conquer strategy. In Section 3 our formulation is introduced. The optimization procedures are detailed in Section 4, and experimental results are presented in Section 5. Finally, Section 6 concludes the article.

A. Related work

As shown in [1], the most commonly used base learners in MKL algorithms are by far support vector machine (SVM). Other methods include Kernel Fisher discriminant analysis (KFDA), regularized kernel discriminant analysis (RKDA), and kernel ridge regression (KRR). Regarding KRR, up to our knowledge there exist three main MKL approaches: multiple additive regression kernels (MARK) [11], L2 Regularization for Learning Kernels [12] and learning nonlinear combination of kernels [13]. All these methods are the most closely related to the proposed algorithms but differ from them in different aspects. The MARK algorithm is inspired by boosting methods and uses column generation

[1]Proofs of the upper bounds associated to the mean-squared error of the averaged model parameters estimates will be the subject of future work.

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

to construct the heterogeneous kernel matrix. In [12] an interpolated iterative algorithm is devised. The authors of [13] study the problem of learning nonlinear combination of base kernels (instead of linear combinations), and propose a projection-based gradient descent algorithm for solving the MKL optimization problem. Another related method is the ensemble support vector machine (ESVM) algorithm [14]. Like in the proposed techniques, ESVM employs a divide-and-conquer strategy by aggregating many SVM models trained on bootstrap subsamples of the training set in order to reduce the total runtime. The predictions of the base models are then aggregated through majority voting.

II. Preliminaries

A. Base kernel classifier

The base learner used in the proposed algorithms relates to kernel ridge[2] regression (KRR) [15], least squares SVM [16] and kriging estimator [17]. Let $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$, with $\mathbf{x}_i \in \mathbb{R}^d$, denote the set of training examples and $\mathbf{y} = [y_1, \ldots, y_N]^T$, with $y_i \in \{-1, 1\}$, the corresponding vector of training set labels, and let $\varphi(\mathbf{x}_i)$ indicate the feature vector associated to $\mathbf{x}_i$, being $\varphi : \mathbb{R}^d \to \mathbb{R}^{d_h}$ the feature map. Then, the primal optimization problem of the base kernel classifier is:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}||\mathbf{w}||_2^2 + \frac{C}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\varphi(\mathbf{x}_i) - b)^2, \tag{1}$$

where $C > 0$ is a trade-off parameter. After reformulating problem (1) as a constrained optimization problem [16], the related (Lagrange) dual solution is given by:

$$\left[\begin{array}{c|c} \mathbf{K} + \frac{\mathbf{I}}{C} & \mathbf{1}_N \\ \hline \mathbf{1}_N^T & 0 \end{array}\right]\left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \mathbf{y} \\ \hline 0 \end{array}\right] \tag{2}$$

where $\mathbf{I}$ indicates the $N \times N$ identity matrix, $\mathbf{1}_N = [1, \ldots, 1]^T$ denotes a N dimensional vector of ones, $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the kernel matrix associated to the positive definite kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, i.e. $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$, and $\boldsymbol{\alpha} \in \mathbb{R}^N$ is the vector of dual variables. Finally, the estimated class label in the dual for data item $\mathbf{x}_j$ can be computed as:

$$\hat{y}_j = \text{sign}(\sum_{i=1}^{N}\alpha_i k(\mathbf{x}_i, \mathbf{x}_j) + b). \tag{3}$$

B. Nyström method

The Nyström method has been shown to be a valuable solution to scale-up kernel-based algorithms including SVM, Gaussian processes, kernel principal component analysis [18]. When N is large, even constructing and storing the kernel matrix can be prohibitive. In this case, replacing $\mathbf{K}$ with its low-rank approximation $\tilde{\mathbf{K}}$ can greatly reduce the computational cost of computing the dual solution. Furthermore, approximate explicit feature maps corresponding to certain kernels can serve as a basis for reducing the cost of learning nonlinear classification models with large datasets, as shown in [19] and in case of the fixed-size method proposed in [16]. Given a positive definite kernel $k(\mathbf{x}_i, \mathbf{x}_j) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and a data density distribution $p(\mathbf{x})$, the Nyström approximation of order N is the feature map $\tilde{\varphi} : \mathbb{R}^d \to \mathbb{R}^N$ that best approximate the kernel at points $\mathbf{x}_i$ and $\mathbf{x}_j$ sampled from $p(\mathbf{x})$. In particular, the components $\tilde{\varphi}_i(\mathbf{x})$ are eigenfunctions of the kernel, i.e. the solutions of the Fredholm integral equation. In practice the eigenfunction equation is approximated by replacing the integral with an empirical average to obtain [6]:

$$\tilde{\varphi}_i(\mathbf{x}) = \frac{\sqrt{N}}{\lambda_i}\sum_{j=1}^{N}\mathbf{u}_{ji}k(\mathbf{x}_j, \mathbf{x}) \tag{4}$$

where λ_i and $\mathbf{u}_i$ are the eigenvalues and eigenvectors of the kernel matrix $\mathbf{K}$. Finally, instead of computing the eigendecomposition of the original $N \times N$ kernel matrix, their Nyström approximations $\tilde{\lambda}_i$ and $\tilde{\mathbf{u}}_i$ can be calculated based on the eigensystem of a submatrix $\mathbf{K}_{m\times m}$ as:

$$\tilde{\varphi}_i(\mathbf{x}) \approx \sqrt{\frac{m}{\tilde{\lambda}_i}}\sum_{j=1}^{m}\tilde{\mathbf{u}}_{ji}k(\mathbf{x}_j, \mathbf{x}) \tag{5}$$

with $\tilde{\lambda}_i = \frac{N}{m}\lambda_i^{(m)}$, $\tilde{\mathbf{u}}_i = \sqrt{\frac{m}{N}}\frac{1}{\lambda_i^{(m)}}\mathbf{K}_{N\times m}\mathbf{u}_i^{(m)}$ and $\tilde{\mathbf{K}} = \mathbf{K}_{N\times m}\mathbf{K}_{m\times m}^{-1}\mathbf{K}_{m\times N}$. Throughout this paper eq. (5) will be used to solve the primal optimization problems.

C. Divide and Conquer

The divide and conquer strategy randomly partitions a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ of size N into n subsets of equal size $\{\mathcal{S}_r\}_{r=1}^n \subset \mathcal{D}$, computes in parallel an independent estimator for each subset, then averages the local solutions into a global predictor. Divide and conquer approaches have been developed for various applications including bootstrap [20], parametric smooth convex optimization problems [21], kernel ridge regression [22], support vector machine [14].

III. Proposed MKL formulation

The proposed MKL approach can be stated as follows:

$$\begin{aligned} &\min_{\mathbf{w}^l,b^l,c^l} J(\mathbf{w}^l, b^l, c^l) = \frac{1}{2}\sum_{l=1}^{q}||\mathbf{w}^l||_2^2 + \frac{C}{2}||\sum_{l=1}^{q}c^l(\boldsymbol{\Phi}^l\mathbf{w}^l + b^l\mathbf{1}_N) - \mathbf{y}||_2^2 \\ &\text{s. t.} \quad c^l \geq 0, \quad l = 1, \ldots, q \\ &\sum_{l=1}^{q}c^l = 1 \end{aligned} \tag{6}$$

where $\boldsymbol{\Phi}^l = [\varphi^l(\mathbf{x}_1)^T; \ldots; \varphi^l(\mathbf{x}_N)^T]$ denotes the l-th $N \times d_h$ feature matrix, and the combination weights c^l are restricted to lie in the probability simplex to avoid large numerical values and enhance interpretability. By assuming that the training data can have different scales of description and to

[2]The primal KRR optimization problem and kriging are defined by setting $b = 0$ in eq. (1).

enhance the diversity among the individual classifiers, let the different feature maps be associated to the Gaussian kernels with bandwidths $\sigma^1, \ldots, \sigma^q$, i.e. let us set $k^l(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma^{l^2}})$. Furthermore, to simplify the derivations, we assume that $b^1 = b^2 = \ldots = b^q = b$.

A. Framework 1: average of base classifiers

Problem (6) can be simplified by taking the average of the individual classifiers. In this case the combination weights are no longer optimization variables:

$$\min_{\mathbf{w}^l, b} F_1(\mathbf{w}^l, b) = \frac{1}{2}\sum_{l=1}^{q} ||\mathbf{w}^l||_2^2 + \frac{C}{2}||\frac{1}{q}\sum_{l=1}^{q} \tilde{\mathbf{\Phi}}^l \mathbf{w}^l + b\mathbf{1}_N - \mathbf{y}||_2^2. \tag{7}$$

In eq. (7) we make use of (5) and replace each feature map φ^l with its finite dimensional approximation $\tilde{\varphi}^l$. The solution to problem (7) can be found by solving a linear system of size $(m+1)q \times (m+1)q$, being m the sample size used for the Nyström approximation. More precisely, given approximate feature maps $\tilde{\varphi}^l : \mathbb{R}^d \rightarrow \mathbb{R}^m$ with $\tilde{\varphi}^l(\mathbf{x}_i) = [k^l(\mathbf{x}_i, \mathbf{x}_1); \ldots; k^l(\mathbf{x}_i, \mathbf{x}_m)]$, regularization constant $C \in \mathbb{R}^+$, the necessary and sufficient condition for optimality is $\nabla F_1(\mathbf{w}^l, \hat{\mathbf{y}}^l, b) = 0$ which implies that the solution satisfies the following linear system:

$$\begin{bmatrix} c^1\tilde{\mathbf{\Phi}}_{\text{ext}}^{1^T}\tilde{\mathbf{\Phi}}^1 + \frac{\mathbf{I}_0}{C} & \ldots & \ldots & c^q\tilde{\mathbf{\Phi}}_{\text{ext}}^{1^T}\tilde{\mathbf{\Phi}}^q & \tilde{\mathbf{\Phi}}_{\text{ext}}^{1^T}\mathbf{1}_N \\ \vdots & \ddots & & \vdots & \vdots \\ c^1\tilde{\mathbf{\Phi}}_{\text{ext}}^{q^T}\tilde{\mathbf{\Phi}}^1 & \ldots & c^q\tilde{\mathbf{\Phi}}_{\text{ext}}^{q^T}\tilde{\mathbf{\Phi}}^q + \frac{\mathbf{I}_0}{C} & \tilde{\mathbf{\Phi}}_{\text{ext}}^{q^T}\mathbf{1}_N \end{bmatrix} \begin{bmatrix} \mathbf{w}^1 \\ \vdots \\ \mathbf{w}^q \\ b \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{\Phi}}_{\text{ext}}^{1^T}\mathbf{y} \\ \vdots \\ \tilde{\mathbf{\Phi}}_{\text{ext}}^{q^T}\mathbf{y} \end{bmatrix} \tag{8}$$

where $\tilde{\mathbf{\Phi}}_{\text{ext}}^{l^T} = [c^l\tilde{\mathbf{\Phi}}^{l^T}; \mathbf{1}_N^T]$ and $\mathbf{I}_0 = [\mathbf{I}; \mathbf{0}_{1\times m}]$. Taking the derivatives $\frac{\partial F_1}{\partial \mathbf{w}^l} = 0$ and $\frac{\partial F_1}{\partial b} = 0$ and combining the resulting equations leads to (8) after some algebraic manipulations of the variables.

The vector with the predicted training class labels is:

$$\hat{\mathbf{y}} = \text{sign}(\sum_{l=1}^{q} c^l\tilde{\mathbf{\Phi}}^l\mathbf{w}^l + b\mathbf{1}_N). \tag{9}$$

Similarly, the predicted class labels for the N_{test} datapoints $\{\mathbf{z}_j\}_{j=1}^{N_{\text{test}}}$ are computed as:

$$\hat{\mathbf{y}}_{\text{test}} = \text{sign}(\sum_{l=1}^{q} c^l\tilde{\mathbf{\Phi}}_{\text{test}}^l\mathbf{w}^l + b\mathbf{1}_N), \tag{10}$$

where $\tilde{\mathbf{\Phi}}_{\text{test}}^l = [\tilde{\varphi}^l(\mathbf{z}_1)^T; \ldots; \tilde{\varphi}^l(\mathbf{z}_{N_{\text{test}}})^T]$ and $\tilde{\varphi}^l(\mathbf{z}_j) = [k^l(\mathbf{z}_j, \mathbf{x}_1); \ldots; k^l(\mathbf{z}_j, \mathbf{x}_m)]$.

B. Framework 2: joint learning of base classifiers and combination weights

Objective (6) can be seen as an optimization problem with respect to the parameters vector $\boldsymbol{\omega} = [\mathbf{w}^1; \ldots; \mathbf{w}^q; b]$ and the vector of combination weights $\mathbf{c} = [c^1; \ldots; c^q]$. Although not jointly convex in ($\boldsymbol{\omega}$,$\mathbf{c}$), it is convex w.r.t. each unknown when the other one is fixed. Therefore block coordinate descent, although not necessarily providing the global optimum, can be used with the guarantee to perform reasonably well [23], [24]. Block coordinate descent consists of iterating between fixing $\mathbf{c}$ and optimizing w.r.t. $\boldsymbol{\omega}$ and fixing $\boldsymbol{\omega}$ and optimizing w.r.t. $\mathbf{c}$. In this last case the following quadratic programming (QP) problem arises:

$$\begin{aligned} \min_{c^l} \quad & F_2(c^l) = \frac{1}{2}\sum_{l=1}^{q} ||\mathbf{w}^l||_2^2 + \frac{C}{2}||\sum_{l=1}^{q} c^l\tilde{\mathbf{\Phi}}^l\mathbf{w}^l + b\mathbf{1}_N - \mathbf{y}||_2^2 \\ \text{s. t.} \quad & c^l \geq 0, \quad l = 1, \ldots, q \\ & \sum_{l=1}^{q} c^l = 1. \end{aligned} \tag{11}$$

IV. Optimization schemes

A generic illustration of the proposed approaches[3] is given in Figure 1. The first framework, which we call MSKC1, encompasses three main steps. First, the N samples $\{\mathbf{x}_i, y_i\}_{i=1}^N$ are randomly partitioned into n subsets of equal size $\mathcal{X}_1, \ldots, \mathcal{X}_n$. Then, for each partition, the approximate feature maps for every scale $\tilde{\varphi}_i^1(\mathbf{x}), \ldots, \tilde{\varphi}_i^q(\mathbf{x})$ are computed. Third, n independent estimates of the model parameters, that is $\mathbf{w}_1^1, \ldots, \mathbf{w}_n^q, b_1, \ldots, b_n$, are computed by solving (8), and the vector of predicted class labels is obtained by means of eq. (9), where $\mathbf{w}^l, b$ are replaced by the averaged estimates $\bar{\mathbf{w}}^l = \frac{1}{n}\sum_{r=1}^{n} \mathbf{w}_r^l$ and $\bar{b} = \frac{1}{n}\sum_{r=1}^{n} b_r$. In case of framework 2, which we call MSKC2, the optimal combination weights $\{c^l\}_{l=1}^q$ are estimated as $\bar{c}^l = \frac{1}{n}\sum_{r=1}^{n} c_r^l$. The related alternate optimization procedure is described in algorithm 1, and in Figure 2 the convergence of c^l and the objective function $J(\mathbf{w}^l, b^l, c^l)$ are shown in case of the *Adult* dataset.

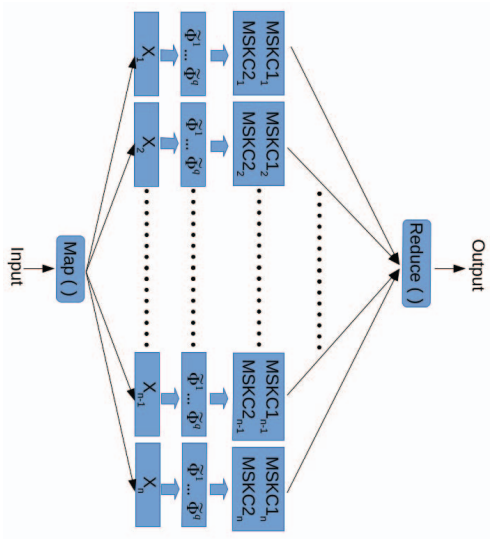


Figure 1. **Architecture of the proposed frameworks.** Schematic visualization of the Map-Reduce implementation.

[3]The related Matlab package can be downloaded from: *http://www.esat.kuleuven.be/stadius/ADB/langone/MSKClab.php*

Algorithm 1: MSKC2

Input : training set $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$, test set $\mathcal{D}_{\text{test}} = \{\mathbf{z}_j\}_{j=1}^{N_{\text{test}}}$.

Settings : size Nyström subset m, scales $\sigma^1, \ldots, \sigma^q$, convergence threshold THR_{stop}, regularization parameter C, number of partitions n for divide and conquer, maximum number of iterations max_iter.

Output : $\hat{\mathbf{y}}$ (vector of estimated training class labels), $\hat{\mathbf{y}}_{\text{test}}$ (vector of predicted test class labels), combination weights $c^1, \ldots, c^q$.

```
/* Compute SVD of kernel submatrices:                          */
for l=1:q do
    Compute K^l_{m×m}
    Compute [U,Λ] = SVD(K^l_{m×m})
end
Map ( )
    /* Divide and conquer:                                     */
    For each partition {S_r}_{r=1}^n do in parallel
        /* Compute approximate feature maps:                   */
        for l=1:q do
            Compute Φ̃^l by means of (5)
        end
        /* Start alternate optimization:                       */
        Initialization : t=1,c^1=...=c^q=1/q.
        while |J_t − J_{t−1}| < THR_stop do
            Solve (8)
            Solve (11)
            Compute J_t(w^l, b^l, c^l)
            t+=1
            if t > max_iter then
                break
            end
        end
    end
end
Reduce ( )
    w̄^l = (1/n) Σ_{r=1}^n w_r^l   b̄ = (1/n) Σ_{r=1}^n b_r   c̄^l = (1/n) Σ_{r=1}^n c_r^l
end
ŷ ← Compute (9), ŷ_test ← Compute (10)
```

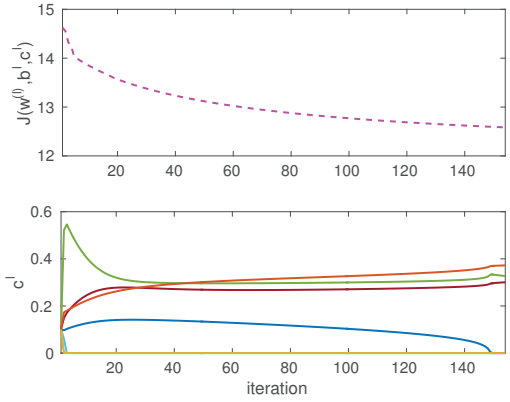


Figure 2. **Illustrative example**. Convergence of the objective function $J(\mathbf{w}^l, b^l, c^l)$ **(top)** and evolution of the combination weights $\{c^l\}_{l=1}^{10}$ **(bottom)** in case of the Adult dataset.

A. Map-Reduce implementation

By a practical point of view, the Divide and Conquer strategy used to learn efficiently the parameters $\mathbf{w}^l, b$ and c^l (in case of the MSKC2 method) is implemented through a Map-Reduce scheme. In particular, in the Map-Reduce framework the `Map()` operation includes (i) partitioning the data in n bootstrap samples, (ii) computing approximated feature maps, calculating per sample estimates of the parameters i.e. $\mathbf{w}_r^l, b^l$ and c^l. In the `Reduce()` step the individual bootstrap estimates are summed-up to obtain the final model parameters. All routines have been implemented in Matlab. More precisely, the `parfor` command is used to execute in parallel the outer for loop in algorithm 1 and the reduce operation needed to fold the results into a single output.

B. Computational complexity

The computational complexity of the proposed algorithms depends on (i) the size m of the Nyström subset used to construct the approximate feature maps $\tilde{\mathbf{\Phi}}^l$ (ii) the number n of partitions in the divide and conquer scheme. Indeed, by assuming that the number q of relevant scales is much lower than m, in case of the MSKC1 method the time complexity, denoted by t_1, is approximately $O(m^3/n^3) + O(mN/n) + O(mN_{\text{test}}/n)$, which is the time needed to solve (8) and to compute (9) and (10). The space complexity, denoted by m_1, is $O(m^2/n^2) + O(mN/n) + O(mN_{\text{test}}/n)$, which is needed to construct (8) and to build the training and test feature matrices $\tilde{\mathbf{\Phi}}^l$ and $\tilde{\mathbf{\Phi}}^l_{\text{test}}$. Concerning the MSKC2 approach, since we assume that $q \ll m$, the time needed to solve the QP problem (11) is negligible. Therefore, the time complexity is approximately Tt_1 and the space complexity is Tm_1, where T denotes the number of iterations performed by the block coordinate descent algorithm.

V. Experiments

In this section we discuss the simulation results. All the experiments have been executed on a 8 cores desktop machine running the Centos operating system.

A. Performance assessment on small-scale datasets

Our focus in these experiments is purely on analyzing the prediction accuracy of the proposed methods. Following the settings of previous MKL studies [25], the performance of the MSKC1 and MSKC2 algorithms is evaluated over a number of UCI datasets [26] and compared with the following state-of-the-art methods: convex multiple kernel learning (MKL) [25], MKL with L_p norm regularization (L_pMKL) [8], [3], [9], generalized MKL (GMKL) [27], and infinite kernel learning (IKL) [28]. For each data set 50% of all instances are used as training data and the remaining 50% as test data, and the mean result of 20 runs is reported.

The parameters of the MSKC1 and MSKC2 approaches are chosen as follows: the regularization constant C is determined by 5-fold cross-validation on the training data over the range $\{10^{-2}, \ldots, 10^2\}$, 10 bandwidths parameters in the range $\{2^{-3}, \ldots, 2^6\}$ are adopted as scales. Similar settings have been used in case of the regularization parameter of the other methods. The Nyström subset size m is chosen as 10% of the training data size, which was empirically found to be a good choice in case of datasets containing up to 5000 instances. Furthermore, for small-scale datasets the divide and conquer strategy is not needed, i.e. $n = 1$.

Table I

PERFORMANCE EVALUATION ON SMALL-SCALE DATASETS. COMPARISON OF THE PROPOSED ALGORITHMS AGAINST (I) VARIOUS STATE-OF-THE-ART MKL TECHNIQUES (II) THE BASE KERNEL CLASSIFIER WITH A SINGLE GAUSSIAN KERNEL WHOSE C AND σ ARE SELECTED VIA 5-FOLD CROSS-VALIDATION, REFERRED AS BKC. BEST PERFORMANCE IN BOLD.

Dataset	N_{tot}	d	MSKC1	MSKC2	MKL [25]	L_pMKL [3], [8], [9]	GMKL [27]	IKL [28]	BKC
Adult	1605	123	81.6 ± 0.9	$\mathbf{83.0 \pm 0.9}$	81.5 ± 1.0	82.1 ± 0.6	75.5 ± 1.1	75.1 ± 1.1	82.0 ± 0.9
Australian	690	14	85.5 ± 1.6	$\mathbf{86.3 \pm 2.1}$	85.0 ± 1.5	84.5 ± 1.6	80.0 ± 2.3	85.4 ± 1.2	86.2 ± 1.4
Breast	683	9	96.6 ± 0.3	96.5 ± 0.5	96.5 ± 0.8	96.2 ± 0.7	$\mathbf{97.0 \pm 1.0}$	96.5 ± 0.7	96.6 ± 0.8
Diabetes	768	8	$\mathbf{76.7 \pm 1.7}$	75.9 ± 2.1	75.8 ± 2.5	72.6 ± 2.5	66.4 ± 2.5	76.0 ± 3.0	75.4 ± 0.3
German	1000	24	75.2 ± 2.1	$\mathbf{76.0 \pm 1.5}$	71.4 ± 2.8	74.3 ± 1.4	70.4 ± 1.6	70.0 ± 1.5	75.6 ± 1.6
Heart	270	13	82.4 ± 2.1	82.1 ± 3.1	83.0 ± 2.9	76.7 ± 3.8	77.0 ± 3.6	$\mathbf{83.3 \pm 2.1}$	81.1 ± 3.1
Ionosphere	351	33	83.0 ± 2.4	82.1 ± 2.6	91.7 ± 1.9	92.6 ± 1.4	92.7 ± 1.8	$\mathbf{93.7 \pm 1.0}$	79.7 ± 3.7
Liver	345	6	68.8 ± 1.8	$\mathbf{71.3 \pm 2.0}$	62.3 ± 4.5	69.4 ± 2.9	63.6 ± 2.6	60.0 ± 2.9	69.6 ± 1.5
Splice	1000	60	82.8 ± 2.5	81.6 ± 1.4	88.4 ± 2.4	87.1 ± 1.6	$\mathbf{92.4 \pm 1.4}$	80.0 ± 1.5	76.1 ± 5.6
Titanic	712	9	77.5 ± 1.1	$\mathbf{78.3 \pm 1.7}$	77.4 ± 2.8	77.3 ± 3.1	76.9 ± 3.0	77.1 ± 2.6	77.4 ± 2.0

Table I illustrates the results of average test classification accuracy. The proposed algorithms achieve higher accuracy than the other MKL methods on 6 out of 10 datasets, where MSKC2 is superior to MSKC1, although at the expense of a higher computational cost due to the alternate optimization scheme. Furthermore, it can be noticed how some MKL algorithms do not always outperform the best classifier whose kernel parameter is tuned by cross-validation. Although it seems surprising, a similar observation has already been documented in some previous studies [30].

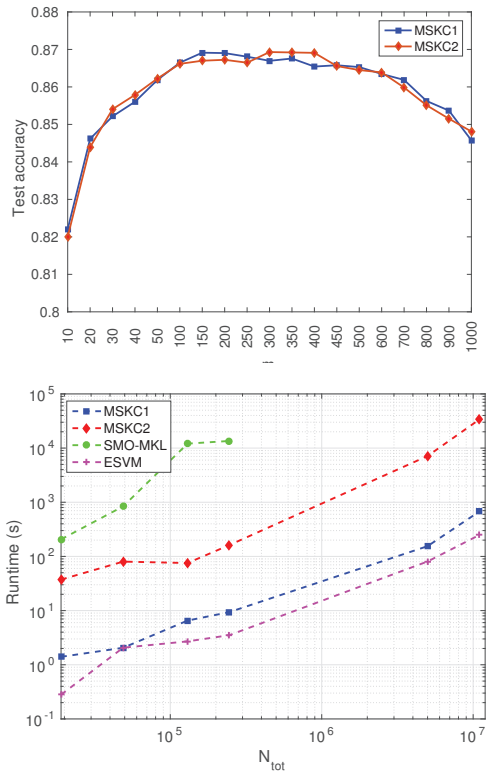


Figure 3. **Scalability analysis on large-scale datasets**. **(Top)** Average test accuracy as function of the Nyström subset size m for the Magic dataset **(Bottom)** Runtime (train + test) of MSKC1, MSKC2, SMO-MKL and ESVM related to the datasets described in Table II.

B. Scalability analysis on large-scale datasets

Our focus in these experiments is mainly on studying the computational efficiency of the proposed methods in the analysis of large-scale data containing several thousands or millions of instances. In this case the baselines for comparison are (i) the SMO-MKL algorithm (with L_2 norm) [7], which up to our knowledge is the most efficient MKL technique together with L_p-MKL (ii) the ensemble SVM (ESVM) algorithm [14], which uses a divide and conquer strategy to speed-up SVM training time.

The UCI datasets reported in Table II are used for the analysis. We randomly selected 80% of the points as training set and 20% as test set.The tuning parameters of all the methods are selected by 5-fold cross-validation. The following user-defined settings have been used for MSKC1 and MSKC2: $\text{THR}_{\text{stop}} = 10^{-3}$, $m = 300$, max_iter = 50 and $n = 20$. These values have been empirically found to represent a good trade-off between test accuracy and runtime in the analysis of large datasets. Indeed, the test accuracy of MSKC1 and MSKC2 algorithms has been observed to not increase significantly when $m \geq 300$, as shown at the top of Figure 3 in case of the Magic dataset. Even worse, at a certain point the test performance starts to decrease as the Nyström subset size m increases. A possible reason is the higher chance of overfitting occurring with high dimensional data, as reported also in [31].

Figure 3 (bottom) illustrates the scalability of the proposed algorithms in the analysis of large datasets. MSKC1 and MSKC2 are much faster than SMO-MKL because the (on-the-fly) construction of the kernel matrix is avoided through the use of the Nyström method and thanks to the divide and conquer strategy. Indeed, SMO-MKL has been designed more with the aim of efficiently handling several thousands or millions of kernels rather than processing a large number of instances. Moreover, the efficiency of MSKC1 is comparable to that of ESVM.

VI. CONCLUSION

We have presented two MKL algorithms for jointly learning a linear combination of kernel classifiers formulated

Table II

LARGE-SCALE DATASETS PERFORMANCE. AVERAGE TEST ACCURACY OF THE PROPOSED ALGORITHMS, SMO-MKL, ESVM AND THE BASE KERNEL CLASSIFIER WITH A SINGLE GAUSSIAN KERNEL WHOSE C AND σ ARE SELECTED VIA 5-FOLD CROSS-VALIDATION, REFERRED AS BKC. NA MEANS EXECUTION TIME > 72 HOURS.

Dataset	N_{tot}	d	MSKC1	MSKC2	SMO-MKL [7]	ESVM [14]	BKC
Magic	19 020	10	86.39	86.65	85.53	85.25	85.89
AdultBig	48 842	123	84.90	84.78	83.37	84.28	84.76
Miniboone	130 064	50	91.36	92.36	88.19	87.25	90.60
Skin	245 057	3	99.86	99.86	99.18	99.28	99.63
Susy	5 000 000	18	79.61	79.74	NA	78.70	79.45
Higgs	11 000 000	28	65.94	66.83	NA	65.37	65.72

in different feature spaces, which are related to various scales of description of the input data. In both algorithms a divide and conquer strategy together with the Nyström approximation is used to make the methods scale to datasets with millions of datapoints. Experiments on several datasets illustrate that the proposed techniques are accurate and efficient in processing large volumes of data instances. Future work will be devoted to extend the proposed frameworks to the regression setting and unsupervised and semi-supervised learning problems and to study the convergence properties of the divide and conquer scheme.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923).

REFERENCES

[1] M. Gönen and E. Alpaydin, "Multiple kernel learning algorithms," *JMLR*, vol. 12, pp. 2211–2268, 2011.

[2] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *JMLR*, vol. 7, pp. 1531–1565, 2006.

[3] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "Lp-norm multiple kernel learning," *JMLR*, vol. 12, pp. 953–997, 2011.

[4] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE TPAMI*, vol. 35, no. 8, pp. 1798–1828, 2013.

[5] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," *Neural, Parallel Scientific Computation*, vol. 11, no. 4, pp. 351–368, 2003.

[6] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *NIPS*, 2001, pp. 682–688.

[7] Z. Sun, N. Ampornpunt, M. Varma, and S. V. N. Vishwanathan, "Multiple kernel learning and the smo algorithm," in *NIPS*, 2010, pp. 2361–2369.

[8] M. Kloft, U. Brefeld, P. Laskov, K.-R. Müller, A. Zien, and S. Sonnenburg, "Efficient and accurate lp-norm multiple kernel learning," in *NIPS*, 2009, pp. 997–1005.

[9] S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. d. Bona, A. Binder, C. Gehl, and V. Franc, "The shogun machine learning toolbox," *JMLR*, vol. 11, pp. 1799–1802, 2010.

[10] J. Moeller, P. Raman, S. Venkatasubramanian, and A. Saha, "A geometric algorithm for scalable multiple kernel learning," in *AISTATS*, 2014, pp. 633–642.

[11] K. P. Bennett, M. Momma, and M. J. Embrechts, "Mark: A boosting algorithm for heterogeneous kernel models," in *KDD*, 2002, pp. 24–31.

[12] C. Cortes, M. Mohri, and A. Rostamizadeh, "L2 regularization for learning kernels," in *UAI*, 2009, pp. 109–116.

[13] ——, "Learning non-linear combinations of kernels," in *NIPS*, 2009, pp. 396–404.

[14] M. Claesen, F. D. Smet, J. A. Suykens, and B. D. Moor, "EnsembleSVM: A library for ensemble learning using support vector machines," *JMLR*, vol. 15, pp. 141–145, 2014.

[15] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *ICML*, 1998, pp. 515–521.

[16] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.

[17] N. Cressie, "The origins of kriging," *Mathematical Geology*, vol. 22, no. 3, pp. 239–252, 1990.

[18] S. Sun, J. Zhao, and J. Zhu, "A review of Nyström methods for large-scale machine learning," *Information Fusion*, vol. 26, no. C, pp. 36–48, 2015.

[19] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE TPAMI*, vol. 34, no. 3, pp. 480–492, 2012.

[20] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan, "The big data bootstrap," in *ICML*, 2012, pp. 1759–1766.

[21] Y. Zhang, J. Duchi, and M. Wainwright, "Communication-efficient algorithms for statistical optimization," *JMLR*, vol. 14, no. 1, pp. 3321–3363, 2013.

[22] ——, "Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates," *JMLR*, vol. 16, pp. 3299–3340, 2015.

[23] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion." *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.

[24] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *NIPS*, 2009, pp. 1033–1040.

[25] Z. Xu, R. Jin, I. King, and M. R. Lyu, "An extended level method for efficient multiple kernel learning," in *NIPS*, 2009, pp. 1825–1832.

[26] A. J. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[27] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *ICML*, 2009, pp. 1065–1072.

[28] P. Gehler and S. Nowozin, "Infinite kernel learning," in *NIPS Workshop on "Kernel Learning: Automatic Selection of Optimal Kernels"*, 2008, pp. 1–4.

[29] M. Li, J. T. Kwok, and B. Lu, "Making large-scale Nyström approximation possible," in *ICML*, 2010, pp. 631–638.

[30] J. Zhuang, I. W. Tsang, and S. C. H. Hoi, "Two-layer multiple kernel learning." in *AISTATS*, 2011, pp. 909–917.

[31] Y. Lee and S. Huang, "Reduced support vector machines: A statistical theory," *IEEE TNN*, vol. 18, no. 1, pp. 1–13, 2007.