Neural Comput & Applic
DOI 10.1007/s00521-015-2066-x

Deep multilayer multiple kernel learning

Ilyes Rebai[1] · **Yassine BenAyed**[1] · **Walid Mahdi**[2]

Received: 4 March 2015 / Accepted: 8 September 2015

Abstract Multiple kernel learning (MKL) approach has been proposed for kernel methods and has shown high performance for solving some real-world applications. It consists on learning the optimal kernel from one layer of multiple predefined kernels. Unfortunately, this approach is not rich enough to solve relatively complex problems. With the emergence and the success of the deep learning concept, multilayer of multiple kernel learning (MLMKL) methods were inspired by the idea of deep architecture. They are introduced in order to improve the conventional MKL methods. Such architectures tend to learn deep kernel machines by exploring the combinations of multiple kernels in a multilayer structure. However, existing MLMKL methods often have trouble with the optimization of the network for two or more layers. Additionally, they do not always outperform the simplest method of combining multiple kernels (i.e., MKL). In order to improve the effectiveness of MKL approaches, we introduce, in this paper, a novel backpropagation MLMKL framework. Specifically, we propose to optimize the network over an adaptive backpropagation algorithm. We use the gradient ascent method instead of dual objective function, or the estimation of the leave-one-out error. We test our proposed method through a large set of experiments on a variety of benchmark data sets. We have successfully optimized the system over many layers. Empirical results over an extensive set of experiments show that our algorithm achieves high performance compared to the traditional MKL approach and existing MLMKL methods.

Keywords Deep learning · Support vector machine · Multilayer multiple kernel learning · Optimization methods · Gradient ascent

1 Introduction

Kernel learning is an active research topic in machine learning and is broadly studied [21]. Support vector machine (SVM) [9, 10] and kernel principal component analysis (KPCA) [18] are the most popular learning algorithms based on kernel methods. These kernel methods have been successfully applied to a variety of real-world applications, and they often demonstrated promising performance. However, their generalization performances are often controlled by the choice of the kernel [26, 27, 29]. Therefore, many works have been proposed for learning the optimal kernel for these methods [1, 12, 27].

Multiple kernel learning (MKL) was proposed to address the limitations of regular kernel methods (i.e., based on a single fixed kernel). Bach et al. [1] introduced the first MKL formulation. In the last decade, it became the most popular technique for kernel learning. It aims at learning a linear (or convex) combination of a set of predefined kernels in order to determine the best target kernel for the application. Over the last years, MKL has been widely studied and has shown good performance for automated kernel parameter tuning compared to

✉ Ilyes Rebai
rebai_ilyes@hotmail.fr

Yassine BenAyed
yassine.benayed@gmail.com

Walid Mahdi
walid.mahdi@isimsf.rnu.tn

[1] MIRACL: Multimedia InfoRmation System and Advanced Computing Laboratory, University of Sfax, Sfax, Tunisia

[2] College of Computers and Information Technology, Taif University, Taif, Saudi Arabia

Published online: 18 September 2015

conventional kernel methods. Several algorithms for extended MKL techniques have been proposed to improve the efficiency of the regular MKL method [16, 19, 22, 25, 28, 30]. Despite their interesting results, existing MKL methods do not always produce considerably better empirical performance when compared with the simplest method based on the average of base kernels, in some practical applications.

Recently, it was shown that deep architectures [3, 6, 15, 17] are very promising alternatives to the shallow models. Multilayer of multiple kernel learning (MLMKL), a new area of kernel methods, was proposed in order to apply the idea of deep learning in order to improve the MKL task. The major research direction for MLMKL methods focuses on the development of efficient learning algorithms, which improve the performance of the system. In [5], the authors proposed a new type of kernel that mimics the deep learning architecture. The drawback of this method is that the obtained network is static. In fact, the authors used fixed kernels without learning the optimal combination of kernels. Zhuang et al. [29] described a general framework for multilayer of MKL. However, they had problems when optimizing the network beyond two layers. Note that the second layer only consisted of a single radial basis function (RBF) kernel. In [23], the authors successfully optimized an MLMKL with many layers with an estimate of the leave-one-out error algorithm. Unfortunately, they did not evaluate their method over MKL ones. Furthermore, no impressive improvements were obtained using more than two layers.

In this work, we present a backpropagation MLMKL framework. Our study is partially inspired by the recent works [23, 29] that explored kernel methods with the idea of deep learning. All base kernels in antecedent layers are linearly combined so as to form new inputs to the base kernels in the subsequent layers. Our study mainly aims to address the challenge of improving the optimization method of the previous works. Specifically, we propose to optimize the network over an adaptive backpropagation algorithm. We use the gradient ascent method instead of the dual objective function, or the estimation of the leave-one-out error. Our choice of the backpropagation algorithm with the gradient method is justified by the fact that they have been widely used with many learning techniques and have shown high performance [6, 11, 26]. This method is very simple from a computational perspective and very strong from a mathematical perspective. It enables to train deep networks, such as deep neural network. As the backpropagation algorithm allows to construct deep networks, we describe, in the scope of this paper, a new technique for training multiple kernels in multilayer architecture with backpropagation algorithm in order to learn the optimal combination of the kernels.

This paper is organized as follows: First, we give more details for the related work, and we show how multiple kernels and deep multiple kernels can be constructed from other kernels in Sect. 2. In Sect. 3, we introduce the neural network architecture and backpropagation algorithm. We give some preliminaries of multiple kernel learning and deep learning in Sect. 4. Section 5 exhibits some characteristics of the proposed framework and the proposed backpropagation algorithm. Experimental setup and results on a large number of data sets are presented in Sects. 6 and 7. The final Sect. 8 summarizes this paper and gives the perspective of the present work.

2 Related work

MKL consists of combining a set of already defined base kernels so as to learn the optimal kernel. It exhibits its strength of learning multiple kernels and its capability of combining heterogeneous data [29]. The basic structure of MKL is shown in Fig. 1.

During the last decade, MKL has been actively investigated, in which lots of algorithms and techniques were proposed in order to improve the efficiency of the MKL method [8, 13, 19, 22, 24, 27, 28, 30]. In [1], the authors solved MKL network using the sequential minimization optimization (SMO) algorithm, while in [22], the authors used semi-infinite linear programming (SILP) method for solving large-scale MKL problems. The experimental results have shown that these methods often achieve worse prediction results than the simple average of the base kernels [7, 14]. Although MKL approach has been widely studied, its architecture is still "shallow," since it consists of a simple (linear/convex) combination of multiple kernels. Indeed, the optimal kernel is not rich enough to solve complicated applications.

To address such limitation, some emerging works have applied the idea of deep learning in order to improve the MKL methods [4, 5, 23, 29]. Since the deep architecture

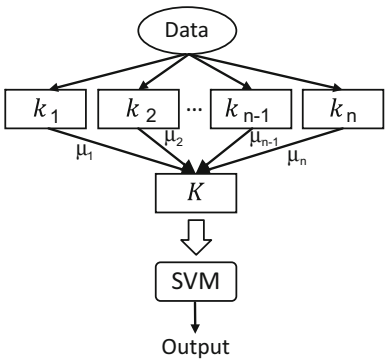

Fig. 1 Multiple kernel method architecture

shows high performance over the shallow one, several studies extended the single layer of multiple kernels by learning multilayers of multiple kernels, and this framework is usually called MLMKL. MLMKL represents a new active research area for kernel methods.

The first kernel method using deep architecture was introduced by Cho and Saul [5]. Their main idea consists on iteratively repeating the inputs mapping L times. A new family of kernel functions, called arccosine kernels, was proposed; it imitates the computational process of the deep neural network. Specifically, L layer's architecture is the inner product after L feature mapping kernels of the input data:

$$K^{(L)}(x,y) = \Phi^{(L)}\left(\ldots\Phi^{(1)}(x)\right)\cdot\Phi^{(L)}\left(\ldots\Phi^{(1)}(y)\right) \quad (1)$$

where x and y are the input vectors, $\Phi^{(L)}$ is a feature mapping function applied L times, and $K^{(L)}$ represents the final layer kernel. This method produces considerably better experimental performance when compared with MKL methods. However, the use of some specific functions (arccosine kernels) may not give a rich representation of data for more complicated tasks. In addition, the proposed multilayer kernel has a static architecture.

Recently, few algorithms have been proposed to improve the efficiency of the existing MLMKL methods. The new methods tend to learn a deep architecture by exploring the combinations of multiple weighted kernels in a multilayer structure. The main goal is to learn, simultaneously, the optimal kernel combinations (i.e., combination of weights) and the decision function. Unlike the shallow MKL (see Fig. 1), the base kernels, in each layer, are combined and used as input to the base kernels in the immediate layer. Zhuang et al. [29] propose to optimize the system based on the dual objective function. The authors were able to optimize an MLMKL of only two layers, where the second layer is composed of a single RBF kernel. Strobl and Visweswaran [23] optimized multiple layers of kernels over an estimation of the leave-one-out error method. Although the empirical results show an improvement comparing with the optimization over the dual objective function, their results are still not as good as a single kernel-based SVM. Furthermore, using more than two layers does not improve the performance of the proposed method.

3 Deep neural network

The artificial neural network design is inspired from human biological neurons. The neural networks are very powerful computational models that can solve complex problems such as nonlinear regression problems (e.g., XOR). With the fast development of hardware and software, it became possible to use neural networks with complex architecture (multiple hidden layers) and to train the network with massive data. A deep neural network is more powerful in expressing complex functions than a neural network based on a single hidden layer. Deep learning achieved high performances in a wide range of tasks, including speech processing [3, 17] and computer vision [6, 15].

The feedforward neural network is the first and simplest type of neural network. The numerical information is transmitted in only one direction, forward, from node to node. The network does not include cycles or loops. It consists of a chain of function compositions which transforms an input to an output vector [20]. Each node, known as *the artificial neuron*, receives the input values of the antecedent layer neurons and produces an output:

$$O^l = f\left(\sum_i w_i o_i^{l-1} + \theta\right) \quad (2)$$

where θ is a random value, o_i is the node output value in the layer $l-1$, weighted by w_i, and $f(\cdot)$ represents the nonlinear transfer function (Fig. 2). The learning problem consists of finding the optimal combination of weights. The popular learning method is the *backpropagation algorithm*, which is the most studied and used algorithms for neural network learning. The backpropagation algorithm consists on searching the global minimum of the error function (Eq. 3) in the weight space using the gradient descent. The method of gradient descent corrects the initial random weights. The final solution is the set of weights that minimizes the error function [20].

Given a training set of n examples $\{(x_1,t_1),(x_2,t_2),\ldots,(x_n,t_n)\}$ with $x_i \in \mathbb{R}^q$ (input vectors) and $t_i \in \mathbb{R}^p$ (output vectors), the network produces from the input vector x_i an output o_i. The learning algorithm goal is to generate o_i close to the target t_i for $i = 1,\ldots,n$ by minimizing the error function of the network:

$$E = \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{p}\|o_{ij} - t_{ij}\|^2 \quad (3)$$

where o_{ij} and t_{ij} represent the jth elements of the output vector o_i and of the target t_i, respectively, and E represents the sum of the quadratic errors of all input vectors (i.e., the total error for a given training set). Then, the m weights

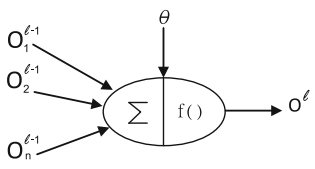

Fig. 2 Representation of an artificial neuron

$w_1, w_2, \ldots, w_m$ in the network are iteratively updated using the gradient descent. For each weight, the gradient is computed, and then, the weight is updated using gradient descent, defined as follows:

$$w_i = w_i - \eta \frac{\partial E}{\partial w_i} \tag{4}$$

where η represents a learning constant, known as the learning rate. This process is expected to minimize the function E.

4 Multiple kernel learning

Given a training set of n samples $\{(x_i, y_i)|i = 1, \ldots, n\}$, where $x_i \in \mathbb{R}^q$ is the input vector and $y_i \in \{-1, 1\}$ is the class label of x_i, each input x_i is mapped to a high-dimensional reproducing kernel Hilbert space using a kernel function $k(x_i, x_j) = \phi(x_i)' \cdot \phi(x_j)$. Subsequently, SVM learns the hyperplane that has the large margin and the small error on the training set. The hyperplane is defined as:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x, x_i) + b \tag{5}$$

where α_i is a coefficient associated with the training sample, and b is a constant. The optimization problem is:

$$\begin{aligned} &\min_{(\alpha,b,\xi)} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \\ &\text{s.t. } y_i(\alpha_i k(x, x_i) + b) \geq (1 - \xi_i), \\ &\qquad \xi_i \geq 0, C > 0, \quad i = 1, \ldots, n \end{aligned} \tag{6}$$

where ξ_i is the slack variable, i.e., the training errors, and C is a regularization parameter. In order to have multiple kernels instead of a single kernel, Bach et al. [1] introduce the first formulation of MKL by using a set of M base kernels $k_1, \ldots, k_M$ with the following optimization problem:

$$\begin{aligned} &\min_{(\alpha,b,\xi)} \frac{1}{2}\left(\sum_{k=1}^{M} \|w\|_2\right)^2 + C\sum_{i=1}^{n} \xi_i \\ &\text{s.t. } y_i\left(\sum_{k=1}^{M} \mu_k \sum_{i=1}^{n} y_i \alpha_i K_k(x, x_i) + b\right) \geq (1 - \xi_i), \\ &\qquad 0 < \alpha < C, \\ &\qquad \xi_i \geq 0, \quad i = 1, \ldots, n, \\ &\qquad \mu_k \geq 0, \quad k = 1, \ldots, M \end{aligned} \tag{7}$$

where μ_k is the coefficient associated with the base kernel $K_k(x, x_i)$. In general, the resultant kernel K is chosen to be a convex combination of the predefined base kernels:

$$K_{\text{conv}} = \left\{ K(x, x_i) = \sum_{k=1}^{M} \mu_k K_k(x, x_i) \middle| \sum_{k=1}^{M} \mu_k = 1, \quad \mu_k > 0 \quad k = 1, \ldots, M \right\} \tag{8}$$

where K_{conv} represents a combination of the M base kernels. Thus, the decision function for the above formulations is defined as:

$$f(x) = \sum_{k=1}^{M} \mu_k \sum_{i=1}^{n} \alpha_i y_i K_k(x, x_i) + b \tag{9}$$

Zhuang et al. [29] expand the decision function in (9) for two layers of multiple kernels method (2LMKL). They define the 2LMK domain as follows:

$$K^{(2)} = \left\{ k^{(2)}(x_i, x_j; \mu) = \exp\left(\sum_{k=1}^{M} \mu_k k_k^{(1)}(x_i, x_j)\right) \middle| \mu \in \mathbb{R}^m, \quad \mu_t > 0 \right\} \tag{10}$$

This formulation (10) can be replaced by the equivalent min–max optimization, as follows:

$$\begin{aligned} &\min_{\mu} \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k^{(2)}(x_i, x_j; \mu) + \sum_{k=1}^{M} \mu_k \\ &\text{s.t. } 0 < \alpha < C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad k = 1, \ldots, M \end{aligned} \tag{11}$$

where K^2 is a kernel function applied to the resultant kernel from the first layer. The final decision function of the two-layer MKL is:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i k^2(x, x_i; \mu) + b \tag{12}$$

5 Deep multilayer multiple kernel with an adaptive backpropagation algorithm

The aim of the present work is to improve the efficiency of the existing MLMKL methods. Indeed, we propose a different optimization algorithms from the state of the art in order to enhance the classification performance of these methods.

We implement an MLMKL framework. As in [23, 29], all base kernels in antecedent layers are combined so as to form new inputs to the base kernels in subsequent layers (as shown in Fig. 3). We define the resultant kernel K of the multilayer multiple kernel domain as follows:

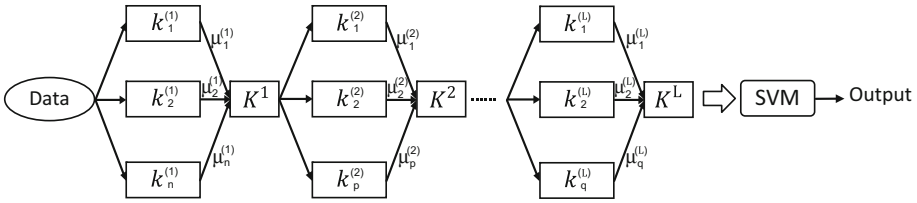

Fig. 3 Structure of multiple layer multiple kernel framework: In the regular SVM, a kernel function is applied to transform the input data, and then, the SVM is trained using this new input data space to learn the final decision (i.e., classification task). Instead of using a single kernel function, a set of functions, organized in a specific structure, map the original data through several layers of kernels, and then, the final kernel is used to learn the decision function of SVM

$$K^{(l)} = \left\{ K^{(l)}\left(K^{(l-1)}; \mu^{(l)}\right) = \sum_{k=1}^{M} \mu_k^{(l)} k_k^{(l)}\left(K^{(l-1)}\right) \middle| \mu_k^{(l)} \geq 0, \quad k = 1, \ldots, M, \quad l = 1, \ldots, L \right\}, \qquad (13)$$
$$K^{(1)} = K^{(1)}\left(x, x_i; \mu^{(1)}\right) = \sum_{k=1}^{M} \mu_k k_k(x, x_i)$$

where $k_k^{(l)}$ is the kth base kernel at layer l and μ_k^l denotes the weight of the kth kernel at layer l. The final decision function of the proposed framework is defined as:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K^{(L)}\left(K^{(L-1)}; \mu^{(L)}\right) + b \qquad (14)$$

In order to obtain the coefficients μ that minimize the true risk of the decision function, we propose to apply the backpropagation algorithm and the gradient ascent. Therefore, the error function (Eq. 3) is adapted for the optimization process. Our goal is to minimize the error function, defined as:

$$E = \frac{1}{2n} \sum_{j=1}^{n} \|f(x_j) - t_j\|^2 \qquad (15)$$

where $f(x_j)$ is the posterior probability of vector x_j. From this formulation, we can thus minimize the error function E by using an iterative process of gradient ascent. First, we compute the gradient of each weight in each layer:

$$\nabla E = \left\{ \underbrace{\frac{\partial E}{\partial \mu_1^{(1)}}, \ldots, \frac{\partial E}{\partial \mu_M^{(1)}}}_{\text{Layer 1}}; \ldots; \underbrace{\frac{\partial E}{\partial \mu_1^{(L)}}, \ldots, \frac{\partial E}{\partial \mu_M^{(L)}}}_{\text{Layer } L} \right\} \qquad (16)$$

Then, all the network weights are updated using the gradient ascent:

$$\mu = \mu + \eta \nabla E, \quad \mu_k^{(l)} \geq 0, \quad k = 1, \ldots, M, \quad \text{and} \quad l = 1, \ldots, L \qquad (17)$$

where $\mu = \{ \underbrace{\mu_1^{(1)}, \ldots, \mu_M^{(1)}}_{\text{Weights in Layer1}}; \ldots; \underbrace{\mu_1^{(L)}, \ldots, \mu_M^{(L)}}_{\text{Weights in Layer}L} \}$ and η is the learning rate. Thus, we simply compute the gradient, w.r.t. μ:

$$\frac{\partial E}{\partial \mu_k^{(l)}} = \frac{\partial\left(\frac{1}{2n}\sum_{j=1}^{n} \|f(x_j) - t_j\|^2\right)}{\partial \mu_k^{(l)}} = \frac{1}{n} \sum_{j=1}^{n} \|f(x_j) - t_j\| \frac{\partial f(x_j)}{\partial \mu_k^{(l)}} \qquad (18)$$

$\frac{\partial f(x_j)}{\partial \mu_k^{(l)}}$ is computed for each weight in each layer. For instance, using a two-layer multiple kernel architecture and for each weight k at the second layer, $\frac{\partial f(x_j)}{\partial \mu_k^{(L)}}$ is equal to:

$$\frac{\partial f(x_j)}{\partial \mu_k^{(2)}} = \sum_{j=1}^{n} \alpha_i y_i \frac{\partial\left(K^{(2)}\left(K^{(1)}; \mu^{(2)}\right)\right)}{\partial \mu_k^{(2)}} = \sum_{i=1}^{n} \alpha_i y_i \frac{\partial\left(\sum_{k=1}^{M} \mu_k^{(2)}\ k_k^{(2)}\left(K^{(1)}\right)\right)}{\partial \mu_k^{(2)}} = \sum_{i=1}^{n} \alpha_i y_i\ k_k^{(2)}(\cdot) \qquad (19)$$

Next, for each weight k at the first layer, $\frac{\partial f(x_j)}{\partial \mu_k^{(1)}}$ is defined as:

$$\frac{\partial f(x_j)}{\partial \mu_k^{(1)}} = \sum_{i=1}^{n} \alpha_i y_i \frac{\partial\left(K^{(2)}\left(K^{(1)}; \mu^{(2)}\right)\right)}{\partial \mu_k^{(1)}} = \sum_{i=1}^{n} \alpha_i y_i \frac{\partial\left(\sum_{k=1}^{M} \mu_k^{(2)}\ k_k^{(2)}\left(K^{(1)}\right)\right)}{\partial \mu_k^{(1)}} = \sum_{i=1}^{n} \alpha_i y_i\ k_k^{(1)}(x_j, x_i) \sum_{k=1}^{M} \mu_k^{(2)} \frac{\partial k_k^{(2)}(\cdot)}{\partial (x_j, x_i)} \qquad (20)$$

We can compute $\frac{\partial f(x_j)}{\partial \mu_k^{(l)}}$ for an architecture with many number of layers ($L > 2$). We can now develop an alternating optimization algorithm to learn the decision function and all the network weights simultaneously. This can be done as follows: (1) fix α and solve μ and (2) fix μ and solve α. Algorithm 1 gives the detailed optimization steps of the proposed MLMKL algorithm for a given weight vector μ.

Algorithm 1: Multi-Layer Multiple Kernel Learning Algorithm: (μ, α)=ML-MKL(D, μ^1, L, η)

Input: Training set $D = \{(x_i, y_i)|i = 1, ..., n\}$, μ_k^1 for every kernel k, number of layers L, and $\eta \in [0..1]$

Output: The learned weight vector μ, and SVM coefficient vector α.

```
for i ← 1 to maxIterations do
    Compute the kernel K^(L) using μ^1;
    Solve the SVM problem with the resultant kernel K^(L);
    Compute ∇E (Eq. 16) by using the equation 18;
    Update the vector of weights: μ^(i+1) = μ^i + η∇E;
    if stopping criterion then
        Break;
```

6 Databases and experimental setup

We conducted an extended set of experiments in order to evaluate the performance of the proposed MLMKL algorithm for classification tasks. Several algorithms were tested on 12 real-world data sets using different methods:[1,2] Australian, Indian liver patient, Liver, Monk3, Postoperative, Pima Indian Diabetes, Sonar, Tic-tac-toe, German, Credit, Breast cancer, Musk1. Table 1 gives a detailed description of the used data sets.

Table 1 The 12 publicly available data sets used in the experiments from UCI and Keel data set repositories

Data set	# Dimensions	# Training set	# Test set
Liver	6	173	172
Monk3	6	216	216
Postoperative	8	43	44
Pima Indian diabetes	8	384	384
Tic-tac-toe	9	479	479
Indian liver patient	10	290	289
Breast cancer	10	342	341
Australian	14	345	345
Credit	15	327	327
German	24	500	500
Sonar	60	104	104
Musk1	166	238	238

6.1 Algorithm details

In this subsection, we present an exhaustive comparative study on the following algorithms:

- *SKSVM* The SVM algorithm with a single RBF kernel.
- *L_2MKL* The proposed MKL algorithm by Kloft et al. [14].
- *SM1MKL* The proposed MKL algorithm by Xu et al. [27]. They developed a soft margin MKL framework.
- *2LMKL* The proposed two-layer MKL algorithm by Zhuang et al. [29].
- *DMKL* The proposed MLMKL algorithm by Strobl and Visweswaran [23]. The base kernel coefficients are optimized over a tight upper bound of the leave-one-out error.
- *MLMKL* Our proposed MLMKL algorithm described in Algorithm 1.

The SKSVM and the proposed MLMKL are implemented in Matlab, and the LIBSVM [2] package is used to solve the SVM optimization problem. For SM1MKL, we downloaded the MATLAB codes provided by Xu from the Web.[3] The L_2MKL implementation is obtained from Xu toolbox. For DMKL, we downloaded their MATLAB codes from the Web.[4] For 2LMKL, the implementation is obtained from [23, 29].

In the following experiments, we fixed the regularization parameter of SVM to ten for all used algorithms. In addition, for 2LMKL, DMKL and MLMKL, another parameter η is defined and is set in the range of $\{1, 10^{-1}, 10^{-2}, \ldots, 10^{-6}, 10^{-7}, 10^{-8}\}$. One more parameter L is introduced for DMKL and MLMKL, and it is, in this paper, in the range of $\{1, 2, 3\}$. Finally, we initialize all the weights $\mu_k^{(l)}$ to $\frac{1}{M}$ for all MKL and MKML algorithms.

6.2 Experimental settings

Usually, RBF and polynomial kernels are the most commonly used functions for the multiple kernel learning methods. In our work, we fixed four base kernels, in the following manner:

- Linear kernel.
- RBF kernel with $\gamma = 1$.
- Two polynomial kernels with two different degrees $d = 2$, $d = 3$ and $\beta = 1$.

For each data set, first, we excluded the instances having missing values. Next, we randomized the samples and divided the data in half: (1) 50 % of all instances were used for training (to built a deep model with the optimal parameters), (2) the remaining 50 % were used as test data

[1] Available at: https://archive.ics.uci.edu/ml/datasets.html.

[2] Available at: http://sci2s.ugr.es/keel/category.php?cat=clas.

[3] Available at: https://sites.google.com/site/xinxingxu666/.

[4] Available at: https://github.com/ericstrobl/deepMKL.

Table 2 The evaluation of classification performance MKL and MLMKL algorithms

Data set	Algorithms				
	SKSVM	L_2MKL	SM1MKL	DMKL	MLMKL
Liver	63.66 ± 3.86	67.50 ± 4.61 (4)	68.83 ± 3.91 (3)	69.01 ± 4.12 (2)	**71.80 ± 2.24 (1)**
Monk3	90.32 ± 1.81	**97.22 ± 1.19 (1)**	96.66 ± 0.89 (3)	96.62 ± 0.72 (4)	96.89 ± 0.79 (2)
Postoperative	67.04 ± 4.70	55.91 ± 6.35 (4)	59.31 ± 6.89 (2)	56.36 ± 5.84 (3)	**59.77 ± 8.23 (1)**
Pima Indian diabetes	67.42 ± 2.90	71.84 ± 1.83 (4)	72.81 ± 1.45 (3)	73.12 ± 1.65 (2)	**75.70 ± 1.38 (1)**
Tic-tac-toe	66.38 ± 2.28	88.62 ± 1.36 (3)	87.72 ± 1.28 (4)	**91.67 ± 1.32 (1)**	91.48 ± 1.44 (2)
Indian liver patient	69.37 ± 1.86	68.44 ± 2.01 (4)	68.75 ± 1.59 (3)	69.10 ± 1.76 (2)	**69.65 ± 1.06 (1)**
Breast cancer	94.25 ± 0.60	96.18 ± 1.26 (4)	96.36 ± 1.13 (3)	96.59 ± 0.94 (2)	**97.21 ± 0.99 (1)**
Australian	71.54 ± 3.01	81.64 ± 1.44 (4)	82.89 ± 1.25 (3)	84.40 ± 1.28 (2)	**85.42 ± 0.91 (1)**
Credit	71.56 ± 5.41	81.71 ± 2.14 (4)	83.27 ± 2.07 (3)	84.74 ± 1.89 (2)	**85.68 ± 1.31 (1)**
German	70.18 ± 0.92	69.98 ± 1.75 (4)	70.14 ± 1.32 (3)	72.02 ± 1.41 (2)	**75.06 ± 1.47 (1)**
Sonar	50.29 ± 2.79	84.51 ± 2.50 (2)	**85.00 ± 3.66 (1)**	83.94 ± 2.75 (3)	83.84 ± 2.74 (4)
Musk1	57.35 ± 2.46	91.00 ± 1.80 (4)	**91.42 ± 1.59 (1)**	91.13 ± 1.72 (3)	91.17 ± 1.77 (2)
Average rank		3.5	2.66	2.33	1.5

Numbers in the parenthesis represent the relative rank of the algorithms on each data set. Each element in the table shows the mean and standard deviation of the classification accuracy (%). The last row presents the average rank score over all data sets achieved by each algorithm

The values highlighted in bold refer to the system that has the rank 1 for each dataset

(to evaluate the performance of the obtained model). The training samples were normalized so as to have zero mean and unit variance, while the test samples were normalized using the same mean and variance of the training data. Finally, all the algorithms were trained and tested using the same training and test sets. In order to get reliable results, we followed the same experiment protocol used in [27, 29]. Indeed, for each data set, we ran all the algorithms ten times, in which the samples are randomized again, divided in half and normalized. We provided the mean classification accuracy and the standard deviation. As it is known, the accuracy is the ratio of correctly classified samples to all samples:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{N} \tag{21}$$

where TP is the number of true positive, TN is the number of true negative, and N is the number of instances in the test set.

7 Experimental results

We have done two sets of experiments. The first one is to compared the MKL and MLMKL methods in order to show that the MLMKL approach outperforms the traditional MKL algorithms. The second set of experiments is done to test the proposed algorithm with the existing MLMKL methods.

7.1 MKL versus MLMKL methods

The aim of these experiments is to evaluate the performance of the MKL algorithms with our algorithm based on MLMKL approach and DMKL. Hence, we evaluated the following algorithms: SKSVM, L_2MKL, SM1MKL, DMKL and MLMKL. For DMKL and MLMKL, we tested three-layer architecture in order to analyze and to determine the effectiveness of using multiple layers structure. Table 2 shows the detailed results of the classification of the different algorithms. Furthermore, we identified the ranks of the algorithms according to their performance on each data set. The average rank is presented in the last row of Table 2.

By comparing the results between SVM and the two existing MKL methods (L_2MKL and SM1MKL), we found that these algorithms do not guarantee high performance over SVM with an RBF kernel. For instance, the SM1MKL algorithm was surpassed by SVM over three data sets: postoperative, Indian liver patient and German.

In terms of the rank, the mean rank of MLMKL over all data sets is equal to 1.5, DMKL rank is 2.33 while the mean rank value of SM1MKL is 2.66 and L_2MKL is equal to 3.5. So, MLMKL achieves the best performance, whereas L_2MKL is the worst. Thus, these results show that MLMKL and DMKL algorithms outperform MKL methods. In general, the obtained results demonstrate that using a deep architecture improves the performance of the shallow methods, i.e., MKL.

Table 3 The evaluation of classification performance [mean classification accuracy (%) ± SD] by comparing with the selected methods

Data set	Algorithms							
	SKSVM	DMKL			2LMKL	MLMKL		
		1-Layer	2-Layer	3-Layer	2-Layer	1-Layer	2-Layer	3-Layer
Liver	63.66 ± 3.86	69.53 ± 2.99 (3)	69.53 ± 4.14 (3)	69.01 ± 4.12 (6)	63.43 ± 2.62 (7)	69.53 ± 4.15 (3)	70.17 ± 3.13 (2)	**71.80 ± 2.24 (1)**
Monk3	90.32 ± 1.81	96.25 ± 2.25 (7)	96.52 ± 0.85 (5)	96.62 ± 0.72 (3)	96.26 ± 1.29 (6)	**96.89 ± 1.00 (1)**	96.62 ± 0.82 (3)	**96.89 ± 0.79 (1)**
Postoperative	67.04 ± 4.70	54.09 ± 8.07 (6)	55.00 ± 6.75 (5)	56.36 ± 5.84 (3)	**60.45 ± 7.66 (1)**	54.09 ± 7.17(6)	56.36 ± 6.77 (3)	59.77 ± 8.23 (2)
Pima Indian	67.42 ± 2.90	73.02 ± 1.84 (6)	73.04 ± 1.13 (5)	73.12 ± 1.65 (4)	69.58 ± 1.74 (7)	73.98 ± 1.26 (3)	75.18 ± 1.46 (2)	**75.70 ± 1.38 (1)**
Tic-tac-toe	66.38 ± 2.28	88.02 ± 7.18 (6)	91.41 ± 1.60 (4)	91.67 ± 1.32 (2)	**91.79 ± 1.41 (1)**	87.87 ± 2.27 (7)	91.33 ± 1.63 (5)	91.48 ± 1.44 (3)
Indian liver	69.37 ± 1.86	68.51 ± 2.05 (4)	68.47 ± 1.87 (5)	69.10 ± 1.76 (2)	66.22 ± 2.49 (7)	68.56 ± 1.75 (3)	68.44 ± 1.87 (6)	**69.65 ± 1.06 (1)**
Breast cancer	94.25 ± 0.60	94.92 ± 2.40 (7)	96.56 ± 1.00 (5)	96.59 ± 0.94 (4)	96.53 ± 0.86 (6)	96.89 ± 1.26 (2)	96.83 ± 1.01 (3)	**97.21 ± 0.99 (1)**
Australian	71.54 ± 3.01	83.56 ± 1.31 (6)	84.26 ± 1.18 (5)	84.40 ± 1.28 (4)	82.11 ± 1.36 (7)	85.13 ± 0.95 (3)	**85.56 ± 1.02 (1)**	85.42 ± 0.91 (2)
Credit	71.56 ± 5.41	83.70 ± 2.81 (6)	84.52 ± 1.93 (5)	84.74 ± 1.89 (4)	81.92 ± 2.34 (7)	85.50 ± 1.62 (2)	85.13 ± 1.01 (3)	**85.68 ± 1.31 (1)**
German	70.18 ± 0.92	70.56 ± 1.61 (7)	71.48 ± 0.89 (6)	72.02 ± 1.41 (5)	72.22 ± 1.95 (4)	73.80 ± 1.60 (3)	74.56 ± 1.47 (2)	**75.1 ± 1.50 (1)**
Sonar	50.29 ± 2.79	82.98 ± 5.60 (7)	84.13 ± 2.87 (2)	83.94 ± 2.75 (3)	83.75 ± 3.28 (6)	83.94 ± 3.10 (3)	**84.23 ± 2.83 (1)**	83.84 ± 2.74 (5)
Musk1	57.35 ± 2.46	83.90 ± 9.65 (7)	91.05 ± 1.65 (4)	91.13 ± 1.72 (2)	91.09 ± 1.71 (3)	90.63 ± 1.81 (6)	90.92 ± 1.56 (5)	**91.17 ± 1.77 (1)**
Average rank		6	4.5	3.5	5.17	3.5	3	1.67

The relative ranking of each method on each data is shown in the parenthesis. The last row presents the average rank score over all data sets achieved by each algorithm

The values highlighted in bold refer to the system that has the rank 1 for each dataset

7.2 Evaluation of multilayers of multiple kernel algorithms

In this set of experiments, we evaluated the classification performance of the different MLMKL algorithms: DMKL, 2LMKL and MLMKL. Both DMKL and MLMKL algorithms are trained up to three layers. Furthermore, we compared these deep algorithms with SKSVM. Table 3 shows the detailed results of the classification of the different algorithms. The average rank is presented in the last row of Table 3.

First, by comparing the results between SKSVM and the MLMKL methods, we found that these algorithms do not guarantee high performance over SVM with an RBF kernel. For instance, for the 2LMKL algorithm, it was surpassed by SKSVM over three data sets (liver, postoperative and Indian liver patient). Note that MLMKL methods were proposed to overcome the shallow architecture of MKL approaches, which do not always outperform SVM with an RBF kernel [29]. This observation validates our motivation for improving the existing MLMKL methods.

Second, the average rank of MLMKL over all layers is 2.72, and the average rank of DMKL over all layers is equal to 4.66. Thus, MLMKL algorithm achieves better performance over DMKL, 2LMKL and SKSVM methods. By comparing two-layer architecture of the deep algorithms, our method gives the best results with an average rank of 3, and DMKL is in the second position with an average rank equal to 4.5. 2LMKL algorithm is the worst with an average rank equal to 5.17.

Third, it can be seen that the classification accuracy increases with the addition of each successive layer for both optimization methods: the estimation of the leave-one-out error and backpropagation function. It can be seen that there was an increase in accuracy with the addition of each successive layer for both DMKL and MLMKL algorithms. Specifically, there was a larger increase in accuracy with the addition of the third layer with our algorithm than with DMKL. For instance, the third layer of MLMKL increases the classification results over nine data sets with an average of 0.96 % (maximum increases is 3.41 %). However, DMKL has an increased accuracy over ten data sets with an average of 0.41 % (maximum increases is 1.36 %).

Finally, the classification performance shows that the developed MLMKL method is efficient when compared with the previous MLMKL algorithms (optimizing over leave-one-out error and the dual objective function). Particularly, on the one hand, MLMKL achieves better average accuracy over six data sets (liver, Pima Indian diabetes, Australian, etc.). On the other hand, it produces slightly better or equal average accuracy over DMKL and 2LMKL algorithms on four data sets (Monk3, Indian Liver, etc.). Hence, the empirical studies show that our MLMKL method is much better than the MKL methods and is also effective than the existing MLMKL methods with deep learning. These results are consistent with the results of deep neural networks, in which training a deep architecture with backpropagation algorithm improves the generalization performance.

8 Conclusions and perspectives

In this paper, we proposed a framework of multilayer of multiple kernel learning (MLMKL), which is a new active research area for kernel methods. Our algorithm consists on combining multiple base kernels in antecedent layers to form new inputs to the base kernels in subsequent layers. Then, we optimized the multilayer multiple kernels over an adaptive backpropagation algorithm. The proposed optimization method achieved better performance than the existing algorithms based on either the dual objective function or the estimation of the leave-one-out error. We have successfully optimized the proposed framework over many layers. Experimental results over 12 benchmark data sets demonstrated the efficiency of our MLMKL approach over both MKL and MLMKL methods.

Despite the encouraging results, some improvements need to be achieved. As future work, we plan to combine the effectiveness of the proposed method with the capacity of multiple classifier methods. Deep learning machines usually perform much better when using big data. Thus, we plan to apply our method on big data applications.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bach F, Lanckriet G, Jordan M (2004) Multiple kernel learning, conic duality, and the SMO algorithm. In: International conference on machine learning, pp 1–9
2. Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol (TIST) 2(3):27
3. Chen G, Parada C, Heigold G (2014) Small-footprint keyword spotting using deep neural networks. In: International conference on acoustics, speech and signal processing, pp 4087–4091
4. Cho Y (2012) Kernel methods for deep learning. Ph.D. thesis, University of California, San Diego, CA, USA
5. Cho Y, Saul L (2009) Kernel methods for deep learning. In: Advances in neural information processing systems, pp 342–350
6. Ciresan D, Meier U, Gambardella L, Schmidhuber J (2010) Deep big simple neural nets excel on handwritten digit recognition. In: CoRR, pp 1–14. arXiv:1003.0358
7. Cortes C, Mohri M, Rostamizadeh A (2009) L2 regularization for learning kernels. In: Conference on uncertainty in artificial intelligence, pp 109–116

8. Cortes C, Mohri M, Rostamizadeh A (2009) Learning non-linear combinations of kernels. In: Advances in neural information processing systems, pp 396–404
9. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
10. Cristianini N, Shawe-Taylor J (2000) Support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge
11. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. Parallel Distrib Process 1:318–362
12. Hu M, Chen Y, Kwok J (2009) Building sparse multiple-kernel SVM classifiers. IEEE Trans Neural Netw 20:827–839
13. Kloft M, Brefeld U, Sonnenburg S, Laskov P, Muller KR, Zien A (2009) Efficient and accurate Lp-norm multiple kernel learning. Adv Neural Inf Process Syst 22(22):997–1005
14. Kloft M, Brefeld U, Sonnenburg S, Zien A (2011) Lp-norm multiple kernel learning. J Mach Learn Res 12:953–997
15. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
16. Lanckriet G, Cristianini N, Bartlett P, Ghaoui L, Jordan MI (2004) Learning the kernel matrix with semidefinite programming. J Mach Learn Res 5:27–72
17. Martin K, Grzl F, Hannemann M, Vesely K, Cernocky J (2013) BUT BABEL system for spontaneous Cantonese. In: Interspeech, pp 2589–2593
18. Mika S, Scholkopf B, Smola A, Muller K, Scholz M, Ratsch G (1998) Kernel PCA and denoising in feature spaces. In: Advances in neural information processing systems, pp 536–542
19. Rakotomamonjy A, Bach FR, Canu S, Grandvalet Y (2008) Simple MKL. J Mach Learn Res 9:2491–2512
20. Rojas R (1996) Neural networks: a systematic introduction. Springer, New York
21. Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, Cambridge
22. Sonnenburg S, Rtsch G, Schfer C, Schlkopf B (2006) Large scale multiple kernel learning. J Mach Learn Res 7:1531–1565
23. Strobl E, Visweswaran S (2013) Deep multiple kernel learning. In: International conference on machine learning and applications, pp 414–417
24. Varma M, Babu B (2009) More generality in efficient multiple kernel learning. In: International conference on machine learning, pp 1065–1072
25. Vishwanathan S, Sun Z, Ampornpunt N, Varma M (2010) Multiple kernel learning and the SMO algorithm. In: Advances in neural information processing systems, pp 2361–2369
26. Wiering MA (2013) The neural support vector machine. In: The 25th Benelux artificial intelligence conference, pp 1–8
27. Xu X, Tsang I, Xu D (2013) Soft margin multiple kernel learning. IEEE Trans Neural Netw Learn Syst 24:749–761
28. Xu Z, Jin R, King I, Lyu M (2009) An extended level method for efficient multiple kernel learning. In: Advances in neural information processing systems, pp 1825–1832
29. Zhuang J, Tsang I, Hoi S (2011) Two-layer multiple kernel learning. In: International conference on artificial intelligence and statistics, pp 909–917
30. Zien A, Ong C (2007) Multiclass multiple kernel learning. In: International conference on machine learning, pp 1191–1198