Hindawi
Computational and Mathematical Methods in Medicine
Volume 2020, Article ID 7980249, 9 pages
https://doi.org/10.1155/2020/7980249

Research Article

EEG-Based Epilepsy Recognition via Multiple Kernel Learning

Yufeng Yao,[1,2] **Yan Ding**,[2] **Shan Zhong**,[2] **and Zhiming Cui**[3]

[1]*The Institute of Intelligent Information Processing and Application, Soochow University, Suzhou 215006, China*
[2]*Department of Computer Science and Engineering, Changshu Institute of Technology, Changshu 215500, China*
[3]*Suzhou University of Science and Technology, Suzhou 215009, China*

Correspondence should be addressed to Zhiming Cui; zmcui@usts.edu.cn

Received 25 June 2020; Accepted 3 August 2020; Published 29 September 2020

Guest Editor: Chenxi Huang

In the field of brain-computer interfaces, it is very common to use EEG signals for disease diagnosis. In this study, a style regularized least squares support vector machine based on multikernel learning is proposed and applied to the recognition of epilepsy abnormal signals. The algorithm uses the style conversion matrix to represent the style information contained in the sample, regularizes it in the objective function, optimizes the objective function through the commonly used alternative optimization method, and simultaneously updates the style conversion matrix and classifier during the iteration process parameter. In order to use the learned style information in the prediction process, two new rules are added to the traditional prediction method, and the style conversion matrix is used to standardize the sample style before classification.

1. Introduction

Due to the proposal of support vector machine (SVM) [1] and the development of related theories, the kernel method has become an effective method to deal with nonlinear fractional data. Since the performance of the classification algorithm depends largely on the representation of data, the kernel method uses relatively simple functional operations to map samples to higher dimensions, avoiding the design of feature space and complex inner product calculation in feature space. For example, in [2], a fast kernel ridge regression was proposed by using the kernel method. In the last decades, the kernel method has been applied in many fields of machine learning [3–5].

However, some data sets contain samples with uneven distribution, heterogeneous features, or irregular data; the single-kernel method using only a single feature space performs poorly. And since different kernel functions have their characteristics, even in the same application, the effect of using different kernel functions may be very different, which makes the selection of kernel functions and their parameters have an important influence on the performance of the algorithm. Since one kernel function often cannot meet the requirements in some practical application scenarios, multikernel learning that combines multiple kernel functions has been attracting more attention [6].

The combination generated by multikernel learning can be the combination of the same kernel function under different parameters or the combination of many different kernel functions [7]. After years of research, compared with single kernel function, multikernel learning has stronger flexibility, higher interpretability, and better performance in data dimension reduction [8], text classification [9], domain adaptation [10], and other fields.

Although the multikernel learning algorithm fully combines the mapping ability of different kernel functions for data, essentially, it only uses the physical characteristics of samples that include similarity and distance and fails to take into account the implicit information in the stylized data set in the real situation. In practical application, in addition to the representative content information, the data set often contains a variety of style information, and samples with the same style often exist in the form of groups. For example, there are two ways of dividing the letters shown in Figure 1(a), i.e., by the content shown in Figure 1(b) and by the font shown in Figure 1(c), where each font is regarded as a style, and such data is regarded as stylized data.

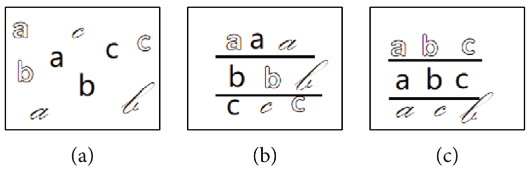

Figure 1: Example of stylistic data: (a) data set; (b) different contents; (c) different styles.

To mine the style information of data, scholars have done many types of research. The second-order statistical model proposed in the literature [11] is applied to the problem of number recognition, but it only has a good effect on the data subject to the Gaussian distribution, which leads to a great limitation in the application scenario of the algorithm. The bilinear discriminant model proposed in the literature [12] has achieved good results in behavior recognition data, but the computational cost of the algorithm is relatively high. The domain Bayesian algorithm proposed in the literature [13] improves the naive Bayesian algorithm to identify the style information in the sample group, but it needs to specify a clear data distribution type for the algorithm in advance. However, the distribution of data in real situations is often complex and difficult to be determined in advance. The algorithm proposed in the literature [14, 15] uses a single mapping to mine the style information of samples and achieves excellent results in regression and classification problems, but it makes limited use of the physical characteristics of samples. The time-series style model of mining sample historical information proposed in the literature [16] and the bilayer clustering model of user's age and gender information proposed in the literature [17] effectively make use of the style information in the data in the unsupervised problem, but the algorithm is only targeted at specific fields, and the use of style information is limited.

Inspired by the above scholars, we propose style regularization least squares support vector machine based on multiple kernel learning (SR-MKL-SVM) to excavate and utilize the physical similarities between sample points and the implied style information in samples. In addition to using the physical characteristics of each basic kernel function for data mapping to express the similarity between samples, the algorithm uses the style transformation matrix to represent and mine the style information contained in the data set and takes it into the objective function. In the training process, the alternate optimization strategy is used to update the style transformation matrix in addition to the classifier parameters, and the mined style information is used to synchronously update the kernel matrix. To use the sample style information obtained by training in the process of prediction, two new prediction rules are added on top of the prediction method of traditional multikernel least squares support vector machine. Because the style information contained in the sample is used effectively in the training and prediction process, the experiments of most of the stylized data sets show that SR-MKL-SVM is relatively recent and the classical multikernel support vector machine algorithm is effective.

2. Related Works

2.1. Multikernel Learning. Let $\mathbf{x}$ and $\mathbf{z}$ be two sample vectors; Φ is a mapping function from the input space to the feature space. If there is a function $k(\cdot,\cdot)$, which can be defined as

$$k(\mathbf{x},\mathbf{z}) = <\Phi(\mathbf{x}),\Phi(\mathbf{z})> = \Phi^T(\mathbf{x})\Phi(\mathbf{z}), \tag{1}$$

then we call $k(\cdot,\cdot)$ the kernel function. Multikernel learning expects to achieve better mapping performance by combining different kernel functions. There are many ways to combine [6] kernel functions. In this study, we use the following way to find a final combined kernel function based on M basic kernel functions $k_i(\cdot,\cdot)$. If we use μ_i to represent the kernel function coefficient, then the final combined kernel function is formulated as

$$k(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{M} \mu_i k_i(\mathbf{x},\mathbf{z}), \tag{2}$$

where

$$\sum_{i=1}^{M} \mu_i, \quad \mu_i > 0, i = 1, 2, \cdots M. \tag{3}$$

According to Mercer's theory, the combined kernel function generated by the above method still meets the Mercer condition.

2.2. Least Squares Support Vector Machine Based on Multikernel Learning. Let $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \cdots, (\mathbf{x}_n, y_n)\}$ be the training sample set; $\mathbf{x}_j \in \mathbf{R}^d$ and $y_i \in \{+1,-1\}$ are the label corresponding to $\mathbf{x}_j$. The objective function of the least squares support vector machine (LSSVM) proposed by Suykens [18] can be formulated as

$$\begin{aligned} &\min_{w,b,e} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\lambda}{2}\sum_{j=1}^{n} e_j^2 \\ &\text{s.t.} \quad y_j = \mathbf{w}^T\mathbf{\Phi}(\mathbf{x}_j) + b + e_j, \quad j = 1, 2, \cdots, n, \end{aligned} \tag{4}$$

where $\mathbf{\Phi}(\mathbf{x}_j)$ represents the mapped $\mathbf{x}_j$ in a high dimension, $\mathbf{w}$ and b are the classification hyperplane parameters, $e_j (j = 1, 2, \cdots n)$ is the error term, and λ is the regularization parameter.

The Lagrange multiplier $\boldsymbol{\alpha}$ is introduced into Equation (4), and its dual form can be further obtained by the Slater constraint specification:

$$\begin{aligned} &\max_{a} -\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} - \frac{1}{2\lambda}\boldsymbol{\alpha}^T\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{Y} \\ &\text{s.t.} \quad \boldsymbol{\alpha}^T\mathbf{I}_n = 0, \end{aligned} \tag{5}$$

```
MK-LSSVM.
Input: Dataset $\mathbf{D}$, threshold $\sigma$, parameter $\lambda$
Output: $\{\boldsymbol{\mu}, \mathbf{w}, b\}$
1. Set $\mu_i = 1/M (i = 1, 2, \cdots, M)$
2. Set $iter = 1$
3. Repeat
4.         Use (2) to combine kernel function and (6) to compute $\{\mathbf{a}, b\}$
5.         Fix $\boldsymbol{\alpha}$, update $\boldsymbol{\mu}$ by equation (8)
6. Until$(|1 - f_{iter}(\boldsymbol{\mu}, \boldsymbol{\alpha})/f_{iter}(\boldsymbol{\mu}, \boldsymbol{\alpha})| \le \sigma)$
7. Uses $\{\boldsymbol{\mu}, \boldsymbol{\alpha}\}$ to compute $\mathbf{w}$
```

Algorithm 1.

where $\mathbf{K} \in \mathbf{R}^{n\times n}$ is the kernel matrix. With (11), we can obtain the following two equations,

$$\begin{aligned} \boldsymbol{\alpha} &= \tilde{\mathbf{K}}^{-1}(\mathbf{Y} - b\mathbf{l}_n), \\ b &= \mathbf{I}_n{}^T\tilde{\mathbf{K}}^{-1}\mathbf{Y}\left(\mathbf{I}_n{}^T\tilde{\mathbf{K}}^{-1}\mathbf{I}_n\right)^{-1}, \end{aligned} \tag{6}$$

where $\tilde{\mathbf{K}} = \mathbf{K} + \mathbf{I}/\lambda$ and $\mathbf{Y} = (y_1, y_2, \cdots, y_n)^T$.

By integrating $\mathbf{K}$ into (2) and (3), we can obtain multiple kernel least squares support vector machine (MK-LSSVM) as

$$\begin{aligned} &\min_{\mu} \max_{\alpha} -\frac{1}{2}\sum_{i=1}^{M}\mu_i\boldsymbol{\alpha}^T\mathbf{K}_i\boldsymbol{\alpha} - \frac{1}{2\lambda}\boldsymbol{\alpha}^T\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{Y} \\ &\text{s.t.} \quad \sum_{i=1}^{M}\mu_i = 1, \quad 0 \le \mu_i, \boldsymbol{\alpha}^T\mathbf{l} = 0. \end{aligned} \tag{7}$$

Let $f(\boldsymbol{\mu}, \boldsymbol{\alpha}) = \max_{\alpha} -(1/2)\sum_{i=1}^{M}\boldsymbol{\alpha}^T\mathbf{K}_i\boldsymbol{\alpha} - (1/2\lambda)\boldsymbol{\alpha}^T\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{Y}$, and replace $f(\boldsymbol{\mu}, \boldsymbol{\alpha})$ with t; we have

$$\begin{aligned} &\min_{\mu} t \\ &\text{s.t.} \quad f(\boldsymbol{\mu}, \boldsymbol{\alpha}) \le t, \\ &\boldsymbol{\mu}\mathbf{l}_n = 1, \quad 0 \le \mu, \boldsymbol{\alpha}^T\mathbf{l} = 0. \end{aligned} \tag{8}$$

It is obvious that (8) is a semi-infinite linear program (SILP) problem, which can be solved by many existing mature optimization toolkits. For an unseen sample $\mathbf{x}$, MK-LSSVM predicts it by using the following equation:

$$\begin{aligned} y &= \text{sign}\left(w^T\Phi(x) + b\right) = \text{sign}\left(\sum_{j=1}^{n}\alpha_j\sum_{i=1}^{M}\mu_i k_i(x_j, x) + b\right) \\ &= \text{sign}\left(\sum_{i=1}^{M}\mu_j\sum_{j=1}^{n}\alpha_i k_i(x_j, x) + b\right) = \\ &\text{sign}\left(\sum_{i=1}^{M}\sum_{j=1}^{n}\mu_j w_i{}^T\Phi_i(x) + b\right). \end{aligned} \tag{9}$$

2.3. MK-LSSVM Algorithm Process. The algorithm steps of MK-LSSVM is shown in Algorithm 1.

3. SR-MKL-SVM

3.1. Objective Function. Let $\mathbf{D} = \{(\mathbf{x}_1^1, y_1^1), \cdots, (\mathbf{x}_1^t, y_1^{t_1}), \cdots, (\mathbf{x}_N^1, y_N^1), \cdots, (\mathbf{x}_N^{t_N}, y_N^{t_N})\}$ be a training set, where the set can be divided into N groups according to the style. The samples in each group have the same style, and the superscript t_j is the number of samples in group j. $\mathbf{x}_j^k \in \mathbf{R}^d (j = 1, 2, \cdots, N, k = 1, 2, \cdots, t_j)$ is the kth sample in group j. Under the above definition, the objective function of **SR-MKL-SVM** can be formulated as

$$\begin{aligned} \min_{w_i, b, \mu_i, A_j} J_{\mathbf{SR-MKL-SVM}} &= \frac{1}{2}\sum_{i=1}^{M}\mu_i\mathbf{w}_i{}^T\mathbf{w}_i + \frac{\lambda}{2}\sum_{j=1}^{N}\sum_{k=1}^{t_j}\left(e_j{}^k\right)^2 \\ &\quad + \lambda\gamma\sum_{j=1}^{N}\left\|\mathbf{A}_j^T - \mathbf{l}\right\|_F^2 \\ \text{s.t.} \quad y_j^k &= \sum_{i=1}^{M}\mu_i\mathbf{w}_i^T\mathbf{A}_j^T\Phi_i\left(x_j^k\right) + b + e_j^k, \\ \sum_{i=1}^{M}\mu_i &= 1, \quad \mu_i \ge 0, i = 1, 2, \cdots, M, j = 1, 2, \cdots, N, k = 1, 2, \cdots, t_j, \end{aligned} \tag{10}$$

where $\mu_i (i = 1, 2, \cdots, M)$ is the weight coefficient of the kernel matrix, where M is the number of predefined kernel matrices, $\{\mathbf{A}_j \in \mathbf{R}^{d\times d}\}$ is the style conversion matrix of the sample of style j, and $\mathbf{I} \in \mathbf{R}^{d\times d}$ is the identity matrix.

The first two subformulas in $J_{\mathbf{SR-MKL-SVM}}$ are standard MK-LSSVM expressions, and the third subformula is a penalty term using the Frobenius norm, which is used to control the degree of style conversion of the style conversion matrix to the sample, where the parameter $\gamma \in \mathbf{R}(\gamma > 0)$ is used. Obviously, when γ is larger, the deviation of the sample $\mathbf{A}_j^T\Phi(\mathbf{x}_j^k)$ is smaller after style conversion from its original style; otherwise, it is larger; especially when $\gamma \to +\infty$ is set, there is $\mathbf{A}_j^T\boldsymbol{\Phi}(\mathbf{x}_j^k) \to \boldsymbol{\Phi}(\mathbf{x}_j^k)$.

3.2. Optimization. The goal of the algorithm is to minimize the value of $J_{\mathbf{SR-MKL-SVM}}$. It is very difficult to directly optimize the objective function. We can use the alternating optimization method to obtain a sufficiently available local optimal solution. When $\mathbf{A}_j$ and $\{\mathbf{w}_i, \mathrm{b}, \mu_i\}$ are given separately, the objective functions are optimization problems about $\{\mathbf{w}_i, \mathrm{b}, \mu_i\}$ and $\mathbf{A}_j$, and the above two processes are repeated until convergence or the maximum number of iterations is exceeded. To be specific,

(1) When fixing $A_j (j = 1, 2, \cdots, N)$, the optimization problem of formula (10) is transformed into

$$\begin{aligned} &\min_{w_i, b, \mu_i} \frac{1}{2}\sum_{i=1}^{M} \mu_i \mathbf{w}_i{}^T \mathbf{w}_i + \frac{\lambda}{2}\sum_{j=1}^{N}\sum_{k=1}^{t_j}\left(e^k\right)^2 \\ &\text{s.t.} \quad y_j^k = \sum_{i=1}^{M} \mu_i \mathbf{w}_i^T \mathbf{A}_j^T \mathbf{\Phi}_i\left(\mathbf{x}_j^k\right) + b + e_j^k, \\ &\sum_{i=1}^{M} \mu_i = 1, \quad \mu_i \geq 0, i = 1, 2, \cdots, M, j = 1, 2, \cdots, N, k = 1, 2, \cdots, t_j. \end{aligned} \tag{11}$$

The above formula is about the standard MK-LSSVM problem of sample $\mathbf{A}_j\mathbf{\Phi}_i(\mathbf{x}_j^k)$ after style conversion, and $\{\mu_i, \mathbf{w}_i, \mathrm{b}\}$ can be determined by Algorithm 1 in Section 2.2 of the article. At this time, the sample $\mathbf{A}_j\mathbf{\Phi}_i(\mathbf{x}_j^k)$ mapped to the high dimension cannot be directly calculated, but the synthetic kernel matrix formed by the style-converted sample $\mathbf{A}_j\mathbf{\Phi}_i(\mathbf{x}_j^k)$ can be updated by the kernel method to obtain the style-converted kernel matrix. The specific method of using the kernel method to obtain the style-converted synthetic kernel matrix will be introduced in Section 3.3 of the article

(2) When $\{\mu_i, \mathbf{w}_i, \mathrm{b}\}$ is fixed, then the optimization problem of Equation (10) is transformed into

$$\begin{aligned} &\min_{\mathbf{A}_j \in \mathbf{R}^{d\times d}} \frac{\lambda}{2}\sum_{k=1}^{t_j}\left(e_j{}^k\right)^2 + \lambda\gamma\left\|\mathbf{A}_j^T - \mathbf{I}\right\|_F^2 \\ &\text{s.t.} \quad y_j^k = \sum_{i=1}^{M} \mu_i \mathbf{w}_i^T \mathbf{A}_j^T \mathbf{\Phi}_i\left(\mathbf{x}_j^k\right) + b + e_j^k, \\ &\sum_{i=1}^{M} \mu_i = 1, \quad \mu_i \geq 0, i = 1, 2, \cdots, M, j = 1, 2, \cdots, N, k = 1, 2, \cdots, t_j. \end{aligned} \tag{12}$$

The above formula is a linear constrained quadratic programming problem for $\mathbf{A}_j$, which can be transformed into N independent problems for each $\mathbf{A}_j$ to be solved. At this time, the parameters of the synthetic kernel matrix and the classifier have been fixed, similar to the original LSSVM, and the dual form can be obtained after introducing the Lagrange multiplier to Equation (12):

$$\begin{aligned} \max_{\boldsymbol{\alpha}_j^k} L = {} & \frac{\lambda}{2}\sum_{k=1}^{t_j}\left(e_j{}^k\right)^2 + \lambda\gamma\left\|\mathbf{A}_j^T - \mathbf{I}\right\|_F^2 \\ & - \sum_{k=1}^{t_j} \boldsymbol{\alpha}_j^k \left(\sum_{i=1}^{M} \mu_i \mathbf{w}_i^T \mathbf{A}_j^T \mathbf{\Phi}_i\left(\mathbf{x}_j^k\right) + b + e_j^k - y_j^k\right) \\ & - \sum_{i=1}^{M}\left[\beta_i\left(1 - \sum_{i=1}^{M}\mu_i\right) - \rho_i \mu_i\right]. \end{aligned} \tag{13}$$

Let $\partial L/\partial \mathbf{A}_j = 0$; we have

$$\mathbf{A}_j^T = \frac{1}{2\lambda\gamma}\sum_{k=1}^{t_j}\sum_{i=1}^{M} \boldsymbol{\alpha}_j^k \mu_i \mathbf{w}_i \mathbf{\Phi}_i^T\left(\mathbf{x}_j^k\right) + \mathbf{I}. \tag{14}$$

Let $\partial L/\partial e_j^k = 0$ get $\boldsymbol{\alpha}_j^k = \lambda e_j^k$. It can be seen that this formula has the same KKT [18] condition as LSSVM.

Through the process of alternating optimization, it can be known that in the process of training classifier parameters, the samples converted by the style conversion matrix are used as training data. In the first iteration, the style conversion matrix is initialized to the identity matrix. At this time, the samples after the style conversion are the same as the original samples, and no style conversion is generated. Therefore, the classifier parameters obtained by the first round of SR-MKL-SVM training are the same as the original MK-LSSVM. In the subsequent iteration process, due to the optimization of the style conversion matrix, the samples in each style group undergo the transformation of the style conversion matrix and gradually approach the standard style. The classifier parameters trained at this time fully consider the style information contained in the sample as a whole. At the same time, the process of solving the style conversion matrix from Equation (14) not only uses the physical characteristics of the samples obtained by training but also effectively uses the style information in the data. The style conversion matrix trained at this time contains each style group style information. According to the above analysis, the processes of training the classifier parameters and the style conversion matrix make full use of the style information contained in the sample, and the two processes promote each other.

3.3. Style Transformation. Since the dimension after the sample is mapped to the high-dimensional space may be infinite, the sample $\mathbf{A}_j\Phi(x_j^k)$ value after the style transformation cannot be obtained directly. At this point, each element in the synthetic kernel matrix can be updated with the help of the kernel method to obtain the synthetic kernel matrix after the style transformation.

Because the synthesis kernel function still has to satisfy the allowed kernel of the theorem, as $k(\mathbf{x}_{j1}^{k1}, \mathbf{x}_{j2}^{k2}) = <\mathbf{\Phi}(\mathbf{x}_{j1}^{k1}), \mathbf{\Phi}(\mathbf{x}_{j2}^{k2}) > = \sum_{i=1}^{M}\mu_i k_i(\mathbf{x}_{j1}^{k1}, \mathbf{x}_{j2}^{k2})$, let ϕ be for the synthesis of the combined map of the core matrix; by formula (9), you can make the synthesis of the core matrix $k(\mathbf{x}_{j1}^{k1}, \mathbf{x}_{j2}^{k2}) =$

$<\mathbf{\Phi}(\mathbf{x}_{j1}^{k1}), \mathbf{\Phi}(\mathbf{x}_{j2}^{k2})> = \sum_{i=1}^{M}\mu_i < \mathbf{\Phi}_i(\mathbf{x}_{j1}^{k1}), \mathbf{\Phi}_i(\mathbf{x}_{j2}^{k2})>$; let $\widehat{\mathbf{\Phi}}(x_j^k) = \mathbf{A}_j^T\mathbf{\Phi}(x_j^k)$; you can get after the style conversion of the core matrix elements as

$$\begin{aligned}\widehat{k}\left(\mathbf{x}_{j1}^{k1}, \mathbf{x}_{j2}^{k2}\right) &= \mathbf{\Phi}\wedge^T\left(\mathbf{x}_{j1}^{k1}\right)\widehat{\mathbf{\Phi}}\left(\mathbf{x}_{j2}^{k2}\right) \\ &= <\widehat{\mathbf{\Phi}}\left(\mathbf{x}_{j1}^{k1}\right), \widehat{\mathbf{\Phi}}\left(\mathbf{x}_{j2}^{k2}\right)> \\ &= \mathbf{\Phi}^T\left(\mathbf{x}_{j1}^{k1}\right)\mathbf{A}_{j1}\mathbf{A}_{j2}^T\mathbf{\Phi}\left(\mathbf{x}_{j2}^{k2}\right),\end{aligned} \tag{15}$$

where $k^\wedge(\mathbf{x}_{j1}^{k1}, \mathbf{x}_{j2}^{k2})$ is the core matrix element after style transformation and $\mathbf{w} = \sum_{j=1}^{N}\sum_{k=1}^{t_j}\boldsymbol{\alpha}_j^k\mathbf{\Phi}(x_j^k)$; formula (15) can be updated to

$$\begin{aligned}\widehat{k}\left(\mathbf{x}_{j1}^{k1}, \mathbf{x}_{j2}^{k2}\right) &= \mathbf{\Phi}^T\left(\mathbf{x}_{j1}^{k1}\right)\mathbf{A}_{j1}\mathbf{A}_{j2}^T\mathbf{\Phi}\left(\mathbf{x}_{j2}^{k2}\right) = \mathbf{\Phi}^T\left(\mathbf{x}_{j1}^{k1}\right) \\ &\quad\cdot\left[\frac{1}{2\lambda\gamma}\sum_{m1=1}^{N}\sum_{n1=1}^{t_{m1}}\boldsymbol{\alpha}_{j1}^{k1}\boldsymbol{\alpha}_{m1}^{n1}\mathbf{\Phi}\left(\mathbf{x}_{j1}^{k1}\right)\mathbf{\Phi}^T\left(\mathbf{x}_{m1}^{n1}\right) + \mathbf{I}\right] \\ &\quad\cdot\left[\frac{1}{2\lambda\gamma}\sum_{m2=1}^{N}\sum_{n2=1}^{t_{m2}}\boldsymbol{\alpha}_{m2}^{n2}\boldsymbol{\alpha}_{j2}^{k2}\mathbf{\Phi}\left(\mathbf{x}_{m2}^{n2}\right)\mathbf{\Phi}^T\left(\mathbf{x}_{j2}^{k2}\right) + \mathbf{I}\right] \\ &\quad\cdot\mathbf{\Phi}\left(\mathbf{x}_{j2}^{k2}\right) \\ &= \frac{1}{4\lambda^2\gamma^2}\cdot\left[\sum_{m1=1}^{N}\sum_{m2=1}^{N}\sum_{n1=1}^{t_{m1}}\sum_{n2=1}^{t_{m2}}\boldsymbol{\alpha}_{m1}^{n1}\boldsymbol{\alpha}_{m2}^{n2}\boldsymbol{\alpha}_{j1}^{k1}\boldsymbol{\alpha}_{j2}^{k2}\right. \\ &\quad\cdot\left\langle\mathbf{\Phi}\left(\mathbf{x}_{j1}^{k1}\right), \left(\mathbf{x}_{j1}^{k1}\right)\right\rangle\cdot\left\langle\mathbf{\Phi}\left(\mathbf{x}_{j2}^{k2}\right), \mathbf{\Phi}\left(\mathbf{x}_{j2}^{k2}\right)\right\rangle \\ &\quad\left.\cdot\left\langle\mathbf{\Phi}\left(\mathbf{x}_{m1}^{n1}\right), \mathbf{\Phi}\left(\mathbf{x}_{m2}^{n2}\right)\right\rangle\right] + \frac{1}{2\lambda\gamma}\sum_{m1=1}^{N}\sum_{n1=1}^{t_{m1}}\boldsymbol{\alpha}_{j1}^{k1}\boldsymbol{\alpha}_{m1}^{n1} \\ &\quad\cdot\left\langle\mathbf{\Phi}\left(\mathbf{x}_{j1}^{k1}\right), \mathbf{\Phi}\left(\mathbf{x}_{j1}^{k1}\right)\right\rangle\cdot\left\langle\mathbf{\Phi}\left(\mathbf{x}_{m1}^{n1}\right), \mathbf{\Phi}\left(\mathbf{x}_{j2}^{k2}\right)\right\rangle \\ &\quad+ \frac{1}{2\lambda\gamma}\sum_{m2=1}^{N}\sum_{n2=1}^{t_{m2}}\boldsymbol{\alpha}_{j2}^{k2}\boldsymbol{\alpha}_{m2}^{n2}\cdot<\mathbf{\Phi}\left(\mathbf{x}_{j1}^{k1}\right), \mathbf{\Phi}\left(\mathbf{x}_{m2}^{n2}\right)> \\ &\quad\cdot<\mathbf{\Phi}\left(\mathbf{x}_{j2}^{k2}\right), \mathbf{\Phi}\left(\mathbf{x}_{j2}^{k2}\right)> + <\mathbf{\Phi}\left(\mathbf{x}_{j1}^{k1}\right), \mathbf{\Phi}\left(\mathbf{x}_{j2}^{k2}\right)>.\end{aligned} \tag{16}$$

Because of $k(x_{j1}^{k1}, x_{j2}^{k2}) = <\Phi(x_{j1}^{k1}), \Phi(x_{j2}^{k2})>$ formula (16) can be updated to:

$$\begin{aligned}\widehat{k}\left(x_{j1}^{k1}, x_{j2}^{k2}\right) &= \frac{1}{4\lambda^2\gamma^2}\cdot\left[\sum_{m1=1}^{N}\sum_{m2=1}^{N}\sum_{n1=1}^{t_{m1}}\sum_{n2=1}^{t_{m2}}\alpha_{m1}^{n1}\alpha_{m2}^{n2}\alpha_{j1}^{k1}\alpha_{j2}^{k2}\right] \\ &\quad\cdot k\left(x_{j1}^{k1}, x_{j1}^{k1}\right)\cdot k\left(x_{j2}^{k2}, x_{j2}^{k2}\right)\cdot k\left(x_{m1}^{n1}, x_{m2}^{n2}\right) \\ &\quad+ \frac{1}{2\lambda\gamma}\sum_{m1=1}^{N}\sum_{n1=1}^{t_{m1}}\alpha_{j1}^{k1}\alpha_{m1}^{n1}\cdot k\left(x_{j1}^{k1}, x_{j1}^{k1}\right) \\ &\quad\cdot k\left(x_{m1}^{n1}, x_{j2}^{k2}\right) + \frac{1}{2\lambda\gamma}\sum_{m2=1}^{N}\sum_{n2=1}^{t_{m2}}\alpha_{j2}^{k2}\alpha_{m2}^{n2} \\ &\quad\cdot k\left(x_{j1}^{k1}, x_{m2}^{n2}\right)\cdot k\left(x_{j2}^{k2}, x_{j2}^{k2}\right) + k\left(x_{j1}^{k1}, x_{j2}^{k2}\right)\end{aligned} \tag{17}$$

```
SR-MKL-SVM.
Input: Dataset D, parameters $\lambda$ and $\gamma$
Output: Weight $\mu_i$, classifier parameter $\{\mathbf{w}, b\}$, style transformation matrix $\{A_j\}$
1. Set $\mu_i = 1/M(i = 1, 2, \cdots, M)$
2. Set $iter = 1$, $A_j = I(j = 1, 2, \cdots, N)$
3. Repeat
4.     Calculate the value of the objective function $V_{iter}$
5.     Update $\mu_i$ and $\{\mathbf{w}, b\}$ by (11)
6.     Update $A_j$ by (14) and the synthetic kernel matrix by (11)
6. Until (Iteration count reaches its maximum)
```

Algorithm 2.

3.4. Algorithm. The training algorithm of SR-MKL-SVM is listed as follows.

SR-MKL-SVM uses alternate optimization method to solve the problem, which can be divided into two steps. The first step is kernel matrix weight coefficient and classifier parameter optimized steps can be divided into two subprocesses, respectively, i.e., solving the kernel weight SILP problems and solving the linear programming problem of the classifier parameters for the synthesis of kernel matrix at the same time, the time complexity $O(M^2n^2)$ and $O(n)$, respectively. Due to $M \geq 1$, the total time complexity can be treated as $O(M^2n^2)$. The second step is to optimize the style-standardization matrix, and the time complexity of this step is $O(N^2n^2)$. Therefore, the total time complexity of the algorithm training process is $O(\text{iter}\cdot(M^2n^3 + N^2n^2))$, where M is the number of predefined basic kernel matrices, N is the total number of samples, n is the number of styles in the data set, and iter is the number of iterations of the algorithm.

Compared with typical MKL-SVM, the MK-SRLSSVM algorithm is in the process of training in style transformation matrix to the regularization processing style samples, but the multikernel support vector machine (SVM) algorithm in solving the basic kernel function in the process of the weight coefficient is applied to solve the need to invoke the original SVM algorithm in this paper; using the original LSSVM subspaces, the SVM training process is essential in solving quadratic programming problem and the nature of the training process of LSSVM for solving linear programming problems. Therefore, the computational complexity of SR-MKL-SVM in this step is far less than that of the typical MKL-SVM algorithm. The algorithm presented in this paper optimizes the weight coefficient by solving SILP problems, which is superior to the support vector machine algorithm that optimizes the weight coefficient by solving SDP problems or QCQP problems and is comparable to the multikernel support vector machine algorithm that uses SILP and other problems to solve the weight coefficient. Therefore, SR-MKL-SVM has the same complexity as typical support vector machine algorithms.

3.5. Prediction Rules. Two new prediction rules were defined based on MK-LSSVM in order to use the weight, classifier parameter $\{\mu_i, \mathbf{w}, b\}$ and the style transformation matrix

$A_j (j = 1, 2, \cdots, N)$. Since the style of the sample may or may not appear in the training process in the practical application, two new prediction rules Rule 2 and Rule 3 are added into the traditional prediction method to deal with the two cases, respectively.

Let $\mathbf{X}_0 = \{\mathbf{x}_0^1, \mathbf{x}_0^2, \cdots, \mathbf{x}_0^{t_0}\}$ be a subset of the entire testing data set in which each element has the same style, and $\mathbf{x}_0^{t_0} \in R^d (k = 1, 2, \cdots t_0)$ is a sample.

Rule 1. Traditional prediction method.

Traditional prediction methods only use weight μ_i and classifier parameters w and b to predict the sample $\mathbf{x}_0^k$ in the testing data set and obtain the corresponding label y_0^k:

$$\begin{aligned} y_0^k &= \operatorname{sign}\left[\mathbf{w}^T \Phi\left(x_0^k\right) + b\right] \\ &= \operatorname{sign}\left[\sum_{j=1}^{N}\sum_{k=1}^{L_j} \alpha_j^k \sum_{i=1}^{M} \mu_i k_i\left(x_j^k, x_0^k\right) + b\right]. \end{aligned} \tag{18}$$

Rule 2. Test sample style is known.

If the style of the test sample already exists in the training data set, the corresponding style transformation matrix acquired during the training process can be directly used to process the style transformation of the sample, so that the sample is close to the standard style. Then, the predicted label y_0^k was obtained by using traditional prediction rules for the processed sample $A_0^T \Phi(x_0^k)$.

$$\begin{aligned} {y_0}^k &= \operatorname{sign}\left[\mathbf{w}^T {A_0}^T \Phi\left({x_0}^k\right) + b\right] \\ &= \operatorname{sign}\left[\sum_{j=1}^{N}\sum_{k=1}^{t_j} \alpha_j^k \sum_{i=1}^{M} \mu_i \widehat{k}_i\left(x_j^k, x_0^k\right) + b\right]. \end{aligned} \tag{19}$$

$\widehat{k}_i(x_j^k, x_0^k)$ can be obtained from Section 3.3.

Rule 3. Test sample style is unknown.

If the sample group $\mathbf{X}_0$'s style does not exist in the training data set, to effectively make use of the style information obtained by training; based on the direct extrapolation idea, we consider the same style of the information contained in the sample group as a new style. The detailed steps are as follows:

Step 1. Obtain the temporary label $Y_{\text{temp}} = \{y_0^1, y_0^2, \cdots, y_0^{t_0}\}$ of testing data set $\mathbf{X}_0$ by using Rule 1.

Step 2. Train $\mathbf{X}_0$ and its temporary label Y_{temp} with the training data set to obtain the new weight $\widehat{\mu}_i$, classifier parameter $\{\widehat{w}, \widehat{b}\}$, and style transformation matrix A_0^T.

Table 1: Description of epileptic EEG data set.

Group	Type
A	Healthy
B	
C	Patient
D	
E	

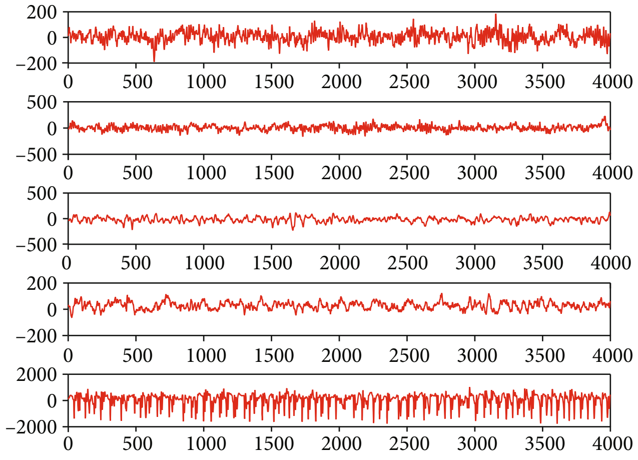

Figure 2: EEG data visualization.

Table 2: Detail of experimental data sets.

Num	Training data	Testing data
DS.1	Each 50% z (A, B, E)	Other 50% (A, B, E)
DS.2	Each 50% (B, D, E)	Other 50% (B, D, E)
DS.3	Each 50% (A, C ,E)	Other 50% (A, C, E)
DS.4	Each 50% (A, C, E)	Other 50% (A, C, E)

Step 3. Use $\{\widehat{\mu}_i, \widehat{w}, \widehat{b}, A_0^T\}$ to predict test set $\mathbf{X}_0$ and get the formal prediction label $Y_0 = \{y_0^1, y_0^2, \cdots, y_0^{t_0}\}$.

Since that most of the data in real scenes contain implicit or obvious style characteristics, the new prediction method added in SR-MKL-SVM takes into account the situation of known style and unknown style. The style information corresponding to the predicted samples is directly used to predict the samples with known styles. The direct extrapolation method is used to predict the unknown style samples, and the trained style information is used effectively, so the algorithm has good universality.

3.6. Analysis of SR-MKL-SVM. Different from SVM, which only searches for the optimal classification hyperplane according to the physical distribution of the original data, SR-MKL-SVM not only considers the physical characteristics contained in the data but also mines the style characteristics of the data. In this paper, the whole training samples are used to optimize the classifier parameters and the data sets with

Table 3: Experimental results of epileptic EEG.

Algorithm	Precision			
	DS.1	DS.2	DS.3	DS.4
simpleMKL	0.9273 $(C = e^4)$	0.7920 $(C = e^1)$	0.8133 $(C = e^5)$	0.8013 $(C = e^1)$
easyMKL	0.9520 $(\lambda_e = 0.1)$	0.8333 $(\lambda_e = 0.8)$	0.5333 $(\lambda_e = 0.6)$	0.7767 $(\lambda_e = 0.3)$
GMKL	0.9260 $(C = e^0)$	0.7867 $(C = e^5)$	0.7507 $(C = e^{-1})$	0.7973 $(C = e^2)$
LMKL	0.9600 $(C = e^0)$	0.8120 $(C = e^3)$	0.8200 $(C = e^3)$	0.8133 $(C = e^{-1})$
NLMKL	0.9507 $(C = e^1)$	0.9493 $(C = e^5)$	0.8293 $(C = e^1)$	0.8480 $(C = e^3)$
RBMKL	0.9413 $(C = e^0)$	0.8440 $(C = e^4)$	0.7787 $(C = e^{-1})$	0.8067 $(C = e^{-1})$
GLMKL	0.9413 $(C = e^1)$	0.8333 $(C = e^3)$	0.7627 $(C = e^0)$	0.8227 $(C = e^{-1})$
CABMKL	0.9373 $(C = e^2)$	0.8453 $(C = e^2)$	0.7560 $(C = e^0)$	0.8027 $(C = e^1)$
NB	0.9520	0.8947	0.8067	0.7087
DT	0.9747	0.8467	0.7533	0.8120
SR-MKL-SVM	0.9503 $(\lambda = e^{-1}, \gamma = e^{-3})$	0.9353 $(\lambda = e^3, \gamma = e^{-5}a)$	0.9153 $(\lambda = e^{-1}, \gamma = e^{-2})$	0.9120 $(\lambda = e^1, \gamma = e^{-1})$

different styles are processed, respectively. With the advantage of multikernel learning for data mapping, the algorithm in this paper can represent and process the data containing more complex styles and make full use of the trained style information to conduct style regularization processing on the original samples in both training and testing methods, so that the data distribution after style transformation can be more easily divided. Compared with traditional SVM and SR-MKL-SVM, we find that SR-MKL-SVM can make full use of the information contained in the stylized data to improve the classification performance.

4. Experimental Results

4.1. Data. In this section, we introduce the EEG data provided by Bonn University to evaluate our proposed method. The EEG data set consists of 5 groups of samples from 2 groups, with detailed information as shown in Table 1, and randomly selected samples from each group as shown in Figure 2. As can be seen from Figure 2, the fluctuation of samples from different groups is very different. For example, the signal fluctuation of patients in group A and healthy people in group E is significantly different. The signal fluctuation of patients in group C and group E also differed greatly under different conditions.

Studies [19] showed that feature extraction of original EEG data in advance could effectively improve classification performance. In this paper, kernel principal component analysis (KPCA) [5, 20] was used to extract features from original data. In this section, the data after dimension reduction is used for experiments. As can be seen above, the number of samples in the data set is 500, the number of categories is 2, and the sample dimension is 70. Samples from the same group are considered to have the same style.

In order to verify the validity of this algorithm, different groups of data are selected to form two types of data sets. The first type of data is all styles contained in the test set exist in the training set at the same time. The second type of data is the test set has a style not found in the training set, and the details of the construction data set are shown in Table 2.

4.1.1. Epileptic EEG Data Set. Data sets DS.1 and DS.2 are the first type of data; DS.3 and DS.4 are the second type of data. All data were random, and 10 experiments were conducted under the same set of parameters, averaging the results. Rule 2 and Rule 3 are used to predict the two types of data. The experimental results and parameters of all algorithms [21–32] are shown in Table 3.

From the experimental results in Table 3, it can be concluded that the decision tree algorithm in data set DS.1 has the best wave signal recognition effect, and the NLMKL algorithm in data set DS.2 has the best classification accuracy, leading all other algorithms including this algorithm. The results of this algorithm in the first two data sets are not as good as DT and NLMKL, but the difference is small.

From the above results, we can see the effectiveness and stability of the proposed algorithm in improving the accuracy of EEG signal recognition by mining and utilizing different fluctuation features contained in each group of samples.

5. Conclusion

In order to use the style information contained in the sample, this paper proposes a style regularization least squares support vector machine (SR-MKL-SVM) based on multicore learning. In addition to the advantage of multicore learning for the expression of physical similarity between samples, the algorithm also mines and uses the style information contained in the samples to improve the classification accuracy of the algorithm. SR-MKL-SVM takes the style information contained in the sample into the objective function, uses the style conversion matrix to standardize the sample, uses the regularization method to limit the degree of style conversion, and optimizes both the classifier parameters and the style standard during the training process conversion matrix. In addition to the traditional prediction methods, new prediction rules that can use the trained style information are added. Experiments in stylized data sets show the effectiveness and certain practicality of the algorithm.

Data Availability

The original EEG data are available and can be downloaded from http://www.meb.unibonn.de/epileptologie/science/physik/eegdata.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant 61876217.

References

[1] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[2] H. Avron, K. L. Clarkson, and D. P. Woodruff, “Faster kernel ridge regression using sketching and preconditioning,” *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 4, pp. 1116–1138, 2017.

[3] J. W. Tao and S. T. Wang, “Kernel support vector machine for domain adaptation,” *Acta Automatica Sinica*, vol. 38, no. 5, pp. 797–811, 2012.

[4] C. Feng and S. Z. Liao, “Large-scale kernel methods via random hypothesis spaces,” *Journal of Frontiers of Comp-uter Science & Technology*, vol. 12, no. 5, pp. 785–793, 2018.

[5] B. Schölkopf, S. Mika, A. Smola, G. Rätsch, and K.-R. Müller, “Kernel PCA pattern reconstruction *via* approximation preimages,” in *Proceedings of the 8th international conference on Artificial neural networks*, pp. 147–152, Skövde, Sweden, Piscataway, September 2-4, 1998.

[6] Y. Zhang, H. Ishibuchi, and S. Wang, “Deep Takagi–Sugeno–Kang fuzzy classifier with shared linguistic fuzzy rules,” *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1535–1549, 2018.

[7] R. Alain and C. Stéphane, “More efficiency in multiple kernel learning,” in *Proceedings of the 24th international conference on Machine learning*, pp. 775–782, Corvalis, Oregon, New York, 2007.

[8] Y. Zhang, F. Chung, and S. Wang, “Fast exemplar-based clustering by gravity enrichment between data objects,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 8, pp. 2996–3009, 2020.

[9] Y. Zhang, F. Chung, and S. Wang, “A multiview and multiexemplar fuzzy clustering approach: theoretical analysis and experimental studies,” *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 8, pp. 1543–1557, 2019.

[10] S. S. Bucak, Rong Jin, and A. K. Jain, “Multiple kernel learning for visual object recognition: a review,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1354–1369, 2014.

[11] S. Veeramachaneni and G. Nagy, “Style context with second-order statistics,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 14–22, 2005.

[12] M. S. Cheema, A. Eweiwi, and C. Bauckhage, “Human activity recognition by separating style and content,” *Pattern Recognition Letters*, vol. 50, pp. 130–138, 2014.

[13] X. Y. Zhang, K. Huang, and C. L. Liu, “Pattern field classification with style normalized transformation,” in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 1621–1626, Catalonia, Spain, 2011.

[14] H. C. Jiang, K. Z. Huang, and R. Zhang, “Field support vector regression,” in *Neural Information Processing: 24th International Conference*, pp. 699–708, Guangzhou, China, 2017.

[15] K. Z. Huang, H. C. Jiang, and X. Y. Zhang, “Field support vector machines,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 6, pp. 454–463, 2017.

[16] Y. Zhang, F. Chung, and S. Wang, “Fast reduced set-based exemplar finding and cluster assignment,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 5, pp. 917–931, 2019.

[17] G. Marzinotto, J. C. Rosales, M. A. El-Yacoubi, and S. Garcia-Salicetti, “Age and gender characterization through a two layer clustering of online handwriting,” in *International Conference on Advanced Concepts for Intelligent Vision Systems*, pp. 428–439, Catania, Italy, 2015.

[18] J. Suykens, T. van Gestel, J. de Brabanter, B. de Moor, and J. Vandewalle, “Least squares support vector machines,” *International Journal of Circuit Theory and Applications*, vol. 27, no. 6, pp. 605–615, 2002.

[19] Y. Jiang, Z. H. Deng, F. L. Chung et al., “Recognition of epileptic EEG signals using a novel multiview TSK fuzzy system,” *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 1, pp. 3–20, 2017.

[20] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, “Singular value decomposition and principal component analysis,” in *A Practical Approach to Microarray Data Analysis*, D. P. Berrar, W. Dubitzky, and M. Granzow, Eds., pp. 91–109, Springer, Boston, MA, 2003.

[21] M. Lu, L. H. Liu, and L. H. Wu, “Research on multi-kernel sup-port vector data description method of classification,” *Computer Engineering and Applications*, vol. 52, no. 18, pp. 68–73, 2016.

[22] H.-Q. Wang, F.-C. Sun, Y.-N. Cai, N. Chen, and L.-G. Ding, “On multiple kernel learning methods,” *Acta Automatica Sinica*, vol. 36, no. 8, pp. 1037–1050, 2010.

[23] X. Chen, N. Guo, Y. Ma, and G. Chen, “More efficient sparse multi-kernel based least square support vector machine,” *Communications and Information Processing*, vol. 289, pp. 70–78, 2012.

[24] M. Kloft, “l_p-norm multiple kernel learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 348–353, 2007.

[25] V. Manic and R. Bodla, “More generality in efficient multiple kernel learning,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1065–1072, Quebec, Canada, 2009.

[26] C. Cortes and M. Mohri, “Learning non-linear combinations of kernels Advances in Neural Infor-mation Processing Systems 22,” in *23rd Annual Conference on Neural Information Processing Systems*, Vancouver, British Columbia, Canada, 2009.

[27] G. Mehmet and E. Alpaydin, “Localized multiple kernel learning,” in *Proceedings of the 25th international conference on Machine learning*, pp. 352–359, Helsinki, Finland, 2008.

[28] C. Nello and S. John, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, 2000.

[29] Z. L. Xu, R. Jin, C. Stephane, H. Yang, I. King, and M. R. Lyu, "Simple and efficient multiple kernel learning by group lasso," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 1175–1182, Haifa, Israel, 2010.

[30] C. Corinna, M. Mehryrar, and R. Afshin, "Two-stage learning kernel algorithms," in *Proceedings of the 27th Annual International Conference on Machine Learning (ICML 2010)*, pp. 239–246, Haifa, Israel, June 21-24, 2010.

[31] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "Simple MKL," *The Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2491–2521, 2008.

[32] F. Aiolli and M. Donini, "EasyMKL: a scalable multiple kernel learning algorithm," *Neurocomputing*, vol. 169, pp. 215–224, 2015.