General Dimensional Multiple-Output Support Vector Regressions and Their Multiple Kernel Learning

Wooyong Chung, Jisu Kim, Heejin Lee, and Euntai Kim, *Member, IEEE*

***Abstract*—Support vector regression has been considered as one of the most important regression or function approximation methodologies in a variety of fields. In this paper, two new general dimensional multiple output support vector regressions (MSVRs) named SOCPL1 and SOCPL2 are proposed. The proposed methods are formulated in the dual space and their relationship with the previous works is clearly investigated. Further, the proposed MSVRs are extended into the multiple kernel learning and their training is implemented by the off-the-shelf convex optimization tools. The proposed MSVRs are applied to benchmark problems and their performances are compared with those of the previous methods in the experimental section.**

***Index Terms*—Convex optimization, dual space, multiple kernel learning (MKL), multiple output, support vector regression (SVR).**

I. Introduction

OVER THE past decade, support vector regression (SVR) has become one of the most popular regression techniques [1] and has been employed in a variety of applications, which range from engineering [2]–[5] and science [6] to economy [7]. Despite the success in various fields, however, the use of the SVR is limited because the standard SVR has a single output and does not fit well with multiple output regression problems such as time series prediction [8], localization in wireless sensor networks [3], [9], pose estimation in computer vision [10], robot control and identification [11], airfoil design [12], and biological data prediction [13].

To solve the problem, two research directions were reported about the multiple-output SVRs (MSVRs): vectorial frameworks [9], [14], [15] and division algebraic frameworks [16]–[18]. Unfortunately, however, both directions have their own problems.

Manuscript received March 18, 2014; revised August 13, 2014 and November 7, 2014; accepted November 12, 2014. Date of publication December 19, 2014; date of current version October 13, 2015. This work was supported by the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science, and Technology under Grant NRF-2013R1A2A2A01015624. This paper was recommended by Associate Editor B. Zhang.

W. Chung, J. Kim, and E. Kim are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Korea (e-mail: etkim@yonsei.ac.kr).

H. Lee is with the Department of Information and Control Engineering, Hankyong National University, Anseong 456-749, Korea.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2377016

First, the vectorial MSVRs are general dimensional [9], [14], [15] and can be applied to any system with an arbitrary number of outputs. However, their associated dual problems are not clearly known and the MSVR should be trained in the primal space. In particular, [14], [15] do not have closed-form solution and relies on Taylor approximation for the optimization, thereby degrading the approximation performance

Second, the division algebraic approaches were reported by Shilton *et al.* [16]–[18]. The division algebraic approaches use a complex, a quaternion, or an octonion to represent multiple outputs and they were named as $\varepsilon_{\mathbb{C}}$-SVR, $\varepsilon_{\mathbb{H}}$-SVR, and $\varepsilon_{\mathbb{O}}$-SVR in [16]–[18], respectively, and briefly $\varepsilon_{\mathbb{X}}$-SVR. However, the output dimension of the $\varepsilon_{\mathbb{X}}$-SVR is not general; it depends on the selected division algebra (whether it is a complex, a quaternion, or an octonion). A thorough knowledge of division algebra is required and use with popular off-the-shelf convex optimization tools such as CVX [19] is not straightforward. Furthermore, to the best of our knowledge, the number of outputs cannot exceed nine even when the $\varepsilon_{\mathbb{O}}$-SVR is selected.

On the other hand, multiple kernel learning (MKL) has received attention within the machine learning field due to its capability of addressing the difficulty in kernel choice in kernel machines. To our knowledge, however, no previous research has investigated MKL in MSVRs. The reason for that is that MKL is formulated in the dual space, while the vectorial MSVRs do not have dual space representations [9], [14], [15]. Division algebraic $\varepsilon_{\mathbb{X}}$-SVRs have dual representations but division algebra makes the use of MKL problematic both in theory and in implementation.

In this paper, two new vectorial MSVRs and their MKL versions are presented. The proposed MSVRs are general in that they are trained in dual space and can be applied to problems with an arbitrary number of outputs. The proposed MSVRs and their MKL versions are all formulated in convex optimization problems and are trained by off-the-shelf convex optimization tools. The specific contribution of this paper includes the following.

1) The proposal of general dimensional MSVRs.
2) The development of MKL versions which are trained by off-the-shelf convex optimization tools.
3) Investigation of the relationship between the proposed MSVR and the previous division algebraic $\varepsilon_{\mathbb{X}}$-SVR.

TABLE I
NOMENCLATURE

Notations	Descriptions
N_X	Number of inputs
N_O	Number of outputs
N_φ	Dimension of the feature vector $\boldsymbol{\varphi}(\cdot)$
N_K	Number of kernels
M	Number of training samples
$\mathbf{K}=\left[\kappa\left(\mathbf{x}_i,\mathbf{x}_j\right)\right]\in\mathbb{R}^{M\times M}$	Kernel matrix
$\mathbf{K}_m=\left[\kappa_m\left(\mathbf{x}_i,\mathbf{x}_j\right)\right]\in\mathbb{R}^{M\times M}$	the m th kernel matrix
$\mathbf{e}_a$	a column vector of ones with the length of a
$\mathbf{I}_{a\times a}$	an identity matrix with the size $a\times a$
$\mathrm{Re}(\bullet)$	a real part of a complex (or a quaternion or an octonion)
$\mathbb{R}$	a set of reals
$\mathbb{C}$	a set of complexes
$\mathbb{H}$	a set of quaternions
$\mathbb{O}$	a set of octonions
$\mathbf{D}_m$	weight matrix for the the m th kernel matrix
$\mathcal{D}_{N_O}$	a set of diagonal matrices with the size of $N_O\times N_O$.

The remainder of this paper is organized as follows. In Section II, the multiple output regression problem is formulated and the previous works are summarized including the $\varepsilon_{\mathbb{X}}$-SVR. In Section III, two vectorial MSVRs are proposed and their relationship with the $\varepsilon_{\mathbb{X}}$-SVR is investigated. In Section IV, the MKL versions are developed. In Section V, the results of the proposed methods are compared with those of the previous works. Finally, the conclusion is drawn in Section VI.

II. Preliminary Fundamentals

A. Problem Formulation

The notation used in this paper is summarized in Table I. First, let us assume that a training data set $\mathbf{S}=\{(\mathbf{x}_i,\mathbf{y}_i)|\mathbf{x}_i\in\mathbb{R}^{N_X},\mathbf{y}_i\in\mathbb{R}^{N_O},i=1,\ldots,M\}$ is given where $\mathbf{x}_i$ and $\mathbf{y}_i$ denote an independent input vector and its associated target vector, respectively; N_X and N_O are the dimensions of the input and target vectors, respectively, and M denotes the number of training samples. The goal of the problem is to find a nonlinear function

$$\mathbf{f}(\mathbf{x}_i)=\mathbf{W}^T\boldsymbol{\varphi}(\mathbf{x}_i)+\mathbf{b} \tag{1}$$

such that the function $\mathbf{f}(\cdot)$ approximates the target values as well as possible, that is

$$\mathbf{y}_i\approx\mathbf{f}(\mathbf{x}_i) \tag{2}$$

where a nonlinear function $\boldsymbol{\varphi}(\cdot):\mathbb{R}^{N_X}\rightarrow\mathbb{R}^{N_\varphi}$ maps an input vector to a higher dimensional vector in a feature space (usually, $N_\varphi\gg N_X$); $\mathbf{W}=[\,\mathbf{w}_1\;\mathbf{w}_2\cdots\mathbf{w}_{N_O}\,]\in\mathbb{R}^{N_\varphi\times N_O}$ is the weight matrix and $\mathbf{b}=[\,b_1\;b_2\cdots b_{N_O}\,]^T\in\mathbb{R}^{N_O\times 1}$ is the bias vector for the above multiple output regression. The use of the nonlinear function $\boldsymbol{\varphi}(\cdot)$ is justified by Mercer's Theorem [16]. Here, it should be noted that $N_O\geq 1$, unlike many other research works. The nonlinear function $\mathbf{f}(\cdot)$ can be trained by minimizing the following objective function:

$$\min_{w_n,\mathbf{b}}\frac{1}{2}\sum_{n=1}^{N_O}\mathbf{w}_n^T\mathbf{w}_n+C\sum_{i=1}^{M}\ell\left(\mathbf{x}_i,\mathbf{y}_i\right) \tag{3}$$

where $\ell(\mathbf{x}_i,\mathbf{y}_i)$ denotes the loss function for a data point $(\mathbf{x}_i,\mathbf{y}_i)$. In the objective function, the first-term denotes the regularization and the second-term denotes the empirical risk. The design constant C controls the trade-off between the regularization $1/2\sum_{n=1}^{N}\mathbf{w}_n^T\mathbf{w}_n$ and the empirical risk $\sum_{i=1}^{M}\ell(\mathbf{x}_i,\mathbf{y}_i)$. Throughout this paper, n will be used as an index for the output dimension and i and j will be used as indices for the data points.

B. Previous Works

Previous works of the single-output or MSVRs are summarized in Table II. The most basic version is a single output ε-SVR [20]. When the multiple output is concerned, vectorial MSVR was reported in [9], [14], and [15]. However, as noted in the table, vectorial MSVRs are trained only in the primal space and do not have dual forms. Thus, they cannot be extended into MKL. Another approach is division algebraic $\varepsilon_{\mathbb{X}}$-SVRs [16]–[18].$\varepsilon_{\mathbb{C}}$-SVR [16], ε_H-SVR [17], and $\varepsilon_{\mathbb{O}}$-SVR [18] can be actually thought of 2-D, 4-D, and 8-D extensions of the ε-SVR [20], respectively. The two SVRs above based on division algebra demonstrate good approximation performance [16], [17], but are also difficult to apply to regression problems with an arbitrary number of outputs. Furthermore, they have dual forms, but their corresponding MKLs have yet to be reported.

Structured output SVR methods have also been reported in [21]–[23]. Structured output regressions, however, are completely different from the MSVRs considered herein in that structured output regressions utilize the correlation between outputs and receive other outputs as inputs. Thus, each output regression has a different set of inputs. Therefore, as far as learning theory is concerned, structured output regressions are actually multiple applications of a single output regression. In a MSVR, however, each output shares input vectors with the other outputs and learning theory can exploit this property.

III. MSVRS

In this section, two different MSVR methods are formulated and their dual representations are derived. The proposed MSVRs are general dimensional in that it can be applied to a regression problem with an arbitrary number of outputs. In the proposed methods, a single L_2 constraint

$$\left\|\mathbf{y}_i-\mathbf{W}^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right)-\mathbf{b}\right\|_2\leq\varepsilon \tag{4}$$

is imposed on a training sample as in [14] and [15]. If the constraint is violated, the training sample is penalized and two kinds of penalties are considered: ε-insensitive linear and

TABLE II
SUMMARY OF PREVIOUS WORKS

SVR	Training and trained machine	Number of outputs (N)	Training space	MKL	Comment
ε-SVR [20]	Training $d^* = \max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}} \left[-\frac{1}{2}\left(\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}\right)^T \mathbf{K}\left(\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}\right) - \varepsilon\left(\boldsymbol{\alpha}+\hat{\boldsymbol{\alpha}}\right)^T \mathbf{e}_M + \left(\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}\right)^T \mathbf{y} \right]$ sub to $\begin{cases} \mathbf{0}_M \le \boldsymbol{\alpha} \le C\mathbf{e}_M \\ \mathbf{0}_M \le \hat{\boldsymbol{\alpha}} \le C\mathbf{e}_M \\ \mathbf{e}_M^T\left(\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}\right) = 0 \end{cases}$, Trained machine $f = \sum_{i=1}^{M} \left(\alpha_i - \hat{\alpha}_i\right)\kappa\left(\mathbf{x},\mathbf{x}_i\right) + b \in \mathbb{R}$	1	Dual space	Yes	$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_M \end{bmatrix}^T \in \mathbb{R}^{M\times 1}$ is a target vector $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_M \end{bmatrix}^T \in \mathbb{R}^{M\times 1}$ $\hat{\boldsymbol{\alpha}} = \begin{bmatrix} \hat{\alpha}_1 & \hat{\alpha}_2 & \cdots & \hat{\alpha}_M \end{bmatrix}^T \in \mathbb{R}^{M\times 1}$ are dual variables (Lagrange multipliers)
$\varepsilon_{\mathbb{X}}$-SVR $\mathbb{X} = \{\mathbb{C},\mathbb{H},\mathbb{O}\}$ [16-18]	Training $d^* = \max_{\bar{\boldsymbol{\alpha}}} \left[-\frac{1}{2}\bar{\boldsymbol{\alpha}}^{\dagger}\mathbf{K}\bar{\boldsymbol{\alpha}} - \varepsilon\sum_{i=1}^{M}\left|\bar{\alpha}_i\right| + \mathrm{Re}\left(\bar{\boldsymbol{\alpha}}^{\dagger}\bar{\mathbf{y}}\right) \right]$ sub to $\begin{cases} \left|\bar{\alpha}_i\right| \le C \\ \mathbf{e}_M^T\bar{\boldsymbol{\alpha}} = 0 \end{cases}$ Trained machine $f = \sum_{i=1}^{M} \bar{\alpha}_i\kappa\left(\mathbf{x},\mathbf{x}_i\right) + b \in \mathbb{X}$	if $\mathbb{X}=\mathbb{C}$ $N \le 2$ if $\mathbb{X}=\mathbb{H}$ $N \le 4$ if $\mathbb{X}=\mathbb{O}$ $N \le 8$	Dual space	No	$\bar{\mathbf{y}} = \begin{bmatrix} \bar{y}_1 & \bar{y}_2 & \cdots & \bar{y}_M \end{bmatrix}^T \in \mathbb{X}^{M\times 1}$ is a target vector $\bar{\boldsymbol{\alpha}} = \begin{bmatrix} \bar{\alpha}_1 & \bar{\alpha}_2 & \cdots & \bar{\alpha}_M \end{bmatrix}^T \in \mathbb{X}^{M\times 1}$ is a dual variable (Lagrange multipliers) . Three $\varepsilon_{\mathbb{X}}$-SVRs have the same training rules. The only difference among them is whether the variables are complexes, quaternions or octonions. † is a conjugate transpose.
Pérez-Cruz *et al.*'s [14,15]	Training ($n = 1,\cdots,N_O$) $\begin{bmatrix} \mathbf{K}+\mathbf{D}_a^{-1} & \mathbf{1} \\ \mathbf{a}^T\mathbf{K} & \mathbf{1}^T\mathbf{a} \end{bmatrix}\begin{bmatrix} \boldsymbol{\beta}^n \\ b^n \end{bmatrix} = \begin{bmatrix} \mathbf{y}^n \\ \mathbf{a}^T\mathbf{y}^n \end{bmatrix}$ Trained machine $\mathbf{f} = \boldsymbol{\beta}^T\boldsymbol{\kappa}\left(\mathbf{x}\right) + \mathbf{b} \in \mathbb{R}^{N_O}$	arbitrary	Primal space	No	$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}^1 & \boldsymbol{\beta}^2 & \cdots & \boldsymbol{\beta}^{N_O} \end{bmatrix} \in \mathbb{R}^{M\times N_O}$ $\boldsymbol{\kappa}\left(\mathbf{x}\right) = \begin{bmatrix} \kappa\left(\mathbf{x},\mathbf{x}_1\right) & \kappa\left(\mathbf{x},\mathbf{x}_2\right) & \cdots & \kappa\left(\mathbf{x},\mathbf{x}_M\right) \end{bmatrix}^T$ $\mathbf{b} = \begin{bmatrix} b^1 & b^2 & \cdots & b^N \end{bmatrix}^T$ $\mathbf{y}^n$ is a vector composed of the n th components of the samples.
Lee *et al.*'s [9]	Training $p^* = \min_{\boldsymbol{\beta}^n,\boldsymbol{\xi}} \left[\lambda\sum_{n=1}^{N_O}\frac{1}{2}\boldsymbol{\beta}^{nT}\mathbf{K}\boldsymbol{\beta}^n + \boldsymbol{\xi}^T\boldsymbol{\xi} \right]$ sub to $\begin{cases} \left\| \mathbf{y}_i - \boldsymbol{\beta}^T\boldsymbol{\kappa}\left(\mathbf{x}_i\right) - \mathbf{b} \right\| \le \varepsilon + \xi_i \\ \xi_i \ge 0 \end{cases}$ Trained machine $\mathbf{f} = \boldsymbol{\beta}^T\boldsymbol{\kappa}\left(\mathbf{x}\right) + \mathbf{b} \in \mathbb{R}^{N_O}$	arbitrary	Primal space	No	$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}^1 & \boldsymbol{\beta}^2 & \cdots & \boldsymbol{\beta}^{N_O} \end{bmatrix} \in \mathbb{R}^{M\times N_O}$ $\boldsymbol{\kappa}\left(\mathbf{x}\right) = \begin{bmatrix} \kappa\left(\mathbf{x},\mathbf{x}_1\right) & \kappa\left(\mathbf{x},\mathbf{x}_2\right) & \cdots & \kappa\left(\mathbf{x},\mathbf{x}_M\right) \end{bmatrix}^T$ $\mathbf{b} = \begin{bmatrix} b^1 & b^2 & \cdots & b^N \end{bmatrix}^T$

The column of "MKL" means whether the associated MKL has been developed. N_X = the dimension of the input, N_O = the dimension of the output, M = the number of training samples, N_φ = the dimension of the feature vector $\boldsymbol{\varphi}(\cdot)$, $\mathbf{K}$ =kernel matrix, $\mathbf{e}_a$ = a column vector of ones with the length of a ;.

second-order penalties. The combination of the constraint (4) and the two penalties fall into convex optimization framework.

A. Second-Order Cone Program With Linear Loss (SOCPL1)

The first method uses the constraint (4) and the ε-insensitive linear loss, as shown in Fig. 1. Unlike the multiple applications of the independent SVRs where multiple element-wise constraints are given, a single L_2 constraint is imposed on a training sample and the sample is penalized by ε-insensitive linear penalty. It can be formulated as

$$\begin{aligned} p^* = \min_{\mathbf{W},\mathbf{b},\xi} \frac{1}{2}Tr\left(\mathbf{W}\mathbf{W}^T\right) + C\sum_{i=1}^{M}\xi_i \\ \text{sub to } \left\|\mathbf{y}_i - W^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right) - \mathbf{b}\right\| \le \varepsilon + \xi_i \\ \xi_i \ge 0 \end{aligned} \tag{5}$$

where p^* is the primal optimal value and $\xi_i \in \mathbb{R}_+$. $\mathbf{W}$ and $\mathbf{b}$ are given in (1). The corresponding dual problem is formulated

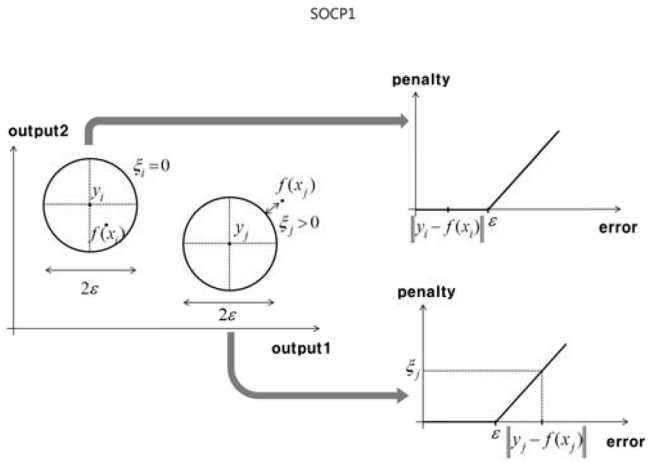

Fig. 1. L_2 constraint is imposed on each data point and the constraint is penalized in a nonelement-wise manner by a linear loss function. Thus, when the estimate $\mathbf{f}(\mathbf{x}_i)$ lies within the L_2 ε-tube around $\mathbf{y}_i$, the ith data point is not penalized. Instead, when the estimate $\mathbf{f}(\mathbf{x}_j)$ lies outside of $\mathbf{L}_2$ ε-tube around $\mathbf{y}_j$, the jth data point is penalized by a linear loss function.

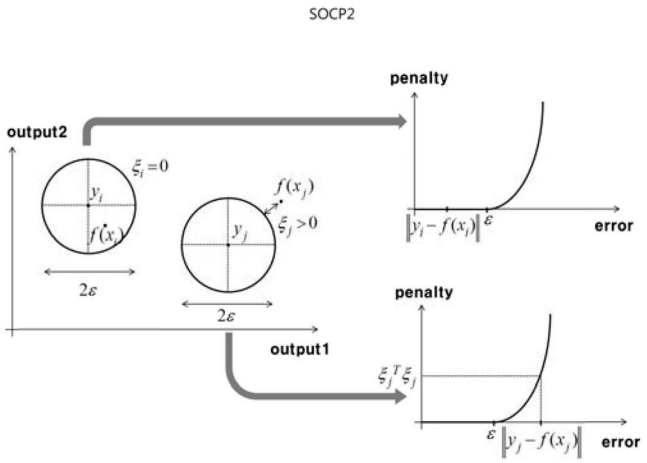

Fig. 2. L_2 constraint is imposed on each data point and the constraint is penalized in a nonelement-wise manner by a second-order loss function. Thus, when the estimate $\mathbf{f}(\mathbf{x}_i)$ lies within the L_2 ε-tube around $\mathbf{y}_i$, the ith data point is not penalized. Instead, when the estimate $\mathbf{f}(\mathbf{x}_j)$ lies outside of L_2 ε-tube around $\mathbf{y}_j$, the jth data point is penalized by a second-order loss function.

as

$$d^* = \max_{\mathbf{u},\boldsymbol{\alpha}} \left[-\frac{1}{2}\mathbf{u}^T \left(\mathbf{K} \otimes \mathbf{I}_{N_O \times N_O}\right) \mathbf{u} + \mathbf{u}^T \mathbf{Y} - \varepsilon \boldsymbol{\alpha}^T \mathbf{e}_M \right]$$
$$\text{sub to} \begin{cases} \mathbf{0} \le \boldsymbol{\alpha} \le C\mathbf{e}_M \\ \|\mathbf{u}_i\| \le \alpha_i \\ \left(\mathbf{e}_M^T \otimes \mathbf{I}_{N_O \times N_O}\right) \mathbf{u} = 0 \end{cases} \tag{6}$$

where d^* is the dual optimal value; $\otimes$ is a Kronecker product; $\mathbf{K} = [\kappa(\mathbf{x}_i, \mathbf{x}_j)] \in \mathbb{R}^{M \times M}$ is a kernel matrix; $\mathbf{I}_{N_O \times N_O}$ is an identity matrix with the size $N_O \times N_O$, and $\mathbf{e}_M$ is a column vector of ones with the length of M;

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_M \end{bmatrix}^T \in \mathbb{R}^{M \times 1}$$
$$\mathbf{u}_i = \begin{bmatrix} u_{i1} & u_{i2} & \cdots & u_{iN_O} \end{bmatrix}^T \in \mathbb{R}^{N_O \times 1}$$
$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_1^T & \mathbf{u}_2^T & \cdots & \mathbf{u}_M^T \end{bmatrix}^T \in \mathbb{R}^{MN_O \times 1}$$

and

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^T & \mathbf{y}_2^T & \cdots & \mathbf{y}_M^T \end{bmatrix}^T \in \mathbb{R}^{MN_O \times 1}.$$

Then, the trained machine is computed by

$$\mathbf{f}(\mathbf{x}) = \mathbf{W}^T \boldsymbol{\varphi}(\mathbf{x}) + \mathbf{b} = \sum_{i=1}^{M} \mathbf{u}_i \kappa\left(\mathbf{x}_i, \mathbf{x}\right) + \mathbf{b} \tag{7}$$

where

$$\mathbf{b} = \arg\min_{\mathbf{b}} \sum_{i=1}^{M} \left\| \mathbf{y}_i - \sum_{j=1}^{M} \mathbf{u}_j \boldsymbol{\kappa}\left(\mathbf{x}_i, \mathbf{x}_j\right) - \mathbf{b} \right\|. \tag{8}$$

The detailed derivation for the dual formulation is given in the Appendix A.

Remark 1.

1) Unlike the previous SVRs, the bias term $\mathbf{b}$ is computed by another optimization problem (8) and it improves the regression accuracy.
2) Both the primal and dual problems belong to the SOCP problem and that is the reason why this problem formulation is named as SOCPL1 (SOCP with the first-order loss function).
3) Since a single constraint is imposed on a training data point via L2 norm, the penalty is independent of the choice of coordinate and the coupling among the outputs are fully considered.
4) It is interesting to note that the SOCPL1 includes the $\varepsilon_{\mathbb{X}}$-SVR as a special case and it is clearly stated in Remark 1e. So, the proposed MSVR can be applied to any dimensional problem while the application of the previous $\varepsilon_{\mathbb{X}}$-SVR is limited.
5) The SOCPL1 represented by (5) or (6) includes *$\varepsilon_{\mathbb{X}}$-SVR* represented by the second row in Table II as a special case. The detailed proof is given in the Appendix B.
6) If the proposed MSVR is compared with the multiple use of single-output ε-SVR, the evaluation of the MSVR is N_O times faster than that of the multiple uses of ε-SVR because the outputs share the same kernel in the MSVR.

B. Second-Order Cone Program With the Second-Order Loss (SOCPL2)

The second method combines the constraint (4) and the ε-insensitive second-order loss. The training of the second method can be formulated as

$$p^* = \min_{\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \frac{1}{2} Tr\left(\mathbf{W}\mathbf{W}^T\right) + \frac{C}{2} \sum_{i=1}^{M} \xi_i^2$$
$$\text{sub to } \left\| \mathbf{y}_i - \mathbf{W}^T \boldsymbol{\varphi}\left(\mathbf{x}_i\right) - \mathbf{b} \right\| \le \varepsilon + \xi_i \tag{9}$$

where p^*, $\mathbf{W}$, and $\mathbf{b}$ are the same as in (1). When $N_O = 2$, SOCPL2 is explained in Fig. 2. The corresponding dual problem is formulated as

$$d^* = \max_{\mathbf{u},\boldsymbol{\alpha}} \left[-\frac{1}{2}\mathbf{u}^T \left(\mathbf{K} \otimes \mathbf{I}_{N \times N}\right) \mathbf{u} + \mathbf{u}^T \mathbf{Y} - \frac{1}{2C}\boldsymbol{\alpha}^T \boldsymbol{\alpha} - \varepsilon \boldsymbol{\alpha}^T \mathbf{e}_M \right]$$
$$\text{sub to} \begin{cases} \boldsymbol{\alpha} \ge \mathbf{0}_M \\ \|\mathbf{u}_i\| \le \alpha_i \\ \left(\mathbf{e}_M^T \otimes \mathbf{I}_{N \times N}\right) \mathbf{u} = \mathbf{0}_N \end{cases} \tag{10}$$

where the definitions of d^*, $\mathbf{K}$, $\boldsymbol{\alpha}$, $\mathbf{u}$, and $\mathbf{Y}$ are given in (6). Then, the trained machine is computed by

$$\mathbf{f}(\mathbf{x}) = \mathbf{W}^T\boldsymbol{\varphi}(\mathbf{x}) + \mathbf{b} = \sum_{i=1}^{M} \mathbf{u}_i \kappa\left(\mathbf{x}_i, \mathbf{x}\right) + \mathbf{b} \tag{11}$$

where

$$\mathbf{b} = \arg\min_{\mathbf{b}} \sum_{i=1}^{M} \left(\left\| \mathbf{y}_i - \sum_{j=1}^{M} \mathbf{u}_j \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) - \mathbf{b} \right\| \right). \tag{12}$$

The detailed derivation is given in the Appendix C.

Remark 2.

1) As in SOCPL1, the bias term $\mathbf{b}$ is solved by another optimization problem (12).
2) Both the primal and dual problems belong to the SOCP problem and this formulation is named as SOCPL2.

IV. MKL for MSVR

In this section, the MKL versions of the proposed MSVRs are developed. Here, we represent our target kernel with a linear combination of multiple kernels by

$$\mathbf{K} = \sum_{m=1}^{N_k} \mathbf{D}_m \mathbf{K}_m$$
$$\sum_{m} \mathbf{D}_m = \mathbf{I}_{N_O}, \mathbf{D}_m \succeq \mathbf{0} \tag{13}$$

where N_k is the number of kernels, $\mathbf{K}_m$ is the mth kernel matrix with an element $\kappa_m(\cdot,\cdot)$, $\mathbf{D}_m$ is the combination coefficient. $\succeq$ denotes the positive definiteness. Unlike [24]–[26], $\mathbf{D}_m$ is assumed to be not a scalar but a diagonal matrix, allowing each regression output has different preference for the kernels.

A. Second-Order Cone Program With Linear Loss (SOCPL1)-MKL

Following the idea of [25], the SOCPL1-MKL can be formulated as

$$p^* = \min_{\mathbf{D},\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \frac{1}{2}\sum_{m=1}^{N_k} Tr\left(\mathbf{W}_m\mathbf{D}_m^{-1}\mathbf{W}_m^T\right) + C\sum_{i=1}^{M}\xi_i$$
$$\text{sub to} \begin{cases} \left\| \mathbf{y}_i - \sum_{m=1}^{N_k} \mathbf{W}_m^T\boldsymbol{\varphi}_m\left(\mathbf{x}_i\right) - \mathbf{b} \right\| \le \varepsilon + \xi_i \\ \xi_i \ge 0 \\ \sum_{m=1}^{N_k} \mathbf{D}_m = \mathbf{I}_{N_O}, \mathbf{D}_m \succeq \mathbf{0}, \mathbf{D}_m \in \mathcal{D}_{N_O} \end{cases} \tag{14}$$

in the primal space, where $\mathcal{D}_{N_O}$ is a set of diagonal matrices with the size of $N_O \times N_O$. The corresponding dual problem is formulated as

$$d^* = \max_{\boldsymbol{\alpha},\mathbf{u}_i,\boldsymbol{\Lambda}} \left[\mathbf{u}^T\mathbf{Y} - \varepsilon\boldsymbol{\alpha}^T\mathbf{e}_M - Tr\boldsymbol{\Lambda}\right]$$
$$\text{sub to} \begin{cases} 0 \le \alpha_i \le C \\ \|\mathbf{u}_i\| \le \alpha_i \\ \sum_{i=1}^{M} \mathbf{u}_i = 0 \\ \boldsymbol{\Lambda} \succeq \mathbf{0},\ \boldsymbol{\Lambda} \in \mathcal{D}_{N_O} \\ \boldsymbol{\Lambda} \ge \frac{1}{2} diag\left(\sum_{i=1}^{M}\sum_{j=1}^{M} \kappa_m\left(\mathbf{x}_i, \mathbf{x}_j\right)\mathbf{u}_i\mathbf{u}_j^T\right) \end{cases} \tag{15}$$

where the definitions of d^*, $\boldsymbol{\alpha}$, $\mathbf{u}$, and $\mathbf{Y}$ are the same as in (6). $diag(\cdot)$ returns a diagonal matrix the diagonal elements of which are the same as those of the argument matrix but the off-diagonal elements of which are zeros. Then, the trained machine is computed by

$$\mathbf{f}(\mathbf{x}) = \sum_{m=1}^{N_k} \mathbf{W}_m^T\boldsymbol{\varphi}_m(\mathbf{x}) + \mathbf{b} = \sum_{i=1}^{M}\sum_{m=1}^{N_k} \mathbf{D}_m\boldsymbol{\kappa}_m\left(\mathbf{x}_i, \mathbf{x}\right)\mathbf{u}_i + \mathbf{b} \tag{16}$$

where

$$\mathbf{b} = \arg\min_{\mathbf{b}} \sum_{i=1}^{M} \left(\left\| \mathbf{y}_i - \sum_{j=1}^{M}\sum_{m=1}^{N_k} \mathbf{D}_m\boldsymbol{\kappa}_m\left(\mathbf{x}_j, \mathbf{x}_i\right)\mathbf{u}_j - \mathbf{b} \right\| \right). \tag{17}$$

The detailed derivation for the dual formulation is given in the Appendix D.

Remark 3.

1) If we compare (6) with (15), the only difference between SOCPL1 and SOCPL1-MKL is that $-1/2\mathbf{u}^T(\mathbf{K} \otimes \mathbf{I}_{N_O\times N_O})\mathbf{u}$ in (6) is replaced with $-Tr\boldsymbol{\Lambda}$ with another constraint $\boldsymbol{\Lambda} \ge 1/2 diag(\sum_{i=1}^{M}\sum_{j=1}^{M}\kappa_m(\mathbf{x}_i,\mathbf{x}_j)\mathbf{u}_i\mathbf{u}_j^T)$. We can see that when we have only one kernel $N_k = 1$, two methods become the same.
2) The CVX codes of SOCPL1 and its MKL version can be downloaded from http://cilab.yonsei.ac.kr/pubs/DB/source_code.zip.

B. Second-Order Cone Program With the Second-Order Loss (SOCPL2)-MKL

As in SOCPL1-MKL (14), the SOCPL2-MKL can be formulated as

$$p^* = \min_{\mathbf{D},\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \frac{1}{2}\sum_{m=1}^{N_k} Tr\left(\mathbf{W}_m\mathbf{D}_m^{-1}\mathbf{W}_m^T\right) + C\sum_{i=1}^{M}\xi_i$$
$$\text{subto} \begin{cases} \left\| \mathbf{y}_i - \sum_{m=1}^{N_k} \mathbf{W}_m^T"\boldsymbol{\phi}_m\left(\mathbf{x}_i\right) - \mathbf{b} \right\| \le \varepsilon + \xi_i \\ \xi_i \ge 0 \\ \sum_{m=1}^{N_k} \mathbf{D}_m = \mathbf{I}_{N_O}, \mathbf{D}_m \succeq \mathbf{0}, \mathbf{D}_m \in \mathcal{D}_{N_O} \end{cases} \tag{18}$$

$$p^* = \min_{\mathbf{D},\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \frac{1}{2}\sum_{m=1}^{N_k} Tr\left(\mathbf{W}_m\mathbf{D}_m^{-1}\mathbf{W}_m^T\right) + \frac{C}{2}\sum_{i=1}^{M}\xi_i^2$$
$$\text{sub to} \begin{cases} \left\| \mathbf{y}_i - \sum_{m=1}^{N_k} \mathbf{W}_m^T\boldsymbol{\varphi}_m\left(\mathbf{x}_i\right) - \mathbf{b} \right\| \le \varepsilon + \xi_i \\ \sum_{m=1}^{N_k} \mathbf{D}_m = \mathbf{I}_{N_O}, \mathbf{D}_m \succeq \mathbf{0} \end{cases} \tag{19}$$

in the primal space. The corresponding dual problem is formulated as

$$d^* = \max_{\boldsymbol{\alpha},\mathbf{u}_i,\boldsymbol{\Lambda}} \left[\mathbf{u}^T\mathbf{Y} - \frac{1}{2C}\boldsymbol{\alpha}^T\boldsymbol{\alpha} - \varepsilon\boldsymbol{\alpha}^T\mathbf{e}_M - Tr\boldsymbol{\Lambda}\right]$$
$$\text{sub to} \begin{cases} \alpha_i \ge 0 \\ \|\mathbf{u}_i\| \le \alpha_i \\ \sum_{i=1}^{M} \mathbf{u}_i = 0 \\ \boldsymbol{\Lambda} \in \mathcal{D}_{N_O} \\ \boldsymbol{\Lambda} \ge \frac{1}{2} diag\left(\sum_{i=1}^{M}\sum_{j=1}^{M} \boldsymbol{\kappa}_m\left(\mathbf{x}_i, \mathbf{x}_j\right)\mathbf{u}_i\mathbf{u}_j^T\right) \end{cases} \tag{20}$$

TABLE III
PARAMETER TUNING FOR EXPERIMENTS AND METHODS

Method	Exp.A	Exp.A (outlier_10%)	Exp.B	Exp.C	Exp.D	Exp.E	Exp.F
MKL_ SOCPL1	$\sigma_G = \{0.25, 0.5, \cdots, 8\}$ $C = \{10^0, 10^1, 10^2, 10^3\}$						
MKL_ SOCPL2							
SOCPL1							
SOCPL2							
Pérez-Cruz's MSVR							
SOAR	$\sigma_G = \{0.25, 0.5, \cdots, 8\}$ $C_1 = \{10^0, 10^1, 10^2, 10^3\}$, $C_2 = \{10^0, 10^1, 10^2, 10^3\}$						
RLS2	$\lambda = \{10^{-2}, \cdots, 10^2\}$, $\sigma_G = \{0.25, 0.5, \cdots, 8\}$						
KNNR	$k = \{5, 7, \cdots, 10\}$						
RBFN	$\sigma_G = \{0.25, 0.5, \cdots, 8\}$, $N_R = \{1, 2, 3\}$						
ANFIS	$N_M = \{2, 3, 4\}$		$N_M = \{2\}$	x			
ELM	$\lambda = \{10^{-2}, \cdots, 10^0\}$						
KELM	$\sigma_G = \{0.25, 0.5, \cdots, 8\}$, $\lambda = \{10^{-4}, \cdots, 10^1\}$						

TABLE IV
MSE FOR EXPERIMENTS AND METHODS (IN EXP. E, MAE IS USED)

Method	Exp.A	Exp.A (outlier_10%)	Exp.B	Exp.C	Exp.D	Exp.E	Exp.F	Mean Performance	Mean Ranking
MKL_ SOCPL1	**0.10338**	**0.59183**	0.36964	0.00527	0.00178	7.11825	0.07033	0.11900	4.85714
MKL_ SOCPL2	0.10638	0.60508	0.28442	0.00524	0.00178	**7.10720**	0.06916	0.10401	5.71429
SOCPL1	0.10404	0.59234	0.36958	0.00394	**0.00130**	10.64034	**0.03750**	**0.08572**	**3.28571**
SOCPL2	0.10636	0.60361	0.28262	**0.00372**	0.00288	10.65066	0.06916	0.09035	6.00000
Pérez-Cruz's MSVR	0.10730	0.60684	0.26765	0.00605	0.00230	10.62859	0.08161	0.15302	7.42857
SOAR	0.10810	0.59920	**0.17510**	0.00580	0.00500	10.72830	0.06770	0.14664	6.00000
RLS2	0.10767	0.60027	0.23813	0.00621	0.01062	33.01333	0.06078	0.37161	7.28571
KNNR	0.11533	0.67936	0.22362	0.00545	0.00175	10.60216	0.06325	0.17282	5.57143
RBFN	0.12448	0.60350	0.19760	0.00502	0.01168	34.38394	0.05508	0.35743	6.71429
ANFIS	0.32710	0.77270	0.63980	x	x	x	x	1.00000	12.00000
ELM	0.25774	0.71484	0.19199	0.00545	0.00177	17.23490	0.06364	0.31714	7.00000
KELM	0.10479	0.59943	0.18986	0.00847	0.00560	11.49787	0.06468	0.24110	6.14286

and the trained machine is computed by

$$\mathbf{f}(\mathbf{x}) = \sum_{m=1}^{N_k} \mathbf{W}_m^T \boldsymbol{\varphi}_m(\mathbf{x}) + \mathbf{b} = \sum_{i=1}^{M} \sum_{m=1}^{N_k} \mathbf{D}_m \boldsymbol{\kappa}_m(\mathbf{x}_i, \mathbf{x})\, \mathbf{u}_i + \mathbf{b} \tag{21}$$

where

$$\mathbf{b} = \arg\min_b \sum_{i=1}^{M} \left(\left\| \mathbf{y}_i - \sum_{j=1}^{M} \sum_{m=1}^{N_k} \mathbf{D}_m \boldsymbol{\kappa}_m(\mathbf{x}_j, \mathbf{x}_i)\, \mathbf{u}_j - \mathbf{b} \right\| \right). \tag{22}$$

The detailed derivation for the dual formulation is the combination of the Appendixes C and D and it is omitted here. Remark 3 also applies here.

V. EXPERIMENTAL RESULTS

In this section, SOCPL1, SOCPL2, and their MKL versions are applied to six benchmark problems and their performance is compared with that of previous methods: Pérez-Cruz's MSVR [14], [15] regularized least squares with two layers (RLS2) [27], k-nearest neighbor regression (KNNR) [28], radial basis function network (RBFN) [29], adaptive neuro fuzzy inference system (ANFIS) [30], extreme learning machines (ELM) [31], Kernel-based ELM (KELM) [32], and structured output-associative regression (SOAR) [21]. The four proposed methods, Pérez-Cruz's MSVR, KNNR, and SOAR are implemented by us using MATLAB and CVX [19]. RLS2 code is downloaded from [27] and ELM and KELM codes are downloaded from [33]. The MATLAB neural network toolbox is used for RBFN and the ANFIS Toolbox is used for ANFIS. Six benchmark problems are introduced in turn and the parameters used in

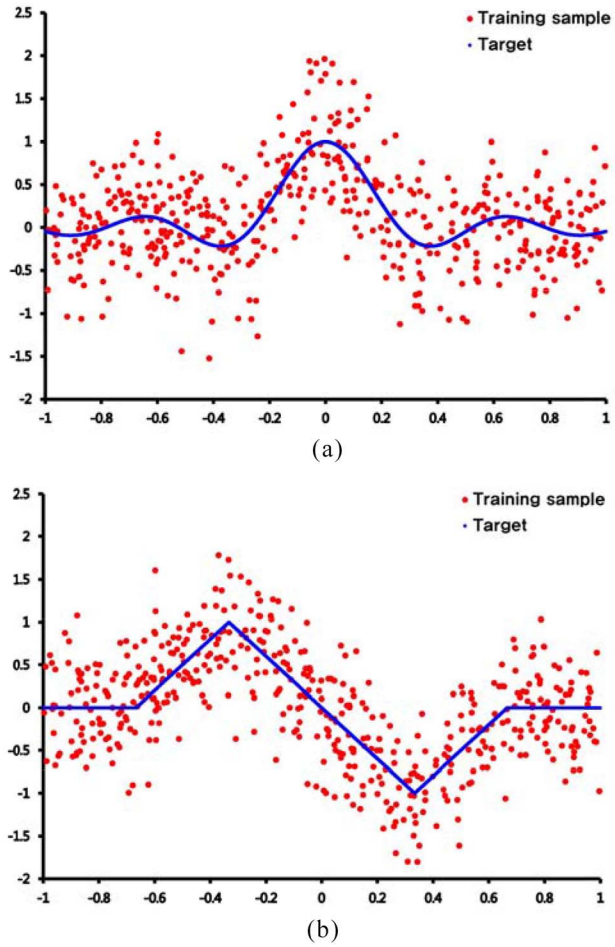

Fig. 3. Target and training sample of complex function regression in (a) real component and (b) imaginary component.

simulation and regression performance are summarized in Tables III and IV, respectively. The parameters of the previous and proposed algorithms are selected such that the best algorithm performance is achieved. Threefold cross validation is performed to compare the proposed and previous algorithms.

A. Complex Function Regression

The first example is taken from [17] and its target system is a complex function represented by

$$\hat{f}(x) = \frac{\sin(12x)}{12x} + i \begin{pmatrix} 0, & if\ x \le -\frac{2}{3} \\ 3x+2, & if\ -\frac{2}{3} \le x \le -\frac{1}{3} \\ -3x, & if\ -\frac{1}{3} \le x \le \frac{1}{3} \\ 3x-2, & if\ \frac{1}{3} \le x \le \frac{2}{3} \\ 0, & if\ x \ge \frac{2}{3} \end{pmatrix} \quad (23)$$

where $i = \sqrt{-1}$. It is actually a single-input, double-output system. Five hundred training samples were generated using (23) by selecting the input variable x_i uniformly over the interval $[-1, 1]$ and adding Gaussian noise with $\sigma^2 = 0.2$ to the target value (23). In Fig. 3, the training samples and the target function (23) are depicted. In the same manner, an additional 500 input data points, x_i^{test}, are selected uniformly from $[-1, 1]$ and we evaluate the regression performance using the mean square error (MSE)

$$\text{MSE} = \frac{1}{500} \sum_{i=1}^{500} \left\{ \hat{f}\left(x_i^{\text{test}}\right) - f\left(x_i^{\text{test}}\right) \right\}^2 \quad (24)$$

where $\hat{f}(x)$ and $f(x)$ are the target function and the approximation, respectively.

Then, outliers are added to the training samples to test the robustness of the algorithms. Some of the training samples (10% of the 500 samples) are randomly selected and the corresponding target values (real and imaginary) are overwritten with values uniformly selected from the interval $[-5, 5]$.

B. Lorenz Dataset

The second target system is the Lorenz attractor and this example is also taken from [34]. The dynamics of the Lorenz attractor is represented by

$$\begin{aligned} \frac{dp_x}{dt} &= \sigma\left(p_y - p_x\right) \\ \frac{dp_y}{dt} &= p_x(r - p_z) - p_y \\ \frac{dp_z}{dt} &= p_x p_y - b p_z \end{aligned} \quad (25)$$

where $x = (p_x, p_y, p_z)^T$ is the state of the attractor and it is initialized by $(0, 0, 1)^T$, $\sigma = 3$, $r = 26.5$, and $b = 1$. $\tau = 0.02\,s$ is selected as the sample rate and a total of 1600 data samples are generated by (25): the 600th to the 999th data points are used as a training set, and the 1000th to the 1166th data samples are used as a test set. The first 599 samples are discarded because they are in the transient state and consequently their behaviors are completely different from those of the remaining samples. All the algorithms are trained to predict the eight-step-ahead state $\mathbf{x}_{t+8\tau}$ based on the current state, $\mathbf{x}_t$, and previous state, $\mathbf{x}_{t-\tau}$. Thus, the target system is a six-input, three-output system. All data points are normalized to have a mean of zero and unit standard deviation. Furthermore, Gaussian noise with $\sigma = 0.5$ is added to the samples.

C. Building 2

This dataset is taken from PROBEN 1 [35], which represents a real-world problem on the prediction of energy consumption in a building. Hourly consumption of electrical energy should be predicted based on the date, the time of day, outside temperature, outside humidity, and solar irradiance, among other factors. This dataset has 14 inputs, three outputs, and 4208 samples, which are divided into the training and test sets. The training set has 2104 samples and the test set has 1052 samples; 300 samples are drawn randomly from each set.

D. Stock Dataset

This dataset is formed from 22 major stock indices from 13 countries (Korea, U.S., Brazil, China, Japan, Hong Kong, Taiwan, India, Indonesia, Malaysia, U.K., France, and Germany). The set is gathered from the internet.

The dataset consists of stock prices for 750 days from Jun. 2011 to Nov. 2013. The goal is to predict tomorrow's 22 stock prices, $p_i(t+1)$, using the current and previous nine days' stock prices, $\{p_i(t-9), p_i(t-8), \ldots, p_i(t)\}$, where $i \in \{Korea, U.S., \ldots\}$. From the previous prices, typical technical stock market indicators [36], are computed as the input for the regression algorithms

$$\begin{aligned}
\mathrm{MA1}_i &= \frac{1}{5}\sum_{t_x=1}^{5} p_i(t-t_x) \\
\mathrm{MA2}_i &= \frac{1}{10}\sum_{t_x=1}^{10} p_i(t-t_x) \\
\mathrm{RSI1}_i &= \frac{NI(p_i(t-5), p_i(t-1))}{NI(p_i(t-5), p_i(t-1)) + DI(p_i(t-5), p_i(t-1))} \\
\mathrm{RSI2}_i &= \frac{NI(p_i(t-10), p_i(t-1))}{NI(p_i(t-10), p_i(t-1)) + DI(p_i(t-10), p_i(t-1))} \\
\mathrm{BIAS}_i &= \frac{|p_i(t-1) - \mathrm{MA2}_i|}{\mathrm{MA2}_i}
\end{aligned} \tag{26}$$

where $NI(p_i(t-t_x), p_i(t-1))$ and $DI(p_i(t-t_x), p_i(t-1))$ denote the number of rises and falls in the stock price, respectively, from time $(t-t_x)$ to $(t-1)$. Thus, the dataset has (22×5) inputs, 22 outputs, and 750 samples. The samples are divided into training and test sets. The training set has 550 samples and the test set has 100 samples. It should be noted that the previous $\varepsilon_{\mathbb{X}}$-SVR cannot be applied to this problem because the number of outputs is 22 and it exceeds 9.

E. Pointing'04 Dataset

This dataset is taken from [10], and is a benchmark for head pose estimation in humans. The set consists of 2790 monocular face images of 15 people that have variations of pitch and yaw angles from -90 to 90 degrees. We use the HOG feature [37] and normalized gray image [38] to estimate the head pose. The HOG feature consists of 3×3 blocks, and each block has nine gradient orientations. The normalized gray image is downsampled to 9×12 pixels. Therefore, the total dimension of the input composed of the HOG and the normalized gray image is 189. The output is the pitch and yaw angles of the head poses. Fig. 4 shows examples of head pose images. Among 15 subjects, five are used for training images and one is used for the test image.

F. Corn Dataset

The corn dataset is also used for multioutput regression [39]. This dataset consists of 80 examples of corn measured on three different NIR spectrometers (m5, mp5, and mp6). The wavelength range is 1100–2498 nm at 2-nm intervals. In this experiment, we use only the m5 instrument and the input has 700 dimensions. The moisture, oil, protein, and starch values represent the output values. The first 40 examples are used as training data. The remaining 20 examples are used as test data.

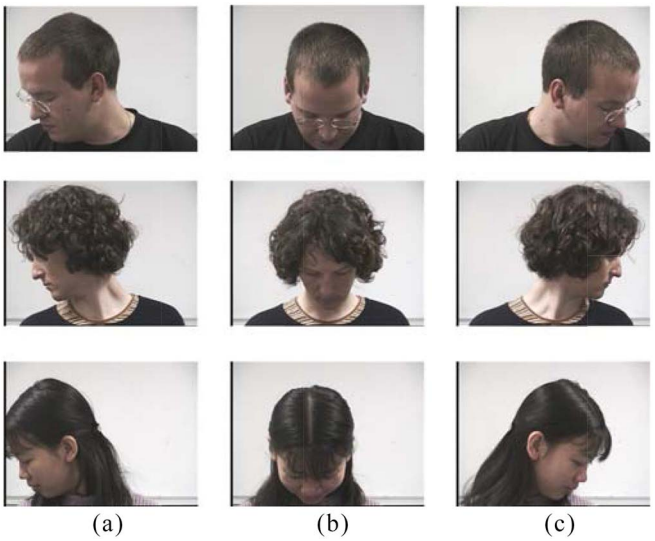

Fig. 4. Various head pose images. (a) Pitch angle: -60, yaw angle: $+90$. (b) Pitch angle: -90, yaw angle: 0. (c) Pitch angle: -60, yaw angle: -90.

G. Summary of the Experimental Results

The parameters used in the previous and proposed methods are summarized in Table III. Pérez-Cruz's MSVR, RLS2, RBFN, KELM, and the four proposed methods use the Gaussian kernel represented by

$$\kappa(x, y) = \exp\left\{-\frac{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})}{\sigma_G^2}\right\} \tag{27}$$

where σ_G is a hyper-parameter that controls the width of the Gaussian kernel. An additional parameter of RLS2, ELM, and KELM is the regularization parameter λ. In the case of KNNR, a design parameter k denotes the number of nearest neighbors for regression. In RBFN and ANFIS, the design parameters are the number of neurons N_R and the number of membership functions N_M for each variable, respectively. In SOAR, C_1 and C_2 are parameters that control the trade-off between the empirical risk and the regularization for input and output, respectively. In all methods, the design parameters are determined such that the lowest MSE for the training set is reached for each method. Using these design parameters, the MSE is evaluated for the test set.

Table IV shows the MSE for all experiments and methods except Exp. E, in which mean angular error (MAE) is used. In each experiment, the best performance is marked in bold face. In all experiments except Exp. B, one of the four proposed methods is the best. In Exp. B, SOAR takes the first place. The mean performance and ranking of the competing 12 methods are listed in the last two columns in Table IV. The mean performance is computed by normalizing the MSEs in each experiment first and then calculating the average of MSEs for each algorithm over seven experiments. The mean ranking is obtained by computing the average of rankings for each algorithm over seven experiments. From Table IV, it can be seen that overall, the four proposed algorithms show good performance. In particular, SOCPL1 demonstrates the best result in terms of the mean performance and the

TABLE V
TESTING TIME FOR EXP. A

Method	MKL_ SOCPL1	MKL_ SOCPL2	SOCPL1	SOCPL2	Pérez-Cruz's MSVR	SOAR	RLS2	KNNR	RBFN	ANFIS	ELM	KELM
Time(s)	0.0563	0.1449	0.0245	0.0240	0.0693	0.7512	0.0037	0.0084	0.0033	0.0477	0.0009	0.0062

mean ranking. In Exp. A, MKL-SOCPL1 is the best and the reason for this might be that the two outputs are not correlated at all and each output can select different kernel in MKL. In Exp. B, not the proposed methods but SOAR shows the best performance. The problem in Exp. B is actually a single chaotic dynamics and the three outputs are highly correlated to each other. SOAR is the algorithm which utilizes the correlation among the outputs and that might be the reason.

In Table V, the 12 methods are compared in terms of testing time for Exp. A. ELM and KELM take shortest time while SOAR takes longest time. The other methods take comparable time. The reason why SOAR takes much longer time is that it should predict the temporary output using BFGS [40].

In the conclusion, when one faces a multiple output regression problem, the most reasonable choice will be a SOCPL1. But when the outputs are completely uncorrelated and have different scales, MKL versions will help because it can select a different kernel which fits the most for each output. When one has to face a problem with high correlation between outputs such as a single dynamic system modeling, the best choice will be SOAR but it takes longer time.

VI. CONCLUSION

In this paper, new general dimensional MSVRs and their MKL versions were proposed and their performance was tested through application to benchmark problems. Unlike those from previous works: 1) the proposed MSVRs can be applied to regression problems with an arbitrary number of outputs; 2) the proposed MSVRs include the previous $\varepsilon_{\mathbb{X}}$-*SVR* as a special case; and 3) the MKL of the MSVR was developed. The proposed MSVRs and their MKL versions were applied to benchmark problems and experimental results confirmed their good performance.

APPENDIX

A. Dual Formulation of SOCPL1

Using the minmax representation, the primal formulation (5) can be rewritten as

$$p^* = \min_{\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq 0,\boldsymbol{\beta}\geq 0} \left[\frac{1}{2} Tr\left(\mathbf{W}\mathbf{W}^T\right) + C\sum_{i=1}^{M}\xi_i + \sum_{i=1}^{M}\alpha_i\left(\left\|\mathbf{y}_i - W^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right) - \mathbf{b}\right\| - \varepsilon - \xi_i\right) - \sum_{i=1}^{M}\beta_i\xi_i \right]. \quad (28)$$

Using the property of the norm $\|\mathbf{a}\| = \max_{\|\mathbf{u}\|\leq 1} u^T\mathbf{a}$ as in [41], (28) can be rewritten into

$$p^* = \min_{\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \max_{\substack{\boldsymbol{\alpha}\geq 0,\boldsymbol{\beta}\geq 0,\\ \|\mathbf{u}_i\|\leq\alpha_i}} \left[\frac{1}{2} Tr\left(\mathbf{W}\mathbf{W}^T\right) + C\sum_{i=1}^{M}\xi_i + \sum_{i=1}^{M}\mathbf{u}_i^T\left(\mathbf{y}_i - \mathbf{W}^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right) - \mathbf{b}\right) - \sum_{i=1}^{M}\alpha_i\left(\varepsilon + \xi_i\right) - \sum_{i=1}^{M}\beta_i\xi_i \right] \quad (29)$$

where the function given inside the bracket in (29) is obviously a convex–concave function (convex in primal variables while concave in dual variables). Thus, the interchange of the order of the min and the max is justified. Then, the associated dual problem is formulated as

$$d^* = \max_{\substack{\boldsymbol{\alpha}\geq 0,\boldsymbol{\beta}\geq 0,\\ \|\mathbf{u}_i\|\leq\alpha_i}} g\left(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{u}\right) \quad (30)$$

where

$$g\left(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{u}\right) = \min_{\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \left[\frac{1}{2} Tr\left(\mathbf{W}\mathbf{W}^T\right) + C\sum_{i=1}^{M}\xi_i + \sum_{i=1}^{M}\mathbf{u}_i^T\left(\mathbf{y}_i - \mathbf{W}^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right) - \mathbf{b}\right) - \sum_{i=1}^{M}\alpha_i\left(\varepsilon + \xi_i\right) - \sum_{i=1}^{M}\beta_i\xi_i \right]$$

is a Lagrange dual function. By setting the derivatives of the function inside the bracket to zeros, we obtain

$$\mathbf{W} = \sum_{i=1}^{M}\boldsymbol{\varphi}\left(\mathbf{x}_i\right)\mathbf{u}_i^T \quad (31)$$

$$\sum_{i=1}^{M}\mathbf{u}_i = 0 \quad (32)$$

$$\alpha_i + \beta_i = C. \quad (33)$$

Substituting (31)–(33) into (30) yields the dual formulation

$$d^* = \max_{\substack{\boldsymbol{\alpha}\geq 0,\boldsymbol{\beta}\geq 0,\\ \|\mathbf{u}_i\|\leq\alpha_i}} \left[-\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\mathbf{u}_i^T\mathbf{u}_j\kappa\left(\mathbf{x}_i,\mathbf{x}_j\right) + \sum_{i=1}^{M}\mathbf{u}_i^T\mathbf{y}_i - \varepsilon\boldsymbol{\alpha}^T\mathbf{e}_M \right] \text{ sub to } \begin{cases} \sum_{i=1}^{M}\mathbf{u}_i = 0 \\ \alpha_i + \beta_i = C \end{cases} \quad (34)$$

$$= \max_{\mathbf{u},\boldsymbol{\alpha}} \left[-\frac{1}{2}\mathbf{u}^T\left(\mathbf{K}\otimes\mathbf{I}_{N_O\times N_O}\right)\mathbf{u} + \mathbf{u}^T\mathbf{Y} - \varepsilon\boldsymbol{\alpha}^T\mathbf{e}_M \right] \quad (35)$$

$$\text{sub to } \begin{cases} \mathbf{0} \le \boldsymbol{\alpha} \le C\mathbf{e}_M \\ \|\mathbf{u}_i\| \le \alpha_i \\ \left(\mathbf{e}_M^T \otimes \mathbf{I}_{N_O \times N_O}\right)\mathbf{u} = \mathbf{0} \end{cases} \tag{36}$$

and the dual formulation (35) is obtained. Finally, the bias $\mathbf{b}$ is determined by

$$\begin{aligned} \mathbf{b} &= \arg\min_{\mathbf{b}} \sum_{i=1}^{M} \left(\|\mathbf{y}_i - \mathbf{W}\boldsymbol{\varphi}(\mathbf{x}_i) - \mathbf{b}\|\right) \\ &= \arg\min_{\mathbf{b}} \sum_{i=1}^{M} \left(\left\| \mathbf{y}_i - \sum_{j=1}^{M} \mathbf{u}_j \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) - \mathbf{b} \right\| \right). \end{aligned} \tag{37}$$

B. Proof of Remark 1.5

For the sake of simplicity, let us consider only the $\varepsilon_{\mathbb{H}}$-SVR. First, let us define $\bar{\boldsymbol{\alpha}} = \begin{bmatrix} \bar{\alpha}_1 \\ \bar{\alpha}_2 \\ \vdots \\ \bar{\alpha}_M \end{bmatrix} \in \mathbb{H}^{M\times 1}$, $\bar{\mathbf{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_M \end{bmatrix} \in \mathbb{H}^{M\times 1}$ where $\mathbb{H}$ is a set of quaternions; $\bar{\alpha}_n = \bar{\alpha}_n^{\Re} + \mathrm{i}\bar{\alpha}_n^i + \mathrm{j}\bar{\alpha}_n^j + \mathrm{k}\bar{\alpha}_n^k \in \mathbb{H}$ and $\bar{y}_n = \bar{y}_n^{\Re} + \mathrm{i}\bar{y}_n^i + \mathrm{j}\bar{y}_n^j + \mathrm{k}\bar{y}_n^k \in \mathbb{H}$. Then, the objective function of the problem given in the second row in Table II is

$$\begin{aligned} f_0(\bar{\boldsymbol{\alpha}}) &= -\frac{1}{2}\bar{\boldsymbol{\alpha}}^{\dagger}\mathbf{K}\bar{\boldsymbol{\alpha}} - \varepsilon\sum_{i=1}^{M}|\bar{\alpha}_i| + Re\left(\bar{\boldsymbol{\alpha}}^{\dagger}\bar{\mathbf{y}}\right) \\ &= -\frac{1}{2}\sum_{n=1}^{M}\sum_{m=1}^{M}\bar{\alpha}_n^{\dagger}\kappa(n,m)\bar{\alpha}_m - \varepsilon\sum_{n=1}^{M}|\bar{\alpha}_n| + \sum_{n=1}^{M}Re\left(\bar{\alpha}_n^{\dagger}\bar{y}_n\right) \\ &= -\frac{1}{2}\left\{\sum_{n=1}^{M}\bar{\alpha}_n^{\dagger}\kappa(n,n)\bar{\alpha}_n \right. \\ &\qquad \left. + \sum_{n<m}\left(\bar{\alpha}_n^{\dagger}\kappa(n,m)\bar{\alpha}_m + \bar{\alpha}_m^{\dagger}\kappa(m,n)\bar{\alpha}_n\right)\right\} \\ &\quad -\varepsilon\sum_{n=1}^{M}|\bar{\alpha}_n| + \sum_{n=1}^{M}Re\left(\bar{\alpha}_n^{\dagger}\bar{y}_n\right) \\ &= -\frac{1}{2}\left[\sum_{n=1}^{M}\left(\bar{\alpha}_n^{\Re} - \mathrm{i}\bar{\alpha}_n^i - \mathrm{j}\bar{\alpha}_n^j - \mathrm{k}\bar{\alpha}_n^k\right)\kappa(n,n) \right. \\ &\qquad \left(\bar{\alpha}_n^{\Re} + \mathrm{i}\bar{\alpha}_n^i + \mathrm{j}\bar{\alpha}_n^j + \mathrm{k}\bar{\alpha}_n^k\right) \\ &\qquad + \sum_{n<m}\left\{\left(\bar{\alpha}_n^{\Re} - \mathrm{i}\bar{\alpha}_n^i - \mathrm{j}\bar{\alpha}_n^j - \mathrm{k}\bar{\alpha}_n^k\right)\kappa(n,m) \right. \\ &\qquad\qquad \left(\bar{\alpha}_m^{\Re} + \mathrm{i}\bar{\alpha}_m^i + \mathrm{j}\bar{\alpha}_m^j + \mathrm{k}\bar{\alpha}_m^k\right) \\ &\qquad\qquad + \left(\bar{\alpha}_m^{\Re} - \mathrm{i}\bar{\alpha}_m^i - \mathrm{j}\bar{\alpha}_m^j - \mathrm{k}\bar{\alpha}_m^k\right)\kappa(m,n) \\ &\qquad\qquad \left.\left.\left(\bar{\alpha}_n^{\Re} + \mathrm{i}\bar{\alpha}_n^i + \mathrm{j}\bar{\alpha}_n^j + \mathrm{k}\bar{\alpha}_n^k\right)\right\}\right] \\ &\quad - \varepsilon\sum_{n=1}^{M}\left|\left(\bar{\alpha}_n^{\Re} + \mathrm{i}\bar{\alpha}_n^i + \mathrm{j}\bar{\alpha}_n^j + \mathrm{k}\bar{\alpha}_n^k\right)\right| \\ &\quad + \sum_{n=1}^{M}Re\left\{\left(\bar{\alpha}_n^{\Re} - \mathrm{i}\bar{\alpha}_n^i - \mathrm{j}\bar{\alpha}_n^j - \mathrm{k}\bar{\alpha}_n^k\right) \right. \\ &\qquad \left.\left(\bar{y}_n^{\Re} + \mathrm{i}\bar{y}_n^i + \mathrm{j}\bar{y}_n^j + \mathrm{k}\bar{y}_n^k\right)\right\} \end{aligned}$$

where $\mathbb{H}$ is the set of quaternions. Using the quaternion product [42] defined by

$$\begin{aligned} &\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1, \mathrm{ij} = -\mathrm{ji} = \mathrm{k}, \\ &\mathrm{jk} = -\mathrm{jk} = \mathrm{i} \text{ and } \mathrm{ki} = -\mathrm{ik} = \mathrm{j} \end{aligned} \tag{38}$$

and the distributive law, then

$$\begin{aligned} &\left(\bar{\alpha}_n^{\Re} - \mathrm{i}\bar{\alpha}_n^i - \mathrm{j}\bar{\alpha}_n^j - \mathrm{k}\bar{\alpha}_n^k\right)\kappa(n,n)\left(\bar{\alpha}_n^{\Re} + \mathrm{i}\bar{\alpha}_n^i + \mathrm{j}\bar{\alpha}_n^j + \mathrm{k}\bar{\alpha}_n^k\right) \\ &\quad = \kappa(n,n)\left(\bar{\alpha}_n^{\Re 2} + \bar{\alpha}_n^{i2} + \bar{\alpha}_n^{j2} + \bar{\alpha}_n^{k2}\right) \end{aligned}$$

and

$$\begin{aligned} &\left(\bar{\alpha}_n^{\Re} - \mathrm{i}\bar{\alpha}_n^i - \mathrm{j}\bar{\alpha}_n^j - \mathrm{k}\bar{\alpha}_n^k\right)\kappa(n,m)\left(\bar{\alpha}_m^{\Re} + \mathrm{i}\bar{\alpha}_m^i + \mathrm{j}\bar{\alpha}_m^j + \mathrm{k}\bar{\alpha}_m^k\right) \\ &\quad + \left(\bar{\alpha}_m^{\Re} - \mathrm{i}\bar{\alpha}_m^i - \mathrm{j}\bar{\alpha}_m^j - \mathrm{k}\bar{\alpha}_m^k\right)\kappa(m,n) \\ &\qquad \left(\bar{\alpha}_n^{\Re} + \mathrm{i}\bar{\alpha}_n^i + \mathrm{j}\bar{\alpha}_n^j + \mathrm{k}\bar{\alpha}_n^k\right) \\ &\quad = 2\kappa(n,m)\left(\bar{\alpha}_n^{\Re}\bar{\alpha}_m^{\Re} + \bar{\alpha}_n^i\bar{\alpha}_m^i + \bar{\alpha}_n^j\bar{\alpha}_m^j + \bar{\alpha}_n^k\bar{\alpha}_m^k\right). \end{aligned}$$

Thus, $\varepsilon_{\mathbb{H}}$-SVR is formulated as

$$\begin{aligned} d^* = \max_{\alpha}\Bigg\{ &-\frac{1}{2}\left[\sum_{n=1}^{M}\kappa(n,n)\left(\bar{\alpha}_n^{\Re 2} + \bar{\alpha}_n^{i2} + \bar{\alpha}_n^{j2} + \bar{\alpha}_n^{k2}\right)\right. \\ &\left. + 2\sum_{n<m}\kappa(n,m)\left(\bar{\alpha}_n^{\Re}\bar{\alpha}_m^{\Re} + \bar{\alpha}_n^i\bar{\alpha}_m^i + \bar{\alpha}_n^j\bar{\alpha}_m^j + \bar{\alpha}_n^k\bar{\alpha}_m^k\right)\right] \\ &- \varepsilon\sum_{n=1}^{M}\sqrt{\bar{\alpha}_n^{\Re 2} + \bar{\alpha}_n^{i2} + \bar{\alpha}_n^{j2} + \bar{\alpha}_n^{k2}} \\ &+ \sum_{n=1}^{M}\left(\bar{\alpha}_n^{\Re}\bar{y}_n^{\Re} + \bar{\alpha}_n^i\bar{y}_n^i + \bar{\alpha}_n^j\bar{y}_n^j + \bar{\alpha}_n^k\bar{y}_n^k\right)\Bigg\} \\ \text{sub to } &\begin{cases} |\bar{\alpha}_i| \le C \\ \mathbf{e}_M^T\bar{\boldsymbol{\alpha}} = 0 \end{cases} \\ = \max_{\alpha,t}\Bigg\{ &-\frac{1}{2}\left[\sum_{n,m}\kappa(n,m)\left(\bar{\alpha}_n^{\Re}\bar{\alpha}_m^{\Re} + \bar{\alpha}_n^i\bar{\alpha}_m^i + \bar{\alpha}_n^j\bar{\alpha}_m^j + \bar{\alpha}_n^k\bar{\alpha}_m^k\right)\right] \\ &- \varepsilon\sum_{n=1}^{M}t_n + \sum_{n=1}^{M}\left(\bar{\alpha}_n^{\Re}\bar{y}_n^{\Re} + \bar{\alpha}_n^i\bar{y}_n^i + \bar{\alpha}_n^j\bar{y}_n^j + \bar{\alpha}_n^k\bar{y}_n^k\right)\Bigg\} \\ \text{sub to } &\begin{cases} \sqrt{\bar{\alpha}_n^{\Re 2} + \bar{\alpha}_n^{i2} + \bar{\alpha}_n^{j2} + \bar{\alpha}_n^{k2}} \le t_n \\ |\bar{\alpha}_i| \le C \\ \mathbf{e}_M^T\bar{\boldsymbol{\alpha}} = 0 \end{cases} \end{aligned}$$

by careful comparison, we can see that the above formulation is the same as (6) in SOCPL1.

C. Dual Formulation of SOCP2

The primal formulation (9) can be rewritten using the minmax representation

$$\begin{aligned} p^* = \min_{\mathbf{W},\mathbf{b},\boldsymbol{\xi}}\max_{\boldsymbol{\alpha}\ge 0}\ &\frac{1}{2}Tr\left(\mathbf{W}\mathbf{W}^T\right) + \frac{C}{2}\sum_{i=1}^{M}\xi_i^2 \\ &+ \sum_{i=1}^{M}\alpha_i\left(\left\|\mathbf{y}_i - \mathbf{W}^T\boldsymbol{\varphi}(\mathbf{x}_i) - \mathbf{b}\right\| - \varepsilon - \xi_i\right) \end{aligned} \tag{39}$$

As in SOCPL1, we use the property of the norm $\|\mathbf{a}\| = \max_{\|\mathbf{u}\|\leq 1} \mathbf{u}^T\mathbf{a}$ [41] and the primal problem (39) becomes

$$p^* = \min_{\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \max_{\substack{\boldsymbol{\alpha}\geq 0 \\ \|\mathbf{u}_i\|\leq \alpha_i}} \left[\frac{1}{2}Tr\left(\mathbf{W}\mathbf{W}^T\right) + \frac{C}{2}\sum_{i=1}^{M}\xi_i^2 + \sum_{i=1}^{M}\mathbf{u}_i^T\left(\mathbf{y}_i - \mathbf{W}^T\phi\left(\mathbf{x}_i\right) - \mathbf{b}\right) - \sum_{i=1}^{M}\alpha_i\left(\varepsilon + \xi_i\right)\right]. \tag{40}$$

Then, the associated dual problem is formulated as

$$d^* = \max_{\substack{\boldsymbol{\alpha}\geq 0 \\ \|\mathbf{u}_i\|\leq \alpha_i}} g\left(\boldsymbol{\alpha}, \mathbf{u}\right) \tag{41}$$

where

$$g\left(\boldsymbol{\alpha}, \mathbf{u}\right) = \min_{\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \left[\frac{1}{2}Tr\left(\mathbf{W}\mathbf{W}^T\right) + C\sum_{i=1}^{M}\xi_i + \sum_{i=1}^{M}\mathbf{u}_i^T\left(\mathbf{y}_i - \mathbf{W}^T\boldsymbol{\varphi}\left(\mathbf{x}_i\right) - \mathbf{b}\right) - \sum_{i=1}^{M}\alpha_i\left(\varepsilon + \xi_i\right) - \sum_{i=1}^{M}\beta_i\xi_i\right].$$

By setting the derivatives of the function inside the bracket to zeros, we obtain

$$W = \sum_{i=1}^{M}\boldsymbol{\varphi}\left(\mathbf{x}_i\right)\mathbf{u}_i^T \tag{42}$$

$$\sum_{i=1}^{M}\mathbf{u}_i = 0 \tag{43}$$

$$\boldsymbol{\xi} = \frac{1}{C}\boldsymbol{\alpha}. \tag{44}$$

Substituting (42)–(44) into (41) yields the dual formulation

$$d^* = \max_{\substack{\boldsymbol{\alpha}\geq 0 \\ \|\mathbf{u}_i\|\leq \alpha_i}} \left[-\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\mathbf{u}_i^T\mathbf{u}_j\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) + \sum_{i=1}^{M}\mathbf{u}_i^T\mathbf{y}_i - \varepsilon\boldsymbol{\alpha}^T\mathbf{e}_M - \frac{1}{2C}\boldsymbol{\alpha}^T\boldsymbol{\alpha}\right] \quad \text{sub to} \sum_{i=1}^{M}\mathbf{u}_i = 0 \tag{45}$$

which is actually the same as (10). Finally, the bias b is determined by

$$\mathbf{b} = \arg\min_{\mathbf{b}} \sum_{i=1}^{M}\left(\|\mathbf{y}_i - \mathbf{W}\boldsymbol{\varphi}\left(\mathbf{x}_i\right) - \mathbf{b}\|\right) = \arg\min_{\mathbf{b}} \sum_{i=1}^{M}\left(\left\|\mathbf{y}_i - \sum_{j=1}^{M}\mathbf{u}_j\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) - \mathbf{b}\right\|\right). \tag{46}$$

D. Dual Formulation of SOCPL1-MKL

The primal formulation (14) can be rewritten using the minmax representation

$$p^* = \min_{\mathbf{D},\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \max_{\substack{\boldsymbol{\alpha}\geq 0, \boldsymbol{\beta}\geq 0 \\ \boldsymbol{\Lambda}\in\mathcal{D}_{N_O}, \boldsymbol{\eta}_m\geq 0}} \left[\frac{1}{2}\sum_{m=1}^{N_k}Tr\left(\mathbf{W}_m\mathbf{D}_m^{-1}\mathbf{W}_m^T\right) + C\sum_{i=1}^{M}\xi_i + \sum_{i=1}^{M}\alpha_i\left(\left\|\mathbf{y}_i - \sum_{m=1}^{N_k}\mathbf{W}_m^T\boldsymbol{\varphi}_m\left(\mathbf{x}_i\right) - \mathbf{b}\right\| - \varepsilon - \xi_i\right) - \sum_{i=1}^{M}\beta_i\xi_i + Tr\left\{\boldsymbol{\Lambda}\left(\sum_{m=1}^{N_k}\mathbf{D}_m - \mathbf{I}_{N_O}\right)\right\} - \sum_{m=1}^{N_k}Tr\left(\boldsymbol{\eta}_m\mathbf{D}_m\right)\right]. \tag{47}$$

Using the property of the norm $\|\mathbf{a}\| = \max_{\|\mathbf{u}\|\leq 1}\mathbf{u}^T\mathbf{a}$ [41], the primal problem (14) becomes

$$p^* = \min_{\mathbf{D},\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \max_{\substack{\boldsymbol{\alpha}\geq 0, \boldsymbol{\beta}\geq 0 \\ \|\mathbf{u}_i\|\leq \alpha_i \\ \boldsymbol{\Lambda}\in\mathcal{D}_{N_O} \\ \boldsymbol{\eta}_m\geq\mathbf{0}}} \left[\frac{1}{2}\sum_{m=1}^{N_k}Tr\left(\mathbf{W}_m\mathbf{D}_m^{-1}\mathbf{W}_m^T\right) + C\sum_{i=1}^{M}\xi_i + \sum_{i=1}^{M}\mathbf{u}_i^T\left(\mathbf{y}_i - \sum_{m=1}^{N_k}\mathbf{W}_m^T\boldsymbol{\varphi}_m\left(\mathbf{x}_i\right) - \mathbf{b}\right) - \sum_{i=1}^{M}\alpha_i\left(\varepsilon + \xi_i\right) - \sum_{i=1}^{M}\beta_i\xi_i + Tr\left\{\boldsymbol{\Lambda}\left(\sum_{m=1}^{N_k}\mathbf{D}_m - \mathbf{I}_{N_O}\right)\right\} - \sum_{m=1}^{N_k}Tr\left(\boldsymbol{\eta}_m\mathbf{D}_m\right)\right]. \tag{48}$$

Then, the associated dual problem is

$$d^* = \max_{\substack{\boldsymbol{\alpha}\geq 0, \boldsymbol{\beta}\geq 0 \\ \|\mathbf{u}_i\|\leq \alpha_i \\ \boldsymbol{\Lambda}\in\mathcal{D}_{N_O}, \boldsymbol{\eta}_m\geq\mathbf{0}}} g\left(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{u}, \boldsymbol{\Lambda}, \boldsymbol{\eta}\right) \tag{49}$$

where

$$g\left(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{u}, \boldsymbol{\Lambda}, \boldsymbol{\eta}\right) = \min_{\mathbf{D},\mathbf{W},\mathbf{b},\boldsymbol{\xi}} \left[\frac{1}{2}\sum_{m=1}^{N_k}Tr\left(\mathbf{W}_m\mathbf{D}_m^{-1}\mathbf{W}_m^T\right) + C\sum_{i=1}^{M}\xi_i + \sum_{i=1}^{M}\mathbf{u}_i^T\left(\mathbf{y}_i - \sum_{m=1}^{N_k}\mathbf{W}_m^T\boldsymbol{\varphi}_m\left(\mathbf{x}_i\right) - \mathbf{b}\right) - \sum_{i=1}^{M}\alpha_i\left(\varepsilon + \xi_i\right) - \sum_{i=1}^{M}\beta_i\xi_i + Tr\left\{\boldsymbol{\Lambda}\left(\sum_{m=1}^{N_k}\mathbf{D}_m - \mathbf{I}_{N_O}\right)\right\} - \sum_{m=1}^{N_k}Tr\left(\boldsymbol{\eta}_m\mathbf{D}_m\right)\right]. \tag{50}$$

By setting the derivatives of the function inside the bracket to zeros, we obtain

$$\mathbf{W}_m = \left\{ \sum_{i=1}^{M} \boldsymbol{\varphi}_m (\mathbf{x}_i)\, \mathbf{u}_i^T \right\} \mathbf{D}_m \tag{51}$$

$$\sum_{i=1}^{M} \mathbf{u}_i = 0 \tag{52}$$

$$\alpha_i + \beta_i = C \tag{53}$$

$$\boldsymbol{\Lambda} - \boldsymbol{\eta}_m = \frac{1}{2} diag\left(\mathbf{D}_m^{-1} \mathbf{W}_m^T \mathbf{W}_m \mathbf{D}_m^{-1} \right). \tag{54}$$

Substituting (51) into (50) yields the dual formulation

$$\begin{aligned} d^* &= \max_{\substack{\boldsymbol{\alpha} \geq 0, \boldsymbol{\beta} \geq 0 \\ \|\mathbf{u}_i\| \leq \alpha_i \\ \boldsymbol{\Lambda} \in \mathcal{D}_{N_O} \\ \boldsymbol{\eta}_m \succeq \mathbf{0}}} g(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{u}, \boldsymbol{\Lambda}, \boldsymbol{\eta}) \\ &= \max_{\substack{\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{u}_i \\ \boldsymbol{\Lambda}, \boldsymbol{\eta}_m}} \left[\mathbf{u}^T \mathbf{Y} - \varepsilon \boldsymbol{\alpha}^T \mathbf{e}_M - Tr\boldsymbol{\Lambda} \right] \\ &\qquad \text{Sub to} \begin{cases} \alpha_i \geq 0, \beta_i \geq 0, \|\mathbf{u}_i\| \leq \alpha_i \\ \boldsymbol{\eta}_m \succeq \mathbf{0}, \boldsymbol{\eta}_m \in \mathcal{D}_{N_O} \\ \boldsymbol{\Lambda} \in \mathcal{D}_{N_O} \\ \sum_{i=1}^{M} \mathbf{u}_i = 0 \\ \alpha_i + \beta_i = C \\ \boldsymbol{\Lambda} - \boldsymbol{\eta}_m = diag\left(\frac{1}{2} \left\{ \sum_{i=1}^{M} \boldsymbol{\phi}_m (\mathbf{x}_i)\, \mathbf{u}_i^T \right\}^T \right. \\ \qquad\qquad \left. \left\{ \sum_{j=1}^{M} \boldsymbol{\phi}_m (\mathbf{x}_j)\, \mathbf{u}_j^T \right\} \right) \\ = diag\left(\frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \kappa_m (\mathbf{x}_i, \mathbf{x}_j)\, \mathbf{u}_i \mathbf{u}_j^T \right) \end{cases} \end{aligned} \tag{55}$$

which is actually the same as (15).

REFERENCES

[1] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.,* vol. 14, no. 3, pp. 199–222, 2004.

[2] Z. Liu *et al.*, "A three-domain fuzzy support vector regression for image denoising and experimental studies," *IEEE Trans. Cybern.*, vol. 44, no. 4, pp. 516–525, Apr. 2014.

[3] W. Kim, J. Park, J. Yoo, H. J. Kim, and C. G. Park, "Target localization using ensemble support vector regression in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 4, pp. 1189–1198, Aug. 2013.

[4] L. Wang, Z. Liu, C. L. P. Chen, Y. Zhang, and S. Lee, "Energy-efficient SVM learning control system for biped walking robots," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 24, no. 5, pp. 831–837, Mar. 2013.

[5] Z. Lu, J. Sun, and K. Butts, "Multiscale asymmetric orthogonal wavelet kernel for linear programming support vector learning and nonlinear dynamic systems identification," *IEEE Trans. Cybern.*, vol. 44, no. 5, pp. 712–724, May 2014.

[6] S. Qiu and T. Lane, "A Framework for multiple kernel support vector regression and its applications to siRNA efficacy prediction," *IEEE-ACM Trans. Comput. Biol. Bioinform.*, vol. 6, no. 2, pp. 190–199, Apr./Jun. 2009.

[7] X. Zhang, L. Hu, and L. Zhang, "An efficient multiple kernel computation method for regression analysis of economic data," *Neurocomputing*, vol. 118, pp. 58–64, Oct. 2013.

[8] Y. Bao, T. Xiong, and Z. Hu, "Multi-step-ahead time series prediction using multiple-output support vector regression," *Neurocomputing*, vol. 129, pp. 482–493, Apr. 2014.

[9] J. Lee, B. Choi, and E. Kim, "A novel range-free localization based on multi-dimensional support vector regression trained in the primal space," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 24, no. 7, pp. 1099–1113, Jul. 2013.

[10] N. Gourier, D. Hall, and J. L. Crowley, "Estimating face orientation from robust detection of salient facial structures," in *Proc. ICPR Workshop Vis. Obs. Deictic Gestures,* Cambridge, U.K., 2004, pp. 17–25.

[11] D. N. Tuong and J. Peters, "Model learning for robot control: A survey," *Cogn. Process.*, vol. 12, no. 4, pp. 319–340, 2011.

[12] X. Liu, Q. Zhu, and H. Lu, "Modeling multiresponse surfaces for airfoil design with multiple-output-Gaussian-process regression," *J. Aircraft*, vol. 51, no. 3, pp. 740–747, 2014.

[13] M. Gonen and S. Kaski, "Kernelized Bayesian matrix factorization," *IEEE Trans. Pattern Recognit. Mach. Intell.*, vol. 36, no. 10, pp. 2047–2060, Oct. 2014.

[14] M. Sánchez-Fernández, M. de Prade-Cumplido, J. Arenas-García, and F. Pérez-Cruz, "SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems," *IEEE Trans. Signal Process.,* vol. 58, no. 8, pp. 2298–2307, Aug. 2007.

[15] F. Pérez-Cruz *et al.*, "Multi-dimensional function approximation and regression estimation," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*, Madrid, Spain, Aug. 2002, pp. 757–762.

[16] A. Shilton, B. Sundaram, and M. Palaniswami, "Ad-hoc wireless sensor network localization using support vector regression," in *Proc. ICT Mobile Summit*, pp. 1–8, Jun. 2008.

[17] A. Shilton, D. T. H. Lai, and M. Palaniswami, "A division algebraic framework for multidimensional support vector regression," *IEEE Trans. Syst., Man, Cybern. B, Cybern.,* vol. 40, no. 2, pp. 517–528, Mar. 2010.

[18] A. Shilton, D. T. H. Lai, B. K. Santhiranayagam, and M. Palaniswami, "A note on octonionic support vector regression," *IEEE Trans. Syst., Man, Cybern. B, Cybern.,* vol. 42, no. 3, pp. 950–955, Jun. 2012.

[19] CVX Research Inc. (Sep. 2009). *CVX: MATLAB Software for Disciplined Convex Programming, Version 1.2 Beta.* [Online]. Available: http://cvxr.com/cvx/

[20] B. Schölkopf, P. Bartlett, A. J. Smola, and R. Williamson, "Support vector regression with automatic accuracy control," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*, Skövde, Sweden, 1998, pp. 111–116.

[21] L. Bo and C. Sminchisescu, "Structured output-associative regression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* Miami, FL, USA, 2009, pp. 2403–2410.

[22] K. Jia, L. Wang, and N. Liu, "Efficient structured support vector regression," in *Computer Vision-ACCV 2010*. Berlin, Germany: Springer, 2011, pp. 586–598.

[23] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proc. 21st Int. Conf. Mach. Learn.*, Banff, AB, Canada, 2004, p. 104.

[24] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, Dec. 2004.

[25] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *J. Mach. Learn. Res.*, vol. 9, pp. 2491–2521, Nov. 2008.

[26] F. Bach, G. R. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. 21st Int. Conf. Mach. Learn.*, Banff, AB, Canada, 2004, p. 6.

[27] F. Dinuzzo. (2010). *Kernel Machines With Two Layers and Multiple Kernel Learning.* [Online]. Available: http://www-dimat.unipv.it/dinuzzo

[28] F. Burba, F. Ferraty, and P. Vieu, "k-Nearest Neighbour method in functional nonparametric regression," *J. Nonparametr. Statist.,* vol. 21, no. 4, pp. 453–469, 2009.

[29] D. S. Huang, "Radial basis probabilistic neural networks: Model and application," *Int. J. Pattern Recognit. Artif. Intell.,* vol. 13, no. 7, pp. 1083–1101, 1999.

[30] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.,* vol. 23, no. 3, pp. 665–685, May/Jun. 1993.

[31] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machines: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.

[32] G. B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 107–122, 2011.

[33] E. Cambria. (2013). *Extreme Learning Machines*. [Online]. Available: http://www.ntu.edu.sg/home/egbhuang/elm_codes.html

[34] E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.,* vol. 20, pp. 130–141, Mar. 1963.

[35] L. Prechelt, "PROBEN 1—A set of benchmarks and benchmarking rules for neural network training algorithms," Fakultät für Informatik, Universität Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, 1994.
[36] Z. Yixin and J. Zhang, "Stock data analysis based on BP neural network," in *Proc. Commun. Softw. Netw.*, Singapore, 2010, pp. 396–399.
[37] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1. San Diego, CA, USA, 2005, pp. 886–893.
[38] R. Stiefelhagen, "Estimating head pose with neural networks-results on the pointing04 ICPR workshop evaluation data," in *Proc. ICPR Workshop Vis. Obs. Deictic Gestures*, Cambridge, U.K., 2004.
[39] S. Xu, X. An, X. Qiao, L. Zhu, and L. Li, "Multi-output least-squares support vector regression machines," *Pattern Recognit. Lett.*, vol. 34, no. 9, pp. 1078–1084, 2013.
[40] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, nos. 1–3, pp. 503–528, 1989.
[41] L. E. Ghaoui. (2012). *EE 227A lecture note 10*. [Online]. Available: http://www.eecs.berkeley.edu/∼elghaoui/Teaching/EE227A/lecture10.pdf
[42] J. B. Kuipers, "Quaternion algebra," in *Quaternions and Rotation Sequences: A Primer With Applications to Orbits, Aerospace, and Virtual Reality*. Princeton, NJ, USA: Princeton Univ. Press, 1999, pp. 103–140.

Wooyong Chung was born in Seoul, Korea, in 1978. He received the B.S. and M.S. degrees in electronic engineering from Yonsei University, Seoul, Korea, in 2004 and 2006, respectively. He is currently pursuing the Ph.D. degree from the School of Electrical and Electronic Engineering, Yonsei University.

His current research interests include machine learning and fuzzy control.

Jisu Kim received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2011, where he is currently pursuing the combined master's and Ph.D. degrees.

His current research interests include machine learning and pattern recognition for vehicle detection.

Heejin Lee was born in Seoul, Korea, in 1964. He received the B.S., M.S., and Ph.D. degrees in electronic engineering, all from Yonsei University, Seoul, Korea, in 1987, 1989, and 1998, respectively.

From 1989 to 1993, he was at Daewoo Telecom Ltd., Seoul, as a Researcher. He is currently an Associate Professor with the Department of Information and Control Engineering, Hankyong National University, Anseong, Korea. His current research interests include fuzzy control theory, fuzzy application system, adaptive and robust control, robotics, and automation.

Euntai Kim (M'00) was born in Seoul, Korea, in 1970. He received the B.S. degree (as the top of the university) and the M.S. and Ph.D. degrees in electronic engineering, all from Yonsei University, Seoul, Korea, in 1992, 1994, and 1999, respectively.

From 1999 to 2002, he was a Full-Time Lecturer at the Department of Control and Instrumentation Engineering, Hankyong National University, Anseong, Korea. Since 2002, he has been with the Faculty of the School of Electrical and Electronic Engineering, Yonsei University, where he is currently a Professor. He was a Visiting Scholar at the University of Alberta, Edmonton, AB, Canada, in 2003, and also was a Visiting Researcher at the Berkeley Initiative in Soft Computing, University of California, Berkeley, Berkeley, CA, USA, in 2008. His current research interests include computational intelligence and statistical machine learning and their application to intelligent robotics, unmanned vehicles, and robot vision.