Published in final edited form as:

IEEE Trans Neural Netw Learn Syst. 2014 October ; 25(10): 1879–1893. doi:10.1109/TNNLS.2013.2297686.

Multiobjective Optimization for Model Selection in Kernel Methods in Regression

Di You, **C. Fabian Benitez-Quiroz**, and **Aleix M. Martinez**

Dept. Electrical and Computer Engineering The Ohio State University, Columbus, OH 43210

Abstract

Regression plays a major role in many scientific and engineering problems. The goal of regression is to learn the unknown underlying function from a set of sample vectors with known outcomes. In recent years, kernel methods in regression have facilitated the estimation of nonlinear functions. However, two major (interconnected) problems remain open. The first problem is given by the bias-vs-variance trade-off. If the model used to estimate the underlying function is too flexible (i.e., high model complexity), the variance will be very large. If the model is fixed (i.e., low complexity), the bias will be large. The second problem is to define an approach for selecting the appropriate parameters of the kernel function. To address these two problems, this paper derives a new smoothing kernel criterion, which measures the roughness of the estimated function as a measure of model complexity. Then, we use multiobjective optimization to derive a criterion for selecting the parameters of that kernel. The goal of this criterion is to find a trade-off between the bias and the variance of the learned function. That is, the goal is to increase the model fit while keeping the model complexity in check. We provide extensive experimental evaluations using a variety of problems in machine learning, pattern recognition and computer vision. The results demonstrate that the proposed approach yields smaller estimation errors as compared to methods in the state of the art.

Keywords

Kernel methods; kernel optimization; regression; Pareto-optimality; optimization

I. Introduction

Regression analysis has been a very active topic in machine learning and pattern recognition, with applications in many problems in science and engineering. In a standard regression problem, a linear or nonlinear model is estimated from the data such that the functional relationship between the dependent variables and the independent variables can be established. Of late, regression with kernel methods [39, 33] has become popular. The success of the kernel methods in regression comes from the fact that they facilitate the estimation of nonlinear function using well-defined and - tested approaches in, for example, computer vision [40], signal processing [38, 7], and bioinformatics [28].

The underlying idea in kernel methods is to map the data samples from an original space to a space of (intrinsically) much higher (or infinite) dimensionality. We refer to the resulting space as the *kernel space*. The goal is to find a mapping that converts the original nonlinear

problem (defined in the original space) into a linear one (in the kernel space) [32]. In practise, this mapping is done using a pre-determined nonlinear function. Given this function, the main challenge is to fine those parameters of the function that convert a nonlinear problem into a linear one. Thus, the selection of these kernel parameters is a type of model selection. This is the problem we consider in this paper – to define a criterion for the selection of the appropriate parameters of this kernel function.

The selection of the appropriate parameters of a kernel is a challenging one [47]. If the parameters were chosen to minimize the model fit, we would generally have an *over-fitting* to the training data. As a consequence, the regressed function would not be able to estimate the testing data correctly. A classical solution is to find a good fit, while keeping the complexity of the function low, e.g., using a polynomial of lower order [16]. However, if the parameters are selected to keep the complexity too low, then we will *under-fit* the data. In both these cases, the regressed function will have a poor generalization, i.e., a high prediction error to the testing data. In general, the kernel parameters should be selected to achieve an appropriate trade-off between the *model fit* and *model complexity*.

To date, the most widely employed technique to do this selection of the kernel parameters is k-fold cross-validation (or CV, for short) [16]. In this approach, the performance of the prediction models is evaluated by setting aside a validation set within the training set. The model which produces the smallest validation error is selected. Unfortunately, this method has three known major drawbacks. First, it is computational expensive. Second, only part of the training data is used to estimate the model parameters. When doing model selection, one wants to employ the largest possible number of training samples, since this is known to yield better generalizations [24]. Third, the value of k as a parameter plays a major role in the process. Note that the value of k affects the trade-off between the fitting error and the model complexity, yet general methods for selecting an appropriate value do not exist.

An alternative to CV is Generalized CV (GCV) [39], an approach originally defined to select the ridge parameter in ridge regression. GCV can be directly extended to do model selection in regression with kernel approaches, as long as the *hat matrix* [39], which projects the original response vector to the estimated one, can be obtained. However, since this criterion is an approximation of the leave-one-out CV (i.e., n-fold CV, n is the number of training samples), the estimated result generally has a large variance and small bias, i.e., the regressed function is highly variant and dependent of the training data.

Another alternative is to minimize the structural risk [37], which is a function of the model fit and the Vapnik-Chervonenkis (VC) dimension of the regressed function. Unfortunately, the VC dimension is infinite in regularized Kernel regression approaches with an induced infinite dimensional space (e.g, Radial Basis Function), limiting the applicability of such approaches [11, 12].

While a single kernel may not be sufficient to describe the data, Multiple Kernel Learning (MKL) [21, 34] has attracted much attention recently as a potential alternative. In [28], MKL is applied to Support Vector Regression (SVR). The coefficients that determine the combination of kernels are learned using a constrained quadratic programming problem.

This method was shown to outperform CV in some applications. Unfortunately, the selection of its parameters is generally problem specific [6, 46, 28], and while the general problem remains unsolved, this is an active area of research. In yet another approach, the kernel parameters are selected by maximizing the marginal data likelihood after reformulating the regression problem as probabilistic models using Bayesian inference. This approach has been used to define the well-known Relevance Vector Machine (RVM) [36] and Gaussian processes for regression [45]. It has been shown [30] that the marginal likelihood has the nice property of automatically incorporating a trade-off between model fit and model complexity. However, since the Bayesian learning generally leads to analytically intractable posteriors, approximations are necessary, and, the results are generally computationally very expensive. Furthermore, the determination of the priors for the parameters is an intrinsic problem in Bayesian learning with no clear solution.

In this paper, we resolve the kernel optimization problem using a completely novel approach. In our proposed approach, the two measures of model fitness and model complexity are simultaneously minimized using a multiobjective optimization (MOP) framework through the study of Pareto-optimal solutions. MOP and Pareto-optimality are specifically defined to find the global minima of several combined criteria. To this end, we will first derive a new criterion for model complexity which can be employed in kernel methods in regression. We then define a method using MOP and derive a new approach called modified ε-constraint. We show that this newly derived approach achieves the lowest mean square error. We provide extensive comparisons with the state of the art in kernel methods for regression and on approaches for model selection. The results show that the proposed framework leads to better generalizations for the (unseen) testing samples.

The remainder of this paper is organized as follows. In Section II we derive the two new measures of model fitness and model complexity. Then, in Section III, we derive a new MOP approach to do model selection. In Section IV, the proposed framework is applied to two typical kernel methods in regression. Experimental results are provided in Section V. We conclude in Section VI.

II. Regression Models

We start with an analysis of the generalization error of a regression model. Given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$, $\mathbf{y}_i \in \mathbb{R}^q$, with the training samples ($\mathbf{x}_i$, $\mathbf{y}_i$), $i = 1, ..., n$ generated from a joint distribution $g(\mathbf{x}, \mathbf{y})$, one wants to find the regression model $\mathbf{f}(\mathbf{x})$ that minimizes the generalization error

$$E = \int L(\mathbf{y}, \mathbf{f}(\mathbf{x}))\, g(\mathbf{x}, \mathbf{y})\, d\mathbf{x} d\mathbf{y}, \quad (1)$$

where $\mathbf{f}(\mathbf{x})$ is the regression function, $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_q(\mathbf{x}))^T$, $f_i(.) : \mathbb{R}^p \to \mathbb{R}$ is the i^{th} regression function, and $L(\mathbf{y}, \mathbf{f}(\mathbf{x}))$ is a given loss function, for instance, the quadratic loss $L(\mathbf{y}, \mathbf{f}(\mathbf{x})) = \frac{1}{2}\|\mathbf{y} - \mathbf{f}(\mathbf{x})\|_2^2 = \frac{1}{2}\sum_{i=1}^q (y_i - f_i(\mathbf{x}))^2$.

A. Generalization error

Holmstrom and Koistinen [18] show that by adding noise to the training samples (both **x** and **y**), the estimation of the generalization error is asymptotically consistent, i.e., as the number of training examples approaches infinity, the estimated generalization error is equivalent to the true one. The addition of noise can be interpreted as generating additional training samples. This means that we can estimate the generalization error by adding the noise to the training samples.

For convenience, denote the training set that consists of n pairs of observation vectors and prediction vectors by $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i), \quad i=1,\ldots,n, \quad \mathbf{z}_i \in \mathbb{R}^m, \quad m=p+q$. Then the generalization error can be rewritten as

$$E = \int L(\mathbf{z})\, g(\mathbf{z})\, d\mathbf{z}. \quad (2)$$

Assume that the training samples $\mathbf{z}_i$ are corrupted by the noise $\boldsymbol{\xi}$. Suppose the distribution of $\boldsymbol{\xi}$ is $\psi(\boldsymbol{\xi})$. The noise distribution is generally chosen to have zero mean and to be uncorrelated, i.e.,

$$\int \xi_i \psi(\boldsymbol{\xi})\, d\xi = 0, \quad (3)$$

$$\int \xi_i \xi_j \psi(\boldsymbol{\xi})\, d\xi = \nu \delta_{ij}, \quad (4)$$

where ν is the variance of the noise distribution, and δ_{ij} is the delta function with $\delta_{ij} = 1$ when $i = j$ and $\delta_{ij} = 0$ otherwise.

We consider the following steps for generating new training samples by introducing additive noise:

1) Select a a training sample $\mathbf{z}_i$ randomly from the training set.

2) Draw a sample noise vector $\boldsymbol{\xi}_i$ from $\psi(\boldsymbol{\xi})$.

3) Set $\mathbf{z} = \mathbf{z}_i + \boldsymbol{\xi}_i$.

Thus, the distribution of a particular sample $\mathbf{z}$ generated from the training sample $\mathbf{z}_i$ is given by $\psi(\boldsymbol{\xi}_i) = \psi(\mathbf{z} - \mathbf{z}_i)$. Then the distribution of $\mathbf{z}$ generated from the entire training set is

$$\hat{g}(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \psi(\mathbf{z} - \mathbf{z}_i). \quad (5)$$

The above result can be viewed as a kernel density estimator of the true distribution of the data $g(\mathbf{z})$ [18]. The distribution of the noise $\psi(.)$ is the kernel function used in the estimator.

Substituting (5) into (2), we have

$$\begin{aligned} E= & \int L(\mathbf{z})\,\hat{g}(\mathbf{z})\,d\mathbf{z} \\ = & \frac{1}{n}\sum_{i=1}^{n}\int L(\mathbf{z})\,\psi(\mathbf{z}-\mathbf{z}_i)\,d\mathbf{z}. \end{aligned} \quad (6)$$

The result shown above in (6) can also be derived from a *known* convolution smoothing function $\psi(.)$ [9, 35]. This is achieved by approximating the cost function iteratively. While this provides further justification for the proposed measure, note that the difference between our derivations and those presented in [9, 35] is that we do not require to work with a known density or specific regression function.

We now reformulate (6) by approximating the loss function $L(.,.)$.

Let $\mathbf{z} - \mathbf{z}_i = \boldsymbol{\xi}_i$, then (6) is reformulated as

$$E=\frac{1}{n}\sum_{i=1}^{n}\int L(\mathbf{z}_i+\boldsymbol{\xi}_i)\,\psi(\boldsymbol{\xi}_i)\,d\boldsymbol{\xi}_i. \quad (7)$$

We expand $L(\mathbf{z}+\boldsymbol{\xi})$ as a Taylor series in powers of $\boldsymbol{\xi}$, i.e.,

$$L(\mathbf{z}+\boldsymbol{\xi})=L(\mathbf{z})+\sum_{i=1}^{m}\frac{\partial L(\mathbf{z})}{\partial z_i}\xi_i+\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\partial^2 L(\mathbf{z})}{\partial z_i \partial z_j}\xi_i\xi_j+\mathcal{O}\left(\boldsymbol{\xi}^3\right). \quad (8)$$

Assuming that the noise amplitude is small, the higher order term $\mathcal{O}\left(\boldsymbol{\xi}^3\right)$ can be neglected. Combining (8) with (7), (3) and (4), we obtain

$$\begin{aligned} E= & \frac{1}{n}\sum_{i=1}^{n}\left(L(\mathbf{z}_i)+\frac{1}{2}\nu\sum_{j=1}^{m}\frac{\partial^2 L(\mathbf{z}_i)}{\partial z_j^2}\right) \\ = & \frac{1}{n}\sum_{i=1}^{n}L(\mathbf{z}_i)+\frac{1}{2n}\nu\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\partial^2 L(\mathbf{z}_i)}{\partial z_j^2}. \end{aligned} \quad (9)$$

Let $L(\mathbf{z})$ be the quadratic loss, i.e., $L(\mathbf{z})=\frac{1}{2}\sum_{i=1}^{q}(y_i - f_i(\mathbf{x}))^2$ Then, as shown in Appendix A, we have

$$\sum_{j=1}^{m}\frac{\partial^2 L(\mathbf{z}_i)}{\partial z_j^2}=\sum_{k=1}^{q}\sum_{j=1}^{p}\left(\left(\frac{\partial f_k(\mathbf{x}_i)}{\partial x_{ij}}\right)^2+(f_k(\mathbf{x}_i)-y_{ik})\frac{\partial^2 f_k(\mathbf{x}_i)}{\partial x_{ij}^2}\right)+q, \quad (10)$$

where y_{ij} is the j^{th} entry of vector $\mathbf{y}_i$ and x_{ij} is the j^{th} entry of vector $\mathbf{x}_i$. Substituting (10) into (9), we have

$$E=E_f+\nu E_c, \quad (11)$$

with

$$E_f = \frac{1}{2n}\sum_{i=1}^{n} \| \mathbf{y}_i - \mathbf{f}(\mathbf{x}_i) \|_2^2 \quad (12)$$

and

$$E_c = \frac{1}{2n}\sum_{i=1}^{n}\sum_{k=1}^{q}\sum_{j=1}^{p}\left(\left(\frac{\partial f_k(\mathbf{x}_i)}{\partial x_{ij}}\right)^2 + (f_k(\mathbf{x}_i) - y_{ik})\frac{\partial^2 f_k(\mathbf{x}_i)}{\partial x_{ij}^2} + q\right). \quad (13)$$

Therefore, the generalization error consists of two terms. The first term E_f measures the discrepancy between the training data and the estimated model, i.e., the model fit. The second term E_c measures the roughness of the estimated function provided by the first and second derivatives of the function, i.e., the model complexity. It controls the smoothness of the function to prevent it from overfitting. The parameter ν controls the trade-off between the model fit and model complexity.

In order to minimize the generalization error E, we need to minimize both E_f and E_c. However, due to the bias and variance trade-off [16], a decrease in the model fit may result in an increase in the model complexity and vice-versa. The regularization parameter ν may achieve a balance between the model fit and complexity to some extent, however, there are two limitations for selecting ν to do model selection. First, a good ν should be chosen beforehand. A common way is to use cross-validation, but this suffers from several drawbacks as we discussed earlier. Second, note that our goal is to simultaneously minimize model fit and model complexity. An ideal solution is that we cannot further decrease one without increasing the other. This means that even when the appropriate ν is selected, minimizing E is not directly related to our goal. To solve these problems, we derive a multiobjective optimization approach in Section III. We first derive the kernel models for model fit E_f and model complexity E_c.

B. Model fit

We start by considering the standard linear regression model, $\mathbf{f}(\mathbf{x}) = \mathbf{W}^T\mathbf{x}$, where $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_q)$ is a $p \times q$ weight matrix, with $\mathbf{w}_i \in \mathbb{R}^p$. And, we assume all the vectors are standardized.

We can rewrite the above model as $f_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x},\ i=1,\ldots,q$. In kernel methods for regression, each sample $\mathbf{x}$ is mapped to $\varphi(\mathbf{x})$ in a reproducing kernel Hilbert space as $\phi(.): \mathbb{R}^p \rightarrow \mathcal{F}$. With this, we can write $f_i(\mathbf{x}) = \mathbf{w}_i^{\phi^T}\phi(\mathbf{x}),\ i=1,\ldots,q$. The Representer's Theorem [39] enables us to use $\mathbf{w}_i^{\phi} = \mathbf{\Phi}(\mathbf{X})\boldsymbol{\alpha}_i$, where $\Phi(\mathbf{X}) = (\varphi(\mathbf{x}_1), ..., \varphi(\mathbf{x}_n))$ and $\boldsymbol{\alpha}_i$ is an $n \times 1$ coefficient vector. Putting everything together, we get

$$\begin{aligned} f_i(\mathbf{x}) &= \boldsymbol{\alpha}_i^T\Phi(\mathbf{X})^T\phi(\mathbf{x}) = \boldsymbol{\alpha}_i^T\mathbf{k}(\mathbf{x}) \\ &= \sum_{j=1}^{n}\alpha_i^j\kappa\langle\mathbf{x}_j, \mathbf{x}\rangle, \quad i=1,\ldots,q, \end{aligned} \quad (14)$$

where α_i^j is the j^{th} element in $\boldsymbol{\alpha}_i$, and $\kappa\langle \mathbf{x}_j, \mathbf{x}\rangle$ is a kernel function on $\mathbf{x}_j$ and $\mathbf{x}$.

Using the results derived thus far, we can write E_f as

$$E_f = \sum_{i=1}^{q} (\tilde{\mathbf{y}}_i - \mathbf{K}\boldsymbol{\alpha}_i)^T (\tilde{\mathbf{y}}_i - \mathbf{K}\boldsymbol{\alpha}_i), \quad (15)$$

where $\mathbf{K} = \Phi(\mathbf{X})^T\Phi(\mathbf{X})$ is the $n \times n$ kernel matrix, $\tilde{\mathbf{y}}_i = (y_{1i}, \ldots, y_{ni})^T$ is an $n \times 1$ vector, and y_{ji} is the i^{th} entry of $\mathbf{y}_j$.

C. Roughness penalty in RBF

We now derive solutions of E_c for two of the most used kernel functions, the Radial Basis Function (RBF) and the polynomial kernels.

The RBF kernel is given by $\kappa\langle \mathbf{x}_i, \mathbf{x}_j\rangle = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$, where σ is the kernel parameter. Since $f_l(\mathbf{x}) = \sum_{m=1}^{n} \alpha_l^m \kappa\langle \mathbf{x}_m, \mathbf{x}\rangle$, the partial derivatives are given by

$$\frac{\partial f_l(\mathbf{x}_i)}{\partial x_{ij}} = \frac{1}{\sigma^2} \sum_{m=1}^{n} \alpha_l^m exp\left(-\frac{\|\mathbf{x}_m - \mathbf{x}_i\|^2}{2\sigma^2}\right)(x_{mj} - x_{ij}).$$

Writing this result in matrix form,

$$\sum_{j=1}^{p} \left(\frac{\partial f_l(\mathbf{x}_i)}{\partial x_{ij}}\right)^2 = \boldsymbol{\alpha}_l^R \mathbf{R}_i \boldsymbol{\alpha}_l,$$

where $\mathrm{R}_i = \frac{1}{\sigma^4}\mathbf{W}_i\mathbf{W}_i^T$, $\mathbf{W}_i$ is a $n \times p$ matrix with the j^{th} column equal to $\left(exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_i\|^2}{2\sigma^2}\right)(x_{1j} - x_{ij}), \ldots, exp\left(-\frac{\|\mathbf{x}_n - \mathbf{x}_i\|^2}{2\sigma^2}\right)(x_{nj} - x_{ij})\right)^T$.

And, the second partial derivatives (see Appendix B-A) are given by

$$\frac{\partial^2 f_l(\mathbf{x}_i)}{\partial x_{ij}^2} = \boldsymbol{\alpha}_l^T \mathbf{p}_{ij},$$

where $\mathbf{p}_{ij}$ is an $n \times 1$ vector whose m^{th} $(m \neq i)$ entry is $\frac{1}{\sigma^2} exp\left(-\frac{\|\mathbf{x}_m - \mathbf{x}_i\|^2}{2\sigma^2}\right)\left(\frac{(x_{mj} - x_{ij})^2}{\sigma^2} - 1\right)$ and i^{th} entry is 0. Then $\sum_{j=1}^{p} \frac{\partial^2 f_l(\mathbf{x}_i)}{\partial x_{ij}^2} = \boldsymbol{\alpha}_l^T \mathbf{p}_i$ where $\mathbf{p}_i = \sum_{j=1}^{p} \mathbf{p}_{ij}$. Thus,

$$\begin{aligned}\left(f_l\left(\mathbf{x}_i\right)-y_{il}\right)\sum_{j=1}^{p}\frac{\partial^2 f_l(\mathbf{x}_i)}{\partial x_{ij}^2} &= \left(\boldsymbol{\alpha}_l^T\mathbf{k}_i-y_{il}\right)\boldsymbol{\alpha}_l^T\mathbf{p}_i \\ &= \boldsymbol{\alpha}_l^T\left(\mathbf{k}_i\mathbf{p}_i^T\right)\boldsymbol{\alpha}_l-y_{il}\mathbf{p}_i^T\boldsymbol{\alpha}_l,\end{aligned}$$

where $\mathbf{k}_i=\left(\kappa\left\langle\mathbf{x}_1,\mathbf{x}_i\right\rangle,\ldots,\kappa\left\langle\mathbf{x}_n,\mathbf{x}_i\right\rangle\right)^T$.

Using the above results, we can define the roughness penalty function in the RBF kernel space as

$$E_c=\sum_{l=1}^{q}\left(\boldsymbol{\alpha}_l^T\mathbf{M}\boldsymbol{\alpha}_l-\mathbf{q}_l^T\boldsymbol{\alpha}_l+n\right), \quad (16)$$

where $\mathbf{M}=\frac{1}{2n}\sum_{i=1}^{n}\left(\mathbf{R}_i+\mathbf{k}_i\mathbf{p}_i^T\right)$, and $\mathbf{q}_l=\frac{1}{2n}\sum_{i=1}^{n}y_{il}\mathbf{p}_i$.

D. Polynomial kernel

A polynomial kernel of degree *d* is given by $\kappa\left\langle\mathbf{x}_i,\mathbf{x}_j\right\rangle=\left(\mathbf{x}_i^T\mathbf{x}_j+1\right)^d$. Its partial derivatives are,

$$\frac{\partial f_l\left(\mathbf{x}_i\right)}{\partial x_{ij}}=\sum_{m=1,m\neq i}^{n}\alpha_l^m d\left(\mathbf{x}_m^T\mathbf{x}_i+1\right)^{d-1}x_{mj}+2\alpha_l^i d\left(\mathbf{x}_i^T\mathbf{x}_i+1\right)^{d-1}x_{ij}.$$

We can write the above result in matrix form as

$$\sum_{j=1}^{p}\left(\frac{\partial f_l\left(\mathbf{x}_i\right)}{\partial x_{ij}}\right)^2=\boldsymbol{\alpha}_l^T\mathbf{B}_i\boldsymbol{\alpha}_l,$$

where $\mathbf{B}_i=d\mathbf{C}_i\mathbf{C}_i^T$, $\mathbf{C}_i$ is a $n\times p$ matrix with i^{th} column equal to $\left(\left(\mathbf{x}_1^T\mathbf{x}_i+1\right)^{d-1}x_{1j},\ldots,2\left(\mathbf{x}_i^T\mathbf{x}_i+1\right)^{d-1}x_{ij},\ldots,\left(\mathbf{x}_n^T\mathbf{x}_i+1\right)^{d-1}x_{nj}\right)^T$.

The second partial derivatives (see Appendix B-B) are

$$\frac{\partial^2 f_l\left(\mathbf{x}_i\right)}{\partial x_{ij}^2}=\boldsymbol{\alpha}_l^T\mathbf{g}_{ij},$$

where $\mathbf{g}_{ij}$ is a $n\times 1$ vector whose m^{th} $(m\neq i)$ entry is $d\left(d-1\right)\left(\mathbf{x}_m^T\mathbf{x}_i+1\right)\left(d-2\right)x_{mj}^2$ and the i^{th} entry is $d\left(\mathbf{x}_m^T\mathbf{x}_i+1\right)^{d-2}\left(3\left(d-1\right)x_{ij}^2+2\left(\mathbf{x}_i^T\mathbf{x}_i+1\right)\right)$. Then, $\sum_{j=1}^{p}\frac{\partial^2 f_l(\mathbf{x}_i)}{\partial x_{ij}^2}=\boldsymbol{\alpha}_l^T\mathbf{g}_i$, where $\mathbf{g}_i=\sum_{j=1}^{p}\mathbf{g}_{ij}$.

Thus,

$$\begin{aligned}\left(f_l\left(\mathbf{x}_i\right)-y_{il}\right)\sum_{j=1}^{p}\frac{\partial^2 f_l(\mathbf{x}_i)}{\partial x_{ij}^2} &= \left(\boldsymbol{\alpha}_l^T\mathbf{k}_i-y_{il}\right)\boldsymbol{\alpha}_l^T\mathbf{g}_i \\ &= \boldsymbol{\alpha}_l^T\left(\mathbf{k}_i\mathbf{g}_i^T\right)\boldsymbol{\alpha}_l-y_{il}\mathbf{g}_i^T\boldsymbol{\alpha}_l,\end{aligned}$$

Using the derivations above, the roughness function for the polynomial kernel can be written as

$$E_c=\sum_{l=1}^{q}\left(\boldsymbol{\alpha}_l^T\mathbf{N}\boldsymbol{\alpha}_l-\mathbf{u}_l^T\boldsymbol{\alpha}_l+n\right), \quad (17)$$

where $\mathbf{N}=\frac{1}{2n}\sum_{i=1}^{n}\left(\mathbf{B}_i+\mathbf{k}_i\mathbf{g}_i^T\right)$, and $\mathbf{u}_l=\frac{1}{2n}\sum_{i=1}^{n}y_{il}\mathbf{g}_i$.

E. Comparison with other complexity measure

Thus far, we have introduced a new model complexity measure E_c. E_c is related to the derivatives of the regressed function **f**(**x**). A commonly seen alternative in the literature is the norm of the regression function instead. The L_2 norm in the kernel Hilbert space being the most common of norms used in this approach [39]. This section provides a theoretical comparison between the approach derived in this paper and this classical L_2 norm alternative. In particular, we show that the L_2 norm does not penalize the high frequencies of the regression function, whereas the proposed criterion emphasizes smoothness by penalizing the high frequency components of this function.

To formally prove the above result, we write generalized Fourier series of $f(x)$,

$$f\left(x\right)=\sum_{k=0}^{\infty}a_k\varphi_k\left(x\right),$$

where $\{\varphi_k\left(x\right)\}_{k=0}^{\infty}$ forms a complete orthonormal basis and a_k are the corresponding coefficients. A commonly used complete orthonormal basis is $\{sin\ kx, cos\ kx\}_{k=0}^{\infty}$ in $[-\pi, \pi]$, with k the index of the frequency component. Using this basis set, $f(x)$ can be written as

$$f\left(x\right)=a_0+\sum_{k=1}^{\infty}\left(a_k\ sin\ kx+b_k\ cos\ kx\right), \quad (18)$$

where a_k and b_k are the coefficients of each frequency components.

Let $\|f\|_2$ be the L_2 norm in the reproducing kernel Hilbert space, then

$$\begin{aligned} \|f\|_2^2 &= \int |f(x)|^2 dx \\ &= \int_{-\pi}^{\pi} \left(a_0 + \sum_{k=1}^{\infty} (a_k \sin kx + b_k \cos kx) \right)^2 dx \\ &= 2\pi a_0 + \pi \sum_{k=1}^{\infty} (a_k^2 + b_k^2). \end{aligned} \quad (19)$$

Note that in this case, all the coefficient are equal, regardless of the frequency component.

The complexity measure derived in the present paper and given in (13) can be reformulated as

$$E_c = \int \left(\left(\frac{\partial f(x)}{\partial x} \right)^2 + (f(x) - y) \frac{\partial^2 f(x)}{\partial x^2} \right) dx, \quad (20)$$

where we have neglected the constant p^{-1}.

Moreover, remember for (11) that the generalization error E can be expressed as $f(x) = y + \mathcal{O}(\nu)$ [4]. Hence, substituting (18) into (20), yields

$$\begin{aligned} E_c &= \int \left(\sum_{k=1}^{\infty} k (-a_k \sin kx + b_k \cos kx) \right)^2 dx \\ &= \pi \sum_{k=1}^{\infty} k^2 (a_k^2 + b_k^2). \end{aligned} \quad (21)$$

Compared to the L_2 norm result shown in (19), the complexity measure (21) of the proposed approach penalizes the higher frequency components of the regressed function. This is due to the squared of the index of the frequency component seen in (21). By emphasizing lower frequencies, the proposed criterion will generally select smoother functions than those selected by the L_2 norm method.

A numerical comparison is provided in Section V. To do this, we will need the explicit equation of the L_2 norm of the regression function f in the kernel space. This is given by,

$$\begin{aligned} \|\mathbf{f}\|_2^2 &= \sum_{i=1}^{q} \|f_i\|_2^2 = \sum_{i=1}^{q} \sum_{j=1}^{n} \sum_{k=1}^{n} \alpha_i^j \alpha_i^k \kappa \langle \cdot, \mathbf{x}_j \rangle \kappa \langle \cdot, \mathbf{x}_k \rangle \\ &= \sum_{i=1}^{q} \sum_{j=1}^{n} \sum_{k=1}^{n} \alpha_i^j \alpha_i^k \kappa \langle \mathbf{x}_j, \mathbf{x}_k \rangle \\ &= \sum_{i=1}^{q} \boldsymbol{\alpha}_i^T \mathbf{K} \boldsymbol{\alpha}_i. \end{aligned} \quad (22)$$

III. Multiobjective Optimization

The parameters in kernel approaches in regression can now be optimized by simultaneously minimizing E_f and E_c of the corresponding fitting function described in the preceding section. Of course, in general, the global minima of these two functions are not the same. For instance, a decrease in the fitting error may lead to an increase in the roughness of the function, and vice-versa. This trade-off is depicted in Figure 1. In the plots in this figure, we

show the performance of the two criteria with respect to the their corresponding parameters, i.e., the kernel parameter σ and the regularization parameter λ in the case of Kernel Ridge Regression (KRR)[16] with an RBF kernel. As can be observed in the figure, the criteria do not share a common global minimum. To resolve this problem, we now derive a multiobjective optimization approach.

A. Pareto-Optimality

As its name implies, multiobjective optimization (MOP) is concerned with the simultaneous optimization of more than one objective function. More formally, MOP can be stated as follows,

$$\begin{array}{ll} \underset{\theta}{\text{minimize}} & u_1(\boldsymbol{\theta}), u_2(\theta), \ldots, u_k(\boldsymbol{\theta}) \\ \text{subject to} & \boldsymbol{\theta} \in \mathbf{S}, \end{array} \tag{23}$$

where we have k objective functions $u_i: \mathbb{R}^p \rightarrow \mathbb{R}$, and $S \subset \mathbb{R}^p$ is the set of possible vectors. Denote the vector of objective functions by $\mathbf{z} = \mathbf{u}(\theta) = (u_1(\theta), u_2(\theta), \ldots, u_k(\theta))^T$, and the decision vectors as $\theta = (\theta_1, \theta_2, \ldots, \theta_p)^T$.

The goal of MOP is to find that θ^* which simultaneously minimizes all $u_j(.)$. If all functions shared a common minimum, the problem would be trivial. In general, however, the objective functions contradict one another. This means that minimizing one function can increase the value of the others. Hence, a compromise solution is needed to attain a maximal agreement of all the objective functions [25]. The solutions of the MOP problem are called *Pareto-optimal* solutions. To provide a formal definition, let us first state another important concept.

Definition 1—A decision vector θ_1 is said to *dominate* θ_2 if $u_i(\theta_1) \leq u_i(\theta_2)$ for all $i = 1, \ldots, k$ and $u_j(\theta_1) < u_j(\theta_2)$ for at least one index j.

This definition now allows us to give the following formal presentation of Pareto-optimality.

Definition 2—A decision vector $\theta^* \in S$ is *Pareto-optimal* if there does not exist another decision vector $\theta \in S$ for which $u_i(\theta) \leq u_i(\theta^*)$ for all $i = 1, \ldots, k$ and $u_j(\theta) < u_j(\theta^*)$ for at least one index j.

In other words, a Pareto-optimal solution is not dominated by any other decision vector. Similarly, an objective vector $\mathbf{z}^* \in Z (= \mathbf{u}(S))$ is called Pareto-optimal if the decision vector corresponding to it is Pareto-optimal. We can see that such a vector is the one where none of the components can be improved without deteriorating one or more of the others. In most problems, there will be many Pareto-optimal solutions. This set of Pareto-optimal solutions is called the *Pareto-optimal set* or *Pareto-frontier*.

B. The ε-constraint approach

One classical method to find the Pareto-optimal solutions is the ε-constraint approach [15]. In this case, one of the objective functions is optimized while the others are considered as constraints. This is done by defining constraints as upper-bounds of their objective functions. Therefore, the problem to be solved can be formulated as follows,

$$\begin{aligned} &\underset{\boldsymbol{\theta}}{arg\,min} && u_l(\boldsymbol{\theta}) \\ &\text{subject to} && u_j(\boldsymbol{\theta}) \le \varepsilon_j, \quad \text{for all} \quad j=1,\ldots,k, \quad j \neq l, \quad \boldsymbol{\theta} \in \mathbf{S}, \end{aligned} \tag{24}$$

where $l \in \{1, ..., k\}$.

Figure 2 demonstrates the idea behind this approach. In this figure, we show a bi-objective example, $k = 2$. The Pareto-optimal solution θ is determined by minimizing $\mathbf{u}_1$ provided that that $\mathbf{u}_2$ is upper-bounded by ε.

Before exploring the Pareto-optimality of the ε-constraint method, let us look at a weaker definition of the term.

Definition 3—A decision vector $\theta^* \in S$ is *weakly Pareto-optimal* if there does not exist another decision vector $\theta \in S$ such that $u_i(\theta) < u_i(\theta^*)$ for all $i = 1, ..., k$.

From the above definition, we can see that the Pareto-optimal set is a subset of the weakly Pareto-optimal set and that a weakly Pareto-optimal solution may be dominated by any Pareto-optimal solution.

It has been shown [25] that the solution of the ε-constraint method defined in (24) is weakly Pareto-optimal. This means that the solution to (24) cannot be guaranteed to be Pareto-optimal. Although the solution is determined by the prespecified upper-bounds ε_j's and some ε_j's may lead to Pareto-optimal solutions, in practice, we do not know how to choose ε_j's to achieve the Pareto-optimal solutions. In the following, we propose a modified version of this method and prove that the solution to this modified approach is guaranteed to be Pareto-optimal.

C. The modified ε-constraint

The main idea of our approach is to reformulate the constraints in (24) as equalities. This can be achieved if these equalities are multiplied by a scaler smaller than or equal to s on the right. Formally, $u_j(\theta) = h_j\varepsilon_j$, $h_j \in [0, s]$, for all $j = 1, ..., k, j \neq l$. Let $\mathbf{h} = (h_1, ..., h_{l-1}, h_{l+1}, ..., h_l)^T$. Then, the modified ε-constraint method is given by

$$\begin{aligned} &\underset{\boldsymbol{\theta},\mathbf{h}}{arg\,min} && u_l(\boldsymbol{\theta}) + s\sum_{j=1,j\neq l}^{k} h_j \\ &\text{subject to} && u_j(\boldsymbol{\theta}) = h_j\varepsilon_j, \quad \text{for all} \quad j=1,\ldots,k, \quad j \neq l, \quad 0 \le h_j \le s, \quad \text{for all} \quad j=1,\ldots,k, \quad j \neq l, \quad \boldsymbol{\theta} \in \mathbf{S}, \end{aligned} \tag{25}$$

where s is a positive constant. We can now prove the Pareto-optimality of (25).

Theorem 4—Select a small scalar s satisfying $s\sum_{j=1,j\neq l}^{k} h_j^* \le u_l(\mathbf{x}) - u_l(\mathbf{x}^*)$, where $\theta^* \in S$ and $\mathbf{h}^*$ are the solutions of (25). Then, $\boldsymbol{\theta}^*$ is Pareto-optimal for any given upper-bound vector $\boldsymbol{\varepsilon} = (\varepsilon_1, ..., \varepsilon_{l-1}, \varepsilon_{l+1}, ..., \varepsilon_k)^T$.

Proof—Let $\boldsymbol{\theta}^* \in S$ and $\mathbf{h}^*$ be a solution of (25). Since $s\sum_{j=1,j\neq l}^{k} h_j^* \le u_l(\boldsymbol{\theta}) - u_l(\boldsymbol{\theta}^*)$, we have $u_l(\boldsymbol{\theta}^*) \le u_l(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in S$ when $u_j(\boldsymbol{\theta}^*) = h_j\varepsilon_j$ for every $j = 1,..., k, j \neq l$. Let us assume

that θ^* is not Pareto-optimal. In this case, there exists a vector $\theta^o \in S$ such that $u_i(\theta^o) \leq u_i(\theta^*)$ for all $i = 1, ..., k$ and $u_j(\theta^o) < u_j(\theta^*)$ for at least one index j.

If $j = l$, this means that $u_l(\theta^o) < u_l(\theta^*)$. Here we have a contradiction with the fact that $u_l(\theta^*) \leq u_l(\theta)$ for all $\theta \in S$.

If $j \neq l$, then $u_l(\theta^o) \leq u_l(\theta^*)$, $u_j(\theta^o) < u_j(\theta^*) = h_j\varepsilon_j$ and $u_i(\theta^o) \leq u_i(\theta^*) = h_i\varepsilon_i$ for all $i \neq j$ and $i \neq l$. Denote $u_i(\theta^o) = h_i^o\varepsilon_i$ for all $i \neq l$. Then, we have $l - 1$ inequalities $h_i^o\varepsilon_i \leq h_i\varepsilon_i$ with at least one strict inequality $h_j^o\varepsilon_j < h_j\varepsilon_j$. Canceling out ε_i on each of the inequality and taking their sum, yields $\sum_{j=1,j\neq l}^{k} h_j^o < \sum_{j=1,j\neq l}^{k} h_j$. This contradicts the fact that the solution to (25) minimizes $\sum_{j=1,j\neq l}^{k} h_j$.

We can demonstrate the utility of this modified ε-constraint method in the following two examples. In our first example, the objective functions are given by

$$u_1(x) = \begin{cases} 1 & x \leq 1 \\ x^2 & \text{otherwise} \end{cases}$$

and $u_2(x) = (x - 5)^2$. In our second example, the two functions are given by $u_1'(x) = 1 - e^{-(x+1)^2}$ and

$$u_2'(x) = \begin{cases} 1 - e^{-(x-2)^2} & x \leq 0.5 \\ 1 - e^{-2.25} & \text{otherwise.} \end{cases}$$

In both these examples, we compare the performance of the proposed modified ε-constraint approach and the ε-constraint method. This is illustrated in Figure 3. In these figures, the blue stars denote the objective vectors and the red circles represent the solution vectors given by each of the two methods. We see that in Figure 3a and 3c, the original ε-constraint method includes the weakly Pareto-optimal solutions, whereas in Figure 3b and 3d the proposed modified approach provides the Pareto-optimal solutions.

Using the solution defined above, we can formulate the parameter optimization problem as follows,

$$\begin{aligned} \underset{\boldsymbol{\theta},\mathbf{h}}{arg\,min} \quad & E_f(\boldsymbol{\theta}) + sh \\ \text{subject to} \quad & E_c(\boldsymbol{\theta}) = h\varepsilon \\ & 0 \leq h \leq s. \end{aligned} \tag{26}$$

Note that given different ε's, we may have different Pareto-optimal solutions. In our parameter optimization problem, we only need one Pareto-optimal solution. Hence, our next goal is to define a mechanism to determine an appropriate value for ε.

To resolve this problem, we select ε such that the corresponding Pareto-optimal objective vector is as close to the ideal point as possible. Specifically, let $\boldsymbol{\theta}_\varepsilon$ be a Pareto-optimal solution given ε, then the optimal ε is

$$\varepsilon^* = arg\ \min_{\varepsilon} \left[w_f \left(E_f\left(\boldsymbol{\theta}_\varepsilon\right) - z_f^* \right)^2 + w_c \left(E_c\left(\boldsymbol{\theta}_\varepsilon\right) - z_c^* \right)^2 \right], \quad (27)$$

where z_f^* and z_c^* are the ideal vectors of $E_f(\boldsymbol{\theta})$ and $E_c(\boldsymbol{\theta})$, respectively, and w_f, w_c are the weights associated to each of the objective functions. The incorporation of these weights can drive the optimization to favor one objective function over the other. If $E_f(\boldsymbol{\theta}_\varepsilon)$ (or $E_c(\boldsymbol{\theta}_\varepsilon)$) is close to its ideal value z_f^* (z_c^*), then w_f (w_c) should be relatively small. But if $E_f(\boldsymbol{\theta}_\varepsilon)$ ($E_c(\boldsymbol{\theta}_\varepsilon)$) is far apart from it ideal value z_f^* (z_c^*), then w_f (w_c) should be large. This can be formally stated as follows,

$$\begin{aligned} w_f &= \left| E_f\left(\boldsymbol{\theta}_{\varepsilon_0}\right) - z_f^* \right|^2, \\ w_c &= \left| E_c\left(\boldsymbol{\theta}_{\varepsilon_0}\right) - z_c^* \right|^2, \end{aligned} \quad (28)$$

where ε_0 is the initialization for ε. The proposed modified ε-constraint approach is summarized in Algorithm 1.

D. Alternative Optimization Approaches

Thus far, we have derived a MOP approach for model selection based on Pareto-optimality. The most pressing question for us is to show that this derived solution yields lower prediction errors than simpler, more straight forward approaches. Two such criteria are the sum and product of the two terms to be minimized [47], given by

$$Q^{sum}\left(\boldsymbol{\theta}\right) = E_f\left(\boldsymbol{\theta}\right) + \nu E_c\left(\boldsymbol{\theta}\right). \quad (29)$$

and

$$Q^{pro}\left(\boldsymbol{\theta}\right) = E_f\left(\boldsymbol{\theta}\right) E_c(\boldsymbol{\theta})^{\gamma}, \quad (30)$$

where ν and γ are regularization parameters needed to be selected. Note that minimizing (30) is equivalent to minimizing

$$lg\, Q^{pro}\left(\boldsymbol{\theta}\right) = lg\, E_f\left(\boldsymbol{\theta}\right) + \gamma\, lg\, E_c\left(\boldsymbol{\theta}\right). \quad (31)$$

which is the logarithm of (30). We could use cross-validation to select the regularization parameters ν and γ. Experimental results comparing these two alternative optimization approaches with the proposed approach will be given in the experiments section.

IV. Applications to Regression

Let us derive two kernel-based regression approaches using the kernels and MOP criteria derived above. In particular, we use our derived results in Kernel Ridge Regression (KRR) [16] and Kernel Principal Component Regression (KPCR) [31].

A. Kernel Ridge Regression

Ridge regression (RR) is a penalized version of the ordinary least squares (OLS) solution. More specifically, RR regularizes the OLS solution with a penalty on the norm of the weight factor. This regularization is used to avoid overfitting. Formally, RR is defined as

$$\mathbf{w}_i=\left(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I}_p\right)^{-1}\mathbf{X}\tilde{\mathbf{y}}_i,\quad i=1,\ldots,q, \quad (32)$$

where $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$, $\mathbf{I}_p$ is the $p \times p$ identity matrix, $\tilde{\mathbf{y}}_i=(y_{1i},\ldots,y_{ni})^T$, and λ is the regularization parameter.

We can now extend the above solution using the kernel trick. The resulting method is know as Kernel Ridge Regression (KRR), and is given by

$$\boldsymbol{\alpha}_i=(\mathbf{K}+\lambda\mathbf{I}_n)^{-1}\tilde{\mathbf{y}}_i,\quad i=1,\ldots,q, \quad (33)$$

where, as above, $\mathbf{K}$ is the kernel matrix.

In KRR, there are two parameters to optimize: the kernel parameter (e.g., σ in the RBF kernel) and the regularization parameter λ. In the following, we derive a gradient descent method to simultaneously optimize the two.

Since both, the residual sum of squares term E_R and the curvature term E_C, are involved in our parameter optimization problem, we need to derive the gradient of these terms with respect to their parameters.

We start with the derivations of the RBF kernel. In this case, we have

$$\begin{aligned}\frac{\partial E_c}{\partial\sigma} &= \frac{\partial\sum_{i=1}^{q}(\tilde{\mathbf{y}}_i-\mathbf{K}\boldsymbol{\alpha}_i)^T(\tilde{\mathbf{y}}_i-\mathbf{K}\boldsymbol{\alpha}_i)}{\partial\sigma}\\ &= 2\sum_{i=1}^{q}(\tilde{\mathbf{y}}_i-\mathbf{K}\boldsymbol{\alpha}_i)^T\frac{\partial(\tilde{\mathbf{y}}_i-\mathbf{K}\boldsymbol{\alpha}_i)}{\partial\sigma}\\ &= -2\sum_{i=1}^{q}(\tilde{\mathbf{y}}_i-\mathbf{K}\boldsymbol{\alpha}_i)^T\left(\frac{\partial\mathbf{K}}{\partial\sigma}\boldsymbol{\alpha}_i+\mathbf{K}\frac{\partial\boldsymbol{\alpha}_i}{\partial\sigma}\right),\end{aligned}$$

where $\frac{\partial\mathbf{K}}{\partial\sigma}=\frac{1}{\sigma^3}\mathbf{K}\circ\mathbf{D}$, $\bigcirc$ defines the Hadamard product of two matrices of the same dimensions, i.e., $(\mathbf{A}\circ\mathbf{B})_{ij}=\mathbf{A}_{ij}\mathbf{B}_{ij}$, with $\mathbf{A}_{ij}$ denoting the $(i, j)^{th}$ entry of matrix $\mathbf{A}$. $\mathbf{D}=\left[\|\mathbf{x}_i-\mathbf{x}_j\|^2\right]_{i,j=1,\ldots,n}$ is the matrix of pairwise sample distances, and $\frac{\partial\boldsymbol{\alpha}_i}{\partial\sigma}=\frac{\partial(\mathbf{K}+\lambda\mathbf{I}_n)^{-1}}{\partial\sigma}\tilde{\mathbf{y}}_i=-(\mathbf{K}+\lambda\mathbf{I}_n)^{-1}\frac{\partial\mathbf{K}}{\partial\sigma}(\mathbf{K}+\lambda\mathbf{I}_n)^{-1}\tilde{\mathbf{y}}_i=-(\mathbf{K}+\lambda\mathbf{I}_n)^{-1}\frac{\partial\mathbf{K}}{\partial\sigma}\boldsymbol{\alpha}_i$. And,

$$\begin{aligned}\frac{\partial E_c}{\partial\sigma} &= \sum_{l=1}^{q}\frac{\partial\left(\boldsymbol{\alpha}_l^T\mathbf{M}\boldsymbol{\alpha}_l-\mathbf{q}_l^T\boldsymbol{\alpha}_l+n\right)}{\partial\sigma}\\ &= \frac{1}{2n}\sum_{i=1}^{n}\sum_{l=1}^{q}\left(\frac{\partial\left(\boldsymbol{\alpha}_l^T\mathbf{R}_i\boldsymbol{\alpha}_l\right)}{\partial\sigma}+\frac{\partial\left(\mathbf{k}_i^T\boldsymbol{\alpha}_l\right)}{\partial\sigma}\mathbf{p}_i^T\boldsymbol{\alpha}_l\left(\mathbf{k}_i^T\boldsymbol{\alpha}_l-y_{il}\right)\frac{\partial\left(\mathbf{p}_i^T\boldsymbol{\alpha}_l\right)}{\partial\sigma}\right),\end{aligned}$$

where

$$\frac{\partial\left(\boldsymbol{\alpha}_l^T\mathbf{R}_i\boldsymbol{\alpha}_l\right)}{\partial\sigma}=\frac{2\boldsymbol{\alpha}_l^T\mathbf{W}_i\left(\frac{\partial\mathbf{W}_i^T}{\partial\sigma}\boldsymbol{\alpha}_l+\mathbf{W}_i^T\frac{\partial\boldsymbol{\alpha}_l}{\partial\sigma}\right)-4\sigma^3\boldsymbol{\alpha}_l^T\mathbf{R}_i\boldsymbol{\alpha}_l}{\sigma^4},$$

$\frac{\partial\mathbf{W}_i^T}{\partial\sigma}$ is a $n\times p$ matrix whose $(j,k)^{th}$ entry is $\frac{\|\mathbf{x}_j-\mathbf{x}_i\|^2}{\sigma^3}exp\left(-\frac{\|\mathbf{x}_j-\mathbf{x}_i\|^2}{2\sigma^2}\right)(x_{jk}-x_{ik})$, and

$$\begin{aligned}\frac{\partial\left(\mathbf{k}_i^T\boldsymbol{\alpha}_l\right)}{\partial\sigma}&=\frac{\partial\mathbf{k}_i^T}{\partial\sigma}\boldsymbol{\alpha}_l+\mathbf{k}_i^T\frac{\partial\boldsymbol{\alpha}_l}{\partial\sigma},\\ \frac{\partial\left(\mathbf{p}_i^T\boldsymbol{\alpha}_l\right)}{\partial\sigma}&=\frac{\partial\mathbf{p}_i^T}{\partial\sigma}\boldsymbol{\alpha}_l+\mathbf{p}_i^T\frac{\partial\boldsymbol{\alpha}_l}{\partial\sigma},\end{aligned}$$

$\frac{\partial\mathbf{k}_i}{\partial\sigma}$ is the i^{th} column of $\frac{\partial\mathbf{K}}{\partial\sigma}$, $\frac{\partial\mathbf{p}_i^T}{\partial\sigma}=\sum_{j=1}^{p}\frac{\partial\mathbf{p}_{ij}^T}{\partial\sigma}$, $\frac{\partial\mathbf{p}_{ij}^T}{\partial\sigma}$ is a $n\times 1$ vector whose m^{th} $(m\neq i)$ entry is $\frac{1}{\sigma^3}exp\left(-\frac{\|\mathbf{x}_m-\mathbf{x}_i\|^2}{2\sigma^2}\right)\left[\left(\frac{1}{\sigma^2}\|\mathbf{x}_m-\mathbf{x}_i\|^2\right)\left(\frac{\left(x_{mj}-x_{ij}\right)^2}{\sigma^2}-1\right)-\frac{2}{\sigma^2}\left(x_{mj}-x_{ij}\right)^2\right]$ and i^{th} entry is 0.

Seemingly, deriving with respect to the regularization parameter λ, yields

$$\begin{aligned}\frac{\partial E_R}{\partial\lambda}&=2\sum_{i=1}^{q}\left(\tilde{\mathbf{y}}_i-\mathbf{K}\boldsymbol{\alpha}_i\right)^T\frac{\partial\left(\tilde{\mathbf{y}}_i-\mathbf{K}\boldsymbol{\alpha}_i\right)}{\partial\lambda}\\ &=-2\sum_{i=1}^{q}\left(\tilde{\mathbf{y}}_i-\mathbf{K}\boldsymbol{\alpha}_i\right)^T\mathbf{K}\frac{\partial\boldsymbol{\alpha}_i}{\partial\lambda},\end{aligned}$$

where $\frac{\partial\boldsymbol{\alpha}_i}{\partial\lambda}=\frac{\partial(\mathbf{K}+\lambda\mathbf{I}_n)^{-1}}{\partial\lambda}\tilde{\mathbf{y}}_i=-\left(\mathbf{K}+\lambda\mathbf{I}_n\right)^{-1}\left(\mathbf{K}+\lambda\mathbf{I}_n\right)^{-1}\tilde{\mathbf{y}}_i=-\left(\mathbf{K}+\lambda\mathbf{I}_n\right)^{-1}\boldsymbol{\alpha}_i$ And,

$$\begin{aligned}\frac{\partial E_c}{\partial\lambda}&=\sum_{l=1}^{q}\frac{\partial\left(\boldsymbol{\alpha}_l^T\mathbf{M}\boldsymbol{\alpha}_l-\mathbf{q}_l^T\boldsymbol{\alpha}_l+n\right)}{\partial\lambda}\\ &=\sum_{l=1}^{q}\left(2\boldsymbol{\alpha}_l^T\mathbf{M}\frac{\partial\boldsymbol{\alpha}_l}{\partial\lambda}-\mathbf{q}_l^T\frac{\partial\boldsymbol{\alpha}_l}{\partial\lambda}\right)\end{aligned}$$

When using the polynomial kernel, we cannot employ a gradient descent technique for finding the optimal value of d, because this is discrete. Thus, we will have to try all possible discrete values of d (within a given range) and select the degree yielding the smallest error. The derivations of E_f with respect to λ are the same for any kernel, and

$\frac{\partial E_c}{\partial\lambda}=\sum_{l=1}^{q}\left(2\boldsymbol{\alpha}_l^T\mathbf{N}\frac{\partial\boldsymbol{\alpha}_l}{\partial\lambda}-\mathbf{u}_l^T\frac{\partial\boldsymbol{\alpha}_l}{\partial\lambda}\right)$.

B. Kernel Principal Component Regression

Solving an overdetermined set of equations is a general problem in pattern recognition. The problem is well studied when there are no collinearities (i.e., close to linear relationships among variables), but special algorithms are needed to deal with them. Principal Component Regression (PCR) is a regression approach designed to deal with collinearities in the exploratory variables. Instead of using the original predictor variables, a subset of principal components of these are selected. By deleting the principal components with small variances, a more stable estimate of the coefficient $\{\mathbf{w}_i\}_{i=1...,q}$ can be obtained. In this way,

the large variances of $\{\mathbf{w}_i\}i=1,...,q$, which were caused by multicollinearities, will be greatly reduced. More formally,

$$\mathbf{w}_i=\sum_{j=1}^{m}\frac{1}{l_j}\mathbf{a}_j\mathbf{a}_j^T\mathbf{X}\tilde{\mathbf{y}}_i,\quad i=1,\ldots,q, \quad (34)$$

where $\mathbf{a}_i$ is the eigenvector of the covariance matrix associated to the i^{th} largest eigenvalue.

The above formulation can once again be calculated in the kernel space as,

$$\boldsymbol{\alpha}_i=\sum_{j=1}^{m}\frac{1}{\lambda_j}\mathbf{v}_j\mathbf{v}_j^T\tilde{\mathbf{y}}_i,\quad i=1,\ldots,q, \quad (35)$$

where $\mathbf{v}_i$ is the eigenvector of the centered kernel matrix $\widetilde{\mathbf{K}}$ associated to the i^{th} largest eigenvalue λ_i. This algorithm is known as Kernel Principal Component Regression (KPCR).

In KPCR, we need to optimize two parameters – the kernel parameter and the number of eigenvectors m we want to keep. Since m is discrete, the cost function with respect to m is non-differentiable. But testing all possible value for m is computationally expensive, because the range of m is dependent on the size of the training set. Here, we present an alternative approach to select the optimal subset. The basic idea is to use the percentage of the variance $r=\frac{\sum_{i=1}^{m}\lambda_i}{\sum_{i=1}^{t}\lambda_i}$, t is the rank of $\widetilde{\mathbf{K}}$. Note that r can now change continuously (from 0 to 1) and can thus be incorporated in a gradient descent framework.

Since KPCR differs from KRR in the solution vector $\{\boldsymbol{\alpha}_i\}i=1,...,q$, we need to derive $\boldsymbol{\alpha}_i$. The derivative with respect to σ is given by,

$$\begin{aligned}\frac{\partial\boldsymbol{\alpha}_i}{\partial\sigma}&=\sum_{j=1}^{m}\frac{\partial\frac{1}{\lambda_j}\mathbf{v}_j\mathbf{v}_j^T}{\partial\sigma}\tilde{\mathbf{y}}_i\\&=\sum_{j=1}^{m}\left(-\frac{1}{\lambda_i^2}\frac{\partial\lambda_i}{\partial\sigma}\mathbf{v}_i\mathbf{v}_i^T+\frac{1}{\lambda_j}\frac{\partial\mathbf{v}_i}{\partial\sigma}\mathbf{v}_i^T+\frac{1}{\lambda_j}\mathbf{v}_i\frac{\partial\mathbf{v}_i^T}{\partial\sigma}\right)\tilde{\mathbf{y}}_i,\end{aligned}$$

where $\frac{\partial\lambda_i}{\partial\sigma}=\mathbf{v}_i^T\frac{\partial\mathbf{K}}{\partial\sigma}\mathbf{v}_i$, $\frac{\partial\mathbf{v}_i}{\partial\sigma}=-\left(\mathbf{K}-\lambda_i\mathbf{I}_d\right)^{+}\frac{\partial\mathbf{K}}{\partial\sigma}\mathbf{v}_i$ [22], and $\mathbf{A}^{+}$ is the pseudoinverse of the matrix $\mathbf{A}$.

The partial derivative with respect to r cannot be given, because an explicit definition of $\boldsymbol{\alpha}_i$ as a function of r does not exist. We resolve this issue by deriving an approximation to $\frac{\partial\boldsymbol{\alpha}_i}{\partial r}$ using a Taylor expansion. That is,

$$\begin{aligned}\boldsymbol{\alpha}_i(r+\Delta r)&=\boldsymbol{\alpha}_i(r)+\Delta r\boldsymbol{\alpha}_i'(r)+\frac{\Delta r^2}{2!}\boldsymbol{\alpha}_i''(r)+\frac{\Delta r^3}{3!}\boldsymbol{\alpha}_i'''(r)+\mathcal{O}\left(\Delta r^4\right),\\\boldsymbol{\alpha}_i(r-\Delta r)&=\boldsymbol{\alpha}_i(r)-\Delta r\boldsymbol{\alpha}_i'(r)+\frac{\Delta r^2}{2!}\boldsymbol{\alpha}_i''(r)-\frac{\Delta r^3}{3!}\boldsymbol{\alpha}_i'''(r)+\mathcal{O}\left(\Delta r^4\right).\end{aligned}$$

Combining the two equations above, we have

$$\boldsymbol{\alpha}_i'(r) = \frac{\boldsymbol{\alpha}_i(r+\Delta r) - \boldsymbol{\alpha}_i(r-\Delta r)}{2\Delta r} + \mathcal{O}\left(\Delta r^2\right).$$

Therefore, we can write

$$\frac{\partial \boldsymbol{\alpha}_i}{\partial r} \approx \frac{\boldsymbol{\alpha}_i(r+\Delta r) - \boldsymbol{\alpha}_i(r-\Delta r)}{2\Delta r} = \frac{\sum_{j=m_1+1}^{m_2} \frac{1}{\lambda_j} \mathbf{v}_j \mathbf{v}_j^T \tilde{\mathbf{y}}_i}{2\Delta r},$$

where m_1 and m_2 are selected such that $\sum_{i=1}^{m_1}\lambda_i / \sum_{i=1}^{t}\lambda_i \leq r - \Delta r < \sum_{i=1}^{m_1+1}\lambda_i \sum_{i=1}^{t}\lambda_i$ and $\sum_{i=1}^{m_2}\lambda_i / \sum_{i=1}^{t}\lambda_i \leq r + \Delta r < \sum_{i=1}^{m_2+1}\lambda_i / \sum_{i=1}^{t}\lambda_i$.

V. Experimental Results

In this section, we will use the Pareto-optimal criterion derived in this paper to select the appropriate kernel parameters of KRR and KPCR. Comparisons with the state of the art as well as the alternative criteria (i.e., sum and product) defined in the preceding section are provided.

A. Standard data-sets

We select fifteen data-sets from the UCI machine learning databases [5] and the DELVE collections [1]. Specifically, these databases include the following sets (in parenthesis we show the number of samples/number of dimensions): Boston housing (506/14), auto mpg (398/8), slump(103/8), price(159/16), diabetes(43/3), wdbc(194/33), servo(167/5), puma-8nm (8192/9), puma-8nh (8192/9), puma-8fm (8192/9), puma-8fh (8192/9), kin-8nm (8192/9), kin-8nh (8192/9), kin-8fm (8192/9) and kin-8fh (8192/9). The Boston housing data-set was collected by the U.S. Census Service and describes the housing information in Boston, MA. The task is to predict the median value of a home. The auto mpg set details fuel consumption predicted in terms of 3 discrete and 4 continuous attributes. In the slump set, the concrete slump is predicted by 7 different ingredients. The price data-set requires predicting the price of a car based on 15 attributes. In the diabetes set, the goal is to predict the level of the serum C-peptide. In the Wisconsin Diagnostic Breast Cancer (wdbc) set, the time of the recurrence of breast cancer is predicted based on 32 measurements of the patients. The servo set concerns a robot control problem. The rise time of a servomechanism is predicted based on two gain settings and two choices of mechanical linkages. The task in the Pumadyn is to predict angular accreditation from a simulation of the dynamics of a robot arm. And, the Kin set requires us to predict the distance of the end-effector from a target in a simulation of the forward dynamics of an 8 link all-revolute robot arm. There are different scenarios in both Pumadyn and Kin data-sets according to the degree of non-linearity (fairly-linear or nonlinear) and the amount of noise (moderate or high).

When using the ε-constraint criterion, we employ the interior-point method of [19]. Recall that in our proposed modified ε-constraint criterion, we also need to select a small scalar s.

We tested for values of $s = \{10^{-1}, 10^{-2}, 10^{-3}\}$ obtaining the same results regardless of the value of this parameter. The results reported in the present paper are for $s = 10^{-3}$.

We compare our model selection approaches to the two typical criteria used in the literature, Cross-Validation (CV) and Generalized Cross-Validation (GCV) [39]. In particular, we employ a 10-fold CV. The kernel parameter in the RBF is searched in the range $[\mu - 2\delta, \mu + 2\delta]$, where μ and δ are the mean and standard deviation of the distances between all pairwise training samples. In the polynomial kernel, its degree is tested in the range of 1 to 6. The regularization parameter in KRR is selected among the set $\{10^{-5}, \ldots, 10^{4}\}$, and the percentage of variance r in KPCR is searched in the range [0.8, 1]. Moreover, we compare our modified ε-constraint approach with the original ε-constraint method.

To test all the different algorithms, for each data-set, we generate five random permutations and conduct 10-fold cross-validation on each one. This yields a total of 50 estimated errors. Herein, we report their means and the standard deviations (stdv). Unless otherwise noted, in the experiments below, we report the Root Mean Squared Error (RMSE) as our measure of the deviation between the true response $\mathbf{y}_i$ and the predicted response $\hat{\mathbf{y}}$, i.e., $RMSE = \left[n^{-1} \sum_{i=1}^{n} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2 \right]^{1/2}$. It is important to that the random permutations utilized at this testing stage is different from the one employed for parameter selection. Note also that in order to provide a fair comparison, we tested all the algorithms using the same sets. In addition, the Bonferroni-Dunns test and a two-sided paired Wilcoxon signed rank test [44, 17, 20] are used to check for statistical significance. Errors in bold mean that they were significantly smaller than the others with significance level of .05.

Table I shows the regression results of KRR using both the RBF and the polynomial kernels. We see that regardless of the kernel used, the proposed modified ε-constraint approaches consistently provide the smallest RMSE. We also note that the modified ε-constraint approach obtains smaller RMSE than the ε-constraint method.

Table II shows the regression results of KPCR using the RBF and polynomial kernels. Once more, the proposed approach generally outperforms the others. Additionally, as in KRR, the modified ε-constraint approach generally yields the best results.

A major advantage of the proposed approach over CV is that it uses all the training data for training. In contrast, CV needs to use part of the training data for verification purposes. This limits the amount of training data used to fit the function to the data.

B. Comparison with the state of the art

We now provide a comparison with the methods available in the literature and typically employed in the above databases. Specifically, we compare our results with Support Vector Regression (SVR) [37] with the RBF and polynomial kernels, Multiple Kernel Learning in SVR (MKL-SVR) [28], and Gaussian Processes for Regression (GPR) [45]. In SVR, the parameters are selected using CV. In MKL-SVR, we employ three kernel functions: the RBF, the polynomial and the Laplacian defined as $k(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\beta}\right)$. The RBF kernel parameter is set to be the mean of the distances between all pairwise training samples;

the degree of the polynomial kernel is set to 2; and β in the Laplacian kernel is set as $\beta=\frac{2}{n(n+1)}\sum_{i=1}^{n}\sum_{j=i}^{n}\|\mathbf{x}_i-\mathbf{x}_j\|$, where n is the number of training samples. MOSEK [3] is used to solve the quadratically constrained programming problems to get the combinational coefficients of the kernel matrices. In GPR, the hyperparameters of the mean and covariance functions are determined by minimizing the negative log marginal likelihood of the data.

We compare the results given by the above algorithms with those obtained by our approach applied to KRR and using the RBF kernel, because this method tends to yield more favorable results. The comparisons are shown in Table III. Note that our approach generally yields smaller RMSE.

Furthermore, for each of the data-sets described above, we provide a comparison between our results and the best results found in the literature. For the Boston housing data-set, [36] reports the best fits with Relevance Vector Machine (RVM); for the Auto mpg data-set, the best result is obtained by MKL-SVR [28]; for the Slump data, [8] proposes a k nearest neighbor based regression method and shows its superiority over others; for the price data-set, [41] reports the best result with pace regression; Diabetes data-set is used in [10] and the best results is obtained using Least Angle Regression; for the servo data-set, [13] shows that regression with random forests gets best results; and for the last eight data-sets, Gaussian processes for regression trained with a maximum-a-posteriori approach is generally considered to provide state of the art results [43]. The comparison across all the data-sets is given in Table IV. We see that our approaches provide better or comparable results to the top results described in the literature but with the main advantage that a single algorithm is employed in all data-sets.

C. Alternative Optimizations

In Section III-D, we presented two alternatives for combining different objective functions – the sum and the product criteria. Here we provide a comparison of these criteria and the approach derived in this paper. In particular, we combine model fit E_f and model complexity E_c via the summation and product in KRR and KPCR. The regularization term in (29) and in (31) is selected by 5-fold CV. Table V shows the corresponding regression results. In this table, $\mathscr{A}_R$ and $\mathscr{A}_P$ denote the method $\mathscr{A}$ with a RBF and a polynomial kernel, respectively. We see that these two alternative criteria generally perform worse than the Pareto-optimal based approach.

D. Comparison with the L_2 norm

We provide a comparison between our complexity measure E_c and the commonly used L_2 norm. This was done by plugging in the L_2 norm as the constraint in the modified ε-constraint algorithm. The RMSE results are shown in Table VI. We see that the proposed complexity measure generally outperforms the L_2 norm in penalizing the regression function.

E. Age estimation

In the last two sections we want to test the derived approach on two classical applications – age estimation from faces and weather prediction.

The process of aging can cause significant changes in human facial appearances. We used the FG-NET aging database described in [2] to model these changes. This data-set contains 1,002 face images of 82 subjects at different ages. The age ranges from 0 to 69. Face images include changes in illumination, pose, expression and occlusion (e.g., glasses and beards). We warp all images to a standard size and constant position for mouth and eyes as in [23]. All the pictures are warped to a common size of 60 × 60 pixels and converted to 8-bit graylevel images. Warped images of one individual are shown in Figure 4. We represent each image as a vector concatenating all the pixels of the image, i.e., the appearance-based feature representation.

We generate five random divisions of the data, each with 800 images for training and 202 for testing. Results in Table VII are presented using the Mean Absolute Errors (MAE). Here, we report the MAE rather than the RMSE because it is the common measure reported in the literature in age estimation [14] and thus facilitates comparison with the state of the art. We can see that the modified ε-constraint method outperforms the other algorithms. In [48], the authors represent the images using a set of highly redundant Haar-like features and select relevant features using a boosting method. We implemented this method using the same five divisions of the data. Our approach is slightly better using a simpler appearance-based representation.

F. Weather prediction

The weather data of the University of Cambridge [42] is used in this experiment. The maximum temperature of a day is predicted based on several parameters measured every hour during the day. These parameters include pressure, humanity, dew point (i.e., the temperature at which a parcel of humid air must be cooled for it to condense), wind knots, sunshine hours and rainfall. We use the data in a period of five years (2005-2009) for training and the data between January and July of the year 2010 for testing. This corresponds to 1,701 training samples and 210 testing samples. The results are in Table VIII. In [29], the authors employed support vector regression and report state of the art results. Our experiment shows that our approach performs better than their algorithm. The predictions obtained from the modified ε-constraint approach are also plotted in Figure 5. We observe that our approach can provide the prediction of the daily maximum temperature with high accuracy.

VI. Conclusions

Non-linear regression is a fundamental problem in machine learning and pattern recognition with multiple applications in science and engineering. Many approaches have been proposed for linear regressions, but their non-linear extensions are known to present several limitations. A major limitation is the lack of regularization of the regressor. Without proper regularization, the complexity of the estimated function (e.g., the degree of the polynomial

describing the function) increases very rapidly, yielding poor generalizations on the unseen testing set [27]. To resolve this problem, we have derived a roughness penalty that measures the degree of change (of the regressed function) in the kernel space. This measure can then be used to obtain estimates that (in general) generalize better to the unseen testing set. However, to achieve this, the newly derived objective function needs to be combined with the classical one measuring its fitness (i.e., how well the function estimates the sample vectors). Classical solutions would be to use the sum or product of the two objective functions [47]. However, we have shown that these solutions do not generally yield desirable results in kernel methods in regression. To resolve this issue, we have proposed a multiple optimization approach based on the idea of Pareto-optimality. In this MOP framework, we have derived a novel method: the modified ε-constraint approach. While the original ε-constraint method cannot guarantee Pareto-optimal solutions, we have proven that the derived modified version does. Extensive evaluations with a large variety of databases has shown that this proposed modified ε-constraint approach yields better generalizations than previously proposed algorithms. Although the proposed method was applied to classical non-linear regression methods such as KRR and KPCR, our methodology is general enough that can be applicable to other regularized regression approaches methods such as [26].

The other major contribution of the paper has been to show how we can use the derived approach for optimizing the kernel parameters. In any kernel method, one always has to optimize the parameters of the kernel mapping function. The classical approach for this task is CV. This technique suffers from two main problems. First, it is computationally expensive. Second, and arguably most important, it cannot use the entire sample set for training, because part of it is employed as a validation set. But, we know that (in general) the larger the training set, the better. Our proposed MOP framework is ideal for optimizing the kernel parameters, because it yields nice objective functions that can be minimized with standard gradient descent techniques.

We have provided extensive comparisons of the proposed approach against CV and GCV and the other state of the art techniques in kernel methods in regression. We have also compared our results to those obtained with the sum and product criteria. And, we have compared our results to the best fits found in the literature for each of the databases. In all cases, these comparisons demonstrate that the proposed approach yields fits that generalize better to the unseen testing sets.

Acknowledgement

This research was partially supported by the US National Institutes of Health under grant R21 DC 011081 and R01 EY 020834.

Appendix A

Partial Derivatives of the quadratic loss function

Let $L(\mathbf{z}) = \frac{1}{2}\sum_{i=1}^{q}(y_i - f_i(\mathbf{x}))^2$ be the quadratic loss then (10) is given by

$$\begin{aligned}\sum_{j=1}^{m}\frac{\partial^2 L(\mathbf{z}_i)}{\partial z_j^2} &= \frac{1}{2}\sum_{j=1}^{m}\sum_{k=1}^{q}\frac{\partial^2(y_{ik}-f_k(\mathbf{x}_i))^2}{\partial z_j^2}\\ &= \frac{1}{2}\sum_{k=1}^{q}\left(\sum_{j=1}^{p}\frac{\partial^2(y_{ik}-f_k(\mathbf{x}_i))^2}{\partial x_{ij}^2}+\sum_{j=1}^{q}\frac{\partial^2(y_{ik}-f_k(\mathbf{x}_i))^2}{\partial y_{ij}^2}\right)\\ &= \sum_{k=1}^{q}\left(\sum_{j=1}^{p}\left(\frac{\partial f_k(\mathbf{x}_i)}{\partial x_{ij}}\right)^2+\sum_{j=1}^{p}\left(f_k(\mathbf{x}_i)-y_{ik}\right)\frac{\partial^2 f_k(\mathbf{x}_i)}{\partial x_{ij}^2}+1\right)\\ &= \sum_{k=1}^{q}\sum_{j=1}^{p}\left(\left(\frac{\partial f_k(\mathbf{x}_i)}{\partial x_{ij}}\right)^2+\left(f_k(\mathbf{x}_i)-y_{ik}\right)\frac{\partial^2 f_k(\mathbf{x}_i)}{\partial x_{ij}^2}\right)+q.\end{aligned}$$

Appendix B

Partial derivatives of the roughness function

Sections B-A and B-B show the derivations to obtain the second partial derivatives for the RBF and polynomial kernels. Note that this is a key component for the roughness function Ec. To do so, recall that the regression function is defined as $f_l(\mathbf{x}) = \sum_{m=1}^{n}\alpha_l^m \kappa\langle \mathbf{x}_m, \mathbf{x}\rangle$

A. RBF Kernel

The RBF kernel is given by $\kappa\langle \mathbf{x}_i, \mathbf{x}_j\rangle = exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}\right)$, where σ is the kernel parameter. Then the second partial derivative of f_l with respect to x_{ij} is

$$\begin{aligned}\frac{\partial^2 f_l(\mathbf{x}_i)}{\partial x_{ij}^2} &= \frac{1}{\sigma^2}\frac{\partial\left[\sum_{m=1}^{n}\alpha_l^m\, exp\left(-\frac{\|\mathbf{x}_m-\mathbf{x}_i\|^2}{2\sigma^2}\right)(x_{mj}-x_{ij})\right]}{\partial x_{ij}}\\ &= \frac{1}{\sigma^2}\left[\sum_{m=1}^{n}\alpha_l^m\, exp\left(-\frac{\|\mathbf{x}_m-\mathbf{x}_i\|^2}{2\sigma^2}\right)\frac{(x_{mj}-x_{ij})^2}{\sigma^2}-\sum_{m=1,m\neq i}^{n}\alpha_l^m\, exp\left(-\frac{\|\mathbf{x}_m-\mathbf{x}_i\|^2}{2\sigma^2}\right)\right]\\ &= \frac{1}{\sigma^2}\left[\alpha_l^i+\sum_{m=1}^{n}\alpha_l^m\, exp\left(-\frac{\|\mathbf{x}_m-\mathbf{x}_i\|^2}{2\sigma^2}\right)\left(\frac{(x_{mj}-x_{ij})^2}{\sigma^2}-1\right)\right]\\ &= \boldsymbol{\alpha}_l^T\mathbf{p}_{ij}.\end{aligned}$$

B. Polynomial kernel

The polynomial kernel is given by $\kappa\langle \mathbf{x}_i, \mathbf{x}_j\rangle = \left(\mathbf{x}_i^T\mathbf{x}_j+1\right)^d$, where d is the kernel parameter. The second partial derivative of $f_l(.)$ with respect to x_{ij} is

$$\begin{aligned}\frac{\partial^2 f_l(\mathbf{x}_i)}{\partial x_{ij}^2} &= \sum_{m=1,m\neq i}^{n}\alpha_l^m\, dx_{mj}\frac{\partial\left(\mathbf{x}_m^T\mathbf{x}_i+1\right)^{d-1}}{\partial x_{ij}}+2\alpha_l^i d\frac{\partial\left[\left(\mathbf{x}_i^T\mathbf{x}_i+1\right)^{d-1}x_{ij}\right]}{\partial x_{ij}}\\ &= d\left[\alpha_l^i\left(\mathbf{x}_i^T\mathbf{x}_i+1\right)^{d-2}\left(3(d-1)x_{ij}^2+2\left(\mathbf{x}_i^T\mathbf{x}_i+1\right)\right)+(d-1)\sum_{m=1}^{n}\alpha_l^m\left(\mathbf{x}_m^T\mathbf{x}_i+1\right)^{d-2}x_{mj}^2\right]\\ &= \boldsymbol{\alpha}_l^T\mathbf{g}_{ij}.\end{aligned}$$

Appendix C

Running Time

Table C.1 presents the running time for four different model selection algorithms: Modified ε-constraint, ε-constraint, CV and GCV. These values correspond to the execution time for KRR with an RBF Kernel using the cross-validation testing procedure described in Section V.

TABLE C.1

Running time for KRR with a Gaussian Kernel in Seconds

Data set	Modified ε-constraint	ε-constraint	CV	GCV
HOUSING	1562.1	10502.3	1282.3	757.7
MPG	878.1	3083.9	1233.7	904.2
SLUMP	412.7	935.6	37.8	29.9
PRICE	573.7	4064.2	71.8	20.2
DIABETES	24.3	203.3	4.7	10.4
WBDC	1735.3	20689.2	88.7	248.3

REFERENCES

1. Data for evaluating learning in valid experiments (DELVE). http://www.cs.toronto.edn/delve/
2. FG-NET aging database. http://www.fgnet.rsunit.com/
3. Andersen, ED.; Andersen, KD. The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm.. In: Frenk, Hans; Roos, Kees; Terlaky, Tams; Zhang, Shuzhong, editors. High Performance Optimization, volume 33 of Applied Optimization. Springer; US: 2000. p. 197-232.
4. Bishop, CM. Neural Networks for Pattern Recognition. Oxford University Press; 1995.
5. Blake, CL.; Merz, CJ. UCI repository of machine learning databases. University of California; Irvine: 1998. http://www.ics.uci.edu/mlearn/MLRepository.html
6. Chapelle O, Rakotomamonjy A. Second order optimization of kernel parameters. NIPS Workshop on Kernel Learning. 2008
7. Chen B, Zhao S, Zhu P, Principe JC. Quantized kernel least mean square algorithm. IEEE Transactions on Neural Networks and Learning Systems. 2012; 23(1):22–32. [PubMed: 24808453]
8. Desai A, singh H, Pudi V. Gear: Generic, efficient, accurate knn-based regression. Intl Conf on Knowledge Discovery and Information Retrieval. 2010
9. Edmonson W, Principe J, Srinivasan K, Wang Chuan. A global least mean square algorithm for adaptive iir filtering. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. 1998; 45(3):379–384.
10. Efron B, Hastie T, Johnstone I, Tibshirani R. Least angle regression. Annals of Statistics. 2004; 32(2):407–499.
11. Evgeniou, T.; Pontil, M. On the v-gamma-dimension for regression in reproducing kernel hilbert spaces. A.i. memo. MIT Artificial Intelligence Lab; 1999.
12. Evgeniou T, Pontil M, Poggio T. Regularization networks and support vector machines. Advances in Computational Mathematics. 2000; 13(1):1–50.
13. Fan G, Gray J. Regression tree analysis using target. Journal of Computational and Graphical Statistics. 2005; 14(1):1–13.

14. Fu Y, Guo G, Huang TS. Age synthesis and estimation via faces: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010; 32(11):1955–1976. [PubMed: 20847387]
15. Haimes YY, Lasdon LS, Wismer DA. On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Transactions on Systems, Man, and Cybernetics, SMC-1. 1971:296–297.
16. Hastie, T.; Tibshirani, R.; Friedman, J. The elements of statistical learning: Data mining, inference and prediction. Springer-Verlag; New York: 2001.
17. Hochberg, Y.; Tamhane, AC. Multiple comparison procedures. Wiley; 1987. Wiley series in probability and mathematical statistics: Applied probability and statistics..
18. Holmstrom L, Koistinen P. Using additive noise in back-propagation training. IEEE Transactions on Neural Networks. 1992; 3(1):24–38. [PubMed: 18276403]
19. Karmarkar N. A new polynomial time algorithm for linear programming. Combinatorica. 1984; 4(4):373–395.
20. Kvam, PH.; Vidakovic, B. Nonparametric Statistics with Applications to Science and Engineering. Wiley; 2011.
21. Lanckriet GRG, Cristianini N, Bartlett P, Ghaoui LE, Jordan MI. Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research. 2004; 5:27–72.
22. Magnus, JR.; Neudecker, H. Matrix Differential Calculus with Applications in Statistics and Econometrics. 2nd Edition.. John Wiley and Sons; 1999.
23. Martinez AM. Recognizing imprecisely localized, partially occluded and expression variant faces from a single sample per class. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002; 24(6):748–763.
24. Martinez AM, Zhu M. Where are linear feature extraction methods applicable? IEEE Transactions on Pattern Analysis and Machine Intelligence. 2005; 27(12):1934–1944. [PubMed: 16358412]
25. Miettinen, K. Nonlinear Multiobjective Optimization, volume 12 of International Series in Operations Research and Management Science. Kluwer Academic Publishers; Dordrecht: 1999.
26. Poggio T, Girosi F. Networks for approximation and learning. Proceedings of the IEEE. 1990; 78(9):1481–1497.
27. Poggio T, Rifkin R, Mukherjee S, Niyogi P. General conditions for predictivity in learning theory. Nature. 2004; 428:419–422. [PubMed: 15042089]
28. Qiu S, Lane T. A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2009; 6(2):190–199. [PubMed: 19407344]
29. Radhika Y, Shashi M. Atmospheric temperature prediction using support vector machines. International Journal of Computer Theory and Engineering. 2009; 1(1):55–58.
30. Rasmussen CE, Ghahramani Z. Occam's razor. Advances in Neural Information Processing Systems. 2001; 13
31. Rosipal R, Girolami M, Trejo LJ, Cichoki A. Kernel pca for feature extraction and de-noising in nonlinear regression. Neural Computing and Applications. 2001; 10:231243.
32. Schölkopf, B.; Smola, AJ. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press; 2001.
33. Shawe-Taylor, J.; Cristianini, N. Kernel Methods for Pattern Analysis. Cambridge University Press; 2004.
34. Sonnenburg S, Ratsch G, Schafer C, Scholkopf B. Large scale multiple kernel learning. Journal of Machine Learning Research. 2006; 7:1531–1565.
35. Styblinski MA, Tang TS. Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing. Neural Netw. 1990; 3(4):467–483.
36. Tipping ME. Sparse bayesian learning and the relevance vector machine. Journal of Machine Learning Research. 2001; 1(1):211–244.
37. Vapnik, V. Statistical Learning Theory. John Wiley and Sons; 1998.
38. Vapnik V, Golowich S, Smola A. Support vector method for function approximation, regression estimation, and signal processing. Advances in Neural Information Processing Systems. 1996; 9

39. Wahba, G. Spline Models for Observational Data. Society for Industrial and Applied Mathematics; 1990.
40. Wang S, Zhu W, Liang Z. Shape deformation: SVM regression and application to medical image segmentation. Proceedings of International Conference on Computer Vision. 2001
41. Wang, Y. PhD thesis. University of Waikato; New Zeland: 2000. A New Approach to Fitting Linear Models in High Dimensional Spaces..
42. Cambridge Weather Database. University of Cambridge; http://www.cl.cam.ac.uk/research/dtg/weather/
43. Weinberger KQ, Tesauro G. Metric learning for kernel regression. Eleventh International Conference on Artificial Intelligence and Statistics, Omnipress. 2007
44. Wilcoxon F. Individual comparisons by ranking methods. Biometrics Bulletin. 1945; 1(6):8083.
45. Williams CKI, Rasmussen CE. Gaussian processes for regression. Advances in Neural Information Processing Systems. 1996; 8
46. Yang H, Xu Z, Ye J, King I, Lyu MR. Efficient sparse generalized multiple kernel learning. Neural Networks, IEEE Transactions on. 2011; 22(3):433–446.
47. You D, Hamsici OC, Martinez AM. Kernel optimization in discriminant analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2011; 33(3):631–638. [PubMed: 20820072]
48. Zhou S, Georgescu B, Zhou X, Comaniciu D. Image based regression using boosting method. Proceedings of the Tenth IEEE International Conference on Computer Vision. 2005

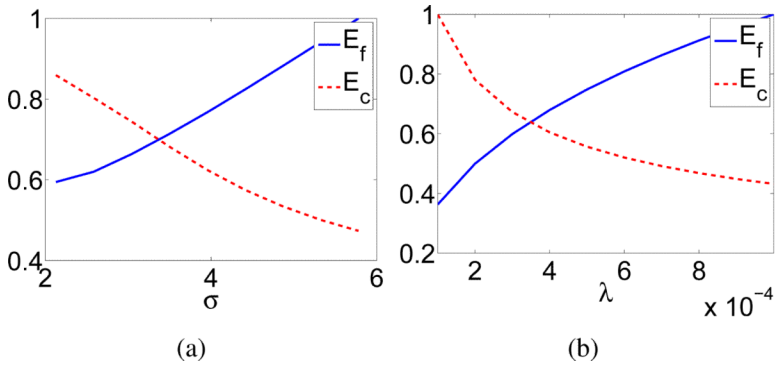

Fig. 1.

The two plots in this figure show the contradiction between the RSS and the curvature measure with respect to: (a) the kernel parameter σ, and (b) the regularization parameter λ in Kernel Ridge Regression. The Boston Housing data-set [5] is used in this example. Note that in both cases, while one criterion increases, the other decreases. Thus, a compromise between the two criteria ought to be determined.

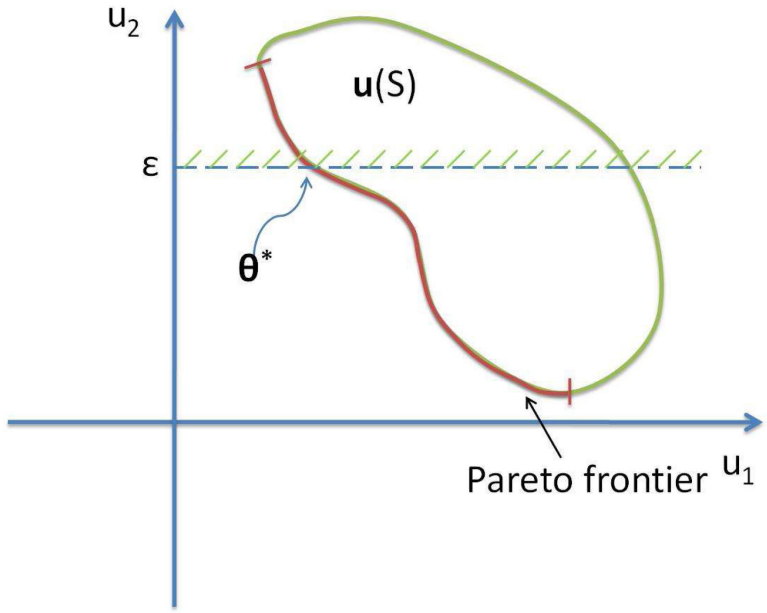

Fig. 2.

Here we show a case of two objective functions. **u**(*S*) represents the set of all the objective vectors with the Pareto frontier colored in red. The Pareto-optimal solution θ can be determined by minimizing $\mathbf{u}_1$ given that $\mathbf{u}_2$ is upper-bounded by ε.

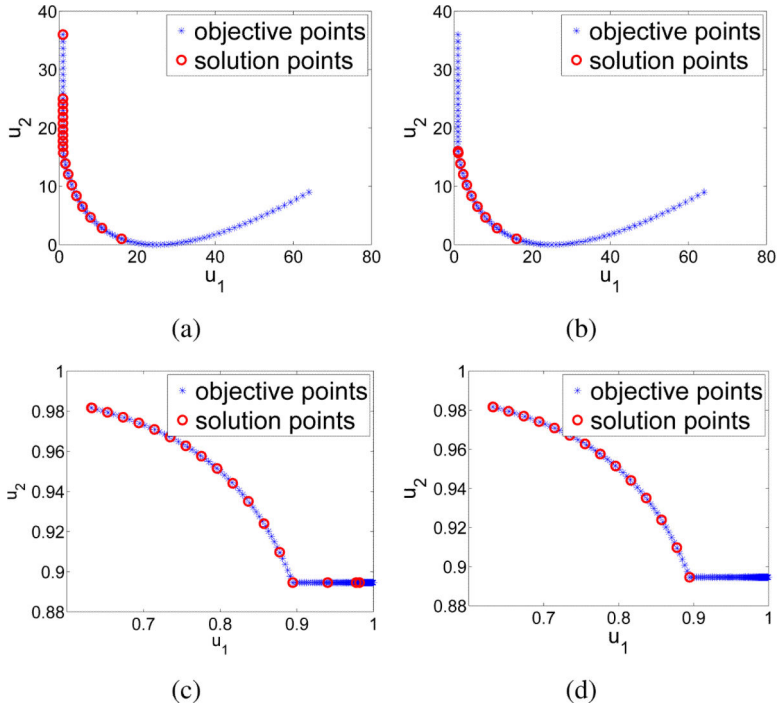

Fig. 3.

Comparison between the proposed modified and the original ε-constraint methods. We have used '*' to indicate the objective vector and 'o' to specify the solution vector. Solutions given by (a) the ε-constraint method and (b) the proposed modified ε-constraint approach on the first example, and (c) the ε-constraint method and (d) the modified ε-constraint approach on the second example. Note that the proposed approach identifies the Pareto-frontier, while the original algorithm identifies weakly Pareto-solutions, since the solution vectors go beyond the Pareto-frontier.

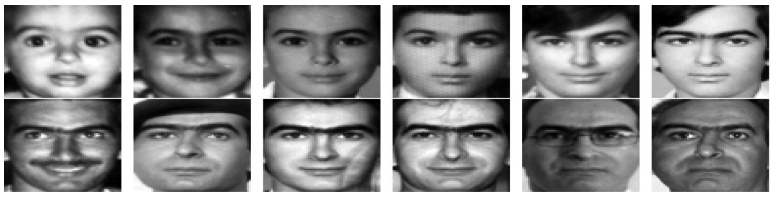

Fig. 4.
Sample images showing the same person at different ages.

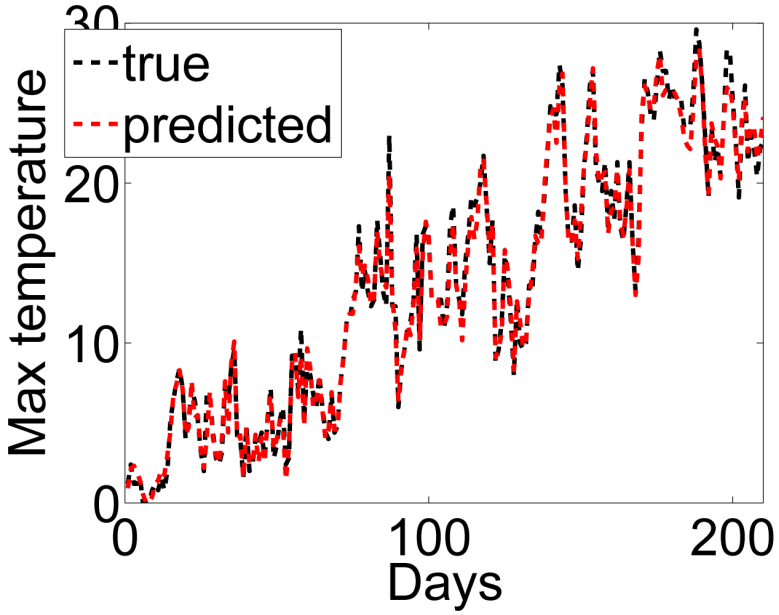

Fig. 5.

This figure plots the estimated (lighter dashed curve) and actual (darker dashed curve) maximum daily temperature for a period of more than 200 days. The estimated results are given by the algorithm proposed in this paper.

TABLE I

Results for KRR. Mean RMSE and standard deviation (in parentheses).

KERNEL	RBF				POLYNOMIAL			
DATA SET/METHOD	MODIFIED ε-CONSTRAINT	ε-CONSTRAINT	CV	GCV	MODIFIED ε-CONSTRAINT	ε-CONSTRAINT	CV	GCV
HOUSING	**2.89** * (0.77)	**3.01**(0.78)	3.25(0.84)	4.01(1.01)	**3.71**(0.87)	4.38(0.99)	4.24(1.03)	8.67(6.78)
MPG	**2.51** * (0.52)	**2.59** * (0.57)	**2.72** * (0.40)	2.61(0.52)	**2.82**(0.45)	3.25(0.58)	3.24(0.57)	3.21(0.80)
SLUMP	**6.62** * (1.49)	7.36(2.29)	**6.70**(1.53)	22.1(8.95)	**7.09**(1.22)	8.85(2.05)	9.86(1.53)	**7.20**(1.77)
PRICE	**2.21** * (0.90)	2.73(1.54)	2.42(0.90)	8.88(5.43)	**3.08**(1.20)	3.29(1.50)	4.01(1.48)	3.41(1.5)
DIABETES	**0.55** * (0.23)	0.72(0.33)	**0.57**(0.19)	0.88(0.31)	**0.52** * (0.17)	0.60(0.20)	2.31(0.87)	0.62(0.33)
WDBC	**31.46** * (1.59)	**32.15** * (4.86)	**31.50** * (4.37)	50.30(9.13)	34.11(4.23)	35.12(5.21)	46.61(6.89)	**32.04** * (4.35)
SERVO	**0.51** * (0.29)	0.56(0.30)	0.59(0.32)	0.81(0.52)	**0.70**(0.25)	**0.70**(0.25)	0.75(0.25)	0.65(0.27)
PUMA-8NM	**1.44** * (0.02)	1.51(0.03)	2.42(0.05)	**1.44** * (0.03)	**1.42** * (0.02)	1.89(0.04)	1.89(0.04)	1.46(0.02)
PUMA-8NH	3.65(0.03)	3.64(0.03)	3.98(0.06)	**3.56** * (0.04)	5.08(1.26)	5.28(0.19)	4.11(0.14)	**3.61** * (0.06)
PUMA-8FM	**1.13** * (0.01)	1.19(0.02)	1.19(0.01)	**1.14** * (0.02)	**1.27**(0.01)	1.37(0.09)	1.29(0.005)	1.27(0.01)
PUMA-8FH	**3.23** * (0.01)	3.45(0.02)	**3.23** * (0.01)	**3.23** * (0.01)	3.78(0.16)	4.86(0.14)	3.23(0.01)	**3.24** * (0.02)
KIN-8NM	**0.11** * (0.002)	0.15(0.003)	0.16(0.002)	0.19(0.02)	**0.18**(0.0008)	0.24(0.03)	0.22(0.002)	0.19(0.01)
KIN-8NH	**0.18** * (0.001)	**0.18**(0.002)	0.19(0.002)	**0.18**(0.002)	**0.20**(0.002)	0.29(0.006)	0.24(0.003)	0.22(0.003)
KIN-8FM	0.016(0.002)	0.016(0.03)	**0.013** * (0.0001)	0.339(0.202)	**0.013** * (0.0001)	0.02(0.0001)	0.16(0.003)	0.015(0.0001)
KIN-8FH	0.07(0.002)	0.061(0.002)	**0.043** * (0.0002)	**0.043** * (0.0002)	**0.046**(0.0002)	0.046(0.0002)	0.16(0.003)	0.050(0.0003)

In each kernel, the best result is in bold.

* The symbol is used to indicate the top result over all methods and kernels.

TABLE II

Results for KPCR. Mean RMSE and standard deviation (in parentheses).

KERNEL	RBF				POLYNOMIAL			
DATA SET/METHOD	MODIFIED ε-CONSTRAINT	ε-CONSTRAINT	CV	GCV	MODIFIED ε-CONSTRAINT	ε-CONSTRAINT	CV	GCV
HOUSING	**4.04***(0.88)	4.56(0.67)	9.14(1.10)	11.99(6.89)	8.45(1.72)	9.12(2.30)	**6.05**(0.95)	9.37(1.77)
MPG	**3.00***(0.58)	4.64(0.82)	7.71(0.90)	3.64(1.63)	7.30(0.81)	7.82(1.54)	**5.92**(1.00)	8.16(1.78)
SLUMP	**6.39***(1.53)	7.55(1.68)	9.28(1.94)	7.64(1.42)	**7.68**(1.88)	8.15(2.11)	8.48(2.80)	9.49(3.00)
PRICE	**3.90***(2.16)	4.67(2.15)	12.62(2.02)	9.78(2.98)	**6.06**(1.93)	6.27(2.29)	**5.79**(1.49)	6.61(1.61)
DIABETES	**0.76***(0.33)	0.96(0.43)	0.99(0.53)	**0.74***(0.34)	1.01(1.47)	**0.73***(0.80)	1.85(1.92)	1.23(1.31)
WDBC	**30.66***(4.71)	35.32(5.87)	33.5(4.53)	43.53(7.05)	**34.47**(10.27)	56.68(7.71)	47.21(13.44)	41.13(14.89)
SERVO	**0.71***(0.30)	1.35(0.33)	1.41(0.34)	1.29(0.40)	1.13(0.25)	1.11(0.25)	**0.74**(0.24)	0.81(0.24)
PUMA-8NM	3.69(0.02)	3.66(0.02)	2.42(0.05)	**1.75***(0.07)	3.71(0.32)	4.12(0.25)	4.13(0.53)	4.15(0.70)
PUMA-8NH	**4.39**(0.04)	**4.39**(0.02)	4.56(0.13)	**3.65***(0.08)	**4.58**(0.29)	4.84(0.22)	**4.56**(0.16)	5.59(0.58)
PUMA-8FM	**1.28***(0.05)	1.73(0.77)	4.04(1.13)	**1.26***(0.01)	**1.29***(0.005)	1.46(0.36)	1.56(0.61)	1.81(0.44)
PUMA-8FH	**3.22***(0.01)	**3.33***(0.28)	3.49(0.08)	**3.26***(0.07)	**3.75**(0.24)	3.92(0.41)	3.99(0.39)	5.04(0.74)
KIN-8NM	**0.19***(0.01)	**0.19***(0.01)	0.22(0.02)	0.22(0.01)	**0.22**(0.04)	**0.21**(0.03)	0.26(0.05)	0.30(0.07)
KIN-8NH	**0.21***(0.007)	**0.21***(0.01)	0.23(0.01)	0.24(0.01)	**0.25***(0.05)	0.30(0.09)	0.27(0.05)	0.33(0.07)
KIN-8FM	0.05(0.01)	0.06(0.04)	**0.03**(0.007)	0.04(0.01)	**0.02***(0.0001)	0.05(0.08)	0.08(0.11)	0.10(0.13)
KIN-8FH	**0.06***(0.01)	0.07(0.02)	0.05(0.006)	**0.06***(0.01)	**0.07***(0.07)	**0.07***(0.07)	0.12(0.12)	0.12(0.12)

TABLE III

Mean and standard deviation of RMSE of different methods.

DATA SET/METHOD	MODIFIED ε-CONSTRAINT	SVR_{rbf}	SVR_{pol}	MKL-SVR	GPR
HOUSING	**2.89**(0.77)	3.45(1.04)	5.66(1.88)	3.34(0.70)	**3.05**(0.82)
MPG	**2.51**(0.52)	2.69(0.60)	4.03(0.96)	2.67(0.61)	**2.64**(0.50)
SLUMP	**6.62**(1.49)	**6.77**(1.90)	8.37(2.86)	6.90(1.41)	6.88(1.51)
PRICE	**2.21**(0.90)	2.40(0.84)	3.72(1.55)	2.51(0.91)	11.2(2.26)
DIABETES	**0.55**(0.23)	0.68(0.31)	0.78(0.39)	0.65(0.35)	**0.59**(0.20)
WDBC	**31.46**(1.59)	32.08(4.76)	44.1(9.87)	32.20(4.65)	**31.60**(4.3)
SERVO	**0.51**(0.29)	0.61(0.35)	1.37(0.41)	0.60(0.36)	0.57(0.30)
PUMA-8NM	**1.44**(0.02)	**1.44**(0.03)	3.35(0.11)	1.51(0.02)	1.47(0.03)
PUMA-8NH	**3.65**(0.03)	**3.67**(0.06)	4.55(0.07)	3.78(0.05)	**3.65**(0.03)
PUMA-8FM	**1.13**(0.01)	1.17(0.02)	2.04(0.05)	1.21(0.03)	1.17(0.02)
PUMA-8FH	**3.23**(0.01)	**3.24**(0.02)	3.84(0.06)	3.35(0.05)	**3.23**(0.01)
KIN-8NM	**0.11**(0.002)	0.12(0.002)	0.21(0.003)	0.16(0.03)	0.12(0.002)
KIN-8NH	**0.18**(0.001)	0.19(0.003)	0.23(0.01)	0.20(0.002)	**0.18**(0.002)
KIN-8FM	**0.016**(0.002)	0.043(0.002)	0.048(0.001)	0.045(0.002)	**0.013**(0.00009)
KIN-8FH	0.07(0.002)	0.047(0.0009)	0.06(0.006)	0.05(0.001)	**0.043**(0.0007)

TABLE IV

Comparison of our results with the state of the art.

	Housing	**Mpg**	**Slump**	**Price**	**Diabetes**	**servo**	**Puma-8nm**
BEST	3.46(0.93)	2.67(0.61)	**6.79**(1.89)	2.62(0.87)	0.68(0.25)	0.59(0.30)	1.47(0.03)
OURS	**2.89**(0.77)	**2.51**(0.50)	**6.62**(1.49)	**2.21**(0.90)	**0.55**(0.23)	**0.51**(0.29)	**1.44**(0.02)

	Puma-8nh	**Puma-8fm**	**Puma-8fh**	**Kin-8nm**	**Kin-8nh**	**Kin-8fm**	**Kin-8fh**
BEST	**3.65**(0.03)	1.17(0.02)	**3.23**(0.01)	0.12(0.002)	**0.18**(0.002)	**0.013**(0.00009)	**0.043**(0.0007)
OURS	**3.65**(0.03)	**1.13**(0.01)	**3.23**(0.01)	**0.11**(0.002)	**0.18**(0.002)	0.016(0.002)	0.07(0.002)

TABLE V

Regression performance with alternative optimization criteria.

Method	KRR_R			KRR_P			PCR_R			PCR_F		
Data set	**Ours**	**Sum**	**Product**	**Ours**	**Sum**	**Product**	**Ours**	**Sum**	**Product**	**Ours**	**Sum**	**Product**
HOUSING	**2.89**(0.77)	**3.06**(0.78)	3.30(0.85)	**3.71**(0.87)	4.75(0.89)	4.66(1.75)	**4.04**(0.88)	9.46(5.73)	6.48(3.85)	8.45(1.72)	**5.56**(6.77)	**4.97**(3.98)
MPG	**2.51**(0.52)	2.63(0.50)	2.64(0.49)	**2.82**(0.45)	4.34(2.51)	18.04(2.59)	**3.00**(0.58)	4.56(0.83)	4.25(0.69)	7.30(0.81)	**5.48**(6.04)	**4.29**(3.21)
SLUMP	**6.62**(1.49)	6.87(1.51)	6.85(1.51)	**7.09**(1.22)	8.03(1.78)	13.17(2.84)	**6.39**(1.53)	7.65(1.86)	7.64(1.87)	**7.68**(1.88)	14.70(9.04)	16.79(9.10)
PRICE	**2.21**(0.90)	**2.73**(1.45)	2.76(1.46)	**3.08**(1.20)	**2.72**(1.11)	**3.10**(1.11)	**3.90**(2.16)	4.17(2.85)	4.17(2.86)	**6.06**(1.93)	8.76(1.99)	13.43(20.9)
DIABETES	**0.55**(0.23)	0.66(0.29)	0.75(0.33)	**0.52**(0.17)	0.60(0.21)	3.47(2.15)	**0.76**(0.33)	0.86(0.45)	0.86(0.43)	**1.01**(1.47)	**1.09**(1.56)	**0.65**(0.22)
WDBC	**31.46**(1.59)	47.99(7.01)	48.60(8.98)	**34.11**(4.23)	51.31(16.98)	64.29(73.02)	**30.66**(4.71)	34.01(4.86)	38.91(5.31)	**34.47**(10.27)	56.61(19.90)	56.58(32.64)
SERVO	**0.51**(0.29)	0.60(0.31)	0.57(0.30)	**0.70**(0.25)	0.94(0.36)	1.33(0.40)	**0.71**(0.30)	0.83(0.50)	0.86(0.49)	1.13(0.25)	**0.70**(0.27)	0.91(0.33)
PUMA-8NM	**1.44**(0.02)	1.50(0.03)	1.50(0.03)	**1.42**(0.02)	3.40(0.59)	3.89(0.04)	3.69(0.02)	**2.25**(0.38)	**2.37**(0.53)	**3.71**(0.32)	8.50(0.87)	6.96(3.00)
PUMA-8NH	**3.65**(0.03)	3.80(0.03)	3.86(0.04)	5.08(1.26)	**4.36**(0.29)	4.54(0.03)	4.39(0.04)	**3.90**(0.64)	**3.97**(0.65)	**4.58**(0.29)	13.82(3.81)	11.27(5.41)
PUMA-8FM	**1.13**(0.01)	1.18(0.01)	1.18(0.01)	**1.27**(0.01)	1.52(0.48)	2.79(0.12)	**1.28**(0.05)	2.58(0.97)	2.09(0.84)	**1.29**(0.005)	6.98(1.23)	3.88(2.81)
PUMA-8FH	**3.23**(0.01)	3.28(0.02)	3.28(0.02)	**3.78**(0.16)	3.81(0.38)	**3.79**(0.08)	**3.22**(0.01)	3.40(0.38)	3.30(0.12)	**3.75**(0.24)	**9.78**(5.64)	**7.82**(5.44)
KIN-8NM	**0.11**(0.002)	0.12(0.008)	0.13(0.01)	**0.18**(0.0008)	0.21(0.02)	0.68(0.35)	**0.19**(0.01)	**0.21**(0.02)	0.22(0.02)	**0.22**(0.04)	**0.27**(0.24)	**0.26**(0.25)
KIN-8NH	**0.18**(0.001)	0.19(0.002)	0.19(0.002)	0.20(0.002)	0.22(0.005)	0.42(0.27)	0.21(0.007)	0.22(0.002)	0.23(0.01)	0.25(0.005)	0.50(0.47)	0.29(0.01)
KIN-8FM	**0.016**(0.002)	0.020(0.0005)	0.020(0.0005)	**0.013**(0.0001)	0.020(0.0001)	0.57(0.22)	**0.05**(0.01)	**0.07**(0.03)	0.06(0.01)	**0.02**(0.0001)	0.11(0.29)	0.03(0.01)
KIN-8FH	0.07(0.002)	0.05(0.0007)	**0.046**(0.0005)	**0.046**(0.0002)	0.05(0.0001)	0.75(0.30)	**0.06**(0.01)	**0.08**(0.03)	**0.08**(0.02)	**0.07**(0.07)	**0.13**(0.25)	**0.06**(0.02)

TABLE VI

Comparison with L_2 norm.

Method	KRR_R		KRR_P		PCR_R		PCR_P	
Data set	**Ours**	**L_2 norm**	**Ours**	**L_2 norm**	**Ours**	**L_2 norm**	**Ours**	**L_2 norm**
HOUSING	**2.89**(0.77)	3.45(0.95)	**3.71**(0.87)	4.96(0.92)	**4.04**(0.88)	**4.36**(0.96)	8.45(1.72)	**7.40**(1.72)
MPG	**2.51**(0.52)	3.09(0.51)	**2.82**(0.45)	4.19(2.23)	**3.00**(0.58)	3.45(0.75)	**7.30**(0.81)	**7.42**(1.29)
SLUMP	**6.62**(1.49)	6.98(1.48)	**7.09**(1.22)	14.97(2.23)	**6.39**(1.53)	**6.43**(1.47)	**7.68**(1.88)	**8.12**(2.08)
PRICE	**2.21**(0.90)	**2.81**(1.21)	**3.08**(1.20)	**2.45**(3.77)	**2.35**(1.04)	**2.73**(1.31)	**6.06**(1.93)	**5.88**(1.73)
DIABETES	**0.55**(0.23)	0.68(0.25)	**0.52**(0.17)	0.78(0.20)	**0.76**(0.33)	0.87(0.43)	**1.01**(1.47)	**0.94**(1.40)
WDBC	**31.46**(1.59)	**32.10**(4.56)	**34.11**(4.23)	42.69(13.41)	**30.66**(4.71)	**30.69**(4.66)	**34.47**(10.27)	45.79(15.69)
SERVO	**0.51**(0.29)	0.90(0.31)	**0.70**(0.25)	0.96(0.34)	**0.71**(0.30)	**0.73**(0.31)	1.13(0.25)	**1.03**(0.25)
PUMA-8NM	**1.44**(0.02)	1.47(0.03)	**1.42**(0.02)	3.84(0.04)	3.69(0.02)	**3.37**(0.04)	**3.71**(0.32)	4.21(0.16)
PUMA-8NH	**3.65**(0.03)	3.75(0.03)	5.08(1.26)	**4.66**(0.06)	4.39(0.04)	**4.19**(0.14)	**4.58**(0.29)	4.61(0.31)
PUMA-8FM	**1.13**(0.01)	1.23(0.01)	**1.27**(0.01)	1.63(0.49)	**1.28**(0.05)	**1.26**(0.003)	**1.29**(0.005)	1.58(0.64)
PUMA-8FH	**3.23**(0.01)	**3.23**(0.01)	**3.78**(0.16)	4.06(0.03)	**3.22**(0.01)	3.30(0.12)	**3.75**(0.24)	3.97(0.52)
KIN-8NM	**0.11**(0.002)	0.17(0.001)	**0.18**(0.0008)	0.21(0.03)	0.19(0.01)	**0.16**(0.03)	**0.22**(0.04)	**0.22**(0.03)
KIN-8NH	**0.18**(0.001)	0.20(0.001)	**0.20**(0.002)	0.26(0.007)	**0.21**(0.007)	**0.21**(0.002)	**0.25**(0.005)	0.29(0.09)
KIN-8FM	**0.016**(0.002)	0.020(0.0003)	**0.013**(0.0001)	0.024(0.0005)	**0.05**(0.01)	**0.03**(0.003)	**0.02**(0.0001)	**0.05**(0.08)
KIN-8FH	0.07(0.002)	**0.06**(0.0007)	**0.046**(0.0002)	0.067(0.0005)	**0.06**(0.01)	**0.06**(0.004)	**0.07**(0.07)	**0.05**(0.003)

TABLE VII

MAE of the proposed approach and the state of the art in age estimation.

	Modified ε-constraint	**CV**	**GCV**	**SVR$_{rbf}$**	**SVR$_{pol}$**	**MKL-SVR**	**GPR**	**[48]**
MAE	**5.85**	6.59	13.83	6.46	6.95	7.18	15.46	5.97

TABLE VIII

RMSE of several approaches applied to weather prediction.

	Modified ε-constraint	**CV**	**GCV**	**SVR$_{rbf}$**	**SVR$_{pol}$**	**MKL-SVR**	**GPR**
RMSE	**0.81**	0.83	0.90	0.87	0.95	1.07	2.53

Algorithm 1

Modified ε-constraint algorithm

Input: Training set $\{(x_1, y_1), ..., (x_n, y_n)\}$, $\boldsymbol{\theta}_0$, h_0, ε_0, s.

1. Calculate the ideal vector point $\left(z_f^*, z_c^*\right)$.

2. Specify the weights w_f and w_c using (28).

3. Obtain ε^* using (27).

4. Obtain $\boldsymbol{\theta}^*$ using (26).

Return: The optimal model parameter $\boldsymbol{\theta}^*$.