Simplex basis function based sparse least squares support vector regression

Simplex Basis Function Based Sparse Least Squares Support Vector Regression

Xia Hong, Richard Mitchell, Giuseppe Di Fatta

Abstract

In this paper, a novel sparse least squares support vector regression algorithm, referred to as LSSVR-SBF, is introduced which uses a new low rank kernel based on simplex basis function, which has a set of nonlinear parameters. It is shown that the proposed model can be represented as a sparse linear regression model based on simplex basis functions. We propose a fast algorithm for least squares support vector regression solution at the cost of $O(N)$ by avoiding direct kernel matrix inversion. An iterative estimation algorithm has been proposed to optimize the nonlinear parameters associated with the simplex basis functions with the aim of minimizing model mean square errors using the gradient descent algorithm. The proposed fast least square solution and the gradient descent algorithm are alternatively applied. Finally it is shown that the model has a dual representation as a piecewise linear model with respect to the system input. Numerical experiments are carried out to demonstrate the effectiveness of the proposed approaches.

Index Terms

least squares support vector regression, low rank kernels, simplex basis function.

I. Introduction

With strong support from its underlying statistical learning theory, the support vector machine (SVM) [2] including the support vector machine for regression (SVR) [3] is a powerful modelling tool able to provide excellent model generalization. Originally the model estimation of SVM corresponds to a quadratic programming optimization problem. The SVM/SVR are regarded as sparse models which also avoid local minima for exceptional modeling performance, however they are computationally demanding. There are significant algorithmic developments such as sequential minimal optimization (SMO) [5] which is aimed at increasing computation speed. In the smooth SVR (SSVR) approach an accurate smooth approximation has been proposed to replace the insensitive loss function in SVR, hence relaxing the optimization problem into an unconstrained one [4], providing a significant speed up. A heuristic method based on a measurement of similarity among samples to reduce data samples for accelerating training was proposed in [6]. A novel geometric framework, based on which new SVR models and algorithms are proposed in [7], these use convex hull algorithms for data reduction. The SVR has also been applied in applications in nonstationary settings, e.g., time-series prediction in nonlinear environment [8], and interval regression analysis

The authors are with Department of Computer Science, School of Mathematical, Physical and Computational Sciences, University of Reading, Reading, RG6 6AY, UK. (x.hong/r.j.mitchell/g.difatta@reading.ac.uk)

in time-varying environment [9]. A major advance for computational efficiency is the introduction of least squares support vector machine (LSSVM) and least squares support vector regression (LSSVR), which uses equality constraints [10] so that closed form least squares type solutions are available. Note that LSSVM/LSSVR will produce a nonsparse model loosing sparseness property. In common to general kernel methods, LSSVM/LSSVR also have the limitation that the kernel function must be evaluated for all possible pairs of system inputs, potentially infeasible for large scale data modeling problems.

The modelling of real-world data from highly complex, nonlinear structures demands computationally fast kernel methods. Low-rank matrix approximation is an effective tool in alleviating the memory and computational burdens of kernel methods and sampling. For example, the Nyström method has been successfully applied to efficient kernel learning [11]. It randomly samples a subset of training examples and computes a kernel matrix for the random samples. It then represents each data point by a vector based on its kernel similarity to the random samples and the sampled kernel matrix. Alternatively the method of random Fourier features is another popular approach [12], in which the kernel function is approximated by the inner product of a randomized feature map. Note that these low rank kernel approximations will introduce additional generalization errors in the generation performance, which is of theoretic interest [13].

Alternatively, a nonlinear system can be approximated by locally linear systems as piecewise linear systems. Various piecewise linear models exist such as lattice piecewise linear representation [14], hinging hyperplanes (HH) [15] and piecewise affine models [16]. Notably the hinging hyperplane (HH), which uses a hinge function as basis functions, is shown to be a powerful model representation for nonlinear systems since it is endowed with proven approximation capabilities to arbitrary nonlinear functions [15]. Recently a new simplex basis function (SBF) model [20] has been introduced which can be viewed as a HH model and hence has the same approximation capability as HH. Often the use of linear functions to the system input brings the benefit of model interpretability. Moreover there are also rich linear system theory which could be exploited if the model is in the form of linear or piecewise linear form.

In this paper a new low rank kernel based on the simplex basis function has been introduced, based on which a novel sparse least squares support vector machine regression is developed. The proposed model can be represented as a sparse nonlinear SBF model, as well as a piecewise linear model. This framework enables the derivation of a fast algorithm for least squares support vector regression solution at the cost of $O(N)$. The proposed kernels come with a set of nonlinear parameters, which needs to be adjusted, and it is proposed that the gradient descent algorithm is applied to minimize the mean square errors. Our overall estimation algorithm therefore alternates between the fast LSSVR solution and SBF kernel parameters estimation of the gradient descent algorithm. The advantage of the proposed model is twofold; not only achieve fast LSSVR-SBF estimation, but also to obtain a sparse model which has a dual representation as a piecewise linear model which provides gradient information with ease. Specifically our proposed SBF based low kernels enable us to exploit the structural property of the resultant kernel matrix for fast matrix inversion, which is not valid for conventional Gaussian kernels commonly used in conventional LSSVR models. Similarly since we adopted the recently proposed simplex basis function (SBF) model [20], the model

requires us to devise a new gradient algorithm which is devoted to parameter estimation within the SBF functions.

The remainders of this paper are organized as follows. Section II describes preliminaries of the LSSVR model. Section III initially introduces the proposed kernels based on SBF, the LSSVR-SBF model representation a sparse model, and then learning algorithms are introduced including computational complexity analysis and the property of the dual representation of piecewise linear model. Numerical experiments are carried out in Section IV with comparisons with other benchmark approaches in terms of the modeling performance. Section V is devoted to conclusions.

II. Least squares support vector regression

Many kernels methods are based on a fixed nonlinear feature mapping in some feature space $\mathcal{F}$ as

$$\boldsymbol{x} \in \Re^{\mathrm{m}} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) \in \mathcal{F}, \tag{1}$$

endowed with a *Mercer* kernel function

$$k(\boldsymbol{x}(i), \boldsymbol{x}(j)) = \phi(\boldsymbol{x}(i))^{\mathrm{T}} \phi(\boldsymbol{x}(j)) \tag{2}$$

which is defined as the inner product of two points in the feature space $\mathcal{F}$ as a symmetric function of its arguments. Suppose a block of training data set denoted as D_N, comprising N input vectors $\boldsymbol{x}(1)$ ... $\boldsymbol{x}(N)$, with corresponding target values $y(1)$ ... $y(N)$ where $y(k) \in R$. The least square support vector machine for regression problem (LSSVR) [19] is based on a linear model of the form

$$\hat{y}(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \phi(\boldsymbol{x}) + b, \tag{3}$$

where $\boldsymbol{w}$ is vector of parameters. The optimization problem is given by

$$\frac{\gamma}{2} \sum_{n=1}^{N} e^2(n) + \frac{1}{2} \|\boldsymbol{w}\|^2, \tag{4}$$

subject to $y(n) = \boldsymbol{w}^{\mathrm{T}} \phi(\boldsymbol{x}(n)) + e(n),\ n = 1, ..., N$. Denote $\boldsymbol{e} = [e(1), e(2), ..., e(N)]^{\mathrm{T}}$ as the modeling error vector, and introduce $\boldsymbol{a} = [a_1, ..., a_N]^{\mathrm{T}}$, where a_n are Lagrange multipliers for each of the equality constraint, giving the Lagrangian function

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{e}; \boldsymbol{a}) &= \frac{\gamma}{2} \sum_{n=1}^{N} e^2(n) + \frac{1}{2} \|\boldsymbol{w}\|^2 \\ &- \sum_{n=1}^{N} a_n \{\boldsymbol{w}^{\mathrm{T}} \phi(\boldsymbol{x}(n)) + e(n) - y(n)\}, \end{aligned}$$

where $\gamma > 0$ is a preset regularization parameter. We now optimize out $\boldsymbol{w}, b$ and $\{e(n)\}$ to give

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{w}} = \mathbf{0} \rightarrow & \qquad \boldsymbol{w} = \textstyle\sum_{n=1}^{N} a_n \phi(\boldsymbol{x}(n)) \\
\frac{\partial L}{\partial b} = 0 \rightarrow & \qquad \textstyle\sum_{n=1}^{N} a_n = 0 \\
\frac{\partial L}{\partial e(n)} = 0 \rightarrow & \qquad a_n = \gamma e(n) \\
\frac{\partial L}{\partial a_n} = 0 \rightarrow & \qquad \boldsymbol{w}^{\mathrm{T}} \phi(\boldsymbol{x}(n)) + b + e(n) - y(n) = 0.
\end{aligned}$$

In many kernel methods we do not need to know $\boldsymbol{\phi}(\boldsymbol{x})$, but work on a well defined kernel function $k(\boldsymbol{x}, \boldsymbol{y})$ using the kernel trick due to (2). After eliminating $\boldsymbol{w}, e(n)$, and using kernel trick, the solution for the dual problem is

$$\begin{bmatrix} b \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \boldsymbol{K} + \boldsymbol{I}/\gamma \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \boldsymbol{y} \end{bmatrix} \tag{5}$$

where $\boldsymbol{y} = [y(1), ..., y(N)]^{\mathrm{T}}$, and the kernel matrix over the training input data set is a positive semi(definite) matrix and given as

$$\boldsymbol{K} = \begin{bmatrix} k(\boldsymbol{x}(1), \boldsymbol{x}(1)) & \cdots & k(\boldsymbol{x}(1), \boldsymbol{x}(N)) \\ \cdots & \cdots & \cdots \\ k(\boldsymbol{x}(N), \boldsymbol{x}(1)) & \cdots & k(\boldsymbol{x}(N), \boldsymbol{x}(N)) \end{bmatrix} \in \Re^{N \times N} \tag{6}$$

$\boldsymbol{I}$ is the identity matrix and $\mathbf{1}$ is the vector of all ones with appropriate dimensions. The model prediction is given by

$$\hat{y}(\boldsymbol{x}) = \sum_{n=1}^{N} a_n k(\boldsymbol{x}, \boldsymbol{x}(n)) + b. \tag{7}$$

In spite of the advantage of providing a neat closed form solution, LSSVR has two limitations as would happen in other kernel methods. The computational cost of matrix inversion is in the order of $O(N^3)$, meaning that it does not scale well with data size N, limiting the training data size to be moderate. Additionally, the LSSVR is not a sparse model since the model size is the same as train sample size N, implying that to evaluate a new data point one needs to access all the training data set. In general, one disadvantage associated with most kernel methods is that they do not scale well with large data sets. Hence the low rank kernel approximations are often considered [11, 12] since they allow reduction of computation time to $O(N^2)$. In this work we propose a new type of low rank kernel, that lends itself to new learning algorithms and model representation free of the above two limitations.

III. Sparse least squares support vector regression based on low rank SBF kernels

In the following a new type of low rank kernels is initially introduced for LSSVR which leads to some desirable properties. The proposed kernel has a set of nonlinear parameters and model size M, hence in addition to estimating b and $\boldsymbol{a}$, our modeling task includes estimation of these extra kernel parameters. Our proposed model estimation algorithm is an iterative and hybrid one with the aim of gaining computational advantage by exploiting the special model functional structure induced by the proposed low rank kernel.

We propose a new type of low rank kernel in the form of

$$k(\boldsymbol{x},\boldsymbol{y}) = \sum_{j=1}^{M} \phi_j(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{c}_j)^{\mathrm{T}} \phi_j(\boldsymbol{y};\boldsymbol{\mu}_j,\boldsymbol{c}_j), \tag{8}$$

where $\phi_j(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{c}_j)$ is the simplex basis function (SBF) defined as [20]

$$\phi_j(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{c}_j) = \max\left(0, 1 - \sum_{i=1}^{m} \mu_{i,j}|x_i - c_{i,j}|\right) \tag{9}$$

in which $\boldsymbol{c}_j = \begin{bmatrix} c_{1,j} & c_{2,j} \cdots c_{m,j} \end{bmatrix}^{\mathrm{T}} \in \Re^m$ is known as the center vector of the jth SBF unit which controls the location of jth SBF, and $\boldsymbol{\mu}_j = \begin{bmatrix} \mu_{1,j} & \mu_{2,j} \cdots \mu_{m,j} \end{bmatrix}^{\mathrm{T}} \in \Re_+^m$ is the shape parameters vector that control the shape of jth SBF. The output of SBF functions is illustrated in Fig. 1. From Fig. 1(a) it is clear that the maximum output is limited as one at the centers, and the responsive region width in one input direction is $\frac{1}{\mu}$, inversely proportional to the shaping parameter μ. From Fig. 2(b) it can be seen the SBF output is only nonzeros around the center within a hyper-polygon shaped region. Within this responsive region, it is composed of 2^m locally linear models, each occupying one of 2^m subregions. The SBF can be visually compared with the well known Gaussian radial basis function (RBF) via Fig. 1(a) versus its counterpart Fig. 1(b) as well as Fig. 1(c) versus its counterpart Fig. 1(d).

Given a collection of data D_N, clearly the associated kernel matrix is given by

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} \tag{10}$$

with $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_M]$. $\boldsymbol{\phi}_j = [\phi(\boldsymbol{x}(1)), ..., \phi(\boldsymbol{x}(N))]^{\mathrm{T}}$. Note that (6) can be represented as

$$\begin{aligned} \hat{y}(\boldsymbol{x}) &= \sum_{n=1}^{N} a_n \sum_{j=1}^{M} \phi_j(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{c}_j)^{\mathrm{T}} \phi_j(\boldsymbol{x}(n);\boldsymbol{\mu}_j,\boldsymbol{c}_j) + b \\ &= \sum_{j=1}^{M} \theta_j \phi_j(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{c}_j) + b = [\phi(\boldsymbol{x})]^{\mathrm{T}}\boldsymbol{\theta} + b \end{aligned} \tag{11}$$

where $\theta_j = \sum_{n=1}^{N} a_n \phi_j(\boldsymbol{x}(n);\boldsymbol{\mu}_j,\boldsymbol{c}_j)$, $\boldsymbol{\theta} = [\theta_1, ..., \theta_M]^{\mathrm{T}}$, which can be expressed in vector form as

$$\boldsymbol{\theta} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{a}. \tag{12}$$

Equation (12) bridges the original nonsparse LSSVR into a sparse SBF model with only M terms, based on M SBF basis functions. In the proposed algorithm, we will set M as small as possible depending on data sets. From our experience very sparse model can be obtained in comparison with other modeling schemes due to the flexibility of the SBF basis functions.

A. Fast LSSVR parameter estimation

Given that the set of nonlinear parameters $\boldsymbol{c}_j$ and $\boldsymbol{\mu}_j$, $\boldsymbol{\Phi}$ becomes fixed, we propose a fast way of calculating (5) by exploiting the special matrix property induced by the proposed low rank kernel. Specifically, it is shown that the

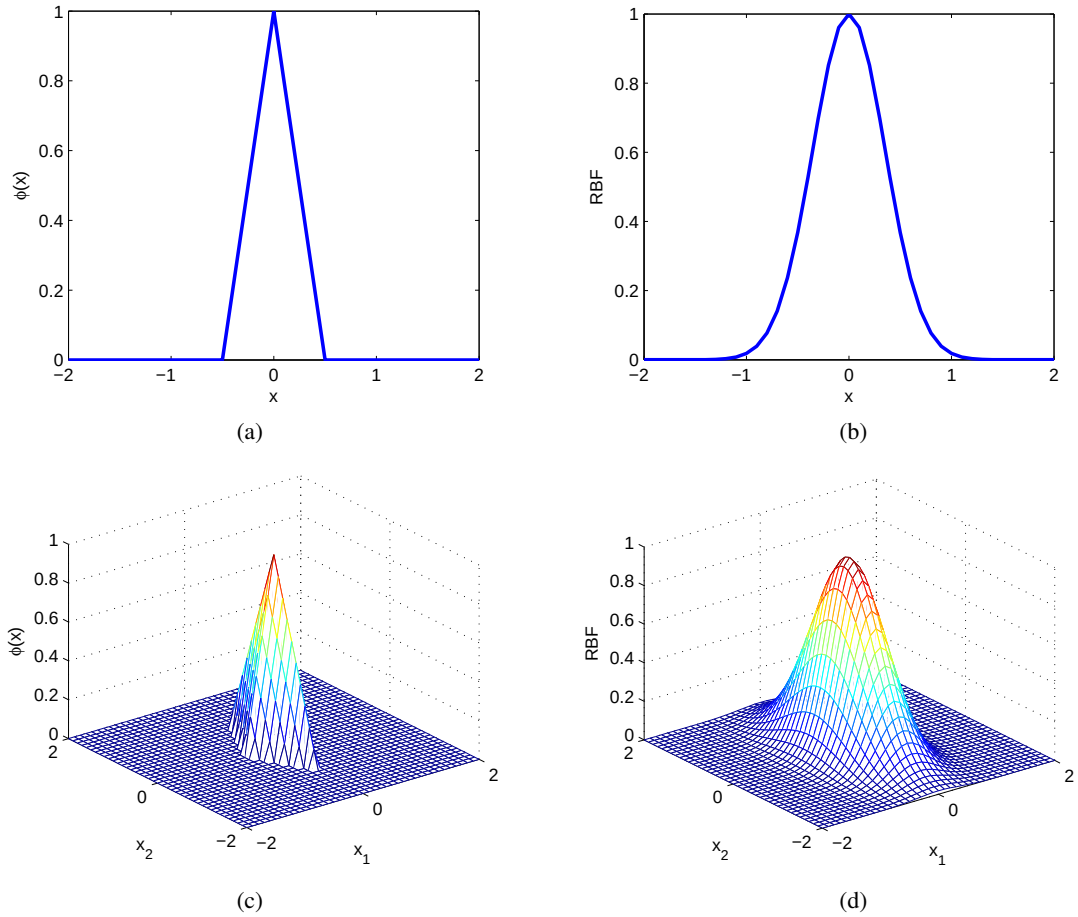


Fig. 1. Graphical illustration of SBF functions centered at origin in comparison with RBF functions; (a) 1D function $\phi(x) = \max\left(0, 1 - 2|x|\right)$;(b) 1D RBF function function $\exp\left(-4x^2\right)$; (b) 2D function $\phi(\boldsymbol{x}) = \max\left(0, 1-2|x_1|-|x_2|\right)$; (d) 2D RBF function $\exp\left(-4x_1^2-x_2^2\right)$

computational cost of matrix inversion can be reduced to $O(MN)$, enabling fast parameter estimation. Defining

$$\tilde{\boldsymbol{\Phi}} = \left[\begin{array}{ccc} 0 & \cdots & 0 \\ \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_M \end{array}\right] \in \Re^{(N+1)\times M} \tag{13}$$

(5) becomes

$$\left[\begin{array}{c} b \\ \boldsymbol{a} \end{array}\right] = \left\{\left[\begin{array}{cc} 0 & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \boldsymbol{I}/\gamma \end{array}\right] + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{\Phi}}^{\mathrm{T}}\right\}^{-1} \left[\begin{array}{c} 0 \\ \boldsymbol{y} \end{array}\right]. \tag{14}$$

Applying the matrix inversion lemma to (14), we obtain

$$\begin{aligned}
&\begin{bmatrix} b \\ \boldsymbol{a} \end{bmatrix} = \\
&\Big(\boldsymbol{I} - \begin{bmatrix} 0 & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \boldsymbol{I}/\gamma \end{bmatrix}^{-1} \tilde{\boldsymbol{\Phi}}\big(\boldsymbol{I} + \tilde{\boldsymbol{\Phi}}^{\mathrm{T}} \begin{bmatrix} 0 & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \boldsymbol{I}/\gamma \end{bmatrix}^{-1} \tilde{\boldsymbol{\Phi}}\big)^{-1}\tilde{\boldsymbol{\Phi}}^{\mathrm{T}}\Big) \\
&\begin{bmatrix} 0 & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \boldsymbol{I}/\gamma \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \boldsymbol{y} \end{bmatrix} \\
&\stackrel{def}{=} \boldsymbol{q} - \boldsymbol{P}\tilde{\boldsymbol{\Phi}}\big(\boldsymbol{I} + \tilde{\boldsymbol{\Phi}}^{\mathrm{T}}\boldsymbol{P}\tilde{\boldsymbol{\Phi}}\big)^{-1}\tilde{\boldsymbol{\Phi}}^{\mathrm{T}}\boldsymbol{q} \\
&\stackrel{def}{=} \boldsymbol{q} - \boldsymbol{P}\tilde{\boldsymbol{\Phi}}\big(\boldsymbol{I} + \boldsymbol{Q}\big)^{-1}\tilde{\boldsymbol{\Phi}}^{\mathrm{T}}\boldsymbol{q}
\end{aligned} \tag{15}$$

in which

$$\boldsymbol{P} = \begin{bmatrix} 0 & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \boldsymbol{I}/\gamma \end{bmatrix}^{-1} = \frac{1}{N}\begin{bmatrix} -1/\gamma & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \gamma(N\boldsymbol{I} - \mathbf{1}\mathbf{1}^{\mathrm{T}}) \end{bmatrix}, \tag{16}$$

and

$$\begin{aligned}
\boldsymbol{q} &= \frac{1}{N}\begin{bmatrix} -1/\gamma & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \gamma(N\boldsymbol{I} - \mathbf{1}\mathbf{1}^{\mathrm{T}}) \end{bmatrix}\begin{bmatrix} 0 \\ \boldsymbol{y} \end{bmatrix} \\
&= \frac{1}{N}\begin{bmatrix} \mathbf{1}^{\mathrm{T}}\boldsymbol{y} \\ \gamma(N\boldsymbol{y} - \mathbf{1}\mathbf{1}^{\mathrm{T}}\boldsymbol{y}) \end{bmatrix}
\end{aligned} \tag{17}$$

and

$$\boldsymbol{Q} = \tilde{\boldsymbol{\Phi}}^{\mathrm{T}}\boldsymbol{P}\tilde{\boldsymbol{\Phi}} = \gamma\tilde{\boldsymbol{\Phi}}^{\mathrm{T}}\tilde{\boldsymbol{\Phi}} - N\gamma\bar{\boldsymbol{\phi}}\bar{\boldsymbol{\phi}}^{\mathrm{T}}. \tag{18}$$

in which $\bar{\boldsymbol{\phi}} = [\bar{\phi}_1, ..., \bar{\phi}_M]^{\mathrm{T}}$, and $\bar{\phi}_j = \frac{1}{N}\sum_{k=1}^{N}\phi_j(k)$.

The proposed algorithm needs to be initialized with a predetermined model size M and an initial design matrix $\boldsymbol{\Phi}$, which is based on preset values of $\boldsymbol{c}_j, \boldsymbol{\mu}_j$, $j = 1, ...M$. Clustering algorithms can be used to initialize the centers $\boldsymbol{c}_j$, which accurately reflects the distribution of the data points. The $k-$means [21] clustering algorithm is applied to obtain initial simplex function centers, which is detailed in Algorithm 1 for completeness. While for shaping parameters $\mu_{i,j}$ in all simplex functions, it is uniformly initialized as $\mu_{i,j} = \mu$, where $\mu > 0$ is a predetermined constant, i.e. we preset $\boldsymbol{\mu}_j = \mu\mathbf{1}$.

The fast LSSVR algorithm using SBF low rank kernel is summarized in Algorithm 2, of which the computational cost is determined by a series of matrix-vector multiplication with the cost at $O(NM)$, matrix multiplication at $O(NM^2)$, and the matrix inverse in computing (15) costs only $O(M^3)$, which is negligible in the case of $M \ll N$.

Algorithm 1 The k-means clustering algorithm

Require: $\boldsymbol{x}(k)$, $k = 1, \cdots, N$, and a preset number of centers M.
Ensure: $\boldsymbol{c}_j$, $j = 1, \cdots, M$, that yields the minimum of $J = \sum_{j=1}^{M} \sum_{\boldsymbol{x}(k) \in S_j} \|\boldsymbol{x}(k) - \boldsymbol{c}_j\|^2$

1: Randomly select M data points from D_N as initial centers $\boldsymbol{c}_j^{old}$.
2: Randomly draw a data point $\boldsymbol{x}(k)$ from D_N.
3: From all $\boldsymbol{c}_j^{old}$, $j = 1, \cdots, M$, find the nearest center to $\boldsymbol{x}(k)$, denoted as $\boldsymbol{c}_i^{old}$.
4: Update $\boldsymbol{c}_i^{new} = \boldsymbol{c}_i^{old} + \epsilon(\boldsymbol{x}(k) - \boldsymbol{c}_i^{old})$, where $\epsilon > 0$ is the predetermined learning rate.
5: Set $\boldsymbol{c}_i^{new}$ as $\boldsymbol{c}_i^{old}$.
6: Goto Step 2 until a sufficient large number of iterations, or convergence has been reached.
7: Return $\boldsymbol{c}_j^{new}$ as $\boldsymbol{c}_j$, $i = 1, \cdots, M$.

Algorithm 2 Summary of fast LSSVR algorithm using SBF low rank kernel.

Require: $\boldsymbol{\Phi}$, $\boldsymbol{y}$.
Ensure: Ensure b and $\boldsymbol{a}$ are found for given $\boldsymbol{\Phi}$.

1: Construct $\tilde{\boldsymbol{\Phi}}$.
2: Calculate b, $\boldsymbol{a}$ using (15)-(18).
3: Return b, $\boldsymbol{a}$.

B. Iterative estimation of SBF kernels using proposed gradient descent algorithm

Given a training data set D_N, consider parameter estimation for the proposed kernel which is specified by a set of nonlinear parameters $\boldsymbol{c}_j$ and $\boldsymbol{\mu}_j$ ($j = 1, ...M$). We propose an iterative approach by adjusting $\boldsymbol{c}_j, \boldsymbol{\mu}_j$, $\forall j$, associated with $\phi_j(\boldsymbol{x})$ using a new gradient descent algorithm, while $\{\boldsymbol{a}, b\}$ are fixed. Our optimization criterion is based on minimizing the sum of squares of errors (SSE), given by

$$\begin{aligned} J^{(j)}(\boldsymbol{c}_j, \boldsymbol{\mu}_j) &= \sum_{n=1}^{N} e^2(\boldsymbol{x}(k)) \\ &= \boldsymbol{e}^{\mathrm{T}}\big(\boldsymbol{y} - \boldsymbol{K}\boldsymbol{a} - b\mathbf{1}\big). \end{aligned} \tag{19}$$

We have

$$\begin{cases} \frac{\partial J^{(j)}}{\partial \mu_{i,j}} = -\boldsymbol{e}^{\mathrm{T}} \frac{\partial \boldsymbol{K}}{\partial \mu_{i,j}} \boldsymbol{a} & i = 1, ..., m \\ \frac{\partial J^{(j)}}{\partial c_{i,j}} = -\boldsymbol{e}^{\mathrm{T}} \frac{\partial \boldsymbol{K}}{\partial c_{i,j}} \boldsymbol{a} & i = 1, ..., m \end{cases} \tag{20}$$

in which

$$\begin{aligned} \frac{\partial \boldsymbol{K}}{\partial \mu_{i,j}} &= (\frac{\partial}{\partial \mu_{i,j}} \boldsymbol{\phi}_j)\boldsymbol{\phi}_j^{\mathrm{T}} + \boldsymbol{\phi}_j(\frac{\partial}{\partial \mu_{i,j}} \boldsymbol{\phi}_j)^{\mathrm{T}} \\ \frac{\partial \boldsymbol{K}}{\partial c_{i,j}} &= (\frac{\partial}{\partial c_{i,j}} \boldsymbol{\phi}_j)\boldsymbol{\phi}_j^{\mathrm{T}} + \boldsymbol{\phi}_j(\frac{\partial}{\partial c_{i,j}} \boldsymbol{\phi}_j)^{\mathrm{T}} \end{aligned} \tag{21}$$

where

$$\begin{aligned} \frac{\partial}{\partial \mu_{i,j}} \boldsymbol{\phi}_j &= [\frac{\partial \phi_j(\boldsymbol{x}(1))}{\partial \mu_{i,j}}, ..., \frac{\partial \phi_j(\boldsymbol{x}(N))}{\partial \mu_{i,j}}]^{\mathrm{T}} \\ \frac{\partial}{\partial c_{i,j}} \boldsymbol{\phi}_j &= [\frac{\partial \phi_j(\boldsymbol{x}(1))}{\partial c_{i,j}}, ..., \frac{\partial \phi_j(\boldsymbol{x}(N))}{\partial c_{i,j}}]^{\mathrm{T}} \end{aligned} \tag{22}$$

Algorithm 3 The proposed LSSVR Algorithm with SBF Kernels.

Require: Data D_N. Model size M. Regularization parameter γ. Initial shaping parameter μ. Iteration number $Iter$.
Ensure: Parameters ($\boldsymbol{c}_j, \boldsymbol{\mu}_j$, $j = 1, ...M$), b, $\boldsymbol{a}$ are obtained.

```
Apply Algorithm 1 (the k−means clustering algorithm) to initialize c_j, j = 1,...M. Set all μ_{i,j} as μ.
for l = 1, ..., Iter do
    Form Φ from D_N based on c_j, μ_j, j = 1,...M.
    Apply Algorithm 2 to find b, a.
    for l = 1, ..., M do
        Adjust c_j, μ_j using (20)-(27).
    end for
    Apply Algorithm 2 to update b, a.
end for
Return c_j, μ_j, j = 1,...M, b and a.
```

which are calculated by

$$\frac{\partial \phi_j(\boldsymbol{x}(k))}{\partial \mu_{i,j}} = -|x_i(k) - c_{i,j}| Id(k), \quad i = 1, ..., m \tag{23}$$

$$\frac{\partial \phi_j(\boldsymbol{x}(k))}{\partial c_{i,j}} = \mu_{i,j} \text{sign}(x_i(k) - c_{i,j}) Id(k), \quad i = 1, ..., m \tag{24}$$

in which $Id(k)$ is an indication function given as

$$Id(k) = \begin{cases} 1 & \text{if } \sum_{i=1}^{m} \mu_{i,j} |x_i(k) - c_{i,j}| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

and

$$\text{sign}(s) = \begin{cases} 1 & s > 0 \\ 0 & s = 0 \\ -1 & s < 0 \end{cases}. \tag{26}$$

Finally by taking into account the positive constraints for the shaping parameters $\boldsymbol{\mu}_j$, we propose the constrained normalized gradient descent algorithm, as expressed as

$$\begin{cases} c_{i,j} &= c_{i,j} - \eta \cdot \frac{\partial J^{(j)}}{\partial c_{i,j}} / \| \frac{\partial J^{(j)}}{\partial \boldsymbol{c_j}} \| \\ \tilde{\mu}_{i,j} &= \mu_{i,j} - \eta \cdot \frac{\partial J^{(j)}}{\partial \mu_{i,j}} / \| \frac{\partial J^{(j)}}{\partial \boldsymbol{c_j}} \| \\ \mu_{i,j} &= \max\left(0, \tilde{\mu}_{i,j}\right) \end{cases} \tag{27}$$

for $i = 1, ...m$, where $\eta > 0$ is a preset smaller learning rate.

To update all SBF kernel, Equations (20)-(27) are simply applied to M SBF units ($j = 1, ..., M$) in turn while fixing b, $\boldsymbol{a}$, and all other SBF units. The computational cost is therefore $O(M^2 N)$. Note that this low computation cost is based on the fact that (21) has low rank, and in calculating the gradient vector (20), (21) can be in its split form, which are premultiplied with $\boldsymbol{e}^{\mathrm{T}}$, $\boldsymbol{a}^{\mathrm{T}}$ respectively.

C. Summary of the LSSVR-SBF algorithm

The proposed algorithm is summarized in Algorithm 3. It starts with k-means clustering algorithm for initialization of SBF centers with a cost at $O(N)$, then the fast least squares LSSVR solution in Section III-A and gradient decent algorithm in Section III-B are alternatively applied for a predetermined number of iterations. The overall computational complexity is dominated by the gradient descent algorithm with $O(M^2N)$. Hence our proposed algorithm has a linear complexity of $O(N)$, scaled further by model size M, number of intermediate variables and iteration number. Thanks to the special structure of the proposed low rank kernels, which enables fast least squares solution as proposed, this further allows the kernels to be trained using the gradient descent algorithm iteratively. Furthermore we can often obtain a very sparse model due to the use of the flexible SBF kernels which are learnt from data automatically, with very little presetting parameters. The overall algorithm is based on the alternative application of the LSSVR and gradient descent algorithms which have known convergence. LSSVR is a global optimal algorithm which finds the optimal b, $\boldsymbol{a}$. The gradient descent algorithm depends on the learning rate η which should be set as sufficiently small so that the MSE is decreasing per iteration for general smooth function. Since the SBF model is not based on a smooth function, the MSE per time step may still be slightly increasing even for very small learning rates, so that the MSE is in general decreasing but not smooth.

D. Piecewise linear model dual representation

In the following, a special property is analyzed, which is referred to as the dual representation of SBF as a piecewise linear model (Lemma 1).

Lemma 1: The SBF model $\hat{y}(\boldsymbol{x})$ can be represented as a piecewise locally linear model with respect to input $\boldsymbol{x}$ as

$$\hat{y}(\boldsymbol{x}) = \boldsymbol{\alpha}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{x} + \beta(\boldsymbol{x}) + b, \tag{28}$$

where $\boldsymbol{\alpha}(\boldsymbol{x})$ and $\beta(\boldsymbol{x})$ are piecewise constants, with the properties

$$(i)\ \ \frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{\alpha}(\boldsymbol{x}) = \mathbf{0}, \quad \frac{\partial}{\partial \boldsymbol{x}}\beta(\boldsymbol{x}) = 0, \tag{29}$$

$$(ii)\ \ \frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{\alpha}(\boldsymbol{x}). \tag{30}$$

Proof: Consider any given input vector $\boldsymbol{x}$, (5) can alternatively be represented as

$$\hat{y}(\boldsymbol{x}) = \sum_{j \in S(\boldsymbol{x})} \theta_j \Big(1 - \sum_{i=1}^{m} \mu_{i,j} |x_i - c_{i,j}|\Big), \tag{31}$$

where $S(\boldsymbol{x}) \in [1, ..., M]$ is the index set of j, satisfying condition $\sum_{i=1}^{m} \mu_{i,j}|x_i - c_{i,j}| < 1$. We have

$$\begin{aligned}\hat{y}(\boldsymbol{x}) &= \sum_{j \in S(\boldsymbol{x})} \theta_j - \sum_{j \in S(\boldsymbol{x})} \theta_j \sum_{i=1}^{m} \mu_{i,j}|x_i - c_{i,j}| + b \\ &= \sum_{i=1}^{m} x_i \sum_{j \in S(\boldsymbol{x})} \theta_j \mu_{i,j} \text{sign}\big(c_{i,j} - x_i\big) \\ &+ \sum_{j \in S(\boldsymbol{x})} \theta_j \big(1 - \sum_{i=1}^{m} \mu_{i,j} c_{i,j} \text{sign}\big(c_{i,j} - x_i\big)\big) + b \\ &= \boldsymbol{\alpha}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{x} + \beta(\boldsymbol{x}) \end{aligned} \tag{32}$$

and $\boldsymbol{\alpha}(\boldsymbol{x}) = [\alpha_1(\boldsymbol{x}), ..., \alpha_m(\boldsymbol{x})]^{\mathrm{T}}$, in which

$$\begin{aligned}\alpha_i(\boldsymbol{x}) &= \sum_{j \in S(\boldsymbol{x})} \theta_j \mu_{i,j} \text{sign}\big(c_{i,j} - x_i\big), i = 1, ..., m \\ \beta(\boldsymbol{x}) &= \sum_{j \in S(\boldsymbol{x})} \theta_j \big(1 - \sum_{i=1}^{m} \mu_{i,j} c_{i,j} \text{sign}\big(c_{i,j} - x_i\big)\big) + b \end{aligned} \tag{33}$$

So that we have $\frac{\partial}{\partial x_i} \alpha_i(\boldsymbol{x}) = 0$, $\frac{\partial}{\partial x_i} \beta(\boldsymbol{x}) = 0$. Hence

$$\frac{\partial}{\partial \boldsymbol{x}} \hat{y}(\boldsymbol{x}) = \boldsymbol{\alpha}(\boldsymbol{x}). \tag{34}$$

This concludes the proof. ■

Clearly SBF's dual representation as a piecewise linear model with respect to input vector $\boldsymbol{x}$ is useful for extracting gradient information from an identified model in the similar way as a linear model, except that these are locally dependent. Moreover, since there are abundant theories in linear systems, e.g. in linear control systems/signal processing, this piecewise linear model representation can be further exploited for its usefulness depending on applications, which are our current/future work. For example, the proposed model can be extended to a generalized predictive controller where the gradient information are needed for calculating the optimal control signals [22], in which the least square support regression model has been successfully applied. Further works will appear as separate publications.

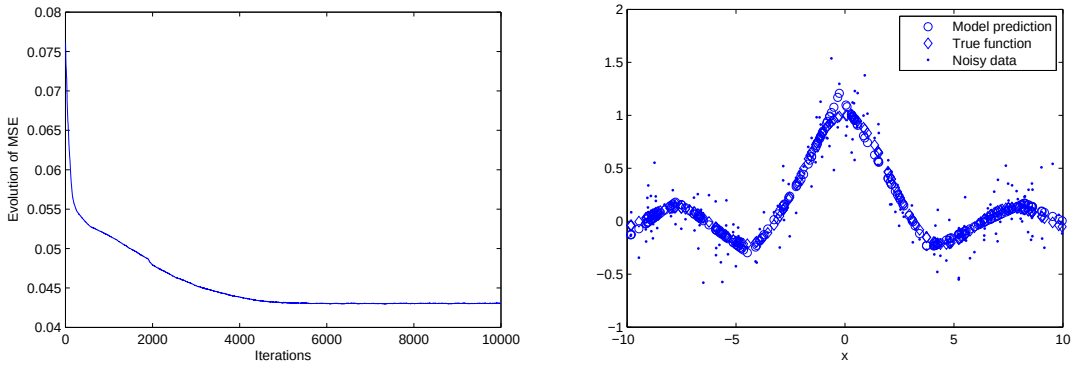
IV. Simulation Study

Example 1 (Nonlinear static function approximation): Consider using LSSVR-SBF to approximate an unknown scalar function

$$f(x) = \frac{\sin(x)}{x}. \tag{35}$$

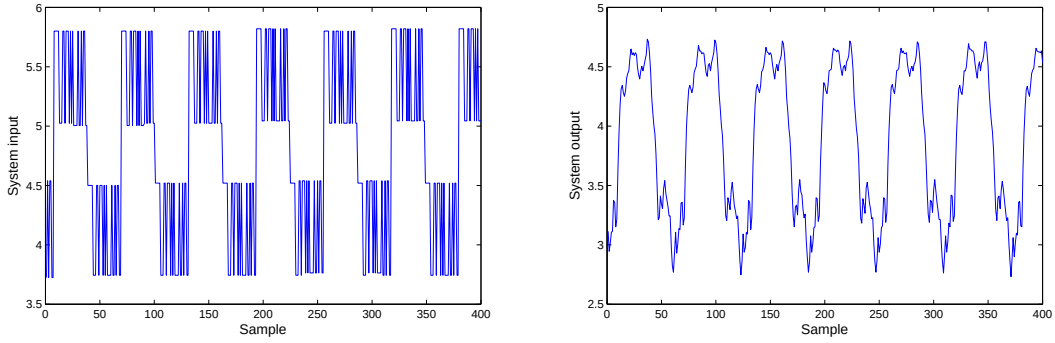
A data set of two hundred points was generated from $y = f(x) + \xi$, where the input x was uniformly distributed in [-10,10] and the noise ξ was Gaussian with zero mean and standard deviation 0.2. The data were very noisy. The proposed LSSVR-SBF algorithm was applied with model size $M = 3$, $\mu = 0.2$, $Iter = 10000$, $\gamma = 500$, and the learning rate $\eta = 0.001$. The modeling results are plotted in Fig. 2 (a) and (b), respectively. Fig. 2 (b) demonstrates

that the model can approximate the true underlying function well. In fact the modeling MSE with respective to the true function converges to 0.0022.



(a) (b)

Fig. 2. Results of Example 1: (a)The evolution of mean square error (MSE) and (b) Noisy data, true function and model predictions of a three term model.



(a) (b)

Fig. 3. Engine Data: (a) the system input $u(k)$ and (b) the system output $y(k)$

Example 2 (Nonlinear dynamical system): This Engine data set [23] contains 410 data samples of the fuel rack position (the input $u(k)$) and the engine speed (the output $y(k)$), collected from a Leyland TL11 turbocharged, direct injection diesel engine which was operated at a low engine speed. The 410 input and output data points of the engine data set are plotted in Fig. 3 (a) and (b), respectively. The first 210 data samples were used in training and the last 200 data samples for model testing. The previous study has shown that the data set can be modeled adequately using the system input vector $\boldsymbol{x}(k) = \left[y(k-1)\ u(k-1)\ u(k-2)\right]^{\mathrm{T}}$, and the best Gaussian RBF model was provided by the l^2-norm local regularization assisted OLS (LROLS) algorithm based on the LOOMSE (LROLS-LOO) [24] which was quoted in Table II for comparison. The ε-SVM algorithm [25] and the LASSO were

also experimented based on the Gaussian kernel with a common variance τ^2. For the ε-SVM, the Matlab function *quadprog.m* was used with the algorithm option set as 'interior-point-convex'. The tuning parameters in the ε-SVM algorithm, such as soft margin parameter C [25], were set empirically so that the best possible result was obtained after several trials. For the LASSO, the Matlab function *lasso.m* was used with 10-fold CV being used to select the associated regularization parameter. For both the ε-SVM and LASSO, we list the results obtained for a range of kernel width τ values in Table II, for comparison. The proposed LSSVR-SBF algorithm was applied, with the parsimonious principle in mind, by setting the desired model size M as small as possible. For all model sizes, the experimental setting parameters are uniformly set as $\mu = 0.2$, $Iter = 2000$, $\gamma = 1000$, and the learning rate $\eta = 0.002$. The results obtained for a range of model size M are listed in Table I. It can be seen that the proposed approach is capable of producing the excellent predictive performance for this benchmark example, even when the model size is only ten.

TABLE I
COMPARISON OF THE MODELING PERFORMANCE FOR ENGINE DATA.

Algorithm	MSE training set	MSE test set	Model size
LROLS-LOO [24]	0.000453	0.000490	22
ε-SVM ($\tau = 3$)	0.000502	0.000482	208
ε-SVM ($\tau = 2.5$)	0.000480	0.000475	208
ε-SVM ($\tau = 2$)	0.000461	0.000486	208
ε-SVM ($\tau = 1.5$)	0.000415	0.000579	208
ε-SVM ($\tau = 1$)	0.000370	0.000794	208
LASSO ($\tau = 1.5$)	0.000923	0.001010	70
LASSO ($\tau = 1$)	0.000708	0.000748	44
LASSO ($\tau = 0.5$)	0.000706	0.000842	54
LASSO ($\tau = 0.2$)	0.000565	0.000800	81
LASSO ($\tau = 0.1$)	0.000644	0.001907	76
Proposed LSSVR-SBF	0.000473	0.000487	10
Proposed LSSVR-SBF	0.000440	0.000477	15
Proposed LSSVR-SBF	0.000437	0.000464	20

Example 3 (Nonlinear dynamical system): The continuous-stirred tank reactor (CSTR) plant is very common in chemical and petrochemical plants. It has an irreversible, exothermic reaction that occurs in a constant volume reactor and is cooled by a single coolant stream [26]. Within the CSTR two chemicals are mixed and react to produce a product compound A at a concentration $C_a(t)$, with the temperature of the mixture being $T(t)$. The system output concentration $C_a(t)$ is controlled by regulating the coolant flow rate $q_c(t)$ (system input). The CSTR can be simulated by the following continuous-time, nonlinear, simultaneous, differential equations:

$$\frac{d}{dt}C_a(t) = \frac{q}{v}\big(C_{a0} - C_a(t)\big) - k_0 C_a(t) e^{-\frac{E}{RT(t)}} \tag{36}$$

$$\frac{d}{dt}T(t) = \frac{q}{v}\big(T_0 - T(t)\big) + k_1 C_a(t) e^{-\frac{E}{RT(t)}} + k_2 q_c(t)\big(1 - e^{-\frac{k_3}{q_c(t)}}(T_{c0} - T(t))\big) \tag{37}$$

where $k_1 = -\Delta H k_0/\rho C_p$, $k_2 = \rho_c C_{pc}/\rho C_p v$, $k_3 = h_a/\rho_c C_{pc}$, in which CSTR parameters are given in Table II. For nonlinear system identification of the CSTR, the system input are designed as persistent exciting signal, and we obtain the data set containing 7500 input and output data points from [27]. We build the model based on the normalized data set, with standard deviation for both input and output data as one. Then the output is added a zero mean Gaussian noise with variance of 4×10^{-4}. 2000 data points are used for training, and the rest are for validation. The system input vector was set as $\boldsymbol{x}(k) = \big[y(k-1), ..., y(k-n_y),\ u(k-1), ..., u(k-n_u)\big]^{\mathrm{T}}$, where $n_y = n_u = 3$ are adopted according to [26]. The proposed LSSVR-SBF algorithm was applied using model size $M = 5$. The experimental setting parameters are uniformly set as $\mu = 0.01$, $Iter = 5000$, $\gamma = 5000$, and the learning rate $\eta = 0.001$. We obtained MSE for training and validation data set of 4.87×10^{-4}, 4.85×10^{-4}, respectively. For comparison the ε-SVM algorithm [25] was applied in a similar fashion as in Example 2, and $\tau = 2.5$ was empirically found to the best choice of kernel width. The MSE for training and validation data set of 3.68×10^{-4}, 6.45×10^{-4} were obtained for ε-SVM algorithm, although the model size was found as 1997, i.e., it uses all training data points as support vectors. It can be seen that the proposed approach is capable of producing the excellent predictive performance for this benchmark example, even when the model size is only five. The modeling results for this example are plotted in Fig. 4 (a)-(c) respectively. This experiment was carried out on a PC using processor Intel i5-2500 CPU under Matlab, and we recorded the running time as 380 seconds.

TABLE II
THE CSTR PARAMETERS

Parameter	Description	Nominal Value
q	process flow rate	100 l/min
v	reaction volume	100 l
k_0	reaction rate constant	$7.2 \times 10^{10}\mathrm{min}^{-1}$
E/R	activation energy	1×10^4K
T_0	feed temperature	300 K
T_{co}	inlet coolant temperature	300K
ΔH	heat of reaction	-2×10^5 cal/mol
C_{a0}	Inlet feed concentration	1 mol/l
C_p, C_{pc}	specific heats	1 cal/g/K
ρ, ρ_c	liquid densities	1×10^3 g/l
h_a	heat transfer coefficient	7×10^5 cal/min/K

V. CONCLUSIONS

In this paper we have introduced a least squares support vector regression algorithm using a novel low rank kernel based on a simplex basis function. The proposed model lends itself to efficient computation due to low rank kernel matrix. A fast least squares solution has been proposed without matrix inversion to achieve $O(N)$ complexity. It is shown that the proposed model is equivalent to a sparse SBF model. Since the proposed kernel is parameterized, the learning algorithm is proposed which alternates between two sub-algorithms (i) the proposed fast least square algorithm, while the simplex basis functions are fixed and (ii) the gradient descent algorithm for nonlinear SBF

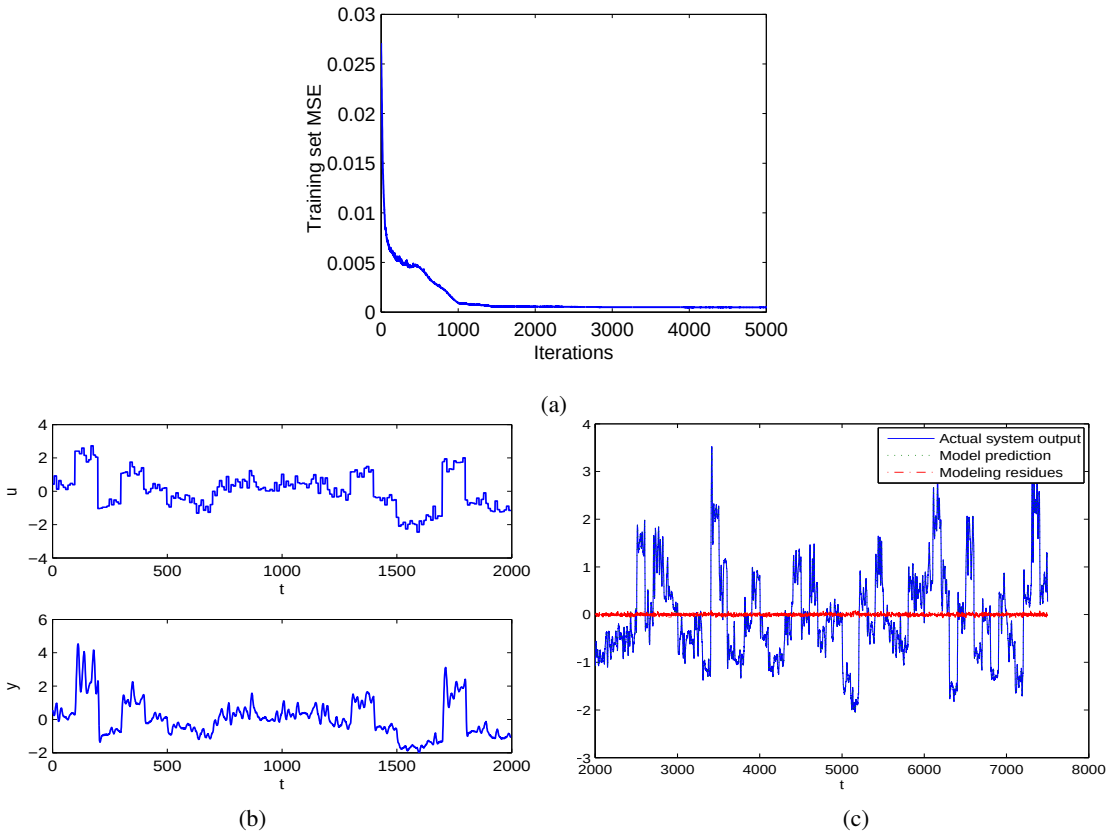


Fig. 4. CSTR Data: (a) The evolution of MSE for the training set; (b) the system input/output data set (2000 data points); (b) the system output, model prediction and model residues for validation data set (6500 data points).

parameters in which each simplex basis function parameters are adjusted by minimizing sum of squares errors. We have shown the proposed model can be dually represented as a piecewise linear model, which can be useful for extracting gradient information for applications requiring knowledge discovery and control. Numerical experiments are performed to demonstrate the effectiveness of the proposed approach. Current work is devoted to extending the method to generalized predictive controller where the gradient information are needed for calculating the optimal control signals in LSSVR setting [22], which will appear in a separate publication.

REFERENCES

[1] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, Local features and kernels for classification of texture and object categories: A comprehensive study, *International Journal of Computer Vision*, Vol. 73, No 2, pp213238, 2007

[2] V. Vapnik, *Statistical Learning Theory*, Wiley, 1998.

[3] A. J. Smola, B. Sch´olkopf, “A tutorial on support vector regression,” *Statistics and Computing*, Vol 14, pp199-222, 2004.

[4] Y. J. Lee, W. F. Hsieh, C. M. Huang,"ε-SSVR: A Smooth Support Vector Machine for ε-Insensitive Regression," *IEEE Transactions on Knowledge and Data Engineering*, 17(5):678-685, 2005.

[5] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in B. Sch$\acute{}$olkopf, C. Burges, A. Smola, eds, *Advances in Kernel Methods: Support Vector Machines*, MIT Press, Cambridge, MA, December, 1998.

[6] W. Wang, Z. Xu, "A heuristic training for support vector regression," *Neurocomputing*, 61(1):259-275, 2004.

[7] J. Bi, K. P. Bennett, "A geometric approach to support vector regression," *Neurocomputing*, 55(1-2):79-108, 2003.

[8] S. Mukherjee, E. Osuna, F. Girosi, "Nonlinear prediction of chaotic time series using a support vector machine," In *Neural Networks Signal Processing VII, Proc. of the 1997 IEEE Signal Processing Society Workshop*, Amelia Island, FL, USA, Sep 24-26, 1997.

[9] J. T. Jeng, C. C. Chuang, S. F. Su, "Support vector interval regression networks for interval regression analysis," *Fuzzy Sets and Systems*, vol. 138, no. 2, pp. 283-300, Sep. 2003.

[10] J. A. K. Suykens and J. Vandewalle: Least squares support vector machine classifiers, *Neural Processing Letters*, Vol. 9, 3, 1999, pp293-300.

[11] C. Williams and M. Seeger: Using the nyström method to speed up kernel machines. In NIPS, pages 682688, 2001.

[12] A. Rahimi and B. Recht: "Random Features for Large-Scale Kernel Machines",In NIPS, 2017.

[13] T. Yang, Y. Li, M. Mahdavi, R. Jin and Z. Zhou, Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison, In NIPS, pages 476484, 2012.

[14] J. Xu, T. J. J. van den Boom, B. D. Schutter and S. Wang, "Irredundant lattice representations of continunous piecewise affine functions", *Automatica* 70, pp109-120, 2016.

[15] L. Breiman, "Hinging Hyperplanes for regression, classification and function approximation", *IEEE Trans. on Information Theory* 39(3), pp999-1013, 1993.

[16] J. Roll, A. Bemporad and L. Ljung, "Identification of piecewise affine systems via mixed-integer programming", *Automatica* 40(1),pp37-50, 2004.

[17] J. Yu, S. Wang and L. Li, "Incremental Design of Simplex Basis Function Model for Dynamic System Identification," *IEEE Trans. on Neural Networks and Learning Systems*, In Press.

[18] S. Haykin, *Neural Networks and Learning Machines*. Pearson Education Inc, 2009.

[19] J. A. K. Suykens, J. De Brabanter, L. Lukas and J. Vandewalle: Weighted least squares support vector machines: robustness and sparse approximation, *Neurocomputing*, Vol. 48, 1-4, 2002, pp85-105.

[20] J. Yu, S. Wang and L. Li, "Incremental Design of Simplex Basis Function Model for Dynamic System Identification," *IEEE Trans. on Neural Networks and Learning Systems*, In Press.

[21] S. Haykin, *Neural Networks and Learning Machines*. Pearson Education Inc, 2009.

[22] S. Iplikci, "Support vector machine-based generalized predictive control," *International Journal of Robust and Nonlinear Control*, Vol. 16, 843-862, 2006.

[23] S. A. Billings, S. Chen, and R. J. Backhouse, "The identification of linear and non-linear models of a turbocharged automotive diesel engine," *Mechanical Systems and Signal Processing*, vol. 3, no. 2, pp. 123–142, 1989.

[24] S. Chen, X. Hong, C. J. Harris, and P. M. Sharkey, "Sparse modelling using orthogonal forward regression with PRESS statistic and regularization," *IEEE Trans. Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 34, no. 2, pp. 898–911, Apr. 2004.

[25] S. R. Gun, "Support vector machines for classification and regression," Research Report, Dept. Electronics and Computer Science, University of Southampton, U.K, 1998.

[26] G. Lightbody and G. W. Irwin, "Nonlinear Control Structures Based on Embedded Neural System Models," *IEEE Tran. on Neural Networks* Vol. 8, No. 3, pp. 553-567, 1997,

[27] De Moor BLR (ed.). DaISy: Database for the Identification of Systems, Department of Electrical Engineering, ESAT/SISTA, K.U. Leuven, Belgium, URL: http://www.esat.kuleuven.ac.be/sista/daisy/ [Used dataset: Continuous stirred tank reactor, Process Industry Systems, 98-002.]