Sparse Least Squares Low Rank Kernel Machines

Di Xu[1], Manjing Fang[1], Xia Hong[2], and Junbin Gao[1]

[1] Discipline of Business Analytics,The University of Sydney Business School, The University of Sydney, Sydney, NSW 2006, Australia
{dixu3140,mfan9400}@uni.sydney.edu.au
junbin.gao@sydney.edu.au

[2] Department of Computer Science, University of Reading, Reading RG6 6AH, UK
x.hong@reading.ac.uk

Abstract. A general framework of low rank least squares support vector machine (LR-LSSVM) is introduced in this paper. The special structure of controlled model size of the low rank kernel machine brings in remarkable sparsity and hence gigantic breakthrough in computational efficiency. In the meantime, a two-step optimization algorithm with three regimes for gradient descent is proposed. For demonstration purpose, experiments are carried out using a novel robust radial basis function (RRBF), the performances of which mostly dominate.

Keywords: Least Squares Support Vector Machine, Low Rank Kernels, Robust RBF Function, End-to-end modeling

1 Introduction

With the proliferation of big data, one wishes to build sparse models with more efficient algorithms in the settings of practical nonlinear modeling. Kernel machines (KMs) have attracted great attention since support vector machines (SVM) was introduced [1]. In fact, KMs have extended SVM by implementing the linearity in high dimensional feature space under the feature mapping implicitly determined by Mercer kernel function. As one of the most well-known members of the KM family, SVM has good generalization and insensitivity to overfitting [2].

The Performance of SVM greatly depends on kernel types, among which Gaussian RBF kernel normally champions others. Nonetheless, Gaussian RBF has non-negligible deficiencies in nonlinear boundary separation and therefore further optimization of its structure is intensively needed. Amari and Wu [3] modified kernels using a so-called information-geometric and data-dependent method to boost the performance of kernelized SVM. Yu *et al.* [4] enhanced kernels by taking advantages of regularization.

One of the main advantages of standard SVM is its model sparsity determined by the so-called support vectors. However, sparsity is not pre-determined and a computationally demanding quadratic programming problem has to be optimized in order to obtain support vectors [5]. Thus, a massive progress in proposing computationally efficient SVM algorithms has been explored. One of the examples is the introduction of least squares SVM (LSSVM) [6]. Instead of the margin constraints in the standard SVM, equality constraints are introduced in LSSVM. The resulting quadratic programming problem can be solved by a set of linear equations [6]. Nevertheless, equality constraints of LSSVM lead to an evaluation of all possible pairs of data in kernel functions. Therefore, LSSVM loses the sparsness in the standard SVM and is infeasible for large scale data learning. To make LSSVM sparse, researchers considered extending LSSVM to the Ramp loss function and produce sparse models with extra computational complexity, see [7]. This strategy has been extended to more general insensitive loss functions in [8]. Recently, Zhu et al. [9] proposed a way to select effective patterns from training datasets for fast support vector regression learning. However, there is not yet extension in classification. Chen [10] proposed a method for building a sparse kernel model by extending the so-called orthogonal least squares (OLS) algorithm [11] and kernel techniques. Based on the so-called significant vectors, Gao et al. [12] proposed a more straightforward way to learn the significant regressors from training data for the kernel regression modelling.

Almost all the aforementioned modelling methods extract patterns from the entire training dataset. Recently, Hong *et al.* [5] proposed a new LR-LSSVM based on simplex basis functions (LR-LSSVM-SBF), successfully building a sparse and fast modeling algorithm. The model size is no longer determined by the given training data while the key patterns will be learnt straightaway. We further explore this idea and extend it to fit a kind of robust radial basis functions. The main contributions of this paper are summarized as follows,

1. Given that the aforementioned models learn the data patterns under regression settings, this paper focuses on classification settings with a controlled or pre-defined model size;
2. The kernel function proposed in this paper takes the form of composition of basis function components which are adaptive to the training data. This composition form opens the door for a fast closed form solution, avoiding the issue of kernel matrix inversion in the case of large scale datasets;
3. A new criterion is proposed for the final model selection in terms of pattern parameters of location and scale of basis functions; and
4. A two-step optimization algorithm is proposed to simplify the learning procedure.

The rest of this paper is organized as follows. In Section 2, we present a brief background on several related models. Section 3 proposes our robust RBF kernel function and its classification model. Section 4 describes the artificial and real-world datasets and conducts several experiments to demonstrate the performance of the model and algorithm. And Section 5 concludes the paper.

2 Background and Notation

In this section, we start introducing necessary notation for the purpose of presenting our model and algorithm. We mainly consider binary classification problems. For the multi-class classification problems, as usual, the commonly used heuristic approach of "one-vs-all" or "one-vs-one" can be adopted.

Given a training dataset $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{t}) = \{(\boldsymbol{x}_n, t_n)\}_{n=1}^{N}$ where N is the number of data, $\boldsymbol{x}_n \in \mathbb{R}^D$ is the feature vector and $t_n \in \{-1, 1\}$ is the label for the n-th data, respectively.

KM methods have been used as a universal approximator in data modeling. The core idea of the KMs is to implement a linear model in a high dimensional feature space by using a feature mapping $\boldsymbol{\phi}$ defined as [1]

$$\boldsymbol{x} \in \mathbb{R}^D \to \boldsymbol{\phi}(\boldsymbol{x}) \in \mathcal{F},$$

which induces a *Mercer* kernel function in the input space

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle,$$

where $\langle \cdot, \cdot \rangle$ is the inner product on the feature space $\mathcal{F}$.

In general, an affine linear model of KMs is defined as

$$y(\boldsymbol{x}) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{w} \rangle + b, \tag{1}$$

where $b \in \mathbb{R}$ is the bias parameter and $\boldsymbol{w} \in \mathcal{F}$ is the parameter vector of high dimensionality, most likely in infinite dimension. It is infeasible to solve for the parameter vector $\boldsymbol{w}$ directly. Instead, the so-called kernel trick transforms the infinite dimension problem to a finite dimension problem by relating the parameters $\boldsymbol{w}$ to the data as

$$\boldsymbol{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\boldsymbol{x}_n). \tag{2}$$

A learning algorithm will focus on solving for N parameters $\boldsymbol{a} = (a_1, a_2, ..., a_N)^T \in \mathbb{R}^N$ under an appropriate learning criterion.

For the sake of convenience, define

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}) = \left[k(\boldsymbol{x}, \boldsymbol{x}_1)\ k(\boldsymbol{x}, \boldsymbol{x}_2) \cdots k(\boldsymbol{x}, \boldsymbol{x}_N)\right]^T \in \mathbb{R}^N.$$

Then, under (2), model (1) can be expressed in terms of new parameters $\boldsymbol{a}$ as[3]

$$y(\boldsymbol{x}) = \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X})^T (\boldsymbol{a} \circ \boldsymbol{t}) + b. \tag{3}$$

where $\circ$ means the component-wise product of two vectors.

[3]If we are considering a regression problem, there is no need to add t_n in the model (3).

All the KMs algorithms are involved with the so-called kernel matrix, as defined below

$$\boldsymbol{K} = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \in \mathbb{R}^{N \times N}$$

and

$$\boldsymbol{\Omega} = \begin{bmatrix} t_1 t_1 k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & t_1 t_N k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ t_N t_1 k(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & t_N t_N k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \in \mathbb{R}^{N \times N}.$$

Both $\boldsymbol{K}$ and $\boldsymbol{\Omega}$ are symmetric matrices of size $N \times N$.

In the following subsections, LSSVM and sparse least square support vector machine using simplex basis function (LSSVM-SBF) [5] are outlined.

2.1 LSSVM

To reduce the computational complexity of the standard SVM, the least square support vector machine introduces the equality constraints.

The standard LSSVM is formulated in the following programming problem

$$\min_{\boldsymbol{w}, b, \boldsymbol{\eta}} \frac{1}{2}\|\boldsymbol{w}\|_{\mathcal{F}}^2 + \frac{\gamma}{2}\sum_{n=1}^{N} \eta_n^2, \quad \text{s.t. } t_n(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x_n}) \rangle + b) = 1 - \eta_n, \ n = 1, ..., N, \tag{4}$$

where $\gamma > 0$ is a penalty parameter.

With the given equality constraints, the Lagrangian multiplier method produces a kernel model (3) such that the parameters $\boldsymbol{a}$ and b are given by the following set of closed form linear equations

$$\begin{bmatrix} b \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{t}^T \\ \boldsymbol{t} & \Omega + \boldsymbol{I}/\gamma \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \boldsymbol{1} \end{bmatrix} \tag{5}$$

where $\boldsymbol{I}$ is the identity matrix of size $N \times N$. However, the computational hurdle lies in the massive matrix inverse in (5) which has complexity of order $O(N^3)$.

2.2 LSSVM-SBF

Despite of a closed form solution obtained by LSSVM, the model is still limited in computational burden and non-sparsity. Alternatively, the novel LSSVM-SBF [5] overcomes these two issues by introducing the so-called low rank Simplex Basis Function (SBF) kernel, which is defined as follows,

$$\phi_j(\boldsymbol{x}; \boldsymbol{\mu}_j, \boldsymbol{c}_j) = \max\left\{0, 1 - \sum_{i=1}^{D} \mu_{i,j} |x_i - c_{i,j}|\right\}, \tag{6}$$

where $\boldsymbol{c}_j = [c_{1,j}, \cdots, c_{D,j}]^T \in \mathbb{R}^D$ and $\boldsymbol{\mu}_j = [\mu_{1,j}, \cdots, \mu_{D,j}]^T \in \mathbb{R}_+^D$ are the center and shape vectors. The proposed new kernel in [5] is defined as

$$k(\boldsymbol{x}', \boldsymbol{x}'') = \sum_{j=1}^{M} \phi_j(\boldsymbol{x}'; \boldsymbol{\mu}_j, \boldsymbol{c}_j)^T \phi_j(\boldsymbol{x}''; \boldsymbol{\mu}_j, \boldsymbol{c}_j) \tag{7}$$

in which the SBF kernels use only $M \ll N$ basis functions, where M is a predefined model size.

With the low rank kernel structure defined as (7), the closed form solution (5) for $\boldsymbol{a}$ and b can be rewritten as, see [5],

$$\begin{bmatrix} b \\ \boldsymbol{a} \end{bmatrix} = \boldsymbol{q} - \boldsymbol{P}\tilde{\boldsymbol{\Phi}}(\boldsymbol{I} + \tilde{\boldsymbol{\Phi}}^T \boldsymbol{P}\tilde{\boldsymbol{\Phi}})^{-1}\tilde{\boldsymbol{\Phi}}^T \boldsymbol{q}, \tag{8}$$

where

$$\boldsymbol{P} = \frac{1}{N}\begin{bmatrix} -1/\gamma & \boldsymbol{t}^T \\ \boldsymbol{t} & \gamma(N\boldsymbol{I} - \boldsymbol{t}\boldsymbol{t}^T) \end{bmatrix}, \quad \boldsymbol{q} = \frac{1}{N}\begin{bmatrix} \boldsymbol{t}^T\boldsymbol{1} \\ \gamma(N\boldsymbol{I} - \boldsymbol{t}\boldsymbol{t}^T) \end{bmatrix}, \quad \tilde{\boldsymbol{\Phi}} = \begin{bmatrix} 0 & \cdots & 0 \\ \boldsymbol{t} \circ \boldsymbol{\phi}_1 & \cdots & \boldsymbol{t} \circ \boldsymbol{\phi}_M \end{bmatrix}$$

with $\boldsymbol{\phi}_j = [\phi_j(\boldsymbol{x}_1; \boldsymbol{\mu}_j, \boldsymbol{c}_j), \phi_j(\boldsymbol{x}_2; \boldsymbol{\mu}_j, \boldsymbol{c}_j), ..., \phi_j(\boldsymbol{x}_N; \boldsymbol{\mu}_j, \boldsymbol{c}_j)]^T$, i.e., the vector of basis function values at the training inputs.

The new solution (8) only involves the matrix inverse of size $M \times M$, which is superior to (5) where the inverse is of size $N \times N$.

3 The Proposed Model and Its Algorithm

From subsection 2.2, we have found that the special choice of low rank SBF kernel as defined in (6) and (7) brings model efficiency. To extend the idea of using low rank kernel, in this section, we propose a general framework for fast algorithm and validate it with several examples.

We would like to emphasize that our idea of using low rank kernel is inspired by the original low rank kernel approximation such as Nyström approximation [13]. However the standard low rank kernel methods aim to approximate a given kernel function, while our approach involves learning (basis) functions and constructs the kernel with composite structure in order to assist fast algorithms.

3.1 The Low Rank Kernels and Models

Consider M learnable "basis" functions

$$\phi_j(\boldsymbol{x}; \boldsymbol{\lambda}_j) : j = 1, 2, ..., M. \tag{9}$$

with adaptable parameters $\boldsymbol{\lambda}_j$ $(j = 1, 2, ..., M)$. In the case of SBF in (6), we have in total $2D$ parameters

$$\boldsymbol{\lambda}_j = \{\boldsymbol{\mu}_j, \boldsymbol{c}_j\}.$$

As another example, we will consider the so-called robust RBF

$$\phi_j(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{c}_j)=\exp\left\{-\sum_{i=1}^{D}\mu_{i,j}|x_i-c_{i,j}|\right\}. \tag{10}$$

Similar to the SBF, while $c_{i,j}$ determines the location of $\phi_j(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{c}_j)$ in the ith dimensional direction, $\mu_{i,j}$ restricts the sharpness of $\phi_j(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{c}_j)$ in the ith dimension. In fact, the SBF (6) can be regarded as the first order approximation of the robust RBF in terms of $\exp\{-t\}=1-t+\frac{1}{2!}t^2+\cdots$. We expect the robust RBF will have better modeling capability.

More generally, each learnable basis function $\phi_j(\boldsymbol{x};\boldsymbol{\lambda}_j)$ can be a deep neural network. We will leave this for further study.

Given a set of learnable basis functions (9), define a finite dimensional feature mapping

$$\boldsymbol{\phi}_r:\boldsymbol{x}\in\mathbb{R}^D\rightarrow\boldsymbol{\phi}_r(\boldsymbol{x})=\begin{bmatrix}\phi_1(\boldsymbol{x};\boldsymbol{\lambda}_1)\\ \vdots\\ \phi_M(\boldsymbol{x};\boldsymbol{\lambda}_M)]\end{bmatrix}\in\mathcal{F}.$$

This feature mapping naturally induces the following learnable low rank kernel

$$k(\boldsymbol{x}',\boldsymbol{x}'')=\sum_{j=1}^{M}\phi_j(\boldsymbol{x}';\boldsymbol{\lambda}_j)^T\phi_j(\boldsymbol{x}'';\boldsymbol{\lambda}_j). \tag{11}$$

Consider the "linear" model $y(\boldsymbol{x})=\langle\boldsymbol{w},\boldsymbol{\phi}_r(\boldsymbol{x})\rangle+b$ and define the following low rank LSSVM (LR-LSSVM)

$$\min_{\boldsymbol{w},b,\boldsymbol{\eta}}\frac{1}{2}\|\boldsymbol{w}\|^2_{\mathcal{F}}+\frac{\gamma}{2}\sum_{n=1}^{N}\eta_n^2,\quad\text{s.t. } t_n(\langle\boldsymbol{w},\boldsymbol{\phi}_r(\boldsymbol{x_n})\rangle+b)=1-\eta_n,\ n=1,...,N. \tag{12}$$

The LR-LSSVM problem takes the same form as the standard LSSVM (4), however our low rank kernel carries composition structure and is learnable with adaptable parameters. In the following subsections, we propose a two-steps alternative algorithm procedure to solve the LR-LSSVM.

3.2 Solving LR-LSSVM with Fixed Feature Mappings

When all the feature mappings $\phi_j(j=1,2,...,M)$ are fixed, problem (12) gives back to the standard LSSVM. Denote $\boldsymbol{\eta}=[\eta_1,\eta_2,...,\eta_N]^T$ and consider the Lagrangian function

$$L(\boldsymbol{w},b,\boldsymbol{\eta};\boldsymbol{a})=\frac{\gamma}{2}\|\boldsymbol{\eta}\|^2+\frac{1}{2}\|\boldsymbol{w}\|^2-\sum_{n=1}^{N}a_n\{t_n(\quad\langle\boldsymbol{w},\boldsymbol{\phi}_r(\boldsymbol{x_n})\rangle+b)-1+\eta_n\},$$

where $\boldsymbol{a}=[a_1,a_2,...,a_N]^T$ are Lagrange multipliers for all the equality constraints. We now optimize out $\boldsymbol{w}$, b and $\boldsymbol{\eta}$ to give

$$\frac{\partial L}{\partial\boldsymbol{w}}=\boldsymbol{0}\rightarrow\boldsymbol{w}=\boldsymbol{\Phi}^T(\boldsymbol{a}\circ\boldsymbol{t}),\quad\frac{\partial L}{\partial b}=0\rightarrow\boldsymbol{t}^T\circ\boldsymbol{a}=0,\quad\frac{\partial L}{\partial\boldsymbol{\eta}}=\boldsymbol{0}\rightarrow\boldsymbol{a}=\gamma\boldsymbol{\eta} \tag{13}$$

where

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_1(\boldsymbol{x}_1;\lambda_1) & \cdots & \phi_M(\boldsymbol{x}_1;\lambda_M) \\ \vdots & \cdots & \vdots \\ \phi_1(\boldsymbol{x}_N;\lambda_1) & \cdots & \phi_M(\boldsymbol{x}_N;\lambda_M) \end{bmatrix} \in \mathbb{R}^{N\times M}. \tag{14}$$

Furthermore, setting the partial derivative with respect to each Lagrange multiplier and after a long algebraic manipulation, the solution for the dual problem is given by

$$\begin{bmatrix} b \\ \boldsymbol{a} \end{bmatrix} = \left\{ \begin{bmatrix} 0 & \boldsymbol{t}^T \\ \boldsymbol{t} & \boldsymbol{I}/\gamma \end{bmatrix} + \widetilde{\boldsymbol{\Phi}}\widetilde{\boldsymbol{\Phi}}^T \right\}^{-1} \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}. \tag{15}$$

where $\widetilde{\boldsymbol{\Phi}}$ the $(N+1)\times M$ is the matrix with one row of all zeros on the top of matrix diag$(\boldsymbol{t})\boldsymbol{\Phi}$.

Applying the matrix inversion formula to (15) results in the exactly same solution as (8). Once $\boldsymbol{a}$ and b are worked out, the final model can be written as

$$y(\boldsymbol{x}) = \sum_{j=1}^{M} \theta_j \phi_j(\boldsymbol{x};\boldsymbol{\lambda}_j) + b, \tag{16}$$

where

$$\theta_j = \sum_{n=1}^{N} a_n t_n \phi_j(\boldsymbol{x}_n;\boldsymbol{\lambda}_j).$$

Thus the final model is expressed in terms of sparse form of size M.

3.3 Training Learnable Low Rank Kernels

Given $\boldsymbol{a}, b$ which are solved by the closed-form solution in the first step, we estimate the kernel parameters $\boldsymbol{\lambda}_j$ $(j = 1, \dots, M)$ using a gradient descent algorithm. The algorithm seeks to maximize the magnitude of model outputs, which leads to overall further distance from the model outputs to the existing decision boundary. Taking the robust RBF functions (10) as an example, this objective function can be expressed as

$$J^{(j)}(\boldsymbol{c}_j, \boldsymbol{\mu}_j) = \sum_{n=1}^{N} |y(\boldsymbol{x}_n)|. \tag{17}$$

Another objective function is

$$J^{(j)}(\boldsymbol{c}_j, \boldsymbol{\mu}_j) = \sum_{n=1}^{N} t_n y(\boldsymbol{x}_n), \tag{18}$$

which gives similar results as (17).

Denote $\text{sign}(\boldsymbol{y}) = [\text{sign}(y(\boldsymbol{x}_1)), \ldots, \text{sign}(y(\boldsymbol{x}_N))]^T$. Given the objective function above, we have

$$\begin{cases} \dfrac{\partial J^{(j)}}{\partial \mu_{i,j}} = [\text{sign}(\boldsymbol{y})]^T \dfrac{\partial \boldsymbol{K}}{\partial \mu_{i,j}} (\boldsymbol{a} \circ \boldsymbol{t}) \quad i = 1, \ldots, D \\ \dfrac{\partial J^{(j)}}{\partial c_{i,j}} = [\text{sign}(\boldsymbol{y})]^T \dfrac{\partial \boldsymbol{K}}{\partial c_{i,j}} (\boldsymbol{a} \circ \boldsymbol{t}) \quad i = 1, \ldots, D \end{cases} \tag{19}$$

in which

$$\frac{\partial \boldsymbol{K}}{\partial \mu_{i,j}} = (\frac{\partial \boldsymbol{\phi}_j}{\partial \mu_{i,j}}) \boldsymbol{\phi}_j^T + \boldsymbol{\phi}_j (\frac{\partial \boldsymbol{\phi}_j}{\partial \mu_{i,j}})^T; \quad \frac{\partial \boldsymbol{K}}{\partial c_{i,j}} = (\frac{\partial \boldsymbol{\phi}_j}{\partial c_{i,j}}) \boldsymbol{\phi}_j^T + \boldsymbol{\phi}_j (\frac{\partial \boldsymbol{\phi}_j}{\partial c_{i,j}})^T \tag{20}$$

where

$$\frac{\partial \boldsymbol{\phi}_j}{\partial \mu_{i,j}} = \left[\frac{\partial \phi_j(\boldsymbol{x}_1)}{\partial \mu_{i,j}}, ..., \frac{\partial \phi_j(\boldsymbol{x}_N)}{\partial \mu_{i,j}} \right]^T; \quad \frac{\partial \boldsymbol{\phi}_j}{\partial c_{i,j}} = \left[\frac{\partial \phi_j(\boldsymbol{x}_1)}{\partial c_{i,j}}, ..., \frac{\partial \phi_j(\boldsymbol{x}_N)}{\partial c_{i,j}} \right]^T \tag{21}$$

which are calculated by, for $i = 1, ..., D$,

$$\frac{\partial \phi_j(\boldsymbol{x}_n)}{\partial \mu_{i,j}} = -|x_{i,n} - c_{i,j}| \phi_j(\boldsymbol{x}_n; \boldsymbol{\mu}_j, \boldsymbol{c}_j), \tag{22}$$

$$\frac{\partial \phi_j(\boldsymbol{x}_n)}{\partial c_{i,j}} = \mu_{i,j} \text{sign}(x_{i,n} - c_{i,j}) \phi_j(\boldsymbol{x}_n; \boldsymbol{\mu}_j, \boldsymbol{c}_j). \tag{23}$$

where $\phi_j(\boldsymbol{x}; \boldsymbol{\mu}_j, \boldsymbol{c}_j)$ is defined in (10).

Meanwhile, we should also consider the positivity constraints for the shape parameters vector $\boldsymbol{\mu}_j$ and thus, we have the following constrained normalized gradient descent algorithm, which is, for $i = 1, \ldots, D$,

$$\begin{cases} c_{i,j} = & c_{i,j} + \eta \cdot \dfrac{\partial J^{(i)}}{\partial c_{i,j}} / \left\| \dfrac{\partial J^{(i)}}{\partial \boldsymbol{c}_j} \right\| \\ \tilde{\mu}_{i,j} = & \mu_{i,j} + \eta \cdot \dfrac{\partial J^{(i)}}{\partial \mu_{i,j}} / \left\| \dfrac{\partial J^{(i)}}{\partial \boldsymbol{\mu}_j} \right\| \\ \mu_{i,j} = & \max(0, \tilde{\mu}_{i,j}) \end{cases} \tag{24}$$

where $\eta > 0$ is a preset learning rate. By applying (19) to (24) to all M Robust RBF units while keeping $b, \boldsymbol{a}$ to their current values and other RBF units constant, we manage to update all RBF kernels.

3.4 Initialization of Robust Radial Basis Functions

As is shown in (16), the model requires a preset kernel model size M and a set of initial kernel parameters $\boldsymbol{\lambda}_j$, $j = 1, \ldots, M$. In the case of robust RBFs, both $\boldsymbol{c}_j$ and $\boldsymbol{\mu}_j$ need to be initialized. The initialization of the center vector $\boldsymbol{c}_j$ can be obtained using a clustering algorithm. We propose a k-medoids algorithm here to solve for the Robust RBF centers since it is more robust to unbalanced distribution of data. It seeks to divide the data points into M subsets and iteratively adjust the centers $\boldsymbol{c}_j$ of each subset S_j until reaching convergence while minimizing the clustering objective objection given by

$$J = \sum_{j=1}^{M} \sum_{\boldsymbol{x}_n \in S_j} \|\boldsymbol{x}_n - \boldsymbol{c}_j\| \tag{25}$$

where the centers $\boldsymbol{c}_j$ of each subset are the members of that subset. As for the initial values of the shaping parameters $\boldsymbol{\mu}_j$, we preset $\mu_{i,j}$ as a predetermined constant for all basis functions, e.g., 1s.

3.5 The Overall Algorithm and Its Complexity

Algorithm 1[4] summarizes the overall procedure of LR-LSSVM using the example of robust RBF kernel. The algorithm starts with the k-medoids clustering algorithm for initialization of the robust RBF centres in Section 3.2, then the fast LSSVM solution is achieved and the gradient descent algorithm in Section 3.3 or 3.6 are alternatively applied for a predefined number of iterations. A simple complexity analysis indicates that the overall computational complexity is $O(M^2N)$ which is dominated by the gradient descent algorithm for training learnable basis functions, scaled by the iteration number. Many examples in Section 4 have shown that a minor size M gives competitive model prediction performance. In this sense, the newly proposed algorithm has a complexity of $O(N)$. The lower complexity benefits from the special structure of low rank kernel functions. It should be pointed out again that the proposed framework contains the SBF model in [5] as a special case, that the framework can be applied for more generic extension, for example using deep neural networks for learning kernel functions.

3.6 The Differentiable Objective Functions

The objective defined in (17) is non-differentiable. For the purpose of maximizing the magnitude of model outputs, we propose the following squared objective which is differentiable, for $j = 1, 2, ..., M$,

$$F^{(j)}(\boldsymbol{c}_j, \boldsymbol{\mu}_j) = \sum_{n=1}^{N} y(\boldsymbol{x}_n)^2. \tag{26}$$

It is not hard to prove that

$$\frac{\partial F^{(j)}}{\partial \nu} = (\boldsymbol{a} \circ \boldsymbol{t})^T \left(\mathbf{K} \frac{\partial \mathbf{K}}{\partial \nu} + \frac{\partial \mathbf{K}}{\partial \nu} \mathbf{K} \right) (\boldsymbol{a} \circ \boldsymbol{t}) + 2b(\boldsymbol{a} \circ \boldsymbol{t})^T \frac{\partial \mathbf{K}}{\partial \nu} \mathbf{1} \tag{27}$$

where $\frac{\partial \mathbf{K}}{\partial \nu}$ is the matrix given by (19) and (20), and ν means either $\mu_{i,j}$ or $c_{i,j}$.

[4]The algorithm can be easily adopted to any learnable kernels.

Algorithm 1 The Proposed LR-LSSVM Algorithm with robust RBF kernel

```
Input: Dataset $\mathcal{D}$. Model size $M$. Regularization parameter $\gamma$. Initial shape parameter $\boldsymbol{\mu}_j$. Iteration number $T$.
Output: The obtained model parameters $\boldsymbol{a}$, $b$, $\boldsymbol{\lambda}_j = (\boldsymbol{c}_j, \boldsymbol{\mu}_j)$ for $j = 1, 2, ..., M$.
1: Apply the k-medoids clustering algorithm to initialize $\boldsymbol{c}_j$ $(j = 1, 2, ..., M)$. Set all $\mu_{i,j}$ to a constant $\mu$.
2: for $t = 1, 2, ..., T$ do
3:   Form $\boldsymbol{\Phi}$ from the dataset $\mathcal{D}$ and the current kernel parameters $\boldsymbol{\lambda}_j = (\boldsymbol{c}_j, \boldsymbol{\mu}_j)$ for $j = 1, 2, ..., M$;
4:   Construct $\widetilde{\boldsymbol{\Phi}}$ by adding one row of $\mathbf{0}$ on the top of matrix diag$(\boldsymbol{t})\boldsymbol{\Phi}$;
5:   Update $b$ and $\boldsymbol{a}$ according to the closed form solution (8);
6:   for $j = 1, 2, ..., M$ do
7:     Apply (19) to (24) to adjust $\boldsymbol{\lambda}_j = (\boldsymbol{c}_j, \boldsymbol{\mu}_j)$
8:   end for
9: end for
```

Table 1. The misclassification rate (%) on different datasets.

Models	Synthetic		Titanic		Diabetes		German Credit	
	Mis Rate	Size	Mis Rate	Size	Mis Rate	Size	Mis Rate	Size
Adaboost with RBF	N/A	N/A	22.6 ± 1.2	4	26.5 ± 1.9	15	27.5 ± 2.5	8
QPReg-AdaBoost	N/A	N/A	22.7 ± 1.1	4	25.4 ± 2.2	15	25.3 ± 2.1	8
SVM with RBF kernel	N/A	N/A	22.4 ± 1.0	N/A	23.5 ± 1.7	N/A	23.6 ± 2.1	N/A
LSSVM-Gaussian (σ=1)	9.2	250	N/A	N/A	N/A	N/A	N/A	N/A
LSSVM-SBF	8.3	4	22.5 ± 0.8	2	23.5 ± 1.7	5	24.9 ± 1.9	3
M1	8.0	3	22.3 ± 0.8	2	23.8 ± 1.7	5	25.6 ± 2.3	2
M2	8.3	3	22.6 ± 1.5	3	23.5 ± 2.0	4	24.7 ± 1.9	2
M3	8.0	3	22.4 ± 0.8	2	24.7 ± 2.0	5	25.6 ± 2.4	2

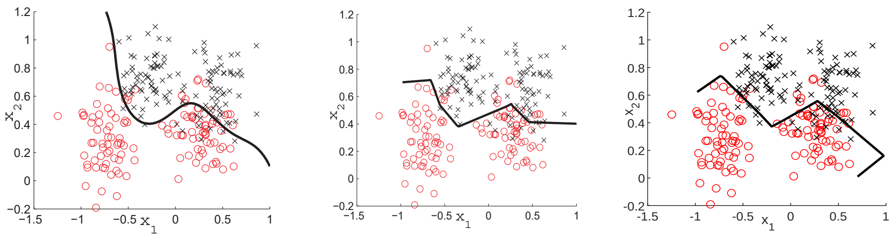
4 Experimental Studies

In this section, validations are conducted on synthetic, Titanic, Diabetes, and German Credit datasets. The descriptions of data and model parameters are in Table 2, where M1, M2 and M3 are the proposed models by using absolute (17), squared (26) and target (18) objectives, respectively. D is the input dimension of dataset. No. Train and No. Test are the number of realizations in each group, while 100 means that the group is divided into 100 subsets. μ, γ, η, and T are steepness, penalization, learning rate and times of iteration. Outcomes of the proposed and benchmark models are shown in Table 1 with $\sigma = 1.0$ being the optimal value for LSSVM-Gaussian among 0.5-3, step 0.5 and the first five benchmarks in the table citing from [5].

Overall, LR-LSSVM mostly stands out in synthetic, Titanic and Diabetes, while remains comparable in German Credit. Within LR-LSSVM, RRBF using square objective is fairly unstable with good performance in Diabetes whereas less ideal in the rest. Thus, inference that square-objectived LR-LSSVM-RRBF suits more for high-dimensional cases would be made, still more validations needed for solid conclusion. The size column of Table 1 represents model spar-

Table 2. The parameter setting of different datasets.

Parameters	Synthetic			Titanic			Diabetes			German Credit		
	M1	M2	M3	M1	M2	M3	M1	M2	M3	M1	M2	M3
D		2			3			8			20	
No. Train		250			100×150			100×468			100×700	
No. Test		1000			100×2051			100×300			100×300	
μ		0.2		0.03	0.001	0.001	0.01	0.001	0.0001		0.005	
γ		100		50000	500000	50000		50000			200000	
η	0.0008	0.0005	0.0008	0.0005	0.0001	0.0001		0.001			0.003	
T	150	20	150		100			100			100	



(a) Gaussian SVM ($\sigma = 1$) (b) LSSVM-SBF (c) proposed model 1

Fig. 1. The decision boundaries for synthetic dataset

sity. Superb sparseness of LR-LSSVM can be easily observed within this column and thus lead to notable progress of computational speed. Figure 1 depicts the decision boundaries of some typical models in Synthetic dataset, in which the decision boundary of Gaussian LSSVM is smooth and round whereas that of LR-LSSVM local-linearly wiggles. Plus, though both of locally linear traits, the lineshape of decision boundary of LR-LSSVM-RRBF significantly deviates from that of LR-LSSVM-SBF.

5 Conclusions

In this paper, we present a general framework of fast LR-LSSVM and take a RRBF kernel as demonstration. After kernel parameter initialization with k-medriods clustering, training alternates between closed form solution of $\boldsymbol{a}, b$ and gradient descent of $\boldsymbol{c}, \boldsymbol{\mu}$. Three objective functions are suggested for gradient descent, where solely the second one is differentiable. In the end, experiments are conducted on synthetic and real-world cases.

References

1. B. Schölkopf and A. J. Smola, *Learning with Kernels.* MIT Press, 2002.

2. F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for EEG-based brain-computer interfaces," *J. of Neural Engineering*, vol. 4, no. 2, pp. R1–13, 2007.
3. S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.
4. K. Yu, W. Xu, and Y. Gong, "Deep learning with kernel regularization for visual recognition," in *NIPS*, vol. 21, 2009, pp. 1889–1896.
5. X. Hong, H. Wei, and J. Gao, "Sparse least squares support vector machine using simplex basis function," *IEEE Transactions on Cybernetics*, vol. XX, pp. Submission No. CYB–E–2018–06–1246, 2018.
6. J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, pp. 293–300, 1999.
7. D. Liu, Y. Shi, Y. Tian, and X. Huang, "Ramp loss least squares support vector machine," *J. of Computational Science*, vol. 14, pp. 61–68, 2016.
8. Y. Ye, J. Gao, Y. Shao, C. Li, and Y. Jin, "Robust support vector regression with general quadratic non-convex ϵ-insensitive loss," *ACM Trans. on Knowledge Discovery from Data*, vol. XX, p. submitted, 2019.
9. F. Zhu, J. Gao, C. Xu, J. Yang, and D. Tao, "On selecting effective patterns for fast support vector regression training," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3610–3622, 2018.
10. S. Chen, "Local regularization assisted orthogonal least squares regression," *NeuroComputing*, vol. 69, pp. 559–585, 2006.
11. S. Chen, C. Cowan, and P. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.
12. J. Gao, D. Shi, and X. Liu, "Critical vector learning to construct sparse kernel regression modelling," *Neural Networks*, vol. 20, no. 7, pp. 791–798, 2007.
13. C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc of NIPS*, 2001, pp. 682–688.