Pattern Recognition 42 (2009) 837 – 842

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Recursive reduced least squares support vector regression

Yongping Zhao*, Jianguo Sun

Department of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, PR China

ARTICLE INFO

Article history:
Received 16 February 2008
Received in revised form 15 July 2008
Accepted 19 September 2008

Keywords:
Least squares support vector regression
Reduced technique
Iterative strategy
Parsimoniousness
Classification

ABSTRACT

Combining reduced technique with iterative strategy, we propose a recursive reduced least squares support vector regression. The proposed algorithm chooses the data which make more contribution to target function as support vectors, and it considers all the constraints generated by the whole training set. Thus it acquires less support vectors, the number of which can be arbitrarily predefined, to construct the model with the similar generalization performance. In comparison with other methods, our algorithm also gains excellent parsimoniousness. Numerical experiments on benchmark data sets confirm the validity and feasibility of the presented algorithm. In addition, this algorithm can be extended to classification.

1. Introduction

Support vector machine (SVM) [1–3] has been gaining more and more in popularity and is regarded as one state-of-the-art tool for solving pattern recognition and regression estimation problem and has been demonstrated to be valuable for many real-world applications. However, the training computational burden is very expensive, say, $O(N^3)$, where N is the total size of training patterns. In the past few years, many algorithms speeding up SVM have been proposed, taking Chunking [4], SMO [5], ISMO [6], SVMLight [7], SVMTorch [8], and LIBSVM [9] for example. The time complexity of these algorithms is $T \cdot O(Nq+q)$, where T is the number of iterations and q is the scale of working set. There exists one common drawback about these algorithms which is that they only accelerate the training speed without speedup the predicted speed which is proportional to the number of support vectors. SVM is a sparse machine learning algorithm in theory, but the parsimoniousness of the solution is not as good as what we expect [10]. In order to besiege the dilemma, Lee and Mangasarian [11] proposed to restrict the number of support vectors by solving the reduced support vector machine (RSVM). The main characteristic of this method is randomly selecting a subset of training data, as little as 1–10% of the large training patterns, considered as candidates of support vectors. RSVM is different from traditional directly solving smaller SVM problems with subset of training data because the N constraints in the primal are still kept during the optimization process, which maybe enhancing the generalization performance. However, Lin and Lin [12] showed that the testing accuracy of RSVM is a little lower than normal SVM. Recently, Lee and Huang [13] gave out the statistical theory for RSVM.

Using equality constraints instead of inequality, a variable SVM, called least squares support vector machine (LSSVM) [14,15], was proposed. Extensive empirical comparisons show that LSSVM obtains good performance on various classification problems [16]. However, there exists one obvious limitation. In comparison with normal SVM, LSSVM is not parsimonious, which blocks its predicted speed, i.e., slower than SVM. In recent years, considerable attentions have been paid to the aforementioned limitation. Suykens et al. [17] and Chu et al. [18], respectively, presented a conjugate gradient algorithm and the improved form to speed up the training speed, and Keerthi and Shevade [19] successfully applied sequential minimal optimization (SMO) algorithm to LSSVM. However, the resultant models are not parsimonious using these algorithms. That is to say, the limitation still exists. As for obtaining parsimonious model, several tricks are proposed. Suykens et al. [20] proposed a simple approach to introduce the parsimonious model. de Kruif et al. [21] presented a pruning mechanism to omit the vectors which bore the least errors. Based on SMO, Zeng and Chen [22] gave another pruning method. Recently, Kuh et al. [23] speeded up the method proposed by de Kruif et al., and Jiao et al. [24] introduced a fast approach to establish the parsimonious LSSVM. However, the collective defect of these tricks is without considering the constraints brought by other vectors, so called non-support vectors, during building the parsimonious model of LSSVM. This more or less affects the parsimoniousness of LSSVM, because there are not non-support vectors for LSSVM, so called non-support vectors generated by some compulsive strategies. If we completely discard the constraints generated by non-support vectors, it is not reasonable in a way. So we can apply the reduced technique,

* Corresponding author.
E-mail address: zhaoyongping_007@163.com (Y. Zhao).

0031-3203/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.09.028

which considers the whole N constraints during the optimization process, to LSSVM [25]. From Ref. [12], we know that if we randomly select a subset from training patterns as training data set, it will degrade the test accuracy. Combining the iterative strategy in Ref. [24] with the reduced technique, a recursive reduced least squares support vector regression (RR-LSSVR) is proposed. RR-LSSVR gains advantage over the common parsimonious tricks, since it is involved in the whole constraints generated by all training patterns in the modeling process. Meantime, in comparison with randomly selecting subset to build the reduced least squares support vector regression, RR-LSSVR needs less scale subset, which will shorten predicted time and strengthen parsimoniousness.

This paper is organized as follows. In Section 2 we will briefly introduce the reduced least squares support vector regression. Section 3 describes the proposed RR-LSSVR, and we provide the proof of RR-LSSVR's convergence. In the following section, benchmark data sets confirm the validity and feasibility of our algorithm and we compare it with other methods. The discussion and conclusion follow in Section 5.

2. Reduced least squares support vector regression

In this section, we will concisely introduce the reduced least squares support vector regression. Considering a regression problem with training patterns $\{(\boldsymbol{x}_i, d_i)\}_{i=1}^N$ where $\boldsymbol{x}_i$ is the input pattern and d_i is the corresponding target. Using kernel function, we can obtain a nonlinear predictor called least squares support vector regression (LSSVR) through solving the following optimization problem:

$$\min_{\boldsymbol{w},\boldsymbol{e}} \left\{\frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \frac{C}{2}\sum_{i=1}^N e_i^2\right\}$$
$$\text{s.t.} \quad d_i = \boldsymbol{w}^{\mathrm{T}}\varphi(\boldsymbol{x}_i) + b + e_i, \quad i = 1, \dots, N \tag{1}$$

where $\boldsymbol{w}$ represents the model complexity, $\boldsymbol{e} = [e_1, \dots, e_N]^{\mathrm{T}}$, $C \in \Re^+$ is the regulator, $\varphi(\cdot)$ is a nonlinear mapping which maps the input data into a high-dimensional feature space whose dimension can be infinite. One defines the Lagrangian

$$L(\boldsymbol{w}, b, \boldsymbol{e}; \bar{\boldsymbol{\alpha}}) = \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \frac{C}{2}\sum_{i=1}^N e_i^2 + \sum_{i=1}^N \bar{\alpha}_i(d_i - \boldsymbol{w}^{\mathrm{T}}\varphi(\boldsymbol{x}_i) - b - e_i) \tag{2}$$

where $\bar{\alpha}_i$'s are the Lagrange multipliers, which can be either positive or negative.

The conditions for optimality

$$\begin{cases} \frac{\partial L}{\partial \boldsymbol{w}} = 0 \to \boldsymbol{w} = \sum_{i=1}^N \bar{\alpha}_i \varphi(x_i) \\ \frac{\partial L}{\partial b} = 0 \to \sum_{i=1}^N \bar{\alpha}_i = 0 \\ \frac{\partial L}{\partial e_i} = 0 \to \bar{\alpha}_i = Ce_i \\ \frac{\partial L}{\partial \alpha_i} = 0 \to \boldsymbol{w}^{\mathrm{T}}\varphi(\boldsymbol{x}_i) + b + e_i - d_i = 0 \end{cases} \tag{3}$$

Eliminating the variables $\boldsymbol{w}$ and e_i's, we can get the following linear equations:

$$\begin{bmatrix} 0 & \vec{\mathbf{1}}^{\mathrm{T}} \\ \vec{\mathbf{1}} & \bar{K} \end{bmatrix} \begin{bmatrix} b \\ \bar{\boldsymbol{\alpha}} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{d} \end{bmatrix} \tag{4}$$

where $\vec{\mathbf{1}} = [1_1, \dots, 1_N]^{\mathrm{T}}$, $\bar{\boldsymbol{\alpha}} = [\bar{\alpha}_1, \dots, \bar{\alpha}_N]^{\mathrm{T}}$, $\boldsymbol{d} = [d_1, \dots, d_N]^{\mathrm{T}}$, $\bar{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^{\mathrm{T}}\varphi(\boldsymbol{x}_j) + \delta_{ij}/C$ with

$$\delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases} \quad i, j = 1, \dots, N$$

$k(\cdot,\cdot)$ is the kernel function, which can be expressed as the inner product of two vectors in some feature space. Among all the kernel functions, Gaussian kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\gamma^2)$ is the most popular choice. For a new pattern $\boldsymbol{x}$, we can predict its target by

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\varphi(\boldsymbol{x}) + b = \sum_{i=1}^N \bar{\alpha}_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{5}$$

where $\bar{\boldsymbol{\alpha}}$ and b are the solutions of Eq. (4).

From Eq. (5), we understand that LSSVR is not parsimonious. After letting $\boldsymbol{w} = \sum_{i \in S}\alpha_i k(\boldsymbol{x}_i, \cdot)$ and substituting it into Eq. (1) where the subset $\{(\boldsymbol{x}_i, d_i)\}_{i \in S} \subset \{(\boldsymbol{x}_i, d_i)\}_{i=1}^N$, we get the corresponding form as follows:

$$\min \left\{ L(b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}K\boldsymbol{\alpha} + \frac{C}{2}\sum_{i=1}^N \left(d_i - \sum_{j \in S}\alpha_j \varphi(\boldsymbol{x}_j)^{\mathrm{T}}\varphi(\boldsymbol{x}_i) - b\right)^2 \right\} \tag{6}$$

where $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i, j \in S$. Let $\partial L/\partial b = 0$ and $\partial L/\partial \alpha_i = 0$, so the following linear equations are gained:

$$(R + ZZ^{\mathrm{T}})\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = Z\boldsymbol{d} \tag{7}$$

where

$$R = \begin{bmatrix} 0 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & K/C \end{bmatrix}, \quad Z = \begin{bmatrix} \vec{\mathbf{1}}^{\mathrm{T}} \\ \hat{K} \end{bmatrix}$$

with $\hat{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i \in S$, $j = 1, \dots, N$. In Eq. (7), if $R + ZZ^{\mathrm{T}}$ is singular, a small change $R + ZZ^{\mathrm{T}} + 10^{-8}I$ will find the solution. Therefore we find the reduced predictor for a new sample $\boldsymbol{x}$:

$$f(\boldsymbol{x}) = \sum_{i \in S}\alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{8}$$

This is the reduced least squares support vector regression.

3. Recursive reduced least squares support vector regression

From Eq. (8), we know that the subset $\{(\boldsymbol{x}_i, d_i)\}_{i \in S}$ is not determined. If we select the subset randomly, it leads to either reduced LSSVR is not parsimonious enough or the test accuracy is degraded. So it is especially important to select the subset. The patterns selected should represent the main characteristics of the whole training data set, i.e., they take crucial roles in constructing the reduced LSSVR model. Under the illumination of Ref. [24], we pick up the patterns which make more contribution to optimization target (6) to form the subset. Reformulating (6), we have

$$\min \left\{ L = [b \ \ \boldsymbol{\alpha}^{\mathrm{T}}]\left(\begin{bmatrix} 0 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & K/C \end{bmatrix} + \begin{bmatrix} \vec{\mathbf{1}}^{\mathrm{T}} \\ \hat{K} \end{bmatrix}[\vec{1} \ \ \hat{K}^{\mathrm{T}}]\right)\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix}^{\mathrm{T}} - 2\left(\begin{bmatrix} \vec{\mathbf{1}}^{\mathrm{T}} \\ \hat{K} \end{bmatrix}\boldsymbol{d}\right)^{\mathrm{T}}\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} \right\} \tag{9}$$

3.1. Iterative computation of the kernel matrix inversion

We unfold (7) as follows firstly:

$$\left(\begin{bmatrix} 0 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & K/C \end{bmatrix} + \begin{bmatrix} \vec{\mathbf{1}}^{\mathrm{T}} \\ \hat{K} \end{bmatrix}[\vec{\mathbf{1}} \ \ \hat{K}^{\mathrm{T}}]\right)\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{1}}^{\mathrm{T}} \\ \hat{K} \end{bmatrix}\boldsymbol{d} \tag{10}$$

i.e.,

$$\left(\begin{bmatrix} 0 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & K/C \end{bmatrix} + \begin{bmatrix} N & \vec{\mathbf{1}}^{\mathrm{T}}\hat{K}^{\mathrm{T}} \\ \hat{K}\vec{\mathbf{1}} & \hat{K}\hat{K}^{\mathrm{T}} \end{bmatrix}\right)\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{1}}^{\mathrm{T}} \\ \hat{K} \end{bmatrix}\boldsymbol{d} \tag{11}$$