Expert Systems with Applications 38 (2011) 1611–1618

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Multikernel semiparametric linear programming support vector regression

Yong-Ping Zhao [a,*], Jian-Guo Sun [b]

[a] ZNDY of Ministerial Key Laboratory, Nanjing University of Science & Technology, Nanjing 210094, China
[b] Department of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

ARTICLE INFO

Keywords:
Linear programming support vector regression
Semiparametric technique
Multikernel trick
Classification

ABSTRACT

In many real life realms, many unknown systems own different data trends in different regions, i.e., some parts are steep variations while other parts are smooth variations. If we utilize the conventional kernel learning algorithm, viz. the single kernel linear programming support vector regression, to identify these systems, the identification results are usually not very good. Hence, we exploit the nonlinear mappings induced from the kernel functions as the admissible functions to construct a novel multikernel semiparametric predictor, called as MSLP-SVR, to improve the regression effectiveness. The experimental results on the synthetic and the real-world data sets corroborate the efficacy and validity of our proposed MSLP-SVR. Meantime, compared with other multikernel linear programming support vector algorithm, ours also takes advantages. In addition, although the MSLP-SVR is proposed in the regression domain, it can also be extended to classification problems.

1. Motivation

During the past decade, known as an excellent kernel modeling technique, support vector machine (SVM) (Burges, 1998; Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2002; Vapnik, 1995) has been becoming of popularity in the realm of machine learning and has been referred to as the state-of-the-art strategy for classification and regression applications like image segmentation (Chen & Wang, 2005), time series prediction (Lau & Wu, 2008), and face recognition (Guo, Li, & Chan, 2001). Its basis is the so-called structural risk minimization principle which consists of two parts, one of which is empirical risk also owned in the cost function of the artificial neural networks (ANNs) such as the well-known back propagation (BP) networks, and another is the confidence interval which is capable of controlling the model complexity through the Vapnik–Chervonenkis (VC) dimension. Initially, the SVM is constructed to solve the binary classification problem using a decision hyperplane with as soon as large margin, which is expected to obtain good generalization performance. As a consequence, a small subset of the training samples, termed as support vectors, is chosen to support the optimal hyperplane. From the inception of the SVM, it is broadly extended to various fields like density estimation, regression and linear operator equation. When it is exploited to cope with the regression estimation and function approximation, a variable, viz. support vector regression (SVR), is born. Like the hinge loss function in SVM, an innovative loss function, namely ε-insensitive loss function, is proposed by Vapnik (1995) to construct a tolerance band for the purpose of realizing the sparse solution for the SVR. Although it is very effective for the function estimation, in particular for the problems involving the high-dimensional input space, there exist two limitations more or less. One is the computational complexity, that is to say, the computational burden of the conventional quadratic programming support vector regression (QP-SVR) scales cubically with the number of the training samples. Although the existing algorithms such as Osuna, Freund, and Girosi (1997), SMO (Platt, 1998; Shevade, Keerthi, Bhattacharyya, & Murthy, 2000), SVM^{light} (Joachims, 1999), SVMTorch (Collobert & Bengio, 2001), LIBSVM (Chang & Lin, 2001), and so forth alleviate the computational cost, this problem, that is, $R \cdot O(Nq + q)$, where R is the iterative number and q is the scale of working set, survives to a certain extent. Besides, the solution of QP-SVR is not sparse enough, even hardly controllable. For example, in system identification, the QP-SVR is not always able to construct sparse models (Drezet & Harrison, 1998). As an extreme, if error insensitivity is not contained, it will select the ensemble training set as support vectors (Drezet & Harrison, 2001). Lee and Billings (2002) compared the conventional SVM with uniformly regularized orthogonal least squares algorithm on time series prediction problems, and showed that both algorithms own similar excellent generalization performance but the resultant model from SVM is not sparse enough. Actually, the conventional QP-SVR can only give an upper bound on the number of necessary and sufficient support vectors because of the linear dependencies of support vectors in the high-dimensional feature space. Thus, some efforts (Drezet & Harrison, 2001; Li, Jiao, & Hao, 2007; Nguyen & Ho, 2006) have been made to let it gain

* Corresponding author.
E-mail addresses: y.p.zhao@nuaa.edu.cn, zhaoyongping_007@163.com (Y.-P. Zhao), jgspe@nuaa.edu.cn (J.-G. Sun).

sparser representation due to the Occam's razor, that is, "plurality should not be posited without necessity".

As a modification, the linear programming support vector regression (LP-SVR) (Weston et al., 1999; Smola, Schölkopf, & Rätsch, 1999) is developed to change the support vector problem from a quadratic programming to a linear programming problem. And from Kecman (2001) and Hadzic and Kecman (2000), we know that the linear programming support vector regression holds the superiorities of model sparse representation and computational efficiency to QP-SVR. The motivation of linear programming support vector machines is to utilize the linear kernel combination as an ansatz for the solution, and to employ a different regularizer, viz. the ℓ_1 norm of the coefficient vector, in the cost function. That is to say, the LP-SVR is treated as a linear one in the kernel space as a surrogate of the case of QP-SVR in the feature space. As we know, the choice of kernel function plays a paramount role in the modeling process of the kernel methods. When we face with the systems owning different data trends in different regions, the single kernel to construct model commonly does not achieve the satisfying result, i.e., the model does not globally fit the system. Furthermore, recently multikernel learning algorithms (Bi, Zhang, & Bennett, 2004; Lanckriet, Cristianini, Bartlett, Ghaoui, & Joran, 2004; Ong, Smola, & Williamson, 2005) have demonstrated the necessity to consider multiple kernels or the combination of kernels rather than a single fixed kernel. Hence, it is necessary to realize the multikernel learning for LP-SVR. Both the conventional QP-SVR and LP-SVR are implemented using the nonparametric technique. However, when one happens to have additional knowledge about the learning system, it is unwise not to take advantage of it. In this situation, the semiparametric technique (Smola, Frie, & Schölkopf, 1998) is a good selection to utilize the prior knowledge. If we are able to combine the semiparametric technique with multikernel trick, the identification results of the systems owning different data trends in different regions will be improved more or less (Nguyen & Tay, 2008). When data are provided us to identify the system, usually there is no additional knowledge to utilize, so some strategies may be used to mine the additional knowledge for the data-driven modeling. In general, the data selected by the support vector learning algorithms as support vectors are commonly significant for system identification. Here, we treat the pre-selected data as the additional knowledge to utilize the semiparametric technique to model the identification system, and after combining with the multikernel trick, a novel linear programming support vector learning algorithm, called as multikernel semiparametric linear programming support vector regression and MSLP-SVR for short, is proposed in this letter. The experimental results on the synthetic and real-world data sets corroborate the efficacy and validity of the presented MSLP-SVR.

The remainder of this letter is organized as follows. In the following section, the conventional linear programming support vector regression is dwelled on. Subsequently, after combining the semiparametric technique with the multikernel trick, we give the multikernel semiparametric linear programming support vector regression. To show the effectiveness of the MSLP-SVR, in Section 4 we do experiments on the synthetic and real-world data sets, and we compare MSLP-SVR with the other multikernel linear programming algorithm. Finally, conclusions follow.

2. Linear programming support vector regression

In this section, we will firstly introduce the conventional QP-SVR, and then LP-SVR is derived, because both algorithms have some similarities like the ε-insensitive loss function and kernel functions in the feature space. Given the training data set $\{(\boldsymbol{x}_i, d_i)\}_{i=1}^N$ of the size N, where $\boldsymbol{x}_i \in \Re^n$ is the input and d_i is the corresponding output, we construct a linear predictor $f(\cdot)$ in the high-dimensional (even infinite-dimensional) feature space with ε-insensitive loss function in the following:

$$\begin{aligned} \min_{w,b} \quad & \left\{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C'\sum_{i=1}^N(\xi_i^* + \xi_i)\right\} \\ \text{s.t.} \quad & d_i - \left(\boldsymbol{w}^T\cdot\varphi(\boldsymbol{x}_i) + b\right) \leqslant \varepsilon + \xi_i^* \\ & \boldsymbol{w}^T\cdot\varphi(\boldsymbol{x}_i) + b - d_i \leqslant \varepsilon + \xi_i \\ & \xi_i,\ \xi_i^* \geqslant 0, \quad i = 1,\ldots,N \end{aligned} \tag{1}$$

where ξ_i and ξ_i^* are the slack variables, $\varphi(\cdot)$ is a nonlinear mapping which can transform the input data x_is in the input space into $\varphi(\boldsymbol{x}_i)$ s in the feature space, $C' > 0$ is the regularization parameter which can control the tradeoff between the flatness of f and closeness to the training data. By defining the following ε-insensitive loss function,

$$H_\varepsilon(d_i - f(\boldsymbol{x}_i)) = \max\{0, |d_i - f(\boldsymbol{x}_i)| - \varepsilon\} \tag{2}$$

and letting $\lambda = (2C')^{-1}$, the optimization problem (1) can be reformulated as the following regularization problem:

$$\min_f \left\{\lambda\|\boldsymbol{w}\|^2 + \sum_{i=1}^N H_\varepsilon(d_i - f(\boldsymbol{x}_i))\right\} \tag{3}$$

Eq. (3) is completely equivalent to (1). From the representer theorem (Kimeldorf & Wahba, 1970), the optimal function of (3) can be represented as a linear combination of the kernel functions centering the training examples,

$$f(\boldsymbol{x}) = \sum_{i=1}^N \beta_i k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{4}$$

where $k(\boldsymbol{x}_i, \boldsymbol{x}) = \varphi(\boldsymbol{x}_i)\cdot\varphi(\boldsymbol{x})$ is the kernel function. $k(\cdot,\cdot)$ can be chosen from Gaussian, polynomial, or MLP, etc. Commonly, the Gaussian is the choice. Through changing the norm used in (3), i.e., ℓ_2 norm being replaced by the ℓ_1 norm (3) can be altered as

$$\min_f \left\{\lambda\|\boldsymbol{\beta}\|_1 + \sum_{i=1}^N H_\varepsilon(d_i - f(\boldsymbol{x}_i))\right\} \tag{5}$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_N]^T$. Hence, the corresponding form of Eq. (1) is stated in the following:

$$\begin{aligned} \min_{\beta,b} \quad & \left\{\frac{1}{2}\|\boldsymbol{\beta}\|_1 + C'\sum_{i=1}^N(\xi_i^* + \xi_i)\right\} \\ \text{s.t.} \quad & d_i - \left(\sum_{j=1}^N \beta_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) + b\right) \leqslant \varepsilon + \xi_i^* \\ & \sum_{j=1}^N \beta_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) + b - d_i \leqslant \varepsilon + \xi_i \\ & \xi_i,\ \xi_i^* \geqslant 0, \quad i = 1,\ldots,N \end{aligned} \tag{6}$$

In order to transform the above optimization problem into a linear programming problem, here we use a variable replacement trick to decompose β_j and $|\beta_j|$ as

$$\beta_j = \alpha_j^* - \alpha_j, \quad |\beta_j| = \alpha_j^* + \alpha_j \tag{7}$$

where $\alpha_j^*,\ \alpha_j \geqslant 0$. It is worth noting that the decompositions of (7) are unique and $\alpha_j^*\cdot\alpha_j = 0 \quad (j = 1,\ldots,N)$ are guaranteed implicitly. If $\alpha_j^* > 0,\ \alpha_j > 0$ and $\alpha_j^* > \alpha_j$ (similar for $\alpha_j^* < \alpha_j$) arise, then we will obtain a better solution $\alpha_j'^* = \alpha_j^* - \alpha_j,\ \alpha_j' = 0\ (i = 1, 2, \ldots, N)$ which satisfies the constraints and makes the cost function of (6) much smaller. Thus, the implicit constraints $\alpha_i^*\cdot\alpha_i = 0$ are guaranteed in the final solution of (6), that is to say, α_i^* or α_i must equal zero. If both α_i^* and α_i are equal to zeros, the corresponding sample $\boldsymbol{x}_i$ is non-support vector. After plugging (7) into (6), we obtain