Neural Networks 23 (2010) 1276–1285

Contents lists available at ScienceDirect

Neural Networks

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

A fast method to approximately train hard support vector regression

Yongping Zhao[a,*], Jianguo 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

Article history:
Received 30 September 2008
Received in revised form 17 March 2010
Accepted 2 August 2010

Keywords:
Support vector regression
Greedy algorithm
Regularization technique

ABSTRACT

The hard support vector regression (HSVR) usually has a risk of suffering from overfitting due to the presence of noise. The main reason is that it does not utilize the regularization technique to set an upper bound on the Lagrange multipliers so they can be magnified infinitely. Hence, we propose a greedy stagewise based algorithm to approximately train HSVR. At each iteration, the sample which has the maximal predicted discrepancy is selected and its weight is updated only once so as to avoid being excessively magnified. Actually, this early stopping rule can implicitly control the capacity of the regression machine, which is equivalent to a regularization technique. In addition, compared with the well-known software LIBSVM2.82, our algorithm to a certain extent has advantages in both the training time and the number of support vectors. Finally, experimental results on the synthetic and real-world benchmark data sets also corroborate the efficacy of the proposed algorithm.

1. Introduction

As a state-of-the-art tool for classification and regression, the support vector machine (SVM) (Burges, 1998; Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2002; Vapnik, 1995), which is built on the structural risk minimization principle that minimizes the upper bound of the generalization error which consists of training errors and a confidence interval term which is decided by the Vapnik–Chervonekis dimension, enjoys many successful applications to real-life fields, such as optimal control (Suykens, Vandewalle, & De Moor, 2001), image segmentation (Chen & Wang, 2005), time series prediction (Lau & Wu, 2008) and so on. Generally, there exist two stages for a good machine learning theory, for example, an artificial neural network (ANN), one of which is the model or frame selection, which includes the definition of the optimization problem and the parameter selection, i.e., what to do. Another is that what method or algorithm will be taken to solve the defined model or frame, i.e., how to do. However, these two stages are always dependent and not separate. Undoubtedly, SVM cannot escape from these two stages. Though very successful and familiar, there are shortcomings in each stage for SVM. In the first stage, the selection of parameters, including the kernel parameters and the regularization parameter, is still an open problem. There is no foolproof method to choose them before training. Usually, the cross validation (CV) technique is used to cope with this embarrassment, but the computational complexity is very expensive, even unaffordable. Another block is which optimization technique is employed to solve the constrained quadratic programming for training SVM in the second stage. Although the training SVM is solvable in principle, actually, a sophisticated optimization technique like the interior point method is ineffective to settle this intractable problem, especially for a large scale problem, because the training cost, viz. $O(N^3)$, where N is the size of the ensemble training samples, is huge.

Over the past few years, many fast algorithms have been developed to accelerate the training of SVM. Altogether, these algorithms are grouped into two categories, one of which trains SVM in the dual, which is the familiar strategy to cope with the training problem. Firstly, Osuna, Freund, and Girosi (1997) proposed the chunking algorithm to speed up the SVM training, but the large number of support vectors limits its application. Hence, Joachims (1999) proposed an efficient algorithm, namely, SVM$^{\text{light}}$. As a special case of SVM$^{\text{light}}$, Platt (1999) selected two variables as the working set and presented the famous sequential minimal optimization (SMO) so that the analytical solution for subproblem is easily obtained. Subsequently, the SMO algorithm was extended to the regression realm (Flake & Lawrence, 2002; Shevade, Keerthi, Bhattacharyya, & Murthy, 2000). Based on these works, Chang and Lin (2001) developed SMO and encoded it as the well-known software, viz. LIBSVM2.82. Usually, it is very easy to give readers the impression that this is the only possible way to train an SVM. In fact, there exists another way to train an SVM, i.e., the training procedure can be processed in the primal. Fung and Mangasarian (2003), and Mangasarian (2002) presented a finite Newton method to train a linear SVM and revealed that it is rather effective. Furthermore, some appropriate tricks were

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

0893-6080/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2010.08.001

introduced by Keerthi and DeCoste (2005) to modify and accelerate the finite Newton method. Chapelle (2007) proposed a recursive finite Newton method for nonlinear SVM and shown that an SVM can be solved in the primal as efficiently as the dual methods. Recently, Wang, Jia, and Li (2008) proposed a robust method to solve SVM in the primal. Bo, Wang, and Jiao (2007) presented a recursive finite Newton method for nonlinear support vector regression (SVR), and Zhao and Sun (2008) proposed a robust SVR in the primal with a non-convex loss function. Apart from these training algorithms to accelerate SVM/SVR in the dual or primal, some algorithms (Dong, Krzyzak, & Suen, 2005; Tsang, Kwok, & Cheung, 2005) exist to approximately train an SVM. No matter what algorithm is used to train a SVM/SVR, the aim is to accelerate the training procedure.

As we know, SVM is proposed based on a geometrical viewpoint, i.e., SVM is a large margin based classifier for binary classification. It seeks a linear hyperplane to separate the training samples as much as possible in the high dimensional feature space. We are familiar with this viewpoint. However, there exists another viewpoint to interpret SVM, that's to say, it matches the regularization learning frame *loss* + *penalty* (Evgeniou, Pontil, & Poggio, 2000; Girosi, 1998; Poggio & Smale, 2005) with the hinge loss function. Recently, the regularization learning frame has been generalized (Li, Lee, & Leung, 2007; Li, Leung, & Lee, 2007). Thus, besides the kernel parameters, the regularization parameter will be chosen to build an SVM, thus definitely increasing the computational complexity. Although the hard-margin support vector machine (HSVM) does not select the regularization parameter, it easily experiences overfitting in the presence of noise. To obtain a state-of-the-art performance as in the soft-margin support vector machine, viz. the so-called support vector machine, Bo, Wang, and Jiao (2008) proposed a greedy stagewise algorithm to cope with the overfitting of HSVMs without employing the regularization term. That is, the greedy stagewise strategy based HSVM (GS-HSVM) not only obtains a comparable generalization performance to the soft-margin support vector machine but also needs less computational complexity due to the absence of the regularization term. Enlightened by their work, we extend it to regression realm and propose a fast algorithm to approximately train a hard SVR (HSVR) using the greed stagewise strategy, named as GS-HSVR. This algorithm does not need to choose the extra regularization parameter and does to suffer from the overfitting due to the existing noise, so the computational complexity is clearly reduced, especially when using the CV technique to select an appropriate regularization parameter for SVR. As a matter of fact, the experimental results show that the early stopping rule of the GS-HSVR can control the capacity of the regression machine, which equivalently acts in an implicit regularization role. Moreover, the computational complexity of GS-HSVR is $O(N_{|SV|} \cdot N)$, where $N_{|SV|}$ is the number of support vectors. Even in the worst case, that is, all the training samples are chosen as support vectors, the computational burden is only $O(N^2)$. Compared with the well-known solver, i.e., LIBSVM2.82, our algorithm to a certain extent has advantages in the training time and the number of support vectors (#SV), which is corroborated by the experiments on the synthetic and real-world benchmark data sets.

In this paragraph, we will introduce the paper's organization. In Section 2, the classical SVR, namely soft SVR, is depicted briefly, and the hard SVR is deduced. Subsequently, we use the greedy stagewise strategy to approximately train a hard SVR and propose the GS-HSVR algorithm. Experiments on synthetic and real-life benchmark data sets confirm the effectiveness of the GS-HSVR in Section 4. Finally, conclusions follow.

2. Support vector regression

In the following paragraphs, we will briefly depict the classical SVR, viz. soft SVR, which follows the regularization learning frame. Given a training set $\{(\boldsymbol{x}_i, d_i)\}_{i=1}^N$, where $\boldsymbol{x}_i$ is the input data and d_i is the corresponding target, and the ε-insensitive loss function, the predictor, viz. the classical SVR model, can be got from solving the following optimization problem:

$$\min_{\boldsymbol{w},b} \left\{ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^N \left(\xi_i + \xi_i^*\right) \right\} \tag{1}$$
$$\text{s.t.}\quad d_i - (\boldsymbol{w}\cdot\varphi(\boldsymbol{x}_i) + b) \le \varepsilon + \xi_i^*$$
$$\boldsymbol{w}\cdot\varphi(\boldsymbol{x}_i) + b - d_i \le \varepsilon + \xi_i$$
$$\xi_i^*, \xi_i \ge 0, \quad i = 1, \dots, N$$

where C represents the regularization parameter which controls the tradeoff between the model complexity and the training errors. Using the kernel trick, we can obtain the dual expression of (1) below.

$$\min_{\alpha^*,\alpha} \frac{1}{2}\sum_{i,j=1}^N \left(\alpha_i^* - \alpha_i\right)\left(\alpha_j^* - \alpha_j\right) k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$$
$$+\,\varepsilon\sum_{i=1}^N \left(\alpha_i^* + \alpha_i\right) - \sum_{i=1}^N d_i\left(\alpha_i^* - \alpha_i\right) \tag{2}$$
$$\text{s.t.}\quad \sum_{i=1}^N \left(\alpha_i^* - \alpha_i\right) = 0$$
$$0 \le \alpha_i^* \le C,\ 0 \le \alpha_i \le C, \quad i = 1, \dots, N$$

where $\alpha^* = [\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*]^T$ and $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_N]^T$ are Lagrange multipliers, $k(\cdot,\cdot)$ is the kernel function which can be usually chosen from Gaussian, polynomial or MLP types. After solving (2), we can get the predictor, viz. SVR, as follows:

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{x}_i \in SV} \left(\alpha_i^* - \alpha_i\right) k\left(\boldsymbol{x}_i, \boldsymbol{x}\right) + b \tag{3}$$

where SV is the set of support vectors. (3) is a soft SVR which satisfies the regularization learning frame and is familiar to us. Actually, there is another hard SVR which usually leads to overfitting in the presence of noise, so we seldom use it. The HSVR is described as:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{4}$$
$$\text{s.t.}\quad d_i - (\boldsymbol{w}\cdot\varphi(\boldsymbol{x}_i) + b) \le \varepsilon$$
$$\boldsymbol{w}\cdot\varphi(\boldsymbol{x}_i) + b - d_i \le \varepsilon, \quad i = 1, \dots, N.$$

From (4), we know that HSVR will let all the training samples lie in the ε-insensitive tube. That is to say, all the training samples will not cause training errors. In other words, for HSVR the empirical risk is zero, which can be regarded as an extreme case emphasizing empirical risk. As a result, the decision hyperplane is too hard so as to suffer from overfitting in the presence of noise. If HSVR is understood from the framework of *loss* + *penalty*, the optimization objective of HSVR can be referred to as $C' \cdot$ *loss* + *penalty*, where C' is a very large positive number. Equivalently, the optimization objective is transformed to *loss* + $\frac{1}{C'}$*penalty*. According to this form, we understand that HSVR actually overwhelmingly emphasizes *loss*, viz. empirical risk, instead of *loss* + *penalty*. From this point, it is easily understood that why HSVR can cause overfitting. The dual form of (4) is given as follows:

$$\min_{\alpha^*,\alpha} \frac{1}{2}\sum_{i,j=1}^N \left(\alpha_i^* - \alpha_i\right)\left(\alpha_j^* - \alpha_j\right) k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$$