Pattern Recognition Letters 34 (2013) 1078–1084

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

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

Multi-output least-squares support vector regression machines

Shuo Xu [a], Xin An [b], Xiaodong Qiao [a], Lijun Zhu [a], Lin Li [c,*]

[a] Information Technology Supporting Center, Institute of Scientific and Technical Information of China No. 15 Fuxing Rd., Haidian District, Beijing 100038, China
[b] School of Economics and Management, Beijing Forestry University No. 35 Qinghua East Rd., Haidian District, Beijing 100038, China
[c] College of Information and Electrical Engineering, China Agricultural University No. 17 Qinghua East Rd., Haidian District, Beijing 100083, China

ARTICLE INFO

Article history:
Received 10 May 2012
Available online 4 February 2013

Communicated by S. Sarkar

Keywords:
Least-squares support vector regression machine (LS-SVR)
Multiple task learning (MTL)
Multi-output LS-SVR (MLS-SVR)
Positive definite matrix

ABSTRACT

Multi-output regression aims at learning a mapping from a multivariate input feature space to a multivariate output space. Despite its potential usefulness, the standard formulation of the least-squares support vector regression machine (LS-SVR) cannot cope with the multi-output case. The usual procedure is to train multiple independent LS-SVR, thus disregarding the underlying (potentially nonlinear) cross relatedness among different outputs. To address this problem, inspired by the multi-task learning methods, this study proposes a novel approach, Multi-output LS-SVR (MLS-SVR), in multi-output setting. Furthermore, a more efficient training algorithm is also given. Finally, extensive experimental results validate the effectiveness of the proposed approach.

1. Introduction

By changing the inequality constraints in the support vector regression machine (SVR) (Vapnik, 1999; Vapnik, 1998) by the equality ones, the least-squares SVR (LS-SVR) (Saunders et al., 1998; Suykens and Vandewalle, 1999; Suyken et al., 2002) replaces convex quadratic programming problem with convex linear system solving problem, thus largely speeding up training. It has been shown through a meticulous empirical study that the generalization performance of the LS-SVR is comparable to that of the SVR (Van Gestel et al., 2004). Therefore, the LS-SVR has been attracting extensive attentions during the past few years, such as (An et al., 2009; Choi, 2009; Xu et al., 2011b; Xu et al., 2011a) and references therein.

Multi-output regression aims at learning a mapping from a multivariate input space to a multivariate output space. Compared with the counterpart classification problem—multi-label classification problem (Tsoumakas and Katakis, 2007), the multi-output regression problem remains largely under-studied. To the best of our knowledge, only PLS (Partial Least Squares) regression (Abdi, 2003), kernel PLS regression (Rosipal and Trejo, 2001), MSVR (Multi-output SVR) (Tuia et al., 2011), and multi-output regression on the output manifold (Liu and Lin, 2009) have been put forward in literatures. What is more, it is difficult to generalize directly multi-label classification methods to counterpart regression ones.

Despite its potential usefulness, the standard formulation of the LS-SVR cannot cope with the multi-output case. The usual procedure considers developing a different LS-SVR to learn each parameter individually. That is to say, traditional approach treats the different outputs separately in the multi-output case, thus disregarding the underlying (potentially nonlinear) cross relatedness among different outputs. However, when there are relations between different outputs, it can be advantageous to learn all outputs simultaneously.

Then the problem is how to model the relatedness between different outputs. In fact, some clues from some multi-task learning methods such as hierarchical Bayesian methods (Bakker and Heskes, 2003; Heskes, 2000; Allenby and Rossi, 1998; Arora et al., 1998), which are based on some formal definition of the notion of relatedness of the tasks, motivate this work. Evgeniou and his coworkers (Evgeniou and Pontil, 2004; Evgeniou et al., 2005) proposed a regularized multi-task learning method by following the intuition of Hierarchical Bayes (Heskes, 2000; Allenby and Rossi, 1998; Arora et al., 1998). Our previous work (Xu et al., 2011b) is also based on the intuition with general setting. But, this paper restricts us to multi-output setting, since this setting permits us to design a more efficient training algorithm.

The organization of the rest of this paper is as follows. After LS-SVR for both single-output and multi-output cases are briefly described in Section 2, a novel multi-output regression approach, MLS-SVR, is proposed in Section 3. Similar to the LS-SVR, one only solves a convex linear system in the training phrase, too. In Section 4 and Section 5, extensive experimental evaluations are conducted, and Section 6 concludes this work.

* Corresponding author. Tel.: + 86 10 62732323.
E-mail addresses: xush@isitc.ac.cn (S. Xu), anxin927@bjfu.edu.cn (X. An), qiaox@isitc.ac.cn (X. Qiao), zhulj@isitc.ac.cn (L. Zhu), lilincau@gmail.com (L. Li).

0167-8655/$ - see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.patrec.2013.01.015

Notation

The following notations will be used in this study. Let $\mathbb{R}$ be the set of real numbers and $\mathbb{R}_+$ the subset of positive ones. For every $n \in \mathbb{N}$, the set of positive integers, we let $\mathbb{N}_n = \{1, 2, \ldots, n\}$. A vector will be written in bold case $\mathbf{x} \in \mathbb{R}^d$ with x_i as its i-th elements. The transpose of $\mathbf{x}$ is written as $\mathbf{x}^{\mathrm{T}}$. The vector $\mathbf{1}_d = [1, 1, \ldots, 1]^{\mathrm{T}} \in \mathbb{R}^d$ and $\mathbf{0}_d = [0, 0, \ldots, 0]^{\mathrm{T}} \in \mathbb{R}^d$. The inner product between two vectors is defined as $\mathbf{x}^{\mathrm{T}}\mathbf{z} = \sum_{k=1}^{d} x_k z_k$.

Matrices are denoted by capital bold letters $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $A_{i,j}$ as its (i,j)-th elements. The transpose of $\mathbf{A}$ is written as $\mathbf{A}^{\mathrm{T}}$. If $\mathbf{A}$ is an $m \times n$ matrix, we denote by $\mathbf{a}^i \in \mathbb{R}^m$ and $\mathbf{a}_j \in \mathbb{R}^n$ the i-th row and the j-th column of $\mathbf{A}$, respectively. If $\mathbf{A}$ is an $m \times m$ matrix, we define $trace(\mathbf{A}) := \sum_{i=1}^{m} A_{i,i}$. The identity matrix of dimension $m \times m$ is written as $\mathbf{I}_m$.

The function $repmat(\mathbf{A}, m, n)$ or $repmat(\mathbf{x}, m, n)$ creates a large block matrix consisting of an $m \times n$ tiling of copies of $\mathbf{A}$ or $\mathbf{x}$. The function $blockdiag(\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n)$ or $blockdiag(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ creates a block diagonal matrix, having $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n$ or $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ as main diagonal blocks, with all other blocks being zero matrices.

2. Least-squares support vector regression machine (LS-SVR)

2.1. Single-output case

The single-output regression is regarded as finding the mapping between an incoming vector $\mathbf{x} \in \mathbb{R}^d$ and an observable output $y \in \mathbb{R}$ from a given set of independent and identically distributed (i.i.d.) samples, i.e., $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$. Let $\mathbf{y} = (y_1, y_2, \ldots, y_l)^{\mathrm{T}} \in \mathbb{R}^l$. The single-output LS-SVR solves this problem by finding $\mathbf{w} \in \mathbb{R}^{n_h}$ and $b \in \mathbb{R}$ that minimizes the following objective function with constraints:

$$\min_{\mathbf{w} \in \mathbb{R}^{n_h}, b \in \mathbb{R}} \mathcal{J}(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \gamma \frac{1}{2}\xi^{\mathrm{T}}\xi, \tag{1}$$

$$\text{s.t.} \quad \mathbf{y} = \mathbf{Z}^{\mathrm{T}}\mathbf{w} + b\mathbf{1}_l + \xi, \tag{2}$$

where $\mathbf{Z} = (\varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2), \ldots, \varphi(\mathbf{x}_l)) \in \mathbb{R}^{n_h \times l}$, $\varphi : \mathbb{R}^d \to \mathbb{R}^{n_h}$ is a mapping to some higher (maybe infinite) dimensional Hilbert space $\mathcal{H}$ (also known as feature space) with n_h dimensions, $\xi = (\xi_1, \xi_2, \ldots, \xi_l)^{\mathrm{T}} \in \mathbb{R}^l$ is a vector consisting of slack variables, and $\gamma \in \mathbb{R}_+$ is a positive real regularized parameter.

The Lagrangian function for the problem 1,2 is

$$\mathcal{L}(\mathbf{w}, b, \xi, \alpha) = \mathcal{J}(\mathbf{w}, \xi) - \alpha^{\mathrm{T}}(\mathbf{Z}^{\mathrm{T}}\mathbf{w} + b\mathbf{1}_l + \xi - \mathbf{y}), \tag{3}$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_l)^{\mathrm{T}} \in \mathbb{R}^l$ is a vector consisting of Lagrange multipliers. The Karush–Kuhn–Tucker (KKT) conditions for optimality yield the following set of linear equations:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 & \Rightarrow \quad \mathbf{w} = \mathbf{Z}\alpha, \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \Rightarrow \quad \alpha^{\mathrm{T}}\mathbf{1}_l = 0, \\ \frac{\partial \mathcal{L}}{\partial \xi} = 0 & \Rightarrow \quad \alpha = \gamma\xi, \\ \frac{\partial \mathcal{L}}{\partial \alpha} = 0 & \Rightarrow \quad \mathbf{Z}^{\mathrm{T}}\mathbf{w} + b\mathbf{1}_l + \xi - \mathbf{y} = \mathbf{0}_l. \end{cases} \tag{4}$$

By eliminating $\mathbf{w}$ and ξ, one can obtain the following linear system:

$$\begin{bmatrix} 0 & \mathbf{1}_l^{\mathrm{T}} \\ \mathbf{1}_l & \mathbf{H} \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \tag{5}$$

with the positive definite matrix $\mathbf{H} = \mathbf{K} + \gamma^{-1}\mathbf{I}_l \in \mathbb{R}^{l \times l}$. Here, $\mathbf{K} = \mathbf{Z}^{\mathrm{T}}\mathbf{Z} \in \mathbb{R}^{l \times l}$ is defined by its elements $K_{i,j} = \varphi(\mathbf{x}_i)^{\mathrm{T}}\varphi(\mathbf{x}_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ for $\forall (i,j) \in \mathbb{N}_l \times \mathbb{N}_l$, and $\kappa(\cdot,\cdot)$ is a kernel function meeting the Mercer's theorem (Vapnik, 1999; Vapnik, 1998).

However, it is more difficult to solve directly the linear system (5), since its coefficient matrix is not positive definite. This can be overcome by reformulating it into the following one (Suyken et al., 2002; Suykens et al., 1999)

$$\begin{bmatrix} s & \mathbf{0}_l^{T} \\ \mathbf{0}_l & \mathbf{H} \end{bmatrix} \begin{bmatrix} b \\ \alpha + b\mathbf{H}^{-1}\mathbf{1}_l \end{bmatrix} = \begin{bmatrix} \mathbf{1}_l^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{y} \\ \mathbf{y} \end{bmatrix}, \tag{6}$$

where $s = \mathbf{1}_l^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{1}_l \in \mathbb{R}_+$. This new linear system (6) has a unique solution, and thus opens many opportunities for using fast and efficient numerical optimization methods. In fact, the solution of the problem (6) can be found by the following three steps (Suyken et al., 2002; Suykens et al., 1999):

1. Solve η, v from $\mathbf{H}\eta = \mathbf{1}_l$ and $\mathbf{H}v = \mathbf{y}$;
2. Compute $s = \mathbf{1}_l^{\mathrm{T}}\eta$;
3. Find solution: $b = \eta^{\mathrm{T}}\mathbf{y}/s, \alpha = v - b\eta$.

Therefore, the solution of the training procedure can be found by solving two sets of linear equations with the same positive definite coefficient matrix $\mathbf{H} \in \mathbb{R}^{l \times l}$. Since $\mathbf{H}$ is positive definite, one typically first finds the Cholesky decomposition $\mathbf{H} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$. Then since $\mathbf{L}$ is lower triangular, solving the system is simply a matter of applying forward and backward substitution. Other commonly used methods include the conjugate gradient, single value decomposition (SVD) or eigendecomposition, etc.

Let the solution of (5) be $\alpha^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_l^*)^{\mathrm{T}}$ and b^*. Then, the corresponding decision function is

$$\begin{aligned} f(\mathbf{x}) &= \varphi(\mathbf{x})^{\mathrm{T}}\mathbf{w}^* + b^* = \varphi(\mathbf{x})^{\mathrm{T}}\mathbf{Z}\alpha^* + b^* = \sum_{i=1}^{l} \alpha_i^* \varphi(\mathbf{x})^{\mathrm{T}}\varphi(\mathbf{x}_i) + b^* \\ &= \sum_{i=1}^{l} \alpha_i^* \kappa(\mathbf{x}, \mathbf{x}_i) + b^*. \end{aligned} \tag{7}$$

Thus, the single-output LS-SVR can be solved using only inner products between $\varphi(\cdot)$s, not needing to know the nonlinear mapping. However, in contrast to SVR, α^* is not sparse. This means that the whole training set needs to be used at prediction time.

2.2. Multi-output case

One can easily extend the single-output regression to the multiple output case (An et al., 2009). Let $\mathbf{Y} = [y_{ij}] \in \mathbf{R}^{l \times m}$. Given a set of i.i.d. samples $\{(\mathbf{x}_i, \mathbf{y}^i)\}_{i=1}^{l}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{y}^i \in \mathbb{R}^m$, the multi-output regression aims at predicting an output vector $\mathbf{y} \in \mathbb{R}^m$ from a given input vector $\mathbf{x} \in \mathbb{R}^d$. That is to say, the multi-output regression problem can be formulated as learning a mapping from $\mathbb{R}^d$ to $\mathbb{R}^m$. The multi-output LS-SVR (MLS-SVR) solves this problem by finding $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m) \in \mathbb{R}^{n_h \times m}$ and $\mathbf{b} = (b_1, b_2, \ldots, b_m)^{\mathrm{T}} \in \mathbb{R}^m$ that minimizes the following objective function with constraints:

$$\min_{\mathbf{W} \in \mathbb{R}^{m \times n_h}, \mathbf{b} \in \mathbb{R}^m} \mathcal{J}(\mathbf{W}, \Xi) = \frac{1}{2} trace(\mathbf{W}^{\mathrm{T}}\mathbf{W}) + \gamma \frac{1}{2} trace(\Xi^{\mathrm{T}}\Xi), \tag{8}$$

$$\text{s.t.} \quad \mathbf{Y} = \mathbf{Z}^{\mathrm{T}}\mathbf{W} + repmat(\mathbf{b}^{\mathrm{T}}, l, 1) + \Xi, \tag{9}$$

where $\Xi = (\xi_1, \xi_2, \ldots, \xi_m) \in \mathbb{R}_+^{l \times m}$.

On closer examination, it is not difficult to see that this is equivalent to m optimization problems similar to the problem 1,2. That is to say, the solution to the regression problem 8,9 decouples between the different output variables, and we need only compute a single inverse matrix, which is shared by all of the vectors $\mathbf{w}_i (\forall i \in \mathbb{N}_m)$. But it is much more efficient to solve 8,9 directly than to solve 1,2 m times, since they all share the same matrix $\mathbf{H} \in \mathbb{R}^{l \times l}$, the inverse matrix of which need be computed only once with the Cholesky decomposition, conjugate gradient, or SVD, etc.