Comparison of ℓ_1-Norm SVR and Sparse Coding Algorithms for Linear Regression

Qingtian Zhang, Xiaolin Hu, and Bo Zhang

***Abstract*— Support vector regression (SVR) is a popular function estimation technique based on Vapnik's concept of support vector machine. Among many variants, the ℓ_1-norm SVR is known to be good at selecting useful features when the features are redundant. Sparse coding (SC) is a technique widely used in many areas and a number of efficient algorithms are available. Both ℓ_1-norm SVR and SC can be used for linear regression. In this brief, the close connection between the ℓ_1-norm SVR and SC is revealed and some typical algorithms are compared for linear regression. The results show that the SC algorithms outperform the Newton linear programming algorithm, an efficient ℓ_1-norm SVR algorithm, in efficiency. The algorithms are then used to design the radial basis function (RBF) neural networks. Experiments on some benchmark data sets demonstrate the high efficiency of the SC algorithms. In particular, one of the SC algorithms, the orthogonal matching pursuit is two orders of magnitude faster than a well-known RBF network designing algorithm, the orthogonal least squares algorithm.**

***Index Terms*— Newton linear programming (NLP), radial basis function (RBF) neural network, regression, sparse coding (SC), support vector machine (SVM).**

I. Introduction

The support vector machine (SVM) is a famous learning algorithm proposed in [1] and [2]. It is one of the most popular models for both classification [3], [4] and regression [5], [6]. There is an important model for regression, termed the ℓ_1-norm support vector regression (SVR), which is able to identify the critical features for regression. This is useful in applications where a lot of noisy and redundant features are present, e.g., regression for biological experiment data. In general, the ℓ_1-norm SVR can be posed as a linear programming (LP) problem. Then, the standard LP solvers, such as the simplex algorithm [7] and the interior point algorithm [8] can be used to solve it. However, for large-scale problems, some more efficient algorithms are available and a popular one is the Newton LP (NLP) algorithm [9].

Sparse coding (SC) is an algorithm for finding a succinct representation of stimuli [10]. Given a set of bases, SC reduces to an ℓ_1-norm minimization problem, which amounts to finding the minimum ℓ_1-norm solution to an undetermined linear system $\boldsymbol{y} = B\boldsymbol{x}$, where $\boldsymbol{y}$ is the stimuli, B is the basis matrix, and $\boldsymbol{x}$ is a representation of $\boldsymbol{y}$ under the basis matrix. In what follows, we always assume that the bases are given in SC. In the past few years, a number of efficient SC algorithms have been proposed, such as matching pursuit (MP) [11], [12], homotopy (HOM) [13]–[15], feature sign search (FSS) [16], and so on. These algorithms have been applied in many real-world problems, such as face recognition [17] and image classification [18].

Both ℓ_1-norm SVR and SC are ℓ_1-minimization problems, and there should be some connections between them, though few researches have addressed this issue. We show that, ℓ_1-norm SVR with Gaussian loss function [19] is equivalent to SC with ℓ_2-norm penalty term, namely, least absolute shrinkage and selection operator (LASSO). Then, both algorithms can be used to solve linear regression problem. But it is unclear which one is more efficient. Through extensive experiments we will show that many SC algorithms are indeed more efficient.

Some notations are introduced first. For $\boldsymbol{x} \in \mathbb{R}^n$ and $p \in [1, \infty)$, $||\boldsymbol{x}||_p$ denotes the ℓ_p-norm of $\boldsymbol{x}$: $(\sum_{i=1}^n |x_i|^p)^{1/p}$. Let $\boldsymbol{e}$ denotes a column vector of all ones. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, $\boldsymbol{a}^T\boldsymbol{b}$ denote the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

II. Relationship Between ℓ_1-Norm SVR and SC

A. ℓ_1-Norm SVR

Given an unknown system $f : \mathbb{R}^{n-1} \to \mathbb{R}$ that transforms the input vector $\tilde{x} \in \mathbb{R}^{n-1}$ to a real number $f(\tilde{x})$. The objective of SVR is to estimate $f(\tilde{x})$ by observing m training instances $(\tilde{x}_1, f(\tilde{x}_1))$, $(\tilde{x}_2, f(\tilde{x}_2)), \ldots, (\tilde{x}_m, f(\tilde{x}_m))$. For linear SVR, f takes the form

$$f(\tilde{x}) = \tilde{\boldsymbol{w}}^T\tilde{x} + b \quad \text{with} \quad \tilde{\boldsymbol{w}} \in \mathbb{R}^{n-1}, \quad b \in \mathbb{R}. \tag{1}$$

By introducing $\boldsymbol{x} = \begin{pmatrix} \tilde{x} \\ 1 \end{pmatrix} \in \mathbb{R}^n$ and $\boldsymbol{w} = \begin{pmatrix} \tilde{\boldsymbol{w}} \\ b \end{pmatrix} \in \mathbb{R}^n$, (1) can be written in the homogeneous form

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}. \tag{2}$$

The primal problem of ℓ_1-norm SVR is

$$\begin{aligned} &\min \ ||\boldsymbol{w}||_1 \\ &\text{s.t. } y_i = \boldsymbol{w}^T\boldsymbol{x}_i, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{3}$$

In real-world applications, the output y_i may be corrupted by some unknown distributed noise. A number of loss functions $L(\boldsymbol{x}, y, f(\boldsymbol{x}))$, and a penalty term $\boldsymbol{e}^T\boldsymbol{\xi}$ have been introduced into (3) to cope with the noise. The formulation becomes

$$\begin{aligned} &\min \ ||\boldsymbol{w}||_1 + C\boldsymbol{e}^T\boldsymbol{\xi} \\ &\text{s.t. } L(\boldsymbol{x}_i, y_i, f(\boldsymbol{x}_i)) \le \xi_i, \quad i = 1, 2, \ldots, m, \quad \xi_i \ge 0 \end{aligned} \tag{4}$$

where $C > 0$ is a constant controlling the tradeoff between the regularization of f and the deviation tolerated. Different loss functions suit for different problems. Among them, the ϵ-insensitive loss function

$$L(\boldsymbol{x}, y, f(\boldsymbol{x})) = \max\{|y - f(\boldsymbol{x})| - \epsilon, 0\} \tag{5}$$

is one of the most commonly used loss functions, where ϵ is a prespecified value. With the ϵ-insensitive loss function, the formulation becomes

$$\begin{aligned} &\min \ ||\boldsymbol{w}||_1 + C\boldsymbol{e}^T\boldsymbol{\xi} \\ &\text{s.t. } |y_i - \boldsymbol{w}^T\boldsymbol{x}_i| \le \epsilon + \xi_i, \quad i = 1, 2, \ldots, m, \quad \xi_i \ge 0. \end{aligned} \tag{6}$$

Manuscript received May 24, 2013; revised August 28, 2014; accepted November 27, 2014. Date of publication December 18, 2014; date of current version July 15, 2015. This work was supported in part by the National Basic Research Program (973 Program) of China under Grant 2013CB329403 and Grant 2012CB316301, in part by the National Natural Science Foundation of China under Grant 61273023, in part by the Natural Science Foundation of Beijing under Grant 4132046, and in part by the Tsinghua National Laboratory for Information Science and Technology Cross-Discipline Foundation.

The authors are with the State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: forgettingzqt@yahoo.cn; xlhu@tsinghua.edu.cn; dcszb@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2377245

Note that this formulation can be recast into an LP problem and many efficient LP algorithms, such as [9] can be used to solve it. With the Gaussian loss function

$$L(\boldsymbol{x}, y, f(\boldsymbol{x})) = \frac{1}{2}(y - f(\boldsymbol{x}))^2 \tag{7}$$

the formulation becomes

$$\begin{aligned} &\min\ ||\boldsymbol{w}||_1 + C\boldsymbol{e}^T\boldsymbol{\xi} \\ &\text{s.t. } (y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 \le \xi_i, \qquad i = 1, 2, \ldots, m. \end{aligned} \tag{8}$$

The Gaussian-type noise is very common in practice and the Gaussian loss function matches the Gaussian distribution noise density model in the maximum likelihood sense under the assumption that the samples are generated by an underlying functional dependence plus additive noise [19]. One disadvantage of (8), compared with (6), seems to be that it cannot be recast into an LP problem and in the sequel cannot take advantages of LP solvers. But in what follows we will show that this is actually computationally advantageous because there exist many efficient algorithms for solving it. One should note that with Gaussian noise, the notion of support vector is meaningless. We call the problem SVR following the convention of [19].

B. Sparse Coding

Given an input signal $\boldsymbol{y} \in \mathbb{R}^m$ and a basis matrix $X \in \mathbb{R}^{m\times n}$, each column of X is a basis vector, and n is the number of basis vectors. The goal of SC is to represent the input signal as a weighted sum of a small number of basis vectors $\boldsymbol{y} = X\boldsymbol{w}$, where $\boldsymbol{w}$ is sparse, which means that most elements of $\boldsymbol{w}$ are zero. The basis set is often overcomplete, that is, $n > m$. Thus, SC is a nontrivial linear inversion problem. A popular formulation of SC is as follows [20]:

$$\begin{aligned} &\min\ ||\boldsymbol{w}||_1 \\ &\text{s.t. } \boldsymbol{y} = X\boldsymbol{w} \end{aligned} \tag{9}$$

which is actually the matrix form of (3). One should note that the columns of X (bases) do not correspond to the training points $\boldsymbol{x}_i$ in SVR; the rows of X do.

However, most signals in real-world applications cannot be represented perfectly and it is necessary to introduce a penalty term to deal with the noise in the signal. The ℓ_2-norm penalty is often used and the formulation becomes

$$\begin{aligned} &\min\ ||\boldsymbol{w}||_1 + C\xi \\ &\text{s.t. } ||\boldsymbol{y} - X\boldsymbol{w}||_2^2 \le \xi. \end{aligned} \tag{10}$$

C. Relationship Between ℓ_1-Norm SVR and SC

We have shown that the ideal forms of the ℓ_1-norm SVR (3) and the SC (9) are exactly the same. However, most of the time we do not solve these ideal cases directly. Noise must be considered and the formulations (8) and (10) are more often used. The following proposition shows that they are equivalent to each other.

Theorem 1: The ℓ_1-norm SVR with Gaussian loss function (8) is equivalent to the SC with ℓ_2-norm penalty term (10).

Proof: The Lagrangian function of (8) is

$$L_1 := ||\boldsymbol{w}||_1 + C\boldsymbol{e}^T\boldsymbol{\xi} + \sum_{i=1}^{m}\alpha_i((y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 - \xi_i) \tag{11}$$

where $\alpha_i \ge 0, i = 1, 2, \ldots, m$ are Lagrangian multipliers. It follows from the saddle point theorem that the optimal ξ_i's have to vanish:

$$\frac{\partial L_1}{\partial \xi_i} = C - \alpha_i = 0. \tag{12}$$

Substituting (12) into (11) yields an equivalent optimizing problem to (8)

$$\min\ ||\boldsymbol{w}||_1 + C\sum_{i=1}^{m}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2. \tag{13}$$

The Lagrangian function of (10) is

$$L_2 := ||\boldsymbol{w}||_1 + C\xi + \beta\left(||\boldsymbol{y} - X\boldsymbol{w}||_2^2 - \xi\right) \tag{14}$$

where $\beta \ge 0$ is the Lagrangian multiplier. With similar arguments to above we have

$$\frac{\partial L_2}{\partial \xi} = C - \beta = 0. \tag{15}$$

Substituting (15) into (14) yields an equivalent form of (10)

$$\min\ ||\boldsymbol{w}||_1 + C||\boldsymbol{y} - X\boldsymbol{w}||_2^2. \tag{16}$$

This formulation is the same as (13) with $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m)^T$. Therefore, (8), (10), (13), and (16) are equivalent. ■

Equation (16) is often called LASSO. In other words, the ℓ_1-norm SVR with Gaussian-type noise is actually a LASSO problem.

III. Algorithms for Comparison

Several efficient algorithms for ℓ_1-norm SVR and SC problems are briefly reviewed in this section.

A. Newton Linear Programming

Consider a general LP problem

$$\begin{aligned} &\min\ \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{y} \\ &\text{s.t. } A\boldsymbol{x} + B\boldsymbol{y} \ge \boldsymbol{b}, E\boldsymbol{x} + G\boldsymbol{y} = \boldsymbol{h}, \quad \boldsymbol{x} \ge 0 \end{aligned} \tag{17}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^l$ are variables and $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{d} \in \mathbb{R}^l$, $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times l}$, $E \in \mathbb{R}^{k\times n}$, $G \in \mathbb{R}^{k\times l}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{h} \in \mathbb{R}^k$ are constants, and its dual

$$\begin{aligned} &\min\ \boldsymbol{b}^T\boldsymbol{u} + \boldsymbol{h}^T\boldsymbol{v} \\ &\text{s.t. } A^T\boldsymbol{u} + E^T\boldsymbol{v} \le \boldsymbol{c}, B^T\boldsymbol{u} + G^T\boldsymbol{v} = \boldsymbol{d}, \quad \boldsymbol{u} \ge 0 \end{aligned} \tag{18}$$

where $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^k$ are variables. The NLP algorithm applies the Newton method to solve the exterior penalty function for the dual problem

$$\begin{aligned} \min\ &\epsilon(-\boldsymbol{b}^T\boldsymbol{u} - \boldsymbol{h}^T\boldsymbol{v}) + \frac{1}{2}(||(A^T\boldsymbol{u} + E^T\boldsymbol{v} - \boldsymbol{c})_+||^2 \\ &+ ||B^T\boldsymbol{u} + G^T\boldsymbol{v} - \boldsymbol{d}||^2 + ||(-\boldsymbol{u})_+||^2) \end{aligned} \tag{19}$$

which is a completely unconstrained differentiable piecewise-quadratic function that contains a single finite parameter. Both ℓ_1-norm support vector classification and ℓ_1-norm SVR can be written in the form of (17), and thus can be solved using NLP. An NLP is proved faster than the simplex algorithm and interior point algorithm for ℓ_1-norm SVM classification problem [21] and regression problem [22] when the number of features are larger than the number of samples.

B. Orthogonal Matching Pursuit

The orthogonal MP (OMP) method [11] is a greedy algorithm for solving (9). With a finite bases set of size n, OMP converges to the projection of $\boldsymbol{y}$ onto the span of the basis vectors in no more than n iterations. It is a recursive method based on the MP algorithm [23]. It maintains full backward orthogonality of the residual at every step and thereby leads to improved convergence. The kth iteration of the OMP results in an intermediate representation of $\boldsymbol{y}$ in the form

$$\boldsymbol{y} = \sum_{i=1}^{k} w_i\boldsymbol{x}_i + R_k(\boldsymbol{y}) = \boldsymbol{y}_k + R_k(\boldsymbol{y}) \tag{20}$$

where $\boldsymbol{y}_k$ is the current approximation and $R_k(\boldsymbol{y})$ is the current residual. The improvement of OMP based on MP is that it requires $\langle R_k(\boldsymbol{y}), \boldsymbol{x}_i \rangle = 0$. With a limited number of iterations N, where $N <= n$, OMP obtains the best approximation of $\boldsymbol{y}$ using the N basis vectors that have been selected from the basis matrix such that $||R_N(\boldsymbol{y})||_2^2$ is minimum. OMP was shown to be the most effective algorithm for noise-free data among many SC algorithms for face recognition [17].

C. Homotopy

The HOM algorithm [13]–[15] is an iterative method for solving (16). An HOM exploits the fact that the objective function $F(\boldsymbol{w})$ undergoes a HOM from the ℓ_2-norm constraint to the ℓ_1-norm objective (16) as C increases. One can further show that the solution path is piecewise constant as a function of C. Therefore, we can get the solution path by constructing an increasing sequence of C. It is only necessary to identify those breakpoints that lead to changes of the support set of $\boldsymbol{w}_C^*$, namely, either a new nonzero coefficient (N0C) added or a previous N0C removed. If the ground truth signal $\boldsymbol{w}$ has at most k nonzero components with $k \ll n$, HOM can recover $\boldsymbol{w}$ in k iterations. It was reported that HOM achieved the highest face recognition accuracy, while the computational cost was comparable with other SC algorithms [17].

D. Alternating Direction Method

The alternating direction method (ADM) [24] is a fast algorithm for solving the following structured optimization problem:

$$\begin{aligned} &\min\ f(\boldsymbol{x}) + g(\boldsymbol{y}) \\ &\text{s.t. } A\boldsymbol{x} + B\boldsymbol{y} = \boldsymbol{b} \end{aligned} \tag{21}$$

where $\boldsymbol{x} \in \mathbb{R}^m$, $\boldsymbol{y} \in \mathbb{R}^n$, and $\boldsymbol{b} \in \mathbb{R}^p$. $f(\boldsymbol{x}) : \mathbb{R}^m \to \mathbb{R}$ and $g(\boldsymbol{y}) : \mathbb{R}^n \to \mathbb{R}$ are both convex functions. It is obvious that $\boldsymbol{x}$ and $\boldsymbol{y}$ are decoupled in the objective function, and coupled only in the constraint. The ADM utilizes the decoupled structure and replaces the joint minimization by two simpler subproblems. Specially, ADM minimizes $\boldsymbol{x}$ and $\boldsymbol{y}$ separately via a Gauss–Seidel type iteration.

Rewrite (16) as follows:

$$\begin{aligned} &\min\ ||\boldsymbol{w}||_1 + C||\boldsymbol{r}||_2^2 \\ &\text{s.t. } X^T\boldsymbol{w} + \boldsymbol{r} = \boldsymbol{y}. \end{aligned} \tag{22}$$

Then, ADM can be applied. The ADM was reported to be efficient with random Gaussian basis set [17].

E. Feature Sign Search

The FSS method [16] also solves (16). The basic idea is to treat it as an unconstrained quadratic optimization problem. If we know the sign of w_i at the optimal value, we can replace each of the terms $|w_i|$ with either w_i (if $w_i > 0$), $-w_i$ (if $w_i < 0$), or 0 (if $w_i = 0$). Then, the problem becomes a standard unconstrained quadratic optimization problem, which can be solved analytically. The goal of the FSS method degenerates to search the correct signs of w_i. The algorithm proceeds in a series of feature-sign steps. At each step, it computes the analytical solution to the problem with current signs of w_i and updates the solution and the signs of w_i with an efficient line search strategy. Each step will reduce the objective function, and the overall algorithm always converges to the optimal solution. An FSS was shown to be much faster than least angle regression [25] and grafting method [26] for learning sparse representation of natural image patches [16].

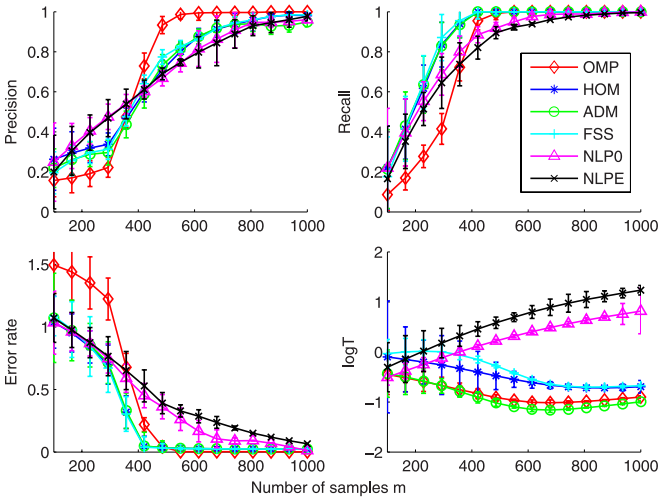

Fig. 1. Comparison of the six algorithms with fixed $n = 1000$ and $k = 100$. The curves show the average over 50 trials and the error bars show the standard deviation.

IV. Numerical Comparisons

We compared the SC algorithms described in the previous section with NLP on some linear regression problems. Notice that both (3) and (6) can be recast as LP problems and NLP was applied on both of them. The algorithms are denoted by NLP0 and NLPE, respectively. All algorithms were implemented in MATLAB and tested on the same PC (Intel Core i5-3450 CPU 3.50 GHz, RAM 4.0 GB).

We designed the regression problems as follows. The sample matrix $X \in \mathbb{R}^{m \times n}$ was generated as a random Gaussian matrix. Each entry in the matrix was independent and identically distributed, and every column vector was normalized to unit ℓ_2-norm. Randomly set k entries in the ground truth $\boldsymbol{w}_0$ to be nonzero and the others zero. The nonzero values were drawn from a uniform distribution in $[-5, 5]$. The observed value $\boldsymbol{y}$ was calculated as $\boldsymbol{y} = X\boldsymbol{w}_0$. Then, a Gaussian distribution noise was added to $\boldsymbol{y}$. The signal to noise ratio is 20 dB. To measure how the training time of each algorithm changes with different m, n, and k, two parameters were fixed while the other one was varied. The parameters of each algorithm were tuned by cross-validation on a particular data set size ($m = 500$, $n = 1000$, and $k = 100$).

The following quantities were used for evaluating the performances of the algorithms.

1) *Precision:* TP/TP + FP, where TP stands for the number of true positive samples and FP stands for the number of false positive samples. Here, a nonzero element in the learned $\boldsymbol{w}$ is called a true positive sample if the corresponding element in $\boldsymbol{w}_0$ is also nonzero; otherwise it is called a false positive sample.
2) *Recall:* TP/TP + FN, where TP is the same as above and FN stands for the number of false negative samples. Here, a zero element in the learned $\boldsymbol{w}$ is called a false negative sample if the corresponding element in $\boldsymbol{w}_0$ is nonzero.
3) *Error Rate:* $||\boldsymbol{w} - \boldsymbol{w}_0||_2 / ||\boldsymbol{w}_0||_2$.
4) *logT:* The training time (seconds) in $\log_{10}$.

First, we fixed n and k to 1000 and 100, respectively, and increased m from 100 to 1000 with a step size of 60. We had the following observations (Fig. 1).

1) As the sample size m increased, the quality of solutions obtained by all algorithms also increased (higher precision and recall, and smaller testing error).

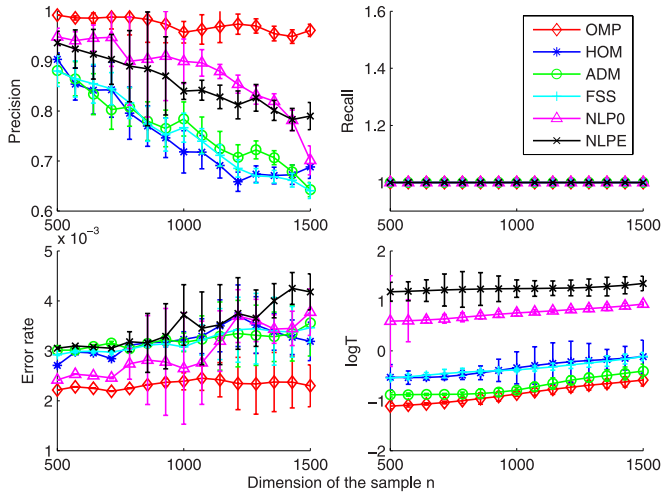

Fig. 2. Comparison of the six algorithms with fixed $m = 500$ and $k = 100$. The curves show the average over 50 trials and the error bars show the standard deviation.

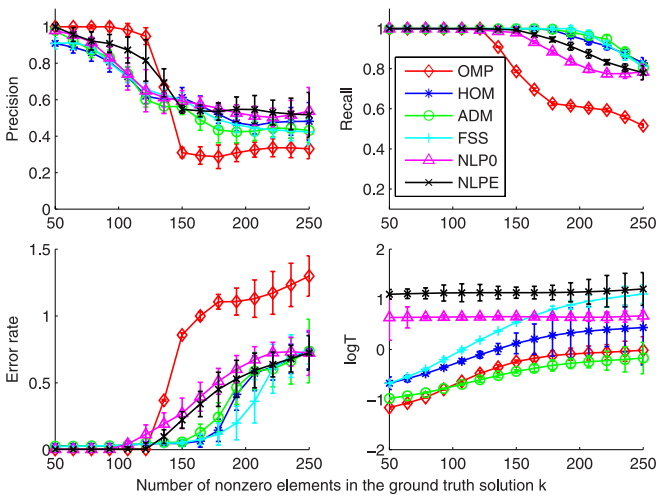

Fig. 3. Comparison of the six algorithms with fixed $m = 500$ and $n = 1000$. The curves show the average over 50 trials and the error bars show the standard deviation.

2) With $m \leq 500$, none of the algorithms found the ground truth solution.
3) With $m > 500$, OMP obtained the ground truth solution with 100% precision and recall. While other algorithms obtained an approximate solution which was very close to the ground truth solution.
4) With $m > 500$, the SC algorithms were much faster than the two NLP algorithms.

Second, we fixed m and k to 500 and 100, respectively, and increased n from 500 to 1500 with a step size of 125. We had the following observations (Fig. 2).

1) All algorithms obtained high-quality solutions with a very small testing error and 100% recall.
2) As the number of redundant features grew, the precision of all algorithms decreased linearly. The OMP was more robust to the redundant features than other algorithms.
3) NLP0 and NLPE took much longer training time than the SC algorithms.

Finally, m and n were fixed to 500 and 1000, respectively, and k was increased from 50 to 250 with a step size of 15 (Fig. 3). The error rate of all algorithms were nearly zero when $k < 100$. All SC algorithms ran much faster than NLP0 and NLPE. The OMP and ADM were again the first and second in speed and OMP selected the N0Cs perfectly. But when the ground truth solution became dense (larger k), the average error rates of all algorithms blew up very quickly and their precision and recall dropped down dramatically. The OMP was most affected by dense ground truth solutions, while ADM and FSS were most robust to dense ground truth solutions.

In all of the three scenarios discussed above, NLP0 and NLPE obtained solutions of similar quality, but the latter always took longer training time. It is because that both algorithms solved the problem in the dual space but the formulation (6) of NLPE has double number of dual variables as the formulation (3) of NLP0.

Overall, on this synthetic problem, when the samples were enough or the useful features were sparse enough, OMP performed best, which followed ADM.

In general, increasing the sample size, the number of nonzero elements in the ground truth, or decreasing sample dimension make the problem easier (higher precision and recall and smaller testing error). Likewise, increasing the samples size also has the knock on effect of increasing training time as there is more data to process. But in our results, the training times of SC algorithms decreased at first then slowly increased as the sample size increased. It is because the initial sample size was not enough for SC algorithms to get a reasonable solution (Fig. 1). They spent more time on searching the solution.

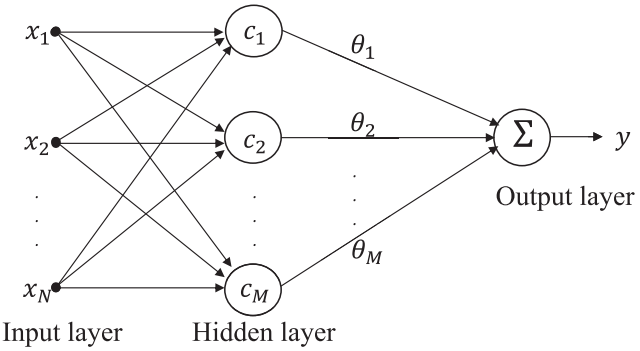

Fig. 4. Structure of the RBF network with a single output neuron.

V. Design of RBF Neural Network

The radial basis function (RBF) neural network is widely used for nonlinear system modeling and identification due to their simple topology [27]. Let $\boldsymbol{x} \in \mathbb{R}^n$ denote the input of the network. The output is

$$y = \sum_{j=1}^{M} \theta_j \psi(||\boldsymbol{x} - \boldsymbol{c}_j||_2, \sigma_j) \tag{23}$$

where M is the number of hidden units, $\{\theta_1, \ldots, \theta_M\}$ are the output weights, $\psi(.)$ is the Gaussian function, $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M\}$ are the centers, and $\{\sigma_1, \ldots, \sigma_M\}$ are the widths of the Gaussian functions. The structure of the RBF network is shown in Fig. 4.

By defining

$$\psi_{i,j} = \psi(||\boldsymbol{x}_i - \boldsymbol{c}_j||_2, \sigma_j) \tag{24}$$

TABLE I
RESULT FOR THE NONLINEAR FUNCTION APPROXIMATION

	Results for (26)			Results for (27)		
	MSE	T (s)	N0C	MSE	T (s)	N0C
OMP	0.090±0.002	0.11±0.01	20.2±0.8	**.027±0.005**	**0.14±0.03**	56.2±4.9
HOM	0.122±0.013	0.22±0.03	**11.8±0.5**	0.092±0.013	0.68±0.07	44.8±8.9
ADM	0.159±0.008	**0.09±0.01**	47.2±8.6	0.113±0.018	0.54±0.06	102.7±16.3
FSS	**0.087±0.002**	0.16±0.04	21.6±0.5	0.033±0.004	0.28±0.04	51.9±4.4
NLP0	0.091±0.008	2.69±0.38	27.4±2.7	0.028±0.004	10.13±0.15	79.3±2.4
NLPE	0.097±0.006	3.34±0.56	25.6±3.3	0.031±0.005	24.06±0.20	76.0±3.2
OLS	0.096±0.004	3.81±0.49	14.6±1.7	0.096±0.009	27.81±0.18	**36.1±2.8**

the network output y_i becomes

$$y_i = \sum_{j=1}^{M} \theta_j \psi_{i,j} \tag{25}$$

which is an undetermined linear system if $\{\psi_{i,j}, y_i\}$ is given.

Model structure determination (how to calculate $\psi_{i,j}$ from $\boldsymbol{x}_i$) and parameter optimization (solving θ_j) are the two important issues in RBF neural network design [28]. For the first issue, the linear-in-the-parameter approach is often used [29]–[31]. In this approach, all training samples are used as the candidate RBF centers as $\boldsymbol{c}_i = \boldsymbol{x}_i$, and the Gaussian function widths are set *a priori* $\sigma_j = \sigma_0$. With this setting, the ℓ_1-norm SVR can be used to deal with the second issue by solving the undetermined linear system (25). A well-known approach for dealing with the second issue refers to the orthogonal least squares (OLS) [29]. Though many years have passed, OLS remains to be one of the most efficient algorithms for solving this problem [30]. In what follows we will compare it with the ℓ_1-norm SVR and SC algorithms. The parameters of each algorithm were tuned by cross-validation on a small data set.

A. Noisy Nonlinear Function

We tested the performance of these algorithms with the RBF network (25) on the problem of nonlinear fitting. We designed the two nonlinear functions. The first one had a single variable

$$y_1 = \frac{\sin(x)}{x} + \ln(x^2 + 1) - e^{\frac{|x|}{10}}. \tag{26}$$

The second one had two variables

$$y_2 = 3(1 - x_1)^2 e^{-x_1^2 - (x_2+1)^2} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)e^{-x_1^2 - x_2^2} - \frac{1}{3}e^{-(x_1+1)^2 - x_2^2}. \tag{27}$$

Then, we added Gaussian noises η_1 and η_2 to them, respectively. η_1 was sampled from a Gaussian distribution with mean 0 and standard deviation 0.05 and η_2 was sampled from a Gaussian distribution with mean 0 and standard deviation 0.2. For the first problem, 1000 samples were generated where x was drawn from a uniformly distribution in $[-15, 15]$. We randomly selected 800 samples for training, and the rest for testing. The Gaussian function widths were set to $\sigma_0 = 10$. For the second problem, 49×49 samples were generated in which x_1 and x_2 were both 49 linearly equally spaced values from -3 to 3. We randomly selected 1500 points for training and the rest for testing. The Gaussian function widths were set to $\sigma_0 = 1$. The regression results are shown in Table I, where the mean squared testing error (MSE), the average training time (T) and the average number of nonzero elements in the solution N0C over 50 trials are reported. The N0C is the number of active hidden units in the trained RBF neural networks which directly determines the predicting time.

In all experiments presented hereafter, the one tailed t-test was used to assess the statistical significance with $p < 0.05$ for comparison. Before the t-test, the Kolmogorov–Smirnov test was used to verify that all these results are Gaussian distributed with the significant level 0.05.

For (26), the SC algorithms ran considerably faster than OLS (Table I). The two NLP algorithms obtained solutions of similar quality to OLS but ran slightly faster ($p < 10^{-4}$) which is in agreement with [22]. Among the SC algorithms, FSS obtained almost the best testing accuracy ($p < 0.06$) with acceptable training time and network size. An HOM obtained smallest model ($p < 10^{-4}$) and ADM took least time ($p < 4 \times 10^{-3}$) but their prediction accuracy were lower ($p < 10^{-4}$) than other algorithms. The accuracy of OMP was not significantly lower than the best FSS ($p = 0.06$), but its training time was less than FSS ($p < 10^{-4}$) and model size was smaller than FSS ($p = 0.02$).

The result for (27) in Table I shows that, the four SC algorithms were tens of times faster for training than the two NLP algorithms and OLS. Among the SC algorithms, OMP and FSS predicted better than HOM and ADM ($p < 10^{-4}$). Compared with FSS, OMP achieved a similar network size ($p = 0.09$) in less time ($p < 10^{-4}$).

In summary, the SC algorithms were usually tens of times faster than NLP and OLS, although none of them obtained higher prediction accuracy and smaller N0C than NLP and OLS at the same time. Among the four SC algorithms, no one outperformed others under all of the three criteria.

B. Real-World Problems

We tested these algorithms on some real-world problems from the University of California Irvine (UCI) regression database [32] with the RBF network regression. Six regression data sets [Abalone, Boston housing, Auto miles per gallon (MPG), Elevators, Census (house8L), and Computer Activity] were used. The results are shown in Table II. The average MSE, training time (T), and the number of N0C over 20 trials for each data set were reported. In each trail, the samples were randomly split for training and testing. The size of each training and testing data set are shown in Table II. The testing time for all algorithms were almost the same (hundreds of times less than their training time), so we did not concern it in comparison.

The observations are summarized as follows.

1) No algorithms had obtained the highest testing accuracy on all data sets.
2) No algorithms had obtained the smallest N0C on all data sets.
3) The SC algorithms were tens of times faster than NLP0, NLPE, and OLS for training. In particular, OMP was two orders of magnitude faster than OLS. Among the SC algorithms, OMP was the fastest, though this result was not always significant (significant on Boston housing [$p < 0.01$], Auto MPG [$p < 10^{-4}$], Elevators [$p < 10^{-4}$], and Census [$p < 5 \times 10^{-3}$]).

TABLE II
RESULT ON THE UCI REGRESSION DATA SETS

	Abalone (4177, 2000) *			Boston Housing (506, 400)			Auto MPG (398, 300)		
	MSE	T(s)	N0C	MSE	T(s)	N0C	MSE	T(s)	N0C
OMP	5.44±0.11	**0.242±0.003**	**70.3±4.2**	**11.02±3.03**	**0.020±0.001**	24.7±2.8	9.73±1.80	**0.013±0.001**	**9.1±0.8**
HOM	6.77±0.18	0.291±0.016	96.1±8.7	11.08±3.32	0.106±0.026	**22.9±1.6**	8.87±2.1	0.025±0.002	15.2±2.1
ADM	6.61±0.13	0.244±0.007	113.9±16.3	13.82±4.11	0.021±0.001	32.1±3.0	10.45±2.61	0.016±0.001	18.6±2.9
FSS	5.28±0.22	0.267±0.002	80.7±5.3	11.30±3.06	0.023±0.002	16.2±1.8	9.10±2.35	0.020±0.001	13.8±2.0
NLP0	5.38±0.26	15.11±0.63	135.0±9.9	12.08±2.43	4.001±0.132	28.5±4.2	**8.75±1.30**	2.894±0.072	18.2±1.9
NLPE	5.18±0.24	32.92±1.64	110.4±8.0	12.93±3.50	5.603±0.183	26.6±2.6	9.03±2.59	3.413±0.099	12.1±1.3
OLS	**4.90±0.34**	35.81±1.90	119.2±11.5	12.17±4.28	6.317±0.208	28.0±3.8	9.45±2.92	4.127±0.107	15.5±2.4
	Elevators (16659, 8000)			Census (house8L) (22784, 11000)			Computer Activity (8192, 4096)		
	MSE	T(s)	N0C	MSE	T(s)	N0C	MSE	T(s)	N0C
OMP	4.43±0.10	**0.702±0.023**	**116.1±5.7**	**5.23±0.34**	**1.246±0.098**	173.1±8.5	**8.75±0.99**	**0.518±0.011**	80.7±5.29
HOM	6.01±0.20	1.037±0.092	158.3±9.1	14.61±0.32	1.878±0.109	**127.7±9.3**	12.67±1.81	0.672±0.028	**35.2±5.2**
ADM	5.98±0.16	0.739±0.028	178.5±16.3	9.27±0.41	1.270±0.033	219.2±19.0	9.11±1.61	0.520±0.008	111.8±9.5
FSS	**4.28±0.14**	0.759±0.025	120.2±8.1	5.77±0.26	1.299±0.028	183.5±10.2	9.23±1.53	0.528±0.011	79.2±5.9
NLP0	5.01±0.16	58.60±1.20	145.8±7.7	6.01±0.33	83.68±2.18	192.3±10.5	9.32±1.53	40.63±1.69	91.5±7.4
NLPE	5.03±0.16	95.03±2.16	152.4±9.2	6.16±0.30	120.2±4.7	184.0±9.3	9.63±1.11	73.72±3.37	86.1±6.0
OLS	5.12±0.28	113.2±4.2	119.2±.3	5.89±0.28	129.4±5.2	201.1±11.5	8.93±1.81	81.40±2.73	95.3±6.4

* The first number behind the name of each dataset stands for the total number of samples, and the second number stands for the number of training samples.

VI. CONCLUSION

We explored the relationship between the ℓ_1-norm SVR and SC, and compared the NLP algorithms, an efficient ℓ_1-norm SVR solver, with some typical SC algorithms. The contribution of this brief is as follows. First, we proved that the ℓ_1-norm SVR with Gaussian noise is equivalent to SC. Second, through extensive experiments we found that many SC algorithms were significantly more efficient than the NLP algorithm for linear regression. As an application, we formulated the design of RBF neural network as a linear regression problem. Experiments on some benchmark data sets demonstrated the higher efficiency of the SC algorithms compared with the well-known method, the OLS algorithm. In particular, the OMP algorithm was two orders of magnitude faster than OLS.

REFERENCES

[1] V. Vapnik and A. Lerner, "Pattern recognition using generalized portrait method," *Autom. Remote Control*, vol. 24, no. 6, pp. 774–780, 1963.

[2] V. Vapnik and A. Chervonenkis, "A note on one class of perceptrons," *Autom. Remote Control*, vol. 25, no. 1, pp. 821–837, 1964.

[3] B. Schölkopf, C. Burges, and V. Vapnik, "Incorporating invariances in support vector learning machines," in *Proc. Int. Conf. Artif. Neural Netw.*, 1996, pp. 47–52.

[4] B. Schölkopf, P. Simard, A. Smola, and V. Vapnik, "Prior knowledge in support vector kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 1998, pp. 640–646.

[5] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, "Predicting time series with support vector machines," in *Proc. Int. Conf. Artif. Neural Netw.*, 1997, pp. 999–1004.

[6] M. O. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston, "Support vector regression with ANOVA decomposition kernels," in *Proc. Adv. Kernel Methods-Support Vector Learn.*, 1999, pp. 285–292.

[7] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965.

[8] M. Kojima, S. Mizuno, and A. Yoshise, *A Primal-Dual Interior Point Algorithm for Linear Programming*. New York, NY, USA: Springer-Verlag, 1989.

[9] O. L. Mangasarian, "Exact 1-norm support vector machines via unconstrained convex differentiable minimization," *J. Mach. Learn. Res.*, vol. 7, no. 2, pp. 1517–1530, 2006.

[10] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, Jun. 1996.

[11] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. 27th Asilomar Conf. Signals, Syst. Comput.*, Nov. 1993, pp. 40–44.

[12] S. S. Chen, D. L. Donoho, and A. S. Michael, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.

[13] M. R. Osborne, B. Presnell, and B. A. Turlach, "A new approach to variable selection in least squares problems," *IMA J. Numer. Anal.*, vol. 20, no. 3, pp. 389–403, 2000.

[14] D. M. Malioutov, M. Cetin, and A. S. Willsky, "Homotopy continuation for sparse signal representation," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, vol. 5. Mar. 2005, pp. 733–736.

[15] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution," *Commun. Pure Appl. Math.*, vol. 59, no. 6, pp. 797–829, Jun. 2006.

[16] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19. 2007, p. 801.

[17] A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Fast ℓ1-minimization algorithms and an application in robust face recognition: A review," in *Proc. Int. Conf. Image Process.*, Sep. 2010, pp. 1849–1852.

[18] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1794–1801.

[19] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.

[20] A. Y. Ng, "Feature selection, L_1 vs. L_2 regularization, and rotational invariance," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 78.

[21] G. M. Fung and O. L. Mangasarian, "A feature selection Newton method for support vector machine classification," *Comput. Optim. Appl.*, vol. 28, no. 2, pp. 185–202, 2004.

[22] M. Han and J. Yin, "The hidden neurons selection of the wavelet networks using support vector machines and ridge regression," *Neurocomputing*, vol. 72, nos. 1–3, pp. 471–479, Dec. 2008.

[23] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[24] J. Yang and Y. Zhang, "Alternating direction algorithms for ℓ_1-problems in compressive sensing," *SIAM J. Sci. Comput.*, vol. 33, no. 1, pp. 250–278, 2011.

[25] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.

[26] S. Perkins and J. Theiler, "Online feature selection using grafting," in *Proc. 20th Int. Conf. Mach. Learn.*, vol. 20. 2003, pp. 592–599.

[27] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, Jun. 1991.

[28] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[29] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[30] X. Hong, R. J. Mitchell, S. Chen, C. J. Harris, K. Li, and G. W. Irwin, "Model selection approaches for non-linear system identification: A review," *Int. J. Syst. Sci.*, vol. 39, no. 10, pp. 925–946, Jul. 2008.

[31] J.-X. Peng, K. Li, and D.-S. Huang, "A hybrid forward algorithm for RBF neural network construction," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1439–1451, Nov. 2006.

[32] K. Bache and M. Lichman. (2013). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml