SUPPORT VECTOR REGRESSION FOR BLACK-BOX SYSTEM IDENTIFICATION

Arthur Gretton[1], *Arnaud Doucet*[2], *Ralf Herbrich*[3], *Peter J. W. Rayner*[1] *and Bernhard Schölkopf*[3]

[1] Signal Processing Group, University of Cambridge
Department of Engineering, Trumpington Street
CB2 1PZ, Cambridge, UK
{alg30,pjwr}@eng.cam.ac.uk
[2] Department of Electrical and Electronic Engineering
The University of Melbourne
Victoria 3010, Australia
a.doucet@ee.mu.oz.au
[3] Microsoft Research, Cambridge
St. George House, 1 Guildhall Street
Cambridge, CB2 3NH, UK
{rherb,bsc}@microsoft.com

ABSTRACT

In this paper, we demonstrate the use of support vector regression (SVR) techniques for black-box system identification. These methods derive from statistical learning theory, and are of great theoretical and practical interest. We briefly describe the theory underpinning SVR, and compare support vector methods with other approaches using radial basis networks. Finally, we apply SVR to modeling the behaviour of a hydraulic robot arm, and show that SVR improves on previously published results.

1. INTRODUCTION

System identification of nonlinear black-box models is a crucial but complex problem. There have been numerous recent papers in the area based on neural networks, wavelet networks, hinging hyperplanes, etc. Roughly speaking, one selects a set of regressors/basis functions, and tries to determine the number of basis/regressors and their parameters according to a given statistical criterion. Many methods are based on a penalised maximum likelihood criterion. Performing model selection and estimation is usually a difficult task, however, as it involves solving complex integration and/or optimisation problems. Gradient methods are often used, but are only guaranteed to converge toward local optima. Recently, in a Bayesian framework, Markov chain Monte Carlo algorithms have also been developed. These methods are computationally intensive, however.

We propose here an alternative approach based on support vector machines. These comprise a set of powerful tools to perform classification and regression [8], and have become very popular recently in the machine learning community. This approach, motivated by Statistical Learning Theory [10], is systematic and principled. One can list its main advantages:

- There are very few free parameters to adjust.
- Estimating the unknown parameters only involves optimisation of a convex cost function. This can be achieved using standard quadratic programming algorithms. This is *fast* and there are *no local minima.*
- The model constructed depends explicitly on the most "informative" data (the support vectors).
- It is possible to obtain theoretical bounds on the generalisation error and the sparseness of the solution (see [8]). These bounds are *independent* of the distribution generating the training and test data.

To the best of our knowledge, support vector regression (SVR) has never been used in the context of system identification, although it has been used in estimating time series by Müller *et al.* [4], and Mattera and Haykin [3]. This work differs from these previous studies in that it investigates the ν-SVR method [5], which does not require us to specify an a priori level of accuracy. We demonstrate the application of this algorithm to modeling a standard data set, and show that it is possible to obtain results that improve on current state-of-the-art methods [6], [7], with very little tuning.

2. BLACK-BOX SYSTEM IDENTIFICATION

The problem of nonlinear black-box system identification consists of conducting non-parametric regression, as described in Sjöberg *et al.* [6], [7], among others. This means that random variables $(\mathbf{x}, \mathrm{y})$, which take values in $\mathcal{X} \times \mathcal{Y}$, are generated according to a distribution $P_{\mathbf{x},\mathrm{y}}$, and we are required to estimate the *regression function* y on $\mathbf{x}$, or

$$\mathbf{E}_{\mathrm{y}}\left(\mathrm{y} | \mathbf{x} = \mathbf{x}\right) = f\left(\mathbf{x}\right).$$

We call $\mathbf{x}$ the regressor, and y the output. We further define $\mathcal{X} \triangleq \mathbb{R}^d$ and $\mathcal{Y} \triangleq \mathbb{R}$. We want to estimate $f\left(\cdot\right)$ from the training sample

$$z_N = \left(\left(\mathbf{x}_1, y_1\right), \ldots, \left(\mathbf{x}_N, y_N\right)\right) \in \left(\mathcal{X} \times \mathcal{Y}\right)^N,$$

0-7803-7011-2/01/$10.00 ©2001 IEEE

each element of which is drawn from $P_{\mathbf{x},\mathrm{y}}$. Since we do not know the mapping $f(\cdot)$, we define a learning algorithm $\mathcal{A}$, which gives us an estimate $f_{\boldsymbol{z}}(\cdot)$ of $f(\cdot)$,

$$\mathcal{A}: \bigcup_{N=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^N \rightarrow \mathcal{H}$$
$$\boldsymbol{z} \mapsto f_{\boldsymbol{z}}(\cdot),$$

within a class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ (here $\mathcal{Y}^{\mathcal{X}}$ refers to the set of functions mapping $\mathcal{X}$ to $\mathcal{Y}$), called the *hypothesis space*, which is flexible enough to model a wide range of functions. An estimate $f_{\boldsymbol{z}}(\cdot)$ associated with the *loss* $c(\mathbf{x}, y, f_{\boldsymbol{z}}(\cdot))$ is attained by minimising the *risk*,

$$f_{\boldsymbol{z}}(\cdot) = \underset{g_{\boldsymbol{z}}(\cdot)\in\mathcal{H}}{\operatorname{argmin}} \left[R(g_{\boldsymbol{z}}(\cdot)) \triangleq \mathbf{E}_{\mathbf{x},\mathrm{y}}\left[c(\mathbf{x}, \mathrm{y}, g_{\boldsymbol{z}}(\mathbf{x}))\right]\right]. \quad (1)$$

Possible loss functions include quadratic loss,

$$c(\mathbf{x}, y, g_{\boldsymbol{z}}(\cdot)) = |y - g_{\boldsymbol{z}}(\mathbf{x})|^2,$$

Vapnik's ϵ-insensitive loss [10],

$$c(\mathbf{x}, y, g_{\boldsymbol{z}}(\cdot)) = \max\{0,\ |g_{\boldsymbol{z}}(\mathbf{x}) - y| - \epsilon\},$$

and Huber loss,

$$c(\mathbf{x}, y, g_{\boldsymbol{z}}(\cdot)) = \begin{cases} \epsilon|g_{\boldsymbol{z}}(\mathbf{x}) - y| - \frac{\epsilon^2}{2} & \text{for } |g_{\boldsymbol{z}}(\mathbf{x}) - y| \geq \epsilon, \\ \frac{1}{2}|g_{\boldsymbol{z}}(\mathbf{x}) - y|^2 & \text{otherwise,} \end{cases}$$

among others.

In practice, the regression function $f_{\boldsymbol{z}}(\cdot)$ cannot readily be obtained from equation (1), since we do not usually know the distribution $\mathbf{P}_{\mathbf{x},\mathrm{y}}$. Minimising the empirical risk alone does not take into account other requirements that we would like to satisfy, such as smoothness, and can therefore result in overfitting [8], [10].

Classes of system identification problems falling within the nonlinear black-box identification framework are described in [6], [7]. These include nonlinear finite impulse response models, nonlinear autoregressive models with external input, nonlinear output error models, nonlinear autoregressive moving average, nonlinear Box-Jenkins models, etc.

3. SUPPORT VECTOR REGRESSION

We now describe how support vector machines may be used to solve the system identification problem described in the previous section. The results in this section are derived in Schölkopf *et al.* [5].

To describe the ν-SVR procedure, we must first define a mapping from the space $\mathcal{X}$ of regressors to the possibly infinite dimensional hypothesis space $\mathcal{H}$, in which an inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ is defined. We formally describe this map as

$$\boldsymbol{\Phi}: \mathcal{X} \rightarrow \mathcal{H}$$
$$\mathbf{x} \mapsto \boldsymbol{\Phi}(\mathbf{x}).$$

We choose to limit our choice of regression function $f_{\boldsymbol{z}}(\cdot)$ to the class of functions which can be expressed as inner products in $\mathcal{H}$,

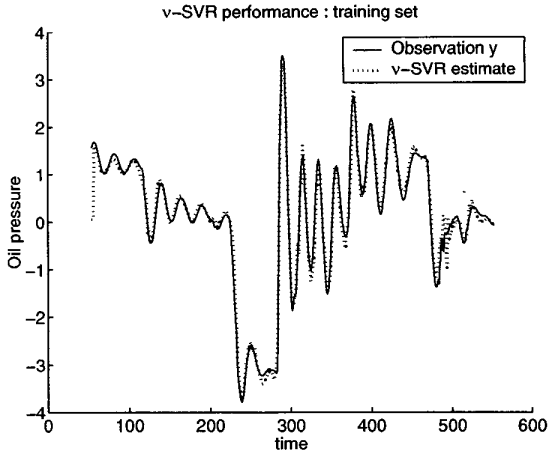

Fig. 1. Robot arm data and ν-SVR model : Training set and model approximation.

taken between some *weight vector* $\mathbf{w}$ and the mapped regressor $\boldsymbol{\Phi}(\mathbf{x})$;

$$f_{\boldsymbol{z}}(\mathbf{x}) = \langle \mathbf{w}, \boldsymbol{\Phi}(\mathbf{x})\rangle_{\mathcal{H}} + b. \quad (2)$$

The regression function in the hypothesis space is consequently linear, and thus the *nonlinear* regression problem of estimating $f_{\boldsymbol{z}}(\mathbf{x})$ has become a *linear* regression problem in the hypothesis space $\mathcal{H}$. Note that the mapping $\boldsymbol{\Phi}(\cdot)$ need never be computed explicitly; instead, we use the fact that if $\mathcal{H}$ is the reproducing kernel Hilbert space induced by $k(\cdot,\cdot)$, then writing $\boldsymbol{\Phi}(\mathbf{x}) = k(\mathbf{x},\cdot)$, we get

$$\langle \boldsymbol{\Phi}(\mathbf{x}_i), \boldsymbol{\Phi}(\mathbf{x}_j)\rangle_{\mathcal{H}} = k(\mathbf{x}_i, \mathbf{x}_j).$$

The latter requirement is met for kernels fulfilling the Mercer conditions [8]. These conditions are satisfied for a wide range of kernels, including Gaussian radial basis functions (see equation (6)). We emphasise that the feature space need never be defined explicitly, since only the kernel is used in SVR algorithms. Indeed, it is possible for multiple feature spaces to be induced by a single kernel.

We now describe the optimisation problem to be undertaken in finding $f_{\boldsymbol{z}}(\cdot)$. All support vector regression methods involve the minimisation of a regularised risk functional, which represents a tradeoff between smoothness and training error (the latter is determined by the cost functional). In the case of the ν-SVR method, the regularised risk $R^{\epsilon}_{\mathrm{emp}}(f_{\boldsymbol{z}}, \boldsymbol{z})$ at the optimum is given by

$$\min_{\mathbf{w},b,\epsilon}\left[R^{\epsilon}_{\mathrm{reg}}(f_{\boldsymbol{z}}(\cdot), \boldsymbol{z})\right] = \min_{\mathbf{w},b,\epsilon}\left[\frac{1}{2}\|\mathbf{w}\|^2_{\mathcal{H}} + C\left(\nu\epsilon + R^{\epsilon}_{\mathrm{emp}}(f_{\boldsymbol{z}}(\cdot), \boldsymbol{z})\right)\right], \quad (3)$$

where we use the Vapnik ϵ-insensitive loss in the empirical risk;

$$R^{\epsilon}_{\mathrm{emp}}(f_{\boldsymbol{z}}(\cdot), \boldsymbol{z}) = \frac{1}{N}\sum_{i=1}^{N} c(\mathbf{x}_i, y_i, f_{\boldsymbol{z}}(\cdot)) = \frac{1}{N}\sum_{i=1}^{N} \xi_i + \xi_i^*,$$

in which

$$\xi_i = \max\{0,\ f_{\boldsymbol{z}}(\mathbf{x}) - y - \epsilon\} \quad \text{and}$$
$$\xi_i^* = \max\{0,\ -f_{\boldsymbol{z}}(\mathbf{x}) + y - \epsilon\}.$$

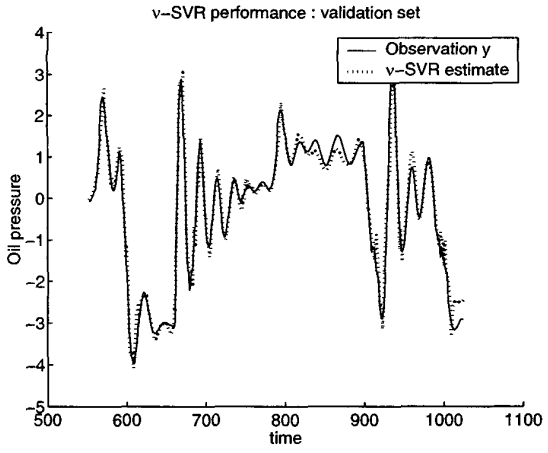

Fig. 2. Robot arm data and ν-SVR model : Validation set and model approximation.

All training points $(\mathbf{x}_i, y_i)$ for which $|f_{\mathbf{z}}(\mathbf{x}_i) - y_i| \geq \epsilon$ are known as *support vectors*; it is only these points that determine $f_{\mathbf{z}}(\cdot)$. Note that other loss functions, such as the Huber loss, can also be used in support vector regression, although not all loss functions result in a sparse representation. The terms C and ν in equation (3) specify the tradeoff between model simplicity, the size of the parameter ϵ below which the loss is zero, and the total empirical loss over the training set, $R^{\epsilon}_{\text{emp}}(\cdot)$. Schölkopf *et al.* [5] describe the theoretical behaviour of ν and C in more detail.

It can be shown [5] that the component $\mathbf{w}$ in equation (2) is a linear combination of the mapped training points,

$$\mathbf{w} = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) \mathbf{\Phi}(\mathbf{x}_i), \tag{4}$$

and that solving equation (3) is equivalent to finding

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} \; -\frac{1}{2} \sum_{i,j=1}^{N} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{N} y_i (\alpha_i^* - \alpha_i), \tag{5}$$

subject to

$$\sum_{i=1}^{N} (\alpha_i - \alpha_i^*) = 0, \qquad \alpha_i, \alpha_i^* \in \left[0, \frac{C}{N}\right],$$

$$\sum_{i=1}^{N} (\alpha_i + \alpha_i^*) \leq C\nu.$$

There exist a number of methods that can be used to solve this quadratic programming problem. Our results were obtained using the LOQO algorithm in Vanderbei [9]. In the case of large training sets, data decomposition methods exist to speed convergence; see

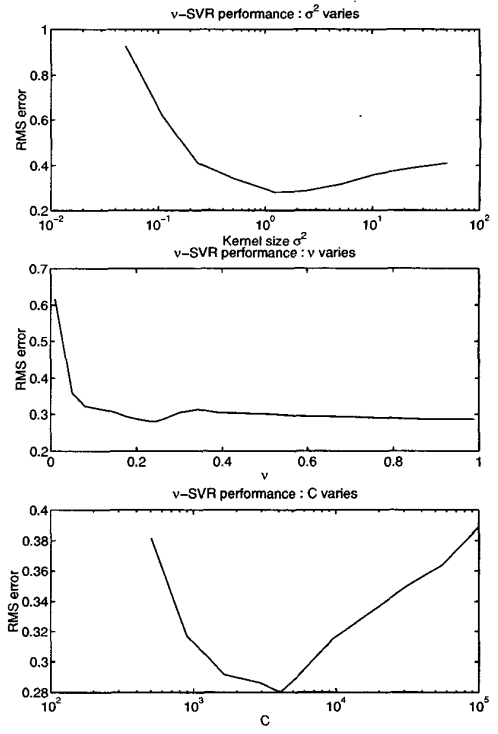

Fig. 3. RMS error variation with ν, σ^2 and C. In each case, the fixed parameters take their optimal values.

e.g. Chang *et al.* [2]. The offset b is found using

$$\langle \mathbf{w}, \mathbf{\Phi}(\mathbf{x}_j) \rangle_{\mathcal{H}} + b - y_j = \epsilon \quad \text{when } \alpha_j \in \left(0, \frac{C}{m}\right),$$

$$y_j - \langle \mathbf{w}, \mathbf{\Phi}(\mathbf{x}_j) \rangle_{\mathcal{H}} - b = \epsilon \quad \text{when } \alpha_j^* \in \left(0, \frac{C}{m}\right);$$

the set of equations thus obtained can be solved via linear least squares.

4. COMPARISON WITH STANDARD RBF APPROACHES

A popular set of regression functions are the radial basis functions. The radial basis function expansion is

$$f_{\mathbf{z}}(\mathbf{x}) = \sum_{i=0}^{M} w_j k_j(\boldsymbol{\mu}_j, \mathbf{x}),$$

where M is the number of radial basis functions used (this need not be the same as the number N of training points), $w_j \in \mathbb{R}$ scale the various basis functions $k_j(\boldsymbol{\mu}_j, \cdot)$, w_0 scales the constant offset term $k_0(\boldsymbol{\mu}_0, \cdot) \triangleq 1$, and each basis function $k_j(\boldsymbol{\mu}_j, \cdot)$ has an individual centre parameter $\boldsymbol{\mu}_j$ and width parameter σ_j. For instance, in the case of Gaussian radial basis networks,

$$k_j(\boldsymbol{\mu}_j, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right). \tag{6}$$

It is also possible to use a more general covariance matrix, rather than σ_j^2; this results in a greater number of parameters that require adjustment.

It is clear that SVR methods in fact produce radial basis function networks, with all width parameters σ_j^2 set at the same value, and centres $\boldsymbol{\mu}_j$ corresponding to support vectors $\mathbf{x}_j$ (thus M is the number of support vectors). As discussed previously, the SVR training procedure selects the training points to be used in this expansion so as to avoid overfitting, and to achieve sparseness with regards to the training data. Furthermore, the attendant optimisation process is convex, and has a single optimum.

There is a great deal of literature, past and present, on methods for training radial basis function networks; see for instance Bishop [1]. Without going into detail, it is fairly common practice to centre the basis functions on the training data and fix the basis width a priori, as in SVR. Model selection (determining the number of non-null weight vector components $\mathbf{w}_i$) and parameter estimation (estimating the values of the $\mathbf{w}_i$) in traditional radial basis function network methods, however, are usually based on Bayesian/penalized maximum likelihood approaches; the associated optimisation problems are often non-convex and possess multiple local minima, which can lead to greater computational complexity.

5. EXPERIMENTAL RESULTS

In the following experiments, we make use of a Gaussian radial basis function kernel, as described in equation (6), with kernel width σ^2 (note that other kernel options, such as polynomial kernels or sigmoid kernels, could also be used). As we perform SV regression, the kernel centres are set at the training point locations $\mathbf{x}_j$. We apply the ν-SVR algorithm to modeling behaviour of a hydraulic robot arm; our result will be compared with the neural network NARX and wavelet network NARX models in Sjöberg *et al* [6]. The input u_t represents the size of the valve through which oil flows into the actuator, and the output y_t is a measure of oil pressure (the latter determines the arm position). For the purpose of comparison, we used the regressor

$$\mathbf{x}_t = \begin{bmatrix} y_{t-1} & y_{t-2} & y_{t-3} & u_{t-1} & u_{t-2} \end{bmatrix}^T,$$

since this is also used by Sjöberg *et al*. We also used half the data set for training, and half as validation data, again following the procedure of Sjöberg *et al*. The kernel width was set at $\sigma^2 = 1.2242$, and we used the ν-SVR parameters $\nu = 0.2444$ and $C = 4.07 \times 10^3$. It must be emphasised that the experimental outcome varies little for a wide range of parameter values; see figure 3. Note also that prior knowledge of the observation noise would allow us to select a value of ν that is asymptotically optimal in the number of data [8].

The ν-SVR model output on the training data is given in figure 1, and the model output on the validation data in figure 2. The RMS error of this prediction on the validation set is 0.280, which is lower than both the wavelet network RMS error (0.579), and the prediction made by a one-hidden-layer sigmoid neural network with ten hidden units (0.467). Although Sjöberg *et al.* were able to further reduce the RMS error to 0.328 on this data set, this required assumptions regarding the model structure not made in our algorithm. Further advantages of the ν-SVR solution include simplicity, computational efficiency, robustness in the face of decreased training set size, and ease of tuning, due to the low sensitivity of the solution to changes in σ^2, ν and C. Our implementation of the ν-SVR algorithm required 56 lines of Matlab code (excluding the standard quadratic programming component), and took 193 seconds to train on the data set in figure 1, using a Pentium III processor running at 500MHz.

6. CONCLUSION

In this study, we describe the important theoretical and practical advantages of support vector regression for back box system identification. The simplicity of implementation, coupled with good performance in both this and other studies on time series prediction, make SVR methods an attractive alternative to standard system identification techniques.

7. REFERENCES

[1] C. Bishop. *Neural Networks for Pattern Recognition*, chapter 5. Oxford University Press, Oxford, 1995.

[2] C.-C. Chang, C.-W. Hsu, and C.-J. Lin. The analysis of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 11(4):1003–1008, 2000.

[3] D. Mattera and S. Haykin. Support vector machines for dynamic reconstruction of a chaotic system. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[4] K.R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Using support vector machines for time series prediction. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[5] B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

[6] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, 1995.

[7] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: Mathematical foundations. *Automatica*, 31(12):1725–1750, 1995.

[8] A. Smola and B. Schölkopf. *Learning with Kernels*. MIT press, To appear.

[9] R. J. Vanderbei. LOQO: An interior point code for quadratic programming. Technical Report TR SOR-94-15, Department of Civil Engineering and Operations Research, Princeton University, 1995.

[10] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, N.Y., 1995.