Orthogonal Least Squares Regression with Tunable Kernels

S. Chen, X.X. Wang and D.J. Brown

Abstract

A novel technique is proposed to construct sparse regression models based on the orthogonal least squares method with tunable kernels. The proposed technique tunes the centre vector and diagonal covariance matrix of individual regressor by incrementally minimising the training mean square error using a guided random search algorithm, and it offers a state-of-the-art method for constructing very sparse models that generalise well.

I. Introduction

A basic principle in practical nonlinear data modelling is the parsimonious principle of ensuring the smallest possible model that explains the training data. The existing sparse kernel modelling methods [1]–[6] place kernel centres at the training input data and adopt a single common variance for all the kernel regressors. We present a flexible construction method for parsimonious regression modelling. The proposed algorithm tunes the centre vector and diagonal covariance matrix of individual regressor by incrementally minimising the training mean square error (MSE) in an orthogonal forward selection procedure using a guided random search algorithm, called the repeated weighted boosting search (RWBS) [7]. This novel orthogonal least squares (OLS) algorithm with tunable kernels is capable of producing very sparse models that generalise well.

II. Orthogonal least squares with tunable kernels

Consider approximating the N pairs of training data $\{\mathbf{x}_l, y_l\}_{l=1}^N$ with the regression model

$$y(\mathbf{x}) = \hat{y}(\mathbf{x}) + e(\mathbf{x}) = \sum_{i=1}^{M} w_i g_i(\mathbf{x}) + e(\mathbf{x}) \tag{1}$$

where $\mathbf{x}$ is the m-dimensional input variable, $y(\mathbf{x})$ the desired output, $\hat{y}(\mathbf{x})$ the model output, and $e(\mathbf{x})$ the modelling error at $\mathbf{x}$; w_i, $1 \leq i \leq M$, denote the model weights, M is the number of regressors, and $g_i(\bullet)$, $1 \leq i \leq M$, denote the regressors. The general Gaussian kernel function

$$g_i(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)} \tag{2}$$

S. Chen is with School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.
X.X. Wang is with Neural Computing Research Group, Aston University, Birmingham B4 7ET, U.K.
D.J. Brown is with Department of Creative Technologies, University of Portsmouth, Portsmouth PO1 3HE, U.K.

is adopted, where $\boldsymbol{\mu}_i$ is the ith kernel centre and $\boldsymbol{\Sigma}_i$ the diagonal covariance matrix of the ith regressor. By defining $\mathbf{y} = [y_1\ y_2 \cdots y_N]^T$, $\mathbf{e} = [e(\mathbf{x}_1)\ e(\mathbf{x}_2) \cdots e(\mathbf{x}_N)]^T$, $\mathbf{w} = [w_1\ w_2 \cdots w_M]^T$, and

$$\mathbf{G} = [\mathbf{g}_1\ \mathbf{g}_2 \cdots \mathbf{g}_M] \quad \text{with} \quad \mathbf{g}_k = [g_k(\mathbf{x}_1)\ g_k(\mathbf{x}_2) \cdots g_k(\mathbf{x}_N)]^T \tag{3}$$

the regression model (1) over the training data set can be written in the matrix form

$$\mathbf{y} = \mathbf{G}\mathbf{w} + \mathbf{e} \tag{4}$$

Let an orthogonal decomposition of the regression matrix $\mathbf{G}$ be $\mathbf{G} = \mathbf{P}\mathbf{A}$, where $\mathbf{A}$ is an upper triangular matrix with the unit diagonal elements, and $\mathbf{P} = [\mathbf{p}_1\ \mathbf{p}_2 \cdots \mathbf{p}_M]$ with the orthogonal columns that satisfy $\mathbf{p}_i^T \mathbf{p}_j = 0$, if $i \neq j$. The regression model (4) can alternatively be expressed as

$$\mathbf{y} = \mathbf{P}\boldsymbol{\theta} + \mathbf{e} \tag{5}$$

where the new weight vector $\boldsymbol{\theta} = [\theta_1\ \theta_2 \cdots \theta_M]^T$ satisfy the triangular system $\mathbf{A}\mathbf{w} = \boldsymbol{\theta}$.

For the orthogonal regression model (5), the training MSE can be expressed as $J = \mathbf{e}^T\mathbf{e}/N = \mathbf{y}^T\mathbf{y}/N - \sum_{i=1}^{M} \mathbf{p}_i^T \mathbf{p}_i \theta_i^2/N$. Thus the training MSE for the k-term subset model can be expressed recursively as

$$J_k = J_{k-1} - \frac{1}{N}\mathbf{p}_k^T \mathbf{p}_k \theta_k^2 \tag{6}$$

where $J_0 = \mathbf{y}^T\mathbf{y}/N$. At the kth stage of regression, the kth regressor is determined by maximising the error reduction criterion

$$\mathrm{ER}_k(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{N}\mathbf{p}_k^T \mathbf{p}_k \theta_k^2 \tag{7}$$

with respect to the kernel centre $\boldsymbol{\mu}_k$ and its diagonal covariance matrix $\boldsymbol{\Sigma}_k$. The orthogonal forward selection procedure is terminated at the kth stage if $J_k < \xi$ is satisfied, where the small positive scalar ξ is a chosen tolerance. This produces a parsimonious model containing k significant regressors.

The task of determining the kth regressor is performed using the RWBS algorithm, which is a simple yet effective global search optimisation algorithm [7]. Define P_S – population size, N_G – number of generations in the repeated search, ξ_B – accuracy for terminating the weighted boosting search, and/or N_B – maximum number of iterations in the weighted boosting search. Let the vector $\mathbf{u}$ contain $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. The algorithm is summarised.

Outer loop: generations For $n = 1 : N_G$

Generation initialisation: Initialise the population by setting $\mathbf{u}_1^{(n)} = \mathbf{u}_{\text{best}}^{(n-1)}$ and randomly generating rest of the population members $\mathbf{u}_i^{(n)}$, $2 \leq i \leq P_S$, where $\mathbf{u}_{\text{best}}^{(n-1)}$ denotes the solution found in the previous generation. If $n = 1$, $\mathbf{u}_1^{(n)}$ is also randomly chosen.

Weighted boosting search initialisation: Assign the initial distribution weightings $\delta_i(0) = \frac{1}{P_S}$, $1 \leq i \leq P_S$, for the population. Then

1. For $1 \leq i \leq P_S$, generate $\mathbf{g}_k^{(i)}$ from $\mathbf{u}_i^{(n)}$, the candidates for the kth model column, and orthogonalise them:

$$\alpha_{j,k}^{(i)} = \frac{\mathbf{p}_j^T \mathbf{g}_k^{(i)}}{\mathbf{p}_j^T \mathbf{p}_j},\ 1 \leq j < k \tag{8}$$

$$\mathbf{p}_k^{(i)} = \mathbf{g}_k^{(i)} - \sum_{j=1}^{k-1} \alpha_{j,k}^{(i)} \mathbf{p}_j \tag{9}$$

2. For $1 \leq i \leq P_S$, calculate the cost function value of each $\mathbf{u}_i^{(n)}$:

$$\theta_k^{(i)} = \frac{\left(\mathbf{p}_k^{(i)}\right)^T \mathbf{y}}{\left(\mathbf{p}_k^{(i)}\right)^T \mathbf{p}_k^{(i)}} \tag{10}$$

$$J_k^{(i)} = J_{k-1} - \frac{1}{N} \left(\mathbf{p}_k^{(i)}\right)^T \mathbf{p}_k^{(i)} \left(\theta_k^{(i)}\right)^2 \tag{11}$$

Inner loop: weighted boosting search For $t = 1 : N_B$

Step 1: Boosting

1. Find

$$i_{\text{best}} = \arg \min_{1 \leq i \leq P_S} J_k^{(i)} \quad \text{and} \quad i_{\text{worst}} = \arg \max_{1 \leq i \leq P_S} J_k^{(i)}$$

Denote $\mathbf{u}_{\text{best}}^{(n)} = \mathbf{u}_{i_{\text{best}}}^{(n)}$ and $\mathbf{u}_{\text{worst}}^{(n)} = \mathbf{u}_{i_{\text{worst}}}^{(n)}$.

2. Normalise the cost function values

$$\bar{J}_k^{(i)} = \frac{J_k^{(i)}}{\sum_{l=1}^{P_S} J_k^{(l)}},\ 1 \leq i \leq P_S$$

3. Compute a weighting factor β_t according to

$$\eta_t = \sum_{i=1}^{P_S} \delta_i(t-1) \bar{J}_k^{(i)},\ \beta_t = \frac{\eta_t}{1 - \eta_t}$$

4. Update the distribution weightings for $1 \leq i \leq P_S$

$$\delta_i(t) = \begin{cases} \delta_i(t-1) \beta_t^{\bar{J}_k^{(i)}}, & \text{for } \beta_t \leq 1 \\ \delta_i(t-1) \beta_t^{1 - \bar{J}_k^{(i)}}, & \text{for } \beta_t > 1 \end{cases}$$

and normalise them

$$\delta_i(t) = \frac{\delta_i(t)}{\sum_{l=1}^{P_S} \delta_l(t)},\ 1 \leq i \leq P_S$$

Step 2: Parameter updating

1. Construct the $(P_S + 1)$th point using

$$\mathbf{u}_{P_S+1} = \sum_{i=1}^{P_S} \delta_i(t) \mathbf{u}_i^{(n)}$$

2. Construct the $(P_S + 2)$th point using $\mathbf{u}_{P_S+2} = \mathbf{u}_{\text{best}}^{(n)} + \left(\mathbf{u}_{\text{best}}^{(n)} - \mathbf{u}_{P_S+1}\right)$

3. Calculate $\mathbf{g}_k^{(P_S+1)}$ and $\mathbf{g}_k^{(P_S+2)}$ from $\mathbf{u}_{P_S+1}$ and $\mathbf{u}_{P_S+2}$, orthogonalise these two candidate model columns (as in (8) and (9)), and compute their corresponding cost function values $J_k^{(i)}$, $i = P_S + 1, P_S + 2$ (as in (10) and (11)). Find

$$i_* = \arg \min_{i=P_S+1,P_S+2} J_k^{(i)}$$

4. The pair $(\mathbf{u}_{i_*}, J_k^{(i_*)})$ then replaces $(\mathbf{u}_{\rm worst}^{(n)}, J_k^{(i_{\rm worst})})$ in the population

If $\|\mathbf{u}_{P_S+1} - \mathbf{u}_{P_S+2}\| < \xi_B$, exit **inner loop**.

End of inner loop

The solution found in the nth generation is $\mathbf{u} = \mathbf{u}_{\rm best}^{(n)}$.

End of outer loop

This yields the solution $\mathbf{u} = \mathbf{u}_{\rm best}^{(N_G)}$, i.e. $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ of the kth regressor, the kth model column $\mathbf{g}_k$, the orthogonalisation coefficients $\alpha_{j,k}$, $1 \le j < k$, as well as the corresponding orthogonal model column $\mathbf{p}_k$, the weight θ_k and the MSE of the k-term model J_k.

III. A MODELLING EXAMPLE

We considered constructing a model representing the relationship between the fuel rack position (input $u(t)$) and the engine speed (output $y(t)$) for a Leyland TL11 turbocharged, direct injection diesel engine operated at low engine speed [8]. The input-output data set contained 410 samples. The first 210 data points were used in training and the last 200 points in model validation. The previous study [3] has shown that this data set can be modeled adequately as $y_i = f_s(\mathbf{x}_i) + \epsilon_i$, where $y_i = y(i)$, $\mathbf{x}_i = [y(i-1)\ u(i-1)\ u(i-2)]^T$, $f_s(\bullet)$ describes the unknown underlying system to be identified and $\epsilon_i = \epsilon(i)$ denotes the system noise. The proposed OLS algorithm with tunable kernels constructed a 6-term generalised Gaussian model as listed in Table I. The MSE of this constructed 6-term model over the test data set was 0.000563. We also applied the support vector machine (SVM) algorithm [6] to construct a sparse kernel model for this data set. To achieve a similar generalisation performance, the SVM algorithm required a Gaussian kernel model of 70 support vectors. We also point out that the recorded training time for the OLS algorithm with tunable kernels was 60 times faster than that of the SVM algorithm.

REFERENCES

[1] S. Chen, S.A. Billings and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *Int. J. Control*, Vol.50, No.5, pp.1873–1896, 1989.

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

[3] S. Chen, X. Hong and C.J. Harris, "Sparse kernel regression modelling using combined locally regularized orthogonal least squares and D-optimality experimental design," *IEEE Trans. Automatic Control*, Vol.48, No.6, pp.1029–1036, 2003.

[4] B. Schlkopf and A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.

[5] M.E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Machine Learning Research*, Vol.1, pp.211–244, 2001.

[6] S. Gunn, "Support vector machines for classification and regression," *Technical Report*, ISIS Research Group, Department of Electronics and Computer Science, University of Southampton, UK, May 1998.

[7] Chen, S., Wang, X.X., and Harris, C.J., "Experiments with repeating weighted boosting search for optimization in signal processing applications," *IEEE Trans. Systems, Man and Cybernetics, Part B*, to appear, June 2005.

[8] S.A. Billings, S. Chen and R.J. Backhouse, "The identification of linear and non-linear models of a turbocharged automotive diesel engine," *Mechanical Systems and Signal Processing*, Vol.3, No.2, pp.123–142, 1989.

TABLE I

OLS WITH TUNABLE KERNELS FOR MODELLING THE ENGINE DATA SET.

regression step k	centre vector $\boldsymbol{\mu}_k$			diagonal covariance $\boldsymbol{\Sigma}_k$			weight w_k	MSE $J_k \times 100$
0	–			–			–	1558.9
1	5.2219	5.5839	5.6416	7.3532	21.0894	22.4661	6.0396	0.3866
2	4.2542	5.2741	4.1028	1.8680	10.0863	49.8826	-1.2845	0.1311
3	3.8826	5.1707	6.3200	0.1600	0.1600	64.0000	-0.1539	0.0996
4	2.3154	3.2544	5.4897	0.9447	0.3329	11.7564	-0.1433	0.0913
5	4.0673	4.4276	3.5963	0.1608	18.3731	0.2207	0.1945	0.0740
6	2.3663	3.2377	5.1376	0.1754	0.9317	0.1600	0.9658	0.0547