Chaotic time-series prediction using resource-allocating RBF networks *

Roman Rosipal, Miloš Koska, Igor Farkaš
Institute of Measurement Science
Slovak Academy of Sciences
Dúbravská cesta 9, 842 19 Bratislava, Slovakia
e-mail: {rosipal, koska, farkas}@neuro.savba.sk

Abstract

One of the main problems associated with artificial neural networks on-line learning methods is the estimation of model order. In this paper, we report about a new approach to constructing a resource-allocating network exploiting weights adaptation using QRD-based recursive least-squares technique. Further, we studied the performance of Dynamic Cell Structures algorithm for on-line adaptation of centers positions. The proposed method was tested on the task of Mackey-Glass time-series prediction. Order of resulting networks and their prediction abilities were superior to those previously reported by Platt [6].

1 Introduction

The resource-allocating network (RAN) was introduced by Platt [6] and further extended by McLachlan and Lowe [5]. Similar approach was also outlined in [1]. RANs were usually based on radial basis function (RBF) networks. Associated with these, two essential problems – weights adaptation and center selection – need to be solved.

Our approach to this task can be characterized by the following features:

- adaptation of the output layer weights using QR decomposition (QRD) algorithm for recursive least-squares estimation [4],
- center selection using Platt's method,
- on-line adaptation of the positions of existing centers using modified Dynamic Cell Structures (DCS) algorithm, originally proposed by Bruske and Sommer [2].

Generally, RAN allocates far fewer centers than the number of presented examples, but it can lead to exaggerated number of centers in the case of long period data sequence [6]. Our modification prevents this undesirable effect and thus holds complexity of the network on the "low" level.

* This work was partially supported by Slovak Grant Agency for Science (grants No. 2/2040/95 and 95/5305/468).

2 Methods

2.1 RBF network

A two-layer RBF network implements a mapping $\hat{y} : R^n \mapsto R$ according to

$$\hat{y}(\mathbf{x}) = \beta_0 + \sum_{i=1}^{n_r} \beta_i \phi_i(\|\mathbf{x} - \mathbf{c}_i\|/h_i)$$

where $\mathbf{x} \in R^n$ is an input vector, $\phi_i(.)$ is the transfer function, h_i is the i-th center width, $\|.\|$ denotes the Euclidean norm, $\beta_i \in R$ are the weights, $\mathbf{c}_i \in R^n$ represent the positions of RBF centers, and n_r is the number of centers. In the same way as in Platt's algorithm, the polynomial approximation of the Gaussian transfer function was used.

2.2 Platt's algorithm

The Platt's method can be described with the following pseudo-code:

$\delta_0 = \delta_{max}$, $\beta_0 = y_0$, $\mathbf{c}_1 = \mathbf{x}_1$, $\beta_1 = y_1 - y_0 = e_1$, $h_1 = \kappa\delta_0$
$\delta_1 = \delta_0 \exp(-1/\tau)$
`for j = 2 to number of iterations {`
    `evaluate output of network` $\hat{y}_j = \sum_{i=1}^{n_r} \beta_i \phi_i(\mathbf{x}_j) + \beta_0$
    `compute error` $e_j = y_j - \hat{y}_j$
    `find distance to nearest center` $d = min_i \|\mathbf{c}_i - \mathbf{x}_j\|$
    `if` $|e_j| > \epsilon$ `and` $d > \delta_j$
        `allocate new unit:` $n_r = n_r + 1$, $\mathbf{c}_{n_r} = \mathbf{x}_j$, $\beta_{n_r} = e_j$, $h_{n_r} = \kappa\delta_j$
    `else`
        `Update`$(\beta_0, \{\beta_i, \mathbf{c}_i\}_{i=1}^{n_r})$
    `if` $\delta_j > \delta_{min}$
        $\delta_{j+1} = \delta_j \exp(-1/\tau)$
    `else`
        $\delta_{j+1} = \delta_j$
`}`

The network was initialized with the first two input-output pairs. The condition for allocating a new center in step j exploits two criteria. The first criterion was based on prediction error $|e_j| = |y_j - \hat{y}_j|$ (y_j was desired output). This was compared with the critical value ϵ. The second criterion was satisfied if the distance of input $\mathbf{x}$ from the nearest center $\mathbf{c}$ was larger than the critical scale resolution δ_j. The learning started at largest scale of resolution, i.e. $\delta_0 = \delta_{max}$ and was shrunk during N time steps until it reached the smallest value δ_{min}. The parameter τ (decay constant) was thus evaluated from number of steps N.

The LMS algorithm was used to update weights

$$\Delta\beta_i = \alpha e_j \phi_i(\mathbf{x}_j) \qquad \Delta\beta_0 = \alpha e_j \ ,$$

and centers

$$\Delta\mathbf{c}_i = 2\frac{\alpha}{(h_i)^2}(\mathbf{x}_j - \mathbf{c}_i)\phi_i(\mathbf{x}_j) e_j \beta_i \ , \tag{1}$$

where α is the learning factor.

2.3 Weights adaptation

Our first modification of Platt's algorithm consists in using different algorithm for estimation of weights $\{\beta_i\}_{i=0}^{n_r}$. Instead of the LMS algorithm we used QRD algorithm with square-root-free Givens rotations (GQRD) [4]. In GQRD algorithm, the forgetting factor η was adjusted at each time step according to $\eta(j) = \eta_0\eta(j-1) + 1 - \eta_0$. After adding a new center $\eta = \eta(0)$ and in order to re-adjust the network, further center addition was not allowed for next T time-steps. It is necessary to note that in the case of non-stationary data, the application of the forgetting factor might improve estimate of weights but convergence is not guaranteed.

The similar method of weights adaptation, based on extended Kalman filter (EKF) algorithm, was recently applied to RANs by McLachlan and Lowe [5]. Comparison of GQRD and EKF can be found in [4].

2.4 Dynamic Cell Structures

The second modification we have studied consists in using modified DCS algorithm for on-line adaptation of positions of existing centers $\{\mathbf{c}_i\}_{i=1}^{n_r}$. DCS employ a modified Kohonen learning rule in conjunction with competitive Hebbian learning. Hebbian learning is exploited to establish lateral connection between centers with the goal to reflect the topology of the approximated manifold. Symmetric lateral connections was assumed. A new center was connected with the center nearest to the presented input with weight equal to 1. The Kohonen-like learning rule served to adjust the position of centers

$$\Delta\mathbf{c}_w = \sigma_1(\mathbf{x}_j - \mathbf{c}_w) \qquad \Delta\mathbf{c}_i = \sigma_2 A_{wi}(\mathbf{x}_j - \mathbf{c}_i) \quad \text{for} \quad i \in Nh(w)\ , \tag{2}$$

where σ_1, σ_2 are the learning factors, A_{wi} is the symmetric real-valued matrix of lateral connections and $Nh(w) = \{i \mid (A_{wi} \neq 0, 1 \leq i \leq n_r)\}$ is the neighborhood of a center $\mathbf{c}_w$, closest to presented input $\mathbf{x}_j$. We also investigated modification of this rule derived from (1):

$$\Delta\mathbf{c}_w = \frac{2\sigma_1}{(h_w)^2}(\mathbf{x}_j - \mathbf{c}_w)\phi_w(\mathbf{x}_j)e_j\beta_w \qquad \Delta\mathbf{c}_i = \frac{2\sigma_2}{(h_i)^2}A_{wi}(\mathbf{x}_j - \mathbf{c}_i)\phi_i(\mathbf{x}_j)e_j\beta_i \quad \text{for } i \in Nh(w). \tag{3}$$

Both the update of matrix A and the adjustment of the centers position were done in procedure `Update()` (see pseudo-code of Platt's algorithm).

2.5 Time-series prediction

RAN was applied to prediction of chaotic Mackey-Glass time series (available from CMU Learning Benchmark Archive [3]). Network was trained to predict the value at time $t + 85$, from inputs at time t, $t - 6$, $t - 12$, and $t - 18$. The first 3103 generated data points were presented to the network with the following values of parameters: $\kappa = 0.87$, $T = 30$, $\eta(0) = 0.9$, $\eta_0 = 0.99$, $\alpha = 0.05$, $N = 2500$, $\delta_{max} = 0.7$, $\delta_{min} = 0.07$, $\sigma_1 = 0.01$, $\sigma_2 = 0.001$.

3 Results

The quality of prediction was evaluated in terms of normalized root mean square error (NRMSE) and number of inserted centers (NIC) (Table 1). One can see, that using (B) decreased NRMSE by 30% on average compared to "pure" Platt's approach (A). Moreover, improvement was obtained also in terms of NIC which significantly decreased in two cases,

$\epsilon = 0.1$ and 0.05. On the whole, the best results were achieved with method (C). NIC for (C) was 1.8 times lower on average than for (B) and NRMSE was smaller for (C) as well. Both (D) and (E) slightly decreased NIC compared to (C), but, on the other hand, NRMSE increased in both cases, especially in (D).

	A		B		C		D		E	
	LMS LMS 1		GQRD LMS 1		GQRD LMS 30		GQRD DCS2 30		GQRD DCS3 30	
ϵ	NIC	NRMSE	NIC	NRMSE	NIC	NRMSE	NIC	NRMSE	NIC	NRMSE
0.01	114	0.395	113	0.258	52	0.242	47	0.318	48	0.266
0.02	113	0.402	90	0.260	46	0.241	41	0.318	44	0.268
0.05	95	0.383	48	0.272	32	0.250	31	0.362	31	0.302
0.10	67	0.388	28	0.294	20	0.285	22	0.399	19	0.346

Table 1: Comparison of achieved results. The items in the "triplets" on the second line relate to the methods for weights adaptation, center adaptation and the length of window during which the next center was not allowed to be inserted. DCS2 and DCS3 represent the methods based on DCS algorithm associated with equation (2) and (3), respectively.

4 Discussion

In this paper we report about a new method for constructing RANs. Developed networks were applied to chaotic time-series prediction. Using NRMSE and NIC as criteria for model evaluation, we found out that our modifications with GQRD method provided results that were superior to those reported by Platt. We also tried to use DCS as a different method for approximation of the input data manifold. Although this approach performs well in off-line learning [2], we didn't find any significant improvement in on-line case. We made several investigations with a different on-line version of DCS [1] and got the same results as with DCS described in this paper. As these algorithms strongly depend on parameters setting, we think that more experiments have to be done to make conclusions.

Although we achieved the improvement in on-line model order estimation, this question still remains open. We hypothesize that DCS approach can lead to even more "considerate" strategy of inserting centers, resulting in lower network complexity with the same accuracy.

References

[1] I. Ahrns, J. Bruske, and G. Sommer. On-line learning with dynamic cell structures. In *Proceedings of ICANN'95*, volume 2, pages 141–146, 1995.

[2] J. Bruske and G. Sommer. Dynamic Cell Structure learns perfectly topology preserving map. *Neural Computation*, 7:845–865, 1995.

[3] `http://legend.gwydion.cs.cmu.edu/neural-bench/benchmarks/mackey-glass.html`.

[4] N. Kalouptsidis and S. Theodoridis. *Adaptive system identification and signal processing algorithms*. Prentice Hall, 1993.

[5] A. McLachlan and D. Lowe. Tracking of non-stationary time-series using resource allocating RBF networks. In R. Trappl, editor, *Cybernetics and Systems '96. Proceedings of the 13th European Meeting on Cybernetics and Systems Research.*, pages 1066–1071, 1996.

[6] C.J. Platt. A resource allocating network for function interpolation. *Neural Computation*, 3(2):213–225, 1991.