Prediction of Chaotic Time-Series with a Resource-Allocating RBF Network

Roman Rosipal, Miloš Koska and Igor Farkaš
Institute of Measurement Science
Slovak Academy of Sciences
Dúbravská cesta 9,
842 19 Bratislava, Slovakia
tel.: ++421-7-378 2839, 378 2518
fax.: ++421-7-375 943
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 radial basis function network exploiting weights adaptation using recursive least-squares technique based on Givens QR decomposition. Further, we study the performance of pruning strategy we introduced to obtain the same prediction accuracy of the network with lower model order. The proposed methods were tested on the task of Mackey-Glass time-series prediction. Order of resulting networks and their prediction performance were superior to those previously reported by Platt [12].

Key words: Givens QR decomposition, on-line learning, resource-allocating RBF network, time-series prediction

1. Introduction

The resource-allocating network (RAN) was introduced by Platt [12] and further extended by Kadirkamanathan and Niranjan [7], McLachlan and Lowe [11]. Since RANs are based on radial basis function (RBF) networks, two essential problems – weights adaptation and center selection – need to be solved.

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

- center allocation using Platt's method,
- adaptation of output-layer weights using Givens QR decomposition (GQRD) algorithm for recursive least-squares estimation [8],
- introduction of a pruning strategy for existing centers based on monitoring error-reduction proportion of individual centers.

Generally, RAN allocates far fewer centers than is the number of presented examples, but it can lead to exaggerated number of centers in the case of long period data sequence [12]. Our modification prevents this undesirable effect and thus holds complexity of the network on the "low" level.

2. Methods

2.1. RBF Network

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

$$\hat{y} = b_0 + \sum_{i=1}^{N_r} b_i \phi_i(\|\mathbf{x} - \mathbf{c}_i\|/h_i)$$

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, $b_i \in R$ are the weights, $\mathbf{c}_i \in R^n$ represent the positions of RBF centers, and N_r is the number of centers. In our study Gaussian transfer function was used.

2.2. Center Allocation and Learning Strategy

The network starts to learn with no centers. The condition for allocating a new center at (discrete) time j exploits two criteria proposed by Platt [12]. The first criterion is based on prediction error $|e(j)| = |y(j) - \hat{y}(j)|$, where $y(j)$ is desired output. Error is compared with the critical value ϵ. The second criterion is satisfied if the Euclidean distance of input $\mathbf{x}(j)$ from the nearest center $\mathbf{c}_{nearest}$ is greater than the critical scale resolution $\delta(j)$. The learning starts with the largest scale of resolution, i.e. $\delta(0) = \delta_{max}$, and δ is multiplied at each time step by a decay constant $0 < \gamma < 1$ until it reaches the smallest value δ_{min}. If both criteria are satisfied, a new center is set as

$$\mathbf{c}_{new}(j) = \mathbf{x}(j)$$

with width

$$h_{new} = \kappa\|\mathbf{x}(j) - \mathbf{c}_{nearest}(j)\|^2 ,$$

where κ is a constant, and corresponding output weight

$$b_{new}(j) = e(j) .$$

LMS gradient descent is used to update positions of the centers

$$\Delta\mathbf{c}_i(j) = 2\frac{\alpha}{(h_i)^2}(\mathbf{x}(j) - \mathbf{c}_i(j))\phi_i(\mathbf{x}(j))e(j)b_i(j) ,$$

where $\alpha > 0$ is the learning factor.

Summary of the network learning strategy can be described with the following pseudo-code:

$\delta(0) = \delta_{max},\ b_0(0) = y(0),\ N_r = 0$, set ϵ
for $j = 1$ to number of iterations {
    present a new input-output pair $(\mathbf{x}(j), y(j))$
    evaluate output of network $\hat{y}(j) = b_0(j) + \sum_{i=1}^{N_r} b_i(j)\phi_i(\mathbf{x}(j))$
    compute error $e(j) = y(j) - \hat{y}(j)$
    find distance to nearest center $d = \min_{1\le i\le N_r} ||\mathbf{c}_i(j) - \mathbf{x}(j)||$
    if $|e(j)| > \epsilon$ and $d > \delta(j)$
        allocate new center: $N_r = N_r + 1,\ \mathbf{c}_{N_r}(j) = \mathbf{x}(j)$,
            $b_{N_r}(j) = e(j),\ h_{N_r} = \kappa d^2$
    else
        Update$(b_0(j), \{b_i(j), \mathbf{c}_i(j)\}_{i=1}^{N_r})$
    if $\delta(j) > \delta_{min}$
        $\delta(j+1) = \gamma\delta(j)$
}

Our research confirmed the fact revealed by McLachlan [10], that after adding a new center it is appropriate, for T time steps, just to adapt parameters of the network and not to allow another center allocation. The restriction is motivated by increase of the output error observed after adding a new center and thus indicating another center allocation.

2.3. Weights Adaptation

The on-line adaptation of output-layer weights $\mathbf{b} = [b_0, b_1, \cdots, b_{N_r}]^T$ can be formulated as a problem of finding a weights vector $\mathbf{b}$ which minimizes some performance criterion. A very popular criterion is the quadratic function defined at time t as

$$V_t(\mathbf{b}) = \sum_{j=0}^{t} w_t(j)e^2(j|\mathbf{b}) \ , \tag{1}$$

where $e(j|\mathbf{b})$ is the error signal at time j defined as $e(j|\mathbf{b}) = y(j) - \hat{y}(j|\mathbf{b})$ and $w_t(j)$ is a window function imposed on the error signal to reflect time-varying significance of past and recent information. A popular choice of the window function is the exponential window

$$w_t(j) = \lambda^{t-j} \ ,$$

where $0 < \lambda < 1$ is a forgetting factor [1].

It is obvious that from the viewpoint of weights adaptation, the RBF network can be understood as a special case of linear regression model

$$\mathbf{y}(t) = \Phi(t)\mathbf{b}(t) + \mathbf{e}(t) \ , \tag{2}$$

[1] It is necessary to note that in the case of non-stationary data, the application of the forgetting factor might improve estimate of the weights, but convergence is not guaranteed (see, e.g. [8]).

where $\mathbf{y}(t) = [y(1), y(2), \ldots, y(t)]^T$, $\mathbf{e}(t) = [e(1), e(2), \ldots, e(t)]^T$, $\Phi(t)$ is $t \times (N_r + 1)$ matrix whose transpose $\Phi^T(t) = [\phi(1), \phi(2), \ldots, \phi(t)]$, in which $\phi(j) = [1, \phi_1(j), \ldots, \phi_{N_r}(j)]^T$ is a $(N_r + 1) \times 1$ vector of the hidden-layer outputs (+ bias term) at time j. Using the criterion (1), the determination of $\mathbf{b}(t)$ is a least-squares problem which leads to recursive solving of normal equations of the form [8]

$$\Phi(t)^T \Lambda(t) \Phi(t) \hat{\mathbf{b}}(t) = \Phi(t)^T \Lambda(t) \mathbf{y}(t) \;, \tag{3}$$

where $\Lambda(t) = diag[\lambda^{t-1}, \lambda^{t-2}, \cdots, 1] \equiv diag[\lambda(t)\Lambda(t-1), 1]$ is a $(t \times t)$ diagonal matrix. Let $\Lambda^{1/2}(t)$ be a Cholesky factor of the matrix $\Lambda(t)$. Then we can rewrite (3) to the form

$$(\Lambda^{1/2}(t)\Phi(t))^T (\Lambda^{1/2}(t)\Phi(t)) \hat{\mathbf{b}}(t) = (\Lambda^{1/2}(t)\Phi(t))^T \Lambda^{1/2}(t) \mathbf{y}(t) \,.$$

Gentleman [5] and Hammarling [6] derived an extremely efficient QR decomposition algorithms to solve the system of equations (3). The algorithms are based on modified Givens rotations which require no square-root operation (for detailed analysis see [8, 2, 4]). QR decomposition constitutes a form of orthogonal triangulization and has particularly good numerical properties. By writing

$$\Lambda^{1/2}(t)\Phi(t) = Q(t)R(t) \;,$$

where $Q(t)$ is a matrix with mutually orthogonal columns and $R(t)$ is an upper triangular matrix, system (3) can be rewritten as

$$\Theta(t)\hat{\mathbf{b}}(t) = \mathbf{q}(t) \;, \tag{4}$$

where

$$\Theta(t) = [Q^T(t)Q(t)]^{1/2} R(t)$$

is an upper triangular square matrix and $\mathbf{q}(t)$ is a vector

$$\mathbf{q}(t) = [Q^T(t)Q(t)]^{-1/2} Q^T(t) \Lambda^{1/2}(t) \mathbf{y}(t) \,.$$

Thus, the estimation of the vector $\mathbf{b}$ is based on solution of the system (4), which consists of the following steps:

1. initialize matrix $\Theta(.)$ and vector $\mathbf{q}(.)$

- at time $k = 0$ set $\Theta(0) = \eta$ and $\mathbf{q}(0) = 0$
- if a new center N_r is allocated at time $k \neq 0$, increase dimensions of $\Theta(k)$ and $\mathbf{q}(k)$ to $(N_r + 1) \times (N_r + 1)$ and $(N_r + 1)$, respectively, and set diagonal elements $\Theta_{N_r+1, N_r+1}(k) = \eta / e(k)$ and $q_{N_r+1}(k) = \eta$

η is a small positive number

2. at each time $j \neq k$, update $\lambda(j)$ via $\lambda(j) = \lambda_0 \lambda(j-1) + 1 - \lambda_0$, the appropriate values for λ_0 and $\lambda(0)$ are just less than one [2, 9]

3. at time $j \neq k$, form the following $(N_r+2) \times (N_r+2)$ matrix

$$\Omega(j) = \left[\begin{array}{c|c} \lambda^{1/2}\Theta(j-1) & \lambda^{1/2}\mathbf{q}(j-1) \\ \hline \phi^T(j) & y(j) \end{array} \right]$$

4. at time $j \neq k$, perform sequence of elementary Givens rotations (GR) [8] to transform the matrix $\Omega(j)$ to upper triangular matrix

$$\Omega(j) = \left[\begin{array}{c|c} \lambda^{1/2}\Theta(j-1) & \lambda^{1/2}\mathbf{q}(j-1) \\ \hline \phi^T(j) & y(j) \end{array} \right] \overset{\text{GR}}{\Longrightarrow} \left[\begin{array}{c|c} \Theta(j) & \mathbf{q}(j) \\ \hline \mathbf{0} & * \end{array} \right]$$

where $\mathbf{0}$ is $1 \times (N_r + 1)$ zeros vector and $*$ is a *don't care* variable

5. at time $j \neq k$, compute $\hat{\mathbf{b}}(j) = \Theta^{-1}(j)\mathbf{q}(j)$

The initialization of the matrix $\Theta(k)$ and the vector $\mathbf{q}(k)$ (at time $k \neq 0$) in step 1 of the algorithm is based on solving the system (4). According to Platt's algorithm, we set a new weight $b_{N_r}(k) = e(k)$ after allocating a center N_r, and by solving the equation defined by the last row of (4) we achieve the proposed initialization.

The similar method of weights adaptation, based on extended Kalman filter (EKF) algorithm, was applied to RANs by Kadirkamanathan and Niranjan [7] and extended with Bayesian approach by McLachlan and Lowe [11]. Comparison of algorithms based on described GQRD and EKF can be found in [8].

It is known that convergence of GQRD algorithm causes lower adaptability of the network to slowly varying non-stationarity of the time-series and to the increase of the order of the network after allocating a new center. To constrain this undesirable effect, we re-initialized $\lambda(j)$ to $\lambda(0)$ in the case that a new center was allocated at time j.

2.4. Pruning Strategy

Due to the fact that the above proposed algorithm does not remove existing centers whose contribution to prediction accuracy is becoming negligible or significantly falling down, the order of the network can unsuitably increase without improving performance accuracy of the network. To measure the contribution of the individual centers, we used analogy with linear regression model. Billings and Chen [1] proposed a criterion to select the most important regressors called *error-reduction-ratio* (ERR) also used in *orthogonal least squares* method [3].

Transforming the columns of matrix Φ in linear regression model (2) into a set of orthogonal basis vectors [3], we can rewrite (2) into

$$\mathbf{y} = W\mathbf{g} + \mathbf{e} \ ,$$

where W is a $t \times (N_r + 1)$ matrix with orthogonal columns $\mathbf{w}_i$. Then the orthogonal least-squares solution $\hat{g}_i$ is given as

$$\hat{g}_i = \frac{\mathbf{w}_i \mathbf{y}}{(\mathbf{w}_i^T \mathbf{w}_i)} \ , \quad 1 \leq i \leq N_r + 1 \ .$$

Because the $\mathbf{w}_i$ and $\mathbf{w}_j$ are orthogonal, the desired output variance is

$$\frac{1}{t}\mathbf{y}^T\mathbf{y} = \frac{1}{t}\sum_{i=1}^{N_r+1} g_i^2 \mathbf{w}_i^T \mathbf{w}_i + \frac{1}{t}\mathbf{e}^T\mathbf{e} \ .$$

Thus, $\frac{1}{t}\sum_{i=1}^{N_r+1} g_i^2 \mathbf{w}_i^T \mathbf{w}_i$ represents the part of desired output variance explained by the regressors and $\frac{1}{t}\mathbf{e}^T\mathbf{e}$ is the unexplained desired output variance. The ERR of the i-th regressor is then defined as

$$ERR_i = \frac{\hat{g}_i^2 \mathbf{w}_i^T \mathbf{w}_i}{\mathbf{y}^T\mathbf{y}} \ , \ \ 1 \leq i \leq N_r + 1 \ .$$

By examining equation (4), as mentioned in [4], it is obvious that elements $q_i(j)$ of the vector $\mathbf{q}$ at time j are directly related to the relative strengths of the existing regressors (in our case centers). Thus, we can write ERR of the i-th center at time j in the form

$$ERR_i(j) = \frac{{q_i}^2(j)}{\mathbf{y}^T\mathbf{y}} \ . \tag{5}$$

A large value of $ERR_i(j)$ indicates the significant contribution of the i-th center to the output error and vice versa.

In our on-line algorithm, we normalized ${q_i}^2(j)$ in (5) by $y^2(j)$ instead of $\mathbf{y}^T\mathbf{y}$, and used the following idea to monitor the contribution of individual centers to prediction accuracy. Let's assume that at time k a new center N_r was allocated. For each center i $(1 \leq i \leq N_r)$, at each time j, $k < j < T < K$, we compute number of cases C_i which detect momentary ERR_i decrease, i.e. $ERR_i(j) \leq ERR_i(j-1)$. K denotes the time instant at which the criteria for allocating a new center are satisfied. At time K we prune a center s if

$$\max_{1 \leq i \leq N_r} C_i = C_s > \zeta(K - k) \ ,$$

where $0 \leq \zeta \leq 1$ is a proportion constant and C_s represents the center with the highest number of decreases of the ERR during the time interval between two demands for allocating a new center. If the pruning criterion is not satisfied, we just allocate a new center.

During the experiments, we observed several cases of a short-time instability (usually 5-10 time steps) of the GQRD algorithm which occurred after pruning a center. This is due to a sudden enormous change of elements of the matrix $\Theta(.)$ and the vector $\mathbf{q}(.)$, which is caused by pruning a center and adding a new one. To avoid this, we experimentally found the criterion for initializing the s-th row and s-th column of the matrix $\Theta(K)$ and the s-th element of the vector $\mathbf{q}(K)$. After pruning the center C_s at time K, we set the s-th diagonal element of $\Theta(K)$ to $1/\eta$ and other elements of s-th row and s-th column we set to zero. The s-th element of the vector $\mathbf{q}(K)$ is set to η. In both cases, η is the same small positive number as in step 1 of the algorithm in section 2.3.

2.5. Time-Series Prediction

The proposed approach was applied to prediction of chaotic Mackey-Glass time series defined by differential delay equation

$$\frac{ds(t)}{dt} = -bs(t) + a\frac{s(t-\tau)}{1+s(t-\tau)^{10}}$$

with $a = 0.2, b = 0.1$. We used the first 3103 data points (training part) of data set available from CMU Learning Benchmark Archive [2]. The network was trained to predict the value at time $t + 85$, from inputs at time t, $t - 6$, $t - 12$, and $t - 18$. The following network parameters were used: $\delta_{max} = 0.7$, $\delta_{min} = 0.07$, $\gamma = 0.999$, $\lambda(0) = 0.9$, $\lambda_0 = 0.99$, $\alpha = 0.05$, $\eta = 10^{-5}$. We used several combinations of the parameters ϵ, κ, T, ζ (Table I). The smaller value of ϵ causes allocation of the higher number of centers, so we can decrease the widths of the centers (by parameter κ) and "soften" the criterion for pruning centers (by parameter ζ) . In our experiments, we did not allow pruning of the centers during the first 1000 time-steps.

To compare our results with those reported by Platt, we constructed a RAN with the same methodology as described in [12]. We used Gaussian transfer function instead of Platt's function approximation and Platt's strategy of δ decay so that δ decreases to δ_{min} at the end

[2] http://legend.gwydion.cs.cmu.edu/neural-bench/benchmarks/mackey-glass.html. These data were generated with $\tau = 17$ and using a second-order Runge-Kutta method with a step size 0.1.

Table I. Combinations of the parameters values used in experiments.

ϵ	κ	T	ζ
0.1	2.0	30	0.8
0.05	2.0	30	0.8
0.02	1.75	30	0.75
0.01	1.5	40	0.75

of learning process. The widths of the centers were defined in the same way as in Platt's approach.

3. Results

The quality of prediction was evaluated in terms of number of centers (NC) and normalized root mean squared error (NRMSE) defined as

$$\mathrm{NRMSE}(j) = \sqrt{\frac{\sum_{i=1}^{j}(y(i) - \hat{y}(i))^2}{\sum_{i=1}^{j}(y(i) - \bar{y}(i))^2}} \;, \quad \bar{y}(j) = \frac{1}{j}\sum_{i=1}^{j} y(i) \;.$$

We compared our two modifications of RAN – RAN using Givens QR decomposition (RAN-GQRD) and RAN using pruning and GQRD (RAN-P-GQRD) – with algorithm proposed by Platt (RAN). The results achieved at the end of iterative learning are summarized in Table II. One can see, that by using RAN-GQRD network NRMSE decreased by 50% on average compared to RAN. Moreover, improvement was obtained also in terms of NC which significantly decreased (4 times on average).

For the case of $\epsilon = 0.05$ we also present three characteristics (NC, NRMSE and WPE which is defined below) of the iterative learning process.

Figure 1 displays the NC. It can be observed that, during the first half of the learning process, the NC was approximately equal for both approaches. In the later stages, RAN allocated new centers more rapidly and final NC was approximately 3.5 times greater than in the case of RAN-GQRD.

In Figure 2, the dependence of NRMSE on number of observations is depicted. During the whole learning process, RAN-GQRD was consistently better than RAN.

The third measure – exponentially weighted prediction error (WPE) – was proposed by Kadirkamanathan & Niranjan [7]. It represents

Table II. Comparison of achieved results of RAN, RAN-GQRD and RAN-P-GQRD algorithms for various values of ϵ.

	RAN		RAN-GQRD		RAN-P-GQRD	
ϵ	NC	NRMSE	NC	NRMSE	NC	NRMSE
0.1	57	0.378	14	0.206	14	0.206
0.05	92	0.376	24	0.170	24	0.174
0.02	113	0.373	44	0.172	31	0.160
0.01	123	0.374	50	0.165	38	0.183

the prediction performance of the network which continually adapts to incoming data. WPE, at time j, can be recursively computed as

$$\mathrm{WPE}(j)^2 = \vartheta\, \mathrm{WPE}(j-1)^2 + (1-\vartheta)\, |e(j)|^2$$

for some $0 < \vartheta < 1$. We used the same $\vartheta = 0.95$ as in [7]. From the Figure 3 it is apparent that also in terms of WPE, RAN-GQRD was significantly better than RAN (on average 2 times smaller WPE).

To evaluate the significance of the pruning strategy, we compared the performance of RAN-GQRD and RAN-P-GQRD networks for $\epsilon = 0.01$. By using the pruning strategy we achieved similar prediction performance as with GQRD network (Fig. 5 and Fig. 6), but the final number of centers was 1.3 times smaller (Fig. 4). The high peaks of WPE and small increases of NRMSE reflect the short-time numerical instability of the GQRD algorithm, which occurs after pruning.

4. Conclusions

In this paper we report about a new method for constructing RANs. Developed networks were applied to chaotic time-series prediction. Having used NRMSE and NC as criteria for model evaluation, we found out that our modifications using GQRD method provided results that were superior to results achieved by Platt's algorithm.

We introduced a strategy for pruning centers with low or decreasing contribution to prediction accuracy of the network. The results achieved with this modification in on-line prediction task were similar to the best results we got without pruning, but the number of allocated centers was smaller.

We think that the presented approach leads to on-line construction of neural network models with optimal (or near optimal) complexity and preserved prediction performance.

Acknowledgements

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

References

1. S. A. Billings and S. Chen. Extended model set, global data and threshold model identification of severely non-linear systems. *International Journal of Control*, 50(5):1897–1923, 1989.
2. S.A. Billings and C.F. Fung. Recurrent radial basis function networks for adaptive noise cancellation. *Neural Networks*, 8(2):273–290, 1995.
3. S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. *IEEE Transactions on Neural Networks*, 2(2), March 1991.
4. C. F. Fung, S. A. Billings, and W. Luo. On-line Supervised Adaptive Training Using Radial Basis Function Networks. *Neural Networks*, 9(9):1597–1617, December 1996.
5. W.E. Gentleman. Least squares computations by Givens transformations without square roots. *Journal of Instrumentation of Mathematical Applications*, 12:329–336, 1973.
6. S. Hammarling. A note on modifications to the Givens transformations without square-roots. *J. Inst. Math. Applics.*, 13:215–218, 1974.
7. V. Kadirkamanathan and M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5:954–975, 1993.
8. N. Kalouptsidis and S. Theodoridis. *Adaptive system identification and signal processing algorithms*. Prentice Hall, 1993.
9. L. Ljung and T. Söderström. *Theory and practice of recursive identification*. MA: MIT Press, Cambridge, 1985.
10. A. McLachlan. An improved novelty criterion for RAN. Technical report, Aston University, Birmingham UK, 1996.
11. 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.
12. C.J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3(2):213–225, 1991.

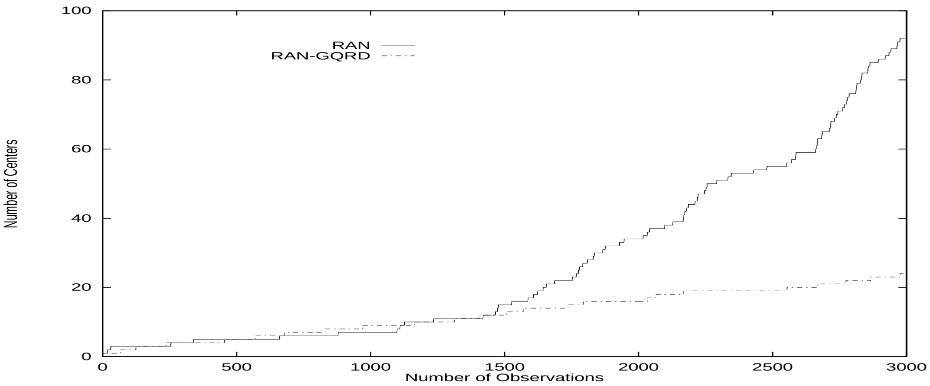

Figure 1. Growth of RAN and RAN-GQRD during the learning process ($\epsilon = 0.05$).

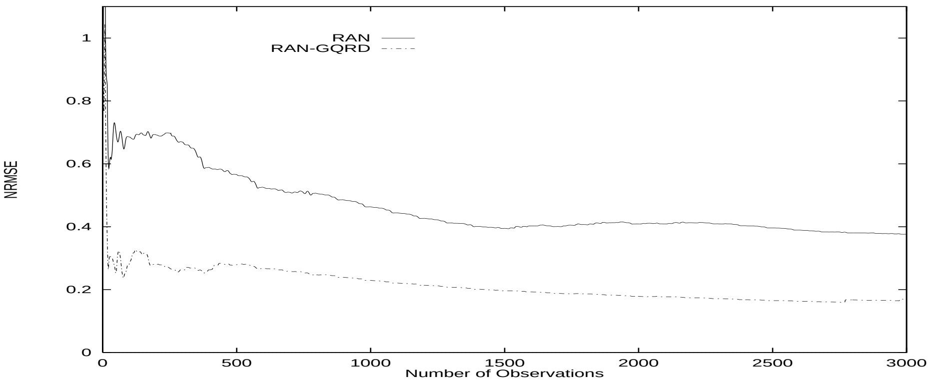

Figure 2. Prediction accuracy of RAN and RAN-GQRD in terms of NRMSE ($\epsilon = 0.05$)

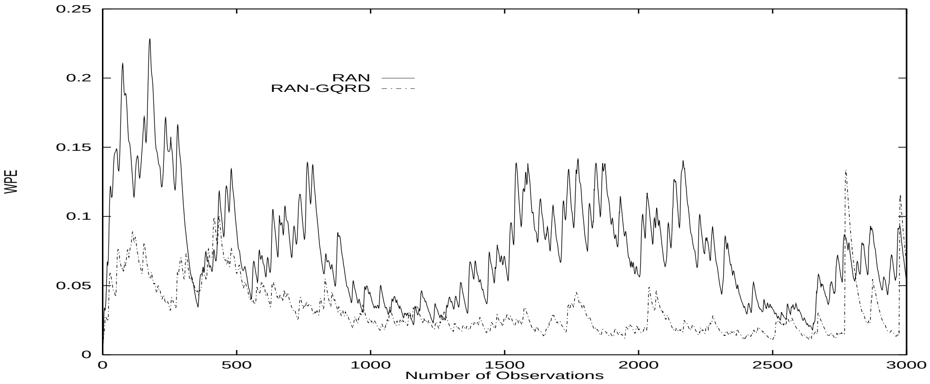

Figure 3. Prediction accuracy of RAN and RAN-GQRD in terms of WPE ($\epsilon = 0.05$)

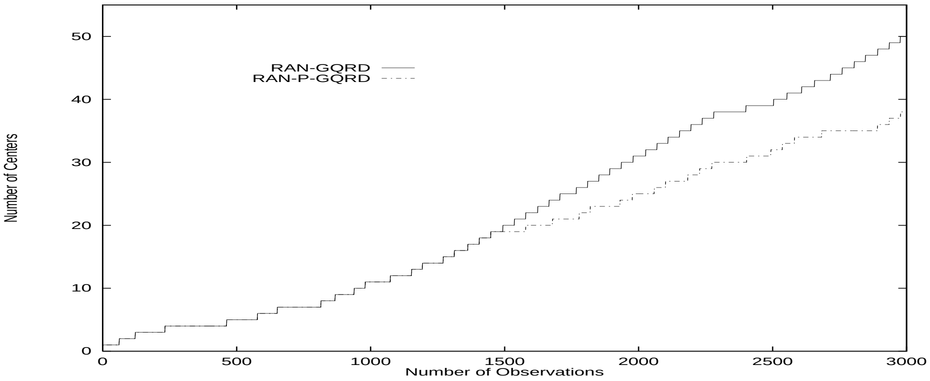

Figure 4. Growth of RAN-GQRD and RAN-P-GQRD during the learning process ($\epsilon = 0.01$).

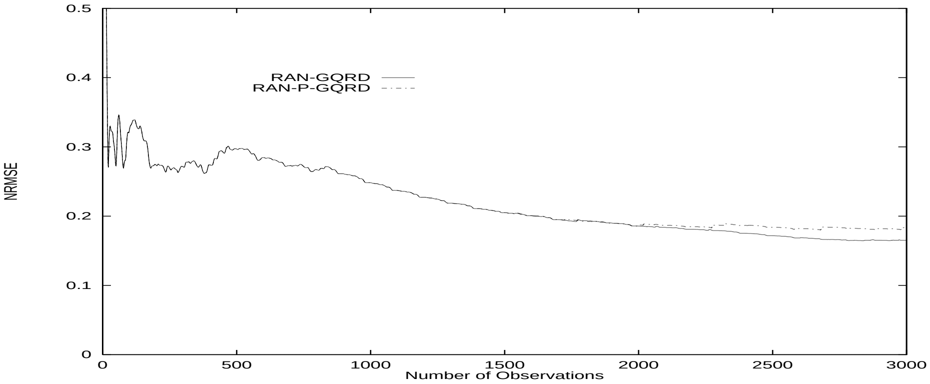

Figure 5. Prediction accuracy of RAN-GQRD and RAN-P-GQRD in terms of NRMSE ($\epsilon = 0.01$)

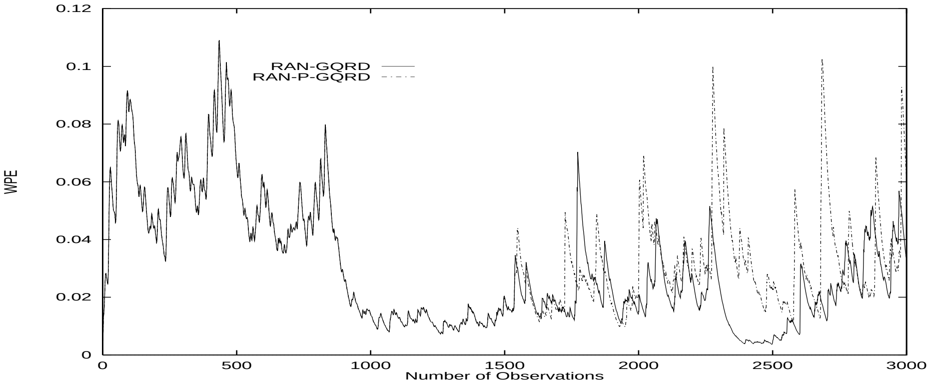

Figure 6. Prediction accuracy of RAN-GQRD and RAN-P-GQRD in terms of WPE ($\epsilon = 0.01$)