Experimental Evaluation of Growing and Pruning Hyper Basis Function Neural Networks Trained with Extended Information Filter

Najdan Vuković*, Marko Mitić**,

Milica Petrović**, Jelena Petronijević**, Zoran Miljković**

* University of Belgrade-Faculty of Mechanical Engineering/Innovation Center, Belgrade, Republic of Serbia

** University of Belgrade-Faculty of Mechanical Engineering/Production Engineering Department, Belgrade, Republic of Serbia

{nvukovic, mmitic, mmpetrovic, jpetronijevic, zmiljkovic}@mas.bg.ac.rs

***Abstract*— In this paper we test Extended Information Filter (EIF) for sequential training of Hyper Basis Function Neural Networks with growing and pruning ability (HBF-GP). The HBF neuron allows different scaling of input dimensions to provide better generalization property when dealing with complex nonlinear problems in engineering practice. The main intuition behind HBF is in generalization of Gaussian type of neuron that applies Mahalanobis-like distance as a distance metrics between input training sample and prototype vector. We exploit concept of neuron's significance and allow growing and pruning of HBF neurons during sequential learning process. From engineer's perspective, EIF is attractive for training of neural networks because it allows a designer to have scarce initial knowledge of the system/problem. Extensive experimental study shows that HBF neural network trained with EIF achieves same prediction error and compactness of network topology when compared to EKF, but without the need to know initial state uncertainty, which is its main advantage over EKF.**

I. INTRODUCTION

Radial basis function (RBF) neural networks are among the most used single layered neural networks [1, 2]. They are popular choice made by many engineers for modeling of complex real world problems [1-7].

In this paper, we develop and evaluate sequential learning algorithm based on Extended Information Filter (EIF) of special class of generalized RBF neural networks with Gaussian basis function that allows: (i) growing of neurons, (ii) pruning of neurons, and (iii) scaling of local input dimensions. Compared to the RBF, the hyper basis function (HBF) has different scale for each dimension of the input vector. Research results [1, 2] have shown that HBF generates neural network with same accuracy as RBF or even higher accuracy than RBF, but with less number of basis functions [2, 8-12]. Learning algorithm for HBF network learning algorithm is sequential and during sequential learning using Extended Information Filter (EIF) it changes number of HBF neurons according to predefined optimality criteria. In this paper, growing and pruning ability of HBF neural network is founded on the original contribution of the neuron's significance, introduced by Huang et al. in series of papers [13, 14], modified by Bortman and Aladjem [15] and generalized in [2], with introduction of Gaussian mixture model (GMM) for modeling of complex input densities [16].

This paper is structured as follows: in the second part of the paper we provide basic information related to intuition behind HBF network; in the third part the main learning algorithm is presented. Experimental results are presented in the fourth part, while concluding remarks are given in the final part.

II. HYPER BASIS FUNCTION NEURAL NETWORKS

The general mathematical form of RBF neural network is given as:

$$\mathbf{y}_i = \mathbf{f}\left(\mathbf{x}_i\right) = \sum_{j=1}^{J} w_j g_j\left(\mathbf{x}_i, \boldsymbol{\mu}_j, \sigma_j\right) \tag{1}$$

where $g_j(\cdot,\cdot,\cdot)$ stands for the j-th basis function; $\mathbf{x}_i \in \mathbb{R}^{n_x}$ is the input vector in the RBF neural network and n_x stands for the number of the input dimensions, $\boldsymbol{\mu}_j \in \mathbb{R}^{n_x}$ is the center of j-th basis function (also referred as the prototype vector); w_j is connecting weight of the j-th basis function, and σ_j is the spread ($\sigma_j \in \mathbb{R}^1$). Hyper basis function uses the Mahalanobis-like distance and it is given in the following form [1, 2]:

$$\begin{aligned} g_j\left(\mathbf{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) &= \exp\left(-0.5\left\|\mathbf{x}_i - \boldsymbol{\mu}_j\right\|_{\boldsymbol{\Sigma}_j}^2\right) = \\ &\exp\left\{-0.5\left(\mathbf{x}_i - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j \left(\mathbf{x}_i - \boldsymbol{\mu}_j\right)\right\} \end{aligned} \tag{2}$$

where $\|\cdot\|_{\boldsymbol{\Sigma}_j}^2$ is Mahalanobis-like norm weighted with positive definite square matrix $\boldsymbol{\Sigma}_j$. Weighting matrix $\boldsymbol{\Sigma}_j$ makes similarity between the input vector $\mathbf{x}_i$ and j-th center vector $\boldsymbol{\mu}_j$ invariant to scaling and local orientation of the data [8-12]. The Mahalanobis-like distance can be seen as a generalization of the Euclidian distance for

similarity measure. The general case ($\mathbf{\Sigma}_j$ is full matrix) provides flexibility and bigger number of parameters to optimize but it results in severe over-fitting to data [2]. Therefore, in HBF network each HBF neuron has a unique diagonal weighting matrix $\mathbf{\Sigma}_j$, with varying size and restricted orientation, i.e. $\mathbf{\Sigma}_j = diag\left(1/\sigma_1^2, 1/\sigma_2^2, ..., 1/\sigma_{n_x}^2\right)$ [2].

This parameterization provides a trade-off between two extremes: on one hand, we have case where local scaling of data is not allowed, and on the other hand, the case with high degree of freedom of the RBF neural network model. Therefore, with diagonal weighting matrix we are trying to capture additional information. Let us show how important is to have the ability of local scaling of the data, especially for modeling of nonlinear dynamical systems. Let us observe NARX model [2, 8]

$$\begin{aligned}\mathbf{y}(t) = f\big(&\mathbf{y}(t-1),...,\mathbf{y}(t-l_y),...\\ &\mathbf{u}(t-1),...,\mathbf{u}(t-l_u)\big)+\mathbf{e}(t)\end{aligned} \tag{3}$$

where $\mathbf{u}(t)$, $\mathbf{y}(t)$ and $\mathbf{e}(t)$ represent system input, output and noise variables (respectively), l_y and l_u are maximum lags of the input and output, and $f(\cdot)$ is some unknown nonlinear function. Let us form the input vector into neural network given by:

$$\mathbf{z} = \left[\mathbf{z}_1(t),...,\mathbf{z}_l(t)\right]^T, \tag{4}$$

where $l = l_y + l_u$ and elements of $\mathbf{z}$ are:

$$\mathbf{z}_k = \begin{cases} \mathbf{y}(t-k),\ k = 1,...,l_y \\ \mathbf{u}\left(t-\left(k-l_y\right)\right),\ k = l_y+1,...,l_y+l_u \end{cases} \tag{5}$$

Now, without loss of generality, let us assume that input $\mathbf{u}(t)$ and output $\mathbf{y}(t)$ are bounded in $\left[\underline{u},\overline{u}\right]$, $\left[\underline{y},\overline{y}\right]$ and let us define associated ranges as $r_u = \left[\underline{u},\overline{u}\right]$ and $r_y = \left[\underline{y},\overline{y}\right]$. Two common cases can be distinguished: (i) $r_y \ll r_u$ (r_u is much greater than r_y) and (ii) $r_y \gg r_u$ (r_u much less than r_y). In the first case, influence of lagged input variables $\mathbf{u}(t)$ may be exaggerated while the role of states $\mathbf{y}(t)$ may be downplayed; in the second case the opposite is valid: the role of system variables $\mathbf{y}(t)$ will be exaggerated and the role of system inputs $\mathbf{u}(t)$ will be downplayed. This problem is commonly encountered in black box modeling of nonlinear dynamical systems when designers has no prior knowledge of the influence input dimensions have on the output.

It goes without saying that RBF neuron is especially vulnerable to this problem. Some parts of this problem may be solved by normalizing the data; however, the major part will still be unsolved. Having this in mind, the HBF neuron provides natural extension of the RBF neuron that provides us with ability to locally scale input data.

III. EXTENDED INFORMATION FILTER FOR GROWING AND PRUNING HBF NETWORK TRAINING – EIF-HBF-GP

A. Extended Information Filter

EKF is a widely established as one of the most successful learning algorithms for neural networks [1-3, 13-15, 17]. Although theory behind Kalman filtering and filtering in information space is well known [3], to best of our knowledge, EIF has not been applied for machine learning of neural networks, which is the reason why we decided to test its performance. The first attempt in this direction is undertaken in references [3-5] for RBF neural networks without ability to change network topology while learning. In this paper, we test performance of EIF for HBF-GP network training.

The attractiveness of EIF-based sequential learning algorithm is in parameterization of Gaussian distribution. Namely, instead of mean and covariance, information filter parametrizes Gaussian distribution with information vector and information matrix, which is defined as inverse of covariance, i.e.

$$\mathbf{I} = \mathbf{P}^{-1} \tag{6}$$

Now, the covariance matrix tells us how much we do not know about our system; bigger **P** means more uncertainty. If all elements of **P** are large, than it means that, we have no knowledge of our system's initial state. Similarly, all elements of **P** are small, than it means that we have all available information about initial state of our system (needless to say that we as engineers are aware that this ideal situation is not possible). Now, from computational perspective the problem arises when we have no initial knowledge of system's state; the problem is how to "tell" the computer this information. This is why we perform estimation in information space; namely, when we invert covariance matrix it is possible to tell the computer that we have little or no knowledge of system's initial state, which means that out lack of knowledge may be represented in the following symbolical mathematical form:

$$\begin{aligned} \mathbf{P} = \mathbf{0} \quad &\Rightarrow \quad \mathbf{I} = \infty \\ \mathbf{P} = \infty \quad &\Rightarrow \quad \mathbf{I} = \mathbf{0} \end{aligned} \tag{7}$$

In the first case, all elements of covariance matrix **P**(i,j) are small numbers, and all elements of **I**(i,j) are large numbers (symbolically - ∞); in the second case, the opposite is true, elements of covariance matrix **P**(i,j) are large numbers, while elements of **I**(i,j) are small numbers. To summarize, when our initial knowledge of system/problem is scarce, we may change our estimation space and move to information space in which this lack of knowledge is easily defined with $\mathbf{I} = \mathbf{0}$. This is the main reason of EIF deployment in this paper; we would like to model and test those problems in which designers

have little initial knowledge of the system. For additional EIF and EKF analysis the reader is referred to [3-5].

B. EIF based training of Hyper Basis Function Neural Networks with Growing and Pruning Ability

In this section we briefly introduce and explain EIF HBF-GP training algorithm; for additional information and deeper understanding of advanced theoretical concepts the reader is referred to [2]. Firstly, we model the input density $p(\mathbf{x})$ with GMM, which is why closed form analytical solution is enabled; now, we may define the concept of neuron significance [2]:

$$\hat{E}_{sig}(k) = \|\mathbf{w}_k\|_q \left((2\pi/q)^{n_x/2} \det(\mathbf{\Sigma}_j)^{-1/2} \mathbf{N}_j^T \mathbf{A} \right)^{1/q} \quad (8)$$

where q is the vector norm, $\mathbf{A}$ denotes vector of mixing coefficients of GMM, i.e. $\mathbf{A} = [\alpha_1, \alpha_2, ..., \alpha_M]^T$, and M Gaussian distributions $\mathbf{N}_j$ are defined as:

$$\begin{aligned} \mathbf{N}_j = \Big[& N(\boldsymbol{\mu}_j - \mathbf{v}_1; \mathbf{0}, \mathbf{\Sigma}_j^{-1}/q + \mathbf{\Sigma}_1), ..., \\ & N(\boldsymbol{\mu}_j - \mathbf{v}_M; \mathbf{0}, \mathbf{\Sigma}_j^{-1}/q + \mathbf{\Sigma}_M) \Big]^T \end{aligned} \quad (9)$$

where $N(\mathbf{v}_k, \mathbf{\Sigma}_k); k = 1, ...M$ denote k-th Gaussian distribution in GMM. The EIF HBF-GP sequential learning algorithm is given in Table I.

HBF-GP sequential learning algorithm requires several input parameters: initial number of HBF neurons (J_0), parameters required for growth criterion $(\varepsilon_{\min}, \varepsilon_{\max})$, threshold for neuron significance $E_{\min}$, overlap of the hyper basis functions κ, and desired learning accuracy $e_{\min}$.

Prior to beginning of the learning process, the estimation of input density $p(\mathbf{x})$ requires $N_{pre\_history}$ data points.

Algorithm continues with calculation of parameter ε_i needed for growth criterion. In the same step, the algorithm calculates current error of the network $\mathbf{e}_i$ and nearest neuron k to the newest input sample $\mathbf{x}_i$. Neuron significance is calculated using (6). The next step of the HBF-GP determines whether new neuron should be added, i.e. step 3.4: the HBF-GP adds new neuron to the hidden layer only if the possible new neuron *J+1* is sufficiently far from existing neurons, i.e. $\|\mathbf{x}_i - \boldsymbol{\mu}_k\|_{\mathbf{\Sigma}_k} > \varepsilon_i$. The second criterion is same as in [13, 14], and insures that significance of possible new neuron $\hat{E}_{sig}(J+1)$ is greater than threshold significance $E_{\min}$. When neuron *J+1* satisfies these conditions it is added to the HBF structure, and learning procedure goes back to step 3 and presents new learning pair $(\mathbf{x}_i, \mathbf{y}_i)$. Otherwise, the HBF-GP will not add new neuron *J+1* to the HBF hidden layer. In this case, the learning procedure continues with update of the nearest neuron *k* (calculated at step 3.2) with EIF. The reader may notice that only parameters of neuron *k* nearest to the input sample $\mathbf{x}_i$ are updated with EIF: greater difference between input vector $\mathbf{x}_i$ and the *j*-th prototype vector $\boldsymbol{\mu}_j$ (*j* = 1,…,*J*) will result in smaller output/activation of the Gaussian basis function.

TABLE I.
SEQUENTIAL LEARNING ALGORITHM FOR HYPER BASIS FUNCTION NEURAL NETWORKS WITH GROWING AND PRUNING ABILITY

input $(J_0, \varepsilon_{\min}, \varepsilon_{\max}, E_{\min}, \kappa, e_{\min})$

1. Estimate input density $p(\mathbf{x})$ with GMM to obtain estimate $\hat{p}(\mathbf{x})$

2. Set initial parameters of HBF network: $\mathbf{w}_J, \boldsymbol{\mu}_J, \mathbf{\Sigma}_J$

2.1 Define EIF state vector

$\boldsymbol{\lambda} = [\mathbf{w}_1^T \ \mathbf{w}_2^T \ \dots \ \mathbf{w}_J^T \ \boldsymbol{\mu}_1^T \ \boldsymbol{\mu}_2^T \ \dots \ \boldsymbol{\mu}_J^T$

$\dots \ \mathbf{\Sigma}_1^1 \ \mathbf{\Sigma}_1^2 \ \dots \ \mathbf{\Sigma}_1^{n_x} \ \dots \ \mathbf{\Sigma}_J^1 \ \dots \ \mathbf{\Sigma}_J^{n_x}]^T$

3. **For** each observation $(\mathbf{x}_i, \mathbf{y}_i)$ do:

3.1 Calculate parameters for growth criterion

$\varepsilon_i = \max[\varepsilon_{\max}\gamma^i, \varepsilon_{\min}], (0 < \gamma < 1)$

Calculate current error of the HBF network

$\mathbf{e}_i = \mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)$

Find neuron *k* nearest to the input sample $\mathbf{x}_i$

$k = \arg\min \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_{\mathbf{\Sigma}_k}$

3.2 Calculate parameters for potential new neuron

$\mathbf{w}_{J+1} = \mathbf{e}_i, \ \boldsymbol{\mu}_{J+1} = \mathbf{x}_i, \ \mathbf{\Sigma}_{J+1} = \kappa \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_{\mathbf{\Sigma}_k} \mathbf{I}$

3.3 Compute significance of newly added neuron $J+1$, i.e. $\hat{E}_{sig}(J+1)$

$\hat{E}_{sig}(J+1) = \|\mathbf{w}_{J+1}\|_q \left((2\pi/q)^{n_x/2} \det(\mathbf{\Sigma}_{J+1})^{-1/2} \mathbf{N}_{J+1}^T \mathbf{A} \right)^{1/q}$

3.4 Update parameters of HBF network

If $\|\mathbf{x}_i - \boldsymbol{\mu}_k\|_{\mathbf{\Sigma}_k} > \varepsilon_i$ and $\hat{E}_{sig}(J+1) > E_{\min}$

allocate new unit with parameters $\mathbf{w}_{J+1}, \boldsymbol{\mu}_{J+1}, \mathbf{\Sigma}_{J+1}$

Else

Update parameters of nearest neuron *k* using EIF

$\hat{\boldsymbol{\lambda}}_{t|t-1} = \hat{\boldsymbol{\lambda}}_{t-1|t-1}; \ \mathbf{I}_{k|k-1} = \left((\mathbf{I}_{t-1|t-1})^{-1} + \mathbf{Q} \right)^{-1}$

$\hat{\mathbf{y}}_t = \mathbf{g}(\hat{\boldsymbol{\lambda}}_{t|t-1}, \mathbf{x}(t))$

$\mathbf{I}_{t|t} = \mathbf{I}_{k|k-1} + \mathbf{H}_t^T \mathbf{R}_t^{-1} \mathbf{H}_t; \ \mathbf{K}_t = (\mathbf{I}_{t|t})^{-1} \mathbf{H}_t^T \mathbf{R}_t^{-1}$

$\hat{\boldsymbol{\lambda}}_{t|t} = \hat{\boldsymbol{\lambda}}_{t|t-1} + \mathbf{K}_t (\mathbf{y}(t) - \hat{\mathbf{y}}_t)$

Compute significance of nearest neuron *k*, *i.e.* $\hat{E}_{sig}(k)$

$\hat{E}_{sig}(k) = \|\mathbf{w}_k\|_q \left((2\pi/q)^{n_x/2} \det(\mathbf{\Sigma}_k)^{-1/2} \mathbf{N}_k^T \mathbf{A} \right)^{1/q}$

If $\hat{E}_{sig}(k) < E_{\min}$

Remove *k*-th neuron

EndIf

EndIf

EndFor

output $(J, \mathbf{w}_j, \boldsymbol{\mu}_j, \mathbf{\Sigma}_j), j = 1, ..., J$

The Jacobian matrix $\mathbf{H}_k = \nabla_{\boldsymbol{\lambda}} \hat{\mathbf{y}}_k^i = [\nabla_{\mathbf{w}} \mathbf{f}(\cdot) \ \ \nabla_{\boldsymbol{\mu}} \mathbf{f}(\cdot) \ \ \nabla_{\sigma} \mathbf{f}(\cdot)]$ will have non-zero elements only for the nearest neuron *k*; for other neurons $(\forall j \notin k)$ the Jacobian will approach to zero or small neglecting value [13, 14]. Finally, EIF update of HBF-GP neuron is performed in step 3.4. After update of parameters of the nearest neuron *k*, the algorithm moves

on to calculate the significance of the nearest neuron $\hat{E}_{sig}(k)$ and checks if it is greater than threshold significance $E_{\min}$; pruning of neuron occurs if its significance $\hat{E}_{sig}(k)$ is smaller than $E_{\min}$. This learning procedure is sequentially applied for all training samples; when final sample *N* is processed, the learning stops and algorithm outputs parameters of optimized HBF network. Outputs of the learning algorithm are parameters of HBF network: total number of HBF processing units (*J*), connecting weights $\mathbf{w}_j$, centers $\boldsymbol{\mu}_j$ and widths $\boldsymbol{\Sigma}_j$ of HBFs $(j=1,...,J)$.

IV. Experimental Results

A. Experimental Setup and Intuition

To fully assess performance of EIF we setup a series of experiments to perform fair comparison between EIF and its dual EKF. Experimental evaluation is performed using two well-known and widely recognized problems for system identification of unknown nonlinear systems: (i) dynamical system with long input delays-in which the behavior of the system is to be predicted using previous system output and controls in HBF neural network, (ii) nonlinear dynamical system-the dynamical system is nonlinear and HBF-GP neural network is to "figure out" the behavior of system using previous systems states and previous controls. EIF HBF-GP neural network is compared to EKF HBF-GP neural network. We used the same initial parameters for both neural networks. These simulated engineering problems should provide fair comparison between two sequential learning algorithms of hyper basis function neural network with growing and pruning ability, and provide assessment of their performance in terms of compactness of network topologies and generalization. All codes for HBF-GP neural network and EIF-based sequential learning (optimization of parameters) are written and run in Matlab 7.12 programming environment; all experiments are conducted on laptop computer with Intel(R) Core™ i5-4200U CPU @ 1.6GHz (2.3GHz) with 6GB of RAM, running on 64-bit Windows 7.

B. Dynamical System with Long Input Delays

Consider the dynamical plant represented with following equation [2]:

$$y_p(t+1)=0.72y_p(t)+0.025y_p(t-1)u(t-1)+... \\ 0.01u^2(t-2)+0.2u(t-3) \quad (10)$$

There are two input values $y_p(t)$ and $u(t)$ fed into HBF-GP neural network to generate desired output value $y_p(t+1)$. The training inputs are uniformly distributed in the [-2,2] interval for about half of the training time and a single sinusoid signal $1.05\sin(\pi t/45)$ for the remaining training time. The training data has 1000 training examples. To verify performance of the generated HBF-GP network and analyze identification results, the testing signal is adopted as:

$$u(t)=\begin{cases}\sin(\pi t/25), & 0<t<250,\\ 1.0, & 250\le t<500\\ -1.0, & 500\le t<750\\ 0.3\sin(\pi t/25)+0.1\sin(\pi t/32) & \\ +0.6\sin(\pi t/10), & 750\le t<1000\end{cases} \quad (11)$$

The testing set consists of 1000 examples. The input into HBF neural network is formed by previous system state and the most recent control, i.e. input is given by the vector $\mathbf{x}=\left[y_p(t)\quad u(t)\right]^T$. Figure 1 shows training and testing sets. As stated, HBF neuron enables local scaling of data, which is especially important for dynamical systems [2, 8]. Therefore, in experiments we partitioned weighting matrix $\boldsymbol{\Sigma}_j$ into two blocks, i.e.

$$\boldsymbol{\Sigma}_j=\kappa\cdot diag\left(\boldsymbol{\Sigma}_{y_p},\boldsymbol{\Sigma}_u\right) \quad (12)$$

where first block $\boldsymbol{\Sigma}_{y_p}$ defines spreads of state variables $y_p(t)$, while the second block $\boldsymbol{\Sigma}_u$ defines spread of controls $u(t)$; scalar $\kappa\in\mathbb{R}^1$ represents initial scale. Initial values of weighting matrix $\boldsymbol{\Sigma}$ are adopted as:

$$\boldsymbol{\Sigma}_{y_p}=\left\{\max\left(y_p(t)\right)-\min\left(y_p(t)\right)\right\}_{t=1}^{1000};\\ \boldsymbol{\Sigma}_u=\left\{\max\left(u(t)\right)-\min\left(u(t)\right)\right\}_{t=1}^{1000} \quad (13)$$

HBF parameters are set as: $\varepsilon_{\max}=5$, $\varepsilon_{\min}=1$, and $\gamma=0.99$, the initial state uncertainty for EKF are: $p_0=0.9, \mathbf{P}_0=p_0\mathbf{I}_\mathbf{P}$, state transition uncertainty $q_0=0.001, \mathbf{Q}=q_0\mathbf{I}_\mathbf{Q}$, and measurement uncertainty $r_0=1, \mathbf{R}=r_0\mathbf{I}_\mathbf{R}$, where $\mathbf{I}_\mathbf{P}$, $\mathbf{I}_\mathbf{Q}$, and $\mathbf{I}_\mathbf{R}$ are the identity matrices of appropriate dimensions (not to be confused with information matrix **I** defined and used in (8), (9) and Table I). On the other hand, the initial state uncertainty for EIF is different. Namely, we simulated situation in which engineer is faced with real situation in which he has no knowledge about the problem or its knowledge is scarce, but the neural model has to be developed, tested and implemented. Therefore, we move to information space.

Figure 1 depicts training set (upper half) and testing set (lower half) as given by equation (11). Figure 2 shows performance of HBF-GP neural network trained with EKF (above) and EIF (below) for testing set. As it may be seen, both sequential learning algorithms are able to learn unknown relationship between long input delayes and output, and generate desired testing response.

Experimental results averaged over 30 independent trials are presented in Table 1; Table 1 shows Root Mean Square Error (RMS), Mean Absolute Error (MAE), and number of processing units, all averaged over 30 independent trials.

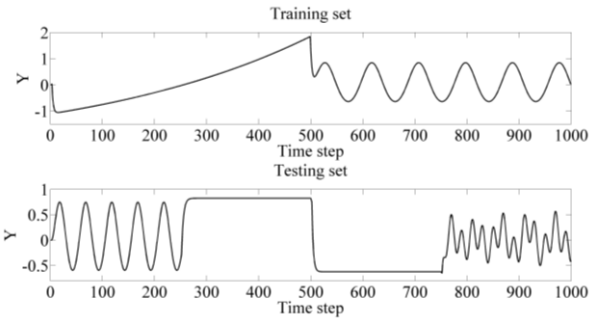

Figure 1. Training and testing set for dynamical system with long input delays.

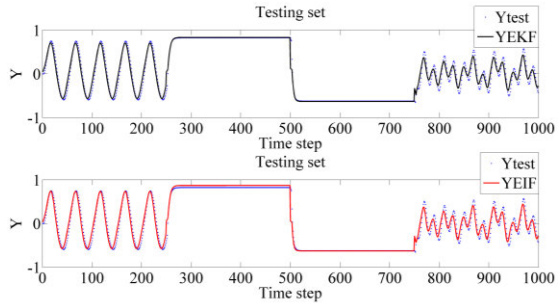

Figure 2. YEKF (EKF HBF – GP) and YEIF (EIF HBF – GP) versus testing set for dynamical system with long input delays.

C. Nonlinear Dynamical System

Consider nonlinear dynamical system given as:

$$y(t+1) = \frac{y(t)y(t-1)y(t-2)u(t-1)\left(y(t-2)-1\right)+u(t)}{1+y^2(t-2)+y^2(t-1)} \quad (14)$$

where $u(t)$ is the control variable (identically independently distributed) in uniform range $[-2,2]$. 800 data points is generated for training of the model. On the other hand, testing set is defined with following dynamics of the control input $u(t)$:

$$u(t) = \begin{cases} \sin(3\pi t/250) & , \quad t \le 500 \\ 0.25\sin(2\pi t/250) + & \\ 0.2\sin(3\pi t/50) & , \quad t > 500 \end{cases} \quad (15)$$

800 data points is used for testing of the model. Figure 3 show training and testing sets. Input vector into HBF neuron is given as $\mathbf{x} = [y(t), y(t-1), y(t-2), u(t), u(t-1)]$, whereas output is given as scalar $y_p = [y(t+1)]$. The initial covariance is initiated using (12) and (13); similarly, we set $p_0 = 0.9, \mathbf{P}_0 = p_0\mathbf{I}_\mathbf{P}$ for EKF while $\mathbf{I} = 0$ for EIF. 30 independent repetitions of HBF-GP learning are conducted. Figure 4 shows how HBF-GP neural network trained with EKF and EIF is able to learn unknown relationship between input vector and output, given training set (14). Experimental results are given in Table II. Furthermore, Figure 5 shows the evolution of total number of HBF neurons during learning process of HBF-GP. Results shown in Figure 5 are averaged over 30 independent trials; as it may be seen, both EKF and EIF converged to approx. three HBF neurons.

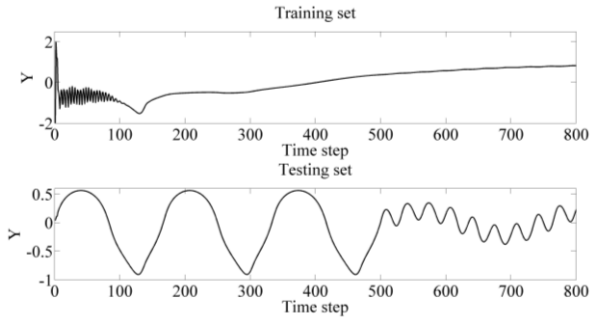

Figure 3. Training and testing set for nonlinear dynamical system.

TABLE I.
EXPERIMENTAL RESULTS FOR HBF-GP NEURAL NETWORK TRAINED WITH EKF AND EIF

	Initial scale of weighting matrix $\mathbf{\Sigma}_j$	Learning algorithm	RMS		MAE		Number of Units
			test	train	test	train	
Dynamical system with long input delays	$\kappa = 0.5$	EKF	0.1025±0.0059	0.0542±0.0069	0.00691±0.0055	0.0388±0.0059	2.9±0.3051
		EIF	0.1141±0.0108	0.0694±0.0112	0.0785±0.0111	0.0524±0.0110	3±0
	$\kappa = 1$	EKF	0.1141±0.0053	0.0630±0.0061	0.0749±0.0046	0.0447±0.0033	2.0667±0.2537
		EIF	0.1258±0.0079	0.0818±0.0067	0.0855±0.0046	0.0587±0.0052	2±0
Nonlinear dynamical system	$\kappa = 0.5$	EKF	0.0524±0.0082	0.1134±0.0182	0.0427±0.0056	0.0554±0.0042	2.9333±0.5833
		EIF	0.0587±0.0174	0.1367±0.0252	0.0477±0.0133	0.0736±0.0131	3.1333±0.6814
	$\kappa = 1$	EKF	0.0717±0.0374	0.1396±0.0213	0.0607±0.0349	0.0705±0.0199	2.6333±0.6149
		EIF	0.0591±0.0377	0.1779±0.020	0.0492±0.0354	0.0836±0.0207	2.1667±0.379

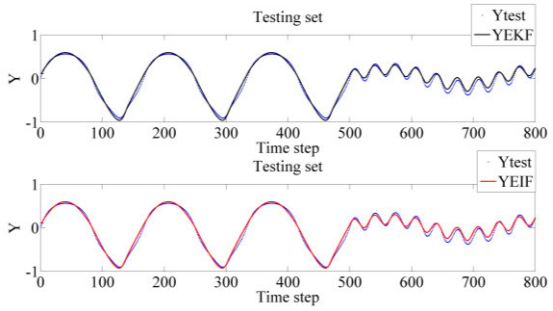

Figure 4. YEKF (EKF HBF – GP) and YEIF (EIF HBF – GP) versus testing set for nonlinear dynamical system.

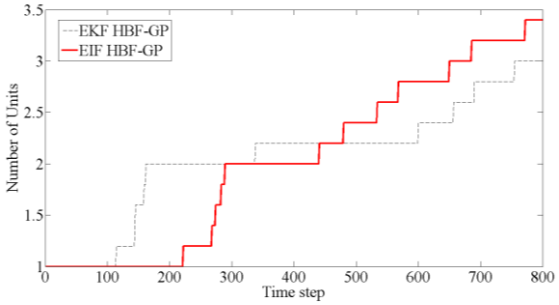

Figure 5. Evolution of number of units during learning for EKF and EIF.

V. Discussion and Conclusion

In this paper, we developed and tested Extended Information Filter (EIF)-based sequential learning algorithm for Hyper Basis Function (HBF) neural network. Unlike a conventional approach, in our research we developed algorithm that enables on line growing and pruning of HBF neural network according to the developed concept of neuron significance [2]. From engineering perspective, EIF has attractive properties when it comes to modeling of complex real world engineering problems with neural networks [3]. Namely, unlike its dual, the extended Kalman Filter (EKF), EIF enables one to set (almost) infinitely large initial covariance matrix; this is important because when faced with hard problems, designer is still able to develop neural network based model/solution although initial knowledge of the problem may be scarce. Furthermore, in our model, we enabled growing and pruning of HBF network topology; the HBF network learns with EIF and simultaneously modifies number of neurons.

EIF is directly compared to its dual (sibling) EKF. As experimental results effectively demonstrate (Figure 1, Figure 4, Figure 5, Table II) EIF-HBF-GP neural network is able to learn complex relations between multidimensional input and output and generate desired response for previously unseen data.

Both of these features are important for engineers working in real world, because real world problems impose mechanisms of how to handle scarce initial knowledge of the problem and how to generate compact network structures. In this paper, we provided a solution for these two problems.

Acknowledgment

This work is supported by the Serbian Government - the Ministry of Education, Science and Technological Development through grant TR35004 (2011-2015).

References

[1] N. Vuković, and Z. Miljković, "Robust Sequential Learning of Feedforward Neural Networks in the Presence of Heavy-Tailed Noise", *Neural Networks*, vol. 63, pp.31-47, April 2015.

[2] N. Vuković, and Z. Miljković, "A Growing and Pruning Sequential Learning Algorithm of Hyper Basis Function Neural Network for Function Approximation", *Neural Networks*, vol. 46C, pp.210-226, October 2013.

[3] N. Vuković, *Machine Learning of Intelligent Mobile Robot Based on Artificial Neural Networks*. Ph.D. dissertation (in Serbian). University of Belgrade – Faculty of Mechanical Engineering, 2012.

[4] N. Vuković, and Z. Miljković, "Machine Learning of Radial Basis Function Neural Networks with Gaussian Processing Units Using Kalman filtering– Introduction", TEHNIKA (in serbian), Vol. LXIX, No. 4, pp. 613-620, 2014.

[5] N. Vuković, and Z. Miljković, "Machine Learning of Radial Basis Function Neural Networks with Gaussian Processing Units Using Kalman filtering – Experimental Results", TEHNIKA (in serbian), Vol. LXIX, No. 4, pp. 621-628, 2014.

[6] N. Vuković, Z. Miljković, B. Babić, and B. Bojović, Training of Radial Basis Function Networks with H^{∞} Filter – Initial Simulation Results, Proceedings of the 6th International Working Conference "Total Quality Management-Advanced and Intelligent Approaches", pp. 163-168, Belgrade, Serbia, 2011.

[7] Z. Miljković, and D. Aleksendrić, *Artificial neural networks—solved examples with theoretical background* (In Serbian). University of Belgrade-Faculty of Mechanical Engineering, Belgrade, 2009.

[8] S.A. Billings, H.L. Wei, and M.A. Balikhin, "Generalized multiscale radial basis function networks", *Neural Networks*, vol. 20(10), pp. 1081-1094, 2007.

[9] T. Poggio, and F. Girosi, "A theory of networks for approximation and learning", A. I. Memo 1140, MIT, 1989.

[10] T. Poggio, and F. Girosi, "Networks for approximation and learning", *Proceedings of the IEEE*, pp. 1481-1497, 1990.

[11] K. Nishida, K. Yamauchi, and T. Omori, "An Online Learning Algorithm with Dimension Selection Using Minimal Hyper Basis Function Networks", *Systems and Computers in Japan*, vol. 37(11), pp. 11-21, 2006.

[12] R.N. Mahdi, and E.C. Rouchka, "Reduced HyperBF Networks: Regularization by Explicit Complexity Reduction and Scaled Rprop Based Training", *IEEE Transactions on Neural Networks*, vol. 22(5), pp. 673-686, 2011.

[13] G.B. Huang, P. Saratchandran, and N. Sundararajan, "An Efficient Sequential Learning Algorithm for Growing and Pruning RBF (GAP-RBF) Networks", *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 34(6), pp. 2284–2292, 2004.

[14] G.B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation", *IEEE Transactions on Neural Networks*, vol. 16(1), pp. 57–67, 2005.

[15] M. Bortman, and M. Aladjem, "A Growing and Pruning Method for Radial Basis Function Networks", *IEEE Transactions on Neural Networks*, vol. 20(6), pp. 1039-1045, 2009.

[16] M. A. T. Figueiredo, and A.K. Jain, "Unsupervised learning of finite mixture models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3), 381-396, 2002.

[17] D. Simon, "Training Radial Basis Function Neural Networks with the Extended Kalman Filter", *Neurocomputing*, vol. 48, pp. 455-475, 2001.