Nonlinear Bayesian Estimation of Recurrent Neural Networks

Branimir Todorović,
Faculty of Occupational Safety
University of Niš
Čarnojevića 10a, 18000 Niš, Serbia
bssmtod@EUnet.yu

Miomir Stanković
Faculty of Occupational Safety
University of Niš
Čarnojevića 10a, 18000 Niš, Serbia
cesiru@ptt.yu

Claudio Moraga
Department of Computer Science
University of Dortmund
Germany
moraga@cs.uni-dortmund.de

Abstract –**We consider the problem of recurrent neural network training as a Bayesian state estimation. The proposed algorithm uses Gaussian sum filter for nonlinear, non-Gaussian estimation of network outputs and synaptic weights. The performances of the proposed algorithm and other Bayesian filters are compared in noisy chaotic time series prediction.**

I INTRODUCTION

Recurrent Neural Networks (RNN) form a wide class of neural networks in which feedback connections between processing units (artificial neurons) are allowed. In this way the static input-output mapping of a neural network is changed into a dynamic system. A simple way of introducing feedbacks into neural networks is to apply local feedback, where adaptive feedback connections are provided only from a processing unit to itself. Such networks are called locally recurrent neural networks. If feedback connections exist between distinct processing units, the networks are called globally recurrent. At least in theory, their modeling capabilities should be much richer than for the simple local feedback networks.

In this paper we will consider training of globally recurrent neural network which can be represented by the following state space model:

$$s_k = \phi(s_{k-1}, w_{k-1}, u_k) + d_{s_k} \quad (1a)$$

$$w_k = w_{k-1} + d_{w_k} \quad (1b)$$

$$y_k = h(s_k, w_k) + v_k \quad (1c)$$

Dynamics of recurrent neuron outputs s_k and synaptic weights w_k is described by (1a) and (1b) respectively. Observation equation (1c) represents the additional layer of output neurons (e.g in Elman RNN) or it simply selects observable among existing recurrent neurons (for fully connected RNN or NARX RNN).

The state space model (1) can be written in a more compact form as:

$$x_k = f(x_{k-1}, u_k) + d_k \quad (2a)$$

$$y_k = h(x_k) + v_k \quad (2b)$$

where x_k represents the hidden state of a RNN, the concatenated vectors of neuron outputs and synaptic weights:

$$x_k = \begin{bmatrix} s_k \\ w_k \end{bmatrix}, \quad f(x_{k-1}, u_k) = \begin{bmatrix} \phi(s_{k-1}, w_{k-1}, u_k) \\ w_{k-1} \end{bmatrix}, \quad d_k = \begin{bmatrix} d_{s_k} \\ d_{w_k} \end{bmatrix}.$$

There are numbers of algorithms for training synaptic weights of recurrent neural networks. Such algorithms are usually based on the exact or approximate computation of the gradient of an error measure in the weight space. Well known approaches that use methods for exact gradient computation are back-propagation through time (BPTT) and real time recurrent learning (RTRL) [8].

In this paper we consider the problem or recurrent neural network (RNN) training as Bayesian state estimation and propose an algorithm based on Gaussian sum filter for nonlinear, non-Gaussian estimation of RNN.

The central problem of the Bayesian estimation is determination of the probability density function of the hidden state of a dynamical system. In a sequential estimation framework, the state filtering probability density function (pdf) $p(x_k / y_{0:k})$, where $y_{0:k} = \{y_0, y_1, ..., y_k\}$ denotes the set of all observations, represents the complete solution. The optimal state estimate with respect to any criterion can be calculated based on this pdf.

Recursive Bayesian estimation algorithm for determination of the filtering pdf consist of two steps: **prediction** and **update**. In the first step the previous posterior $p(x_{k-1} / y_{0:k-1})$ is projected forward in time using the probabilistic process model:

$$p(x_k / y_{0:k-1}) = \int p(x_k / x_{k-1}) p(x_{k-1} / y_{0:k-1}) dx_k \quad (3)$$

The state transition density $p(x_k / x_{k-1})$ is completely specified by $f(\cdot)$ and the process noise distribution.

In the second step, the predictive density is updated by incorporating the latest noisy measurement y_k using the observation likelihood $p(y_k / x_k)$ to generate the posterior:

$$p(x_k / y_{0:k}) = \frac{p(y_k / x_k) p(x_k / y_{0:k-1})}{\int p(y_k / x_k) p(x_k / y_{0:k-1}) dx_k} \quad (4)$$

The recurrence relations (3) and (4) are only conceptual solutions and the posterior density cannot be determined analytically in general. The restrictive set of cases includes the well known Kalman filter. Kalman filter represents the optimal solution of (3) and (4) if the posterior density $p(x_k / y_{0:k})$ and dynamic and observation noise are Gaussian and $f(\cdot)$ and $h(\cdot)$ are known linear functions.

In case of RNN training, these assumptions do not hold. RNN's are in general nonlinear and noise densities are not strictly Gaussain. When the analytic solution is not tractable, some approximations and suboptimal solutions have to be considered. Perhaps the most celebrated suboptimal solution is the Extended Kalman Filter (EKF), which assumes the Gaussian property of noise and uses Taylor expansion of $f(\cdot)$ and $h(\cdot)$ (usually up to the linear term) to obtain the recursive estimation for $p(x_k / y_{0:k})$. EKF has been

successfully applied in RNN training [9,5] due to important advantigies compared to RTRL and BPTT. First, EKF uses second order information, recursively estimated as covariance matrix, to improve convergence speed. RTRL and BPTT are based on the first order derivative information and usually exhibit slow convergence. Second, EKF generalizes the notion of teacher forcing [8,9]. The idea of teacher forcing is to use desired values of neuron outputs where specified, in place of actual values, to compute the future output of the network. In this way the convergence of training is additionally improved. EKF generalizes teacher forcing for RNN's with hidden recurrent neurons (e.g. for Elman RNN with adaptable both feed-forward and recurrent connections), and for noisy training data.

Recently, a family of new derivative free filters have been proposed as an alternative to EKF for estimation in nonlinear systems with noise distributions approximated by single Gaussian. Divided Difference Filters (DDF), derived in [3], are based on polynomial approximation of nonlinear transformations using multidimensional extension of Stirling's interpolation formula. The Unscented Kalman Filter (UKF) [2] uses the true nonlinear models and approximates the state distribution using deterministically chosen sample points. Surprisingly, DDF and UKF result in similar equations and are usually called derivative free filters. It has been shown that DDF and UKF outperform EKF in state and parameter estimation [7, 6].

In this paper we consider Gaussian Sum (GS) filter in RNN training when noise densities cannot be approximated by a single Gaussian. Derivation of GS filter equations is based on the assumption that any probability density function can be approximated sufficiently accurately using finite Gaussian mixture. We have implemented GS filter as a bank of parallel Bayesian filters (EKF, UKF, DDF or others).

II GAUSSIAN FILTERS IN RNN TRAINING

We shall consider the training of Non-linear AutorRegressive with eXogenous inputs (NARX) recurrent neural network. They outperform classical recurrent neural networks in tasks that involve long term dependencies for which the desired output depends on inputs presented at times far in the past.

A NARX model of a dynamic system is given by:

$$s_k = f(s_{k-1},...,s_{k-\Delta_s},u_{k-1},...u_{k-\Delta_u}) \tag{5}$$

where s_k corresponds to the true (noiseless) output of the system, u_k is the known input at time step k, Δ_u and Δ_s are the input and the output order, and $f(\cdot)$ is a non-linear function. We shall consider a NARX model for which $f(\cdot)$ is implemented as a Multilayer Perceptron.

The output of the ith hidden neuron of a NARX recurrent network is given by:

$$\phi_i(\mathbf{s}_{k-1},\mathbf{u}_{k-1},\mathbf{b}_i)=\tanh\left\{b_{i0}+\sum_{l=1}^{\Delta_s}b_{il}s_{k-l}+\sum_{j=1}^{n_u}\sum_{\tau=1}^{\Delta_u}b_{ij\tau}u_{j,k-\tau}\right\} \tag{6}$$

where $\mathbf{s}_{k-1}=[s_{k-1}\cdots s_{k-\Delta_s}]^{\mathrm{T}}_{1\times\Delta_s}$ denotes the vector of previous network outputs, $\mathbf{u}_{k-1}=[u_{k-1}{}^{\mathrm{T}}\cdots u_{k-\Delta_u}{}^{\mathrm{T}}]^{\mathrm{T}}_{1\times\Delta_s}$ is the vector of previous Δ_u inputs, $\mathbf{b}_i=[b_{i0}b_{i1}...b_{i\Delta_s}b_{i11}...b_{in_u\Delta_u}]^{\mathrm{T}}$ denotes the vector of hidden neuron weights, and n_u denotes the dimension of the input u_k.

The network output is given by:

$$f(\mathbf{s}_{k-1},\mathbf{u}_{k-1},w)=a_0+\sum_{i=1}^{n_H}a_i\phi_i(\mathbf{s}_{k-1},\mathbf{u}_{k-1},\mathbf{b}_i) \tag{7}$$

where $\mathbf{a}=[a_0\ a_1...a_{n_H}]$ are the output weights, $\mathbf{w}$ denotes the n_w dimensional vector of unknown network weights, $w=[\mathbf{a}^{\mathrm{T}}\mathbf{b}^{\mathrm{T}}]^{\mathrm{T}}$ and n_H is the number of hidden neurons.

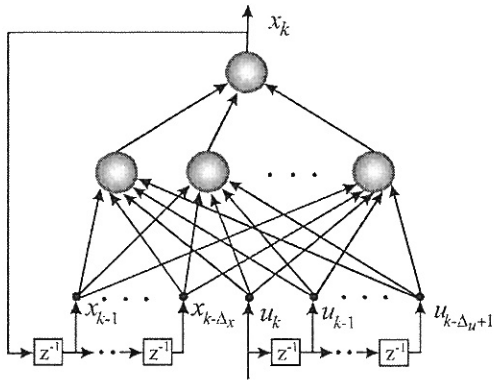

Figure 1. NARX recurrent neural network

Estimation of recurrent neural networks can be put in the framework of nonlinear state estimation by defining the state space model of network dynamics. The state space model of NARX recurrent network is given by:

$$\begin{bmatrix}s_k\\s_{k-1}\\\vdots\\s_{k-\Delta_x+1}\\w_k\end{bmatrix}=\begin{bmatrix}f(s_{k-1},..,s_{k-\Delta_x},u_{k-1},..,u_{k-\Delta_u},w_{k-1})\\s_{k-1}\\\vdots\\s_{k-\Delta_x+1}\\w_{k-1}\end{bmatrix}+\begin{bmatrix}d_{s_k}\\0\\\vdots\\0\\d_{w_k}\end{bmatrix} \tag{8}$$

$$y_k=H\cdot\begin{bmatrix}s_k\\s_{k-1}\\\vdots\\s_{k-\Delta_x+1}\\w_k\end{bmatrix}+v_k,\quad H=[I_{n_X\times n_X}\,0_{n_X\times(n_A-n_X)}] \tag{9}$$

The previous model can be expressed in a compact form (2) if we use x to denote the vector of concatenated network outputs s_k and parameters w_k.

Given the state space model of NARX RNN we can make assumptions that the state, observation and noise terms can be modeled as Gaussian random variables. In that case only the conditional mean $\hat{x}_k=E[x_k/y_{0:k}]$ and its covariance $P_{x_k}=E[(x_k-\hat{x}_k)(x_k-\hat{x}_k)^{\mathrm{T}}/y_{0:k}]$ have to be recursively calculated to determine the Gaussian posterior density.

$$\hat{x}_k=\hat{x}_k^-+K_k(y_k-\hat{y}_k^-) \tag{10a}$$

$$K_k=P_{x_ky_k}P_{y_k}^{-1} \tag{10b}$$

$$P_{x_k}=P_{x_k}^--K_kP_{y_k} \tag{10c}$$

where the optimal terms in this recursions are given by

$$\hat{x}_k^- = E[x_k / y_{0:k-1}] \quad (11a)$$

$$P_{x_k}^- = E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^{\mathrm{T}} / y_{0:k-1}] \quad (11b)$$

$$\hat{y}_k^- = E[y_k / y_{0:k-1}] \quad (12a)$$

$$P_{y_k} = E[(y_k - \hat{y}_k^-)(y_k - \hat{y}_k^-)^T / y_{0:k-1}] \quad (12b)$$

$$P_{x_k y_k} = E[(x_k - \hat{x}_k^-)(y_k - \hat{y}_k^-)^T / y_{0:k-1}] \quad (12c)$$

Terms (11) and (12) are obtained by propagating the state through the nonlinear dynamic and observation model (2). In the following subsection we shall illustrate the techniques that EKF, DDF and UKF use to propagate random variable through the nonlinear transformation.

Suppose that x is a random variable with mean $\hat{x}$ and covariance P_x. A random variable y is related to x through the nonlinear function $y = f(x)$. We wish to calculate the mean $\hat{y}$ and covariance P_y of y.

A. Extended Kalman filter

In case that f and h are linear, the Kalman filter calculates all terms exactly, and it can be considered as an efficient method for analytical propagation of Gaussian random variable through linear system dynamics. Kalman filter can also be applied for nonlinear model if Taylor-series expansion of dynamic and observation equations are provided.

Taylor series expansion of a nonlinear transformation around the mean of the considered random variable x is given as:

$$y = f(x) = f(\hat{x}) + \nabla f \delta x + \tfrac{1}{2!}\nabla^2 f \delta x^2 + \tfrac{1}{3!}\nabla^3 f \delta x^3 + \ldots \quad (13)$$

where the zero mean random variable δx has the same covariance P_x as x. The first order extended Kalman filter equations are obtained by approximating $f(x)$ with $f(\hat{x}) + \nabla f \delta x$ resulting in the approximations of mean and covariance: $\hat{y} \approx g(\hat{x})$ and $P_y \approx \nabla f P_x \nabla f^{\mathrm{T}}$.

In this way the prediction of the state is given by:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k) + \bar{d}_k \quad (14a)$$

$$P_{x_k}^- = F_k P_{x_{k-1}} F_k^T + G_k Q_k G_k^T \quad (14b)$$

where $F_k = \nabla_{\hat{x}_{k-1}} f$, $G_k = \nabla_{\bar{d}_k} f$, $\bar{d}_k = E[d_k]$, $Q_k = E[d_k d_k^T]$.

The prediction of the observation is given by:

$$\hat{x}_k^- = h(\hat{x}_k^-) + \bar{v}_k \quad (15a)$$

$$P_{y_k} = H_k P_{x_k}^- H_k^T + L_k R_k L_k^T \quad (15b)$$

where $H_k = \nabla_{\hat{x}_k^-} h$, $G_k = \nabla_{\bar{v}_k} h$, $\bar{v}_k = E[v_k]$, $R_k = E[v_k v_k^T]$.

B. Divided difference filter

In [3] Nørgaard et al. proposed a new set of estimators based on derivative free polynomial approximation of nonlinear transformations using multidimensional extension of Stirling's interpolation formula. This formula is particularly simple if only first and second order polynomial approximation are considered:

$$f(x) \approx f(\hat{x}) + \tilde{D}_{\Delta x} f + \tilde{D}_{\Delta x}^2 f \quad (16)$$

where divided difference operators are defined by:

$$\tilde{D}_{\Delta x} f = \frac{1}{h}\left(\sum_{p=1}^{n} \Delta x_p \mu_p \delta_p\right) f(\bar{x}) \quad (17)$$

δ_p is a "partial" difference operator:

$$\delta_p f(\hat{x}) = f(\hat{x} + 0.5 \cdot h \cdot e_p) - f(\bar{x} - 0.5 \cdot h \cdot e_p) \quad (18)$$

and μ_p is an average operator:

$$\mu_p f(\hat{x}) = 0.5 \cdot (f(\hat{x} + 0.5 \cdot h \cdot e_p) + f(\hat{x} - 0.5 \cdot h \cdot e_p)) \quad (19)$$

where e_p is the pth unit vector.

Applying a stochastic decoupling of the variables in x by the following transformation $z = S_x^{-1} x$, (S_x is the Cholesky factor of the covariance matrix $P_x = S_x S_x^T$), approximation of mean and covariance of $y = f(x)$ is obtained:

$$\hat{y} = \frac{h^2 - n}{h^2} f(\bar{x}) + \frac{1}{2h^2}\sum_{p=1}^{n} (f(\bar{x} + h s_{x,p}) + f(\bar{x} - h s_{x,p})) \quad (20a)$$

$$\begin{aligned} P_y = {} & \frac{1}{4h^2}\sum_{p=1}^{n} (f(\hat{x} + h s_{x,p}) - f(\hat{x} - h s_{x,p})) \\ & \cdot (f(\hat{x} + h s_{x,p}) - f(\hat{x} - h s_{x,p}))^{\mathrm{T}} \\ & + \frac{h^2 - 1}{4h^4}\sum_{p=1}^{n} (f(\hat{x} + h s_{x,p}) + f(\hat{x} - h s_{x,p}) - 2 f(\hat{x})) \\ & \cdot (f(\hat{x} + h s_{x,p}) + f(\hat{x} - h s_{x,p}) - 2 f(\hat{x}))^{\mathrm{T}} \end{aligned} \quad (20b)$$

The interval length h is set equal to the kurtosis of the prior random variable x. For Gausians it holds $h^2 = 3$.

In [3] Nørgaard et al. also derived the alternative covariance estimate:

$$\begin{aligned} P_y = {} & \frac{1}{2h^2}\sum_{p=1}^{n} (f(\hat{x} + h s_{x,p}) - \hat{y}) f(\hat{x} + h s_{x,p}) - \hat{y})^{\mathrm{T}} \\ & + \frac{1}{2h^2}\sum_{p=1}^{n} (f(\hat{x} - h s_{x,p}) - \hat{y}) f(\hat{x} - h s_{x,p}) - \hat{y})^{\mathrm{T}} \\ & + \frac{h^2 - n}{h^2} (f(\hat{x}) - \hat{y}) f(\hat{x}) - \hat{y})^{\mathrm{T}} \end{aligned} \quad (21)$$

This estimate is less accurate than (20). Moreover, for $h^2 < n$ the last term becomes negative semi-definite with a possible implication that the covariance estimate (27) becomes non-positive definite too. The reason why this estimate is considered here is comparison with the covariance estimate obtained by the Unscented Transformation described in the next subsection.

C. Unscented Kalman filter

Julier and Uhlman proposed the Unscented Transformation (UT) [2] in order to calculate the statistics of a random variable x propagated through the nonlinear function $y = f(x)$. The n_x-dimensional continuous random variable x with mean $\hat{x}$ and covariance P_x is approximated by $2n_x + 1$ sigma points $\mathcal{X}_p$ with corresponding weights ω_p, $p = 0,1,\ldots,2n_x$:

$$\mathcal{X}_0 = \hat{x}, \quad \omega_0 = \lambda/(n+\lambda), \quad \lambda = \alpha^2(n_x+\kappa) - n_x \text{ for } p = 1,2...,n$$

$$\mathcal{X}_p = \hat{x} + \sqrt{n+\lambda}\cdot s_{x,p} \qquad \omega_p = 0.5/(n+\lambda)$$

$$\mathcal{X}_{p+n_x} = \hat{x} - \sqrt{n+\kappa}\cdot s_{x,p} \quad \omega_{p+n_x} = 0.5/(n+\lambda)$$

where α determines the spread of the sigma points around $\hat{x}$ (usually $1.e-4 \le \alpha \le 1$) and $\kappa \in \Re$ is the scaling parameter, usually set to 0 or $3-n_x$[ ]. $s_{x,p}$ is the pth row or column of the matrix square root of P_x

Each sigma point is instantiated through the function $f(\cdot)$ to yield the set of transformed sigma points $\mathcal{Y}_p = f(\mathcal{X}_p)$, and the mean $\hat{y}$ of transformed distribution is estimated by:

$$\hat{y} = \sum_{p=0}^{2n_x}\omega_p\mathcal{Y}_p = \frac{\lambda}{n+\lambda}f(\hat{x}) + \frac{1}{2(n+\lambda)}\sum_{i=1}^{n}(f(\hat{x}+\sqrt{n+\lambda}\cdot s_{x,i}) + f(\hat{x}-\sqrt{n+\lambda}\cdot s_{x,i})) \quad (22)$$

The covariance estimate obtained by the unscented transformation is:

$$P_y = \sum_{p=0}^{2n_x}\omega_p(\mathcal{Y}_p-\hat{y})(\mathcal{Y}_p-\hat{y})^{\mathrm{T}} = \frac{\lambda}{n+\lambda}(f(\hat{x})-\hat{y})f(\hat{x})-\hat{y})^{\mathrm{T}} + \frac{1}{2(n+\lambda)}\sum_{p=1}^{n}(f(\hat{x}+\sqrt{n+\lambda}\cdot s_{x,p})-\hat{y})f(\hat{x}+\sqrt{n+\lambda}s_{x,p})-\hat{y})^{\mathrm{T}} + \frac{1}{2(n+\lambda)}\sum_{p=1}^{n}(f(\hat{x}-\sqrt{n+\lambda}s_{x,p})-\hat{y})f(\hat{x}-\sqrt{n+\lambda}s_{x,p})-\hat{y})^{\mathrm{T}} \quad (23)$$

It can be easily verified that for $h = \sqrt{n+\lambda}$, the estimates of mean (20a) and covariance (21) obtained by DDF are equivalent to the estimates (22) and (23) obtained by UKF.

III Gaussian Sum filters in RNN training

Application of a weighted sum of Gaussian densities for approximation of the posterior density was considered for linear systems with Gaussian noise but a non Gaussian distribution of the initial state $p(x_0)$ [1]. In [4] the idea was further developed for nonlinear systems. However, equations of recursive estimator were derived applying linearization of system dynamics resulting in a bank of parallel EKF's.

In this section we derive equations of Gaussian Sum (GS) filters and consider it in RNN training when noise densities cannot be approximated by a single Gaussian. As in [4] we assume that any probability density function can be approximated sufficiently accurately using finite Gaussian mixture:

$$p(x) \approx \sum_{i=1}^{n} P\{A_i\}p(x/A_i) = \sum_{i=1}^{n} w_i N(x;\bar{x}_i, P_i) \quad (24)$$

where A_i is the event that the x is Gaussian distributed with mean $\bar{x}_i$ and covariance P_i, that is $A_i = \{x \sim N(\bar{x}_i, P_i)\}$. Events $A_i, i = 1,...,n$ are mutually exclusive $P\{A_i A_j\} = 0, \forall i \ne j$, and exhaustive $\sum_{i=1}^{n} P\{A_i\} = 1$, and $P\{A_i\} = w_i$.

A. Gaussian Sum filter equations

To derive GS filter equations we will assume that the filtering and prediction densities as well as non-Gaussian noise densities can be represented as finite Gaussian mixtures.

$$p(x_{k-1}/y_{0:k-1}) = \sum_{j=1}^{n_{k-1}} P\{A_{k-1,j}/y_{0:k-1}\}p(x_{k-1}/y_{0:k-1}, A_{k-1,j}) = \sum_{j=1}^{n_{k-1}} w_{k-1,j}N(x_{k-1};\hat{x}_{k-1,j}, P_{x_{k-1},j}) \quad (25)$$

$$p(d_k) = \sum_{j=1}^{n_{d_k}} P\{B_{k,j}\}p(d_k/B_{k,i}) = \sum_{j=1}^{n_{d_k}} wd_{k,i}N(d_k;\bar{d}_{k,i}, Q_{k,i}) \quad (26)$$

$$p(v_k) = \sum_{j=1}^{n_{v_k}} P\{C_{k,j}\}p(v_k/C_{k,i}) = \sum_{j=1}^{n_{v_k}} wv_{k,i}N(v_k;\bar{v}_{k,i}, R_{k,i}) \quad (27)$$

The predictive density is obtained as:

$$p(x_k/y_{0:k-1}) = \int p(x_k/x_{k-1})p(x_{k-1}/y_{0:k-1})dx_{k-1} = \sum_{i=1}^{n_d}\sum_{j=1}^{n} P\{B_{k,i}\}P\{A_{k-1,j}/y_{0:k-1}\}\int p(x_k/x_{k-1}, B_{k,i})\cdot p(x_{k-1}/y_{0:k-1}, A_{k-1,i})dx_{k-1} \quad (28)$$

If we introduce $D_{k,l}$ to denote a joint event $B_{k,i} \cap A_{k-1,j}$, we have:

$$p(x_k/y_{0:k-1}) = \sum_{l=1}^{n_k^-} P\{D_{k,l}/y_{0:k-1}\}p(x_k/y_{0:k-1}, D_{k,l}) \quad (29)$$

where $l = (i-1)\cdot n + j$ and $n_k^- = n_{d_k}\cdot n_{k-1}$

$$P\{D_{k,l}/y_{0:k-1}\} = P\{B_{k,i}\}P\{A_{k-1,j}/y_{0:k-1}\} \quad (30)$$

since $B_{k,i}$ are independent events. Finally, based on assumptions (25) and (26) we obtain the predictive density as the finite Gaussian mixture:

$$p(x_k/y_{0:k-1}) = \sum_{l=1}^{n_k^-} w_{k,l}^- N(x_k;\hat{x}_{k,l}^-, P_{x_k,l}^-). \quad (31)$$

where $w_{k,l}^- = wd_{k,i}\cdot w_{k-1,j}$, and

$$\hat{x}_{k,l}^- = E[x_k/y_{0:k-1}, D_{k,l}], \quad (32a)$$

$$P_{x_k,l}^- = E[(x_k-\hat{x}_{k,l}^-)(x_k-\hat{x}_{k,l}^-)^T/y_{0:k-1}, D_{k,l}]. \quad (32b)$$

The posterior state density is obtained as:

$$p(x_k/y_{0:k}) = \frac{p(y_k/x_k)p(x_k/y_{0:k-1})}{\int p(y_k/x_k)p(x_k/y_{0:k-1})dx_k} = \frac{\sum_{i=1}^{n_{v_k}}\sum_{j=1}^{n_k^-} P\{C_{k,i}\}P\{D_{k,j}/y_{0:k-1}\}p(y_k/x_k, C_{k,i})\cdot p(x_k/y_{0:k-1}, D_{k,i})}{\sum_{i=1}^{n_{v_k}}\sum_{j=1}^{n_k^-} P\{C_{k,i}\}P\{D_{k,j}/y_{0:k-1}\}\int p(y_k/x_k, C_{k,i})p(x_k/y_{0:k-1}, D_{k,i})dx_k} \quad (33)$$

and in a compact form:

$$p(x_k/y_{0:k}) = \frac{\sum_{l=1}^{n_k^*} P\{A_{k,l}/y_{0:k-1}\}p(y_k/y_{0:k-1}, A_{k,l})\,p(x_k/y_{0:k}, A_{k,l})}{\sum_{l=1}^{n_k^*} P\{A_{k,l}/y_{0:k-1}\}p(y_k/y_{0:k-1}, A_{k,l})} \quad (34)$$

where $A_{k,l}$ denotes a joint event $C_{k,i} \cap D_{k,j}$ and $n_k^* = n_{d_k}\cdot n_k^-$, $l = (i-1)\cdot n_k^- + j$. It can be proved that:

$$p(A_{k,l}/y_{0:k}) = \frac{P\{A_{k,l}/y_{0:k-1}\}p(y_k/y_{0:k-1}, A_{k,l})}{\sum_{l=1}^{n_k^*} P\{A_{k,l}/y_{0:k-1}\}p(y_k/y_{0:k-1}, A_{k,l})} \quad (35)$$

therefore the posterior density is given by:

$$p(x_k/y_{0:k}) = \sum_{l=1}^{n_k^*} P\{A_{k,l}/y_{0:k}\}p(x_k/y_{0:k}, A_{k,l}) \quad (36)$$

The recurrence relations (28) and (36) constitute conceptual solution of nonlinear, non-Gaussian estimation using finite mixture approximation. When components of a mixture are Gaussians, we can solve integrals in (28) and (33) applying the same ideas as in EKF, DDF and UKF. Thus Gaussian sum filter is implemented as a bank of parallel EKF's, DDF's or UKF's. The major drawback of proposed algorithm is exponential growth of the number of components in a posterior density (36), since $n_k^* = n_{v_k} \cdot n_k^- = n_{v_k} \cdot (n_{d_k} \cdot n_{k-1})$. To solve this problem, after updating the posterior density $p(x_k/y_{0:k})$ we apply a mixture reduction procedure to prevent exponential explosion of the number of mixture components.

B. Mixture reduction

We apply the algorithm which clusters the components of the mixture and replace the cluster with a single Gaussian. The component of the mixture (36) with largest probability $w_{k,j}$ is selected as the principal component. All components that are close to it are clustered, together with the principal component. Closeness is defined using Kullback distance between two Gaussains:

$$D_i^2 = \frac{1}{2}(\hat{x}_c - \hat{x}_i)^T (P_c^{-1} + P_c^{-2})(\hat{x}_c - \hat{x}_i) + \frac{1}{2} tr(P_1^{-1}P_2 + P_1 P_2^{-1} - 2I) \quad (37)$$

where w_c, $\hat{x}_c$ and P_c are the probability, mean and covariance of the principal component and w_i, $\hat{x}_i$ and P_i are the probability, mean and covariance of the ith component. A component for which $D_i^2 < T_{\min}$ is selected as a class member. Threshold $T_{\min}$ defines the acceptable modification of the original distribution (36).

The cluster of components is approximated by a single Gaussian:

$$w_c = \sum_{i \in I_C} w_i \quad (38a)$$

$$\hat{x}_c = \frac{1}{w_c} \sum_{i \in I_C} w_i \hat{x}_i \quad (38b)$$

$$P_c = \frac{1}{w_c} \left(\sum_{i \in I_C} w_i (P_i + \hat{x}_i \hat{x}_i^T) - \hat{x}_c \hat{x}_c^T \right) \quad (38c)$$

where I_C contains the indices of components close to the principle component.

The clustering procedure continues on the remaining components of the original mixture. If the number of components after clustering is bellow the user-defined maximal number $N_{\max}$ the mixture reduction is completed. Otherwise the minimum distance is incremented $T_{\min} = T_{\min} + \Delta T$ and the clustering procedure is repeated. ΔT is selected as a compromise between a number of iterations required and the possibility of clustering more components than necessary.

IV Examples

In this section, we shall give the results of Mackey-Glass chaotic time series prediction using NARX recurrent neural networks trained using EKF, DDF, UKF and GS filters.

A. Gaussian noise

In the first example we considered the single step prediction of Mackey Glass time series corrupted by additive Gaussian noise with standard deviation equal to 0.2 ($SNR \approx 3dB$).

We have used EKF, UKF, DDF, GS_EKF, GS_UKF and GS_DDF to train NARX recurrent network with 6 recurrent inputs, 5 hidden neurons and one output (41 adjustable parameters). After sequential adaptation on 2000 samples (presented only once), single step prediction of next $N = 100$ samples is used to calculate Normalized Root Mean Squared Error (NRMSE):

$$NRMSE = \sqrt{\frac{1}{\sigma^2 N} \sum_{k=1}^{N} (y_k - \hat{y}_k^-)^2} \quad (40)$$

where σ is the standard deviation of clean time series, y_k is the true value of sample at time step k, and $\hat{y}_k^-$ is a prediction of NARX RNN. Note that during training only noisy signal was presented to the network.

In Table1, we give mean and variace of NRMSE, when network was trained using UKF, DDF and EKF.

Table 1. NRMSE of NARX_RNN single step prediction

	UKF	DDF	EKF
mean(NRMSE)	0.3321	0.3370	0.3761
var(NRMSE)	0.06	0.04	0.05

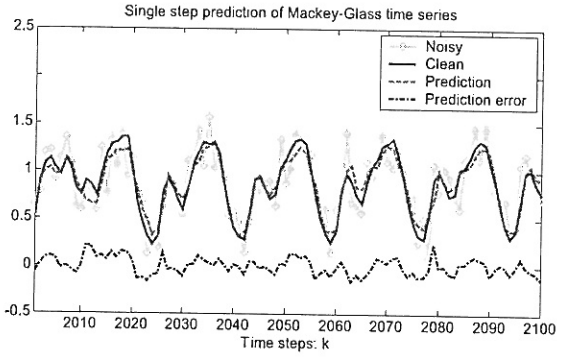

Figure 2. Single step prediction of NARX recurrent network trained by GS_UKF (Gaussian observation noise – SNR=3dB)

In Table 2, the results are obtained for GS_UKF, GS_DDF and GS_EKF. All Gaussian sum filters use a four component Gaussian mixture for state posterior and one component Gaussian mixture of both process and observation noise.

Table 2. NRMSE of NARX_RNN single step prediction

	GS_UKF	GS_DDF	GS_EKF
mean(NRMSE)	0.3198	0.3301	0.3640
var(NRMSE)	0.01	0.05	0.01

B. Non-Gaussian noise

In a second example the additive observation noise was distributed as a Gaussian mixture given by:

$$p(v_k) = \alpha \cdot N(v_k; m_1, \sigma_1^2) + (1-\alpha) \cdot N(v_k; m_2, \sigma_2^2) \quad (41)$$

where $\alpha = 2/3$, $m_1 = -0.2$, $\sigma_1 = 0.05$, $m_2 = 0.4$, $\sigma_2 = 0.1$.

We have considered a long term - iterated prediction of Mackey Glass time series. Note that when EKF, UKF and DDF were applied for training, the observation noise (41) was approximated with Gaussian noise of the same mean and variance. While in the first example the performance of EKF, UKF and DDF was comparable to the corresponding Gaussian sum filters, in case of non-Gaussian observation noise they failed in a long term prediction (Figure 3).

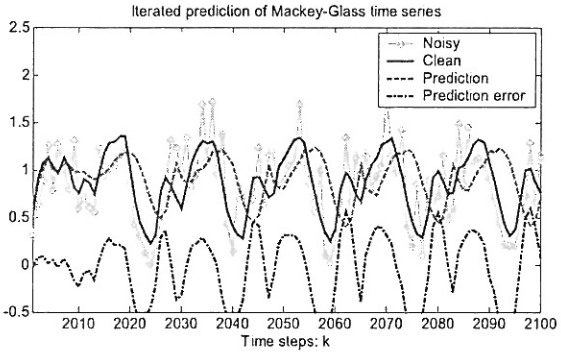

Figure 3. Iterated prediction of NARX recurrent network trained by DDF (non-Gaussain observation noise)

Performance of Gaussian sum filters for non-Gaussian noise is documented in Table 3. Figure 4 shows a typical result of iterated long term prediction for Gaussian sum filter implemented as a bank of parallel DDF's. All Gaussian sum filters use a five component Gaussian mixture for state posterior, Gausian process noise and two component Gaussian mixture of observation noise.

Table 3. NRMSE of NARX_RNN iterated prediction

	GS_UKF	GS_DDF	GS_EKF
mean(NRMSE)	0.2005	0.1429	0.2317
var(NRMSE)	0.0013	0.005	0.052

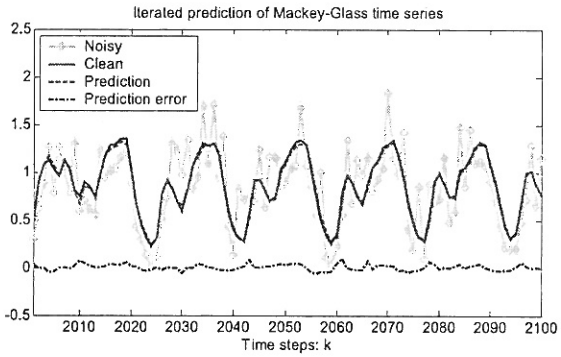

Figure 4. Iterated prediction of NARX recurrent network trained by GS_DDF (non-Gaussain observation noise)

V CONCLUSIONS

We consider the problem of recurrent neural network training as the Bayesian state estimation. Proposed algorithm uses Gaussian sum (GS) filter for nonlinear, non-Gaussian estimation of network outputs and synaptic weights. We show that GS filter can be implemented as a bank of parallel Bayesian filters (EKF's, UKF's, DDF's or other).

The performance of proposed algorithm and other Bayesian filters is compared in noisy chaotic time series prediction. In case of Gaussian noise the performances of Gaussian and Gaussian sum filters in RNN training are comparable, where as in case of non-Gaussian noise, Gaussian sum filters are superior.

VI REFERENCES

1. Alspach, D. L. and Sorenson, H. W., "Nonlinear Bayesian estimation using Gaussian Sum Approximations," IEEE Trans. on Automatic Control, vol. Ac-17, No.4, August 1972
2. Julier, S. J., & Uhlmann, J. K., A new extension of the Kalman filter to nonlinear systems. Preoceedings of AeroSense: The 11th international symposium on aerospace/defence sensing, simulation and controls, Orlando, FL, 1997.
3. Nørgaard, M., Poulsen, N. K., & Ravn, O., Advances in derivative free state estimation for nonlinear systems, Technical Report, IMM-REP-1998-15, Department of Mathematical Modelling, DTU, revised April 2000.
4. Sorenson, H. W., and Alspach, D. L., "Recursive Bayesian estimation using Gaussian Sums," Automatica, vol. 7, pp 465-479, 1971
5. Todorović, B., Stanković, M., Moraga, C.: "Extended Kalman Filter trained Recurrent Radial Basis Function Network in Nonlinear System Identification," in *Proc. of ICANN 2002*, Spain, LNCS 2415, pp 819-824, Springer, August 2002
6. Todorović, B., Stanković, M., Moraga, C.: "On-line Learning in Recurrent Neural Networks using Nonlinear Kalman Filters," in *Proc. of ISSPIT 2003*, Darmstadt, Germany, December 2003
7. van der Merwe, R. & Wan, E. AS., Efficient Derivative-Free Kalman Filters for Online Learning, in *Proc.* of ESSAN, Bruges, Belgium, April 2001.
8. Williams, R.J. & Zipser, D., "Gradient-based learning algorithms for recurrent connectionist networks," TR NU_CCS_90-9. Boston: Northeastern University, CCS, 1990
9. Williams, R.J.: "Some observations on the use of the extended Kalman filter as a recurrent network learning algorithm," TR NU_CCS_92-1. Boston: Northeastern University, CCS, 1992