Pre-Trained Neural Networks used for Non-Linear State Estimation

Bayramoglu, Enis; Andersen, Nils Axel; Ravn, Ole; Poulsen, Niels Kjølstad

Published in:
The tenth International Conference on Machine Learning and Applications

Link to article, DOI:
10.1109/ICMLA.2011.118

Publication date:
2011

Pre-Trained Neural Networks used for Non-Linear State Estimation

Enis Bayramoğlu, Nils Axel Andersen, Ole Ravn
Department of Electrical Engineering
Technical University of Denmark
Elektrovej DTU Building 326 DK-2800 Kongens Lyngby
{eba,naa,or}@elektro.dtu.dk

Niels Kjølstad Poulsen
Department of Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads DTU Building 321
DK-2800 Lyngby
nkp@imm.dtu.dk

***Abstract*— The paper focuses on nonlinear state estimation assuming non-Gaussian distributions of the states and the disturbances. The posterior distribution and the aposteriori distribution is described by a chosen family of paramtric distributions. The state transformation then results in a transformation of the paramters in the distribution. This transformation is approximated by a neural network using offline training, which is based on monte carlo sampling. In the paper, there will also be presented a method to construct a flexible distributions well suited for covering the effect of the non-linearities. The method can also be used to improve other parametric methods around regions with strong non-linearities by including them inside the network.**

I. INTRODUCTION

In Engineering, especially since the advent of electronics, estimation of the internal state of a dynamic system has been an important challenge . Application of the principles of Bayesian statistics to this problem is the subject of Bayesian filtering field. This field has seen a recent research explosion, owing to its optimality under the assumption of accurate system models. However, the optimal solution brought by the Bayes theorem involves analytically intractable equations for the general non-linear filtering problem.

The application of Bayesian filtering to real problems using electronic processors requires some approximations to the exact solution. The research in this field focuses on various approximation methods each of them with some trade-offs in computational resources, accuracy under different conditions, ease of implementation etc. Chen [1] provides a recent comprehensive survey of the various approaches.

It is possible to model the involved Bayesian probability distributions in parametric or non-parametric methods, while commonly a hybrid is used. Parametric methods approximate the distribution using an analytic distribution described by some parameters, or they solve the filtering problem analytically under certain simplifications. The well known Kalman filter(KF) [2] is probably the most significant example in this category and it is the exact solution for the linear, Gaussian prior case. KF assumes that the system state vector is normal distributed, and it estimates the parameters of this distribution. Among the many extensions of KF The unscented Kalman filter [3] is an important recent method in this category, where the parameters of the Gaussian is calculated through so called sigma points, propagated through the system non-linearity. Daum filters [4] are exact solutions to a restricted class of non-linear systems, and they use a member of the exponential family to represent the state distribution. Projection filters [5] on the other hand provide an approximation to exact non-linear filters. They can use a variety of parametric distributions.

The non-parametric methods do not make an assumption about the shape of the distribution. Most important in this category are the particle filters (E.g [6], [7]). They approximate the actual distribution with a set of state samples. These samples are propagated through the system along with resampling to keep track of the actual distribution. Non-parametric methods give better results under strong non-linearities compared to parametric methods. However, their computational requirements are higher and they grow exponentially with state dimensions, in contrast to usually polynomial growth with parametric methods.

Hybrid methods also exist such as mixture Kalman filter [8], which is basically a bank of extended Kalman filters borrowing resampling from particle filters. Particle filters can also be improved using Rao-Blackwellization [9]. In this approach, the particles are augmented with parametric distributions along dimensions, where an analytic solution exists or a parametric method would be accurate enough.

Our method can be used with any parametric distribution, and it approximates the function which gives the set of parameters that best fits it to the actual distribution. A feed-forward neural network for the state transition and another for the measurement are trained to give a matched set of filters. We train the neural networks using random samples of the input distribution parameters, control inputs, measurements and system states. Therefore, as the training iterations go to infinity, the resulting networks provide a global fit to the actual distribution, in contrast to most parametric methods that use local properties of the system non-linearities.

The Gaussian requirement is also lifted by using neural networks. We also describe a way of obtaining distributions with flexible parametric shapes to be used with the neural networks.

Our method is feasible for lower order systems and in a custom but restricted subset of the parameter space. Other parametric methods can be used as blocks in the neural networks to alleviate this and obtain a performance superior to both. The desired global properties of the well-known filters such as stability can be combined with the ability of the neural networks to handle wild regional non-linearities.

II. Formulation

A general discrete system modelled as a Markov chain is usually described by (1) in the state-space representation.

$$\begin{aligned} \boldsymbol{x}_{n+1} &= f(n, \boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{d}_n) \\ \boldsymbol{y}_n &= g(n, \boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{v}_n) \end{aligned} \tag{1}$$

In (1), $\boldsymbol{x}_n$ is the state vector at sample n, $\boldsymbol{y}$ is the meausrement, $\boldsymbol{u}_n$ is the control input, $\boldsymbol{d}_n$ and $\boldsymbol{v}_n$ are the process and measurement noises and $f()$ and $g()$ are the state transition and measurement functions respectively.

In Bayesian filtering, $\boldsymbol{x}_n$ is treated as a random vector with the probability density function (pdf) $p(\boldsymbol{x}_n)$. The aim of the filter is to find the distribution of the state vector at time step n given the measurements up to n ($p(\boldsymbol{x}_n|\mathcal{Y}_n)$, $\mathcal{Y}_n = \{\boldsymbol{y}_1 \dots \boldsymbol{y}_n\}$).

This is usually achieved iteratively over time steps with the assumption that the initial distribution $p(\boldsymbol{x}_0)$ is known. Then, first, $p(\boldsymbol{x}_n|\mathcal{Y}_{n-1})$ is given by;

$$p(\boldsymbol{x}_n|\mathcal{Y}_{n-1}) = \int p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{x}_{n-1}|\mathcal{Y}_{n-1})d\boldsymbol{x}_{n-1} \tag{2}$$

and $p(\boldsymbol{x}_n|\mathcal{Y}_n)$ is obtained from $p(\boldsymbol{x}_n|\mathcal{Y}_{n-1})$ through;

$$p(\boldsymbol{x}_n|\mathcal{Y}_n) = \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\mathcal{Y}_{n-1})}{\int p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\mathcal{Y}_{n-1})d\boldsymbol{x}_n} \tag{3}$$

(2) is governed by the state transition function (f) as $p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$ is obtained from it. Similarly, (3) is governed by the measurement function (g) as $p(\boldsymbol{y}_n|\boldsymbol{x}_n)$ is obtained from it. Approximations to $p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$ and $p(\boldsymbol{x}_n|\boldsymbol{x}_n)$ are needed, because in general the sufficient statistics of these pdfs are infinite dimensional, so we would need an unbounded amount of parameters to describe them, even if we assume a $p(\boldsymbol{x}_0)$ with finite sufficient statistics.

In order to approximately represent $p(\boldsymbol{x}_n|\mathcal{Y}_n)$ (or similarly $p(\boldsymbol{x}_n|\mathcal{Y}_{n-1})$), in our method we choose a parametric distribution, denoted as $q(\boldsymbol{x}_n|\boldsymbol{\alpha}_{n|n})$. The vector $\boldsymbol{\alpha}_{n|n}$ parameterizes the distribution q and in this context we choose $\boldsymbol{\alpha}_{n|n}$ such that $q(\boldsymbol{x}_n|\boldsymbol{\alpha}_{n|n})$ is as close to $p(\boldsymbol{x}_n|\mathcal{Y}_n)$ as possible, in some appropriate distance measure.

Note that, for instance, KF based filters use a Gaussian distribution with the parameters $\boldsymbol{\alpha} = \{\boldsymbol{\mu}, C\}$ (mean and covariance). The alternatives for our method span any parametric distribution, as long as it is described by a reasonably few number of parameters. In section IV, we propose applying a parameterized transformation on a Gaussian to obtain a custom flexible distribution, which is collectively parameterized by the parameters of the Gaussian appended with those of the transformation ($\boldsymbol{\alpha} = \{\boldsymbol{\mu}, C, \boldsymbol{\alpha}_{trans}\}$). Another example is a mixture of two Gaussians as in the given sample application. The mixture is described by the parameters of the individual Gaussians and the mixing coefficient ($\boldsymbol{\alpha} = \{\boldsymbol{\mu}_1, C_1, \boldsymbol{\mu}_2, C_2, c_m\}$).

The filtering problem for state transition is to find the parameters $\boldsymbol{\alpha}_{n|n-1}$ that best approximate, w.r.t some measure, $p(\boldsymbol{x}_n|\mathcal{Y}_{n-1})$ given $p(\boldsymbol{x}_{n-1}|\mathcal{Y}_{n-1}) = q(\boldsymbol{x}_{n-1}|\boldsymbol{\alpha}_{n-1|n-1})$. The new parameter vector $\boldsymbol{\alpha}_{n|n-1}$ is a function of the old parameters, the control input and possibly the sample number as in (4a). The filtering problem is also similar for measurements. The parameters $\boldsymbol{\alpha}_{n|n}$ that best approximate $p(\boldsymbol{x}_n|\mathcal{Y}_n)$ given $p(\boldsymbol{x}_n|\mathcal{Y}_{n-1}) = q(\boldsymbol{x}_n|\boldsymbol{\alpha}_{n|n-1})$ will be a function of the prior parameters, the measurement and again possibly the sample number as in (4b).

$$\boldsymbol{\alpha}_{n+1|n} = h(n, \boldsymbol{\alpha}_{n|n}, \boldsymbol{u}_n) \tag{4a}$$

$$\boldsymbol{\alpha}_{n+1|n+1} = j(n, \boldsymbol{\alpha}_{n+1|n}, \boldsymbol{u}_n, \boldsymbol{y}_n) \tag{4b}$$

To clarify the meaning of the functions $h()$ and $j()$, one can form an analogy with the Kalman filter. In this analogy, $h()$ would be the prediction, or the time update, stage of the Kalman filter. In order to calculate the expected value of the system states at sample n, given the measurements up to $n-1$, $\hat{\boldsymbol{x}}_{n|n-1}$ and the covariance of those states, $P_{n|n-1}$; one uses $\hat{\boldsymbol{x}}_{n-1|n-1}$, $P_{n-1|n-1}$, the system input at sample n, u_n and the state transition matrix at sample n (if the system is time varying), F_n. Collecting $\{\hat{x}_{n|n-1}, P_{n|n-1}\}$ together as $\boldsymbol{\alpha_{n|n-1}}$, the parameters of the distribution, one can summarize this calculation as (4a). Fig. 1 summarizes these relationships graphically. The analogy follows similarly between the KF measurement update and (4b). The representation in (4a) and (4b) could be generalized to non-linear systems, where $h()$ and $j()$ do not in general have analytic expressions.

Note that, in the Kalman filter analogy, the solution is exact, in the sense that if the prior distribution $p(\boldsymbol{x}_0)$ is Gaussian, the system is linear and the process and measurement noises are also Gaussian; $p(\boldsymbol{x}_n|\mathcal{Y}_n)$ will be Gaussian for all n. We do not require this for (4a) and (4b). If the state distribution does not remain in the family of distributions described by $q(\boldsymbol{x}_n|\boldsymbol{\alpha})$, we define that $h()$ and $j()$ return the parameters that best approximate (in terms of cross entropy) the actual distribution within the assumed family.

We propose that these two functions could be approximated by neural networks. Training of these networks is not a trivial task however. $h()$ and $j()$ are not easily computed for even single values. Section III describes our training approach.

III. Training

The neural networks should be trained such that the parametric pdf described by the output parameters is as close to the actual pdf produced by the non-linear process. We used the cross-entrophy as the distance between the two distributions. The aim is to minimize the integral given in (5).

$$D = -\int_X p(\boldsymbol{x}_{new}|\boldsymbol{\alpha}_{old}, \boldsymbol{i}_{extras}) \log\left(q(\boldsymbol{x}_{new}|\boldsymbol{\alpha}_{new})\right) d\boldsymbol{x_{new}} \tag{5}$$

Here, $\boldsymbol{i}_{extras}$ is used to represent the extra inputs such as the sample number(n), control inputs ($\boldsymbol{u}$) and measurements($\boldsymbol{y}$). We have defined the network error function to be the expected value of this distance measure D. The expectation is taken over an assumed distribution of

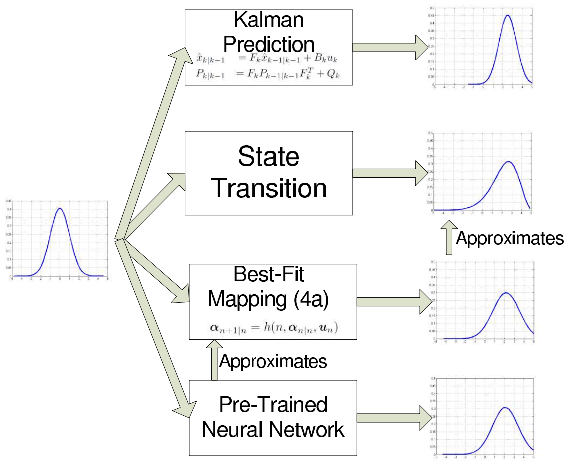

Kalman Prediction
$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k$
$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$
State Transition
Approximates
Best-Fit Mapping (4a)
$\boldsymbol{\alpha}_{n+1|n} = h(n, \boldsymbol{\alpha}_{n|n}, \boldsymbol{u}_n)$
Approximates
Pre-Trained Neural Network

Fig. 1: Analogous to the Kalman filter prediction update, a pre-trained neural network receives the parameters for the distribution for the previous state, and outputs those for the current state.

old parameters and extra inputs. This error, as a function of network weights vector $\boldsymbol{w}$ is given in (6). Note that $\boldsymbol{\alpha}_{nnout}(\boldsymbol{w})$ is the network estimate, so it is a function of the network weights.

$$\begin{aligned} E(\boldsymbol{w}) =& \mathbb{E}\{D\left(\boldsymbol{\alpha}_{old}, \boldsymbol{i}_{extras}, \boldsymbol{w}\right)\} \\ E(\boldsymbol{w}) =& \int_A \int_I \int_X p(\boldsymbol{x}_{new}|\boldsymbol{\alpha}_{old}, \boldsymbol{i}_{extras}) p(\boldsymbol{\alpha}_{old}, \boldsymbol{i}_{extras}) \\ & \log\left(q(\boldsymbol{x}|\boldsymbol{\alpha}_{nnout}(\boldsymbol{w}))\right) d\boldsymbol{x}_{new}\, d\boldsymbol{i}_{extras}\, d\boldsymbol{\alpha}_{old} \end{aligned} \tag{6}$$

The calculation of this error function, even for a single network state, is a computational challenge. Therefore, we use an approximation inspired by the Monte Carlo method. The approximation procedure differs slightly for state transition and measurement.

a) State Transition: We draw a sample from the distribution of input parameters $\boldsymbol{\alpha}_s$ and the control inputs $\boldsymbol{i}_s$, the extra input in this case. We then draw a sample $\boldsymbol{x}_s$ from the pdf described by $\boldsymbol{\alpha}_s$. We propagate $\boldsymbol{x}_s$ through the state transition $f(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{d}_n)$ given in 1 using $\boldsymbol{i}_s$ and a sample of $\boldsymbol{d}_n$ to obtain $\boldsymbol{x}_{snew}$. The approximation to E is given in (7).

b) Measurement: We again draw a sample from the distribution of input parameters $\boldsymbol{\alpha}_s$ and a sample $\boldsymbol{x}_s$ from the pdf described by $\boldsymbol{\alpha}_s$. Applying the measurement function $g(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{v}_n)$ on $\boldsymbol{x}_s$ and a sample of $\boldsymbol{v}_n$ we obtain a sample measurement $\boldsymbol{i}_s$, the extra input. The approximation to E is again given in (7) with $\boldsymbol{x}_{snew} = \boldsymbol{x}_s$.

The measurement and state transition cases differ, since there is no general way to draw a sample from the posterior distribution given by the prior distribution parameters and the measurement. We instead generate a measurement sample from the state sample using the measurement equation.

$$\hat{E} = -log\left(q\left(\boldsymbol{x}_{snew}|\boldsymbol{\alpha}_{nnout}(\boldsymbol{w}, \boldsymbol{\alpha}_{old}, \boldsymbol{i}_{extras})\right)\right) \tag{7}$$

It can be shown that the expected value of $\hat{E}$ equals E for both cases. Due to the coarse approximation to E we use, We have chosen to use gradient descend for training the networks. The training iteration i is given as in (8). $\boldsymbol{w}$ is the network weights as a vector and k is the training weight varying over iterations.

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - k(i)\frac{d\hat{E}}{d\boldsymbol{w}_i} \tag{8}$$

k starts with a relatively high value, and it is decreased a few orders of magnitude during the course of the training. This ensures that the final stages of the training averages over a large sample of $\hat{E}$, improving the approximation. As k gets sufficiently small, the steps of the network weights averaged over samples will approach those that would be obtained by using the actual E.

This training algorithm favours the parameter regions where the corresponding actual distribution has a low entrophy. Fitting the parameters in these regions have a larger impact on E compared to the high entropy regions. Therefore the training algorithm so far spends most of the learning resources on these regions. This problem can be solved by weighting E as in (9) to improve learning in desired regions.

$$E = \mathbb{E}\{W(\boldsymbol{\alpha}_{old}, \boldsymbol{i}_{extras}) D(\boldsymbol{\alpha}_{old}, \boldsymbol{i}_{extras})\} \tag{9}$$

A. Training Stability

The training algorithm is, in essence, a stochastic gradient descent. Kiwiel [10] analyzes the convergence of stochastic gradient descent algorithms. Since the objective function $E(\boldsymbol{w})$ given in (6) is not generally quasiconvex with respect to the network weights, convergence to the global minimum can not be guaranteed.

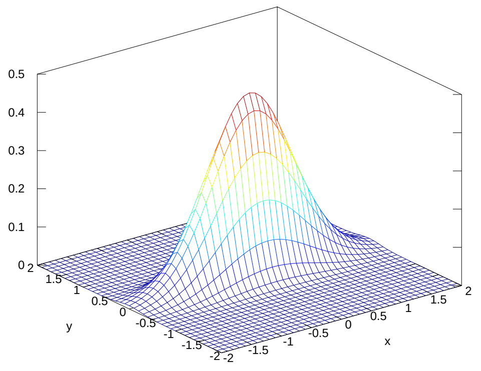
The objective function is a weighted sum of the cross entrophy for the posterior and the estimated distributions, therefore, it has a bounded global minimum as long as the posterior distribution has bounded differential entropy everywhere in the summation region. Although, we have direct information on the prior distribution rather than the posterior distribution, having a prior distribution with a bounded differential entropy will result in such a posterior distribution for a wide range of systems.

Therefore, it can be shown that, with a small enough step size, the stochastic training proposed in this paper will result in convergence to a local minimum of the objective function under conditions that do not impair applicability to practical systems.

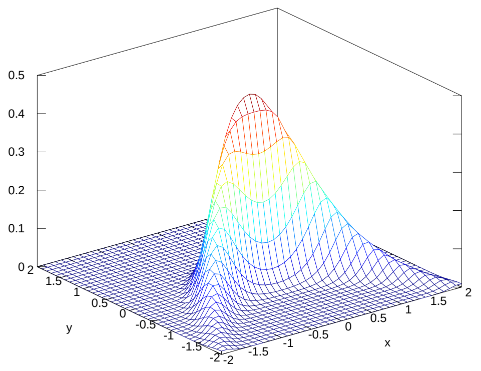
IV. DISTRIBUTIONS

In theory, any parameterized function could be used with our method, as long as it is greater than zero everywhere and its volume converges for the chosen parameter space. However, there is one obstacle. During the calculation of $\hat{E}$ in 7, one needs to calculate the probability density. Therefore, the chosen function needs to be normalized by multiplying it with a normalization constant c. Calculating the derivative of c with respect to the distribution parameters will require too much computation, considering that $\hat{E}$ is a coarse approximation and it is calculated many times.

The numeric calculation of c can be avoided by using a distribution with known c and warping it with a parameterized coordinate transformation. Without losing



(a)



(b)

Fig. 2: (a) The Gaussian described by 11 (b) The resulting distribution after coordinate transformation

much flexibility, the Jacobian of the transformation could be one everywhere so that c remains unchanged. Such a transformation, along with its Jacobian is given in (10). Combining such a transformation with a rotation could provide a wide range of shapes.

$$\begin{aligned} \hat{x} &= x \\ \hat{y} &= f(x, \boldsymbol{\alpha}_t) + y \end{aligned}, J = \left| \begin{bmatrix} 1 & 0 \\ \frac{df(x,\boldsymbol{\alpha}_t)}{dx} & 1 \end{bmatrix} \right| = 1 \tag{10}$$

In order to obtain a blurred arc shape, one could transform a Gaussian with the transformation above using $f(x, \boldsymbol{\alpha}_t) = ax^2$. Combined with the rotation this would result in an arc with desired curvature and angle. A warped Gaussian example is provided in Fig. 2. Fig. 2a is the original gaussian described by (11). Fig. 2b is the distribution obtained by warping this with (10) using $f(x, \boldsymbol{\alpha}_t) = 0.5x^2$. The neural networks could be used to estimate the parameters of such a distribution.

$$\mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{9} \end{bmatrix} \right) \tag{11}$$

V. INTEGRATION WITH OTHER FILTERS

Our approach is meant to work on its own for lower order systems, and in a restricted region of the parameter space. But it can easily be integrated with existing estimators to lift both of these restrictions.

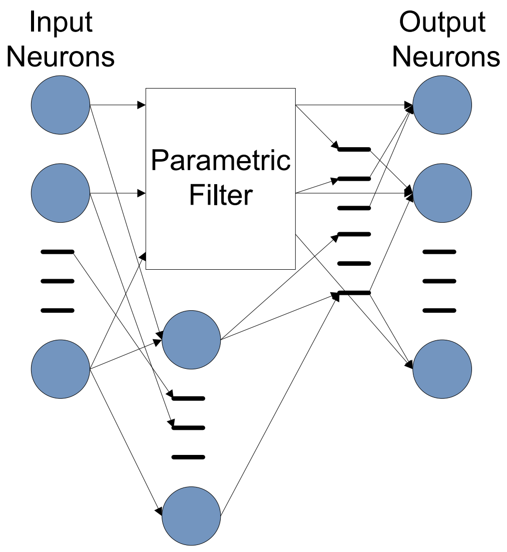


Fig. 3: Integration of the neural state estimator with a guest parametric filter. The guest filter receives the raw inputs and its outputs are directly propagated to the network output. The linear neurons at the output layer are connected also to the hidden layers of the neural network.

Deterministic and parametric filters such as the extended Kalman filter or the unscented Kalman filter can be inserted in the neural network as a block in the first layer as in Fig. 3. In this figure, the guest filter receives a set of raw inputs and its outputs are connected directly to the output neurons as well as the hidden layers. the output neurons are assumed to have a linear activation function. The neurons can be thought applying a correction to the guest filter.

The network structure must be modified as in Fig. 4 so that the neurons do not interfere with the output in the regions that they are not trained for. The neural suppressor block modifies all the neural intervention to the output. It could simply be a multiplication with the assumed distribution for the network inputs used during training. This way, one can use a preferred filter while improving it in a certain region to better reflect the non-linearities.

This method could be integrated with the particle filters to generate a hybrid filter. A bank of neural networks can be used with resampling to obtain a mixture filter. On the other hand, based on Rao-Blackwellization, along appropriate dimensions each particle could be augmented with a parametric distribution, whose parameters are estimated by neural networks.

VI. SAMPLE APPLICATION

We have chosen a non-linear transformation of a single variable to demonstrate the method. The transformation is given in (12). We approximate the pdf of the state variable x by a Gaussian, in order to be able to compare the performance of the neural networks with those of an extended Kalman filter and an unscented Kalman filter. Therefore the pdf has 2 parameters, the mean and the

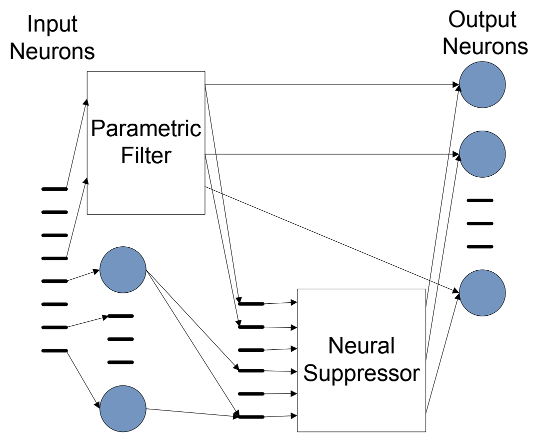


Fig. 4: The suppression block added in this figure is necessary, since the network is trained for a region of the input space. Outside this region the neurons constitute disturbance and their outputs need to be suppressed.

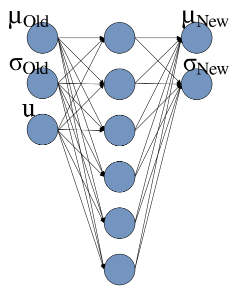


Fig. 5: The network configuration used for the sample application. The inputs are the parameters for the Gaussian distribution and the system control input. The outputs are the parameters for the estimated Gaussian distribution.

standard deviation of the Gaussian (μ, σ). A simple neural network for estimating the parameters is given in Fig. (5). The inputs are the 2 initial distribution parameters and the control input. The outputs are the 2 transformed distribution parameters.

$$x_{n+1} = x_n + p\tanh(\frac{x_n}{w}) + u_n + d_n \qquad (12)$$
$$w = 0.2, p = 0.4, d_n \sim \mathcal{N}(0, 0.1)$$

The assumed initial distribution for parameters are all independent with $\mu \sim \mathcal{N}(0, 2)$, $\sigma \sim \mathcal{N}0.7, 0.4$ and $u_n \sim \mathcal{N}(0, 2)$. This simple network is trained with approximately 3×10^7 iterations, taking around 5 minutes on a 3.4 GHz AMD Phenom^{TM}2 X4 Processor.

The results with the simple network are given in Fig. 6 and 7 along with those of a network containing 15 neurons at the hidden layer for comparison. In Fig. (6), the network inputs are (μ, σ, u) = (0, 0.5, 0). The inputs for Fig. 7 are (μ, σ, u) = (0.3, 0.6, −0.2).

Comparing the two sets of figures, it is clear that the approximation is enhanced using more neurons. The real interesting problem here is the time evolution of the neural network outputs when used as an estimator for a system recursively. For this purpose, we couple the state transition

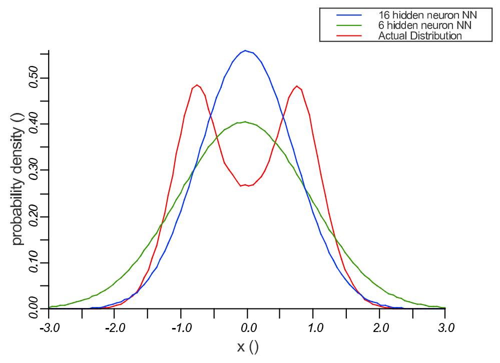


Fig. 6: The output of the simple network in Fig. 10 (green) and the output from a network with 16 neurons in its hidden layer (blue) for the inputs (μ, σ, u) = (0, 0.5, 0), resulting in a bimodal actual output distribution (red), due to the transformation non-linearity.

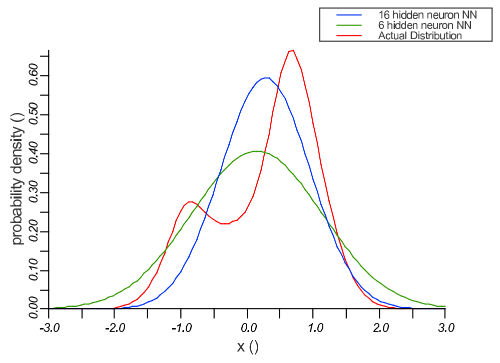


Fig. 7: The outputs of the same networks as in Fig. 6, this time for the inputs (μ, σ, u) = (0.3, 0.6,−0.2)

in (12) with the simple measurement equation given in (13). We further apply the simple estimated state ($\hat{x_n}$) feedback control law given in (14) to keep the system state near the non-linearity around 0. We have used the Kalmtool[11] toolbox to simulate the system and the estimators to obtain a comparison. Note that for the data update of neural network filters, a conventional Kalman filter data update is used since the measurement equation is purely linear.

$$y_n = x_n + v_n \qquad (13)$$
$$v_n \sim \mathcal{N}(0, 2)$$

$$u_n = \hat{x_n} \qquad (14)$$

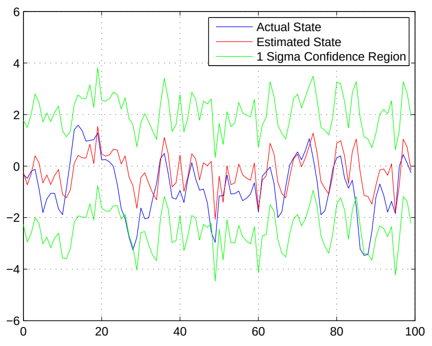


Fig. 8: The simulation results for the system described by the state transition equation (2), the measurement equation (13) and the control law (14) using a neural network estimator for state feedback.

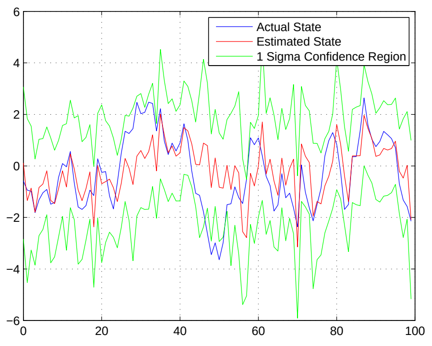


Fig. 9: The simulation results for the same system as in Fig. 8, using an unscented Kalman filter instead of the neural network.

Fig. 8 shows the evolution of the system state along with its estimate over 100 time steps using the neural network with 15 hidden neurons. For comparison, Fig. 9 shows the same plot while the estimator is an unscented Kalman filter. Note how the neural network estimator keeps the system state closer to 0. Since the simulations are probabilistic in nature, we have performed runs with 100000 time steps to obtain some performance metrics. Since the feedback control is trying to keep the system state at 0, we have chosen the RMS (root mean square) of the system state as the first metric. The second metric is the average log-likelihood (ALL) of the real system state on the distribution proposed by the estimator. Table I compares these metrics.

These simulations show that the neural network state estimator behaves in a stable manner for this system. The performance metrics show that the neural network estimator performs better than both the EKF and the UKF even though it is also approximating the state distribution with a Gaussian distribution.

Estimator	RMS	ALL
Neural Network	1.2274	-1.2661
Unscented Kalman Filter	1.2399	-1.2915
Extended Kalman Filter	1.2521	-1.2941

TABLE I: Comparison of performance metrics

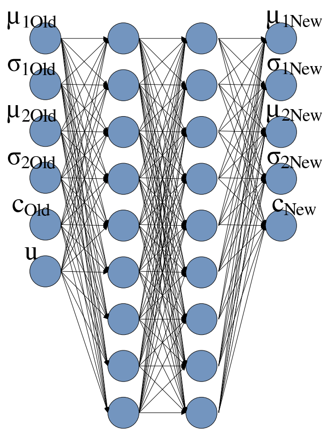


Fig. 10: The network configuration used for estimating the state distribution using a mixture of two Gaussians. The inputs are the parameters for the mixture of two Gaussians and the system control input. The outputs are the parameters for the estimated distribution.

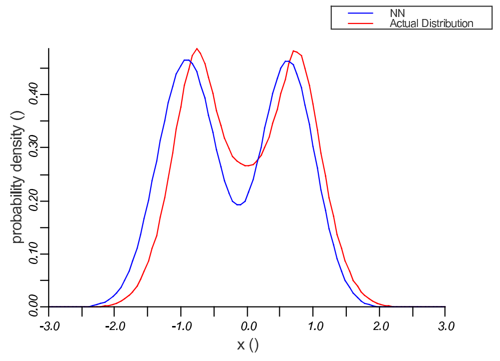


Fig. 11: The output of the neural network given in Fig. 10. The input distribution is the same as in Fig. 6

In order to further show the potential of neural networks with non-Gaussian distributions, we have also trained the neural network shown in Fig. 10, to estimate the system state distribution using a mixture of two Gaussians. In this case the distribution has 5 parameters; the means and the standard deviations of the Gaussians (μ_1, σ_1, μ_2, σ_2) and the mixing coefficient c. Fig. 11 shows the estimate of the neural network for the same input distribution as in Fig. 6. This shows that the neural networks can achieve stable learning also with non-Gaussian distributions.

VII. CONCLUSIONS

Tracking generic probability density functions under non-linear transformations and posterior calculations is an analytically intractable problem. In order to cope with this intractability, parametric methods most often rely on gaussian approximations calculated from values or derivatives of the non-linear functions at specific points. Our method aims to lift both restrictions here. It is applicable to non-Gaussian distributions, and it produces a result asymptotically based on the values of the non-linear functions at all the points under the initial distribution. The heavy computation required by this property is performed offline, during training.

We have shown in this paper that the neural network estimators can be trained to be stable even inside a feedback control loop. Our simulations results further show that the neural network estimator performs better than the EKF and the UKF for that example, even though it is approximating the state distribution with a simple Gaussian.

Parametric filters usually behave well in many regions of the parameter space, where the non-linearities are mild. However, they produce unreliable results close to strong non-linearities. Our method can be used to improve other filters in those regions by including them as a block inside the network. The neurons can be suppressed outside those regions to avoid interference.

The parameter estimation problem is a very sound application for the neural networks, because the approximated function is well defined and it is possible to calculate it. Therefore, the available data for training is unbounded, and overfitting is not a possible problem. By simply increasing the number of neurons and synapses, one can improve the accuracy. The fact that the training is offline and it is easily parallelizable enables the use of very complex networks.

Once the generic training algorithms are implemented, it is very fast to apply the method to a given system. It simply requires the state transition and measurement functions of the system, without even the need to calculate derivatives.

References

[1] Z. Chen, “Bayesian filtering: From kalman filters to particle filters, and beyond,” McMaster University, Tech. Rep., 2003.

[2] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME Journal of Basic Engineering*, no. 82 (Series D), pp. 35–45, 1960.

[3] S. Julier and J. Uhlmann, “A new extension of the kalman filter to nonlinear systems,” in *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997.

[4] F. Daum, “Solution of the zakai equation by separation of variables,” *IEEE Transactions on Automatic Control*, vol. 32, no. 10, pp. 941–943, 1987.

[5] D. Brigo, B. Hanzon, and F. LeGland, “A differential geometric approach to nonlinear filtering: the projection filter,” *IEEE Transactions on Automatic Control*, vol. 43, no. 2, pp. 247–252, 1998.

[6] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. J. Nordlund, “Particle filters for positioning, navigation, and tracking,” *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 425–437, 2002. [Online]. Available: http://dx.doi.org/10.1109/78.978396

[7] H. Tanizaki, “Nonlinear and non-gaussian state space modeling using sampling techniques,” *Annals of the Institute of Statistical Mathematics*, vol. 53, no. 1, pp. 63–81, 2001.

[8] R. Chen and J. S. Liu, “Mixture kalman filters,” *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, vol. 62, no. 3, pp. 493–508 and 2 680 693, 2000.

[9] J. S. Liu and R. Chen, “Sequential monte carlo methods for dynamic systems,” *Journal of the American Statistical Association*, vol. 93, pp. 1032–1044, 1998.

[10] K. C. Kiwiel, “Convergence and efficiency of subgradient methods for quasiconvex minimization,” *Mathematical Programming*, vol. 90, pp. 1–25, 2001, 10.1007/PL00011414. [Online]. Available: http://dx.doi.org/10.1007/PL00011414

[11] E. Bayramoglu, S. Hansen, O. Ravn, and N. K. Poulsen, “Derivative free filtering using kalmtool,” presented at: 13th International Conference on Information Fusion, FUSION 2010 : Edinburgh, UK, 2010.