Nonlinear Bayesian Filters for Training Recurrent Neural Networks

Ienkaran Arasaratnam and Simon Haykin

Cognitive Systems Laboratory
Department of Electrical & Computer Engineering
McMaster University
Hamilton, ON, L8S 4K1
aienkaran@grads.ece.mcmaster.ca, haykin@mcmaster.ca

Abstract. In this paper, we present nonlinear Bayesian filters for training recurrent neural networks with a special emphasis on a novel, more accurate, derivative-free member of the approximate Bayesian filter family called the cubature Kalman filter. We discuss the theory of Bayesian filters, which is rooted in the state-space modeling of the dynamic system in question and the linear estimation principle. For improved numerical stability and optimal performance during training period, a number of techniques of how to tune Bayesian filters is suggested. We compare the predictability of various Bayesian filter-trained recurrent neural networks using a chaotic time-series. From the empirical results, we conclude that the performance may be greatly improved by the new square-root cubature Kalman filter.

1 Introduction

Neural networks are an important tool in the modern engineer's kit of problem-solvers. They essentially use flexibly parameterized classes of nonlinear functions to approximate input-desired output relations. The parameters, hereafter called *weights*, are unknown to be determined. In the supervised training mode, a training data set is provided with a set of examples, each of which has a distinct input/desired-output. Hence, the objective of the training algorithm is to find these weights so as to match the training data set as closely as possible.

Specifically, recurrent neural networks (RNNs) have had successes in areas such as dynamic system identification [13], nonlinear prediction [7] and control [22]. They are capable of storing temporal dependencies spanning a number of time steps between inputs and desired outputs. To train RNNs, we use the gradient descent method. Here, the gradient descent method is applied in a truncated backpropagation through time setup to compute derivatives of a cost function with respect to weights meaningfully [17], [27]. In the truncated backpropagation through time setup, internal states of a RNN at current time are computed by unfolding recurrent loops to some steps backward in time and propagating states corresponding to that time through the unfolded network. The gradient descent method for training RNNs suffers from the following limitations:

(i) Poor convergence (ii) Recency effect (tendency for recent weight update to cause RNNs to forget what they have learned in the past) (iii) Vanishing gradient effect (inability of the gradient method to propagate errors backward deeply). The latter is severe when we use a highly time-correlated signal as the training sequence (e.g., speech signal) [3].

To accelerate the rate of convergence, we may use second-order methods using the Hessian information. Unfortunately, they do not always improve the performance significantly due to Hessians that are intrinsically ill-conditioned. Moreover, their converge rate relies on an accurate evaluation of successive descent directions, which seems meaningful only in a batch of examples. However, updating of weights on an example-by-example (online) basis is preferred when the training data set (i) exhibits non-stationary behavior or (ii) is difficult/expensive to obtain before the training starts. This is where we seek the so-called Bayesian filters.

Bayesian filters utilize Bayes' rule to update weight statistics. To be specific, Bayesian filters track the evolving conditional probability density of weights, given a history of examples. In the recent past, there has been a surge of interest in utilizing nonlinear Bayesian filters for training RNNs [9], [19]. Specifically, they provide the following benefits:

- Bayesian filter-based training algorithms converge more rapidly than the gradient descent method.
- They are well-suited to handle noisy and nonstationary training data.
- The second-order information encoded in the filter-estimated error covariance can often be used to prune the network structure [25].
- Last but by no means least, we mention the benefit of using them in neurocontroller training. The Physics-based models of plants including various non-differential submodels due to dead zones, look-up tables etc., are already in place in an industry. In this case, we may directly utilize derivative-free Bayesian filters for neurocontroller training rather than replacing them with less accurate neural networks using the principle of system identification [19], [26].

Although Bayesian filters achieve better results in fewer iterations than the gradient descent method, this does not necessarily translate to less training time. Of course, their complexity is in the order of w^3 instead of w, where w is the number of weights. As a practical cure, algorithmic implementations included various clever groupings of weights at the expense of 'slightly' degraded performance nearly two decades ago [20], [24]. Fortunately, most of present digital computers are capable of handling this computational demand without seeking any approximative approaches, thanks to the ever-increasing power of computers.

In this paper, one of our objectives is to pique the readers' interest in the Bayesian filter family, namely, the extended Kalman filter (EKF) [1], the central-difference Kalman filter (CDKF) [14] and the *cubature Kalman filter* (CKF) [2], for training networks. However, contributions of this paper mainly lie in how to utilize the CKF, which is novel, more accurate and derivative-free. For example,

– We analyze various Bayesian filters from the optimization perspective. Therein, we go on to prove why the CKF is superior to other filters theoretically and verify it empirically.
– We illustrate how the CKF addresses the following problems: (i) Accommodating a wide range of cost functions, which may be non-differentiable (ii) Vanishing gradient in RNNs: The use of the CKF eliminates the need for unfolding the internal feedback loops of the RNNs in time. The reason is that the CKF is rooted in the philosophy:

Optimization through integration as opposed to differentiation.

– We discuss how to optimize performance of the CKF by tailoring its various hyperparameters.

The rest of the paper is organized as follows: Section 2 introduces a pair of fundamental equations of the optimal nonlinear Bayesian filters for nonlinear filtering problems. We present the theory of the CKF under the Gaussian assumption in Section 3. Section 4 illustrates that the use of the Bayesian filter-based training is equivalent to a second-order optimization method minimizing the regularized sum of squared error cost function with varying degree of accuracies. Therein, we prove theoretically that the CKF estimate is superior to various other Bayesian filters. In Section 5, we suggest a number of possible extensions and refinements of Bayesian filters with a special emphasis on the CKF for an improved and stable performance in training. We present a computer experiment in Section 6, where we compare the predictability of various Bayesian filter-trained RNNs using a chaotic time-series. Finally, in Section 7, we summarize the paper with some insightful remarks.

2 Optimal Bayesian Solution to Filtering Problems

In this section, we present a theoretically relevant optimal Bayesian solution to nonlinear filtering problems and describe its intractability leading to proliferation of many suboptimal solutions. Consider the problem of sequentially estimating a state vector $\mathbf{x}_k \in \mathbb{R}^{n_x}$ that evolves in discrete-time according to the first-order Markov process:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{q}_{k-1} \tag{1}$$

given a mapping function from the hidden state space to the observable space:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{r}_k, \tag{2}$$

where $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^m \to \mathbb{R}^{n_x}$ and $\mathbf{h} : \mathbb{R}^{n_x} \times \mathbb{R}^m \to \mathbb{R}^n$ are known; $\mathbf{u}_k \in \mathbb{R}^m$ is the known input at time k; $\mathbf{z}_k \in \mathbb{R}^n$ is the measurement; $\{\mathbf{q}_{k-1}\}$ and $\{\mathbf{r}_k\}$ are uncorrelated process and measurement Gaussian noise sequences with zero means and covariances Q_{k-1} and R_k, respectively. The pair of equations, namely the *process (state transition) equation* (1) and the *measurement equation* (2) are collectively called the state-space model of the dynamic system.

The Bayesian filter extracts information about the hidden state from noisy measurements. Of course, it computes the posterior density of the state, which provides a complete statistical description of the state at that time, in two basic steps:

- *Prediction,* which involves propagating the old posterior density state one time-step ahead before receiving a new measurement. At the end of this step, we obtain the predictive density using Kolmogorov's forward equation:

$$p(\mathbf{x}_k|D_{k-1}) = \int_{\mathbb{R}^{n_x}} p(\mathbf{x}_{k-1}|D_{k-1})p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_{k-1})d\mathbf{x}_{k-1}, \tag{3}$$

  where $D_{k-1} = \{(\mathbf{u}_i, \mathbf{z}_i), i = 1, 2 \ldots (k-1)\}$ denotes the history of input-measurement pairs up to time $(k-1)$; $p(\mathbf{x}_{k-1}|D_{k-1})$ is the old posterior density at time $(k-1)$ and the transition density $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$ is obtained from the process equation (1).
- *Correction,* which involves updating the predictive density using the new measurement, $\mathbf{z}_k$. Using Bayes' rule, we write the posterior density:

$$p(\mathbf{x}_k|D_k) = \frac{1}{c_k} p(\mathbf{x}_k|D_{k-1})p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{u}_k), \tag{4}$$

  where the normalizing constant

$$c_k = \int_{\mathbb{R}^{n_x}} p(\mathbf{x}_k|D_{k-1})p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{u}_k)d\mathbf{x}_k, \tag{5}$$

  and the measurement likelihood function $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{u}_k)$ is obtained from the measurement equation (2).

From the posterior density (4), any statistic regarding $\mathbf{x}_k$ can be computed. For example, we can provide an optimal estimate and its associated covariance (variance) according to 'some' chosen criterion. The optimal Bayesian filter solution given by (3), (4) and (5) provides a unified recursive approach for nonlinear filtering problems conceptually. However, it is not of direct practical importance for two reasons: (i) For a multidimensional system, we require to compute multi-dimensional integrals involved in (3) and (5) (ii) Even after integrals are computed, it may be difficult to propagate relevant statistics through subsequent time steps.

Consequently, an intense research in the past has resulted in a number of suboptimal solutions to nonlinear filtering problems. *Specifically, in machine learning, suboptimal solutions are considered an advantage because they are unlikely to yield an overfitted solution.* In computational terms, the suboptimal solutions to the posterior density can be obtained using one of two approaches:

1. *Local approach.* Here, we derive nonlinear filters by fixing the posterior density to take *a priori* form. The emphasis on locality allows us to work with a limited set of parameters such as central moments of the probability density rather than the probability density itself. For example, we may assume

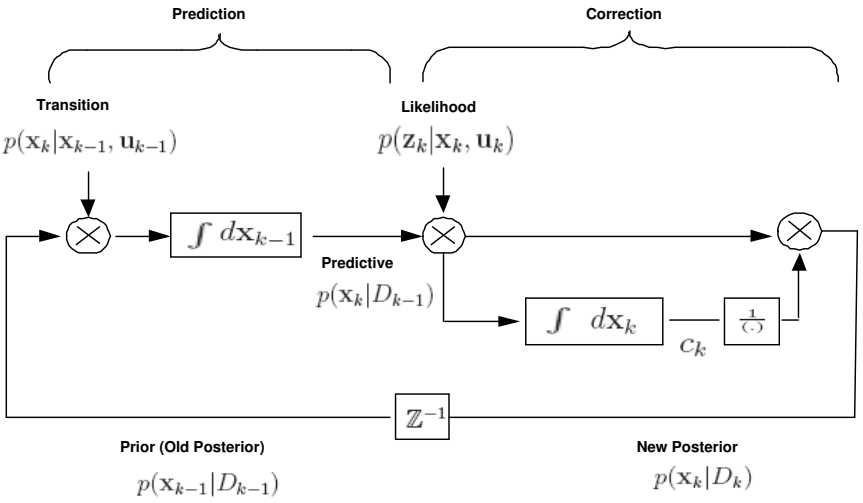


Fig. 1. Signal-flow diagram of the optimal recursive Bayesian filter. ($\mathbb{Z}^{-1}$ denotes the unit time delay operator)

it to be Gaussian, which is completely characterized by mean and variance. Nonlinear filters, namely, the extended Kalman filter (EKF) [1], the central-difference Kalman filter (CDKF) [14], the unscented Kalman filter (UKF) [10], and the cubature Kalman filter (CKF) [2], fall under this first category.

2. *Global approach.* As opposed to the local approach, we do not make any explicit assumption about the posterior density form here. For example, particle filters using Monte Carlo integrations with the importance sampling [4], [5], fall under this second category .

Though the global approaches seem to be more accurate, their accuracy is achieved at the expense of intensive computations. For example, when using particle filters in high-dimensional filtering problems such as training neural networks, a vast majority of particles may be sampled from a non-informative volume of the state-space [12]. Therefore, when the problem dimension increases we use an exponentially increasing number of particles to retain the same accuracy. This is often called the *curse of dimensionality*. In this paper, we focus on supervised training of neural networks, which typically involves a huge number weights to be determined. In this case, we have to be content with a local approach, which makes the design of the filter simple and fast to execute. Next section introduces the CKF using the local approach.

3 Theory of Cubature Kalman Filter

The key assumption of the cubature Kalman filter (CKF) is that the predictive density $p(\mathbf{x}_k|D_{k-1})$ and the filter likelihood density $p(\mathbf{z}_k|D_k)$ are both Gaussian, which eventually leads to a Gaussian posterior density $p(\mathbf{x}_k|D_k)$. Under this assumption, the cubature Kalman filter solution reduces to how to compute their means and covariances more accurately.

3.1 Prediction

In the prediction step, the CKF computes the mean $\hat{\mathbf{x}}_{k|k-1}$ and the associated covariance $P_{k|k-1}$ of the Gaussian predictive density numerically using cubature rules. We write the predicted mean

$$\hat{\mathbf{x}}_{k|k-1} = \mathbb{E}(\mathbf{x}_k|D_{k-1}), \tag{6}$$

where $\mathbb{E}$ is the statistical expectation operator. Substituting (1) into (6) yields

$$\hat{\mathbf{x}}_{k|k-1} = \mathbb{E}\big[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{q}_k|D_{k-1}\big]. \tag{7}$$

Because $\mathbf{q}_k$ is assumed to be zero-mean and uncorrelated with the measurement sequence, we get

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= \mathbb{E}\big[\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})|D_{k-1}\big] \\ &= \int_{\mathbb{R}^{n_x}} \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) p(\mathbf{x}_{k-1}|D_{k-1}) d\mathbf{x}_{k-1} \\ &= \int_{\mathbb{R}^{n_x}} \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \mathcal{N}(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, P_{k-1|k-1}) d\mathbf{x}_{k-1},\end{aligned} \tag{8}$$

where $\mathcal{N}(.,.)$ is the conventional symbol for a Gaussian density. Similarly, we obtain the associated error covariance

$$\begin{aligned}P_{k|k-1} &= \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^T|\mathbf{z}_{1:k-1}] \\ &= \int_{\mathbb{R}^{n_x}} \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \mathbf{f}^T(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \mathcal{N}(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, P_{k-1|k-1}) d\mathbf{x}_{k-1} \\ &\quad -\hat{\mathbf{x}}_{k|k-1} \hat{\mathbf{x}}_{k|k-1}^T + Q_{k-1}.\end{aligned} \tag{9}$$

3.2 Correction

It is known that the innovation process is not only white but also zero-mean Gaussian when the additive measurement noise is Gaussian and the predicted measurement is estimated in the least squares error sense. In this case, we write the predicted measurement density also called the filter likelihood density

$$p(\mathbf{z}_k|D_{k-1}) = \mathcal{N}(\mathbf{z}_k; \hat{\mathbf{z}}_{k|k-1}, P_{zz,k|k-1}), \tag{10}$$

where the predicted measurement and the associated covariance are given by

$$\hat{\mathbf{z}}_{k|k-1} = \int_{\mathbb{R}^{n_x}} \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)\mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, P_{k|k-1})d\mathbf{x}_k \tag{11}$$

$$\begin{aligned} P_{zz,k|k-1} = &\int_{\mathbb{R}^{n_x}} \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)\mathbf{h}^T(\mathbf{x}_k, \mathbf{u}_k)\mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, P_{k|k-1})d\mathbf{x}_k \\ &-\hat{\mathbf{z}}_{k|k-1}\hat{\mathbf{z}}^T_{k|k-1} + R_k. \end{aligned} \tag{12}$$

Hence, we write the Gaussian conditional density of the joint state and the measurement:

$$p\left([\mathbf{x}_k^T \;\; \mathbf{z}_k^T]^T | D_{k-1}\right) = \mathcal{N}\left(\begin{pmatrix} \hat{\mathbf{x}}_{k|k-1} \\ \hat{\mathbf{z}}_{k|k-1} \end{pmatrix}, \begin{pmatrix} P_{k|k-1} & P_{xz,k|k-1} \\ P^T_{xz,k|k-1} & P_{zz,k|k-1} \end{pmatrix}\right), \tag{13}$$

where the cross-covariance

$$P_{xz,k|k-1} = \int_{\mathbb{R}^{n_x}} \mathbf{x}_k\mathbf{h}^T(\mathbf{x}_k, \mathbf{u}_k)\mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, P_{k|k-1})d\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}\hat{\mathbf{z}}^T_{k|k-1}. \tag{14}$$

On the receipt of a new measurement $\mathbf{z}_k$, the CKF computes the posterior density $p(\mathbf{x}_k|D_k)$ from (13) yielding

$$p(\mathbf{x}_k|D_k) = \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, P_{k|k}), \tag{15}$$

where

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + G_k(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}) \tag{16}$$

$$P_{k|k} = P_{k|k-1} - G_k P_{zz,k|k-1} G_k^T, \tag{17}$$

with the Kalman gain

$$G_k = P_{xz,k|k-1} P^{-1}_{zz,k|k-1}. \tag{18}$$

The signal-flow diagram in Fig. 2 summarizes the steps involved in the recursion cycle of the CKF. The CKF theory reduces to the Kalman filter for a linear Gaussian system case. The CKF numerically computes Gaussian weighted integrals that are present in (8)-(9), (11)-(12) and (14) using cubature rules as outlined next.

3.3 Cubature Rules

In general, cubature rules are constructed to numerically compute multi-dimensional weighted integrals. The CKF specifically uses a third-degree cubature rule to numerically compute Gaussian weighted integrals. The third-degree cubature rule is exact for integrands being polynomials of degree up to three or any odd integer. For example, we use the cubature rule to approximate an n-dimensional Gaussian weighted integral as follows:

$$\int_{\mathbb{R}^n} \mathbf{f}(\mathbf{x})\mathcal{N}(\mathbf{x}; \mu, \Sigma)d\mathbf{x} \approx \frac{1}{2n}\sum_{i=1}^{2n} \mathbf{f}(\mu + \sqrt{\Sigma}\xi_i).$$

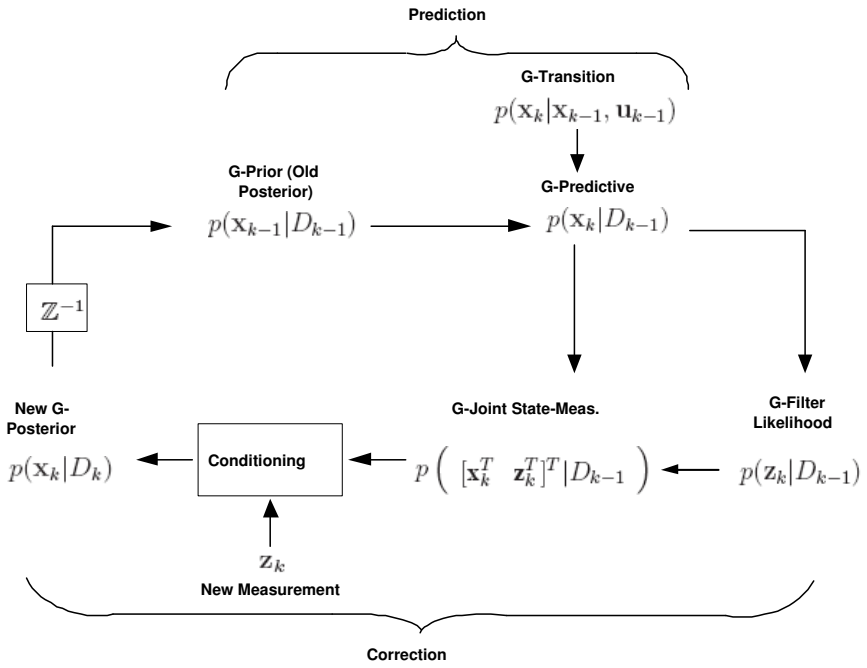


Fig. 2. Signal-flow diagram of the CKF, where 'G-' stands for 'Gaussian-'.

where a square-root factor of the covariance Σ satisfies the relationship $\Sigma = \sqrt{\Sigma}\sqrt{\Sigma}^T$ and the set of $2n$ cubature points are given by

$$\xi_i = \begin{cases} \sqrt{n}\mathbf{e}_i, & i = 1, 2 \dots n \\ -\sqrt{n}\mathbf{e}_{i-n}, & i = n+1, n+2 \dots 2n. \end{cases}$$

with $\mathbf{e}_i \in \mathbb{R}^n$ denoting the i-th elementary column vector. For a detailed exposition of how to derive cubature points, the reader may consult [2].

4 From Learning to Inference: Bayesian filters for supervised training

Mathematically, we may express the functional form of a neural network as

$$\mathbf{d}_k = \mathbf{h}(\mathbf{w}, \mathbf{u}_k), \tag{19}$$

where $\mathbf{w} \in \mathbb{R}^w$ is the weight vector; $\mathbf{u}_k \in \mathbb{R}^m$ is the input vector; $\mathbf{d}_k \in \mathbb{R}^n$ is the desired output; and the form of the nonlinear function $\mathbf{h}$ is determined by the neural network architecture and its activation functions. We shall assume $\mathbf{u}_k$ to encompass the state variables additionally present in a RNN architecture.

When a set of k examples denoted by $D_k = \{(\mathbf{u}_i, \mathbf{d}_i), i = 1, 2 \ldots k\}$ is given, the objective of the supervised training can be posed as follows:

How do we extract the information about $\mathbf{w}$, *which is contained in* D_k *in 'some' optimal fashion? In the Bayesian estimation paradigm, this readily translates into the question of how to compute the posterior density* $p(\mathbf{w}|D_k)$?

To achieve the above objective in the Bayesian paradigm, the first key step is to treat weights to be random variables thereby accounting for uncertainty in their estimate. We assume weight variables to follow the Gaussian random walk model. Hence, we write the state-space model for training neural networks in supervised mode as follows:

$$\text{Process equation:} \quad \mathbf{w}_k = \mathbf{w}_{k-1} + \mathbf{q}_{k-1} \tag{20}$$

$$\text{Measurement equation:} \quad \mathbf{d}_k = \mathbf{h}(\mathbf{w}_k, \mathbf{u}_k) + \mathbf{r}_k, \tag{21}$$

where $\mathbf{w}_k$ may also be interpretable as a *slowly varying parameter* due to the stochastic component $\mathbf{q}_k$, which is assumed to be zero-mean Gaussian process noise with covariance Q_{k-1}; the process noise is purposely injected with decreasing covariance; $\mathbf{r}_k$ is assumed to be zero-mean Gaussian measurement noise with covariance R_k; and the desired output $\mathbf{d}_k$ acts as measurements. Suppose the prior weight statistics are set properly. In this case, the Bayesian filters infer the evolving posterior density of $\mathbf{w}_k$ using the state-space model (20)-(21). The Bayesian filters improve the accuracy of the weight estimate sequentially because they operate on an example-by-example basis.

4.1 Bayesian Filters as Second-Order Optimizers

In this subsection, the nonlinear Bayesian filters under the Gaussian assumption, exemplified by the EKF, the CDKF and the CKF, are viewed to act on the basis of a unified principle: Despite of their algorithmic differences, they all are second-order optimizers in disguise, which minimize a *regularized* sum of squared error cost function with varying degree of accuracies. [1]

Suppose we are given D_k. Under the Gaussian assumption, we write the posterior density (compare (4)):

$$p(\mathbf{w}_k|D_k) = \frac{1}{c_k}\mathcal{N}(\mathbf{w}_k; \hat{\mathbf{w}}_{k|k-1}, P_{k|k-1})\mathcal{N}(\mathbf{t}_k; \mathbf{h}(.,.), \sigma^2\mathbf{I}_n). \tag{22}$$

As the resulting posterior density is also Gaussian, the conditional mean estimate $\hat{\mathbf{w}}_{k|k}$ is the mode of (22) and determined as follows:

$$\begin{aligned}\hat{\mathbf{w}}_{k|k} &= \arg\min_{\mathbf{w}_k} \big(-\log p(\mathbf{w}_k|D_k)\big) \\ &= \arg\min_{\mathbf{w}_k} \big(\sigma^2[\mathbf{w}_k - \hat{\mathbf{w}}_{k|k-1}]^T P_{k|k-1}^{-1}[\mathbf{w}_k - \hat{\mathbf{w}}_{k|k-1}] \\ &\quad +[\mathbf{d}_k - \mathbf{h}(.,.)]^T[\mathbf{d}_k - \mathbf{h}(.,.)]\big),\end{aligned} \tag{23}$$

[1] Regularization is a well-known method for treating mathematically ill-posed problems. It has been applied successfully to several machine learning problems to avoid over-fitting.

where

- the second term on the right hand side of (23) is the usual sum of squared error incurred in function approximation;
- the first term can be thought of as a regularizer for the approximation error criterion; specifically, the regularizer term can be though to measure the discrepancy between two neural networks of same size assessed by a weighted Euclidean distance between their corresponding weight vectors $\mathbf{w}_k$ and $\hat{\mathbf{w}}_{k|k-1}$.

Setting the Jacobian of the regularized sum of squared error cost function (23) to zero yields the explicit solution to the weight estimate of the form (compare (16)):

$$\hat{\mathbf{w}}_{k|k} = \hat{\mathbf{w}}_{k|k-1} + G_k(\mathbf{d}_k - \hat{\mathbf{d}}_{k|k-1}).$$

where the predicted network output

$$\hat{\mathbf{d}}_{k|k-1} = \mathbb{E}\big[\mathbf{h}(\mathbf{w}_k, \mathbf{u}_k)|D_{k-1}\big], \tag{24}$$

and the Kalman gain G_k can be computed in a number of ways depending on which Bayesian filter is employed. From (20), we may equivalently write the weight estimate

$$\hat{\mathbf{w}}_{k|k} = \hat{\mathbf{w}}_{k-1|k-1} + G_k(\mathbf{d}_k - \hat{\mathbf{d}}_{k|k-1}). \tag{25}$$

Next, we illustrate how each Bayesian filter operates in the context of supervised training.

4.2 Extended Kalman filters

Theory of the EKF results from linearizations of nonlinear functions and then applying the Kalman filter theory to nonlinear, Gaussian systems. The linearization relies on use of the first-order Taylor series expansion evaluated at the current estimate. Though the EKF is popular for its simplicity, it is plagued by the following limitations: (i) It quickly diverges in 'highly' nonlinear systems, owing to its linear approximation (ii) The EKF often yields an underestimated error covariance as it does not account for a prior covariance matrix in its analytical linearization approach (iii) Finally, its application is limited to differentiable functions only.

To elaborate the concept, we consider how the EKF computes its predicted network output in the supervised training mode of the neural network. Here, we assume that the prior density of $\mathbf{w}$, $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, I_w)$. In the sequel, we suppress arguments $\mathbf{d}_k$ and $\mathbf{u}_k$ of $\mathbf{h}()$ for notational simplicity. We write the EKF-estimated predicted network output

$$\begin{aligned}\hat{\mathbf{d}}_{k|k-1} &= \mathbb{E}[\mathbf{h}(\mathbf{w}_k)|D_{k-1}] \\ &\approx \mathbb{E}[\mathbf{h}(\mathbf{0}) + J(\mathbf{0})\mathbf{w}_k] \\ &= \mathbf{h}(\mathbf{0}).\end{aligned}$$

Here $J \in \mathbb{R}^{n \times w}$ denotes the Jacobian matrix of $\mathbf{h}$. Here, the EKF estimate does not utilize the second-order statistics. Moreover, we may equivalently obtain the predicted network output of the EKF as

$$\hat{\mathbf{d}}_{k|k-1} = \mathbb{E}[\mathbf{h}(\mathbf{w}_k)|D_{k-1}] \approx \mathbf{h}(\mathbf{0}) = \mathbf{h}(\mathbb{E}(\mathbf{w}_k|D_{k-1})),$$

which implies that the EKF estimate is exact for linear functions, and may be highly biased otherwise. As a solution to mitigate bias errors of the EKF, the CDKF is discussed next.

4.3 Central difference Kalman filters

The CDKF is, in principle, comparable to the second-order EKF. That is, the nonlinear functions of the dynamic system are approximated by the second-order Taylor series expansion, in which the Jacobians and Hessians are replaced by central difference approximations. Use of the CDKF for training neural networks has had successes recently for the following two main reasons [19]: (i) It is derivative-free (ii) The square-root version of the original CDKF ensures a smooth continuous operation during training.[2]

To elaborate its operation, we write the steps involved in computing the predicted network output as follows:
(i) We write the second-order Taylor series expansion of $\mathbf{h}(\mathbf{w}_k)$ about $\mathbf{0}$:

$$\mathbf{h}(\mathbf{w}_k) \approx \mathbf{h}(\mathbf{0}) + J(\mathbf{0})\mathbf{w}_k + \frac{1}{2}\sum_{i=1}^{n} \mathbf{w}_k^T H_i(\mathbf{0})\mathbf{w}_k \mathbf{e}_i, \tag{26}$$

where $H_i \in \mathbb{R}^{w \times w}$ denotes the the square symmetric Hessian matrix whose elements are various second-order partial derivatives of $\mathbf{h}_i(.), i = 1, 2 \ldots n$ and $\mathbf{e}_i \in \mathbb{R}^n$ is the i-th elementary column vector.

(ii) Taking expectation of (26) with respect to $\mathbf{w}_k$ yields

$$\hat{\mathbf{d}}_{k|k-1} = \mathbb{E}[\mathbf{h}(\mathbf{w}_k)|D_{k-1}] \approx \mathbf{h}(\mathbf{0}) + \frac{1}{2}\sum_{j=1}^{w} \mathbf{Tr}[H_i(\mathbf{0})]\mathbf{e}_i, \tag{27}$$

where **Tr** denotes the trace operator.

(iii) Finally, we replace the Hessian in (27) by the central difference approximation. For example, we approximate the j-th diagonal element of the Hessian

$$[H_i(\mathbf{0})]_{j,j} = \frac{\partial^2 \mathbf{h}_i}{\partial w_j^2}(\mathbf{0}) \approx \frac{\mathbf{h}_i(\triangle \mathbf{e}_j) - 2\mathbf{h}_i(\mathbf{0}) + \mathbf{h}_i(-\triangle \mathbf{e}_j)}{2\triangle}$$

where the step-size $\triangle$ is chosen to be $\triangle = \sqrt{3}$ for a Gaussian prior.

The CDKF yields a more accurate estimate than that of the EKF. Secondly, it operates on function evaluations, thus is extendable to non-differential functions. However, several assumptions are made in order to arrive at the CDKF

[2] The square-root CDKF is sometimes referred to as nprKF, where 'npr' stands for the initials of the three authors of [14].

equations. For example, (i) when computing the first two-order moments, namely the mean and covariance, the CDKF essentially takes two approximations: (i.a) Quadratic interpolating polynomials for nonlinear functions and (i.b) numerically approximative Jacobians and Hessians (ii) the choice of the step size $\triangle$ to be $\triangle = \sqrt{3}$ is not always guaranteed to be optimal in all problem settings and (iii) finally, the interpolating point set using the fixed step-size of $\triangle = \sqrt{3}$ irrespective of the state dimension w may fail to capture global properties of a nonlinear function (e.g., peaks and troughs); the reason is that the quadratic function approximation is accurate only in the vicinity of the current estimate [2].

4.4 Cubature Kalman filters

As opposed to functional approximations to nonlinear functions, the CKF takes a more accurate and mathematically straightforward approach. As described earlier, the CKF computes the predicted network output as follows:

$$\begin{aligned}\hat{\mathbf{d}}_{k|k-1} = \mathbb{E}[\mathbf{h}(\mathbf{w}_k)|D_{k-1}] &= \int_{\mathbb{R}^w} \mathbf{h}(\mathbf{w})\mathcal{N}(\mathbf{w};\mathbf{0},I_w)d\mathbf{w} \\ &\approx \frac{1}{2w}\sum_{i=1}^{2w}\mathbf{h}(\xi_i),\end{aligned}$$

where cubature points ξ_i take a similar form as in Subsection 3.3

Remark. So far, we have discussed how each Bayesian filter computes the predicted network output. We may also extend the similar procedure to compute the covariance of the predicted error. The predicted error covariance, which is also called *innovation covariance*, essentially determines the accuracy of the Kalman gain. Because the CKF is proven to retain the information about the first two order moments of a nonlinearly transformed variable completely, we may expect the CKF to update the weights more accurately than other two Bayesian filters.

5 Practical Implementation

In this section, we describe a number of techniques of how to fine tune the CKF for better results.

5.1 Reducing Computational Complexity

As described earlier, the Bayesian filters implicitly minimizes the sum of squared error. To elucidate the relationship of the sum of squared error in (23) and the measurement function, we rewrite (2) as

$$\mathbf{0} = \big(\mathbf{d}_k - \mathbf{h}(\mathbf{w}_k, \mathbf{x}_k)\big) + \mathbf{r}_k, \tag{28}$$

where the output (measurement) is forced to take $\mathbf{0}$. Typically we encounter vector-valued measurements in n-class classification problems. In this case, we may replace the vector-valued measurement function by a scalar-valued measurement function:

$$0 = \tilde{h}(\mathbf{w}_k, \mathbf{x}_k, \mathbf{t}_k) + r_k, \tag{29}$$

where

$$\tilde{h}(.,.,.) = \sqrt{\sum_{i=1}^{n} \left(\mathbf{d}_k^{(i)} - \mathbf{h}^{(i)}(.,.,.)\right)^2}.$$

The above reformulation appears reasonable because the Bayesian filter minimizes the same sum of squared error. Also, the independency among output labels does not alter the *degree of nonlinearity* from the point of view of any single weight-variable. Finally, when any one of the outputs corresponding to the appropriate class is given, the other outputs carry little information. Based on these observations, we expect the Bayesian filter using (29) to yield a weight estimate similarly to that using (2). The similarity has also been verified in a number of simulations. The scalar reformulation offers the following benefits: (i) It reduces the computational complexity of the Bayesian filter based-training significantly (ii) It improves the numerically stability of the Bayesian filter. For example, when computing the Kalman gain, we replace the inversion of the innovation covariance matrix with a trivial scalar division (see Ch. 2, [9] also for a similar exposition in the EKF setup).

5.2 Fitting Various Cost Functions

In this subsection, we consider how to incorporate cost criteria other than the sum of squared error into the state-space model. To fit a given cost function, which may be non-differentiable, into the state-space model, we closely follow the idea of reformulating the measurement equation as explained through a couple of examples below.

Example 1: We consider the cross-entropic cost function that has been found to be appropriate for an n-class pattern-classification problem. When the classifier network uses the softmax output layer and the binary encoding for its outputs, we write the cross-entropic cost function

$$J_k = -\sum_{i=1}^{n} \mathbf{d}_k^{(i)} \log \mathbf{h}^{(i)}(\mathbf{w}_k, \mathbf{x}_k).$$

Correspondingly, we write the reformulated measurement function to be the square-root of J_k. That is, we write the measurement equation to take the form:

$$0 = \sqrt{-\sum_{i=1}^{n} \mathbf{d}_k^{(i)} \log \mathbf{h}^{(i)}(\mathbf{w}_k, \mathbf{x}_k)} + r_k. \tag{30}$$

Example 2: Here we consider the cost function to be the Minskowski's p-power metric. In this case, we write the reformulated measurement equation:

$$0 = \sqrt{\Big(\sum_{i=1}^{n} |\mathbf{d}_k^{(i)} - \mathbf{h}^{(i)}(\mathbf{w}_k, \mathbf{x}_k)|^p\Big)} + r_k. \tag{31}$$

Setting $p = 2$ in (31) yields the usual sum of squared error cost function. Moreover, the CKF and the CDKF can still be utilized when $p = 1$ (absolute error or $\mathcal{L}_1$-norm), whereas the EKF fails due to the unavailability of analytic gradient.

5.3 Choosing Hyperparameters

Both the accuracy of weight estimate and the rate of convergence of the CKF crucially hinge on the right choice of statistics of hyperparameters, namely, noise models and weight priors. Specifically, we assume that the given training data set is normalized and all the models are zero-mean Gaussian. In the Bayesian filtering framework, the assumption of Gaussianity allows to obtain a tractable solution. In this case, we are left to specify right covariances only. Next, we describe simple methods to compute them and explain why they work effectively.

Process noise covariance. Injecting process noise artificially helps escape poor local minima of the error surface. However, as the training progresses, the process noise covariance Q_k needs to be reduced systematically. We use a mathematical relationship of the form $Q_{k-1} = (\frac{1}{\lambda} - 1)P_{k-1|k-1}$, where $\lambda \in (0, 1]$ is called the the forgetting factor [28]. This allows to put exponentially decaying weight on past data. Consequently, for the linear process equation (20), we get the predicted error covariance $P_{k|k-1}$ to be $P_{k|k-1} = \frac{1}{\lambda}P_{k-1|k-1}$; this suggests that we get $P_{k|k-1} > P_{k-1|k-1}$ when predicting a new weight estimate before the receipt of a new measurement. Typically, the choice of λ being slightly less than unity works well for many problems.

Measurement noise covariance. Suppose we are given a set of clean training data set embedding the information about the unknown weight vector. Hence, it is reasonable to assume that the likelihood function is 'highly' peaked. We choose a fixed measurement noise covariance of the form $R_k = \sigma^2 I_n$. However, the choice may vary depending on the quality of the training data.

Initial error covariance. To choose this hyperparameter, we use the *principle of stable estimation*, which suggests to choose the prior density to be relatively more flat than the likelihood function in the case of uninformative prior [16]. The reason is that the shape of the posterior is forced to closely follow the shape of the likelihood rather than that of uninformative prior. We may choose the initial error covariance $P_{0|0}$ to be a diagonal matrix, with diagonal entries being one/two order higher than σ^2. The choice of initial covariance seems contrary to the Bayesian perspective, which is essentially built around the principle that *Priors rule; posteriors just follow around them.* As training progresses however, the prior becomes more influential and decides the shape of the posterior (see Fig. 3).

(a) At time $k = 1$ (b) At time $k = 1$

Fig. 3. Posterior (solid line) resulting from prior (dash-dot line) combined with likelihood function (dotted line) at two consecutive time steps.

5.4 Improving Stability: Square-root Formulation

The two basic properties of an error covariance matrix are (i) symmetry and (ii) positive definiteness. In practice, due to finite word-length arithmetics, these two properties are often lost. Specifically, the positive definiteness of the posterior covariance, which is computed as the difference between two positive definite matrices is likely to be lost (see (17)). The conflict between theoretical properties and practical values may lead to an unstable behavior of a Bayesian filter. Importantly, the loss of the positive definiteness may be more hazardous as it stops the CKF to run continuously. The obvious question in this case is: how do we avoid this undesirable situation so as to assure a continuous stable operation of the CKF? The answer lies in a mathematically principled approach called square-root filtering.

The square-root version of the cubature Kalman filter hereafter called *square-root cubature Kalman filter* (SCKF), essentially propagates a square-root factor of the covariance matrix. To develop the SCKF for training neural networks, we use matrix factorizations. To elaborate further, we write the CKF-estimated covariance $P \in \mathbb{R}^{w \times w}$ of the form:

$$P = AA^T, \tag{32}$$

where $A \in \mathbb{R}^{w \times l}$ with $l > w$ is a fat matrix. In the square-root filtering, we are required to compute a square-root factor of P of dimension $(w \times w)$ without explicitly taking the square-root of P owing to numerical instabilities. Applying the QR decomposition on A^T, we get

$$P = AA^T = R^T Q^T Q R = R^T R = BB^T,$$

where the desired square-root factor is a lower triangular matrix $B = R^T$. Hereafter we call this procedure *triangularization*. Note that in computing the square-root covariance B, we discard the orthogonal matrix Q and exploit the upper triangular matrix R only. Since B is triangular, its sparseness provides efficient computation and reduced storage space. In terms of computational complexity, the SCKF requires $O(w^3)$ *flops*, which comes from the use of a matrix triangularization.

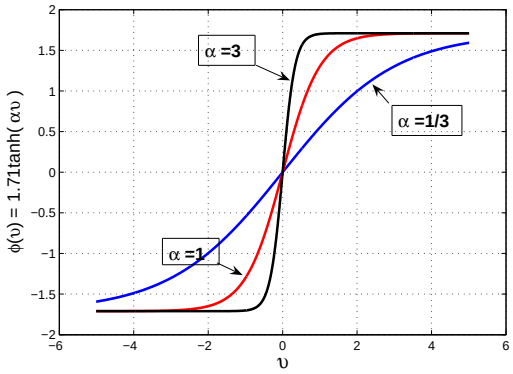


Fig. 4. Effect of α on the shape of the activation function $\varphi(v) = 1.71\tanh(\alpha v)$.

6 Experimental Results

In this section, we report our findings of the experiment dealing with Bayesian filter-trained RNNs. The trained RNNs are employed to predict the chaotic Mackey-Glass time-series. We use the CKF as described in Appendix A, the CDKF and the EKF as training algorithms.

Chaotic Mackey-Glass System. The Mackey-Glass equation is often used to model the production of white-blood cells in Leukemia patients and given by the delay differential equation:

$$\frac{dx_t}{dt} = 0.1x_t + \frac{0.2x_{t-\Delta}}{1 + x_{t-\Delta}^{10}}, \tag{33}$$

where the delay $\Delta = 30$. To sample the time-series at discrete time steps, we numerically integrated (33) using the forth-order Runge-Kutta method with a sampling period of $T = 6$ s, and the initial condition $x_t = 0.9$, for $0 \leq t \leq \Delta$. Given a chaotic system, it is known that the next data sample $x_{k+\tau}$ can be predicted from a properly chosen time-series $\{x_k, x_{k-\tau} \ldots x_{k-[d_E-2]\tau}, x_{k-[d_E-1]\tau}\}$, where d_E and τ are called the embedding dimension and the embedding delay, respectively. For the chaotic Mackey-Glass system, d_E and τ were chosen to be seven and one, respectively.

RNN Architecture. We used the Bayesian filter-trained RNNs to predict the chaotic Mackey-Glass time-series data. We chose the RNN to have seven inputs representing an embedding of the observed time-series, one output, and one self-recurrent hidden layer with five neurons. Hence, the RNN has a total of 71 connecting weights (bias included). The output neuron uses a linear activation function, whereas all the hidden neurons use the hyperbolic tangent function of the form

$$\varphi(v) = 1.71\tanh(\alpha v),$$

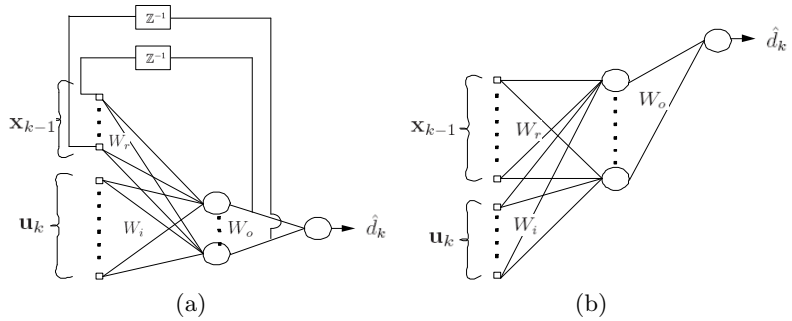


Fig. 5. Schematic diagrams (a). Original RNN (b). Unfolded RNN of unity truncation depth.

where α was assumed to take values ranging from 1/3 - 3. As shown in Fig. 4, the hyperbolic tangent function is 'mildly' nonlinear (that is, close to a linear function) around its origin when $\alpha = 1/3$. Its nonlinearity increases with α, and behaves closely similar to a switch when $\alpha = 3$.

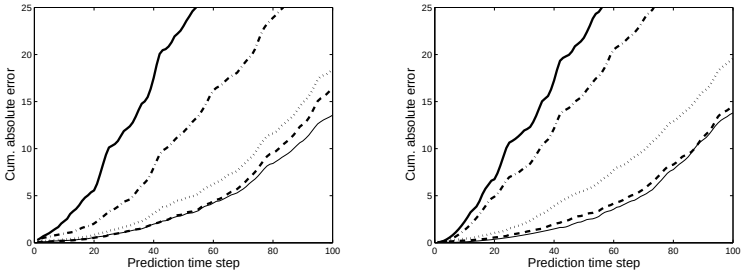
State-Space Model. We represent the RNN architecture following the state-space model:

$$\begin{aligned} \mathbf{w}_k &= \mathbf{w}_{k-1} + \mathbf{q}_{k-1} \\ 0 &= \Big(d_k - W_o\varphi\big(W_r\mathbf{x}_{k-1} + W_i\mathbf{u}_k\big)\Big) + r_k. \end{aligned}$$

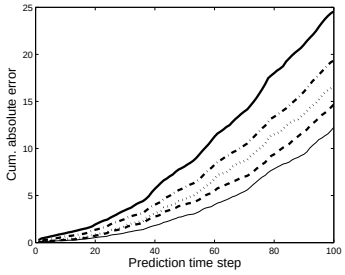
Here φ_o denotes the softmax function of the output layer, whereas φ_i and φ_r denote the hyperbolic tangent functions of the input layer and recurrent layer, respectively; W_i, W_r and W_o are input, recurrent and output weight matrices of appropriate dimensions; the weight vector $\mathbf{w}_k$ is obtained by grouping elements from W_i, W_r and W_o in 'some' orderly fashion.

Data. We obtained the chaotic time series of length 1000, of which the first half was used for training and the rest for testing. To train the RNN using the CKF, we used 10 epochs per run. Each epoch was obtained from a 107 time-step long subsequence, starting from a randomly selected point. That is, each epoch consists of 100 examples gleaned by sliding a window of length eight over the subsequence. The weights were initialized to zero-mean Gaussian with a diagonal covariance of $0.5I_w$; We made Q_{k-1} to decay such that $Q_{k-1} = (\frac{1}{\lambda} - 1)P_{k-1|k-1}$ with $\lambda = 0.9995$. We fixed R_k to be $R_k = 5 \times 10^{-3}$ across all entire epochs; the output of the hidden layer (state) of the RNN at $t = 0$, $\mathbf{x}_0$ was assumed to be zero.

As opposed to the CKF relying on integration, the EKF and the CDKF use gradient information, which necessitate the use of the truncated backpropagation through time method. The truncation depths of longer than unity were tried; surprisingly, a length of unity was found to be sufficient in this experimental setup (see Fig. 5).



(a) EKF-trained RNN. (b) CDKF-trained RNN.



(c) CKF-trained RNN.

Fig. 6. Ensemble-averaged cumulative absolute error curves during the autonomous prediction when $\alpha = 1/3$ (solid-thin), 2/3 (dashed), 1 (dotted), 2 (dash-dot), and 3 (solid-thick).

Performance Metric. During the testing phase, we initialized RNNs with 20 time-step long test sequence and allowed them to run autonomously using their own output for the next 100 steps. To compare the performance of CKF-trained RNNs against the CDKF, and EKF-trained RNNs fairly, we made 50 independent training runs for each value of α. As a performance metric, we used the ensemble-averaged cumulative absolute error

$$e_k = \frac{1}{50} \sum_{r=1}^{50} \sum_{i=1}^{k} |d_i^{(r)} - \hat{d}_i^{(r)}|; \quad k = 1, 2, \ldots 100.$$

The long-term accumulative prediction error is increasing exponentially in time k for the following reasons: (i) the chaotic systems are known to be sensitive even to a slight perturbation in their present state [15] (ii) the prediction error is amplified at each time step due to the closed loop structure. From Figs. 6(a) and 6(b), we see that the EKF and CDKF-trained RNNs break down at $\alpha = 2$ and beyond. The CKF-trained RNN performs reasonably well even when $\alpha = 3$ at which the hyperbolic tangent function is 'severely' nonlinear (Fig. 6(c)). The reason is that the CKF tends to locate a better local minimum of the cost function in the weight space than the EKF or the CDKF. The successful solution

to estimating more accurate weights in a highly nonlinear setup studied herein is a convincing testimony to the superiority of the CKF as a training algorithm.

7 Concluding Remarks

In this paper, we introduced the cubature Kalman filter (CKF) for training RNNs for the first time in the machine learning literature. With the cubature rule at our disposal, the principle behind the CKF can be confined to linear filtering theory. Despite algorithmic differences, the approximate Bayesian filters exemplified by the EKF, the CDKF and the CKF, share a common feature: *they all are second-order optimizers minimizing the regularized sum of square error cost function*. Specifically, it has already been proved that the EKF is equivalent to the Newton's method whereas the backpropagation is equivalent to a degenerate form of the EKF in [21], [23]. Hence, the Bayesian filters provide more accurate solution and rapid convergence than the gradient method. A significant performance boost of Bayesian filters over the gradient descent method results from computing (i) the predicted network output (24) and (ii) the Kalman gain G_k joining the prediction error to the latest weight estimate. Of course, a comparison of Newton's method and the weight update of a Bayesian filter suggests that the Kalman gain encompasses a stochastic counterpart of a scaled inverse Hessian of the cost function. The Kalman gain can also be thought of as a collection of learning rates chosen optimally for each neuron of the network.

Specifically, CKF-trained RNNs seem to outperform other Bayesian filter-trained RNNs. Apart from its more accurate solution to the weight vector, the CKF may be preferred for the following reasons:

1. The CKF does not require analytical derivatives.
2. The CKF includes only the forward pass as opposed to forward and backward passes as in the gradient-based training algorithms. Hence, we do not keep track of information local to each unit.
3. In the CKF-based training of a RNN, we only consider the unfolded architecture of unity truncation depth. Hence, it is extremely easy to code and implement the training algorithm for a RNN.
4. Last but by no means the least, we mention the benefit of the fast square-root algorithm tailored specifically for training neural networks. It ensures a continues smooth training operation.

Acknowledgement

The authors would like to thank the natural sciences and engineering research council (NSERC) of Canada for financially supporting this project.

Appendix A: Square-root Cubature Kalman Filter

State-space model: In the SCKF algorithm outlined below, we assume the following reformulated state-space model:

$$\begin{aligned}\mathbf{w}_k &= \mathbf{w}_{k-1} + \mathbf{q}_{k-1} \\ 0 &= \tilde{h}(\mathbf{w}_k, \mathbf{u}_k, \mathbf{d}_k) + r_k,\end{aligned}$$

where $\tilde{h}(\mathbf{w}_k, \mathbf{u}_k, \mathbf{d}_k) = \sqrt{\sum_{i=1}^{n} \left(\mathbf{d}_k^{(i)} - \mathbf{h}^{(i)}(\mathbf{w}_k, \mathbf{u}_k)\right)^2}$ for the cost function being the sum of squared error; $\mathbf{q}_{k-1}$ and r_k are independent, zero-mean Gaussian noise sequences with covariances Q_{k-1} and σ^2; Q_{k-1} is annealed using the adaptive strategy with a forgetting factor λ.

Initial Assumption: At time $(k-1)$, we are given the posterior density

$$p(\mathbf{w}_{k-1}|D_{k-1}) = \mathcal{N}(\hat{\mathbf{w}}_{k-1|k-1}, S_{k-1|k-1}S_{k-1|k-1}^T).$$

SCKF Algorithm from time $(k-1)$ to k: Compute

1. Cubature points

$$\mathscr{W}_i = \hat{\mathbf{w}}_{k-1|k-1} + \frac{1}{\sqrt{\lambda}} S_{k-1|k-1}\xi_i \quad i = 1, 2, \ldots, 2w. \tag{34}$$

   where the cubature points $\{\xi_i\}$ are defined in Subsection 3.3.
2. Propagated cubature points

$$\mathscr{D}_i = \tilde{h}(\mathscr{W}_i, \mathbf{u}_k, \mathbf{d}_k) \quad i = 1, 2, \ldots, 2w. \tag{35}$$

3. Predicted network output

$$\hat{d}_{k|k-1} = \frac{1}{2w} \sum_{i=1}^{2w} \mathscr{D}_i. \tag{36}$$

   Hence, the innovation is $(-\hat{d}_{k|k-1})$.
4. Innovation variance

$$\sigma_{d,k|k-1}^2 = \mathscr{D}\mathscr{D}^T + \sigma^2, \tag{37}$$

   where the weighted, centered row vector

$$\mathscr{D} = \frac{1}{\sqrt{2w}}[\mathscr{D}_1 - \hat{d}_{k|k-1} \;\; \mathscr{D}_2 - \hat{d}_{k|k-1} \ldots \mathscr{D}_{2w} - \hat{d}_{k|k-1}]. \tag{38}$$

5. Cross-covariance

$$P_{wd,k|k-1} = \mathscr{W}\mathscr{D}^T, \tag{39}$$

   where the weighted, centered matrix

$$\mathscr{W} = \frac{1}{\sqrt{2w}}[\mathscr{W}_1 - \hat{\mathbf{w}}_{k-1|k-1} \;\; \mathscr{W}_2 - \hat{\mathbf{w}}_{k-1|k-1} \ldots \mathscr{W}_{2w} - \hat{\mathbf{w}}_{k-1|k-1}]. \tag{40}$$

6. Kalman gain

$$G_k = P_{wd,k|k-1}/\sigma^2_{d,k|k-1}. \tag{41}$$

7. Updated weight

$$\hat{\mathbf{w}}_{k|k} = \hat{\mathbf{w}}_{k-1|k-1} - G_k \hat{d}_{k|k-1}. \tag{42}$$

8. Square-root factor of the posterior error covariance

$$S_{k|k} = \mathbf{Tria}([\mathscr{W} - G_k \mathscr{D} \quad \sigma G_k]), \tag{43}$$

where we denote the triangularization algorithm as $B = \mathbf{Tria}(A)$ with the output B being a lower triangular matrix. The matrices A and B are related as follows: Let the upper triangular matrix R be obtained from the QR decomposition on A^T; then $B = R^T$.

References

1. Anderson, B. D. O., Moore, J. B.: Optimal Filtering. Prentice Hall, New Jersey (1979).
2. Arasaratnam, I., Haykin, S.: Cubature Kalman Filters, IEEE Trans. Automatic Cont., under 3rd review (2008).
3. Bengio, Y., Simard, P., Frasconi, P.: Learning Long-term Dependencies with Gradient Descent is Difficult, IEEE Trans. Neural Netw., vol. 5, no. 2, 157-166 (1994).
4. Cappe, O., Godsil, S. J., Moulines, E.: An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo, Proc. IEEE, vol. 95, no. 5 (2007).
5. Gordon, N.J, Salmond, D. J., Smith, A. F. M.: Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation, IEE Proc.-F, vol. 140, 107-113 (1993).
6. Hanson, S. J., Burr, D.J.: Minkowski-r Back-Propagation: Learning in Connectionist Models with Non-Euclidian Error Signals, Neural inform. proces. sys., Anderson, D., Ed., NY: American Inst. of Physics, pp. 348-357 (1988).
7. Haykin, S., Li, L., Nonlinear Adaptive Prediction of Nonstationary Signals, IEEE Trans. Signal Process, vol. 43, no. 2, 526-535 (1995).
8. Haykin, S.: Neural Networks: A Comprehensive Foundation, Prentice Hall, New Jersey (1999).
9. Haykin, S., Ed.: Kalman Filtering and Neural Networks, Wiley, New York (2001).
10. Julier, S. J., Ulhmann, J. K., Durrant-Whyte, H. F.: A New Method for Nonlinear Transformation of Means and Covariances in Filters and Estimators, IEEE Trans. Automatic Cont., vol. 45, 472-482 (2000).
11. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition, Proc. IEEE, vol. 86, no. 11, 2278-2324, (1998).
12. Liu, J. S.: Monte Carlo Strategies in Scientific Computing, Springer (2001).
13. Narendra, K. S., Parthasarathy, K.: Identification and Control of Dynamical Systems Using Neural Networks, IEEE Trans. Neural Netw., vol. 1, no. 1, 4-27 (1990).
14. Nørgaard, M. N., Poulsen, N. K., Ravn, O.: New Developments in State Estimation of Nonlinear Systems, Automatica, vol. 36, 1627-1638 (2000).
15. Ott, E.: Chaos in Dynamical Systems, 2nd ed., Cambridge Univ. Press (2002).

16. Peterka, V.: Bayesian Approach to System Identification, in Trends and progress in system identification, Eykhoff, P., Ed., Pergamon Press, Oxford, pp. 239-304 (1981).
17. Pineda, F.: Generalization of Backpropagation to Recurrent Neural Networks, Physical Review Letters, vol. 59, no. 19 (1987).
18. Plumer, E. S.: Training Neural Networks Using Sequential Extened Kalman Filters, Proc. World Congress on Neural Netw., Washigton DC, pp. I-764-I-769 (1995).
19. Prokhorov, D.: Training Recurrent Neurocontrollers for Robustness with Derivative-Free Kalman Filter, IEEE Trans. Neural Netw., vol. 17, no. 6, 1606-1616 (2006).
20. Puskorious, G., Feldkamp,L.: Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Neural Networks, IEEE Trans. Neural Netw., vol. 5, no. 2 (1994).
21. Ruck, D. W., Rogers, S. K., Kabrisky, M., Maybeck, P., Oxle, M. E.: Comparative Analysis of Backpropgation and the Extended Kalman Filter for Training Multilayer Perceptrons, IEEE Trans. Patt. Anal. & Mach. Intell., vol. 14, no. 6, 686-691 (1992).
22. Sarangapani, J.: Neural Network Control of Nonlinear Discrete-Time Systems, CRC Press, Florida (2006).
23. Schottky, B., Saad, D.: Statistical Mechanics of EKF Learning in Neural Networks, J. Physics A, vol. 32, no. 9, 1605-1621 (1999).
24. Shah, S., Palmieri, F., Datum, M.: Optimal Filtering Algorithms for Fast Learning in Feedforward Neural Networks, Neural Net., vol. 5, no. 5, 779-787 (1992).
25. Sum, J., Leung, C., Young, G. H., Kan, W.: On the Kalman Filtering Method in Neural Network Training and Pruning, IEEE Trans. Neural Netw., vol. 10, no. 1 (1999).
26. Yamada, T., Yabuta, T.: Dynamic System Identification Using Neural Networks, IEEE Trans. Systems, Man and Cyber., vol. 23, no. 1, 204-211 (1993).
27. Werbos, P.: Backpropagation Through Time: What It Does and How to Do It, Proc. IEEE, vol. 78, no. 10, 1550-1560 (1990).
28. West, M., Harrison, J.: Bayesian Forecasting and Dynamic Linear Models, Springer-Verlag (1996).