Recurrent Neural Network Training with the Extended Kalman Filter

Peter TREBATICKÝ[*]

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
trebaticky@fiit.stuba.sk

Abstract. Recurrent neural networks, in contrast to the classical feedforward neural networks, better handle inputs that have space-time structure, e.g. symbolic time series. Since the classic gradient methods for recurrent neural network training on longer input sequences converge very poorly and slowly, the alternative approaches are needed. We describe the principles of the training method with the Extended Kalman Filter and with its modifications Unscented Kalman Filter, nprKF and with their joint versions. The Joint Unscented Kalman Filter was not used for this purpose before. We compare the performance of these filters and of the classic Truncated Backpropagation Through Time in an experiment for next-symbol prediction – word sequence generated by Reber automaton. All the new filters achieved significantly better results.

1 Introduction

Recurrent neural networks, in contrast to the classical feedforward networks, better handle inputs that have space-time structure. They can be used to process word sequences of languages generated by grammars or sequences with chaotic character, which is very useful for controlling robotic systems or analysing and predicting of variations in mechanical equipments. Recurrent neural network learning is a very difficult numerical problem, which approaches very poorly and slowly to satisfactory results when being solved with the classic gradient optimisation methods on longer input sequences. That is the reason for searching for the more effective methods of recurrent network learning.

[*] Supervisor: doc. RNDr. Jiří Pospíchal, DrSc., Institute of Applied Informatics, Faculty of Informatics and Information Technologies STU in Bratislava

This paper presents the principles of various methods based on Kalman filtration that serve as a better alternative to classic gradient methods. First of all we briefly introduce the reader into Kalman filtration by describing the original Kalman filter (section 2). Then we describe the extension of introduced Kalman filter to cope also with the nonlinear systems. So called the Extended Kalman Filter (EKF) (section 3) can then be used for the task of neural network training. Furthermore, we describe relatively new filters: Unscented Kalman Filter (UKF), nprKF, as well as their joint versions UKFj and nprKFj (sections 4, 5 and 6 respectively). These new filters can be easily implemented and moreover they provide superior performance, which is demonstrated by two experiments with next-symbol prediction (section 7). The interested reader can find the detailed description and formulas of these filters in [6].

2 Kalman Filter

Kalman filter, which is the set of mathematical equations, is considered as one of the important discoveries in the control theory principles. E. Kalman's article was published in the year 1960. Its most immediate applications were in control of complex dynamic systems, such as manufacturing processes, aircrafts, ships or spaceships (it was part of the Apollo onboard guidance system). It was and still is frequently used not only in automation, but also in the graphical and economical applications, etc. However, the Extended Kalman Filter started to appear in the neural network training applications only relatively recently, which was caused by the progress of computer systems development.

System's state vector $\mathbf{x}_k$ (or simply state) is defined as a minimal set of data that uniquely describes dynamic system's behaviour, where k denotes discrete time. Simply put, system's state is a vector containing all the relevant variables of the system [2].

1. *Process equation* $\mathbf{x}_k = \mathbf{F}_{k,k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}$, where $\mathbf{F}_{k,k-1}$ is the *transition matrix* taking the state $\mathbf{x}_k$ from time $k-1$ to time k. The *process noise* $\mathbf{q}_k$ is assumed to be additive, white and Gaussian with zero mean and covariance matrix $\mathbf{Q}_k$.

2. *Measurement equation* $\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{r}_k$ where $\mathbf{y}_k$ is *observable* at time k and $\mathbf{H}_k$ is the *measurement matrix*. The *measurement noise* $\mathbf{r}_k$ is also assumed to be additive, white and Gaussian with zero mean and covariance matrix $\mathbf{R}_k$. The noises $\mathbf{r}_k$ and $\mathbf{q}_k$ are uncorrelated.

The Kalman filtering problem lies in jointly solving the process and measurement equations for the unknown state in an optimal manner. We have to use all the observed data, comprising of the vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k$ to find the minimum mean-square error estimate of the state $\mathbf{x}_k$ for every $k \geq 1$.

Kalman filter functions in two repeating steps [2]:

1. *Time update,* also called prediction step. Here belong equations: $\hat{\mathbf{x}}_k^- = \mathbf{F}_{k,k-1}\hat{\mathbf{x}}_{k-1}$ and $\mathbf{P}_k^- = \mathbf{F}_{k,k-1}\mathbf{P}_{k-1}\mathbf{F}_{k,k-1}^T + \mathbf{Q}_{k-1}$. That is we compute the apriori estimation of the state and the error covariance.
2. *Measurement update*, also called correction step.
   $$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left[\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right]^{-1} \qquad \hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-)$$
   $$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-$$
   We correct the state estimate obtained in the previous step according to the measurement $\mathbf{y}_k$.

3 Extended Kalman Filter

When the model is *nonlinear*, which is the case of neural networks, we have to extend Kalman filter using linearization procedure. Resulting filter is then called extended Kalman filter (EKF) [2].

Neural network is a nonlinear dynamic system that can by described by equations:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{q}_{k-1} \quad \text{and} \quad \mathbf{y}_k = h(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_{k-1}) + \mathbf{r}_k$$

The process equation expresses the state of neural network as a stationary process corrupted with the process noise $\mathbf{q}_k$, where the state of the network $\mathbf{x}$ consists of network weights. Measurement equation expresses the desired output of the network as a nonlinear function of the input vector $\mathbf{u}_k$, of the weight vector $\mathbf{x}_k$ and for recurrent networks also of the activations of recurrent neurons from previous step $\mathbf{v}_{k-1}$. This equation is augmented by the random measurement noise $\mathbf{r}_k$. The covariance matrix of the noise $\mathbf{r}_k$ is $\mathbf{R}_k = E\left[\mathbf{r}_k\mathbf{r}_k^T\right]$ and the covariance of the noise $\mathbf{q}_k$ is $\mathbf{Q}_k = E\left[\mathbf{q}_k\mathbf{q}_k^T\right]$.

The basic idea of the extended Kalman filter lies in the linearization of the measurement equation at each time step around the newest state estimate $\hat{\mathbf{x}}_k$. We use for this purpose just the first-order Taylor approximation of non-linear equation [2].

We can express the neural network training as a problem of finding the state estimate $\mathbf{x}_k$ that minimizes the least-squares error, using all the previous measurements. We can express the solution of this problem as:

$$\mathbf{K}_k = \mathbf{P}_k\mathbf{H}_k^T\left[\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T + \mathbf{R}_k\right]^{-1} \quad \hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \mathbf{K}_k\left[\mathbf{y}_k - h(\hat{\mathbf{x}}_k, \mathbf{u}_k, \mathbf{v}_{k-1})\right]$$
$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_k + \mathbf{Q}_k$$

where $\hat{\mathbf{x}}_k$ is a vector of all the weights, $h(.)$ is a function returning a vector of actual outputs, **y** is a vector of desired outputs, **K** is the so called Kalman gain matrix, **P** is the error covariance matrix of the state and **H** is the measurement matrix (Jacobian). Matrix **H** contains partial derivatives of *i*'s output with respect to *j*'s weight. Its computation is realised either using the Back-propagation algorithm in forward neural network; or using either Back-propagation Through Time, Truncated Back-propagation Through Time, or Real-time Recurrent Learning in recurrent network.

4 Unscented Kalman Filter

The key element in the EKF method is the development of the covariance matrix P, which is in every step approximated using the Taylor expansion of nonlinear function, which relates the network outputs to the weights. The first-order Taylor linearization provides an insufficiently accurate representation in many cases [4].

The nonlinear generalisation of the Kalman filter, called the Unscented Kalman filter (UKF), was also used with success in various applications including the recurrent neural network training [5]. The basic principle of the UKF is conceptually similar to the EKF, but the implementation is significantly different. No derivatives are needed (i.e. calculation of Jacobian, or Hessian), only function evaluations, as opposed to the Taylor approximation.

The basic difference between the EKF and UKF stems from the manner in which Gaussian random variables (GRV) are represented for propagation through system dynamics [2]. In the EKF, the state distribution is approximated by GRV, which are then propagated analytically through the first-order Taylor approximation of the nonlinear system.

The UKF uses so-called *unscented transformation*, what is a relatively new method for calculating the statistics of a random variable, which undergone a nonlinear transformation [3]. A set of *sigma points* is chosen so that their sample mean and sample covariance are the true mean and true covariance, respectively. When propagated through the *true* nonlinear system, they capture the posterior mean and covariance accurately to the second order of Taylor series expansion for *any* nonlinearity. The EKF, in contrast, only achieves first-order accuracy.

This method resembles the Monte Carlo-type methods, where many samples are randomly chosen, which also are propagated through nonlinear transformation. On the other hand, the sigma points are not chosen randomly, but deterministically and moreover, the number of sigma points is low: $2n_x + 1$, where n_x is the dimension of a random variable (vector) [3].

5 Filter nprKF

The other method, or the other Kalman filter modification is the filter named after its authors the nprKF [5]. As with the UKF, also with this filter it is not necessary to calculate derivatives, since the Taylor expansion is in this case replaced by the Stirling's formula for nonlinear function approximation at the interval, where the function f is approximated by the second-order terms [4]. This formula may be interpreted as a Taylor approximation with the derivatives replaced by central divided differences. The authors worked directly with the Cholesky factors of covariance matrices in order to achieve numerical stability.

Even though the nprKF authors do not explicitly mention the sigma points, they in fact also use them and even the same number of them. The analysis in [4] confirms this, since the state estimate is identical in both filters. But they differ in the covariance estimation. The nprKF provides slightly better estimate - the differences in covariance update in comparison with the UKF are in the fourth- and greater-order of the Taylor expansion. Covariance estimate with the UKF is therefore slightly less accurate and may sometimes even lead to the non-positive definite estimate [4].

6 Joint versions of Unscented Kalman Filter and the nprKF

When training recurrent neural networks with the UKF or nprKF, it is necessary to repropagate the network several steps from the past for every sigma point (i.e. with different weights). The alternative approach lies in including the recurrent neurons activations in the state vector, when we simultaneously use the filter for weights estimation as well as the state estimation [5]. It is therefore necessary for thus formulated problem, which is called the Joint Unscented Kalman Filter (UKFj) or the Joint nprKF (nprKFj), to change the state equation and hence the filter equations. It is worth noting that this name is used also for the filter in which the system's state and inputs are concatenated into single state vector [2]. This is used in the situation when we do not have the noise-free input into a system. We have not found this filter in the literature specifically in the context of *recurrent* neural network training. We have created it on the basis of similarity with the nprKFj filter described in [5].

The joint version of a filter on one hand increases the computation complexity by expanding the state vector, on the other hand it simplifies the computation of network output for various weight vectors (sigma points).

7 Experiments

The main goal of our experiments was to compare the performance of all the mentioned filters, namely EKF, UKF, nprKF, UKFj and nprKFj, in recurrent neural network training. In order to linearise a nonlinear system (neural network) in the EKF, it is necessary to compute the matrix of derivatives of outputs with respect to each weight. We used the Truncated Backpropagation Through Time (BPTT(*h*)) for this purpose according to recommendation in [5].

The next symbol prediction procedure is the following: we present in every time step the first symbol in order, and the desired network's output is the next symbol in sequence order. We chose the Elman's network architecture - i.e. the network with one hidden layer, which is recurrent. The predictive performance was evaluated by the means of a Normalised Negative Log-likelihood (NNL), calculated over symbolic sequence from time step $t = 1$ to T [1]:

$$\mathrm{NNL} = -\frac{1}{T}\sum_{t=1}^{T} \log_{|A|} p^{(t)}(s^{(t)})$$

where the base of the logarithm |A| is the number of symbols in the alphabet A and $p^{(t)}(s^{(t)})$ is the probability of predicting symbol $s^{(t)}$ in the time step *t*. If $\mathrm{NNL} = 0$, then the network predicts next symbol with 100%, while $\mathrm{NNL} \geq 1$ corresponds to a very inaccurate prediction (random guessing).

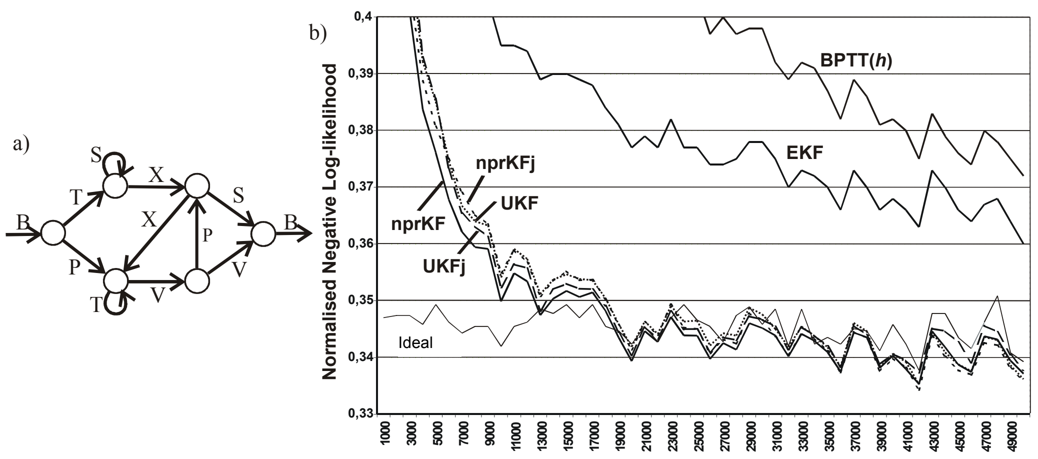

Fig. 1. The NNL dependence of the next symbol prediction (b) in sequence generated by Reber automaton (a) on the number of symbols for various recurrent neural network training methods. The ideal value of NNL for this experiment is also displayed.

7.1 Reber's language

The training sequence in this experiment was generated by the Reber automaton (Fig. 1a), where the next symbol was chosen with 50% probability from two candidate symbols, with the exception of cases when the next symbol was B. The Reber's grammar contains 6 symbols, which we encode using the *one-hot encoding* and we present them to the network as its input. This encoding means that a single input is reserved for each symbol, which we set to 1 and others to 0. The network output is similarly one-hot encoded.

The prediction of symbols generated by the Reber automaton is a relatively simple task, since we are able to tell in which state we are solely from the last two symbols. The problem therefore lies mostly in the "lucky" prediction of one out of two possibilities, i.e. the neural network after training ought to be able to predict with which probability each symbol follows. This also results in the fact, that we should not expect the NNL to be close to zero.

The recurrent neural network with three hidden neurons was trained with each method. The weight update was performed in every time step. The NNL was computed for every 1000 symbols. We present the comparison of all the filters by this indicator in Figure 1b. The graph also contains the curve representing the "ideal" NNL. We obtain it when we predict the next symbol with the probability precisely 0.5, except for the cases, when the symbol B follows with the probability 1.

We can clearly see the performance of each method (in the sense of convergence and final result). The weakest method has shown to be the BPTT(*h*). The Extended Kalman Filter converges more quickly and the final result is better by approximately 0.01 in comparison with the BPTT(*h*). The dominance of the filters UKF, UKFj, nprKF and nprKFj is evident. The convergence of all these four filters is rapid as opposed to the BPTT(*h*), and very quick when compared with the EKF. However, the most interesting is their final result. The differences are minimal - it is impossible to choose the “winner”; they all achieved the NNL which is *better* then “ideal”. We see the possible explanation of this phenomenon in the fact, that these filters helped the network to partially learn the system for generating the pseudo-random number applied during the word creation from Reber language.

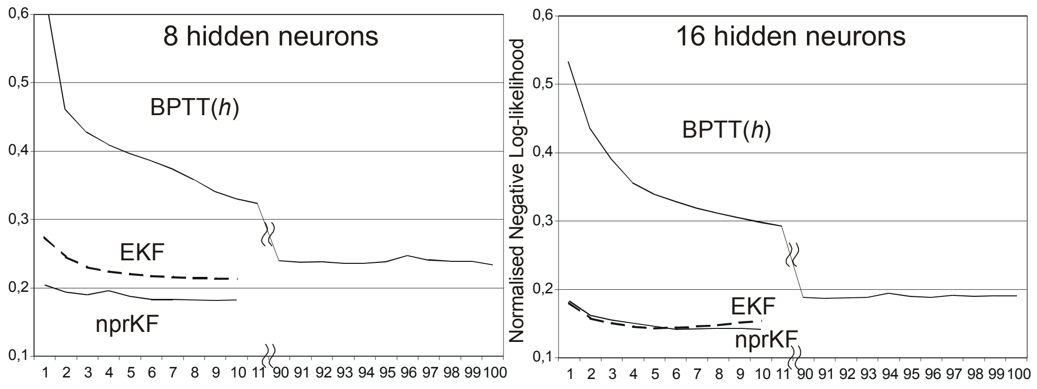

Fig. 2. The NNL dependence on the training cycles for various recurrent neural network training methods for the next symbol prediction in the “laser” sequence.

7.2 Laser in chaotic regime

The predicted sequence is based on real data obtained by quantising activity changes of laser in chaotic regime [1]. The bounds were chosen for positive small and big activity change and for negative small and big activity change. One symbol is assigned for each of these categories. The sequence therefore contains four distinct symbols, where relatively predictable subsequences are followed by much less predictable events. The sequence length is 10000 symbols; we can therefore predict 9999 symbols. The aim of this experiment is the performance comparison of various recurrent neural network training methods. Since the nprKF is comparable or better than other divided difference filters (also confirmed in other experiments), we compared it only with the EKF and BPTT(*h*).

We have used the training and testing procedure for the next symbol prediction from this sequence from [1]. We do not update the weights for the first 50 steps in order to lessen the impact of initial recurrent neurons output values. The training then takes place during next 7949 symbols. The remaining 2000 symbols forms the test data set, through which we compute the NNL. That terminates one cycle. Ten cycles are sufficient for the filters, 100 for the BPTT(*h*). We have used one-hot encoding for inputs as well as for outputs.

The results for 8 and 16 hidden recurrent neurons are showed in Figure 2. The figure shows that the Extended Kalman Filter is again significantly better than the

BPTT(*h*). Similarly the nprKF again achieves visibly better performance than the EKF. One can also spot the fact, that the more hidden neurons, the better performance.

8 Conclusions

We described the basic principle of the Extended Kalman Filter, Unscented Kalman Filter, the nprKF and their joint versions. The detailed description and formulas of these filters can be found in [6].

The main contribution of this paper is the direct comparison of all the filters in experiments, where they are used in the recurrent neural network training task for the next symbol prediction. In the results from these experiments one can see much better performance of the filters UKF, nprKF, UKFj and nprKFj (they all achieve comparable results) when compared with the EKF. On the other hand the EKF filter achieves definitely better results in comparison with the classic gradient method BPTT(*h*).

As far as we know, this paper describes the first application of the UKFj filter in the context of recurrent neural network training.

Acknowledgement: This work was supported by Scientific Grant Agency of Slovak Republic under grants #1/1047/04 and #1/0062/03, and by Science and Technology Assistance Agency under grant APVT-20-002504. I would like to thank Ing. Michal Čerňanský and Ing. Matej Makula for valuable advice and comments.

References

1. Čerňanský, M., Makula, M., Beňušková, Ľ.: Processing Symbolic Sequences by Recurrent Neural Networks Trained by Kalman Filter-Based Algorithms. In: *SOFSEM 2004: Theory and Practice of Computer Science*, Matfyzpress, Praha, Vol. 2 (2004), 58 - 65
2. Haykin, S., Puskorius, G.V., Feldkamp, L.A., Patel, G.S., Becker, S., Racine, R., Wan, E.A., Nelson, A.T., Rowels, S.T., Ghahramani, Z., van der Merwe, R.: *Kalman Filtering and Neural Networks*. John Wiley & Sons, NY, 2002.
3. Julier, S.J., Uhlmann, J.K.: A New Extension of the Kalman Filter to Nonlinear Systems. In: *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, Orlando, FL (1997)
4. Nørgaard, M., Poulsen, N.K., Ravn, O.: *Advances in Derivative-Free State Estimation for Nonlinear Systems*. Technical report, Revised edition, Technical University of Denmark, 2000.
5. Prokhorov, D.V.: Kalman Filter Training of Neural Networks: Methodology and Applications. In: *Int. Joint Conference on Neural Networks*, IJCNN2004 Tutorials, Budapest, Hungary (2004)
6. Trebatický, P.: The Recurrent Neural Network Training with the Extended Kalman Filter. *Neural Network World*. Submitted for publication.