Recurrent Neural Networks for State Estimation

Lee A. Feldkamp and Danil V. Prokhorov
Research and Advanced Engineering
Ford Motor Company
Dearborn, MI, USA

Abstract

We explore the use of recurrent neural networks (RNN) for state estimation, performing the same role as adaptive systems such as Kalman filters or particle filters. We demonstrate that RNN can combine available information from measurements with an implicit representation of state evolution to produce state estimates that are competitive with those produced by modern filter techniques.

1 Introduction

In earlier papers we have argued that time-lagged recurrent networks can be trained to perform tasks that are usually assigned to explicitly adaptive systems. We have argued further that the states of the dynamical system represented by the RNN play the role of the parameters of a conventional adaptive system.

In this paper, explore the extent to which a recurrent neural network can be trained to perform state estimation with results that compare favorably with those of adaptive filters. Our high-level justification for this exploration is that such adaptive systems may be viewed as nonlinear dynamical systems and thereby subject to being represented by recurrent networks. A more practical justification is that the elements of state estimation seem well within the capability range of properly trained RNN.

We emphasize that we are not contemplating the incorporation of neural networks to perform subtasks in a conventional adaptive filter structure. Instead, our approach is to supply the RNN with the available information and entrust structuring of the solution to the process by which the RNN is trained.

We hasten to point out that much the same approach was proposed almost ten years ago by Lo (see [1] and papers cited therein), claiming convergence to a minimum variance filter. However, the proof that supports this claim assumes effectively that estimation at time step t has available to it measurements for all previous time steps. This storage is accomplished for purposes of the proof with the aid of a special architecture that stores a transformed version of all previous measurement inputs to the network (thereby requiring a very large number of nodes). As this architecture is not practical for applications and is not used in the examples of [1], the utility of the proof is not clear.

In contrast to Lo, we make no claim that a minimum variance result can always be achieved, though we apply a far more powerful training method than is used in [1] and have no difficulty reproducing the quality of results presented there.

Recognizing that the forefront of state estimation has moved beyond the use of Kalman filters, whether linear or nonlinear, we take as our standard of comparison the results of approaches that approximate general Bayesian methods, such as particle filters.

2 State Estimation

2.1 General Remarks

Generically, we may regard state estimation as the combining of measurement information with that derived by evolving the state estimate at the previous time step. If the system is linear and process noise and measurement noise are both Gaussian, a Kalman filter is an obvious choice and is known to be optimal. Even when these ideal conditions do not hold, Kalman filters are often used, most commonly in the form of the extended Kalman filter (EKF). Recent developments in nonlinear Kalman filters [2, 3, 4] offer performance superior to the EKF when nonlinearity exists in either the process model or the measurement model, usually at the price of increased computation.

In some cases, e.g., severe nonlinearity, the underlying probability density functions may not be treated adequately as Gaussian, and it may be necessary to return to the general Bayesian framework (from which the Kalman filter can be derived); see ref. [5] for a recent tutorial. Grid methods and particle filters, recently the subject of considerable attention, are examples of methods that closely approximate the Bayesian approach under fairly general conditions. Such methods tend to be computationally intensive but may be the preferred choice for many applications. These methods produce a probability density function, the state estimate being a by-product. At times a more direct calculation of the state estimate, produced at smaller cost, could be advantageous.

2.2 Recurrent Network Approach

For definiteness, we consider here a specific class of state estimation problems, viz., those in which we wish to track the state of a low-dimensional nonlinear system observed with a potentially nonlinear measurement model. We allow both process and measurement noise to be of nontrivial magnitude. We assume that we know the initial value of the state vector, or at least its probability density function. In addition, we know the state equations (in discrete-time form), the measurement equations, and the process and measurement noise functions. Finally, we have measurements at each time step.

We want the (fixed) RNN to take on the complete calculation performed by a conventional adaptive filter for this task, but we do not attempt to force the RNN to mimic the parts of the filter process. For example, in order to balance information properly, the network may need effectively to evolve the state variables as in a Kalman filter, but we do not hand-craft the network architecture to make this happen.

We use knowledge of the process, measurement, and noise models to generate copious training data. Information, such as measurements, that is available at every time step is provided as part of the network input vector. One-time information, such as initial state values, is provided at the beginning of each trajectory. The desired state values are used as training targets, and we attempt to minimize summed square differences of network outputs from these targets. In this work, we are not considering the estimation of time-dependent variances or other measures of confidence in the state estimates, though one could attempt to learn them as well.

2.3 Expectations and Potential Pitfalls

We expect the RNN to combine explicit measurement information with that from its internal representation of state evolution. This should yield state estimates superior to that available from either information source separately. Naturally, we desire the RNN performance to be as close as possible to that best alternative method operating with the same information.

When either process noise or measurement noise is decreased (and reflected in the data generated for training), the estimation accuracy should improve. On-line response to changes in the noise levels is not considered here.

Because we treat the measurements as input to the network, our approach may be vulnerable to measurement functions that are not uniquely invertible. However, the examples we have selected have this property and our results suggest that it can be overcome.

Though we appeal to the very general representation power of RNN, we recognize that even if the architecture selected is capable of solving a particular task, the solution may not be realized through training.

3 Example 1

3.1 Problem Statement

We first consider the example discussed in [5] and treated in other papers cited therein. The process model is given by

$$\begin{aligned} x_k &= \frac{x_{k-1}}{2} + \frac{25x_{k-1}}{1+x_{k-1}^2} \\ &\quad + 8\cos(1.2k) + \nu_{k-1} \end{aligned} \qquad (1)$$

where the process noise ν_{k-1} is Gaussian with variance Q. The measurement model is

$$z_k = \frac{x_k^2}{20} + n_k \qquad (2)$$

where the measurement noise n_k is Gaussian with variance R. As base values for noise variances, we adopt $Q = 10$ and $R = 1$ as in ref. [5], but we report results for other values as well.

The state variable x_k is a highly nonlinear function of its value at the previous time step and is driven by the sinusoidal term and by the process noise. The measurement is ambiguous with respect to the sign of x_k. Hence, even in the limit of vanishing measurement noise, accurate state estimation requires knowledge of the state evolution sufficient to define the sign of x_k.

As this problem is treated in ref. [5], the sinusoidal term is regarded as known (i.e., k is always available) and the initial probability density function for x_0 is known. An alternative and more difficult setting assumes that the series starts with the sinusoidal term at an unknown value of k and that the network is provided with three consecutive values of x_k at the beginning of each trajectory.[1] We will primarily discuss the easier problem statement, but in Section 3.4 we report some results for this alternative.

3.2 RNN Training

We generated four training files, each of length 10 000, from equations 1 and 2, using the nominal values $Q = 10$ and $R = 1$ for generation of the noise terms. We chose a recurrent multilayer perceptron architecture of 3-5R-5R-1L, where the two hidden layers of bipolar sigmoid nodes are fully recurrent and the output node is linear. We choose starting points for training trajectories randomly within the files. At every step, the first network input is the current measurement z_k, and the second is the value of the sinusoidal term $8\cos(1.2k)$. For the first step i of a trajectory, the third network input is the mean value of x_i, i.e., $\frac{x_{i-1}}{2}+\frac{25x_{i-1}}{1+x_{i-1}^2}+8\cos(1.2i)$. For all steps thereafter, the third network input is zero. In this way, the RNN is given information equivalent to that used in the EKF, grid-based, and particle filter methods discussed in [5].

The training process was multistream GEKF, as described in [6]. We scaled network inputs and outputs by a factor of 0.1 for numerical purposes, but all results will be reported with the original scaling. We employed 40 streams, trajectory length 50, and priming length 3. Derivatives were computed with truncated backpropagation through time with truncation depth $h = 40$, though a smaller value would probably suffice. We trained for a total of 400 cycles, making 18 800 weight updates based on 752 000 instances. This required less than four minutes on a 2 GHz PC running the Linux operating system.

[1]In the absence of process noise, a feedforward network mapping from x_{k-3}, x_{k-2}, and x_{k-1} to x_k appears to be possible to essentially arbitrary accuracy, while accuracy with only two lagged values is limited.

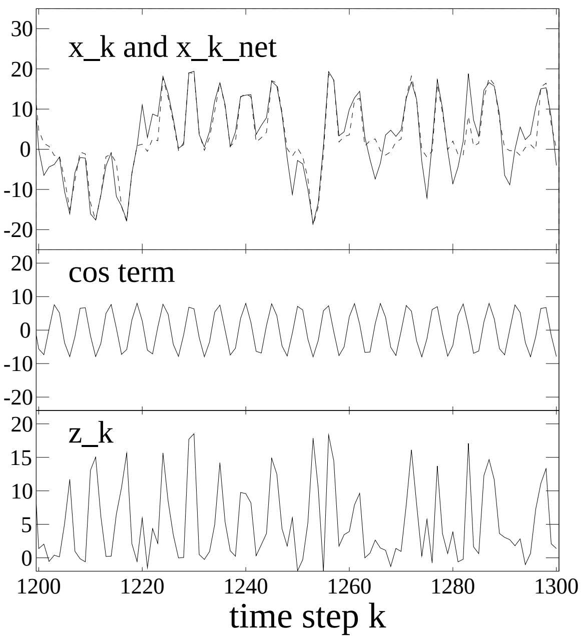

Figure 1: A short segment of testing results for RNN state estimation. The network inputs are shown in the lower two panels. The top panel displays the desired value for the state variable x_k and its estimate, drawn with a dashed line.

The RMS error over the training sets (exclusive of the first 3 points of each file) was 4.69. On two independently generated test files, the RMS errors were 4.71 and 4.72. To dismiss concerns about sensitivity to initial conditions, we also tested the trained network on all length-100 segments of the test files, with overall RMS error of 4.72.

The error distribution was peaked about zero, but with tails more extended than those of a Gaussian. A typical segment of results, taken from within one of the test files, is shown in Figure 1.

We also repeated the training process with two much larger networks. In the first of these, the architecture was 3-15R-10R-1L, resulting in test-file errors of 4.67 and 4.69. In the second case, the architecture was the same except that the third input, after the first step, received the network output with a one-step delay. The test-file errors were 4.69 and 4.72. The fact that these larger networks yield only marginal improvements suggests that the original network is capturing the essence of the problem to the extent consistent with the levels of process and measurement noise imposed.

3.3 Discussion of Results

In ref. [5], several different methods are applied to this problem. The authors demonstrate clearly that the extended Kalman filter is not effective, primarily because it cannot handle multi-modal probability distributions, as required by virtue of the measurement model. Much better results are obtained with a variety of grid-based and particle filters, all of which are suited to multi-modal and non-Gaussian probability distributions. The best result quoted for any of these methods is an RMS of 5.30. We carried out the grid-based procedure with 50 cells as described in [5], obtaining an error of 4.8, very close to our RNN result (it is not clear why our grid result is better than reported in that paper). Increasing the number of cells produced little improvement. We also applied a 300-particle SIR filter, again with an RMS error of 4.8. Hence we speculate that an RMS error of about 4.7 is close to irreducible. Further, the level of performance achieved using an RNN seems to be competitive with state-of-the-art alternatives, while almost surely being superior from the standpoint of computational cost.

If the sign ambiguity of x_k were to be entirely removed, but no other information about the process evolution is used, the error would depend only on the measurement noise. We simulated this situation by estimating $|x_k|$ as $y_x^{\frac{1}{2}}$ when $y_k > 0$ and as 0 otherwise, obtaining an RMS error of about 1.2.

We repeated the training procedure described above with $Q = 0$, i.e., no process noise, obtaining RMS errors of about 1.07 on the testing files. The reduction from the value of 1.2 attributable to measurement noise alone suggests that the network is making use of some evolution information. However, applying a grid-based method with 300 cells to this case, we were able to achieve an RMS error of about 0.5. This suggests that the 3-5R-5R-1L RNN used here is not fully exploiting information from model evolution. Speculating that for optimal performance in the limit of small process noise, the network architecture should be enlarged, we retrained with the larger networks mentioned above. The base 3-15R-10R-1L network gave test-file RMS errors of 0.794 and 0.794. Including output-to-input recurrence improved the results almost to the level of the 300-cell grid-based method, yielding errors of 0.612 and 0.518. This suggests that when the error in the output is small, as in the limit of small process noise, providing the output recurrently as a network input may be useful. In contrast, when the level of error is relatively large, feeding back the output may not be beneficial (for the base level of process noise, the RNN without output-to-input recurrence was, if anything, superior).

We carried out the training procedure in the limit of zero measurement noise, obtaining an RMS error of 4.48 on the testing files.

From these various experiments, the residual error in the base problem appears to result largely from the ambiguity in the sign of x_k, primarily as a result of process noise. As additional evidence, we note that if the measurement model is changed to $z_k = \frac{x_k^3}{20} + n_k$, which does not lead to sign ambiguity, much smaller errors are obtained (RMS values of 0.85 and 0.84 on the testing files).

3.4 Alternative Problem Statement

As mentioned in Section 3.1, we can specify a more difficult and in some respects more interesting problem from the same model. Suppose we are provided with the initial state of the system, in the form of three consecutive values of x_k for an arbitrary value of k, and then are required to estimate the sequence of values of x_k, given only the noisy measurement sequence. The sign ambiguity becomes particularly acute, because the only information that can distinguish x_k from $-x_k$ is available only at the beginning of a trajectory. Essentially, the estimating system must derive sign-defining information from the provided three values of x_k and then retain it throughout the trajectory. This is an example of a *strongly hidden state* [7].

We followed a training procedure similar to that of Section 3.2, except that it was necessary to increase the number of streams to 160. The network used was 2-5R-5R-1L, the first input being the measurement y_k and the second is the actual value of the state variable x_k for the first three steps of the trajectory, and zero thereafter. Training was carried out on four length-10 000 data files. To help the RNN to learn to disambiguate the sign, two of the files have sign-reversed targets relative to the other two. (Of course, the 3 values of x_k provided at the beginning of each trajectory are sign-reversed as well.)

Training proved to be much more difficult than for the base problem. However, after some experimentation, a robust procedure was found. On the training data, an RMS error of 5.08 was obtained, with comparable results on a pair of testing files. However, when we tested over all possible length-100 trajectories in the test files, we observed that for about 0.6% of the starting points the network produced x_k estimates that were approximately the negative of the correct value. Another 0.8% had a sign problem for at least part of the trajectory. We conclude that either or both of the following is true: 1) the three consecutive sequence values are not always enough to specify the subsequent signs; 2) the training process does not always embed the required sign cues stably into the network. This conclusion is consistent with our observation that increasing to four the number of values of x_k provided at the beginning of each trajectory greatly improved the failure rate, to 0.03%.

4 Example 2

Our second example is similar in structure to the first. For the process model we consider the Henon chaotic map

$$x_k = 1 - 1.4x_{k-1}^2 + 0.3x_{k-2} + \nu_{k-1} \qquad (3)$$

The measurement model is

$$z_k = x_k^2 + n_k \qquad (4)$$

This system has two state variables, which can be regarded as x_k and x_{k-1}. Both the process noise ν_{k-1} and the measurement noise n_k are Gaussian with variances Q and R, respectively. We chose $Q = 1 \times 10^{-5}$; values much larger than this were found to cause x_k to increase without bound. We will present results for three values of R: 0.0, 0.16, and 0.49. For ease of comparison, test files for these values were generated with the same sequence of random numbers.

The RNN training procedure was almost the same as that for the first example. The network used was 2-5R-5R-1L, where the first input is again the measurement y_k. The second input is the actual value of the state variable x_k for the first three steps of the trajectory, and zero thereafter. Training was carried out on a single length-10 000 data file and tested on an independently generated file of the same length. Details of the training procedure were unchanged from those described above.

In the absence of measurement noise, $R = 0$, we achieve an RMS error of 0.0029 on the test file. As seen in Figure 2, the network estimate is virtually indistinguishable from the desired value.

With measurement noise corresponding to $R = 0.16$, we get RMS error of 0.325 on the test file.

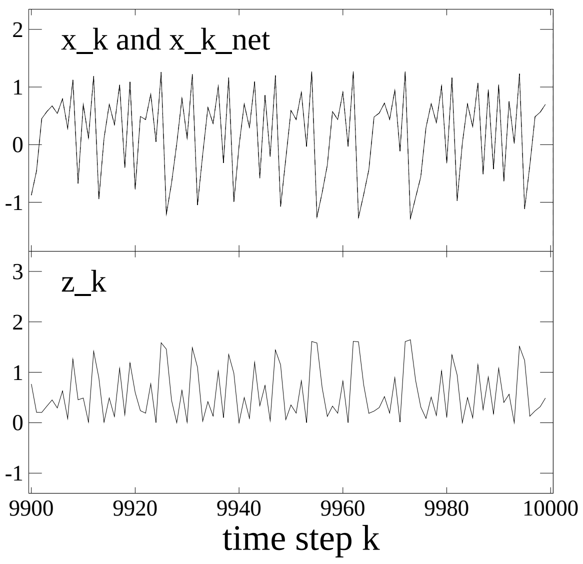

Figure 2: A segment taken from near the end of the file of testing results for RNN state estimation for the Henon system with $R = 0$. The network input is shown in the lower panel. The upper panel displays the desired value for the state variable x_k. The estimate of x_k, drawn with a dashed line, is barely visible.

From Figure 3 we see that the estimate follows the correct value most of the time.

Increasing the variance of the measurement noise to $R = 0.49$, we find the RMS error increasing to 0.522 and see from Figure 4 that the estimates are further degraded; this is not surprising in view of the large measurement noise (note how often z_k goes below zero).

Though we have not yet done so, it would be interesting to apply the grid-based method or a particle filter to this problem. For the former, a two-dimensional grid would be required, which would substantially increase its complexity and computational burden.

5 Conclusions

We have illustrated the use of recurrent neural networks in state estimation. On nontrivial problems, we find performance that is competitive with that of recently studied general methods. This performance varies in a reasonable manner with different underlying assumptions (such as noise levels). Note, however, that because such assumptions enter the training process at the data generation

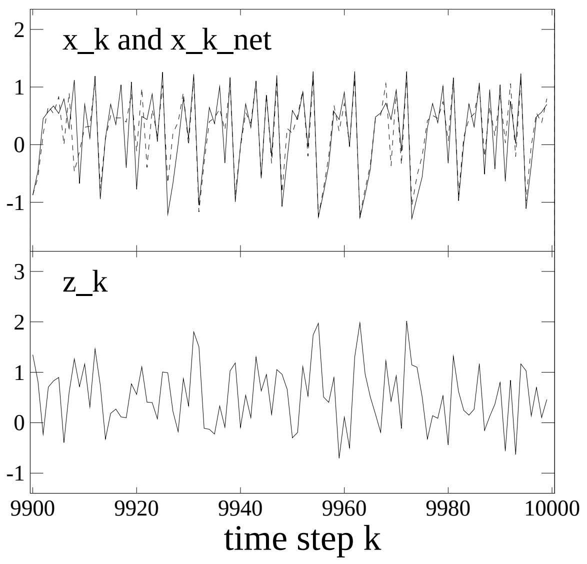

Figure 3: A segment of testing results for RNN state estimation for the Henon system with $R = 0.16$. The network input is shown in the lower panel. The upper panel displays the desired value for the state variable x_k and its estimate, drawn with a dashed line.

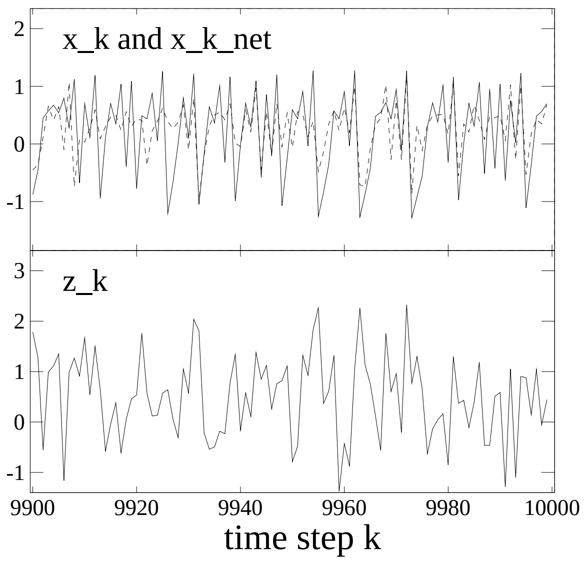

Figure 4: A segment of testing results for RNN state estimation for the Henon system with $R = 0.49$. The network input is shown in the lower panel. The upper panel displays the desired value for the state variable x_k and its estimate, drawn with a dashed line.

stage, handling model changes during operation is not as straightforward as for a parametric system. In particular, one might have to switch from one RNN to another. Alternatively, a single RNN could be trained over a range of models, with or without explicit information describing the current model.

Finally, we note that the RNN approach handles extreme cases, such as the absence of measurement noise, more gracefully than do particle filter methods.

References

[1] J. T. Lo. "Synthetic approach to optimal filtering." *IEEE Transactions on Neural Networks*, vol. 5, pp. 803–811, 1994.

[2] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. "A new approach for filtering nonlinear systems." *Proceedings of the 1995 American Control Conference*, Seattle, Washington, pp. 1628–1632, 1995. Several later papers may be obtained from the authors. Cf. http://www.ox.ac.uk/∼siju.

[3] M. Norgaard, N. K. Poulsen, and O. Ravn. "Advances in derivative-free state estimation for nonlinear systems." Technical Report IMM-REP-1998-15 (revised edition), Technical University of Denmark, 2000. Cf. http://www.imm.dtu.dk/nkp/. A toolbox and report may be found at http://www.iau.dtu.dk/research/control/kalmtool .html.

[4] E. A. Wan and R. van der Merwe. "The unscented Kalman filter," in S. Haykin (ed), *Kalman Filtering and Neural Networks*, Wiley, New York, pp. 221–280, 2001.

[5] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174-188, 2002.

[6] L. A. Feldkamp and G. V. Puskorius, "A signal processing framework based on dynamic neural networks with application to problems in adaptation, filtering and classification," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2259–2277, 1998.

[7] R. J. Williams, "Adaptive state representation and estimation using recurrent neural networks," in W. T. Miller, III, R. S. Sutton, and P. J. Werbos (eds), *Neural Networks for Control*, pp. 97–114. Cambridge, MA: MIT Press, 1990.