Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007)

Efficient Learning in Cellular Simultaneous Recurrent Neural Networks - The Case of Maze Navigation Problem

Roman Ilin
Department of Mathematical Sciences
The University of Memphis
Memphis, TN 38117
E-mail: rilin@memphis.edu

Robert Kozma
Department of Mathematical Sciences
The University of Memphis
Memphis, TN 38117
E-mail: rkozma@memphis.edu

Paul J. Werbos
Room 675, National Science foundation
Arlington, VA 22230
E-mail: pwerbos@nsf.gov

***Abstract*— Cellular Simultaneous Recurrent Neural Networks (SRN) show great promise in solving complex function approximation problems. In particular, approximate dynamic programming is an important application area where SRNs have significant potential advantages compared to other approximation methods. Learning in SRNs, however, proved to be a notoriously difficult problem, which prevented their broader use. This paper introduces an extended Kalman filter approach to train SRNs. Using the two-dimensional maze navigation problem as a testbed, we illustrate the operation of the method and demonstrate its benefits in generalization and testing performance.**

I. INTRODUCTION

Modern control techniques are rooted in the concept of dynamic programming, which allows to plan for the best course of action in a multistage decision problem [1]. Given a Markovian decision process with N possible states and the immediate expected cost of transition between any two states i and j denoted by $c(i, j)$, the optimal cost-to-go function for each state satisfies the following Bellman's optimality equation:

$$J^*(i) = min_\mu(c(i, \mu(i)) + \gamma \sum_{j=1}^{N} p_{ij}(\mu) J^*(j)) \qquad (1)$$

$J(i)$ is the total expected cost from the initial state i, and γ is the discount factor. The cost J depends on the policy μ, which is the mapping between the states and actions causing state transitions. The optimal expected cost results from the optimal policy $\mu*$. Finding such policy directly from Eq. 1 is possible using recursive techniques but computationally expensive as the number of states of the problem grows. The concept of approximate dynamic programming (ADP) refers to techniques used to estimate the exact solution to the Bellman's optimality equation. Neural networks are a very useful technique which has been successfully applied to ADP, see, e.g., [2], [3].

[0]The opinions expressed in this paper are of the authors and do not necessarily reflect the views of their employers, in particular that of NSF.

Artificial neural networks, inspired by the enormous capabilities of living brains, are one of the cornerstones of today's field of artificial intelligence. Their applicability to real world engineering problems has become unquestionable in the recent decades; see for example . Yet most of the networks used in the real world applications use the feed-forward architecture, which is a far cry from the massively recurrent architecture of the biological brains. The introduction of recurrent elements makes training more difficult and even impractical for most non-trivial cases. Nevertheless, the power of recurrent networks for function approximation has been proven to exceed the power of feed-forward networks [4], [5], which means that the attempts to apply the former must continue.

It is well-known that MLP's and a variety of kernel-based networks (like RBF) are universal function approximators. However, when the function to be approximated does not live up to the usual concept of smoothness, or when the number of inputs becomes even larger than what an MLP can readily handle, it becomes important to use a more general class of neural network. The J-function that has to be approximated in order to solve the Maze Navigation problem is extremely challenging. Previous attempts to solve it using the MLP's [4] were unsuccessful thus proving that the Maze problem and probably the ADP problems in general are beyond the capabilities of the feed-forward networks.

We introduce Cellular Simultaneous Recurrent Neural Network (SRN) architectures for solving Dynamic Programming problems. We use the Extended Kalman Filter (EKF) methodology for training the net and we obtain very good training and testing results. This is a novel computationally efficient training methodology to the complex recurrent network architecture. Preliminary results have been presented in [6]. Our results represent a decisive step towards making the powerful methodology of recurrent networks suitable for numerous practical applications. We demonstrate the applications of the introduced method on the 2D maze problem.

1-4244-0706-0/07/$20.00 ©2007 IEEE

II. Simultaneous Recurrent Neural Networks

Simultaneous recurrent neural networks are widely used in the literature. Here the main features are summarized [2], [5]. SRN's can be used for static functional mapping, similarly to the MLP's. It has been shown experimentally that an arbitrary function generated by a MLP can always be learned by an SRN. However the opposite was not true, as not all functions given by a SRN could be learned by a MLP. These results support the idea that the recurrent networks are essential in harnessing the power of brain-like computing.

SRNs are different from more widely known time lagged recurrent networks (TLRN) in that the input is applied over many time steps and the output is read after the initial transitions have died out and the network is in equilibrium state. The concept is illustrated in Fig. 1.

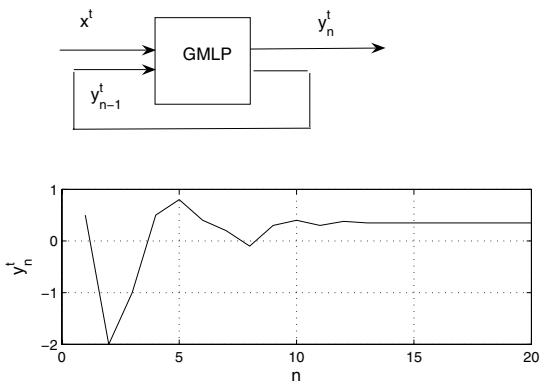

Fig. 1. SRN is a recurrent neural network used for static functional mapping. The superscript t refers to the current training or testing epoch, and the subscript n refers to the current iteration of the SRN. The network input x^t is applied over many network iterations. The output y^t_n gradually converges to a steady value which is taken to be the output of the network. An example of the y^t_n sequence is given on the graph below. Note that the core of the SRN can be any feedforward network. In our case, it is a generalized MLP [2].

.

Many real life problems require to process patterns that form a 2D grid. Such problems arise in image processing, for example, or in playing a game of chess. In such cases, the structure of the neural network should also become a 2D grid. The idea of cellular network is to utilize the symmetry of the problem. If we make all the elements of the grid identical, the resulting cellular neural network benefits from greatly reduced number of independent parameters. The combination of cellular structure with SRN provides a potentially very powerful function approximator.

Training of recurrent networks can be done using back-propagation through time. BPTT extends the classical back propagation by "unfolding" the recurrent network. Imagine that instead of recurring back to themselves, the recurrent links of the network feed forward into a copy of the same network. Let us keep making many copies like this for 10, 20 iterations. If the network comes to steady state, the outputs will stop changing after finite number of iterations; and so we can stop replicating the network and say that our multilayered feed forward network is equivalent to the original recurrent network. It can now be trained using regular back propagation. The only problem is that the weights in each "layer" must stay the same, we cannot adjust each weight independently as we would in a MLP. Usually, weight adjustment is done by summing up all the derivatives and making one change corresponding to the sum. In the case of cellular SRN, the derivatives also have to be summed over each cell of the maze. Such summations impair the efficiency of learning. As it was mentioned above, BPTT was successfully applied to the maze navigation but the learning was slow [4]. We apply Extended Kalman Filter to overcome this adaptation convergence bottleneck.

III. Extended Kalman Filter Learning Method

The initial idea of the improved implementation of EKF for training SRN is given in [6]. Details of the applied methodology are given in a forthcoming publication. An excellent treatment of training the neural networks with Kalman filters is given in [7]. Important overview of neurocontrol applications are given in [8]. Kalman filters present a computational technique which allows to estimate the hidden state of a system based on observable measurements. The estimation is done iteratively, with the state estimate improved with each new measurement. In the case of a neural network, the set of weights becomes the state vector, and the network outputs become the measurement vector. The EKF operates with the following two equations.

$$\vec{w}_{t+1} = \vec{w}_t + \vec{\omega}_t \tag{2}$$

$$\vec{y}_t = F(\vec{w}_t, \vec{u}_t) + \vec{\nu}_t \tag{3}$$

Equation 2 is known as the process equation. It describes our hypothesis about how the state of the system changes over time. In our case the "true" weights do not change. ω_t is the process noise. Equation 3 is the measurement equation. It represents our hypothesis about the dependency between the hidden state of the system $\vec{w}_t$ and the observable measurements $\vec{y}_t$. In case of neural network, the measurements are the outputs of the network. $\vec{\nu}_t$ is the measurement noise.

Neural network training using EKF results in finding the minimum mean-squared error estimate of the state $\vec{w}_t$ using measurements observed prior to the time t. The following equations (4 - 7) describe the recursive algorithm.

$$\Gamma_t = C_t K_t C_t^T + R_t \tag{4}$$

$$G_t = K_t C_t^T \Gamma_t^{-1} \tag{5}$$

$$\vec{w}_{t+1} = \vec{w}_t + G_t \vec{\alpha}_t \tag{6}$$

$$K_{t+1} = K_t - G_t C_t K_t + Q_t \tag{7}$$

Matrix C is the Jacobian matrix of the measurement equation 3, which is a linearized function F. K is the error covariance matrix. It is recursively updated by equation 7. K encodes the second derivative information. R is the process noise covariance matrix. Q is the measurement noise covariance matrix. They are the tunable parameters of the algorithm. G is the Kalman gain matrix. α is the error vector, $\vec{\alpha}_t = \vec{\bar{y}}_t - \vec{y}_t$. Here, $\vec{\bar{y}}_t$ is the desired network output.

The original Kalman filter is an exact technique for linear systems with Gaussian process and measurement noise. The extended Kalman filter, applied to the non-linear systems, is already an approximation to the exact technique. The application of the EKF to the cellular SRN introduces another level of approximation due to the summations of the derivatives over space dimension, similarly to BPTT. However, unlike the direct summation in BPTT, the derivatives in the space dimension are weighted by the matrix K.

The tunable parameters of the system are the process noise matrix, the measurement noise matrix, and the initial error covariance matrix. The latter is initialized to be a diagonal matrix with diagonal elements between 0 and 0.00001. The process noise is zero. The measurement noise matrix R has to be annealed as the learning progresses. It turned out that the way the measurement noise is annealed has significant effect on the rate of convergence. After experimenting with different functional forms we stopped on the following formula:

$$R_t = 0.001 * log(0.001 * \vec{\alpha}_t^2 + 1)I \tag{8}$$

Here I is the identity matrix, and $\vec{\alpha}_t^2$ is the squared error at time t. Making the measurement noise a function of squared error results in fast and reliable learning. There are further practical issues related to the implementation of EKF which are not addressed here, Interested readers are referred to [6].

IV. Experimental Results using the 2D Navigation Problem

The goal of the generalized maze navigation is to find the optimal path from any initial cell position to the goal in a 2D grid world. An example of such a world is given in Fig. 2. One version of an algorithm for solving this problem will take a representation of the maze as its input and return the length of path from each clear cell to the goal. So, for a 5 by 5 maze, the output will consist of 25 numbers. Knowing the numbers, it is very easy to find the optimal path from any cell by simply following the minimum among the neighbors.

Previous results of training the Cellular SRN's showed slow convergence [4]. Those experiments used back-propagation with adaptive learning rate (ALR). The network consisted of 5 recurrent nodes in each cell and was trained on up to 6 mazes. The initial results demonstrated the ability of the network to learn the mazes.

The introduction of EKF significantly speeded up the training of the Cellular SRN. In the case of single maze, the network reliably converges within 10-20 epochs. In comparison, Back-propagation through time with adaptive learning rate (ALR) takes between 500 and 1000 iterations and it is sensitively dependent on the initial network weights [4]. We found that increasing the number of recurrent nodes from 5 to 15 allows to speed up both EKF and ALR training in case of multiple mazes. Still, EKF has a clear advantage as will be described below.

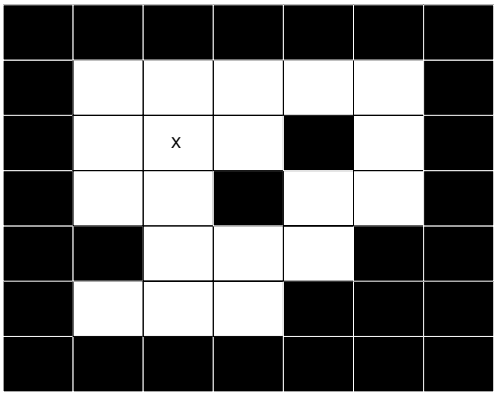

Fig. 2. An example of a 5 by 5 maze. X is the goal. Black cells are obstacles and white cells are clear. The walls around the maze make its size 7 by 7.

We introduce the measure of goodness of navigation achieved with the trained network. The gradient of the J function gives the direction of the next move. As an example, Fig. 3 shows the J function computed by a network and the true J function. We count the number of correct gradient directions. The ratio of the number of correct gradients to the total number of gradients is our goodness ratio G that can vary from 0 to 1. Following this definition, a randomly generated network will result in G $\approx$ 0.5, i.e., there is 50% chance of the correct direction between two neighboring cells. If we use $G = 0.5$ as the base line, the ratio $R = (G - 0.5)/0.5$ indicates the improvement of the solution over the base line.

As we increase the number of training mazes, the generalization capability of our network improves. The following figures (Fig. 4, 5, 6) show the goodness percentage G during the first 100 training steps, where training is done on 5,10, and 20 mazes. We can see that with few mazes the EKF training reaches high level of G very fast however the testing G remains low indicating that the network does not generalize well. As the number of mazes increases, the testing G begins to improve. For comparison, the training and testing results of ALR are shown on the same graph.

We noticed that we have to use between 25 and 30 training mazes in order to have the testing error close to the training error indicating good generalization. Accordingly we ran experiments with 25 through 30 training mazes. For each experiment we generated random mazes with 6 obstacles. We randomly initialized the SRN. Then we train the same

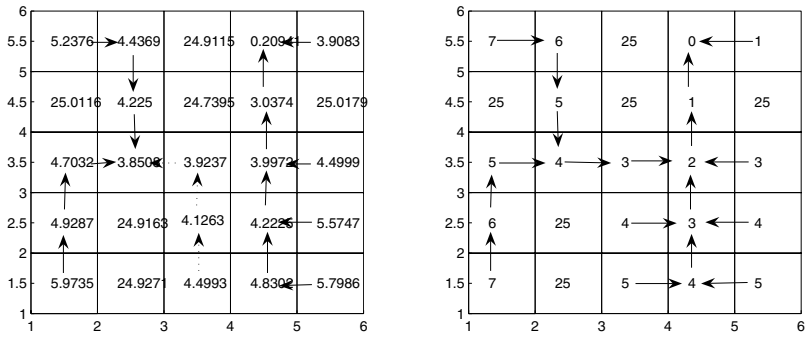

Fig. 3. Comparison of the solution given by the network and the true solution. The dotted arrows point in the wrong direction. Not all arrows are shown to avoid overcrowding.

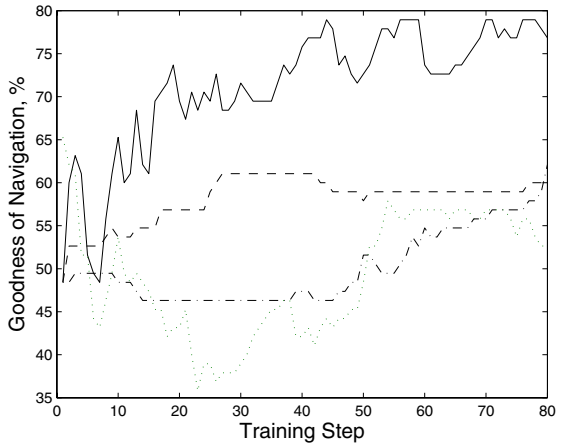

Fig. 4. Goodness of navigation ratio during the first 100 training steps. Training is on 5 mazes and testing is also on 5 mazes. Solid line - EKF training , dotted - EKF testing , dashed - ALR training , dash-dotted - ALR testing.

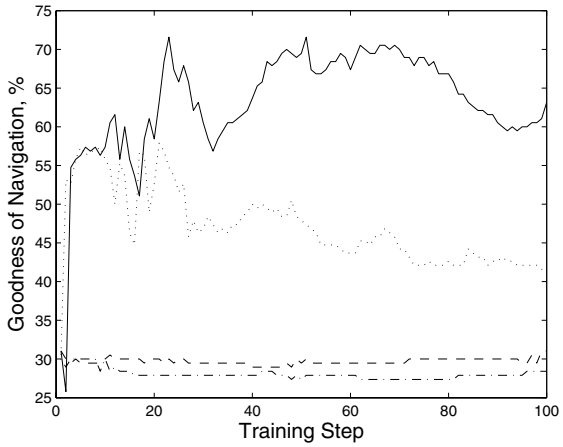

Fig. 5. Goodness of navigation ratio during the first 100 training steps. Training is on 10 mazes and testing is also on 10 mazes. Solid line - EKF training , dotted - EKF testing , dashed - ALR training , dash-dotted - ALR testing.

network with EKF and with ALR for 100 epochs. The results of experiments follow a similar pattern shown in Fig. 7 and 8.

The Kalman filter achieves good level of G very fast. The ALR is learning slowly and there is practically no improvement during the 100 steps. Notice that the testing error in the given example is smaller that the training error. This means that at this level of training error the solution is not meaningful at all. Not so with EKF which achieves the level of error identifiable with good solution.

The version of EKF used in these experiments is prone to divergence. After achieving good level of G the solution may deteriorate.Such phenomenon should be avoided in the future by use of more advanced EKF techniques. Nevertheless, even our straightforward implementation of EKF shows great improvement over ALR. The average values of R for 25 through 30 mazes are plotted in Fig. 9. We can see that the EKF consistently solves the problem to a certain meaningful level as opposed to ALR consistently solving the maze problems typically not much higher than the chance level.

V. Conclusions and Future Research

We use Extended Kalman Filters for training cellular SRNs. We obtain good training and testing results which are typically much better than alternative methods, i.e., as backpropagation with adaptive learning rate. Training with EKF quickly approaches 80% correct performance rate after less than 100 iteration steps. The same network trained with ALR usually does not achieve correctness larger than 0.6 in 100 iterations, and may take 10 times as long training or more to achive the preformance level of EKF. Moreover, ARL training is often stuck stuck in a local minimum. Our results represent a significant step towards establishing the powerful methodology of simultaneous recurrent networks suitable for numerous practical applications. This may help the proliferation of SRN method to new application areas in the future.

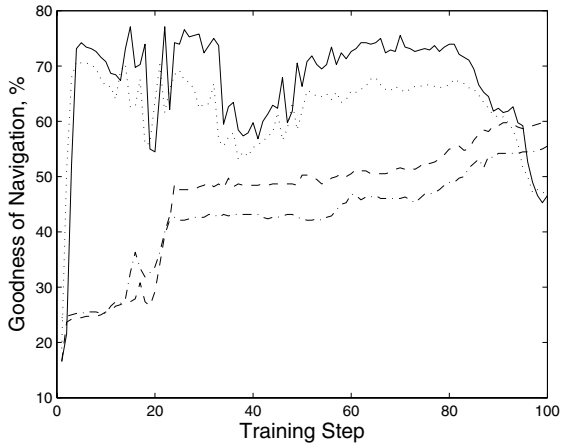

Fig. 6. Goodness of navigation ratio during the first 100 training steps. Training is on 20 mazes and testing is also on 20 mazes. Solid line - EKF training , dotted - EKF testing , dashed - ALR training , dash-dotted - ALR testing.

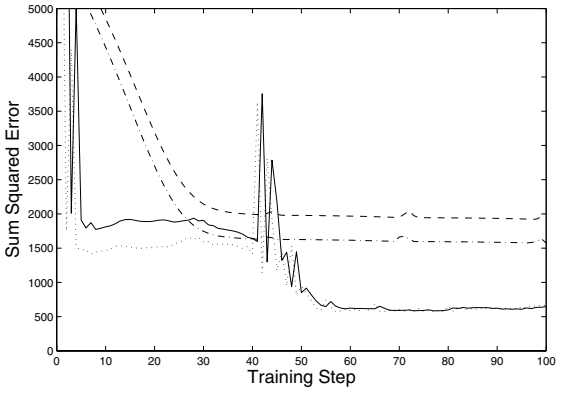

Fig. 7. Sum Squared error during the first 100 training steps. Training is on 29 mazes and testing is also on 29 mazes. Solid line - EKF training error, dotted - EKF testing error, dashed - ALR training error, dash-dotted - ALR testing error.

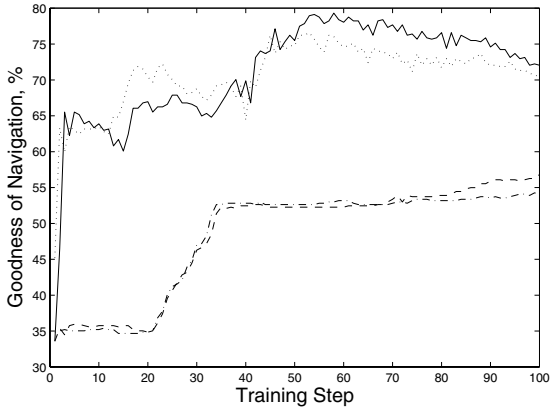

Fig. 8. Goodness of navigation ratio during the first 100 training steps. Training is on 29 mazes and testing is also on 29 mazes. Solid line - EKF training , dotted - EKF testing , dashed - ALR training , dash-dotted - ALR testing.

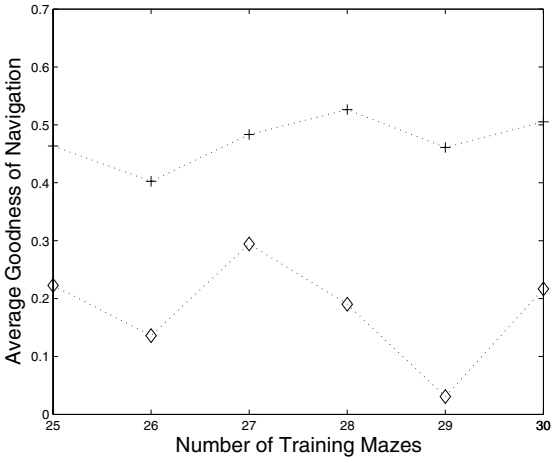

Fig. 9. Comparison of average improvement ratio R of EKF and ALR training of Cellular SRN over the first 100 training steps. Each point is the average of 5 experiments. Plus is EKF and diamond is ALR.

The biological plausibility of the Cellular SRN can be further improved by the introduction of spatial and temporal discount functions. Currently only the direct neighbors provide input to a node. Spatial discount means the introduction of long range neighboring connections with discounting factor depending on the distance between the neighbors. Temporal discount means introducing the discounting factor into the summation of back propagation derivatives.

Spatio-temporal discount functions will likely play an important role of the solution of mixed forwards-backwards stochastic differential equations. Such models are useful in optimal control, time series prediction, and other fields. They may match or exceed the capabilities of Dual Heuristic programming related to the Pontryagin equation [2], [9], [10], [3]. These issues are the topic of ongoing studies and will be introduced in future reports.

VI. Acknowledgements

Valuable discussions with Danil Prokhorov are greatly appreciated.

References

[1] S. Haykin, Neural Networks, A Comprehensive Foundation, Pearson Education, Inc., 1999.
[2] D.A. White and D.A. Sofge, Handbook of Intelligent Control Neural, Fuzzy, and Adaptive Approaches, ch. 3, Van Nostrand Reinhold, 1992.
[3] D. Prokhorov and D. Wunsch, "Adaptive critic designs," IEEE Trans. on Neural Networks, vol.8, pp.997–1007, 1997.
[4] P.J. Werbos and X. Pang, "Generalized maze navigation: Srn critics solve what feedforward or hebbian cannot," Proc. Conf. Systems, Man, Cybernetics, 1996.

[5] G.K. Venayagamoorthy and G. Singhal, "Quantum-inspired evolutionary algorithms and binary particle swarm optimization for training mlp and srn neural networks," Journal of Computational and Theoretical Nanoscience, vol.2, pp.1–8, 2005.
[6] R. Ilin, R. Kozma, and P.J. Werbos, "Cellular srn trained by extended kalman filter shows promise for adp," Proc. World Congress on Computational Intelligence WCCI06, 2006.
[7] S. Haykin, ed., Kalman Filtering and Neural networks, John Wiley and Sons, Inc., 2001.
[8] D. Prokhorov, R. Santiago, and D. Wunsch, "Adaptive critic designs: A case study for neurocontrol," Neural Networks, vol.8(9), pp.1367–1372, 1995.
[9] P.J. Werbos, Origins: Brain and Self-Organization, ch. Self-organization: re-examining the basics and an alternative to big bang, Erlbaum, 1994.
[10] W.J. Freeman, R. Kozma, and P. Werbos, "Biocomplexity: adaptive behavior in complex stochastic dynamical systems," BioSystems, vol.59, pp.109–123, 2001.