A Recurrent Anticipatory Self-Organized Map for Robust Time Series Prediction

Samarth Swarup, Kiran Lakkaraju, Nicholas A. Smith, and Sylvian R. Ray
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois. USA
swarup@uiuc.edu

Abstract - *In robotics applications, we often have noisy data that have temporal constraints due to the real world, such as sensory sequences generated by the motion of a robot through the environment. To recover the underlying structure of this noisy stream of data, we can do clustering with a self-organized map (SOM). We can view the SOM as a generative model, and it is seen to be doing maximum-likelihood estimation in that case. It has been shown that when the SOM is working with a temporal stream of data, it is possible to convert it to a maximum a posteriori estimator by adding lateral weights between the SOM nodes. However, this method has the drawback that it assumes that the temporal data are first order Markovian. In this paper we show how to overcome this limitation. We use a simple recurrent network (SRN) to predict the priors for the SOM. Since the SRN is capable of learning sequences in the data, this overcomes the main limitation of the previous model. We also bring out the connection between our model (the RecAntSOM) and the standard predictor-corrector framework of filtering. We do some simple robot experiments to show the usefulness of our model.*

Key words - Time-series analysis, robotics, signal-processing.

1 Introduction

The self-organizing map [7] has proven to be a very useful tool for dealing with noisy real-world situations. However it completely ignores the temporal aspect that is present in most real-world applications. There have been several attempts in the past to extend the SOM to deal with temporal data, such as [1, 3, 4, 10, 11, 14, 15]. Most of these try to cluster temporal sequences directly. For instance, in the Temporal Kohonen Map, the winning neuron is the the neuron that is closest to all elements of a sequence, with a higher weight for more recent elements. In the recursive SOM, the context is used to represent the previous activation of the SOM. SOMSD is similar, but instead of using the previous activation of the SOM, it uses just the index of the winner node in the previous time step.

In a sense, we also follow this paradigm, where the activation of the previous time step affects the activation in the current time step. But whereas in the other models the activation of the previous time step is directly compared to the node (via a secondary weight vector), in our model the previous activation is used to bias the winner at the current time step. This

allows us to give a clear probabilistic interpretation of the computation being performed.
Our goal is to improve the robustness of the network with respect to noise. It is not clear if the other temporal extensions of the SOM achieve this. To study this problem, we attempted to split up the clustering and the sequencing aspects of the problem, resulting in the anticipatory self-organizing map (AntSOM) [12]. The model we present here is an extension and improvement to the AntSOM, which was only capable of dealing with first order Markovian tasks. The new model, which we call the Recurrent Anticipatory Self-Organizing Map or RecAntSOM, discovers the appropriate depth of the sequences itself, and can be shown to still be computing optimal posterior estimates.
The rest of this paper is organized as follows. In the next section we present the details of the new model, and in the section after that we show its connection to predictor-corrector filtering. After that we present some simple robotic experiments where we compare the performance of the RecAntSOM against the AntSOM and the plain SOM. Finally, in the conclusion, we talk very briefly about the biological relevance of the model, and possible future work.

2 The Model

The model consists of a self-organized map (SOM), which combines the current input with the previous activations to determine current activations. Training proceeds in two stages. The first stage consists of a traditional SOM, where the activation $y_{ij}(t)$, of node (i, j) in the SOM, at time t, is calculated as follows:

$$y_{ij}(t) = \frac{\alpha_{ij}(t)}{1 + e^{-\frac{1}{\beta_{ij}(t)} \cdot u_{ij}(t)}}. \tag{1}$$

This is basically a sigmoid function with some modifications. It is parameterized using β_{ij} and b_{ij} (see equation 2) so that we can fit it to the distribution of points around each cluster center. This enables us to produce better estimates of the likelihood of a point having been generated by this cluster. The value $\alpha_{ij}(t)$ is used during inference, during stage 1 it is set to 1 in equation 1.
The net input, $u_{ij}(t)$, is given by

$$u_{ij}(t) = 1 - \frac{d(\mathbf{w}_{ij}(t), \mathbf{x}(t))}{K} - b_{ij}(t), \tag{2}$$

where $\mathbf{x}(t)$ is the input vector at time t, $\mathbf{w}_{ij}(t)$ is the weight vector at node (i, j) at time t, $d(\mathbf{w}_{ij}(t), \mathbf{x}(t))$ is the Euclidean distance between $\mathbf{w}_{ij}(t)$ and $\mathbf{x}(t)$, K is the maximum possible value of $d(\mathbf{w}_{ij}(t), \mathbf{x}(t))$, and $b_{ij}(t)$ is the bias input for node (i, j) at time t. This somewhat odd function is chosen to make the activation large (close to 1) when the input and the weight vector are close to each other.
The weights are updated as follows.

$$\mathbf{w}_{ij}(t+1) = \mathbf{w}_{ij}(t) + \eta \cdot h_{ij} \cdot \mathbf{x}(t), \quad \forall i,j, \tag{3}$$

where η is the learning rate, and h_{ij} is a neighborhood function, given by

$$h_{ij} = exp(-\frac{d^2_{win}(i,j)}{2\sigma^2}), \tag{4}$$

where $d_{win}(i,j)$ is the Euclidean distance of node (i,j) from the winning node.
After the weights of a node have been updated, they are normalized using the L2 norm

$$w_{ijk} = \frac{w_{ijk}}{\|\mathbf{w}_{ij}\|}, \quad \forall k \tag{5}$$

The bias, b_{ij} is updated as follows:

$$b_{ij}(t+1) = b_{ij}(t) + \eta \cdot h_{ij} \cdot u_{ij}(t) \tag{6}$$

β_{ij} is updated as follows:

$$\beta_{ij}^2(t+1) = \beta^2(t) + \eta \cdot h_{ij} \cdot (u_{ij}^2(t) - \beta_{ij}^2(t)) \tag{7}$$

β_{ij} and b_{ij} are used to fit the logistic function at each node to the distribution of points around it. β_{ij} estimates the standard deviation, and b_{ij} estimates the mean of the distances of the points from the cluster center.
As usual, η, the learning rate, starts out fairly large and is decayed over time. After stage 1 training, the weights are frozen and stage 2 training commences.
In stage 2, we train a simple recurrent network to predict the activations of the SOM nodes. The output of the SOM at each time step forms the input to the SRN, and the output of the SOM at the following time step forms the expected output for the SRN. Since the expected output is available at every time step, training the SRN is a simple matter, and is done using the error backpropagation algorithm [5].
During inference, the $\alpha_{ij}(t)$ values are set to be the same as the outputs of recurrent network.

3 Predictor-Corrector Filters

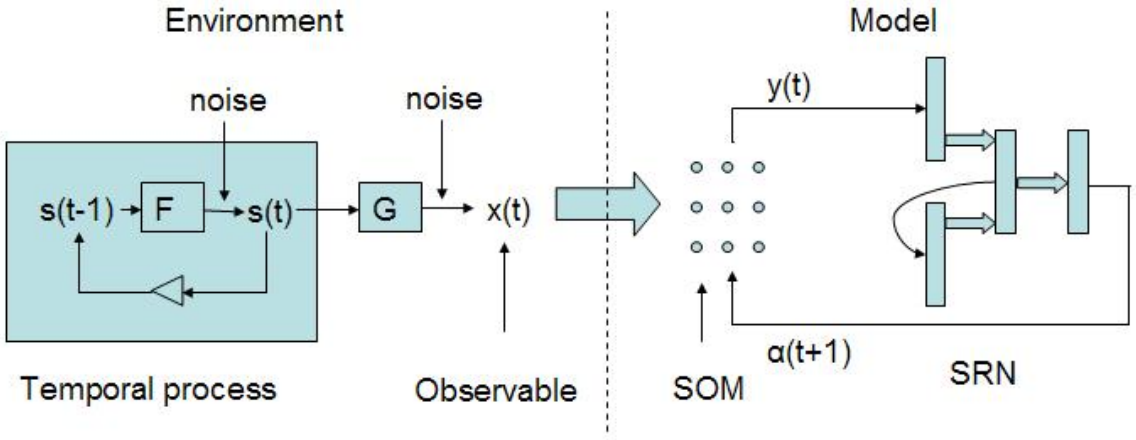

Figure 1: The RecAntSOM mimics the temporal process in the environment by using a self-organizing map and a simple recurrent network. The SOM activations are the "state" of the model, and the SRN estimates the process F. (Figure adapted from [8]).

Here we review basic notions of predictor-corrector filtering, and show that the computation performed by the RecAntSOM fits into this framework.

The notion of predictor-corrector filtering was first given by Kalman, in describing what is now known as the Kalman filter [6]. Suppose that there is an environmental process, with a state $s(t)$, which is hidden from us (the observer). Since the Kalman filter is linear, it assumes that this hidden state evolves linearly according to a process equation,

$$\mathbf{s}(t+1) = \mathbf{F}\mathbf{s}(t) + \mathbf{m}(t), \tag{8}$$

where $\mathbf{F}$ is a square *process* matrix and $\mathbf{m}(t)$ is a noise vector, assumed to be Gaussian and white. The observable vector, $\mathbf{x}(t)$, is generated by the observation process, which is also assumed to be linear.

$$\mathbf{x}(t) = \mathbf{G}\mathbf{s}(t) + \mathbf{n}(t), \tag{9}$$

where $\mathbf{G}$ is the not necessarily square *observation* matrix, and $\mathbf{n}(t)$ is a noise vector assumed to be Gaussian, white, and zero mean. The filtering task is to optimally estimate the hidden state vector $\mathbf{s}(t)$ from the observations $\mathbf{x}(0)$ to $\mathbf{x}(t)$. The Kalman filter is recursive, which means that it does not need to keep the entire history of observations. It optimally estimates the value of $\mathbf{s}(t)$ from its estimate of $\mathbf{s}(t-1)$ and the new observation, $\mathbf{x}(t)$. The Kalman filter is found by minimizing the weighted least-squares criterion,

$$J = (\mathbf{x} - \mathbf{G}\mathbf{s})^T \mathbf{N}^{-1} (\mathbf{x} - \mathbf{G}\mathbf{s}) + (\mathbf{s} - \bar{\mathbf{s}})^T \mathbf{M}^{-1} (\mathbf{s} - \bar{\mathbf{s}}). \tag{10}$$

We have momentarily dropped time from the equation for notational convenience. $\mathbf{N}$ and $\mathbf{M}$ are the covariance matrices for the noise vectors, and $\bar{\mathbf{s}}$ is the mean of the state vector. Minimizing this expression can be shown to give us the Kalman filter equation,

$$\hat{\mathbf{s}}^+(t) = \hat{\mathbf{s}}^-(t) + \mathbf{K}(t)(\mathbf{x}(t) - \mathbf{G}\hat{\mathbf{s}}^-(t)), \tag{11}$$

where $\hat{\mathbf{s}}^+(t)$ is the *maximal a posteriori* estimate of the state, $\hat{\mathbf{s}}^-(t)$ is the prior estimate (i.e. before the observation is made), and $\mathbf{K}(t)$ is the *Kalman gain matrix*, given by,

$$\mathbf{K} = (\mathbf{G}^T \mathbf{N}^{-1} \mathbf{G} + \mathbf{M}^{-1})^{-1} \mathbf{G}^T \mathbf{N}^{-1}. \tag{12}$$

Again, we have momentarily dropped time from the above equation for notational convenience. The Kalman filter is known as a predictor-corrector filter because estimation is performed in two steps: the prediction step generates the prior estimate, $\hat{\mathbf{s}}^-(t)$, from the process equation (an estimate of equation 8), and then the corrector step corrects this to generate the posterior estimate, $\hat{\mathbf{s}}^+(t)$, using the Kalman filter equation (equation 11). Minimizing the optimization function, J (equation 10), can be shown to be equivalent to maximizing the posterior probability of the state given the observation, i.e. the Kalman filter can be shown to be maximizing

$$P(\mathbf{s}|\mathbf{x}) = P(\mathbf{x}|\mathbf{s})P(\mathbf{s})/P(\mathbf{x}). \tag{13}$$

For details, see [8]. There are also several nonlinear versions of the Kalman filter. In the nonlinear case, the process and observation equations are assumed to be nonlinear. The RecAntSOM is a simple special case of the nonlinear system, where the observation function is the clustering performed by the SOM. The process function (i.e. the state transition probabilities) is estimated by the SRN, and the system directly estimates the posterior probability

given in equation 13. It has been shown that a set of recurrent weights can be obtained for any given set of transition probabilities [9].
The state of the system corresponds to the activations of the SOM nodes. The observation noise is estimated by the logistic function at each node of the SOM. If we view the SOM as a generative model, then the probability of any particular node, N_{ij}, having generated the new observation is, by Bayes' theorem,

$$P(N_{ij}(t)|\mathbf{x}(t)) = \frac{P(\mathbf{x}(t)|N_{ij}(t))P(N_{ij}(t))}{P(\mathbf{x}(t))}. \tag{14}$$

The conditional, $P(\mathbf{x}(t)|N_{ij}(t))$, is estimated by the logistic function at each node. Ignoring the input from the recurrent network is equivalent to ignoring the prior term, $P(N_{ij}(t))$, in the above equation. Thus, the basic SOM can be seen to be performing *maximum likelihood* estimation. By including the input from the recurrent network, we are including an estimate of the prior, and thus doing *maximum a posteriori* estimation. The SRN calculates the prior recursively because it takes as input the posterior estimate from the previous time step, and generates the prior estimate for the next time step.

4 Experiments and Results

For testing the model we tried it on a sound localization from inter-aural intensity difference (IID) task. We have a robotic sensory system called the Illinois Self-Aiming Camera (ISAC) [13], which currently has three kinds of sensors: a video camera, an infra-red camera, and microphones. The cameras are mounted on a pan-tilt unit, which itself is mounted on a box. The microphones are attached to the sides of the box. The task here is to turn the cameras towards a sound source based only on the difference in the intensities of the sounds received at the two microphones. Since the box provides some shielding between the microphones, we can estimate the direction of a sound from the inter-aural intensity difference. However the data tend to be noisy, and some averaging or filtering is required to get reasonable accuracy. In the experiments here, we show that we can improve the localization accuracy by using the RecAntSOM.
The data were collected by placing a sound source at each of five locations one by one, with respect to the box: 0° (far right), 45°, 90° (straight ahead), 135°, and 180° (far left). We used two kinds of sound sources: music from the radio, and a person speaking. Sound was sampled at 44.1KHz, and averaged over 0.1 seconds to generate a single data point from each microphone. Approximately 50s of recording from each direction resulted in 500 data points from each microphone for each direction. These were further averaged using a moving window of length 25 to get a time sequence of 476 data vectors (of length 2) for each direction. The length of the window represents a trade-off between localization accuracy and responsiveness. The longer the window-length, the greater is the localization accuracy, but the longer we have to wait before a change in the location of the sound source becomes apparent. Upon examination, it was found that the data from 0° completely overlapped with the data from other directions, and thus were impossible to differentiate. This is probably due to the structure of the room in which the recordings were done. Thus the experiment below only uses the data from the other four directions.
Data vectors were presented to the RecAntSOM in the following order: 180°, 180°, 135°, 135°,

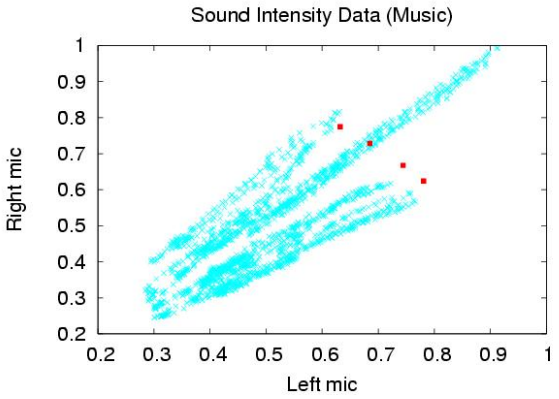

Figure 2: The music data, along with the learned SOM weights.

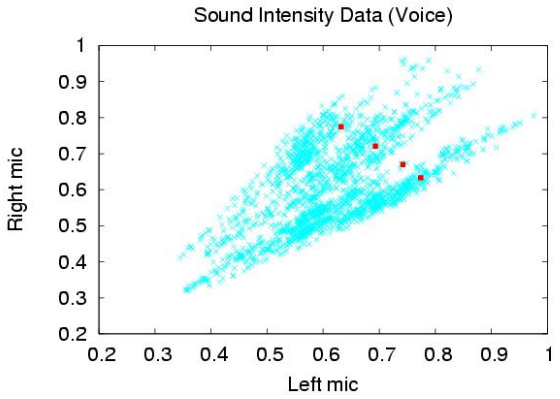

Figure 3: The voice data, along with the learned SOM weights.

90°, 90°, 45°, 45°, 180°, 180°, ..., i.e. two successive inputs from each direction, presented in sequence. This is meant to simulate the situation where there is a steady stream of people walking by slowly from left to right in front of the ISAC. We compared the performance of the RecAntSOM with the AntSOM and a plain SOM, which will ignore the temporal aspect of the data.

Training is done as described in section 2. The learning rate for both the SOM and the SRN was set to 0.03. Since this is a small data set, the algorithm is extremely robust to random initializations, and the results turn out the same every time. We ran the algorithms ten times each, and the standard deviation in the classification error was zero.

The learned SOM weights and the data for both the experiments (music and spoken voice sounds) are shown in figures 2 and 3. A comparison of the performance of the three systems is shown in figures 4 and 5. We see that the RecAntSOM does substantially better than the other methods. It is interesting to note that the AntSOM performs quite poorly on the voice data. This is partly due to the nature of the voice data. It can be seen from figures 2 and 3 that there is much more spread and overlap in the voice data. This is because there tends to be a more or less constant level of background sound in music, whereas there is a great deal of variation in spoken voice. However, the poor performance of the AntSOM is mainly due to the nature of the temporal sequence. Since we present two inputs from the same location in sequence before switching to the next location, an input predicts both the same location and the next location with equal probability. Further, the overlap in the data is between adjacent locations. The AntSOM ends up assigning more or less equal lateral weights (and hence equal priors) from a node to its neighbor, and from a node to itself. This does not help in disambiguating the adjacent locations, and so the AntSOM performs very poorly. The music data are much better separated, and so this drawback of the AntSOM is not as obvious in that case. Since the RecAntSOM actually learns sequences, it learns that there are two inputs from any direction before switching to the next direction, and is able to do a much better job of prediction.

However, a drawback is that the model is not very good at tasks which have long-term dependencies. This is just a limitation of the SRN, however, and, in principle, the SRN

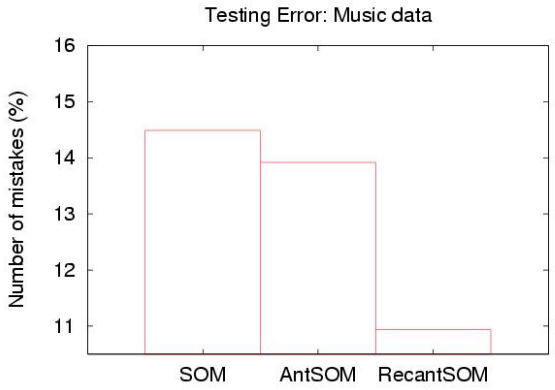

Figure 4: This figure shows a comparison between the three models on the testing set for the music data.

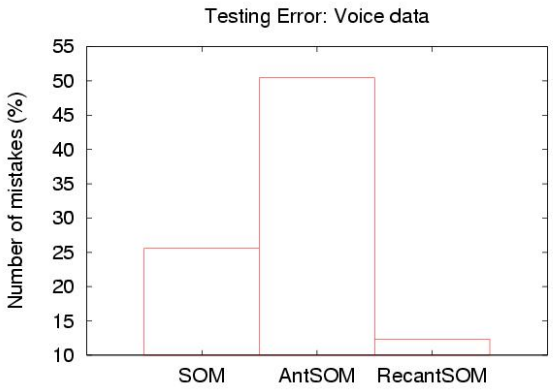

Figure 5: This figure shows a comparison between the three models on the testing set for the voice data.

could be replaced with a better sequence learner to get a better time horizon. Another way to extend it would be to increase the levels of the hierarchy by putting in more SRNs in a hierarchical (layered) manner. This would enable the model to better deal with long term dependencies, as long as there is local temporal structure to the data.

5 Conclusions and Future Work

We have shown how a combination of a self-organizing map and a simple recurrent network can be used as a predictor-corrector filter. This has applications in robotics, and in the design of cognitive architectures, which often use SOMs to cluster the input at the very first stage of processing. It is also relevant to the modeling of biological sensory systems, where the lowest level of processing is known to be map-like. Emerging models of the visual system, e.g., posit that each layer of processing in the visual system is acting as a predictor for the level below it, and a corrector for the level above it. The effect of the prior can also be seen as a temporary widening of the receptive field of the corresponding SOM node, which has been observed in the visual system as well [2].
The main drawback of this system is that it has no way of switching between process models. One possibility is to build a system which has several learned models and also learns to switch between them when the environmental process changes. The ideal situation would be where the system abstracts some aspects of the physical world which help it to learn each new process model quickly. This requires some kind of meta-learning or abstraction process in addition to the filtering process we have described. This is a very interesting area of research, and we are currently engaged in exploring the possibilities in this direction.

References

[1] G. Chappell and J. Taylor. The temporal Kohonen map. *Neural Networks*, 6:441–445, 1993.

[2] A. Das and C. D. Gilbert. Receptive field expansion in adult visual cortex is linked to dynamic changes in strength of cortical connections. *Journal of Neurophysiology*, 74(2):779–792, August 1995.

[3] Guilherme de A. Barreto and Aluizio A. R. Araújo. Unsupervised context-based learning of multiple temporal sequences. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99)*, pages 1102–1106, Washington, DC, 1999.

[4] Peter Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.

[5] Simon Haykin. *Neural Networks: a Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999.

[6] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the* ASME - *Journal of Basic Engineering*, 82 (Series D):35–45, 1960.

[7] Teuvo Kohonen. *Self-Organizing Maps*. Information Sciences. Springer, second edition, 1997.

[8] Rajesh P. N. Rao. An optimal estimation approach to visual perception and learning. *Vision Research*, 39(11):1963–1989, 1999.

[9] Rajesh P. N. Rao. Bayesian computation in recurrent neural circuits. *Neural Computation*, 16:1–38, 2004.

[10] Marc Strickert and Barbara Hammer. Unsupervised recursive sequence processing. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, pages 27–32, Bruges, Belgium, 23-25 April 2003.

[11] Marc Strickert and Barbara Hammer. Self-organizing context learning. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, pages 39–44, 2004.

[12] Samarth Swarup, Kiran Lakkaraju, Alexandre Klementiev, Ernst Sassen, and Sylvian R. Ray. An anticipatory self-organized map for robust recognition. In *Proceedings of the* IASTED *International conference on Artificial Intelligence and Applications (AIA'05)*, Innsbruck, Austria, Feb 14-16 2005.

[13] Samarth Swarup, Tuna Oezer, Sylvian R. Ray, and Thomas J. Anastasio. A self-aiming camera based on neurophysical principles. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'03)*, volume 4, pages 3201–3206, Portland, OR, July 2003.

[14] Markus Varsta, Jukka Heikkonen, and José del R. Millan. Context learning with the self-organized map. In *Proceeding of the Workshop on Self-Organizing Maps*, pages 197–202, Helsinki University of Technology, Espoo, Finland, 1997.

[15] Thomas Voegtlin. Recursive self-organizing maps. *Neural Networks*, 15:979–991, 2002.