Building predictive models on complex symbolic sequences with a second-order recurrent BCM network with lateral inhibition*

Peter Tiňo, Michal Stančík and **Lubica Beňušková**

Department of Computer Science and Engineering
Slovak Technical University, Ilkovičova 3, 81219 Bratislava, Slovakia
E-mail: {tino,benus}@elf.stuba.sk, mstancik@hotmail.com

Abstract. Activation patterns across recurrent units in recurrent neural networks (RNNs) can be thought of as spatial codes of the history of inputs seen so far. When trained on symbolic sequences to perform the next-symbol prediction, RNNs tend to organize their state space so that "close" recurrent activation vectors correspond to histories of symbols yielding similar next-symbol distributions [1]. This leads to simple finite-context predictive models built on top of recurrent activations by grouping close activation patterns via a vector quantization. In this paper we investigate an *unsupervised* alternative to the state space organization. In particular, we use a recurrent version of the Bienenstock, Cooper and Munro (BCM) network with lateral inhibition [2] to map histories of symbols into activations of the recurrent layer. Recurrent BCM networks perform a kind of time-conditional projection pursuit. We compare the finite-context models built on top of BCM recurrent activations with those constructed on top of RNN recurrent activation vectors. As a test bed we use two complex symbolic sequences with rather deep memory structures. Surprisingly, the BCM-based model has a comparable or better performance than its RNN-based counterpart. This can be explained by the familiar information latching problem in recurrent networks when longer time spans are to be latched [3, 4].

1 Introduction

In our recent study of hybrid symbolic-neural time series modeling [1], we trained the second-order RNNs on complex symbolic sequences amd then extracted finite state neural-based predictive models by using only recurrent part of the trained networks and discarding the non-recurrent RNN layers. Loosely speaking, the recurrent part of RNN is used to code histories of seen symbols into vectors in the RNN state space. Using this transformation, simple prediction models can be built by partitioning the RNN state space into a finite set of compartments (representing abstract prediction states) and then approximating the state-conditional RNN next-symbol probabilities by a piece-wise constant functions (constant over the partition compartments). In practical terms, this means driving the trained RNN with fixed weights once more with the training sequence, saving the activation vectors in the recurrent part of the network and then quantizing the recurrent activation stream (e.g. using K-means). We identify abstract prediction states with the resulting quantization centers. Driving the RNN through the training sequence once again, at each step we identify the quantization center (prediction state) to which the current recurrent activation belongs and collect the state-conditional frequences of the next symbols in turn. We found that such finite state neural-based prediction machines often outperform the mother RNN they were extracted from. It is important to note that the RNNs were trained in a *supervised* setting to organize its state space for the next-symbol predictions.

*Accepted to the *IEEE–INNS–ENNS Int. Joint Conference on Neural Networks*, July 24–27 2000, Como, Italy.

In this paper we investigate an *unsupervised* alternative to the state space organization. In particular, we use a recurrent version of the BCM with lateral inhibition [2] to implement the transformation of histories of seen symbols into activations in the recurrent layer. The study was initiated with an intuition that, due to the familiar information latching problem [4], the corresponding transformations in "classical" RNN may not be optimal for sequences exhibiting a deep memory structure. Non-recurrent BCM networks with the so-called feedforward lateral inhibition are known to perform projection pursuit of the input space into the space of BCM activations [5, 6]. The (extended) input to recurrent BCM network consists of the code of the current input symbol and the current network state, which is the activation vector of BCM neurons from the previous time step (Fig. 1a). Recurrent BCM networks (BCM RNN) then perform a *time-conditional projection pursuit*, i.e. organize the state space so that the extended input vectors project into the next states unveiling the most prominent departures from the Gaussian distribution in the extented input. After training with real time recurrent learning (RTRL), we treat BCM RNN in the same way as recurrent parts of RNN in our previous work. We extract finite state prediction machine through vector quantization of the BCM recurrent layer activations.

We test the RNN- and BCM-based predictive models on two complex symbolic sequences, namely, on the sequence of quantized activity changes of a chaotic laser, and on the binary Feigenbaum sequence known for its deep memory structure [3]. The models are compared using normalized negative log-likelihood on unseen continuations of the sequences.

2 Second-order BCM RNN with lateral inhibition

The output of the ith neuron in the 2nd-order BCM RNN comprised of n neurons with lateral inhibition of the constant strength μ between each pair of neurons (Fig. 1a) is [2]:

$$c_i(t+1) = \sigma\Big\{\sum_{j,k}\Big[w_{ijk}(t) - \mu\sum_{\beta\neq i} w_{\beta jk}(t)\Big]d_j(t)c_k(t)\Big\} \tag{1}$$

where σ denotes the sigmoid activation function, i.e. $\sigma(x) = 1/(1+exp(-\lambda x))$ with the slope λ. The input symbol at time t is coded by the binary input vector $\mathbf{d}$(t)=$\{d_j(t)\}$. The lateral inhibition of the strength μ is of the so-called feedforward type [5]. Synaptic weights w_{ijk} are updated at each time step t, i.e. after each presentation of a consecutive symbol in the temporal sequence. The weight changes are derived by means of the gradient descent minimization of the total loss function $L(t) = \sum L_i(t)$ [5], where

$$L_i(t) = -\Big\{\frac{1}{3}c_i^3(t) - \frac{1}{4}E[c_i^2(t)]\ c_i^2(t)\Big\} \tag{2}$$

where $E[c_i^2(t)] = \theta_M^i(t)$ is the moving synaptic modification threshold of the ith neuron calculated as [7]

$$\theta_M^i(t) = \frac{1}{\tau}\int_{-\infty}^{t} c_i^2(t')e^{-\frac{t-t'}{\tau}}dt' \tag{3}$$

where τ is the averaging period. Relation between $\theta_M^i(t)$ and the average square of the ith neuron activity guarantees three important features of the model: (i) $\theta_M^i(t)$ is always ≥ 0, (ii) sliding of $\theta_M^i(t)$ guarantees the upper boundedness of synaptic weights without placing artificial constraints on them, and (iii) if all the synaptic weights approached zero, $\theta_M^i(t)$ must also eventually approach zero, thereby allowing for the possible recovery (potentiation) of synaptic weights. Thus

$$\frac{dw_{ijk}(t+1)}{dt} = -\eta\frac{\partial L(t+1)}{\partial w_{ijk}(t)} = \eta\Big[\sum_{\alpha}\phi_\alpha(t+1)\frac{\partial c_\alpha(t+1)}{\partial w_{ijk}(t)}\Big] \tag{4}$$

where η is the learning speed. $\phi_\alpha(t) = c_\alpha(t)[c_\alpha(t) - \theta_M^\alpha(t)]$ is the synaptic modification function for the αth neuron. When the neuronal activity $0 < c_\alpha(t) < \theta_M^\alpha(t)$, all its active synapses weaken. On

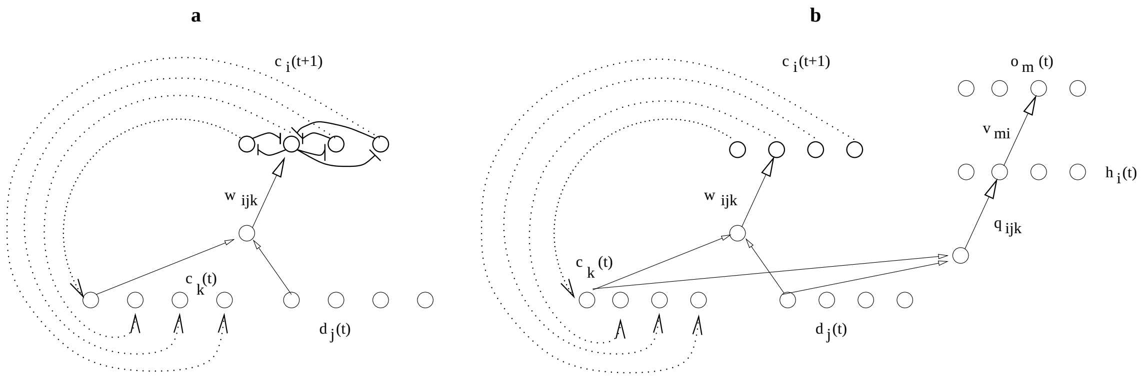

Figure 1: (a) A 2nd-order recurrent BCM neural network with lateral inhibition. (b) A "classical" 2nd-order RNN for processing time series.

the other hand, when $c_\alpha(t) > \theta_M^\alpha(t)$, all its active synapses potentiate. As for the partial derivative of the neuronal activity according to the weight, we get [2]

$$\frac{\partial c_\alpha(t+1)}{\partial w_{ijk}(t)} = \sigma'(t+1)\Big\{[\delta_{\alpha i}^{Kron.} - (1-\delta_{\alpha i}^{Kron.})\mu]\ d_j(t)c_k(t) + \sum_{mp}\Big[w_{\alpha mp}(t) - \mu\sum_{\beta\neq\alpha} w_{\beta mp}(t)\Big]\ d_m(t)\frac{\partial c_p(t)}{\partial w_{ijk}(t-1)}\Big\} \tag{5}$$

3 "Classical" second-order RNN

The output of the mth neuron in the 2nd-order RNN (Fig. 1b) with synaptic weight matrices between layers $\mathbf{v}$, $\mathbf{q}$ and $\mathbf{w}$, is [1]:

$$o_m(t) = \sigma\Big(\sum_k v_{mi}(t)h_i(t)\Big) \tag{6}$$

where the activations $h_i(t)$ of hidden non-recurrent neurons are determined by

$$h_i(t) = \sigma\Big(\sum_{j,k} q_{ijk}(t)d_j(t)c_k(t)\Big) \tag{7}$$

Here, the input symbol at time t is coded by the binary input vector $\mathbf{d}(t) = \{d_j(t)\}$. The activations of recurrent neurons at the next time step, $c_i(t+1)$, are computed as follows:

$$c_i(t+1) = \sigma\Big(\sum_{j,k} w_{ijk}(t)d_j(t)c_k(t)\Big) \tag{8}$$

At each time step t, the RNN is trained to predict the next symbol in the sequence. Training is performed via the RTRL-type minimization of the error function [8]:

$$E(t) = \frac{1}{2}\sum_m (r_m(t) - o_m(t))^2 \tag{9}$$

where the desired (required) output vector $\mathbf{r}(t)=\{r_m(t)\}$ is the binary code of the next symbol in the training sequence.

4 Building predictive models

First we trained both BCM RNN and RNN on a long sequence of quantized activity changes of a real laser in a chaotic regime. The chaotic laser produces periods of oscillations with increasing amplitude, followed by sudden, difficult to predict, activity collapses. Thus, relatively predictable subsequences are followed by highly unpredictable events requiring a deeper memory. The laser sequence (obtained from `http://www.cs.colorado.edu/~andreas/Time-Series/SantaFe.html`) of 10^4 differences Δ_t between successive activations was quantized into a symbolic sequence $S = \{s_t\}$ over the alphabet $\mathcal{A} = \{a, b, c, d\}$, with $A = 4$ symbols. The symbols correspond to low and high positive/negative activity changes such that

$$s_t \equiv \begin{cases} a \ (normal\ up), & \text{if } 0 \le \Delta_t < \rho_2 \\ b \ (high\ up), & \text{if } \rho_2 \le \Delta_t \\ c \ (normal\ down), & \text{if } \rho_1 \le \Delta_t < 0 \\ d \ (deep\ down), & \text{if } \Delta_t < \rho_1 \end{cases} \tag{10}$$

where the parameters ρ_1 and ρ_2 correspond to 10% and 90% sample quantiles, respectively. The number of positive Δ_ts is approximately equal to the number of negative Δ_ts. The first 8000 symbols and the remaining 2000 symbols formed the training and test sequences, S_{train} and S_{test}, respectively.

Second, we applied both networks to the Feigenbaum binary ($A = 2$) sequence over the alphabet $\mathcal{A} = \{a, b\}$ [9]. It can be described by the following iterative composition rule for construction of subsequences w_i:

$$w_0 = b, \quad w_1 = ba, \quad w_{n+1} = w_n w_{n-1} w_{n-1}, \quad n \ge 2 \tag{11}$$

Very specialized prediction contexts are needed to model this sequence. We created a large portion of the Feigenbaum sequence and trained both RNNs on the first 260 000 symbols. The following 200 000 symbols formed S_{test}.

Now we desribe construction of predictive models for both types of RNNs. After training, all synaptic weights are fixed and stored. Next, for each symbol in S_{train}, using these weights, we record the activations $\mathbf{c}(t) = \{c_i(t)\}$. This creates an activation sequence S^{act}_{train} for each network. Then we perform a vector quantization on S^{act}_{train} by means of the K-means clustering for N quantization centers. Quantization centers are identified with predictive states in our predictive models. To calculate the state-conditional probabilities, we associate with each center A counters, one for each symbol from the input alphabet $\mathcal{A}$. Moving through S^{act}_{train}, the closest center C is found for each recurrent activation vector. Next, we identify the next symbol s_{next} in S_{train}. For the given center C, the counter associated with s_{next} is raised by one. After seeing the whole training sequence S_{train}, for each quantization center (prediction state) C, we normalize the counters to calculate the conditional next-symbol probabilities $P_{\mathcal{M}}(s|C)$. Each center now represents a predictive state.

Having observed a sequence $S_{test} = s_1 s_2 ... s_t ... s_m$, the next-symbol probabilities are determined as follows: each $s_t \in S_{test}$ drives the RNN to recurrent activation vector $\mathbf{c}(t) = \{c_i(t)\}$. Find the quantization center C_t closest to $\mathbf{c}(t)$. The next symbol probabilities are $P_{\mathcal{M}}(s_{t+1}|C_t)$.

For each model $\mathcal{M}$, we evaluated its performance by means of the normalized negative log-likelihood NNL on the test sequence S_{test} such that

$$NNL_{\mathcal{M}}(S_{test}) = \frac{-\sum_{t=1}^{m} log_A P_{\mathcal{M}}(s_{t+1}|C_t)}{m} \tag{12}$$

where the base of the logarithm is the number of symbols A in the alphabet $\mathcal{A}$. The $NNL_{\mathcal{M}}(S_{test})$ measures the total amount of "statistical surprise" induced by the model [10].

As the distance function d, we always considered the Euclidean distance.

5 Results

The code vectors for the laser symbols were $\mathbf{d}_a = (1,0,0,0)$, $\mathbf{d}_b = (0,1,0,0)$, etc., and for the Feigenbaum binary symbols, they were $\mathbf{d}_a = (1,0)$, $\mathbf{d}_b = (0,1)$. NNLs were calculated for different numbers of

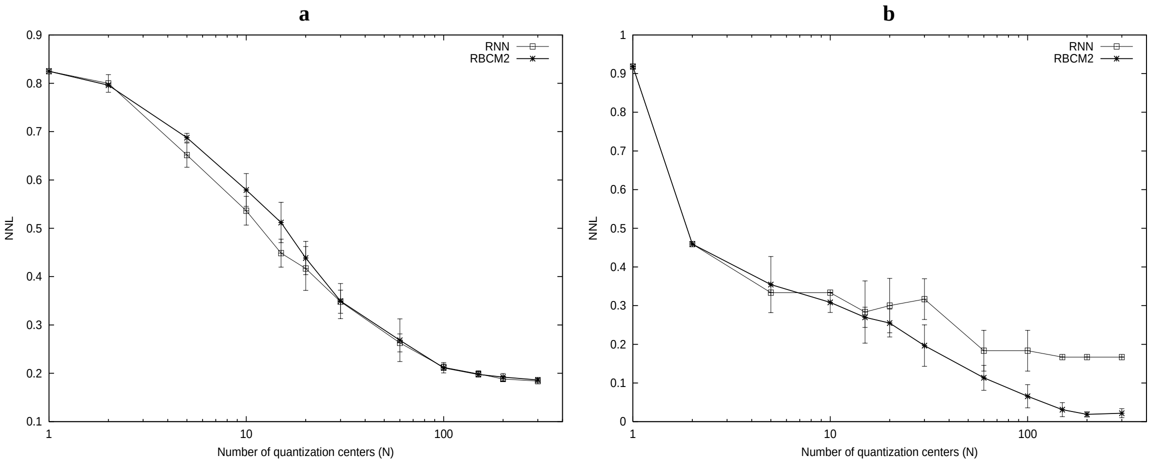

Figure 2: Normalized negative log-likelihoods (NNL) for the (a) laser test sequence, (b) Feigenbaum test sequence. Results of the predictive model built upon the 2nd-order BCM RNN with lateral inhibition are indicated by RBCM2. RNN denotes results of the classical 2nd-order RNN.

predictive centers, i.e. $N = 1, 2, 5, 10, 15, 20, 30, 60, 100, 150, 200, 300$.

The values of parameters of the 2nd-order BCM RNN were set to: number of (recurrent) neurons $n = 4$, $\tau = 100$ iterations, $\eta = 0.01$, $\mu = 0.2$, $\lambda = 0.5$ for the unipolar '0–1' sigmoid. The results were not much different for $\tau = 10$–10^3 iterations, $\mu = 0.1$–0.3, and $\lambda = 0.3$–0.8. For these ranges of parameter values the results deviated no more than they did in the individual runs for a single set of parameter values. The learning was stopped after the weights have not been changing noticably for quite a while ($\sim 10^3$ iterations). This usually happened after 5 epochs of training.

The values of parameters of the 2nd-order RNN were set to: $n = 4$ neurons for all the layers, $\eta = 0.15$, momentum $= 0.08$, and $\lambda = 1$ for the unipolar '0–1' sigmoid. The initial (reset) activations of recurrent neurons were randomly generated from a uniform distribution over [-0.5, 0.5]. Thirty epochs were used to train the RNN on the laser sequence and five epochs for the Feigenbaum sequence.

For both networks, the initial weights were uniformly generated from the interval of [-1, 1].

In Fig. 2, we present NNLs for the laser and Feigenbaum test sequences. All presented values are the averages over 10 vector quantization runs in our predictive model construction (see section 4). Also shown are the corresponding standard deviations.

6 Discussion

The main theme of this paper is related to representation of symbolic sequences by points in a vector space (see [11]). Association map between the recurrent and output layers in RNN is usually smooth and so by training RNN to perform the next-symbol prediction, the network is forced to map sequences of observed symbols close to each other if they lead to similar next-symbol distributions. This intuition can be used to extract simple finite state predictive models via detection of activation clusters in the RNN recurrent layer –"close" activation vectors correspond to subsequences yielding similar prediction probabilities. In other words, the activation clusters correspond to "abstract prediction states" found by RNN. Actually, the extracted models outperform their mother RNN (see [1]).

We were interested whether an *unsupervised* construction of a mapping from symbolic sequences to points in a vector space can lead to comparable predictive models. In this case, there is no explicit driving force in the form of the target next symbols. We chose to work with a recurrent form of a biologically motivated 2nd-order BCM network that performs a time-conditional projection pursuit. The input symbols are coded using the one-of-n encoding scheme, so there is no a-priori metric structure imposed on the input alphabet. The BCM RNN tries to project the current states (code of the symbols it has observed up to time step $t-1$) extended with the code of the current input symbol (at time t), so that the projections (states at time $t+1$) are "non-trivially" spread out in the recurrent activation space. Here, non-triviality means non-gaussianity.

Our results are quite surprising. On two chaotic sequences with relatively deep memory structures, the supervised and unsupervised approaches yield almost the same predictive model performances. On the Feigenbaum sequence, the performance of the BCM RNN-based model can be even better than that of the RNN-based one. We got similar results with other forms of spatial representations of temporal structures in symbolic sequences (see [3]). One of the reasons may be that even though the "classical" RNN are potentially very powerful in building state space representations suited to the specific task (e.g. next-symbol prediction), in practice it is difficult to train them to span large time intervals due to the information latching problem [4].

To conclude, recurrent BCM networks operating in an unsupervised mode are able to form state space representations of the training sequence that yield comparable (if not better) predictive models than those found in the supervised mode in classical RNNs supplied with teacher training signals.

Acknowledgments: Supported by the VEGA grants 2/6018/99 and 1/7611/20.

References

[1] Tiňo P. and Köteleš M. (1999) Extracting finite state representations from recurrent neural networks trained on chaotic symbolic sequences. *IEEE Transactions on Neural Networks* **10**, 284–302.

[2] Bachman C.M., Musman S.A., Luong D. and Shultz A. (1994) Unsupervised BCM projection pursuit algorithms for classification of simulated radar presentations. *Neural Networks* **7**, 709–728.

[3] Tiňo P. and Dorffner G. (1999) Building predictive models from spatial representations of symbolic sequences. (Will appear in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, K-R. Müller (eds.), MIT Press, Cambridge, 2000.)

[4] Bengio Y, Simard P. and Frasconi P. (1994) Learning long-term dependencies with gradient is difficult. *IEEE Transactions on Neural Networks* **5**, 157–166.

[5] Intrator N. and Cooper L.N. (1992) Objective function formulation of the BCM theory of visual cortical plasticity: statistical connections, stability conditions. *Neural Networks* **5**, 3–17.

[6] Dotan Y. and Intrator N. (1998) Multimodality exploration by an unsupervised projection pursuit neural network. *IEEE Transactions on Neural Networks* **9**, 464–472.

[7] Beňušková L., Diamond M.E. and Ebner F.F. (1994) Dynamic synaptic modification threshold: computational model of experience-dependent plasticity in adult rat barrel cortex. *Proc. Natl. Acad. Sci. USA* **91**, 4791–4795.

[8] Williams R.J. and Zipser D. (1989) A learning algorithm for continually running fully recurrent neural networks. *Neural Computation* **1**, 270–280.

[9] Freund J., Ebeling W. and Rateitschak K. (1996) Self-similar sequences and universal scaling of dynamical entropies. *Physical Review E* **5**, 5561–5566.

[10] Ron D., Singer E. and Tishby N. (1996) The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning* **25**, 117–149.

[11] Tiňo P. (1999) Spatial representation of symbolic sequences through iterative function systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans* **29**, pp. 386-392.