Finite-State Reber Automaton and the Recurrent Neural Networks Trained in Supervised and Unsupervised Manner

Michal Čerňanský and Lubica Beňušková

Department of Computer Science and Engineering, FEI Slovak Technical University,
Ilkovičova 3, 812 19 Bratislava 1, Slovakia
{cernansky,benus}@dcs.elf.stuba.sk, http://www.dcs.elf.stuba.sk/∼benus

Abstract. We investigate the evolution of performance of finite-context predictive models built upon the recurrent activations of the two types of recurrent neural networks (RNNs), which are trained on strings generated according to the Reber grammar. The first type is a 2nd-order version of the Elman simple RNN trained to perform the next-symbol prediction in a supervised manner. The second RNN is an interesting unsupervised alternative, e.g. the 2nd-order RNN trained by the Bienenstock, Cooper and Munro (BCM) rule [3]. The BCM learning rule seems to fail to organize the RNN state space so as to represent the states of the Reber automaton. However, both RNNs behave as nonlinear iteration function systems (IFSs) and for a large enough number of quantization centers, they give an optimal prediction performance.

1 Introduction

State-vector clustering serves as the standard mechanism for the extraction of finite-state automata from RNNs [4] as well as for the extraction of predictive models in case of complex symbolic sequences [8]. It was shown, that when trained via the real time recurrent learning (RTRL), RNNs organize their state space so that close recurrent activation vectors correspond to histories of symbols yielding similar next-symbol distributions. Incidentally, by means of the theory of IFSs, Kolen [6] demonstrated that randomly initialized RNNs display a surprising amount of clustering before training .

We investigate the 2nd-order Elman RNN trained to perform the next-symbol prediction and the 2nd-order RNN trained by an unsupervised BCM rule (BCM RNN) [1]. Recently, we found that the latter type of RNN gives comparable predictive performance on the chaotic time series as the former one [9]. This time, both RNNs are trained on strings generated according to the Reber grammar (Fig. 1a). It was already shown that the Elman SRN can learn the Reber automaton perfectly [4]. In this work, we observe the evolution of performance of predictive models built upon the activation of the recurrent layer in both types of RNNs during training as a function of the amount of training and the number of quantization centers in the recurrent state space.

G. Dorffner, H. Bischof, and K. Hornik (Eds.): ICANN 2001, LNCS 2130, pp. 737–742, 2001.

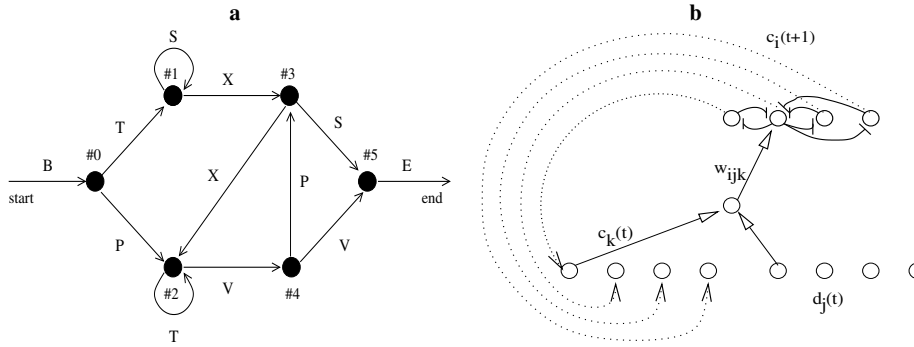

Fig. 1. (a) The Reber automaton. We start with B, and move from one node to the next. If there are two paths we can take, e.g. after B we can take either T or P, we choose one with equal probability, and so on for every node, except for the end node #5. All the Reber states are uniquely determined by each pair of symbols in the Reber string. **(b)** The 2nd-order recurrent BCM neural network with lateral inhibition

2 The 2nd-Order BCM RNN with Lateral Inhibition

First, let us consider one isolated BCM neuron, i, with the output activity $c_i(t) = \sigma(\sum_j w_{ij}(t)d_j(t))$, where σ denotes the sigmoid activation function and $d_j(t)$ is the jth input component. Then, according to the BCM theory [3], the jth synaptic weight changes as

$$\frac{dw_{ij}(t)}{dt} = -\eta \frac{\partial L_i(t)}{\partial w_{ij}(t)} = \eta \phi_i(t) d_j(t) \ , \tag{1}$$

where η is the learning speed and $\phi_i(t) = c_i(t)[c_i(t) - \theta^i(t)]$ is the synaptic modification function [2]. When the neuronal activity $c_i(t) > \theta^i(t)$, all active synapses potentiate (provided $d_j > 0$). On the other hand, when $0 < c_i(t) < \theta^i(t)$, all active synapses weaken. The variable $\theta^i(t)$ is the moving synaptic modification threshold defined as in neuro-computational models [2]

$$\theta^i(t) = E[c_i^2(t)] = \frac{1}{\tau} \int_{-\infty}^{t} c_i^2(t') e^{-\frac{t-t'}{\tau}} dt' \ , \tag{2}$$

where τ is the averaging period. Sliding of θ^i guarantees the upper boundedness of synaptic weights without placing artificial constraints on them. The quantity $L_i(t)$ is the loss function to be minimized [1], i.e.

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

However, the output of the ith neuron in the 2nd-order BCM RNN with the so-called feedforward lateral inhibition of constant strength μ between each pair of neurons (Fig. 1b) is [1]

$$c_i(t+1) = \sigma\Big\{ \sum_{j,k} \Big[w_{ijk}(t) - \mu \sum_{\beta \neq i} w_{\beta jk}(t) \Big] d_j(t) c_k(t) \Big\} \ . \tag{4}$$

Synaptic weights w_{ijk} are updated at each time step t, i.e. after each presentation of a consecutive symbol in the temporal sequence. Weight changes are derived by means of the gradient descent minimization of the total loss function $L(t) = \sum L_i(t)$ [1] yielding a more complicated rule than (1), namely

$$\frac{dw_{ijk}(t+1)}{dt} = -\eta \frac{\partial L(t+1)}{\partial w_{ijk}(t)} = \eta \Big[\sum_{\alpha} \phi_\alpha(t+1) \frac{\partial c_\alpha(t+1)}{\partial w_{ijk}(t)} \Big] \ . \tag{5}$$

3 Normalized Negative Log-Likelihood

We evaluate the performance or quality of predictive models $\mathcal{M}$s extracted from the RNNs by means of normalized negative log-likelihood [7]. The predictive models based on internal state vectors are constructed as follows. After each training epoch, all synaptic weights are fixed and stored. Next, sliding through the training sequence, for each symbol in S_{train}, using these weights, we record the recurrent activations $\mathbf{c}(\text{t+1}) = \{c_i(t+1)\}$. This creates an activation sequence $S^{\text{act}}_{\text{train}}$ for each network. Then we perform a vector quantization on $S^{\text{act}}_{\text{train}}$ for N quantization centers. In our predictive models, the quantization centers are identified with predictive states. To calculate the state-conditional probabilities, we associate with each center A counters, one for each symbol from the input alphabet $\mathcal{A}$. Sliding through the recurrent activations' sequence, $S^{\text{act}}_{\text{train}}$, the closest center C is found for the current activation vector. Next, we identify the next symbol s_{next} in S_{train}. For the 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 $P_{\mathcal{M}}(\cdot|C)$.

On the test sequence $S_{\text{test}} = s_1 s_2 ... s_t ... s_m$, the next-symbol probabilities are determined as follows: for each $t = 1, 2, ..., m-1$, given the network state $\mathbf{c}(t) = \{c_i(t)\}$, the symbol $s_t \in S_{\text{test}}$ drives the network to a new recurrent activation vector $\mathbf{c}(\text{t+1}) = \{c_i(t+1)\}$. We find the quantization center C_{t+1} closest to $\mathbf{c}(\text{t+1})$. The next-symbol probabilities are $P_{\mathcal{M}}(s_{t+1}|C_{t+1})$.

For each model $\mathcal{M}$, we evaluated its performance by means of the normalized negative log-likelihood NNL [7] on the test sequence S_{test}:

$$\text{NNL}_{\mathcal{M}}(S_{\text{test}}) = \frac{-\sum_{t=1}^{m-1} \log_A P_{\mathcal{M}}(s_{t+1}|C_t)}{m-1} \ , \tag{5}$$

where the base of the logarithm is the number of symbols A in the alphabet $\mathcal{A}$. The higher are the next correct-symbol probabilities the smaller is NNL, with NNL $= 0$ corresponding to the 100% correct next-symbol prediction.

The metric in vector quantization and nearest-center detection is Euclidean. We use the hierarchical vector quantization inspired by [5]. The first activation vector becomes the first quantization center. Given a certain number of centers, we iteratively find the corresponding maximal cluster radius γ. For every internal state vector in turn, we find the closest center. If their distance is less than γ, the vector belongs to the center. Otherwise, this vector becomes a new center.

4 Experiments and Results

We trained the networks on randomly generated strings, starting with the "B". The activations of recurrent neurons were reset to the same small random values at the beginning of each string. Before training and after each 10×10^3 symbols we evaluated the quality of the next-symbol prediction by means of NNL. For the NNL calculation we used the test set of the length 20×10^3 symbols consisting of newly generated strings. The input of both networks and output of the Elman RNN had the dimension of 6 with the one-hot encoding of symbols. "E" is equally coded as "B". The values of parameters of the 2nd-order BCM RNN were set to: the number of (recurrent) neurons $n = 8$ or 12, $\tau = 100$ iterations, $\eta = 0.001$, $\mu = 1/n$, $\lambda = 0.4$ for the unipolar '0-1' sigmoid. The values of parameters of the 2nd-order RNN were set to: $n = 6$ or 8, $\eta = 0.03$, momentum $= 0.03$, and $\lambda = 1$ for the unipolar '0-1' sigmoid. The initial (reset) activations of recurrent neurons and initial weights were randomly generated from a uniform distribution over $[-0.5, 0.5]$, for both RNNs.

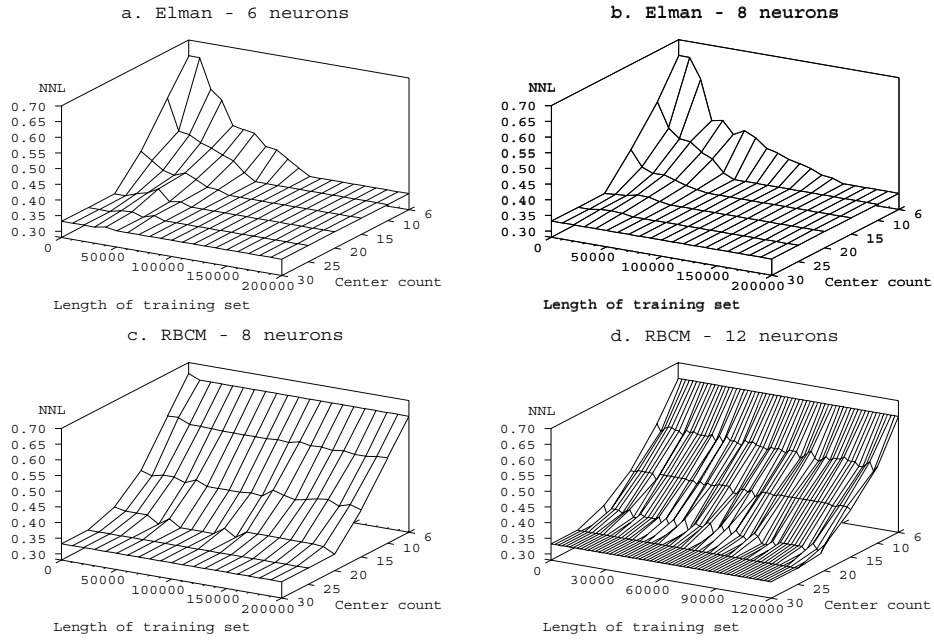

Fig. 2. NNL results for the next-symbol prediction of the Elman RNN for **(a)** 6 and **(b)** 8 recurrent neurons, and of the BCM RNN for **(c)** 8 and **(d)** 12 recurrent neurons. NNL at time 0 expresses the prediction performance of an untrained "naïve" network. Length of training set means the total number of symbols over all training strings. Center count means the number of quantization centers in the hierarchical vector quantization. The lowest NNL matches the theoretically calculated NNL for the occurence of symbols in Reber strings, e.g. 0.331.

In Fig. 2, we present NNLs of the next-symbol prediction on the test Reber strings for both types of RNNs, as they evolve during training for different numbers of quantization centers. All presented values are averages over 5 different runs. The standard deviations (not shown) may reach up to 20% of the corresponding means.

5 Discussion

One can show that the Elman RNN behaves as a kind of nonlinear IFS consisting of a sequence of transformations that map the state space represented by activations of recurrent neurons into separate subspaces of the state space [6]. Each next state of an RNN, represented by the recurrent activations, is mostly determined by the last performed transformation, e.g. by the last presented symbol (input). Within the subspace belonging to the last transformation, the network state is determined by the last previous transformation, and thus by the last previous symbol (input). In this way, the RNN "theoretically" codes an infinite time window to the past, e.g. its current state uniquelly represents the history of inputs. The more distant the input is in the past the less it determines the current RNN state. It turns out, that two subsequences with the common suffix will correspond to the two states that are close to each other in the state space. The longer the common suffix the smaller the distance between the corresponding states in the state space.

All the Reber states are uniquely determined by each pair of symbols in the Reber string. There are 20 of these pairs including the "SE" and "VE" pairs. Thus, even in the "naïve" untrained RNN, the 20 quantization centers obtained by means of the hierarchical clusterization described above, correspond to the RNN states determined by the last 2 symbols. It means that there is a unique mapping between the automaton states and the RNN states and the prediction model gives the minimal possible NNL, in this case 0.331 (Fig. 2). In an untrained case, the individual states (recurrent activities) evoked by the last two symbols are well separated in the state space. Given more than 20 (e.g. 25 or 30) quantization centers, these recurrent activities can become split, due to the third last input. Therefore, the quality of the next-symbol prediction does not get worse.

The Elman RNN is trained by RTRL, thus the weights are updated after each symbol in turn. The errors that backpropagate from the output layer cause the weights to modify in such a way that the recurrent activities which ought to belong to the same automaton state get closer to each other step by step. They get closer, thus reflecting the probability that the continuation of the sequence will be the same. This movement may cause a temporary disturbance of the coding based on the last pair of symbols as we can observe for the large number of quantization centers (Fig. 2a, b).

After the training, the network state reflects not only the history of inputs but, which is perhaps more important, also the probability of a certain continuation. Each network state belongs to a certain automaton state. It can be shown

that the optimal prediction model can be created with only 6 clusters in the state space (see also [4]).

The BCM RNN behaves also like a kind of IFS, albeit different from the previous one because of the lateral inhibition. In this case, we can also obtain the optimal prediction with a large number of quantization centers (e.g. $\geq$25) for the predictive model built upon the "naïve" untrained network (Fig. 2c, d). Then the weights are modified after each presentation of input according to the BCM rules for the recurrent network [1]. The recurent neurons become sensitive to particular statistical characteristics of the input set (see e.g. eq. 2), and they become selective only to individual last symbols and thus we are not able to observe the improvement of prediction for the number of quantization centers smaller than 20.

Acknowledgments

Supported by the VEGA grants 2/6018/99 and 1/7611/20. We thank Peter Tiňo and anonymous reviewers for their valuable comments.

References

1. Bachman, C.M., Musman, S.A., Luong, D., Shultz, A.: Unsupervised BCM projection pursuit algorithms for classification of simulated radar presentations. Neural Networks **7** (1994) 709–728
2. Beňušková, L., Diamond, M.E., Ebner F.F.: Dynamic synaptic modification threshold: computational model of experience-dependent plasticity in adult rat barrel cortex. Proc. Natl. Acad. Sci. USA **91** (1994) 4791–4795
3. Bienenstock, E.L., Cooper, L.N, Munro, P.W.: Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. J. Neurosci. **2** (1982) 32–48
4. Cleeremans, A., Servan-Schreiber, D., McClelland, J.L.: Finite state automata and simple recurrent networks. Neural Comp. **1** (1989) 372–381
5. Das, S., Das, R.: Induction of discrete-state machine by stabilizing a simple recurrent network using clustering. Comp. Sci. Inf. **21** (1991) 35–40
6. Kolen, J.F.: The origin of clusters in recurrent neural network space. In: The Proceedings of the 16th Annual Conference of the Cognitive Science Society, Atlanta, GA, August 13-16, 1994
7. Ron, D., Singer, E., Tishby, N.: The power of amnesia: learning probabilistic automata with variable memory length. Machine Learning **25** (1996) 117–149.
8. Tiňo, P., Köteleš, M.: Extracting finite state representations from recurrent neural networks trained on chaotic symbolic sequences. IEEE Trans. Neural Net. **10** (1999) 284–302
9. Tiňo, P., Stančík, M., Beňušková, L.: Building predictive models on complex symbolic sequences with a second-order recurrent BCM network with lateral inhibition. Proc. Intl. Join Conf. Neural Net. vol. 2 (2000) 265–270