

Abstract

It has been shown [1,2] that feed-forward random Boolean networks (RBNs) can learn to perform specific simple tasks and generalize well if only a subset of the learning examples is provided for learning. Here, we extend this body of work and show experimentally that recurrent random Boolean networks, where both the interconnections and the Boolean transfer functions are chosen at random initially, are able to learn and to generalize. We use both evolutionary algorithms (EAs) and simulated annealing (SA) to train the networks to solve simple tasks. We compare the learning and generalization performance by varying:

- the number of inputs,
- the number of incoming connections k per node, and
- by allowing the learning algorithm to adapt:
 - the connections only or
 - both the connections and the node transfer functions.

Our results suggest that recurrent random Boolean networks are well able to generalize and to uncover the underlying general rules of the tasks.

Random Boolean Networks (RBN)

RBNs are discrete dynamical systems originally introduced by Stuart Kauffman [3] as a model for genetic regulatory networks. They are directed graphs composed of:

- a set of N nodes with a binary internal state (Boolean variable)
- a set of random interconnections among the N nodes.

Each node i receives inputs (either average or exact) from K randomly (with uniform probability) chosen nodes. K is called the in-degree. Each node contains a Boolean (transfer) function with K inputs that can be represented as a truth table (or look-up table, LUT). Initially, the output for each of the 2^k possible inputs per node is chosen at random (unbiased).

Dynamics

RBNs exhibit two different phases [3], depending on a critical value K_c : $K < K_c$ is ordered or frozen ($K < K_c$) and disordered or chaotic ($K > K_c$). $K_c \approx 2$ (the "edge of chaos") has been obtained both numerically and analytically [4]. At the critical point, a small number of stable attractors ($-O(\sqrt{N})$) is observed.

Adding inputs and outputs

For our purpose, we extend the standard NK model with dedicated input and output nodes (see Fig. 1). Output nodes are considered as normal network nodes. The input nodes cannot receive incoming connections from the other network nodes. Their sole purpose is to store the input values.

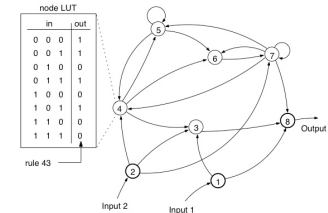


Figure 1: A $N=8, k=3$ RBN with 2 dedicated input and 1 output node. The look-up table (LUT) shows a possible rule for a given 3-input node. To identify the rule, the out-column can be interpreted as a decimal value, i.e. rule 43 (top bit = least significant bit).

Signal interpretation

We use the following procedure to interface the network.

1. Reset node states to all 0 or to a random state.
2. Apply input signal to input nodes and keep state constant.
3. Synchronously update all nodes T times, where T is odd.
4. Record the output node states during the T update steps.
5. If more 1s than 0s, then output interpreted as 1, otherwise 0.

The results are very robust for e.g. $T = 15$ for smaller networks ($N < 30$) and for k up to ~ 5 .

Previous and Related Work

It has been shown [1,2] that feed-forward random Boolean networks (RBNs) can learn to perform specific simple tasks and generalize well if only a subset of the learning examples is provided for learning.

Fig. 2 shows the results for the even-odd task (see next box) obtained by a mean-field approximation [1]. The results were experimentally confirmed by a 25-node network and by changing both the connections and the functions by a simulated annealing procedure. As one can see, the more complex the task (i.e., more inputs), the better the generalization.)

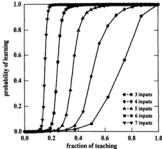


Figure 2: Mean-field results for the probability of learning as a function of the fraction of teaching. From [1].

Tasks

We use three benchmark toy-tasks of varying difficulty for evaluating the learning performance of our RBNs:

#1-Mapping Task

- Same number of inputs and outputs, $I=O$
- Number of 1s in input = number of 1s in the output



Density Task

- J inputs (odd number only), 1 output
- If $> 50\%$ of 1s in input, then output = 1, otherwise 0



Even-Odd Task

- J inputs, 1 output
- If odd number of 1s in input, then output = 1, otherwise 0



Learning Methods

The task of learning a function F by a network G can be seen as a global optimization problem. Here, we use two approaches:

Simulated Annealing (SA)

The energy E represents the difference between the correct and the result obtained by the network, averaged over the training set N_E :

$$E(G, F) \equiv \sum_{i=1}^{N_E} E_i \equiv \sum_{i=1}^{N_E} \frac{1}{N_E} \sum_{k=1}^{N_E} (E_{ik} - A_{ik})^2 \quad \text{Eq. 1}$$

Here, E_{ik} is the exact result from the i^{th} bit in the k^{th} example, A_{ik} is the network output for the same bit and example, N_E is the number of output bits. The Monte Carlo procedure is as following:

1. Calculate E using Eq. 1
2. Rewire the incoming link of a randomly chosen node.
3. Calculate the new energy E_{new} using Eq. 1
4. Accept the move with Boltzmann probability $\exp(-\Delta E/T)$, where T is a control parameter, and $\Delta E = E_{\text{new}} - E$.
5. Repeat steps 1-4 n times or until $E = 0$ (ground state)
6. After n steps, if ground state is not found, decrease T by ΔT and repeat steps 1-5.

Evolutionary Algorithm (EA)

We employ a standard EA for artificial neural nets [5] to find (1) good connections (topology), (2) good functions, or (3) both for a network G in order to perform a function F . The network is directly encoded (integer-valued) in a bit-string and a fitness function (similar to Eq. 1) defines how well it performs. Both single point crossover and mutation are used. Typical parameters we used for our experiments are: population size = 30, mutation rate = 0.01, crossover probability = 0.5, generation gap = 50%, max. generations = 150, number of trials = 10. All node states are set to 0 before each evaluation in order to obtain reproducible results.

Experiment 1: Generalization as a Function of the Number of Inputs

This experiment confirms that learning and generalization in a recurrent RBN works as well as in feed-forward RBNs. "Generalization" is the ability of a network to successfully solve a task even though not all possible input-output mappings were taught. In other words, the system is able to discover the underlying structure of a task.

Fig. 3 illustrates that the higher the number of inputs, i.e., the more complex the task, the better the network is able to generalize. E.g., for 6 inputs, only about 60% of the patterns need to be taught in order for the network to recognize 100% of them.

Task: even-odd. Learning method: simulated annealing (SA), changing connections only. Parameters: $N=20$ nodes, averaged over 1000 runs

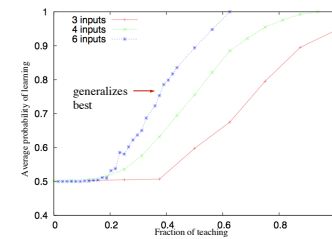


Figure 3: Varying the number of inputs for the even-odd task

Experiment 2: How does the Number of Incoming Links k Influence Generalization?

Fig. 4 (SA) and 5 (EA) show that generalization is most effective for $k=2$ nets. $k=1$ nets are too "frozen" to compute usefully and are unable to learn any pattern.

Task: even odd. Learning: connections only

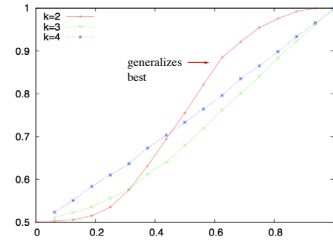


Figure 4: Even-odd task trained by SA. $N=20$ nodes.

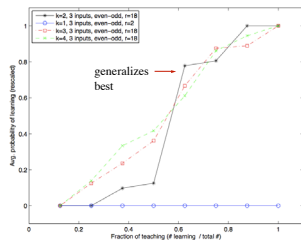


Figure 5: Even-odd task trained by an EA. $N=16$ nodes.

Experiment 3: #1-Mapping and Density Task

Fig. 6 suggests that for the #1-mapping task, $k=2$ nets also generalize better than any other. Again, for higher number of inputs, generalization is better. For 5-input nets, the EA was not able to reach 100% learning performance. The same behavior can be observed in Fig. 7 for the density task. This task is harder than the others ("global" decision).

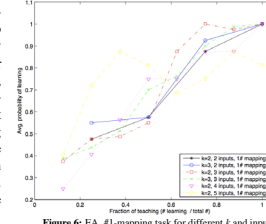


Figure 6: EA, #1-mapping task for different k and inputs

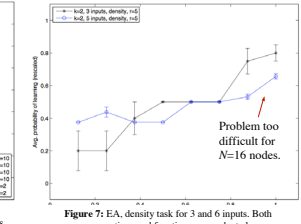


Figure 7: EA, density task for 3 and 6 inputs. Both connections and functions were adapted.

Experiment 4: Changing Functions and Variable k

Fig. 8 shows that randomly updating the node functions from time to time helps generalization. More investigation is needed, however. Allowing variable k 's ($0 < k_i < 10$) for each node i (as opposed to exact k 's) does not improve generalization (Fig. 9)

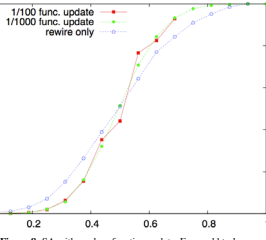


Figure 8: SA with random function update. Even-odd task

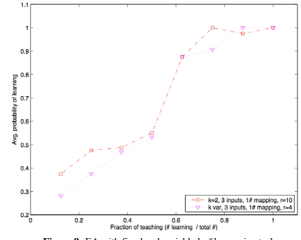


Figure 9: EA with fixed and variable k . #1-mapping task.

Experiment 5: Does the EA favor $k=2$ Networks?

Lenke et al. [6] use a EA to adapt both the functions and connections. For their specific task, they found no evidence that EA performance is better for $k=2$ nets or that the EA prefers $k=2$ nets. We measured the connectivity for the #1-mapping task for variable k nets and can confirm the results if a standard mutation is used.

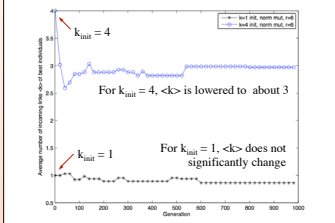


Figure 10: The evolution of the average $\langle k \rangle$ of the best individuals.

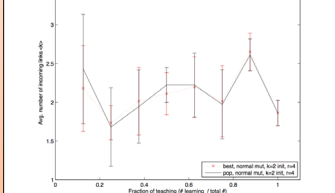


Figure 11: The evolution of the average $\langle k \rangle$ does not show any specific trend.

Conclusions

Both SA and EAs result in similar generalization behavior. Given sufficient number of nodes N , the higher the number of inputs (i.e., the more complex the task), the better the networks generalize. $K=2$ networks "at the edge of chaos" present a special case and they tend to generalize better too. Recurrent RBNs are thus well able to uncover the underlying general rule of simple tasks.

Future work will concentrate on:

- the minimal number of nodes required to solve a given task and to generalize successfully. Can one define the capacity of learning for RBNs?

growing network models that also allow to dynamically change the number of nodes;

topology rewiring algorithms; and

the interaction between topology, connectivity, and learning.

The authors thank Yuzuru Sato for the interesting discussions and comments.

References

- [1] Van den Broeck, C. and Kawai R. (1990). Phys. Rev. A, 42(10):6210-6218
- [2] Patarnello, S. and Carnevali, P. (1987). Europhys. Lett., 4(4):503-508
- [3] Kauffman, S. (1993). The Origins of Order. OUP
- [4] Derrida, B. and Pomeau, Y. (1986). Europhys. Lett., 1(2):45-49
- [5] Yao, X. (1999). Proc. IEEE, 87(9):1423-1447
- [6] Lenke, N. et al. (2001). Physica A, 301:589-600
- [7] Carnevali, P. and Patarnello, S. (1987). Europhys. Lett., 4(10):1199-1204