

## Abstract

**Recurrent neural networks** are important models of neural computation. This work relates the power of them to those of other conventional models of computation like **Turing machines** and **finite automata** and proves results about their learning capabilities. Specifically , it shows ,

- (a) **Probabilistic recurrent networks** and **Probabilistic turing machine** models are equivalent.
- (b) Probabilistic recurrent networks with **bounded** error probabilities are no more powerful than deterministic finite automata.
- (c) Deterministic recurrent networks have the capability of learning **P-complete** language problems ( which are the hardest problems in P to parallelize ).
- (d) Restricting the weight-threshold relationship in deterministic recurrent networks may allow the network to learn only weaker classes of languages ( the **NC** class ).

# Learning Capabilities of Recurrent Neural Networks.

Bhaskar DasGupta

Department of Computer Science

The Pennsylvania State University

University Park , PA 16802

September 15, 1991

## 1 Preliminaries.

**Recurrent neural networks** are important models of neural computation. In such a network, processors (including hidden units) are interconnected in an arbitrary graph (which may be cyclic), and each processor changes its state based on its weighted connection to its neighbours. For deterministic networks (e. g. **recurrent back-propagation**) the state change is deterministic, whereas for probabilistic networks (e. g. **Boltzmann Machines** (abbrev. as **BM**)) the state change is probabilistic. Given an encoding of the input in the input processor, the processors update their states until the network reaches a **stable state** (if there is one) in which no more processor changes state, and the output response is decoded from the states of the output processors.

Following popular learning models, we can assume that the inputs have been labelled (before the learning starts) into **two** categories, namely the **positive** and **negative** sets. (more than two categories is also possible, and it poses no problem in the proofs). During the learning phase, the network is presented a subset of positive or negative examples, and the networks repeatedly adjust its weight according to some learning paradigm based on the compatibility of its response to the actually correct response. After the learning phase is over, such a network should ideally be able to classify any input correctly. An **error** occurs when it **misclassifies** an input presented after the learning phase. For probabilistic networks the **error probability** is defined to be the **maximum** probability of misclassifying any input after the learning phase. Space limitation prevents more formal description of the above concepts.

## 2 Statement of problem addressed.

Although recurrent networks are believed to be extremely powerful, very little is known about the relationship of these models with other conventional models of computation, like **Turing machines** and **finite automata**, as well as their learning capabilities i. e. the class

---

<sup>0</sup>This is intended for the Neural Networks session as a concise paper.

of languages they can learn , which is important to determine their capabilities of solving complex tasks. This paper provides results in the above-mentioned directions.

### 3 Approaches employed and main results.

We use results from graph theory , and computational complexity theory [1] [2] [3] [4] to prove the following results. All proofs are omitted due to lack of space.

**Theorem 1** *Probabilistic recurrent networks ( abbrev. as PRN ) and Probabilistic Turing Machines ( abbrev. as PTM ) are polynomially equivalent. More specifically , a PRN can be simulated by a PTM with a polynomial increase in running time and , conversely , a PTM of time complexity  $T(n)$  can be simulated by a PRN of size at most polynomial in  $T(n)$ .*

The significance of the above result lies in the fact that a PRN with polynomial number of processors can therefore learn NP language problems , whereas , for example , a Hopfield network with polynomial number of processors is already proved not to have such capabilities even if allowed to run for exponentially many steps ( and , hence , PRNs are more powerful ).

**Theorem 2** *A probabilistic recurrent network with error probability bounded by  $\epsilon$  for some constant  $0 < \epsilon < \frac{1}{2}$  can be simulated by a deterministic automaton with  $e$  states , where  $e \leq (1 + \frac{r}{\epsilon})^{n-1}$  ,  $r =$  number of correct ( accepting ) configurations of output processors of the network , and  $n =$  total number of configurations of output processors of the network.*

The above result shows that PRNs with bounded error probabilities are less powerful , since they can be simulated by deterministic finite automata.

The following result shows the learning capabilities of deterministic recurrent networks.

**Theorem 3** *A deterministic recurrent neural network which has at least one stable state can learn any P-complete language problem.*

The above result is extremely important , since P-complete problems are a class of problems which have polynomial time sequential solution , but are most difficult to parallelize.

The next result shows that restricting the thresholds and weights may result in limited learning capabilities ( problems in class NC have efficient parallel solution ).

**Theorem 4** *A deterministic recurrent neural network ( with at least one stable state ) whose each processor  $v$  has a restricted value of its threshold  $h_v$  , namely ,*

$$\sum_{\epsilon} w_{\epsilon} < h_v < \sum_{\epsilon} w_{\epsilon} - 2w$$

*, where  $\epsilon \equiv (v, z)$  is the connection of processor  $v$  to another processor  $z$  ,  $w_{\epsilon}$  is the weight of this connection and*

$$w = \max_{\epsilon} \{w_{\epsilon}\}$$

*can learn only NC language problems.*

## 4 Conclusion.

The above results indicate the learning capabilities of recurrent networks. The efficiency of learning procedure of recurrent neural networks with stable states is currently investigated and any further progress in this area will be reported later.

## References

- [1] J.Gill , Computational Complexity of Probabilistic Turing Machines , *SIAM J. of Computing* , 7 , 4 , pp. 675-695 , 1977.
- [2] I. Parberry , A Primer on the Complexity Theory of Neural Networks , *in Formal Techniques in Artificial Intelligence , A Sourcebook* , ed. R. B. Banerji , pp. 217-268 , 1990.
- [3] I. Parberry , G. Schnitger , Relating Boltzmann Machines to Conventional Models of Computation , *Neural networks* , 2 , pp. 59-67 , 1989.

- [4] M. O. Rabin , Probabilistic Automata , *Information & Control* , 6 , pp. 230-245 , 1963.