ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 151-156

An Algorithm to Learn Sequences within the Connectionist Sequential Machine

Olivier Sarzeaud and Norbert Giambiasi

LERI
Parc Scientifique Georges Besse
30000 Nîmes, FRANCE

Abstract. The connectionist sequential machine is a general neural network structure for dealing with sequences. It adopts the structure of the sequential machine, but replaces the digital circuits with multilayer neural networks. An interesting case of learning occurs when no knowledge about the internal states of the machine is available. The learning algorithm that we propose in this paper is based on the associative reward/penalty and gradient back propagation algorithms. Experiments in automata synthesis have also revealed the generalization of synthesis in the case of an incomplete learning base.

1 Introduction

Dealing with sequences is becoming more and more important today. Nevertheless, it remains a poorly studied problem in the connectionist community. There are at least four approaches to dealing with sequences in neural networks [5]. The approach considered in this paper is that having recurrent connections on the global network; it is the most general. A few models belonging to this approach have been proposed at this time :

- Jordan's model [3]: the network is a multilayer perceptron with recurrent connections from the output units to some input units.
- Elman's model [1]: the network is a mutilayer perceptron with recurrent connections from the hidden units to some input units.
- The connectionist sequential machine [6]: the machine is composed of two multilayer perceptrons. This model includes the models previously mentioned.

Unfortunately, no efficient learning algorithm has yet been proposed for this structure. In this paper, we describe such a learning algorithm and the results obtained.

2 The connectionist sequential machine

The connectionist sequential machine adopts the structure of the sequential machine (Fig. 1), but replaces the digital circuits with multilayer neural networks. The first block carries out the transition function δ and the second performs the output function λ. In the transition network, all output neurons are connected back to a subset of input neurons. The number of connections equals the number of output neurons.

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 151-156

The network which performs the transition function is called the transition network. From the present input i(t) and the past state s(t-1), it computes the next state s(t):

$$s(t) = \delta(i(t), s(t-1)).$$

The network which performs the output function is called the output network. From the present input i(t) and the present state s(t), it computes the present output o(t):

$$o(t) = \lambda(i(t), s(t)).$$

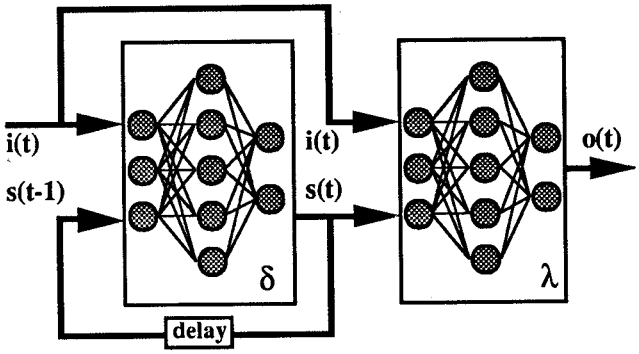

Fig. 1. The connectionist sequential machine

3 Learning

Knowledge about the functions can be classified into several cases:

0/ δ and λ are specified. The learning base is composed of sequences of pairs :
(input, past state) -> (present state, ouput)

1/ δ and the global behavior of the entire machine are specified. The learning bases are composed of sequences of pairs:
(input, past state) -> (present state)
(input) -> (output)

2/ λ and the global behavior of the entire machine are specified. The learning bases are composed of sequences of pairs:
(input, present state) -> (output)
(input) -> (output)

3/ Only the global behavior of the machine is specified. The learning base is composed of sequences of pairs:
(input) -> (output)

The first three cases are easy to solve:

0/ A learning base can be built for each network. A supervised learning algorithm can thus be run on each network separately.

1/ A supervised learning algorithm can first be run on the transition network. A supervised learning algorithm can then be run on the output network, the present state being given by the transition network from a randomly chosen initial state. If the machine has not converged after a given period of time, the initial state is probably wrong. Therefore, another must be chosen before teaching begins again.

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 151-156

2/ The pairs *(input, present state) -> (output)* are put in sequences in the learning base. Thus, it is easy to extract pairs having the form: *(input, past state) -> (present state)* from this base. This case is therefore the same as the first: a supervised learning algorithm can be run on each network separately.

The last case is much more difficult to solve because no knowlege on the internal states is available (i.e. the desired output of the transition network and, consequently, the complete input of the output network cannot be determined). A learning procedure for the global structure must be found in order to modify the behavior of both networks simultaneously.

The basic idea for such an algorithm is to prescribe the use of any reinforcement learning procedure for the transition network and any supervised learning rule for the output network. The algorithm that we propose here is called the CSM algorithm. It uses the associative reward/penalty algorithm (A_{RP}) for the transition network and the back-propagation learning rule for the output network.

Before giving a detailed description of the CSM algorithm, let us briefly recall the gradient back-propagation and A_{RP} algorithms using the same formalism.

3.1 The back-propagation learning algorithm

The version presented here is known as the stochastic back-propagation algorithm. It was originally presented by Rumelhart and al [4]. It is an iterative gradient descent algorithm that tries to minimize the mean square error between the actual output and the desired output of a multilayer feed-forward neural network. Its essential components are non-linear units governed by the rule:

$$y_j = f(x_j) = \frac{1 - e^{-x_j}}{1 + e^{-x_j}} \qquad \text{where } x_j = \sum_i w_{ji}\, y_i$$

w_{ji} is the weight on the connection from unit i to unit j. y_j is the output of unit j. The function f is a sigmoid logistic function whose asymptotic values are -1 and +1.

We can now describe the algorithm:

- First, the weights are initialized at small random values.
- Then, as long as the mean square error for the entire database is too great:
  - For each input-output pair in the learning base:
    - An input x is presented.
    - The corresponding output is computed.
    - The weights are modified using the usual delta rule:

$$\Delta w_{ji} = -\alpha \delta_j f'(x_j) y_i$$

$$\text{where } \delta_j = \begin{cases} y_j - d_j & \text{for the output units} \\ \sum_i \delta_i f'(x_i) w_{ij} & \text{elsewhere} \end{cases}$$

α is the learning rate, to be taken between 0 and 1.

3.2 The Associative Reward/Penalty learning algorithm

This algorithm was originally presented by Barto and Anandan [2]. It fits multilayer feed-forward neural networks, provided the output neurons are stochastic units governed by the standard stochastic dynamical rule:

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 151-156

$$\text{Prob}(y_j = \pm 1) = g(x_j) = \frac{1}{1 + e^{\pm 2x_j}} \qquad \text{where } x_j = \sum_i w_{ji} y_i$$

The function g is a sigmoid logistic function whose asymptotic values are 0 and 1. The other neurons are governed by the same rule as in the back-propagation.
When compared with the back-propagation algorithm, the essential difference with this algorithm is that no desired output is needed; a binary reinforcement signal is used instead. As in the back-propagation algorithm, the weights are modified using a gradient descent method. We consider as the effective output the average output value $\langle y_j \rangle$ that would be obtained after the same input has been presented many times. It should be recalled that the stochastic behavior of the output neurons allows different outputs to be obtained for given inputs and weights. The average output value can be calculated as follows:

$$\langle y_j \rangle = (+1)g(x_j) + (-1)[1 - g(x_j)] = \tanh(x_j)$$

The desired (target) output is constructed according to the reinforcement signal r:

$$\zeta_j = \begin{cases} y_j & \text{if } r = +1 \quad \text{(reward)} \\ -y_j & \text{if } r = -1 \quad \text{(penalty)} \end{cases}$$

The A_{RP} algorithm therefore works as follows:

- First, the weights are initialized at random values that place the probabilities of the output neurons around 0.5.
- Then, as long as the network outputs have not been satisfactory for several loops on the entire learning database (i.e. at least one negative reinforcement signal has been generated):
  - For each input in the learning base:
    - An input x is presented.
    - A possible corresponding output is computed.
    - The weights are modified using the usual delta rule:

$$\Delta w_{ji} = -\eta(r)\delta_j y_i$$

$$\text{where } \delta_j = \begin{cases} \langle y_j \rangle - \zeta j & \text{for the output units} \\ \sum_i \delta_i f'(x_i) w_{ij} & \text{elsewhere} \end{cases}$$

$\eta(r)$ is the learning rate. Typically, η is taken as 10-100 times larger for r=+1 than for r=-1.

At the end of the learning process, the output units are quasi-deterministic.

3.3 The CSM algorithm

For teaching the connectionist sequential machine, we only have a database of input/output pairs representative of the problem. The learning algorithm is therefore as follows:

- The weights of both networks are initialized at small random values.
- Then, as long as the machine outputs have not been satisfactory for several loops on the learning database:
  - For each input-output pair in the database:
    - An input is presented to the connectionist sequential machine.
    - An internal state is generated by the transition network and the output is computed by the output network.

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 151-156

- Back-propagation is run once on the output network, so that it may learn the correspondence between the internal state plus the input and the output.
- If the new output, computed after the modification of the weights on the output network, is close enough to the desired output (inferior to a fixed threshold), a positive reinforcement signal is sent to the transition network, otherwise a negative signal is sent. A_{RP} is run on the transition network.

As in the back-propagation and A_{RP} algorithms, a modification of all the weights occurs after each presentation of an example of the database. To use this algorithm in the most efficient way, the learning rate coefficients η of A_{RP} and α of the back-propagation have to be adjusted manually in order to give a quick convergence.

4 Experiments

The connectionist sequential machine has been tested on problems of sequence recognition. A detailed description of the recognition of 1001 is given below.

Learning data base. The learning data base is composed of a sequence of several input/output values. The recognition of overlapped 1001 sequences is imposed. We present here the learning data base:

Input : 10101001100100100101101001011011000001001
Output : 00000001000100100100000001000000000000001

Topology of the networks used. The transition network contains 4 units on its input layer (3 are used to represent the internal state, the last represents the current input), 3 units on its hidden layer and 3 units on its output layer. The output network contains 4 units on its input layer, 4 units on its hidden layer and 1 unit on its output layer.

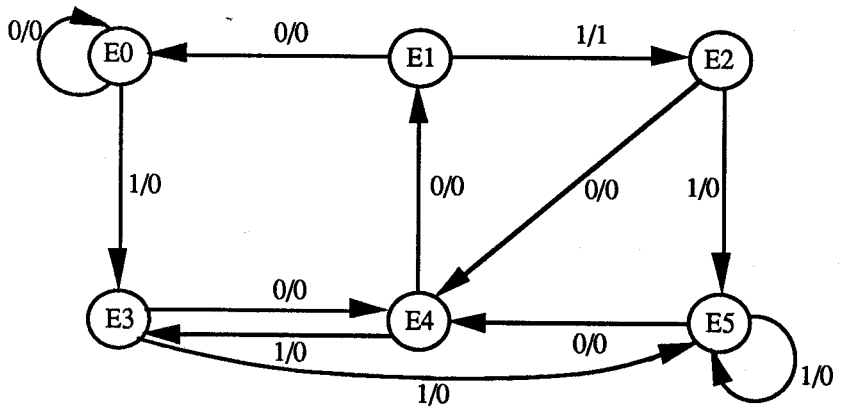

Fig. 2. State transition graph of the connectionist sequential machine after it has been taught to detect the sequence 1001

Results. The convergence of the connectionist sequential machine has been reached after 400 iterations (one iteration consists of one reading of the entire learning data base). The state transition graph of the machine after learning is shown in Figure 2. It contains six internal states. E0 is the initial state. The labels on the directed arcs specify the input and the corresponding output.
The input is a binary value. The synthesized graph is complete: there are two transitions available from any state.
In our example, the connectionist sequential machine uses 3 units to represent the internal state. Thus, eight internal states are available. Nevertheless we can notice that not all the states are used.
Last but not least, it is interesting to note that the 1/0 transition for the state E0 does not appear explicitly in the learning data base. We can nonetheless verify that this transition has indeed been learned when giving the connectionist sequential machine new sequences using this transition.

5 Conclusion

The CSM learning algorithm involves back-propagation learning on the output network and associative reward/penalty learning on the transition network of the connectionist sequential machine. Experiments concerning the recognition of sequences of length 4 demonstrate that learning from sequences of behavior is possible. Furthermore, in the case of an incomplete learning base, the generalization property of the connectionist sequential machine allows the appropriate behavior to appear. Provided with an a priori superior number of possible internal states, the learning only makes use of a subset, but does not generate a minimal state transition graph.
Future research will include extending the learning process to sequences of input/output *vectors*. We will also study the learning of sequences when no knowledge of the desired output is available. This may eventually lead us to examine the cooperation of A_{RP} algorithms.

References

1. J. L. Elman: Finding Structure in Time. Center for Research in Language Technical Report 8801. Univ. of California, San Diego. April 1988
2. A. G. Barto, P. Anandan: Pattern Recognizing Stochastic Learning Automata. IEEE Transactions on Systems, Man and Cybernetics. SMC-15: 360-375. 1985
3. M. I. Jordan: Serial Order: Parallel Distributed Processing Approach. ICS Report 8604. Univ. of California, San Diego. May 1986
4. D. E. Rumelhart, G. E. Hinton, R.J Williams: Learning representations by back-propagating errors. Letters to Nature 323, 533-536. October 1986
5. O. Sarzeaud, N. Giambiasi: Apprentissage de séquences par introduction de retards dans un réseau combinatoire. NeuroNîmes'92. Nîmes, France. November 1992
6. C. Touzet, N. Giambiasi: Connectionist finite-state machines. Proc. of the International Joint Conference on Neural Networks 1990. Washington D.C., USA. 1990