Mechanisms For Handling Sequences With Neural Networks

Claudia Ulbricht, Georg Dorffner
Austrian Research Institute for Artificial Intelligence, Schottengasse 3, Vienna, Austria[1]

Stéphane Canu
Lyonnaise des Eaux Dumez, Rue du Fonds Pernant, Compiègne, France

Didier Guillemyn
Babbage Institute for Knowledge and Information Technology, J.Plateaus. 22, Gent, Belgium

Gurutze Marijuán, Javier Olarte
Elorduy, Sancho y CIA, S.A., Parque Tecnológico de Zamudio, Zamudio, Spain

Clemente Rodríguez, Ignacio Martín
LABEIN Technological Center, Parque Tecnológico de Zamudio, Zamudio, Spain

Abstract

This paper is intended to give an overview of methods for handling sequences with neural networks. Since many typical neural networks cannot be used for processing temporal information they have to be extended by some mechanism to be able to do so. Two types of mechanisms can be distinguished: *non-recurrent mechanisms*, such as windows and time delays, and *recurrent mechanisms* based on feedback.

However, the properties of networks handling sequences are not only dependent on the employed mechanism, but also on the underlying network paradigm (e.g. multi-layer perceptron, Hopfield network, Kohonen network, etc.). Each combination of a network paradigm with a mechanism results in a different temporal network having its own features. The outcome of a comparative study within the ESPRIT-II project NEUFODI ("Neural Networks for Forecasting and Diagnosis Applications")[2] suggested a division of these networks according

[1]The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Ministery of Science and Research.

[2]NEUFODI is partly funded by the EC commission as ESPRIT-II project No. 5433 and is conducted in cooperation with BIKIT (Belgium), Lyonnaise des Eaux Dumez (France), Elorduy y Sancho, and Labein (both Spain). The Austrian contribution is supported by a grant from the Austrian Industrial Research Promotion Fund, Project No. 2/282.

to their overall characteristics into the following four categories: *extended feedforward networks*, *single-step recurrent networks*, *extended stabilizing networks*, and *extended competitive networks*.

This paper provides a description of the advantages and disadvantages of different approaches. It also presents a few selected results out of the comparative study. As such this paper can be used as a basis for deciding which network to take for a given task.

1 Introduction

In order to deal with sequential input a neural network needs some kind of specially designed *mechanism*. Several single input patterns together, in their given order, have to influence the output in some way. Examples for such mechanisms are windows, time delays, or feedback. They are described in detail in section 3. However, not only the mechanism used, but also the underlying network architecture determine the properties of a network that handles sequences. Combinations of the two dimensions, the *mechanism* and the *underlying network paradigm*, result in various approaches to handling sequences with neural networks. Distinguishing these two dimensions turned out to be crucial for analyzing and describing the network properties.

The purpose of this paper is to give an overview of such networks. This is done by providing a classification and by presenting some of the results obtained in the comparative study within the ESPRIT-II project NEUFODI (in section 4). The study involved a wide variety of experiments to obtain empirical criteria about the strengths and weaknesses of different approaches. Finally, in section 5 preliminary heuristics as to which approach is best suited for which type of application are presented.

2 A Classification

Based on the results of a comparative study, the network architectures for processing sequences have been classified into four categories: *extended feedforward networks*, *single-step recurrent networks*, *extended stabilizing networks*, and *extended competitive networks*. They form two groups: *non-attractor* and *attractor networks*. Possible combinations of these *network paradigms* with *mechanisms for handling sequences* are shown in Table 1. The mechanisms for handling sequences that were investigated (*windows* (WI), *time delays* (TD), *partial feedback* (PF), and *full feedback* (FF)) can be found in the columns. A network that can

handle sequences employs at least one sequential mechanism. Combinations of several mechanisms are possible. For instance, a network with feedback can also have an input window, thereby combining the capabilities of two mechanisms. Although this classification has its overlaps, it turned out to be of practical importance because it roughly divides the different models according to their overall characteristics and to their suitability to different types of applications.

Resulting Network Types		Mechanism			
		WI	TD	PF	FF
Non-Attractor Networks	Extended Feedforward Networks	1			
			2		
	Single-step Recurrent Networks			3	
					4
Attractor Networks	Extended Stabilizing Networks				
	Extended Competitive Networks			5	

Table 1: A Network Classification Scheme

The numbers in Table 1 indicate the types of networks selected for being tested within the presented comparative study. One of the tested feedforward networks (1) has an *input window*, the other one is extended by *time delays* (2). While the window affects only the input layer the time delays are distributed over the whole network. The other three tested networks employ delayed feedback. If such networks are updated like feedforward networks (usually a single update per time step) the networks keep their general feedforward characteristics. Networks of this kind are also dubbed "recurrent feedforward networks," "simple recurrent networks," or "partially recurrent networks" [Hertz *et al.*, 1991]. The tested *partially recurrent network* (3) has feedback loops from the hidden and the output layer. The *fully recurrent network* (4) contains only a single layer in which all units have feedback connections to all others. The *competitive recurrent network* (5) belongs to the group of attractor networks. It is equipped with a single feedback loop, but its main characteristics are determined by the competition in the layers. It employs special types of updating and training mechanisms. Due to the competitive layers its behavior is in some ways comparable to that of *stabilizing networks* converging to stable attractors.

The tested mechanisms are described in detail in the following section, and some results of the experiments will be presented thereafter, in section 4.

3 Mechanisms

The mechanism used for handling sequences is independent of the underlying network paradigm. Any combination of a network paradigm with a mechanism results in a different neural network. Usually, the whole network is discussed, but in this section only one dimension (see Table 1), the mechanism, is taken into consideration.

Two types of mechanisms can be distinguished: non-recurrent and recurrent mechanisms. The difference is depicted in Figure 1. A simple three-layer feedfor-

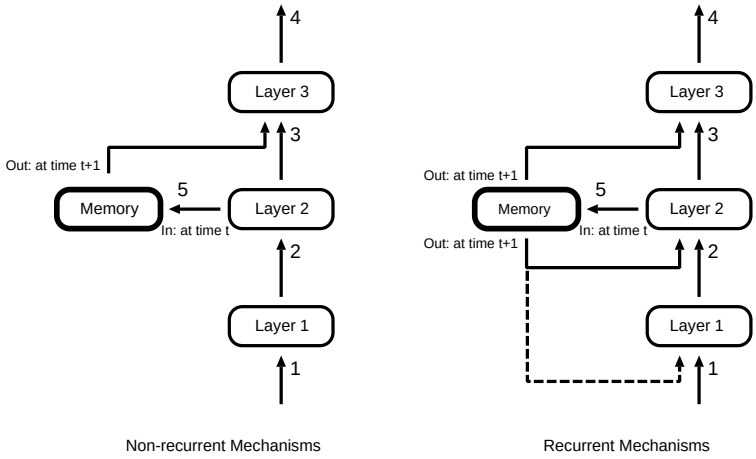

Figure 1: Non-recurrent and Recurrent Mechanisms

ward structure is shown. The numbers denote the order in which the layers are updated at time t. This time scale is determined by the input sequence which contains a single sequence element per time step. It is not dependent on the time scale of the network updates. For a single input pattern, information is propagated through the network as it is usually done (steps 1-4). After that, the content of one layer is copied to a memory (step 5), where it is stored for the next updating cycle (at time t+1). When the next input pattern is active the content of the memory layer is released. As long as no recurrent mechanism is employed, information can only flow into the direction of propagation. It can only stay in the network for a limited number of time steps. In a network employing a recurrent mechanism, though, it can stay longer because it can be fed back — either directly (to layer 2) or indirectly (to layer 1). One of the strengths of such delayed feedback is that it allows information to stay within the system.

An important point is that the type of feedback used in stabilizing or competitive networks is not delayed and is thus different from that of the feedback needed for processing sequential aspects. The former is part of the underlying network paradigm and plays a role only at time step t (for instance, when the network

converges to a stable attractor). The latter, though, can be regarded as feedback delaying information until time step t+1. These delayed feedback signals can then reenter the network together with the next incoming sequence element.

3.1 Non-recurrent Mechanisms

Windows: An approach often investigated is that of employing a *time window* which stores a restricted part of the sequence. An associative network analyses this part of the sequence before the window is shifted by one or more elements further in time. An example can be found in [Tom and Tenorio, 1989]. Such a window can be modeled by collecting sequence elements arriving one after the other in some input memory until they can be used by some regular associative network at once. Thus, the temporal dimension is reduced to zero by parallelizing sequence elements. Input windows are often used because they do not affect the underlying network architecture. Any network paradigm and any tool can be used by just adjusting the input data stream. However, it is also possible to apply windows to several layers to repeat the windowing effect, as it is done in the TRACE model of speech perception [McClelland and Elman, 1986]. Such additional windows let the actual window size of the network grow.

A drawback of the windowing approach is that it results in relatively large networks. The reason is that the number of nodes and connections is directly dependent on the size of the windows. At the same time invariance problems, well-known in visual processing, arise. Single patterns in a sequence that can be recognized at one point in time (i.e. at a certain position in the window) cannot automatically be recognized at all other points.

Time Delays: The effects achieved with the windowing technique can also be achieved with time delays. In so-called *time delay networks*, patterns can be delayed (see for instance [Wan, 1990] or [Waibel, 1988]) so that signals originally ordered in time arrive at a single unit in parallel at the same point in time. Thus, they can be processed at once. If this internal mechanism is employed in more than one layer scattered over the whole network multiple parallelizations occur within the network.

Problems are similar to those of windowing techniques. The time period processed is strictly limited by the number and arrangement of the time delays.

3.2 Recurrent Mechanisms

Since feedback used for sequence processing is delayed this approach is in some ways comparable to the time delay approach. The only — but important — difference is that feedback leads to recurrence. This is important because recurrence allows *state formation* which turned out to be relevant for a wide range of tasks. In contrast to units in feedforward networks, the units of such networks indirectly receive information of their own past activations. Thus, a new input can be processed together with the context provided by the feedback signals of the previous state. Both, the new input and the context determine the output and the new state of the network. As a consequence, the output of a network based on delayed feedback is a function not only of a sequence of inputs, but also of the initial state.

The advantage of using recurrent mechanisms is that, theoretically, any past sequence element can have an influence on the output, whereas the capacity of the memory of networks employing only non-recurrent mechanisms is limited by the size of the windows or by the number of time delays. The memory of a network with feedback has no definitive temporal limitation. The advantage of such a memory has to be paid by incompleteness, though. Past inputs are not kept in their complete original form, but only a few features can be extracted and memorized. However, in reality the knowledge on past inputs and states decays rapidly. On the one hand, this effect seems to be very reasonable because the further back in time the less important events seem to be for the current situation. On the other hand, some past events can be of great importance. This poses the question which information should be stored and which should be forgotten. Therefore a mechanism would be needed which can distinguish events that are relevant for future decisions and those which are not. A possible way to decide on this could be to let another network learn from experience which features are relevant.

Two types of delayed feedback can be distinguished: partial and full feedback. When all the units receive feedback from all the other units in the network (no matter whether they directly receive feedback from themselves or not) one can speak of full feedback. All other forms of feedback can be regarded as some type of partial feedback. It is again important to note that this refers only to feedback used as a mechanism for handling sequences, but not to feedback that is part of the underlying network paradigm.

Partial Feedback: A typical form of partial feedback is achieved by adding a few feedback loops to some layered network. Many such networks have been investigated in detail. An example is the Jordan network [Jordan, 1986] in which the output values together with the activation of the previous state layer deter-

mine the new activation of the state layer. Together with the input, the state layer is fed forward to the hidden layer. Another type of feedback is used in the Elman network [Elman, 1990]. A single feedback loop is placed around the hidden layer. Feeding back the output has different effects than feeding back the content of a hidden layer. Which type of feedback should be used is very much dependent on the type of data set.

The arrangement of the feedback loops is relevant for the performance of the network. Feedback makes state formation possible, but whether a certain arrangement is appropriate for a given task or not is dependent on the nature of the task. Feedback of the network output leads to the problem that at the beginning of the training phase, the network usually produces incorrect output. In most cases, replacing these incorrect unit activations by the correct ones (given in the training set and needed anyway for training) turned out to accelerate the training procedure. This technique is usually referred to as "teacher-forced learning" [Williams and Zipser, 1989].

Full Feedback: Adding more and more feedback connections eventually leads to a fully connected network, in which the output of each unit is propagated to all the other units (and maybe also to itself). In such a network, one can no longer speak of layers, as all the units are part of a single layer. However, the units do not necessarily play the same roles. Some of them might receive input from outside while some others might be used as output units. Similar to above the employed full feedback which is used as a mechanism for handling sequential aspects is independent of the updating and training procedures of the underlying network.

4 Comparative Results

The second goal of the study being reported in this paper was to experimentally compare different neural network approaches to handling sequences so as to obtain empirical criteria about their advantages and disadvantages. Based on the classification introduced in section 2, five networks were chosen and implemented – a *window network*, a *time-delay network*, two *single-step recurrent networks* (one is a partially recurrent combination of the Elman- and Jordan networks, the other one a network with full feedback) and a *competitive recurrent network*. The partially, the fully, and the competitive recurrent network were specifically designed for this study ([Canu, 1992], [Guillemyn, 1992], [Ulbricht, 1992]).

The focus of the experiments was put on practical applications. However, certain properties of networks handling sequences cannot be tested empirically if the

characteristics of the data sequence are not known. Therefore, in addition to several real world time series, sequences generated by well-known mathematical models and artificially created sequences were used for testing. Moreover, the experiments were divided into two further classes depending on the task to be solved – classification of sequences and forecasting future sequence elements.

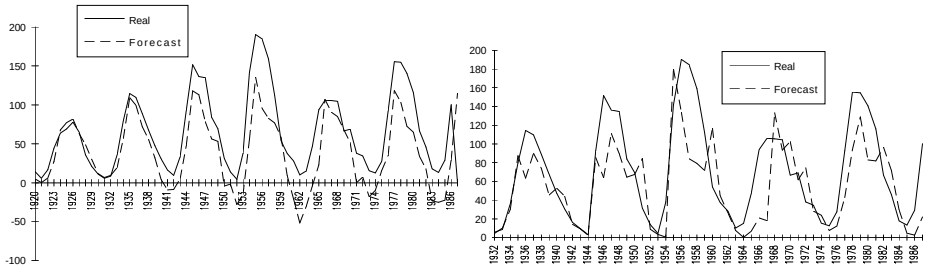

Figure 2: *Window Network and Time Delay Network*

The artificial data mainly served to test criteria such as fault-tolerance or the capacity to process sequences of a certain order. Not surprisingly, the limits of rigid approaches like windowing or time delay techniques were confirmed. Among the model-generated data there were sequences such as those generated by a parity automaton, a hidden Markov process, or an ARIMA time series. Both test phases yielded some very conclusive results concerning the usefulness of each model for certain applications. It is planned to publish those results in detail elsewhere (Ulbricht et al., in preparation).

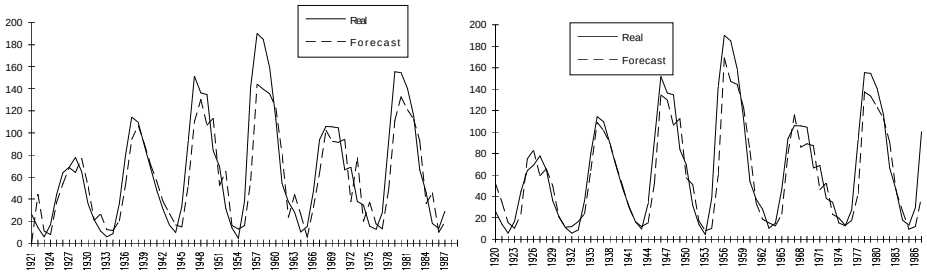

Figure 3: *Partially Recurrent Network and Fully Recurrent Network*

In the final experimental phase several real world time series were tested. The one that lead to the most interesting results was the well-known sequence of the number of sun-spots per year collected since the year 1700 [Weigend *et al.*, 1990]. Figures 2 through 4 show the actual and the forecast values of all five networks while trying to predict the series. For each year, the networks were used to predict the current value based on the previous values of the actual sequence (that is, errors in prediction were not permitted to propagate to subsequent cycles). The single-step recurrent networks generally performed better than the extended feedforward networks and the competitive network. The performance

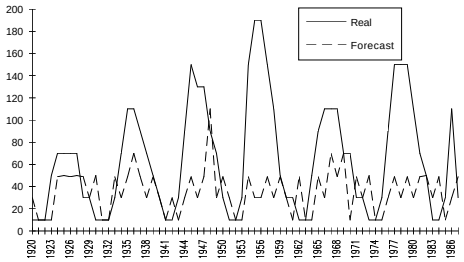

Figure 4: *Competitive Recurrent Network*

of the window network was relatively good in the beginning, but decreased with time. It also occasionally predicted negative values. The time delay network was able to predict the general characteristics, albeit sometimes with rather high relative error. The partially recurrent network, on the other hand, followed the real trajectory much more closely. The best results were obtained with the fully recurrent network. The forecast value matched the actual value in most cases. The network was even able to forecast the peak of around 1957 which does not appear in the sequence before.

Not surprisingly the results for the competitive recurrent network are different due to the fact that it is an attractor network. Its output is not an arbitrary numeric value, but one out of a limited number of categories that are then mapped to output values (in this case to ten values). The graph shows that the network was able to follow some ups and downs, but not as closely as the other networks.

5 Which Network to Take

The results of all three categories of experiment (artificial, model-generated, and real-world data) taken together provide a thorough insight into the workings of the most common network types for handling sequences. They also provide rough guidelines for practitioners as to which network to choose for a given practical application (see [Ulbricht *et al.*, in preparation]). It could be shown that single-step recurrent networks are highly appropriate for realistic forecasting tasks because they are well suited to modelling functional dependencies. Extended attractor networks like the competitive recurrent net, on the other hand, appear much more suitable for sequence classification tasks, such as in speech recognition, because the attractors can be used for representing the classes. The experiments with the model-generated data have shown that a deep analysis of the data of a specific application will in any case be vital for finding the optimal network solution.

The following heuristics can be used as guidelines as to which network to take for a given application:

Which network paradigm is appropriate?

- If the application involves modelling functional dependencies (for example, forecasting the amount of sun spots) a **non-attractor network** is appropriate.
- If the application is based on a classification task (for example, recognition of moving objects or spoken words) and if instances of all classes are available in the training set an **attractor network** is appropriate.

Which mechanism should be used?

- If a tool providing some standard neural network paradigm should be used without changing the supplied network architecture an **input window** can be used.
- If the size of the network does not matter (the network can contain many units and connections, training time can be long), *and* if the influences of one sequence element on another one are mainly short-term influences, *or* if the majority of the influences do not exceed a certain known number of time steps **windows** or **time delays** are appropriate.
- If not much is known about the influence and relevance of past sequence elements, *or* if the influence of past sequence elements is expected to be quite complex, *or* if the network should be able to handle invariance and fluctuations in the temporal dimension, *or* if the *state* of the network is needed to solve the given task **delayed feedback** should be used. It should be noted that the *appropriate arrangement* of feedback connections has to be found in accordance with the application and that the quality of *state formation* is very much dependent on the way the feedback connections are arranged.

6 Conclusion

The intention of this paper was to give an overview of neural networks that are suited to handling input sequences. The novel aspect here was that different approaches have been classified according to two dimensions: the mechanism employed and the network paradigm. Clearly distinguishing the two dimensions turned out to be important for analyzing neural networks that can handle sequences. The results of the experiments performed for the comparative study

reveal the characteristics of the different approaches. Practitioners working on some applications can use the presented heuristics as guidelines to select the network architecture best suited to the given tasks.

References

[Canu, 1992] S. Canu. *Description of Recurrent Neural Architectures Used for Benchmarking, Internal technical report.* 1992.

[Elman, 1990] J.L. Elman. Finding Structure in Time. *Cognitive Science*, 14:179–211, 1990.

[Guillemyn, 1992] D. Guillemyn. *Handling Temporal Sequences with Fully Recurrent Networks, Internal technical report.* 1992.

[Hertz *et al.*, 1991] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley Publishing Company, 1991.

[Jordan, 1986] M.I. Jordan. Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 531–546. Erlbaum, Hillsdale, NJ, 1986.

[McClelland and Elman, 1986] J.L. McClelland and J.L. Elman. Interactive Processes in Speech Processing: The TRACE model. In D.E. Rumelhart D.E. and J.L. McClelland, editors, *Parallel Distributed Processing*, volume I. MIT Press, 1986.

[Tom and Tenorio, 1989] M.D. Tom and M.F. Tenorio. A Spatio-Temporal Pattern Recognition Approach to Word Recognition. In *IEEE International Conference On Neural Networks*, volume I, pages 335–355, Washington D.C., 1989.

[Ulbricht *et al.*, in preparation] C. Ulbricht, G. Dorffner, Stéphane Canu, Didier Guillemyn, Gurutze Marijuán, Javier Olarte, Santiago Rementeria, and Clemente Rodríguez. *Neural Networks for Processing Sequences.* in preparation.

[Ulbricht, 1992] C. Ulbricht. Handling Sequences with a Competitive Recurrent Network. In *International Joint Conference on Neural Networks (IJCNN'92), Baltimore, Maryland*, volume I, pages 731–736, 1992.

[Waibel, 1988] A. Waibel. Connectionist Glue: Modular Design of Neural Speech Systems. In Touretzky D., editor, *Connectionist Models Summer School*, pages 417–425, Los Altos, CA, 1988.

[Wan, 1990] E.A. Wan. Temporal Backpropagation for FIR Neural Networks. In *International Joint Conference on Neural Networks*, volume I, pages 575–580, San Diego, 1990.

[Weigend *et al.*, 1990] A.S. Weigend, B.A. Huberman, and D.E. Rumelhart. Predicting the Future: A Connectionist Approach. *International Journal of Neural Systems*, 1(3):193–209, 1990.

[Williams and Zipser, 1989] R.J. Williams and D. Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1:270–280, 1989.