ADAPTIVE DECOMPOSITION OF TIME

(To appear in O. Simula, editor, Proceedings of the International Conference on Artificial Neural Networks ICANN'91. Elsevier Science Publishers B. V., 1991. Submitted in January 1991.)

Jürgen Schmidhuber

Institut für Informatik
Technische Universität München
München, Germany

Abstract: In this paper we introduce design principles for unsupervised detection of regularities (like causal relationships) in temporal sequences. One basic idea is to train an adaptive predictor module to predict future events from past events, and to train an additional *confidence module* to model the reliability of the predictor's predictions. We select system states at those points in time where there are *changes* in prediction reliability, and use them recursively as inputs for higher-level predictors. This can be beneficial for 'adaptive sub-goal generation' as well as for 'conventional' goal-directed (supervised and reinforcement) learning: Systems based on these design principles were successfully tested on tasks where conventional training algorithms for recurrent nets fail. Finally we describe the principles of the first neural sequence 'chunker' which collapses a self-organizing multi-level predictor hierarchy into a single recurrent network.

1 OUTLINE OF THE PAPER

This paper is based on the 'principle of reduced history description': *As long as an adaptive sequence processing dynamic system is able to predict future environmental inputs from previous ones, no additional knowledge can be obtained by observing these inputs in reality. Only unpredicted inputs deserve attention* ([7][4][9]). This paper demonstrates that it can be very efficient to focus on unexpected inputs and ignore expected ones.

First we motivate this work by describing a major problem of 'conventional' learning algorithms for time-varying inputs, namely, the problem of *long* time lags between relevant inputs. Then we introduce a principle for unsupervised detection of causal chains in streams of input events. Short representations of 'presumed causal chains' recursively serve as inputs for 'higher-level' detec-

tors of presumed causal chains, thus contributing to a self-organizing hierarchy of event sequences. It is shown how TD-methods (and a modification of TD-methods called 'reverse TD-methods') can be used for 'building bridges through time' between events that are causally dependent. It is shown how unsupervised causality detection can be beneficial for supervised learning as well as for adaptive sub-goal generation. Finally we briefly describe the principles of a 2-network 'chunking' system which collapses a self-organizing multi-level predictor hierarchy into a single recurrent network.

2 THE PROBLEM: LONG TIME LAGS

In what follows the ith component of a vector $v(t)$ will be called $v_i(t)$.

A training sequence with discrete time steps (called an episode) consists of n ordered pairs $((x(t), d(t)) \in R^n x R^m$, $0 < t \leq n$, each $x(t)$ being called an input event. At time t of an episode a learning system receives $x(t)$ as an input and produces output $y(t) \in R^m$. The goal of the learning system is to minimize

$$E = \frac{1}{2} \sum_t \sum_i (d_i(t) - y_i(t))^2 .$$

In general the learner experiences many different episodes during training. Since the gradient of the error sum over all episodes is equal to the sum of the corresponding gradients, for convenience we renounce on indices for different episodes.

In general the task above requires to memorize past events. Previous approaches to solving this problem employed either gradient descent in recurrent nets [1] [3] [11] [2] [12], 'adaptive critic'-like methods [5] [6], or (more recently) adaptive 'fast weights' [8].

All these approaches have severe limitations when it comes to *long* time lags between relevant input events. This can be seen e.g. with examples from grammar learning:

With a given grammar G, the task of a learning system may be to observe a string of terminals, one at a time, and finally decide whether the string is generated by G or not. To train the system, an additional grammar T is used to generate terminal strings (examples and counter-examples for strings produced by G) which become visible to the learner during training. T serves to define the environment of the learning system.

In what follows, capitals like A and B denote non-terminals, and non-capitals like $a, x, b_1, \ldots, b_{100}$ denote terminals. A is always used as the start symbol. A simple regular grammar G_1 (which produces only one sentence but is sufficient to illustrate the basic problem) is given by

$$A \rightarrow aB, B \rightarrow b_1 b_2 b_3 \ldots b_{100}.$$

Let the grammar for training examples T_1 be defined by

$$A \rightarrow aB, A \rightarrow xB, B \rightarrow b_1 b_2 b_3 \ldots b_{100}.$$

T_1 generates only two training examples, namely $ab_1b_2b_3 \ldots b_{100}$ and $xb_1b_2b_3 \ldots b_{100}$. With a conventional algorithm as described in [3] [12] it seems to be practically impossible to learn to accept the first (legal) string and to reject the second (illegal) string (see the subsection on experiments below). The problem is the transport of error information 'back into time' for a comparatively *large* number of time steps (100 in this case).

We can make the task for the learning system *easier* by replacing G_1 by G_2

$$A \rightarrow aB, B \rightarrow b_1b_2b_3 \ldots b_{100}, B \rightarrow b_{100}$$

and by replacing T_1 by T_2:

$$A \rightarrow aB, A \rightarrow xB, B \rightarrow b_1b_2b_3 \ldots b_{100}, B \rightarrow b_{100}.$$

Now a conventional algorithm can learn from *short* training examples (ab_{100} and xb_{100}) that the occurrence of a or x is significant and should be memorized (by means of recurrent connections). From the short training sequences the algorithm can 'generalize' to difficult sequences like $xb_1b_2 \ldots b_{100}$. All examples of grammar learning known to the author seem to work only because the grammar to be learned is comparatively 'easy' in the sense that it generates helpful short training sequences.

In general there will not be any helpful short training sequences. In what follows we will describe a system which tries to decompose long time sequences into blocks of shorter sequences which, in a certain sense, 'belong together'. For instance, with T_1 as above the sequence $b_1 \ldots b_{100}$ 'belongs together'.

We define *presumed causal chains* relative to the current state of a learning system: A presumed causal chain is a sequence of events where each event is predictable from the current internal state of the learner which has seen previous events. For convenience we define a single event to be a presumed causal chain, too.

We will be interested in *maximal presumed causal chains*: A maximal presumed causal chain is a presumed causal chain whose event sequence is not contained in a longer presumed causal chain.

Although this paper does not provide a solution to all problems of temporal structure finding, we can demonstrate that certain cases of unsupervised detection of causal chains can be beneficial for goal-directed learning.

3 THE METHOD: UNSUPERVISED CAUSALITY DETECTION

3.1 THE BASIC MODULES: PREDICTOR AND CONFIDENCE NET

The first of our two basic network modules is a 'conventional' predictor network P. At time t, P receives $x(t)$ as input and produces $p(t)$ as n-dimensional output

($\mid p(t) \mid = \mid x(t+1) \mid$). After the output has been generated, the vector of all activations of all units in P is now called $P(t)$. If P is recurrent, then $P(t)$ and $p(t)$ can depend on times $< t$. The contribution of time t to P's error function is

$$E_P = \frac{1}{2}\sum_i (p_i(t) - x_i(t+1))^2.$$

The second basic module is a 'confidence network' C whose input at time t is $P(t)$ and whose output is called $c(t)$. $c(t)$ is interpreted as a measure of the system's *confidence in its own predictions*. We consider two variations.

Variation 1: $\mid c(t) \mid = 1$. C's error-function is

$$E_C = \frac{1}{2}\sum_t (d(t) - c(t))^2,$$

where $d(t)$ is 1 if $p(t)$ matches $x(t+1)$ (within a certain tolerance), and 0 otherwise.

Variation 2: $\mid c(t) \mid = \mid p(t) \mid$. C's error-function is

$$E_C = \frac{1}{2}\sum_t \sum_i (d_i(t) - c_i(t))^2,$$

where $d_i(t)$ is 1 if $p_i(t)$ matches $x_i(t+1)$, and 0 otherwise.

With variation 1, C's one-dimensional output is trained to be high (interpreted as high confidence) for situations where the predictor usually works and to be low (interpreted as low confidence) for situations where the predictor usually fails. Variation 2 is similar, however, confidence is selective with respect to parts of the predictions.

If P is recurrent, note that C need not be so. The non-input units of P provide a potential for making representations of the past available to C.

After some training, changes in predictability can be recognized by observing *the temporal derivative of* $c(t)$. In what follows we will limit ourselves to variation 1 and the special case that C's output changes from a value below 1 to the value of 1. (A forthcoming paper will be concerned with the more general case).

We use C for building 'bridges through time' and for selecting relevant points in time. After some training in a given environment, whenever P produces a sequence of predictions $p(t_0), p(t_0+1), \ldots, p(t_0+k)$ such that

$$c(t_0) = c(t_0+1) = \ldots = c(t_0+k) = 1,$$

this is interpreted as the recognition of a causal chain which is already known to the system. (In practical applications, it will suffice if C's outputs are 'close to' 1.) The causal chain can be represented *in an abbreviating manner* by the state which marks the beginning of the period with high confidence, namely $P(t_0)$.

The information contained in $P(t_0+1), P(t_0+2), \ldots, P(t_0+k)$ is *redundant*, because it can be *derived* by P from $P(t_0)$ (assuming that C is reliable).

Note that if we had no confidence net, but just considered the changes in prediction error observed over time, we would not be able to distinguish between 'good' predictions and predictions that were correct just by chance. This justifies the existence of C.

3.2 A CONFIDENCE NET FOR THE PAST

With T as above (1) both event sequences $ab_1b_2b_3 \ldots b_{100}$ and $xb_1b_2b_3 \ldots b_{100}$ are causal chains (in the sense of the last subsection). Both event sequences overlap in the sense that they have a common suffix of length 100. Just by looking at b_{25} we cannot tell whether the corresponding chain started with a or x. The second token certainly was b_1, however. When decomposing event sequences we sometimes will be interested in such certain information. This can be done by a system analogous to the one above: We use a network PB which learns to look back into time by being trained at each time step to produce the input of the last time step. A network CB is trained analogously to C above: It learns to model the reliability of PB's 'backward predictions'. Whenever CB's output changes from a value below 1 to the value of 1, this indicates the beginning of a presumed causal chain which does not have overlaps with other presumed causal chains.

3.3 INTRODUCING HIGHER-LEVEL PREDICTOR HIERARCHIES

This section is concerned with using abbreviating representations of causal chains as inputs for predictions on a 'higher level', to cover longer and longer time spans.

In addition to P and C we introduce a third (in general recurrent) network P_1. P_1's input units is the set of units in P. P_1's activation vector, however, is updated only at those time steps t where $c(t) \neq 1$, since steps t with $c(t) = 1$ indicate the presence of a causal chain. Thus a new time scale for P_1 is defined with the help of C (in general, P_1 is updated rarely compared to P which is updated at every time step).

Now we may introduce a 'higher-level' confidence network C_1 which is trained to model the reliability of P_1 just like C models the reliability of P. This allows us to use a next-level predictor P_2 which receives representations of 'presumed causal chains of presumed causal chains' as input, and so on.

The motivation for predictor hierarchies is to keep credit assignment paths short: With a given level k , at each 'kth-order time step' credit assignment is performed only for a small number of kth-order time steps. With increasing level number k, the corresponding time steps on the redefined time scales may contain more and more 'primitive time steps', and therefore longer and longer

'credit assignment bridges' through time may be established. Note, however, that at each level a causal chain may consist of the representation of the input at a single primitive time step: There is nothing like a pre-wired length of a kth-order time step. We obtain a *self-organizing temporal hierarchy.*

The scheme described above makes sense only in environments where event sequences are structured in a hierarchical fashion. One assumption is that events which are nearby in time are stronger correlated than events which are separated by long temporal distances. One can construct environments where this assumption is not true. But, with many 'real-world' tasks the assumption seems to be realistic. Below we describe a grammar learning experiment where a system based on the ideas above performs much better than a conventional approach.

3.4 USING TD-LIKE-METHODS FOR TRANSPORTING CAUSALITY INFORMATION

We can use an additional network E with output $e(t)$ at time t (with $\mid e(t) \mid = \mid p(t) \mid$) for associating the states of each time step of a presumed causal chain $P(t_0), P(t_0+1), \ldots, P(t_0+k)$ with a prediction of its end $P(t_0+k)$. This can be done with 'Temporal Difference'(TD-)methods [10]: If $c(t) = 1$ then the contribution of time t to E's error is

$$\sum_i (e_i(t) - e_i(t+1))^2.$$

Otherwise this contribution is

$$\sum_i (e_i(t) - p_i(t))^2.$$

Similarily, we can use an additional network F whose output $f(t)$ at time t (with $\mid f(t) \mid = \mid PB(t) \mid$, where $PB(t)$ is the output at time t of the backward predictor from section 2.2) for associating the states of each time step of a maximal presumed causal chain (which does not overlap with other such chains) $P(t_0), P(t_0+1), \ldots, P(t_0+k)$ with a backward-prediction of its beginning $P(t_0)$. This can be done with 'Reverse TD-Methods' (called RTD-Methods from now on): If the output $cb(t)$ of CB at time t is $cb(t) = 1$ then the contribution of time t to F's error function is

$$\sum_i (f_i(t) - f_i(t-1))^2.$$

Otherwise this contribution is

$$\sum_i (f_i(t) - P_i(t))^2.$$

Note that RTD-Methods can transport information much faster than TD-methods: With TD-methods we need to repeat a training episode n times for

'pushing an expectation back into time' for n time steps. With RTD-methods we are only going 'forward in time': In principle, one episode may be enough for transporting information about the beginning of a presumed causal chain to its end.

4 ALTERNATIVE EMBEDDINGS OF THE BASIC MODULES IN GOAL-DIRECTED LEARNING SYSTEMS

In this section we describe various ways of embedding predictor and confidence modules in goal directed (supervised or reinforcement learning) systems.

4.1 USING BLOCK REPRESENTATIONS AS INPUT FOR SUPERVISED LEARNERS

With a given predictor hierarchy, at a given time step we simply take the representation of each currently active presumed causal chain (one for each level of the hierarchy) as an input for a supervised learning (in general recurrent) network M. A conventional learning algorithm (e.g. [3]) can be used for training M to produce externally given target values. Below we use this method for grammar learning.

4.2 COMBINING CAUSALITY DETECTION AND ADAPTIVE SUB-GOALING

A causality detector like the one mentioned above can provide names for subprograms (beginnings and ends of causal chains) for an adaptive sub-goal generating system as described in [7].

It should be noted that for typical goal-directed learners future research needs to explore additional ways for providing 'relevant points in time'. Simple unsupervised causality detection often will not be the only thing to do: With typical goal-directed learning tasks 'relevant points in time' are highly dependent on the current goals. Nonetheless, causality detection can ease certain goal-directed learning tasks, as will be seen next.

4.3 AN EXPERIMENT: LEARNING A SIMPLE TASK WHERE CONVENTIONAL RECURRENT NETS FAIL

To illustrate the advantages of unsupervised causality detection, Josef Hochreiter (a student at TUM) tested variants of the scheme on the grammar learning task of section 2 (based on G_1 and T_1) and compared the results to the results obtained with the IID-algorithm for fully recurrent continually running networks

[3][12]. Local representations of terminal input symbols were employed: There were as many input units as there were terminal symbols, and each terminal was represented by a bit-vector with only one non-zero component. No episode boundaries were used: strings generated by the training grammar were fed to the learning systems without providing information about their beginnings and their ends.

With various numbers of hidden units and various learning rates it was not possible to obtain significant performance improvement with the conventional algorithm (the test runs were interrupted after 100000 training examples). This indicates that time lags of as few as 100 time steps are already too much for methods based on 'back-propagation through time' (provided that there are no short helpful training sequences).

With unsupervised causality detection only a few hundred training examples were necessary to solve the task. Since the task was comparatively simple, only one level of the predictor hierarchy was needed. None of the involved networks P, C and M needed any hidden units, all learning rates were equal to 1.0. (The TD-variant and the RTD-variant were also successfully tested.)

Additional successful experiments with more complicated grammars were conducted. These will be reported at the conference.

5 COLLAPSING A SELF-ORGANIZING PREDICTOR HIERARCHY INTO A SINGLE RECURRENT NET

This section briefly describes a 2-network 'chunking' system whose details are provided in [9]. The system collapses a self-organizing multi-level predictor hierarchy into a single recurrent network. One term of the first net's error function forces it to behave like a conventional supervised learning dynamic recurrent network. Another term of its error function forces it to predict its next input. Only if it makes an error, its current state plus the unpredicted input is transferred to the second net, where it contributes to a higher level internal representation of the input history. Note that the second net receives all necessary information about the input history: The information which is deducable by means of the predictions of the first net is redundant.

Let us now assume that the second net in certain situations learns to predict the next critical state and input of the first net (or to generate the externally defined desired output). Given this assumption, the second net will develop useful internal representations of previous unexpected input events. The final term of the first net's error function forces it to reproduce these internal representations, *by predicting the internal states of the second net.* This means that the first net will learn to create useful internal representations on its own. After some time, it will be able to use its own internal representations for making fewer and fewer

errors. Therefore, the second net will receive fewer and fewer inputs and will be able to learn to build longer and longer 'bridges through time', etc. Ideally, in deterministic environments the second net will become obsolete after some time.

The chunking system has already been tested on tasks where conventional recurrent nets fail [9].

References

[1] M. I. Jordan. Serial order: A parallel distributed processing approach. Technical Report ICS Report 8604, Institute for Cognitive Science, University of California, San Diego, 1986.

[2] B. A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269, 1989.

[3] A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, 1987.

[4] J. H. Schmidhuber. Dynamische neuronale Netze und das fundamentale raumzeitliche Lernproblem. Dissertation, Institut für Informatik, Technische Universität München, 1990.

[5] J. H. Schmidhuber. A local learning algorithm for dynamic feedforward and recurrent networks. *Connection Science*, 1(4):403–412, 1990.

[6] J. H. Schmidhuber. Recurrent networks adjusted by adaptive critics. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, Washington, D. C.*, volume 1, pages 719–722, 1990.

[7] J. H. Schmidhuber. Towards compositional learning with dynamic neural networks. Technical Report FKI-129-90, Institut für Informatik, Technische Universität München, 1990.

[8] J. H. Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. Technical Report FKI, Institut für Informatik, Technische Universität München, March 1991.

[9] J. H. Schmidhuber. A neural sequence chunker. Technical Report FKI, Institut für Informatik, Technische Universität München, 1991.

[10] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[11] R. J. Williams. Toward a theory of reinforcement-learning connectionist systems. Technical Report NU-CCS-88-3, College of Comp. Sci., Northeastern University, Boston, MA, 1988.

[12] R. J. Williams and D. Zipser. Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, 1(1):87–111, 1989.