CONTINUOUS HISTORY COMPRESSION

In H. Hüning, S. Neuhauser, M. Raus, and W. Ritschel, editors, Proc. of Intl. Workshop on Neural Networks, RWTH Aachen, pages 87-95. Augustinus, 1993.

Jürgen H. Schmidhuber
Michael C. Mozer[*]
Daniel Prelinger
Institut für Informatik,
Technische Universität München

Abstract

Neural networks have proven poor at learning the structure in complex and extended temporal sequences in which contingencies among elements can span long time lags. The principle of history compression [18] provides a means of transforming long sequences with redundant information into equivalent shorter sequences; the shorter sequences are more easily manipulated and learned by neural networks. The principle states that expected sequence elements can be removed from the sequence to form an equivalent, more compact sequence without loss of information. The principle was embodied in a neural net predictive architecture that attempted to anticipate the next element of a sequence given the previous elements. If the prediction was accurate, the next element was discarded; otherwise, it was passed on to a second network that processed the sequence in some fashion (e.g., recognition, classification, autoencoding, etc.). As originally proposed, a binary judgement was made as to the predictability of each element. Here, we describe a *contininuous* version of history compression in which elements are discarded in a graded fashion dependent on their predictability, embodied by their (Shannon) information. We implement continuous history compression using a RAAM architecture, yielding a class of sequence learning algorithms that are both entirely local and still able to bridge long time lags between correlated events.

1 INTRODUCTION

Neural networks for supervised sequence learning have been successful at solving certain interesting sequence learning tasks. However, they have proven poor at learning the structure in extended sequences in which contingencies among elements can span long time lags. This paper provides a technique that can help to greatly improve the performance of sequence processing neural networks on certain sequence learning tasks involving long time lags.

The Problem. A training sequence p with n_p discrete time steps consists of n_p ordered pairs of real-valued vectors $(x^p(t), d^p(t))$, $0 < t \leq n_p$. At time $t > 0$ of sequence p a learning system receives $x^p(t)$ as an input and produces the output $y^p(t)$. The goal is for the output produced by the system to match the desired output. The learning system tries to achieve this by minimizing

$$E = \frac{1}{2} \sum_p \sum_{t>0} \sum_i (d_i^p(t) - y_i^p(t))^2.$$

[*]Department of Computer Science, University of Colorado, Boulder, CO 80309, USA

Here, as well as in the remainder of this paper, the ith component of some vector v is denoted by v_i.

A typical task that fits this framework is grammar learning. There the goal is to predict a set of possible next symbols in a string (generated by an initially unknown grammar) from previous symbols. Other typical tasks include trajectory learning for robot control as well as all kinds of sequence classification problems (like those in speech recognition).

Ideally, we would like to have a *local* learning algorithm for learning such tasks from training examples. A learning algorithm for dynamic neural networks is *local* if, for arbitrary durations of sequences to be learned and for arbitrary network sizes (measured in number of connections), the peak computation and storage complexity per connection and time step is constant.

The most powerful conventional sequence learning algorithms are non-local, however. The following section addresses a weakness of conventional (local and non-local) algorithms. Later we will present a novel *local* algorithm that can solve certain tasks that cannot be solved even by existing non-local algorithms.

1.1 A WEAKNESS OF PREVIOUS LEARNING ALGORITHMS

Exact gradient-based supervised learning algorithms for minimizing E are back-propagation through time (BPTT), e.g. [14][24][10], the real time recurrent learning algorithm (RTRL) [13][27], its accelerated versions [26][28][17], and the recent fast-weight algorithm [19]. All these approaches are non-local – for a restricted class of recurrent networks, however, there is a local gradient-based algorithm [6].

Local (but much weaker) approximations of the general supervised algorithms have been proposed (e.g. [3][1]). Local approaches to *reinforcement* learning in recurrent networks [25][15][5] unfortunately are not very practicable in realistic applications.

Although non-local gradient-based recurrent nets are general and can sometimes learn to perform quite complicated algorithms, they tend to fail when it comes to long time lags (see e.g. [18] and [8]):

Suppose you want your learning algorithm to make the distinction between two possible input sequences: $a, x_1, x_2, \ldots, x_{10}$ and $b, x_1, x_2, \ldots, x_{10}$. The distinction is to be made by switching on a particular output unit at the end of the first sequence, and switching it off at the end of the second sequence. Local supervised learning recurrent networks in the style of [1], receiving one symbol at a time, have no chance to solve this apparently simple task. But even the non-local recurrent networks have the greatest difficulties. If not carefully designed, they hardly learn to bridge the 10 step time lags between the end of the sequences and the discriminating events a and b, respectively. [2] provides some theoretical analysis why this is the case. The essential problem is: In unstructured nets, error signals tend to disperse the further they have to go 'back in time'.

1.2 HISTORY COMPRESSION

The principle of history compression [18][20] can help to overcome problems like the one with extended sequences mentioned in the last subsection. This principle essentially states that only unexpected events (including representations of the time steps at which they occurred) carry non-redundant information. Based on this insight, long sequences containing redundant information can be transformed into much shorter sequences *without loss of information*.

The principle was embodied in a neural net predictive architecture that attempted to anticipate the next input vector given the previous input vectors [18]. If the prediction was accurate, the next input vector was discarded; otherwise, it was passed on to a second network (working on a self-organizing slower time scale) that processed the sequence in some fashion (e.g., recognition, classification, autoencoding, etc.). In simulations with certain grammars involving long time lags, supervised E-minimization became an easy job for the second network. In some cases it was possible to obtain speed-up factors of more than 1,000 over conventional learning algorithms [18] [21].

The history compression technique formulated in [18], however, suffers from the weakness that a binary judgement is made as to the predictability of each input vector. Expectation-mismatches are defined in an all-or-none fashion: Each input pattern that is not predictable at a certain time gives rise to an unexpected event. Only unexpected events provoke updates of the internal state of the higher-level sequence processing network whose input is the reduced sequence description. There is no concept of a partial mismatch or of a 'near-miss'. There is no possibility of updating the higher-level net 'just a little bit' in response to a 'nearly expected input'.

2 CONTINUOUS HISTORY COMPRESSION

In response to this weakness, *continuous history compression* is based on update procedures that let highly informative events have a stronger influence on the history representation than less informative (more likely) events. Input events are discarded in a graded fashion dependent on their predictability, embodied by their (Shannon) information.

We introduce a deterministic discrete time predictor whose state at time t of sequence p is described by an environmental input vector $x^p(t)$, an internal state vector $h^p(t)$, and an output vector $z^p(t)$. For now, assume $x^p(t)$ is a binary vector whose components are all 0 except for one which is 1 – local input representation. The environment may be nondeterministic. At time 0 of each sequence p, the predictor starts with a default input $x^p(0)$ and an initial internal start state $h^p(0)$. At time $t \geq 0$, the predictor computes

$$z^p(t) = f(x^p(t), h^p(t)).$$

At time $t > 0$, the predictor furthermore computes

$$h^p(t) = g(x^p(t), h^p(t-1)).$$

By normalization, the components of $z^p(t)$ are forced to sum to 1 and are interpreted as a prediction of the probability distribution of the possible $x^p(t+1)$: $z^p_j(t)$ is interpreted as the prediction of the probability that $x^p_j(t+1)$ is 1.

The output entropy

$$-\sum_j z^p_j(t) log\ z^p_j(t)$$

can be interpreted as a measure of the predictor's confidence.

How much information is conveyed by $x^p(t+1)$ (relative to the current predictor), once it is observed? According to [22] it is

$$-log\ z^p_j(t)$$

with j chosen such that $x^p_j(t+1) = 1$.

In section 3 we will define an update procedure where the 'strength' of an update in response to a more or less unexpected event will be a monotonically increasing function of the information the event conveys.

2.1 HEURISTIC GENERALIZATION

Using local input representations as above, it is easy to measure the information that an event conveys. In the case of distributed, real-valued inputs this is not possible.

The simple heuristic proposed in this subsection is to make the 'strength' of an update in response to a more or less unexpected event a monotonically increasing function of the current error of the predictor who predicted the event. A similar method has been independently suggested by Don Mathis (personal communication). A tacit and not always justified assumption behind this heuristic is that two similar inputs—quantified by their distance in Euclidean space—have similar 'semantics'.

3 CONTINUOUS HISTORY COMPRESSION WITH RAAMs

One way to implement continuous history compression involves Pollack's recursive auto-associative memories (RAAMs) [11]. This section first explains RAAMs and demonstrates that local supervised learning algorithms based on RAAMs may *theoretically* bridge arbitrary time lags between correlated events (modulo crosstalk). Experiments are then mentioned that show that in *practical* applications time lags ought not to be longer than a few time steps. Finally we describe an architecture that combines RAAM's and continuous history compression, yielding a system capable of uniquely representing much longer sequences.

An interesting aspect of the method described in this section is that, unlike general methods for performing gradient descent through time, it allows for an entirely local algorithm for learning arbitrary sequences.

3.1 INTRODUCTION TO 'SEQUENTIAL RAAMs'

A sequential RAAM is an auto-encoder network A whose only goal is to create different internal representations in response to different input sequences.

A is a 3 layer, strictly layered, fully interconnected feedforward net with $n_I + n_H$ input units, n_H hidden units, and $n_I + n_H$ output units.

A's internal state $h^p(t)$ is the activation vector of the hidden units at time step t of sequence p. A's input at time t is $h^p(t-1) \circ x^p(t)$. Here, as well as in the remainder of this paper, $\circ$ denotes the concatenation operator for vectors. The environment may be nondeterministic. For all p, the initial internal state $h^p(0)$ takes on a default value, say 0. At time $0 < t \leq n_p$, A computes

$$h^p(t) = g(x^p(t), h^p(t-1)).$$

Here g is implemented by the conventional activation spreading rules for back-propagation nets [23] [4] [9] [14]. A's output $z^p(t)$ is computed from $h^p(t)$ according to the same rules. $z^p(t)$ is interpreted as a 'reconstruction' of $x^p(t) \circ h^p(t-1)$.

Following [11], we modify g such that $h^p(t), 0 < t < n_p$ takes on a value that allows to reconstruct $h^p(t-1)$ and $x^p(t)$. A's error at time t of sequence p is

$$E_A(t) = \frac{1}{2}(x^p(t) \circ h^p(t-1) - z^p(t))^T(x^p(t) \circ h^p(t-1) - z^p(t)).$$

Minimizing E_A requires just to store $h^p(t-1)$ but not any previous activations. In other words, we get a local algorithm, essentially the one proposed by Pollack [11].

Why should this local learning algorithm theoretically force A to create unique internal states for arbitrary sequences and subsequences? For the sake of the argument, we neglect all problems specific for the particular gradient descent training algorithm (like crosstalk between internal representations):

1. Let us assume that there are s different training sequences $1, \ldots, s$. The length of sequence p is $n_p > 0$. We train $h^p(1)$ to allow the reconstruction of $h^p(0)$ and $x^p(1)$ for all $p = 1, \ldots, s$. Therefore the beginnings of all sequences will be uniquely represented in h.

2. Now let us assume that all sequences and subsequences with lengths $< k$ cause unique representations in h. For all sequences and subsequences p with length k we train $h^p(k)$ to allow the reconstruction of $h^p(k-1)$ and $x^p(k)$. Therefore all sequences and subsequences with length $< k+1$ will cause unique representations in h. $\Box$

Of course, it may be difficult to find a set of weights in practice that satisfy the reconstruction criteria for all $h^p(k)$.

Note that any reinforcement or supervised learning system can receive the unique sequence representations (across the hidden units of A) as an input and can be trained to do whatever its task is[1].

[1]For instance, a supervised feed-forward net can be trained to emit desired outputs. A reinforcement learner with a non-Markovian interface to its environment [16] will be potentially able to build a Markovian interface using RAAMs.

3.2 RAAMs FAIL WITH LONG TIME LAGS

We conducted experiments similar to the one described in [18] which showed that RAAMs usually fail to create sufficiently distinct representations of sequences with lengths of the order of as few as 10 time steps [12]. The following section shows how to extend the capabilities of sequential RAAMs.

3.3 RAAMs AND CONTINUOUS HISTORY COMPRESSION

By combining continuous history compression, RAAMs, and an update rule previously proposed by Mozer in [7], we can greatly extend the capabilities of sequential RAAMs as follows. We need two modules. The first module is a RAAM A which corresponds to the architecture described in section 3.1. A is going to encode reduced sequence descriptions in its hidden units. The second module is a predictor P which is going to determine the information content of a particular input vector in order to determine the degree to which it should be passed to A.

The basic strategy is: At a given time, P tries to model the (context dependent) probability distribution of the possible next inputs from the previous input and the previous activation vector of A's hidden units. As long as the next input turns out to be (nearly) predictable (in the sense that P predicts a high context-dependent probability) the inputs and targets of A remain (nearly) invariant. In this case the reduced sequence description across A's hidden units hardly changes. Only the highly unexpected events (those with low probability) cause strongly different targets for A, thus forcing it to incorporate the unexpected events into its internal representation.

The strength of an update of A in response to a new input is modulated by a function τ of an estimation of the information conveyed by the input. This estimation is obtained by looking up the probability of the input event as predicted by P. τ returns minimal values (near zero) for arguments representing low information. τ returns maximal values for arguments representing high information.

Here are the details. A has n_H hidden units, $n_H + n_I$ input units and $n_H + n_I$ output units. At time t of sequence p, the input of A is $H^p(t) \circ X^p(t)$. A's internal state vector across the bottleneck of hidden units is called $h^p(t)$. A's $n_H + n_I$-dimensional output vector is called $A^p(t)$. At every time step $t > 0$, A is trained to match its input, using back-propagation. The n_H-dimensional vector $H^p(t)$ (called the 'reduced state' for reasons to become obvious below) takes on a default value (e.g. the zero vector) for $t = 0$. The same holds for the n_I-dimensional vector $X^p(t)$.

At time t of sequence p, the feed-forward predictor P receives $H^p(t) \circ x^p(t)$ as an input vector[2]. Its n_I dimensional output vector $P^p(t)$ is trained to match the probability distribution of the possible $x^p(t+1)$, using back-propagation and normalization of P's output units. For $t > 1$, $H^p(t)$ and $X^p(t)$ are defined by

$$H_i^p(t) = (1 - \tau^p(t))H_i^p(t-1) + \tau^p(t)h_i^p(t-1)$$

and

$$X_i^p(t) = (1 - \tau^p(t))X_i^p(t-1) + \tau^p(t)x_i^p(t),$$

where $\tau^p(t)$ is a monotonically increasing function of $-log\ P_j^p(t-1)$, e.g. $\tau^p(t) = 1 - P_j^p(t-1)$, where j is chosen such that $P_j^p(t-1)$ is the predicted probability of the observed input pattern $x^p(t)$.

By setting $\tau^p(t) = 0$ if $P^p(t-1) = x^p(t)$ and 1 otherwise, we obtain a 'conventional' history compression algorithm.

Unique sequence representations created by our method can be used as inputs to a *feed-forward* supervised learner minimizing E.

[2]In theory it would be necessary to include a unique representation of the time step at which the unexpected event occurred [18]. For simplicity, we will omit such unique time representations.

4 EXPERIMENTS

In the experiments described herein P's outputs were not even normalized, and $\tau^p(t)$ was simply taken to be a monotonically increasing function of P's prediction error (see section 2.1). The good results justified this heuristic simplification.

We tested the system on a task described in [18]. This task essentially requires to find unique representations of two different 20 time step sequences. The only discriminating difference between the two was the input at the first time step. Therefore the entirely local system had to learn to bridge a 20 step time lag.

Details: There were 22 possible input symbols $x, y, b_1, b_2, \ldots, b_{20}$. The system observed one input symbol at a time. Two possible input sequences, $xb_1 \ldots b_{20}$ and $yb_1 \ldots b_{20}$, were presented in random order. Note that in general it was not possible to predict the first symbol of each sequence, due to the random occurrence of x and a. No sequence boundaries were used: Input sequences were fed to the learning systems without providing information about their beginnings and their ends. Therefore there was a continuous stream of input events. The task was considered to be solved if the euclidean distance between the internal representations at the sequence endings exceeded 0.5.

All non-input units employed the logistic activation function $f(x) = \frac{1}{1+e^{-x}}$. Weights were initialized between -0.3 and 0.3. Local input representations of 22 possible input symbols $a, x, b_1, \ldots, b_{20}$ were employed: Each symbol was represented by a bit-vector with only one non-zero component. For both the predictor and the RAAM there was an additional input unit with constant activation 1 for providing a modifiable bias for the non-input units. *No* unique time step representations [18] were necessary to solve this task. All learning rates were equal to 1.0.

Conventional recurrent nets consistently failed to solve an analoguous task within less than 1,000,000 training sequences. The 2-net chunker in [18] usually solved the task after the presentation of a few thousand training sequences. However, in some cases it needed more than 30,000 sequences[3].

The system described above consistently needed less than 600 training sequences.

Therefore, at least sometimes we can get the best of two worlds: Fast learning *and* an entirely local learning algorithm.

5 FURTHER EXTENSIONS

Noise is usually unexpected, otherwise it would not be noise. Therefore, noise conveys information, in the sense of Shannon. This does not fit the human conception of information, however. In the context of the problems and methods above, how can we discover noise and get rid of it?

An event $x^p(t)$ can be considered as noise with respect to sequence p if the current input, $x^p(t)$, does not provide information about the rest of the sequence and therefore ought to be ignored:

$$P(\{x^p(k), k > t\} | \{x^p(k), k < t\}) = P(\{x^p(k), k > t\} | \{x^p(k), k \leq t\}).$$

However, to measure and compare these two probabilities requires to know a unique representation of the future at time t.

This can be achieved with an additional chunking system seeing each sequence in reverse order ('running backward in time'). Two predictors can be trained to predict the internal state of the 'reverse chunker' at time $t+1$, one of them seeing only the current state of the 'forward chunker', the other one also seeing the current input at time t. Comparing both predictions, one can make statements about the information conveyed by $x^p(t)$. This can serve as the basis for calculating the strength of the update of a 'super-chunker' that learns to ignore unexpected noise. These ideas have been implemented and successfully tested with small examples [12]. The limitations of the approach, however, are not clear yet.

[3]A multi-level system (also described in [18]) needed just a few 100 training sequences to solve the analoguous task – the final sequence representation, however, required two networks instead of a single network.

6 ACKNOWLEDGEMENTS

This research was supported in part by a DFG fellowship to J. Schmidhuber, as well as by NSF PYI award IRI–9058450, grant 90–21 from the James S. McDonnell Foundation, and DEC external research grant 1250 to Michael C. Mozer.

References

[1] J. L. Elman. Finding structure in time. Technical Report CRL Technical Report 8801, Center for Research in Language, University of California, San Diego, 1988.

[2] J. Hochreiter. Diploma thesis, 1991. Institut für Informatik, Technische Universität München.

[3] 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.

[4] Y. LeCun. Une procédure d'apprentissage pour réseau à seuil asymétrique. *Proceedings of Cognitiva 85, Paris*, pages 599–604, 1985.

[5] G. Lukes. Review of Schmidhuber's paper 'Recurrent networks adjusted by adaptive critics'. *Neural Network Reviews*, 4(1):41–42, 1990.

[6] M. C. Mozer. A focused back-propagation algorithm for temporal sequence recognition. *Complex Systems*, 3:349–381, 1989.

[7] M. C. Mozer. Connectionist music composition based on melodic, stylistic, and psychophysical constraints. Technical Report CU-CS-495-90, University of Colorado at Boulder, 1990.

[8] M. C. Mozer. Induction of multiscale temporal structure. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 4,*, pages 275–282. San Mateo, CA: Morgan Kaufmann, 1992.

[9] D. B. Parker. Learning-logic. Technical Report TR-47, Center for Comp. Research in Economics and Management Sci., MIT, 1985.

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

[11] J. B. Pollack. Recursive distributed representation. *Artificial Intelligence*, 46:77–105, 1990.

[12] D. Prelinger. Diploma thesis, 1992. Institut für Informatik, Technische Universität München.

[13] A. J. Robinson and F. Fallside. Static and dynamic error propagation networks with application to speech coding. *Proceedings of Neural Information Processing Systems, American Institute of Physics*, 1987.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, 1986.

[15] J. H. Schmidhuber. A local learning algorithm for dynamic feedforward and recurrent networks. *Connection Science*, 1(4):403–412, 1989.

[16] J. H. Schmidhuber. Reinforcement learning in markovian and non-markovian environments. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 500–506. San Mateo, CA: Morgan Kaufmann, 1991.

[17] J. H. Schmidhuber. A fixed size storage $O(n^3)$ time complexity learning algorithm for fully recurrent continually running networks. *Neural Computation*, 4(2):243–248, 1992.

[18] J. H. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[19] J. H. Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. *Neural Computation*, 4(1):131–139, 1992.

[20] J. H. Schmidhuber. Learning unambiguous reduced sequence descriptions. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4*, pages 291–298. San Mateo, CA: Morgan Kaufmann, 1992.

[21] J. H. Schmidhuber. Netzwerkarchitekturen, Zielfunktionen und Kettenregel. Habilitationsschrift, Institut für Informatik, Technische Universität München, 1993. In preparation.

[22] C. E. Shannon. A mathematical theory of communication (parts I and II). *Bell System Technical Journal*, XXVII:379–423, 1948.

[23] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[24] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1, 1988.

[25] 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.

[26] R. J. Williams. Complexity of exact gradient computation algorithms for recurrent neural networks. Technical Report Technical Report NU-CCS-89-27, Boston: Northeastern University, College of Computer Science, 1989.

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

[28] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In *Back-propagation: Theory, Architectures and Applications.* Hillsdale, NJ: Erlbaum, 1992, in press.