E	E/L	$\hat{F}$	K	L	$\hat{V}$	R_{set}	H	$\hat{Q}$
111.7	0.14	223.3	8	800	27.92	96.4	45.8	11.2
246.7	0.206	493.3	12	1200	41.11	97.4	45.1	24.1
439.3	0.275	878.7	16	1600	54.92	97.0	45.2	43.8
698.3	0.349	1396.7	20	2000	69.83	97.3	45.9	70.4
971.7	0.405	1943.3	24	2400	80.97	97.3	44.7	96.8
1309.3	0.468	2618.7	28	2800	93.52	97.2	44.3	130.9
1719.7	0.537	3439.3	32	3200	107.48	97.0	44.6	177.3
2151.7	0.598	4303.3	36	3600	119.54	96.9	44.2	212.7
2671.3	0.668	5342.7	40	4000	133.57	97.0	44.5	268.6

Table 2: *Capacity results for the correlated patterns (CSS) case.* $\hat{Q}$ *is the average instances of each state, across entire set of episodes.*

example—can be represented as sets of CSSs over a finite alphabet. Notice that in the last simulation ($K = 40$) of table 2, the average number of instances of each state is 268.6. More importantly, the trend in $\hat{Q}$ is at least linear in K.

Furthermore, TEMECOR requires only a single presentation of each episode. It is also the case that since synaptic weights do not decay, the memory traces of the episodes remain stable up to the point at which weight saturation effects lead to intrusion errors. Thus, even if a particular word is not accessed for an arbitrarily long period during which all the other words are accessed frequently, that word's trace will still read out perfectly when it finally is re-accessed. In contrast, models based on back-propagation have been shown to be subject to massive ("catastrophic") forgetting (McCloskey and Cohen, 1989) in which newly encountered patterns obliterate old memory traces.

Space does not permit a more detailed discussion of TEMECOR's properties and capabilities. However, numerous other parametric studies which have been made reveal that it is quite robust. Of particular note are: a) TEMECOR functions qualitatively similarly under less-than-full horizontal connectivity, and b) it allows for variation in both the number of timeslices per episode (T) and the number of active features per timeslice (S). These issues and many others will be presented in the future.

5 References

1. Cleeremans, A. (1993) *Mechanisms of Implicit Learning: Connectionist Models of Sequence Processing* The MIT Press, Cambridge, MA

2. Elman, J. L. (1990) Finding structure in time. *Cognitive Science*, *14*(2), 179–212.

3. Guyon, I., Personnaz, L., & Dreyfus, G. (1988) Of points and loops. In Eckmiller, R. & Malsburg, C. v. (Eds.) *Neural Computers*, NATO ASI Series, Vol. F41, 261–269. Springer-Verlag, Berlin, Germany.

4. Jordan, M. I. (1986) Serial order. Tech. rep. 8604, Institute for Cognitive Science, University of California, San Diego.

5. McCloskey, M. & Cohen, N. J. (1989) Catastrophic interference in connectionist networks: The sequential learning problem. In Bower, G. H. (Ed.), *The Psychology of Learning and Motivation*, v.24, 109–165. Academic Press.

6. Rinkus, G. (1993) Context-sensitive spatio-temporal memory. Proceedings of World Congress on Neural Networks. The annual meeting of the International Neural Network Society. Lawrence Earlbaum Associates, Inc., V.2. 344–347

7. Rinkus, G. J. (1995) *A Combinatorial Neural Netork Exhibiting both Episodic Memory and Generalization for Spatio-Temporal Patterns.* Ph.D. thesis, Graduate School of Arts and Sciences, Boston University. In progress.

8. Smith, A. W. & Zipser, D. (1989) Learning sequential structure with the real-time recurrent learning algorithm. *International Journal of Neural Systems*, *1*(2), 125–131.

9. Williams, R. J. & Zipser, D. (1989) A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, *1*, 270–280.

E	E/L	$\hat{F}$	K	L	$\hat{V}$	$var(V)$	W_{inc}	R_{set}	H
129.3	0.162	258.7	8	800	32.33	42.78	327465.7	97.8	51.7
290.3	0.242	580.7	12	1200	48.39	58.61	736201.3	96.6	51.6
517.0	0.323	1034.0	16	1600	64.62	78.58	1309367.0	97.2	51.7
793.0	0.397	1586.0	20	2000	79.3	72.69	2020240.0	97.2	51.0
1141.7	0.476	2283.3	24	2400	95.14	86.94	2908397.0	97.1	51.0
1544.7	0.552	3089.3	28	2800	110.33	98.93	3942919.0	97.2	50.8
2002.3	0.626	4004.7	32	3200	125.15	115.21	5122921.3	97.1	50.5
2506.0	0.696	5012.0	36	3600	139.22	177.57	6433038.0	97.1	50.1
3084.0	0.771	6168.0	40	4000	154.2	193.86	7925805.0	97.7	50.0

Table 1: *Results of simulations using uncorrelated patterns. See text for discussion. All Simulations had* $\Theta = 19$, $S = 20$ *and* $T = 10$. *Abbrevs.:* $\hat{F}$ *= ave. number of instances of each feature, across entire set of episodes;* K *= CM size;* L *= total number of L2 cells;* $\hat{V}$ *= ave. number of times each L2 cell is used;* $var(V)$ *= variance of* V*;* W_{inc} *= total number of increased weights;* R_{set} *= recall accuracy over the whole set of episodes; and* H *= percent of horizontal weights increased.*

TEMECOR's exceptional scaling properties—specifically, the number of times an L2 cell can be used, V, increases (apparently very linearly) with the size of the CMs, K.

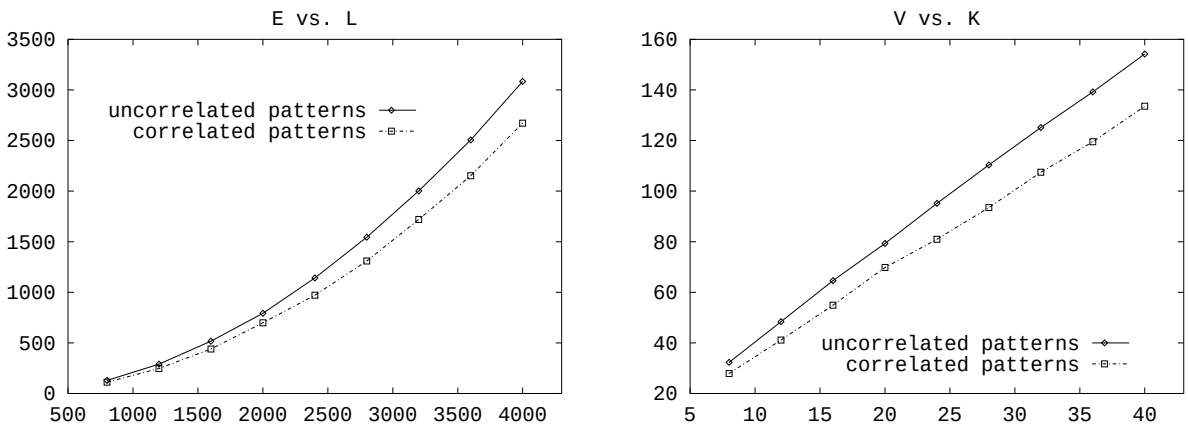

Figure 5: *a) The number of episodes,* E*, that can be stored to criterion accuracy vs. the number of L2 cells,* L*. b) The average number of times an L2 cell is used,* V*, while still maintaining criterion accuracy is linear in CM size,* K*. Solid curves correspond to the uncorrelated pattern case (table 1) and the dotted lines to the correlated (CSS) case (table 2).*

The dotted curves of fig. 5, derived from table 2, show that a slightly slower, although still faster-than-linear, relationship also holds for the case of correlated patterns. The episodes—i.e. complex state sequences—used in the simulations of table 2 were constructed as follows. First, a set (alphabet) of $U = 100$ unique states, each consisting of 20 active features, was built. The timeslices comprising the episodes were then randomly chosen (with replacement) from this alphabet of 100 states.

4 Conclusion

The simulation results, for both uncorrelated and correlated patterns, show that TEMECOR scales well with problem size. Specifically, in both cases, the number of spatio-temporal patterns (episodes) that can be stored to criterion accuracy increases faster-than-linearly in the number of cells in the network. This finding is especially encouraging in the case of correlated patterns—i.e. complex state sequences (CSSs)—since, as stated in the introduction, linguistic information—i.e. phonemic transcriptions of utterances, for

recall threshold, are necessary in order to prevent cells from becomming active inappropriately. The reader may check that Φ^j is also recalled correctly if $S \geq \Theta \geq 2$. This small example shows that TEMECOR can store a set of two sequences that have a common state ([A X B] and [C X D]). Although neither sequence, by itself, is complex, the set as a whole, is complex.

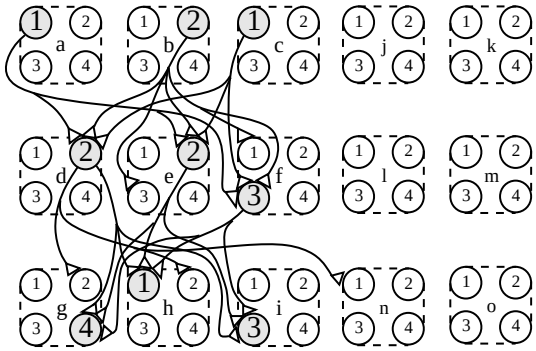

Figure 4: *Recall of* Φ^i *in the case of* $2 \leq \Theta \leq 3$. *If* $\Theta = 1$, *then* n_1 *would become active at* $t = 3$. *If* $\Theta > 3$, *then no recall at all is possible.*

The recall example of fig. 4 assumes that a precise *L2 code*—$\Delta_1^i = \{a_1, b_2, c_1\}$—is provided as a recall prompt. In reality, prompts are necessarily L1 codes; afterall, L1 is the input layer—i.e. the model's interface to the world. Using L2-codes rather than L1-codes as recall prompts is justifiable here because it has very minor bearing on the basic capacity and representational properties of the model, which are the focus of this paper. However, a more complete version of the model, which provides a mechanism whereby episode-initial L1 patterns cause the correct episode-initial L2 codes to become active, is developed in Rinkus (1995, Ph.D. thesis, in prep.).

3 Simulation Results

Table 1 gives the maximal capacity (as well as other statistics) for networks of increasing size, in the case of uncorrelated patterns. Table 2 provides similar information for correlated patterns.

The parameters that are constant for all simulations reported in this paper are: M (the number of L1 cells) = 100 and R_c (the criterion recall accuracy) = 97.0%. Furthermore, all simulations reported in tables 1 and 2 used episodes having $T = 10$ timeslices, with $S = 20$ (out of $M = 100$) active features per timeslice. The recall threshold, Θ, is set to $S - 1 = 19$ for all these simulations. Because the degree of overlap between the L2 codes increases as additional episodes are presented, maximal capacity is achieved by setting Θ as high as possible.[6]

Table 1 was generated in the following way. For each K, the maximal number, E, of episodes which could be stored to criterion accuracy was determined.[7] Recall accuracy, $R(e)$, for a given episode e, is defined as:

$$R(e) = \frac{C(e) - D(e)}{C(e) + I(e)}$$

where $C(e)$ is the number of L2 cells that should become active during recall of e, $D(e)$ is the number of L2 cells which should have become active but did not (i.e. "deletions"), and $I(e)$ is the number of L2 cells which should not have become active but did (i.e. "intrusions"). Recall accuracy for a whole set of episodes, R_{set}, is just the average of the R values. All episodes were presented only once.

Table 1 supports the claim, made in the introduction, that the number of episodes that can be stored to criterion recall accuracy increases faster-than-linearly, at least over the range of network sizes analyzed, in network size. This can also be seen in the solid curves of fig. 5, which are derived from table 1. In particular, fig. 5a depicts E as a function of L. Fig. 5b graphically displays the underlying reason for

[6] Clearly, Θ must be set less than S or else no recall is possible.

[7] Each line (i.e. data point) of all tables represents the average of three simulations with the corresponding parameter set.

separated, in the sense of Hamming distance, L2 codes. L2 codes are denoted with the Greek letter, Δ. The L2 code, Δ^i, corresponding to Φ^i can be written as:

$$\begin{array}{l} \Delta^i_1\text{: } \{a_1, b_2, c_1\} \\ \Delta^i_2\text{: } \{d_2, e_2, f_3\} \\ \Delta^i_3\text{: } \{g_4, h_1, i_3\} \end{array}$$

where the notation, a_1, indicates cell 1 in CM_a.

Fig. 3a also shows the learning that would occur due to presentation of Φ^i. A synapse, w_{xy} is increased to asymptote (i.e. 1) after a single correlation in which cell y is active immediately after cell x. Cell activation levels are {0,1}-valued.

TEMECOR does not require that the set of spatial patterns (timeslices) be orthogonal. Rather, as the challenge we've taken up is to represent sets of CSSs, whole states can recur exactly without presenting a problem to the model. To see this, suppose a second episode, Φ^j, defined as:

$$\begin{array}{lcl} \Phi^j_1\text{: \{a, b, k\}} & & \text{C: \{a, b, k\}} \\ \Phi^j_2\text{: \{d, e, f\}} & \text{or,} & \text{X: \{d, e, f\}} \\ \Phi^j_3\text{: \{g, h, n\}} & & \text{D: \{g, h, n\}} \end{array}$$

is presented to the model. Fig. 3b shows one possible L2 code (and the corresponding learning) that could be chosen for Φ^j. Again, the L2 winners are chosen at random within active CMs. Both episodes have the exact same middle state (i.e. $\Phi^i_2 = \Phi^j_2$) as well as a great deal of featural overlap on the other two timeslices. Nevertheless, the internal representation (L2-code) of that middle state is very different in the two instances. In fact, $\left|\Delta^i_2 \cap \Delta^j_2\right| = 1$. This suggests that we can prevent the two spatio-temporal memory traces from interfering with each other during recall by requiring that a cell have at least Θ active, large (i.e. a weight of 1) synapses in order to fire. The parameter, Θ, is called the *recall threshold*. In addition to having at least Θ active synapses in order to become active during recall, an L2 cell must also have the highest total horizontal synaptic input in its CM. The reason for this second requirement will be made apparent shortly.

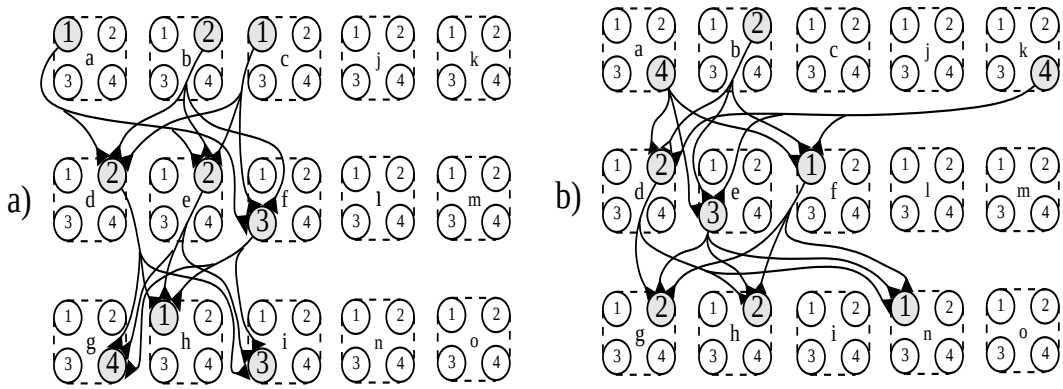

Figure 3: *Active cells are shaded. The three rows correspond to three consecutive time steps. a) shows a particular L2 code, Δ^i, that might be chosen at random for Φ^i, as well as the corresponding learning; white (i.e. open) synapses are ones that have been increased in a prior instance. b) shows an L2 code, Δ^j, for Φ^j. despite a great deal of overlap at the L1 level, the two L2 codes, Δ^i and Δ^j, overlap at only two cells; b_2, on the first timeslice and d_2, on the second timeslice.*

Fig. 4 shows that Φ^i is recalled perfectly if $S \geq \Theta \geq 2$. Cell b_2 sends output to e_3 and f_1, however those cells do not fire because other cells in their corresponding CMs—specifically, the correct ones, e_2 and f_3—receive more horizontal input and therefore prevent them from becomming active. Similarly, g_2 and h_2 remain inactive at $t = 3$ because other cells in their respective CMs have more input. However, n_1 is the only cell in its CM to receive input at $t = 3$. Therefore it *would* become active were it not for Θ being set to at least 2. This example illustrates why the two distinct mechanisms, winner-take-all dynamics and a

2 Description of TEMECOR

TEMECOR has two layers as shown in fig. 2. Layer 1 (L1) contains M binary feature detectors. Layer 2 (L2) contains M winner-take-all *competitive modules* (CMs) which are in one-to-one correspondence with the L1 cells. Each CM has K cells. Whenever a particular L1 cell fires (indicating the presence, in the input, of the corresponding feature), exactly one of the L2 cells in the corresponding CM is chosen winner and becomes active. Both the L1 and L2 representations are distributed, but the L2 representation is much sparser than that of L1. Each L2 cell has an excitatory modifiable {0,1}-valued synapse onto every other L2 cell (except for those in its own CM). It is this set of *horizontal connections* in which the chains encoding the temporal aspect of the inputs are embedded. A simple Hebbian learning rule is used. Every L2 cell active at timeslice t increases its weight onto all L2 cells active at $t+1$ unless the weight has already been increased.[4] Each L2 cell has an unmodifiable synapse onto its corresponding L1 cell. The purpose of these top-down (TD) or *reverse* connections is to allow the appropriate L1 pattern to be reinstated when an L2 pattern reads out during recall.

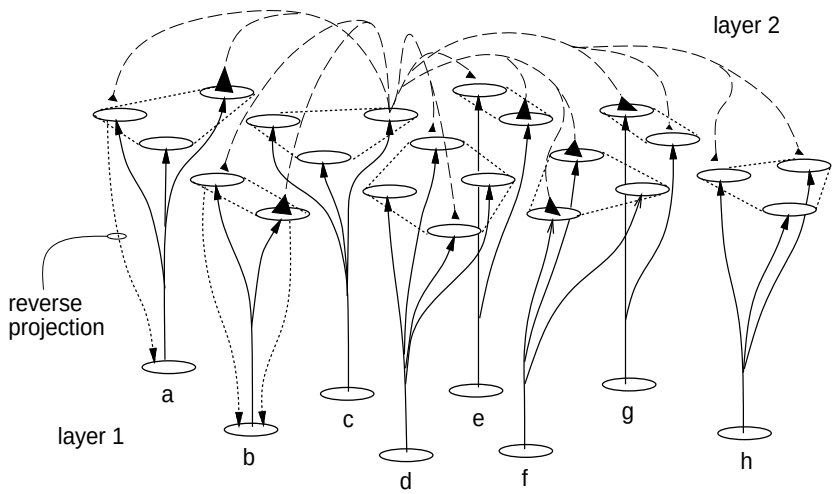

Figure 2: *TEMECOR has two layers. Some of the horizontal connections emanating from one L2 cell are depicted with dashed lines ending in either large (weight = 1) or small (weight = 0) black synapses. Only a few sample reverse (i.e. top-down) projections are shown. See text for more explanation.*

TEMECOR requires that environmental states have multiple features, although it does not require that all states have the *same* number of active features. However, for simplicity of exposition, we assume in this paper that all states, in a given simulation, have the same number S of active features, where $S < M$. The terms "episode," "spatio-temporal binary feature pattern" and "state sequence" are generally interchangeable in this paper. A typical episode, Φ^i, consisting of three timeslices can be expressed as:

Φ^i_1: {a, b, c}		A: {a, b, c}
Φ^i_2: {d, e, f}	or,	X: {d, e, f}
Φ^i_3: {g, h, i}		B: {g, h, i}

where each Φ^i_j denotes a particular timeslice. Lowercase letters denote features. As shown in the right-hand representation, unindexed uppercase letters are sometimes used to represent states; this facilitates representing that a particular state occurs in more than one episode and/or more than once in the same episode.

Fig. 3a shows a particular L2 representation[5] (L2-code) for Φ^i: L2 cells are assumed to be chosen at *random* (in the learning phase) within *active* CMs (i.e. CMs corresponding to active L1 cells). This ensures (statistically) that TEMECOR not only chooses *different* L2 codes (IRs), but that it chooses *highly*

[4] Thus, it is really more appropriate to think of spatio-temporal *swaths* of activation being embedded in the horizontal connections of L2 rather than *chains*.

[5] Essentially, the L2 codes correspond to the IRs discussed in the introduction.

Far more complex than this, however, is the problem of representing *whole sets* of CSSs in a distributed associative memory. For example, consider the set of CSSs:

Seq. 1: [A B B C A D B]
Seq. 2: [B C B B D A A]
Seq. 3: [A C A B B E]

Not only must the model find different IRs for all instances of a given state *within* each individual sequence, it must find different IRs for *all* instances of the state across *all* sequences in the set.

This general problem of representing large sets of CSSs is at the heart of speech and more generally, language. For example, the *spoken* lexicon of English can be adequately represented as a set of CSSs over a set (alphabet) of about 50 phonemes. Assuming a) a typical human speech lexicon contains perhaps 50,000 word forms, b) the average number of states (i.e. phonemes) per word is five, and c) there are 50 phonemes, then the average number of instances of a phoneme, across the whole lexicon, is 5000. Thus, any viable associative model of such a corpus must, at a minimum, allow for on the order of thousands of IRs for each phoneme (i.e. state).

Fig. 1 depicts the basic format of TEMECOR's internal representations of states. The cell groups (a-e) correspond to features. Any cell in the group can be used, in a particular instance, to represent the feature. A state, X, is defined as a set of co-active features. An internal representation of X, IR(X), is a choice of a particular *combination* of cells—one cell in each group corresponding to one of X's features. Thus, fig. 1 depicts one particular IR for the state consisting of features, {a,b,c}. The total number of unique IRs for the state is $(4 \times 4 \times 4 = 4^3 = 64)$. It is the exponentially large number of possible IRs for any particular state that gives TEMECOR its great capacity for storing CSSs.

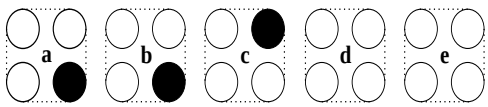

Figure 1: *Depiction of the basic representation format used in TEMECOR. The pattern (i.e. state) has three features, {a,b,c}, and assuming one cell is chosen to represent the corresponding feature in each group, there are* $4^3 = 64$*—i.e. an* exponentially *large number of—unique representations of that state.*

Merely *having* a very large space of possible IRs for any state is not, in itself, enough. A model must also actually *use* them during its natural operation. The SRN, JRN and RTRL all use continuous-valued cells. In principle, therefore, they all allow an *infinite* number of IRs for any state.[3] The problem with these models, however, is that they use a supervised learning procedure, in particular, *back-propagation* (Rumelhart, Hinton, and Williams, 1986), which acts to increase the similarity of [or, "homogenize"; Cleeremans (1993)] the chosen IRs.

For example, suppose the JRN is presented with the CSS, [A B C D C E C]. The operation of the JRN is defined so that on each timeslice, the target pattern is the next state of the sequence. Thus the same target, C, occurs repeatedly. This imparts a "force" which pushes the internal representations of the various predecessor states of C (B, D and E) closer and closer together, in terms of Euclidean distance in IR-space, with each additional training trial. This compression of the usable regions of IR-space is due to back-propagation and therefore occurs in the SRN and the RTRL as well. This effect is clearly revealed by hierarchical cluster analysis of the final set of IRs in any simulation (Smith and Zipser (1989); Elman (1990); Cleeremans (1993). Thus, the learning algorithm itself, tends to obliterate exactly the temporal context (state history) information needed to correctly encode the sequence. Furthermore, as Cleeremans points out, this effect is only made worse by further learning.

TEMECOR does not use a supervised learning scheme. Rather, IRs are chosen at random. That is, specific cells, within each active featural group (see fig. 1), are chosen at random, *not* in a manner dependent on the specific set of cells that were active on the previous timeslice (i.e. previous IR). This guarantees that the IRs the model chooses are uniformly distributed in the space of IRs. Thus, in contrast to the other models, TEMECOR not only *has* a large space of IRs, it definitely *uses* them.

[3] Of course the actual number of IRs is not infinite due to the limited resolution of any physical system. However, even assuming the cells have only four resolvable levels of activity, a hidden layer of 15 cells has $4^{15} > 1$ billion states (Cleeremans, 1993)

TEMECOR: An Associative, Spatio-temporal Pattern Memory for Complex State Sequences

Gerard J. Rinkus

Department of Cognitive and Neural Systems
Boston University, 111 Cummington Street, Boston, MA 02215

ABSTRACT: The problem of representing *large* sets of complex state sequences (CSSs)—i.e. sequences in which states can recur multiple times—has thus far resisted solution. This paper describes a novel neural network model, TEMECOR, which has very large capacity for storing CSSs. Furthermore, in contrast to the various back-propagation-based attempts at solving the CSS problem, TEMECOR requires only a single presentation of each sequence. TEMECOR's power derives from a) its use of a combinatorial, distributed representation scheme, and b) its method of choosing internal representations of states at random. Simulation results are presented which show that the number of spatio-temporal binary feature patterns which can be stored to some criterion accuracy (e.g. 97%) increases faster-than-linearly in the size of the network. This is true for both uncorrelated and correlated pattern sets, although the rate is slightly slower for correlated patterns.

1 Introduction

This paper describes an associative, spatio-temporal pattern memory model, TEMECOR[1], that can store a very large set (e.g. thousands) of complex state sequences defined over a relatively small number (e.g. tens) of states. Guyon, Personnaz, and Dreyfus (1988) define a *complex state sequence* (CSS) as a sequence in which states can recur multiple times; e.g. [A B B C A D B]. Furthermore, the model can do this given only a single presentation of each sequence. On the basis of these two properties, capacity and number of trials needed per sequence, TEMECOR *far exceeds* various other neural models which have been applied to the problem of representing sets of CSSs—specifically, the *Simple Recurrent Network* (SRN) of Elman (1990), the recurrent model of Jordan (1986) which we will call the *Jordan Recurrent Network* (JRN), and the *Real-Time Recurrent Learning* (RTRL) model of Williams and Zipser (1989). These other models have been demonstrated on rather small instances of this problem, however they do not scale well to larger problem sizes. As Cleeremans (1993, p.66) explains, "...the relation between the size of the problem and the number of epochs to reach a learning criterion was exponential for all network sizes."[2]

In contrast, this report contains results of much larger simulations. More importantly, the easily discernible trend in these results is that the number of sequences that can be stored, to some criterion accuracy, increases faster than the number of cells in the model.

In order to successfully represent the example CSS above, [A B B C A D B], a model must find different (although possibly overlapped) internal representations (IRs) for each instance of each state. For example, the IR for the first occurrence of state B, IR(B_1), must be different from IR(B_2) and from IR(B_3); otherwise, during recall, the model will not be able to reliably transition to the correct states following the various instances of state B.

[1] TEMECOR stands for *Temporal Episodic MEmory using COmbinatorial Representations.* A Preliminary description of the basic design and operational principles of TEMECOR can be found in Rinkus (1993), although the model has another name in that paper.

[2] Cleeremans' remarks concern the SRN specifically, however the underlying cause of the problem he identifies is the use of back-propagation which is common to the JRN and the RTRL models as well.