

## Storage and retrieval of complex sequences in neural networks

I. Guyon and L. Personnaz

*Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris, Laboratoire d'Electronique,  
10, rue Vauquelin, 75005 Paris, France*

J. P. Nadal

*Ecole Normale Supérieure, Groupe de Physique des Solides, 24, rue Lhomond, 75005 Paris, France*

G. Dreyfus

*Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris, Laboratoire d'Electronique,  
10, rue Vauquelin, 75005 Paris, France*

(Received 3 November 1987)

The storage and retrieval of complex sequences, with bifurcation points, for instance, in fully connected networks of formal neurons, is investigated. We present a model which involves the transmission of informations undergoing various delays from all neurons to one neuron, through synaptic connections, possibly of high order. Assuming parallel dynamics, an exact solution is proposed; it allows one to store without errors a number of elementary transitions which are of the order of the number of synaptic connections related to one neuron. A fast-learning algorithm, requiring a single presentation of the prototype sequences, is derived; it guarantees the exact storage of the transitions. It is shown that local learning procedures with repeated presentations, used for pattern storage, can be generalized to sequence storage.

### INTRODUCTION

The storage of sequences of patterns in Hopfield-type networks<sup>1</sup> has been investigated recently by several authors.<sup>2-9</sup> Aiming at biological applications, all these studies consider networks with asynchronous dynamics: each pattern is stable over some time period, at the end of which a sharp transition leading to the next pattern occurs. In this context, the main difficulty is the competition between stability of a pattern and transition toward the next pattern. One way to solve this problem is to consider Hebbian-type networks, where some synaptic efficacies evolve slowly in time as a function of the activity of the network.<sup>2,4-6</sup> Another type of difficulty is the storage and retrieval of complex sequences: in most of the preceding models, a given pattern can occur only once among all the stored sequences, which is a severe restriction. Having in mind bird-song acquisition, this problem has been addressed in Ref. 6; the proposed solution involves heterosynaptic interactions, together with sequence-detecting neurons ("hidden-units" coding for the transition from one state to the next one). As an alternative to the introduction of specific neurons, specific attractors (patterns) uncorrelated to the patterns of the sequences can be used.<sup>8</sup>

In this paper, we consider the learning and retrieval of sequences in a network with parallel dynamics. We want a network to retrieve a sequence of informations when presented with a (possibly distorted) part of it; therefore in contrast to asynchronous networks, it is sufficient to impose that the network should evolve, at each parallel

iteration, from one pattern of the sequence to the next. We address the problem of finding learning rules as efficient as possible, in order to store any given set of sequences. Results along these lines have already been obtained;<sup>9-11</sup> however, most of the proposed algorithms apply only to simple sequences. The purpose of this paper is to deal with efficient learning and retrieval of complex sequences. Note that this problem is different from the sequence-recognition problem.

Although we focus on parallel dynamics, part of the results can be useful for asynchronous implementations: optimal expressions of the synaptic matrix could be used to build the part of the synaptic efficacies which allow the transition from one state to the next one, in the models of references 2, 4, and 5.

In Sec. I we recall the results previously obtained; in Sec. II we give a definition of complex sequences and propose a general formulation of the problem leading to a "nonlocal" learning rule providing an exact solution when it exists. In Sec. III we consider the particular case of bifurcation points, we examine the different aspects of various solutions, and we illustrate the methods with examples. In Sec. IV we consider iterative learning procedures: we derive an algorithm requiring one presentation of the prototype sequences only and allowing the computation of the exact solution; it is shown that local learning rules with repeated presentation, on which new results have been obtained recently for pattern storage,<sup>12-14</sup> can be used for temporal associations. Finally, we shortly discuss how to chunk a given set of transitions into a sequence of patterns.

### I. STORAGE AND RETRIEVAL OF SIMPLE SEQUENCES

In this section we briefly report results obtained previously<sup>10</sup> concerning the storage of sequences in networks of formal neurons with parallel dynamics. We consider a fully connected network of  $n$  formal neurons. The dynamics of the network can be described as follows:

$$\sigma_i(t+1) = \text{sgn}(v_i(t) - \theta_i), \quad (1)$$

where  $\sigma_i(t)$  is the state of a neuron  $i$  (which is either  $+1$  or  $-1$ ),  $\theta_i$  its fixed threshold, and  $v_i(t)$  its "potential" (or local field) whose expression is

$$v_i(t) = \sum_j C_{ij} \sigma_j(t). \quad (2)$$

We will assume zero thresholds ( $\theta_i = 0$ ), but the following results can be easily generalized to nonzero thresholds. Synchronous updating of the neuron states makes it possible to use a very convenient matrix formalism. We define the state of the network as an  $n$ -dimensional vector  $\sigma$  whose components are the states of all the neurons. Therefore  $v(t) = C\sigma(t)$  is the  $n$ -dimensional "potential vector" and  $C$  is a  $(n, n)$  matrix which will be referred to as the "synaptic matrix". The next state of the network  $\sigma(t+1)$  is obtained after the "threshold process" (1).

Given a set of transitions in state space,

$$\sigma^k \rightarrow \sigma^{k+}, \quad k = 1, \dots, p$$

we want to compute a matrix  $C$  which guarantees the storage of the sequences. This will be true if the system of equations

$$C\sigma^k = \sigma^{k+}, \quad k = 1, \dots, p \quad (3)$$

can be solved. This can be put in matrix form

$$C\Sigma = \Sigma^+ \quad (4)$$

where  $\Sigma$  and  $\Sigma^+$  are the matrices

$$\Sigma = [\sigma^1, \sigma^2, \dots, \sigma^p], \quad \Sigma^+ = [\sigma^{1+}, \sigma^{2+}, \dots, \sigma^{p+}], \quad (5)$$

whose columns are the  $\sigma^k$  and their successors  $\sigma^{k+}$ , respectively.

We denote by  $\Sigma^I$  the pseudoinverse<sup>15</sup> of matrix  $\Sigma$ . Provided that the condition

$$\Sigma^+ \Sigma^I \Sigma = \Sigma^+ \quad (6)$$

is satisfied (which is always the case when vectors  $\sigma^k$  are linearly independent), Eq. (4) has exact solutions,

$$C = \Sigma^+ \Sigma^I + B(I - \Sigma \Sigma^I) \quad (7)$$

where  $B$  is an arbitrary matrix. In the following, we shall consider the case  $B = 0$  unless otherwise stated. Therefore the prescription becomes

$$C = \Sigma^+ \Sigma^I. \quad (8)$$

The coefficients of matrix  $C$  in relation (8) can be written as

$$C_{ij} = (1/n) \sum_{\mu, \nu} \sigma_i^{\mu+} (Q^I)_{\mu\nu} \sigma_j^\nu, \quad (9)$$

where  $Q$  is the matrix of the overlaps  $Q_{\mu\nu} = (1/n) \sum_i \sigma_i^\mu \sigma_i^\nu$ .

With such a learning rule, one can store faithfully  $O(n)$  transitions. However, no two  $\sigma^k$  can be identical. In the following, we shall see how to deal with sequences of higher complexity.

As mentioned earlier, the preceding results can be extended to asynchronous dynamics in the following way: the stabilizing matrix ( $T_{ij}$  in Ref. 4 and  $J_{ij}^{(1)}$  in Ref. 5) can be taken as the projection matrix  $\Sigma \Sigma^I$ ,<sup>11</sup> and the transition matrix ( $D_{ij}$  in Ref. 4 and  $J_{ij}^{(2)}$  in Ref. 5) can be computed by relation (8).

### II. LEARNING COMPLEX SEQUENCES

Learning sequences where all patterns are different is a severe restriction. On the other hand, it is unlikely that a given network, biological or artificial, could store a sequence whatever its complexity. In the context of biological modeling, it is shown<sup>6</sup> that the type of sequences that can be learned depends on the architecture of the network; this will hold as well for artificial devices. We define the complexity, as in Ref. 6, in the following way: in order to generate a sequence of  $p$  different patterns (or a set of such sequences with no pattern in common), one needs only to learn the  $p$  transitions from one pattern to its successor. Now suppose that one pattern occurs twice; when the network reaches this *bifurcation point*, it is unable to make a decision according to the deterministic dynamics described in (1), since the knowledge of the present state is not sufficient. Thus processing complex sequences require to keep, at each time step of the dynamics, a nonzero memory span. The order of a set of sequences is therefore defined as the minimal memory span necessary to produce all of them. Hence a sequence with no two identical patterns is of order zero. A sequence where a pattern occurs twice or more (bifurcation point) is of order one. Examples are shown in Fig. 1.

We consider now the general formulation for learning a set of sequences of a given order  $g$ . Following the preceding ideas (see also Ref. 9), the retrieval of a set of patterns of order  $g$  requires a dynamics whereby the activity at time  $t+1$  is a function of the activities at times  $t, t-1, \dots, t-g$ ; the potential  $v_i(t)$  must have the general form

$$\begin{aligned} v_i(t) = & \sum_j \sum_l C_{ij}^{(l)} \sigma_j(t-l) \\ & + \sum_{l, l'} \sum_{j, j'} C_{ij}^{(l, l')} \sigma_j(t-l) \sigma_{j'}(t-l') + \dots \\ & + \sum_{j_0, \dots, j_g} C_{ij_0, \dots, j_g}^{(0, \dots, g)} \sigma_{j_0}(t) \sigma_{j_1}(t-1) \dots \sigma_{j_g}(t-g). \end{aligned} \quad (10)$$

In the particular case where  $g = 1$ , this relation is written as

$$\begin{aligned}
 v_i(t) = & \sum_j C_{i,j}^{(0)} \sigma_j(t) + \sum_j C_{i,j}^{(1)} \sigma_j(t-1) \\
 & + \sum_{j,j'} C_{i,jj'}^{(0,0)} \sigma_j(t) \sigma_{j'}(t) \\
 & + \sum_{j,j'} C_{i,jj'}^{(1,1)} \sigma_j(t-1) \sigma_{j'}(t-1) \\
 & + \sum_{j,j'} C_{i,jj'}^{(0,1)} \sigma_j(t) \sigma_{j'}(t-1) .
 \end{aligned}$$

It can be more convenient, for the design of learning rules, to write the potential, with only one matrix  $C$  as

$$v_i(t) = \sum_a C_{i,a} \gamma_a(t)$$

or in a matrix form,

$$\mathbf{v}(t) = C\boldsymbol{\gamma}(t) , \tag{11}$$

where  $\boldsymbol{\gamma}(t)$  is the vector obtained by the concatenation of vectors  $\sigma(t), \sigma(t-1), \dots, \sigma(t) \otimes \sigma(t), \dots, \sigma(t) \otimes \sigma(t-1) \otimes \dots \otimes \sigma(t-g)$  and whose components are noted  $\gamma_a(t)$ . The subsequent vector  $\sigma(t+1)$  is still determined by the threshold process (1), which, for zero thresholds, reduces to

$$\sigma_i(t+1) = \text{sgn}(v_i(t)) . \tag{12}$$

In this form, the problem is a straightforward generalization of the storage of patterns and simple sequences with high-order interactions.<sup>16</sup> Suppose that we want to memorize a set of sequences of global order  $g$ . One has simply to build the vectors  $\boldsymbol{\gamma}^k$  with the  $\sigma^k, \sigma^{k-1}, \dots, \sigma^{k-g}$ , and then solve the system of equations:

$$C\boldsymbol{\gamma}^k = \boldsymbol{\sigma}^{k+1}, \quad k = 1, \dots, p \tag{13}$$

or, in analogy to relation (4),

$$C\Gamma = \Sigma^+ \tag{14}$$

where  $\Gamma$  is a matrix whose columns are the  $\boldsymbol{\gamma}^k$  and  $\Sigma^+$  is defined as previously. If the condition

$$\Sigma^+ \Gamma^I \Gamma = \Sigma^+ \tag{15}$$

is satisfied,  $\Gamma^I$  being the pseudoinverse of  $\Gamma$ , then the solution of Eq. (14) is

$$C = \Sigma^+ \Gamma^I + B(I - \Gamma \Gamma^I) \tag{16}$$

where  $B$  is an arbitrary matrix.

Clearly, this general formulation is not operational, considering the high dimension of vector  $\boldsymbol{\gamma}$ . In practical situations one should use simple solutions involving a particular choice of matrix  $B$  and where most of the high-order terms  $\sigma_j(t) \sigma_j(t-1) \dots \sigma_j(t-l)$  are set to zero, so that one works with a reduced vector  $\boldsymbol{\gamma}$ : such particular choices mean particular architectures of the network. In fact, there is a wide variety of interesting solutions.

Note that in any case, since  $\Gamma^I = (\Gamma^T \Gamma)^I \Gamma^T$ , one has only to compute the pseudoinverse of the matrix  $\Gamma^T \Gamma$ , which is a  $(p, p)$  matrix, whatever the dimension of the vector  $\boldsymbol{\gamma}$ . It should be pointed out that, as before, any particular solution will also be valid for networks with asynchronous dynamics.

### III. SEQUENCES OF ORDER 1

The present section shows various possible network architectures and learning rules for sequences of order 1. The main results are summarized on Table I.

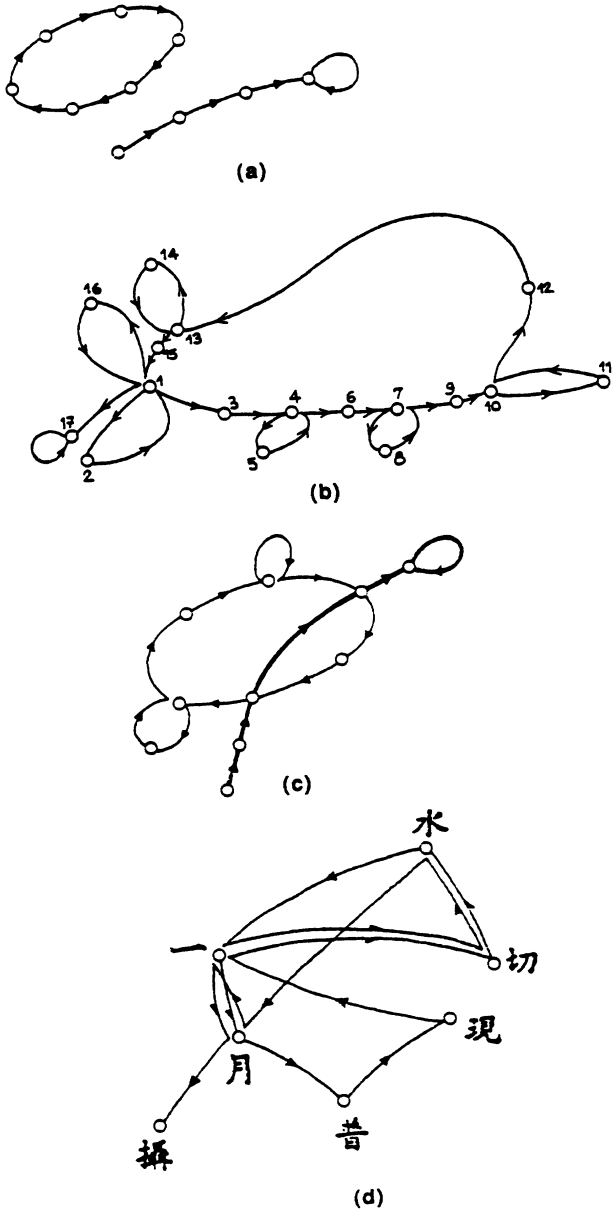


FIG. 1. Sequences of various orders (each dot represents one state of the network). (a) Order 0. (b) Order 1 with no two consecutive bifurcation points (this sequence represents Verlaine's poem "Dame souris trotte," see Fig. 3). (c) Order 1 with consecutive bifurcation points. (d) Complex sequence (order 3) representing a Chinese poem.

TABLE I. Summary of the various learning rules for sequences of order 0 or 1. In the third line,  $b$  is the number of bifurcation points.

Sequence order	Vector potential	Synaptic matrix	Number of synapses $N_s$	Capacity $p_{\max}$	$\alpha_{\max} = \frac{np_{\max}}{N_s}$	Initialization
0	$\mathbf{v} = C\boldsymbol{\sigma}$	$C = \Sigma^+ \Sigma^I{}^a$	$n^2$	$n$	1	1 state
1	$\mathbf{v} = C \begin{bmatrix} \boldsymbol{\sigma} \\ \boldsymbol{\sigma}^- \end{bmatrix} = C\boldsymbol{\gamma}$	$C = \Sigma^+ \Gamma^I{}^b$	$2n^2$	$2n$	1	2 states
1	$\mathbf{v} = C^{(0)}\boldsymbol{\sigma} + C^{(1)}\boldsymbol{\sigma}^-$	$C^{(0)} = S^+ S^I$ $C^{(1)} = S^+ + S^I{}^c$	$2n^2$	$n + b$	$0.5 + b/2n$	1 state
1	$\mathbf{v} = C(\boldsymbol{\sigma} \otimes \boldsymbol{\sigma}^-) = C\boldsymbol{\gamma}$	$C = \Sigma^+ \Gamma^I{}^d$	$n^3$	$n^2$	1	2 states

<sup>a</sup>Equation (8).

<sup>b</sup>Equation (19).

<sup>c</sup>Equation (23).

<sup>d</sup>Equation (24).

**A. Linear potential: Direct solution**

For a set of sequences of order 1, the only necessary pieces of information at each time step are the present state  $\boldsymbol{\sigma}(t) = \boldsymbol{\sigma}$  and the previous one  $\boldsymbol{\sigma}(t-1) = \boldsymbol{\sigma}^-$  in order to determine the successor  $\boldsymbol{\sigma}(t+1) = \boldsymbol{\sigma}^+$ . First, we consider relation (10) keeping only linear terms in  $\boldsymbol{\sigma}(t)$  and  $\boldsymbol{\sigma}(t-1)$ ,

$$\mathbf{v}(t) = C\boldsymbol{\gamma}(t), \text{ with } \boldsymbol{\gamma}(t) = \begin{bmatrix} \boldsymbol{\sigma} \\ \boldsymbol{\sigma}^- \end{bmatrix}. \tag{17}$$

The sequences to be learned can be decomposed into subsequences

$$\boldsymbol{\sigma}^{k-} \rightarrow \boldsymbol{\sigma}^k \rightarrow \boldsymbol{\sigma}^{k+}, k = 1, \dots, p.$$

If we define, as in (5), the matrices  $\Sigma$  and  $\Sigma^+$  whose columns are the vectors  $\boldsymbol{\sigma}^k$  and  $\boldsymbol{\sigma}^{k+}$ , respectively, and the matrix  $\Sigma^-$  whose columns are the predecessors  $\boldsymbol{\sigma}^{k-}$ , matrix  $\Gamma$  can be written as

$$\Gamma = \begin{bmatrix} \Sigma \\ \Sigma^- \end{bmatrix} \tag{18}$$

and relation (16) provides solutions, under condition (15).

The solution

$$C = \Sigma^+ \Gamma^I \tag{19}$$

has the following features.

(i) All kinds of first-order sequences can be stored and retrieved.

(ii) The dimension of the synaptic matrix is  $(n, 2n)$ , so that  $N_s = 2n^2$  synapses are required.

(iii) The same arguments as in Sec. I, for simple sequences, lead to a storage capacity  $O(2n)$ . More specifically, the number of elementary transitions that can be stored must be smaller than  $p_{\max} = 2n$  (the dimension of vectors  $\boldsymbol{\gamma}$ ), otherwise the solution (19) is no longer

exact. For comparison with other network architectures and learning rules, we define a storage coefficient  $\alpha = np/N_s$  (number of bits of information stored per synapse). The maximum storage coefficient  $\alpha_{\max} = np_{\max}/N_s$  is equal to 1 in the present case.

(iv) The retrieval of a sequence requires initializing the network with *two* consecutive (possibly noisy) states.

As an illustration, we show an example in which poems were stored (Fig. 2). In this example, and in the following ones, each character is coded on 6 bits: we use a Gray code such that the codes of two consecutive characters in the alphabet differ by one bit only; in addition, each capital letter is coded by the opposite of the code of the corresponding lower case; the 12 remaining codes are used for punctuations. A state of the network consists in 8 characters (i.e.,  $n = 48$  neurons); lines with less than 8 letters are ended by an appropriate number of space characters. Three poems were stored up to a total of  $p = 20$  elementary transitions, so that the storage coefficient ratio  $\alpha/\alpha_{\max}$  is approximately 0.2.

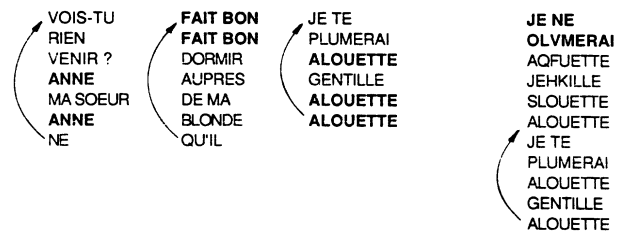


FIG. 2. Sequences of order 1 (with consecutive bifurcation points) were learned with  $p = 21$  transitions for  $n = 48$  neurons. Linear potential with learning rule (19) was used ( $2n^2$  synapses,  $\alpha/\alpha_{\max} \approx 0.2$ ). The three left columns show the poems which were learned; boldface characters indicate bifurcation points. The right column shows the retrieval of one of the poems, starting from an initialization with *two* distorted lines (in boldface).

**B. Linear potential: Solutions by inspection of bifurcation points**

If one first identifies all the bifurcation points, it is possible to construct simple solutions which require the initialization with one state only in the retrieval phase. We consider a vector potential linear with respect to the vectors  $\sigma^-$  and  $\sigma$ , as in (17); in the present case, however, we separate matrix  $C$  into two  $(n, n)$  submatrices  $C^{(0)}$  and  $C^{(1)}$ ,

$$\mathbf{v}(t) = C^{(0)}\sigma + C^{(1)}\sigma^- . \quad (20)$$

Consider the subsequences


$$\sigma^{k-} \rightarrow \sigma^k \rightarrow \sigma^{k+} \rightarrow \sigma^{k++}, \quad k = 1, \dots, p .$$

To guarantee their storage, one can impose, for all  $k$ , that


$$C^{(0)}\sigma^k = \sigma^{k+}, \quad C^{(1)}\sigma^{k-} = \sigma^{k+}; \quad k = 1, \dots, p$$

or equivalently


1 **Dame souris trotte,**  
 2 Noire dans le gris du soir,  
 1 **Dame souris trotte,**  
 3 Grise dans le noir.  
 4 **On sonne la cloche :**  
 5 Dormez, les bons prisonniers,  
 4 **On sonne la cloche :**  
 6 Faut que vous dormiez.  
 7 **Pas de mauvais rêve :**  
 8 Ne pensez qu'à vos amours,  
 7 **Pas de mauvais rêve :**  
 9 Les belles toujours !  
 10 **Le grand clair de lune !**  
 11 On ronfle ferme à côté.  
 10 **Le grand clair de lune !**  
 12 En réalité.  
 13 **Un nuage passe,**  
 14 Il fait noir comme en un four,  
 13 **Un nuage passe,**  
 15 Tiens, le petit jour !  
 1 **Dame souris trotte,**  
 16 Rose dans les rayons bleus,  
 1 **Dame souris trotte,**  
 17 **Debout, paresseux !**



**Noire dans un gris de soie**  
 Dakfksnprqs trotte,  
 Nrise ddns lqknodr.  
 On zoone lt cloche :  
 Dormez, les bons prisonniers,  
 On sonne la cloche :  
 Faut que vous dormiez.  
 Pas de mauvais rêve :  
 Ne pensez qu'à vos amours,  
 Pas de mauvais rêve :  
 Les belles toujours !  
 Le grand clair de lune !  
 On ronfle ferme à côté.  
 Le grand clair de lune !  
 En réalité.  
 Un nuage passe,  
 Il fait noir comme en un four,  
 Un nuage passe,  
 Tiens, le petit jour !  
 Dame souris trotte,  
 Rose dans les rayons bleus,  
 Dame souris trotte,  
 Debout, paresseux !



18 **O saisons, O châteaux !**  
 19 Quelle âme est sans défauts ?  
 20 J'ai fait la magique étude  
 21 Du bonheur, qu'aucun n'élude.  
 22 Salut à lui, chaque fois  
 23 Que chante le coq gaulois.  
 24 Ah ! je n'aurai plus d'envie  
 25 Il s'est chargé de ma vie.  
 26 Ce charme a pris âme et corps  
 27 Et disperse les efforts.  
 18 **O saisons, O châteaux !**  
 28 L'heure de sa fuite, hélas !  
 29 Sera l'heure du trépas.



**O maisons, O chameaux !**  
 Tuel.eV?peEfs,,Etn; ak'uts ?  
 J'ai fakt la h'?lque ftude .  
 Du bonheur, qu'aucun n'élude.  
 Salut à lui, chaque fois  
 Que chante le coq gaulois.  
 Ah ! je n'aurai plus d'envie  
 Il s'est chargé de ma vie.  
 Ce charme a pris âme et corps  
 Et disperse les efforts.  
 O saisons, O châteaux !  
 L'heure de sa fuite, hélas !  
 Sera l'heure du trépas.  
 O saisons, O châteaux !  
 Quelle âme est sans défauts ?  
 J'ai fait la magique étude




FIG. 3. Sequences of order 1 (with no consecutive bifurcation points) were learned, with  $p = 39$  transitions for  $n = 210$  neurons. Linear potential with learning rule (23) was used ( $2n^2$  synapses,  $\alpha/\alpha_{\max} \approx 0.2$ ). The left column shows the poems which were learned [the poem "Dame souris trotte" corresponds to the sequence indicated in Fig. 1(b)]. The right part shows the retrieval of the poems starting from an initialization with *only one* distorted line (in boldface).

$$C^{(0)}\sigma^k = \sigma^{k+}, \quad C^{(1)}\sigma^k = \sigma^{k++}; \quad k=1, \dots, p$$

which gives, in a matrix form,

$$C^{(0)}\Sigma = \Sigma^+, \quad C^{(1)}\Sigma = \Sigma^{++}. \quad (21)$$

Tentative solutions could be

$$C^{(0)} = \Sigma^+ \Sigma^I, \quad C^{(1)} = \Sigma^{++} \Sigma^I.$$

However, because of repetitions of identical vectors in matrix  $\Sigma$ , these are not exact solutions. To deal with this problem, one can simply build a matrix  $S$  which is identical to matrix  $\Sigma$ , except for the bifurcation points which appear only once, and are associated to null vectors. Considering the example of Fig. 1(b):

$$\begin{aligned} \Sigma &= [\sigma^1, \sigma^2, \sigma^1, \sigma^3, \sigma^4, \sigma^5, \sigma^4, \dots], \\ S &= [\sigma^1, \sigma^2, \sigma^3, \sigma^4, \sigma^5, \dots]; \\ \Sigma^+ &= [\sigma^2, \sigma^1, \sigma^3, \sigma^4, \sigma^5, \sigma^4, \sigma^6, \dots], \\ S^+ &= [0, \sigma^1, \sigma^4, 0, \sigma^4, \dots]; \\ \Sigma^{++} &= [\sigma^1, \sigma^3, \sigma^4, \sigma^5, \sigma^4, \sigma^6, \sigma^7, \dots], \\ S^{++} &= [0, \sigma^3, \sigma^5, 0, \sigma^6, \dots]; \end{aligned} \quad (22)$$

one has the following solution, under conditions  $S^+ S^I S = S^+$  and  $S^{++} S^I S = S^{++}$ :

$$C^{(0)} = S^+ S^I, \quad C^{(1)} = S^{++} S^I. \quad (23)$$

This solution acts on the dynamics of the network in a redundant way because the contributions of vectors  $\sigma^{k-}$  and  $\sigma^k$  lead independently to  $\sigma^{k+}$ , which allows us to initialize the network with only one state  $\sigma$  provided the latter is not a bifurcation point,  $\sigma^-$  being replaced by the vector  $0$ . The price to be paid, as compared to the previous solution, is a reduction in the storage capacity. It can also be noticed that the fact that the bifurcation points are associated to null vectors in matrices  $S^+$  and  $S^{++}$  prevents the storage of sequences with two consecutive bifurcation points [Fig. 1(c)]. The capacity of the network is  $O(n)$ .

The example chosen to illustrate the properties of this learning rule is shown in Fig. 3. Using the same code as in Fig. 2, a state now consists of a line with a total of 35 characters (i.e.,  $n=210$  neurons). Three poems have been stored, corresponding to  $p=37$  elementary transitions. The storage coefficient ratio  $\alpha/\alpha_{\max}$  is still approximately 0.2.

### C. Quadratic potential

If the number  $p$  of transitions to be stored is greater than the dimension of vector  $\gamma$ , a convenient solution can be obtained by considering three-neuron interactions only with a time delay, which is very similar in spirit to the architecture proposed in Ref. 6. In other words, matrix  $\Gamma$  is made of columns  $\gamma^k$  whose components are  $\sigma_i^k \sigma_j^{k-}$ ,

$$\gamma^k = \sigma^k \otimes \sigma^{k-}.$$

The solution

$$C = \Sigma^+ \Gamma^I \quad (24)$$

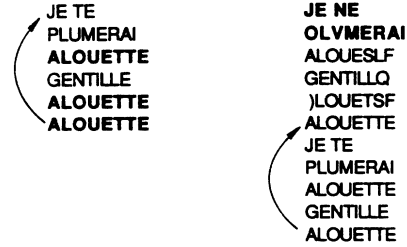


FIG. 4. Sequences of order 1 (with consecutive bifurcation points) were learned with  $p=424$  transitions for  $n=48$  neurons. Quadratic potential with learning rule (24) was used ( $n^3$  synapses,  $\alpha/\alpha_{\max} \approx 0.2$ ). The network learned all the poems of Fig. 2 and several others. The figure shows the retrieval of this poem starting from an initialization with *two* distorted lines.

is a straightforward generalization of the design proposed for associative memory with high-order interactions;<sup>16</sup> it has the following features.

- (i) All kinds of sequences of first order can be stored and retrieved.
- (ii) The dimension of the synaptic matrix is  $(n, n^2)$ , so that  $n^3$  synapses are required.
- (iii) The storage capacity is of order  $n^2$ ;  $\alpha_{\max} = 1$ .
- (iv) Retrieval of a sequence needs to give *two* consecutive states.

The example of Fig. 4 illustrates the retrieval performances of this last learning rule. In a network of  $n=48$  neurons, a large number of poems have been memorized, with a total of  $p=424$  elementary transitions. The storage coefficient ratio  $\alpha/\alpha_{\max}$  is still approximately 0.2.

## IV. ITERATIVE LEARNING

This section is devoted to the presentation of various iterative learning procedures: first, a nonlocal algorithm involving a single presentation of the prototypes (one-shot learning); next, a local procedure with multiple presentations of the prototypes (slow learning), which leads to the same matrix as the previous one; and, finally, alternate local learning rules.

### A. One-shot learning

We consider the general solution (16) for the learning of complex sequences. Matrix  $B$  will be taken equal to zero,

$$C = \Sigma^+ \Gamma^I. \quad (25)$$

One can compute this matrix iteratively, with a one-shot learning algorithm.<sup>17</sup> Suppose that  $k-1$  elementary transitions  $\gamma^h \rightarrow \sigma^{h+}$  ( $h=1, \dots, k-1$ ) have already been learned, leading to a synaptic matrix  $C(k-1)$ . To compute  $C(k)$  one applies

$$C(k) = C(k-1) + (\sigma^{k+} - \mathbf{v}^k) \gamma^{kT} / \|\gamma^k\|^2, \quad (26)$$

where

$$\begin{aligned} \mathbf{v}^k &= C(k-1)\boldsymbol{\gamma}^k, \\ \boldsymbol{\gamma}^k &= M(k-1)\boldsymbol{\gamma}^k, \\ M(k) &= M(k-1) - \boldsymbol{\gamma}^k \boldsymbol{\gamma}^{kT} / \|\boldsymbol{\gamma}^k\|^2, \end{aligned}$$

the initial conditions being

$$C(0)=[0], \quad M(0)=I.$$

This procedure is iterative, but is not local: it is convenient for computer simulations, but it is biologically implausible and difficult to implement in devices.

## B. Slow learning

### 1. Widrow-Hoff-type algorithm

In this section we show that results found for the storage of prototypes as stable states can be extended to sequence learning. For autoassociative memory, it has been shown analytically<sup>12</sup> that the procedure

$$C(k) = C(k-1) + (1/n)(\boldsymbol{\sigma}^k - \mathbf{v}^k)\boldsymbol{\sigma}^{kT} \quad \text{with } C(0)=[0], \quad (27)$$

which is a Widrow-Hoff-type learning rule,<sup>18</sup> yields the projection matrix when the number of presentations of the prototypes  $\boldsymbol{\sigma}^k$  goes to infinity, if the latter are linearly independent. A derivation along the same lines as in,<sup>12</sup> shows that, by repeated presentations of the prototype transitions, the learning rule

$$C(k) = C(k-1) + (1/n)(\boldsymbol{\sigma}^{k+} - \mathbf{v}^k)\boldsymbol{\gamma}^{kT} \quad \text{with } C(0)=[0] \quad (28)$$

leads to the exact solution  $C = \Sigma^+ \Gamma^J$  [relation (19)], if the vectors  $\boldsymbol{\gamma}^k$  are linearly independent.

### 2. Perceptron-type algorithms

The Perceptron-type algorithms<sup>19</sup> compute a matrix  $C$  which tends to satisfy the set of inequalities

$$\sum_j C_{ij} \sigma_i^k \sigma_j^k > \delta \quad \forall i, \quad \forall k \quad (29)$$

where, if learning of prototype states  $\boldsymbol{\sigma}^k$  in fully connected networks is considered, the sum  $\sum_j C_{ij}^2$  being kept fixed. A variant of the Perceptron theorem<sup>12</sup> shows that if  $\delta_{\max}$  is the maximum value of  $\delta$  for which a solution exists, then the Perceptron algorithm allows one to reach a solution with any  $\delta < \delta_{\max}$  in a finite number of learning steps. In Ref. 13 an algorithm is proposed which allows one to find  $\delta_{\max}$  and obtain a solution with  $\delta$  as close to  $\delta_{\max}$  as one wants. These results extend to temporal associations too: Eq. (12) can be put in the form (29),

$$\sum_a C_{ia} \xi_{ia}^k > \delta \quad \forall i, \quad \forall k \quad (30)$$

now with  $\xi_{ia}^k = \sigma_i^k \gamma_a^k$ . Thus the results on the above-mentioned algorithms hold, the number of neurons being replaced by the dimension  $m$  of the vectors  $\boldsymbol{\gamma}^k$ . In particular, the maximal capacity,<sup>20</sup> for which one has a nonzero attractivity, is  $2m$  elementary transitions if the  $\boldsymbol{\gamma}^k$  vectors

are chosen at random. This is not generally the case, even if the  $\boldsymbol{\sigma}^k$  vectors are chosen at random, some components of the  $\boldsymbol{\gamma}^k$  vectors being correlated. However, this capacity can be reached in the case of simple or first-order sequences learning using a linear potential.

## V. CHUNKING PROBLEM

Up to now, we have assumed that a sequence to be learned was made of patterns defined by a given number of  $n$  bits: for example, the poem of Fig. 3 is coded as a sequence of lines. However, one could have coded the poem in a different way: one extreme would be to take the letters as elementary patterns; the opposite extreme would be to take the whole poem as a single pattern. Of course, coding the lines as patterns is the most natural choice, and one can expect that any problem will have its "natural" chunking strategy. We do not attempt, in this section, to give a systematic way of finding the optimum strategy; we only try to give some general feeling on this question.

We would like to know the differences between three chunking strategies when they are allowed by the nature of the problem. In particular, chunking a set of bits in a sequence of  $p$  patterns of  $n$  bits is feasible if the resulting value  $\alpha$  of the storage coefficient remains smaller than  $\alpha_{\max}$ . We assume, for simplicity, that we use only networks with a linear potential and learning rule (19) (see Sec. III A), so that, if  $g$  is the order of the sequences, the dimension of vector  $\boldsymbol{\gamma}$  has to be, at least,  $(g+1)n$  [the initialization has to be done with  $(g+1)$  states], and the number of synapses  $N_s = (g+1)n^2$ . Thus any value of  $n$  is allowed provided  $\alpha = p / [(g+1)n] \leq \alpha_{\max} = 1$ .

Suppose that the total information to be stored has been divided, in a "natural" way, into  $p = p_0$  meaningful patterns of  $n = n_0$  bits. Let  $g = g_0$  be the maximum order of the resulting sequences. In this case, the dimension of vector  $\boldsymbol{\gamma}$  has to be equal to  $(g_0+1)n_0$ ; the number of synapses is  $N_s = (g_0+1)n_0^2$ . The initialization must be done with  $g_0+1$  states of  $n_0$  bits.

Now consider another choice, which consists in dividing all elementary states into two parts so that the number of bits per state becomes  $n = n_0/2$  and the number of elementary transitions  $p = 2p_0$ . The maximal order of the sequences increases: it is at least equal to  $g = 2g_0$ ; however, if  $g$  remains smaller than  $4g_0+3$ , the number of synapses is decreased. For example, if  $g = 2g_0$ , the dimension of vector  $\boldsymbol{\gamma}$  has to be equal to  $(2g_0+1)n_0/2$  and the corresponding number of synapses is  $N_s = (2g_0+1)n_0^2/4$ .

Finally, an alternate choice consists in gathering two elementary states in one, so that the number of bits per state becomes  $n = 2n_0$ , and the number of elementary transitions is  $p = p_0/2$ . The maximal order will decrease at least by a factor 2. In fact, in all the studied examples, this chunking leads to sequences of order zero. The problem lies in the number of synapses, which might, however, provide a better attractivity. It should be noticed that these various strategies are not equivalent with respect to initialization constraints: the adequate strategy is often

the most suitable for a desired initialization (for instance, it is natural to initialize with one line of a poem).

### CONCLUSION

In this paper, we have studied the learning and retrieval of sequences where some patterns occur several times. Considering parallel dynamics makes it possible to use a convenient matrix formalism, in the same way as for pattern recognition. We have shown that there are several interesting solutions, corresponding to various choices of architectures and algorithms. We have detailed particular solutions for sequences with bifurcation points. Moreover, fast- and slow-learning algorithms

used in the case of pattern storage are shown to apply as well for sequence storage. Finally, it has been shown that a given set of pieces of information can be put into the form of sequences in a variety of ways. The choice of a solution is shown to be the result of a tradeoff between various constraints such as storage capacity, sequence order, and number of synapses.

### ACKNOWLEDGMENTS

The authors are grateful to the members of the Groupe de Soutien aux Utilisateurs of the CIRCE computer center (Orsay).

- 
- <sup>1</sup>J. J. Hopfield, Proc. Natl. Acad. Sci. (U.S.A.) **79**, 2554 (1982).  
<sup>2</sup>P. Peretto and J. J. Niez, *Disordered Systems and Biological Organization*, edited by E. Bienenstock, F. Fogelman, and G. Weibusch (Springer, Berlin, 1986), pp. 171–185.  
<sup>3</sup>D. W. Tank and J. J. Hopfield, Proc. Natl. Acad. Sci. (U.S.A.) **84**, 1896 (1987).  
<sup>4</sup>D. Kleinfeld, Proc. Natl. Acad. Sci. (U.S.A.) **83**, 9469 (1986).  
<sup>5</sup>H. Sompolinsky and I. Kanter, Phys. Rev. Lett. **57**, 2861 (1986).  
<sup>6</sup>S. Dehaene, J. P. Changeux, and J. P. Nadal, Proc. Natl. Acad. Sci. (U.S.A.) **84**, 2727 (1987).  
<sup>7</sup>J. Buhmann, K. Schulten, Europhys. Lett. **4**, 1205 (1987).  
<sup>8</sup>D. Amit, Proc. Natl. Acad. Sci. (U.S.A.) **85**, 2141 (1988).  
<sup>9</sup>J. Keeler, J. Cognitive Sci. (to be published).  
<sup>10</sup>T. Kohonen, E. Oja, and P. Lehtiö, *Parallel Models of Associative Memory*, edited by E. Hinton and J. A. Anderson (Lawrence Erlbaum Assoc. Publ., Hillsdale, NJ, 1981), p. 121.  
<sup>11</sup>L. Personnaz and I. Guyon, G. Dreyfus, Phys. Rev. A **34**, 4217 (1986).  
<sup>12</sup>S. Diederich and M. Opper, Phys. Rev. Lett. **58**, 949 (1987).  
<sup>13</sup>W. Krauth and M. Mézard, J. Phys. A **20**, L745 (1987).  
<sup>14</sup>E. Gardner, Europhys. Lett. **4**, 481 (1987); G. Pöppel and U. Krey, *ibid.* **4**, 979 (1987); D. Kleinfeld and D. B. Pendergraft, Biophysics (Engl. Transl.) **51**, 47 (1987).  
<sup>15</sup>A. Albert, *Regression and the Moore-Penrose Pseudoinverse* (Academic, New York, 1972).  
<sup>16</sup>L. Personnaz, I. Guyon, and G. Dreyfus, Europhys. Lett. **4**, 863 (1987).  
<sup>17</sup>T. Kohonen, *Self Organization and Associative Memory* (Springer, Berlin, 1984).  
<sup>18</sup>B. Widrow and M. E. Hoff, 1960 IRE Wescon Convention Record, Part 4, p. 96.  
<sup>19</sup>F. Rosenblatt, *Principles of Neurodynamics* (Spartan, New York, 1962).  
<sup>20</sup>T. M. Cover, IEEE Trans. EC **14**, 3, 326 (1965); S. Venkatesh, *Neural Networks for Computing*, edited by J. S. Denker (AIP, New York, 1986); E. Gardner, Europhys. Lett. **4**, 481 (1987).