Perceptron-like learning in time-summating neural networks

1992 J. Phys. A: Math. Gen. 25 4373

(http://iopscience.iop.org/0305-4470/25/16/014)

J. Phys. A: Math. Gen. 25 (1992) 4373–4388. Printed in the UK

Perceptron-like learning in time-summating neural networks

Paul C Bressloff† and John G Taylor‡

† GEC-Marconi Ltd, Hirst Research Centre, East Lane, Wembley, Middlesex HA9 7PP, UK
‡ Department of Mathematics, King's College, The Strand, London WC2 2LL, UK

Received 23 January 1992, in final form 27 April 1992

Abstract. This paper investigates the ability of a single-layer, time-summating neural network to associate and store temporal sequences. In particular, the associative learning of temporal sequences is reformulated as an equivalent classification task involving static patterns. This leads to a generalization of the perceptron learning rule and convergence theorem to the case of temporal sequences. Using geometrical arguments based on linear separability it is shown how a time-summating network can handle temporal features such as ordering and coarticulation effects. Such an ability is a consequence of the fact that the time-summating network develops an activity trace consisting of a decaying sum of all previous inputs to the network. On the other hand, such an activity trace may also lead to an accumulation of errors in the presence of noisy inputs. This motivates a modification of the perceptron learning rule involving the introduction of a stability parameter that guarantees a certain level of robustness to noise. The performance of the network in the presence of random input sequences is then analysed using statistical-mechanical techniques. Finally, it is shown how, with small modifications, the time-summating network can be trained to store and recall complex sequences.

1. Introduction

A major limitation of standard feedforward neural networks is that they are not suitable for processing temporal sequences, since there is no direct mechanism for correlating separate input patterns belonging to the same sequence. There are a number of distinct features associated with the processing of temporal sequences that a neural network should be able to handle in applications such as speech recognition: (i) the output produced by a pattern in a sequence depends on its order within the sequence (*temporal ordering*); (ii) the output produced by a particular pattern in a sequence depends on previous or future patterns of that sequence (*coarticulation effects*); (iii) an entire sequence of patterns is recognized as a single distinct category; and (iv) variations occur in the rate of data presentation (*time warping*). A network should also be able to determine when a particular sequence terminates.

A common approach to many temporal sequence processing tasks is to convert the temporal pattern into a spatial one by dividing the sequence into manageable segments using a moving window and to temporarily store each sequence segment in a buffer. The resulting spatial pattern may then be presented to the network in the usual way and learning algorithms such as back-error-propagation applied accordingly [1, 2]. In signal-processing applications, each input pattern of the sequence could be a short-time Fourier transform, which has been discretized into a number of frequency bins; the associated spatial pattern of the buffer would then be a frequency-time spectrogram.

0305-4470/92/164373+16$04.50

However, there are a number of drawbacks with the buffer method: (i) Each element of the buffer is connected to all the units in the subsequent layer so that the number of weights increases with the size of the buffer, which may lead to long training times due to the poor scaling of learning algorithms; (ii) the buffer must be sufficiently large to accommodate the largest possible sequence, which must be known in advance; (iii) the buffer converts temporal shifts to spatial ones so that, for example, the representation of temporal correlations is obscure, and a misaligned signal in the spatial representation may bear little resemblance to the correctly aligned input signal; and (iv) the buffer is inefficient if the output response to each pattern of the input sequence is required rather than just the final output.

It would appear that, in spite of these difficulties, the above spatial approach to temporal sequence processing is the one used in the auditory system of the brain. That is, the inner ear acts as a bank of frequency filters that spatially separates the frequency components of the input time series. Such a spatial pattern is preserved by the organization of the auditory nerve fibres. If the firing pattern of each of these fibres is characterized by an average firing rate then one obtains a frequency-time energy spectrogram, which is analogous to that used in the buffer approach. A closer examination of the auditory system, however, suggests that there is also a direct coupling between the time domain input signal and the firing patterns of the fibres. That is, there is information encoded by temporal modulations of the instantaneous firing rates of the fibres [3]. (Such temporal encoding, in contrast to average firing rates, is insensitive to amplitude fluctuations.) Moreover, it is possible that higher levels of auditory processing involve some form of temporal processing.

The deficiencies of the buffer method suggest that a more flexible representation of time is needed. One simple approach is to introduce into the network a layer of so called time-summating neurons [4–7]; the activation state or local field of each of these neurons at the discrete time step $t-1$ is delayed and then fed back as input to the neuron to determine its activation state at time t. Each neuron thus maintains an activity trace consisting of a decaying sum of all previous inputs to that neuron, which forms an internal representation of the temporal input sequences. (This activity trace is similar in form to the local fields arising from synapses with slow dynamic response as considered by Kleinfeld *et al* [8]). The inclusion of such a layer eliminates the need for a buffer and allows the network to operate directly in the time domain. Moreover, the back-error-propagation algorithm may be implemented simply and efficiently, since the activity traces computed locally hold all information necessary for the back-propagation of errors. Such networks have been applied to the classification of speech signals [4, 5], motion detection [6], and the storage and recall of complex sequences [7].

In this paper we study perceptron-like learning algorithms for the association and storage of temporal sequences in a two-layer, time-summating neural network. The network is taken to consist of an input layer of time-summating neurons, each with a linear output function, connected to an output layer of standard binary-threshold neurons. We first consider the problem of learning sequences of input-output mappings in a network with a single ouput neuron (section 2). We reformulate this problem in terms of an equivalent classification task that requires the separation of sets of static patterns. This allows us to generalize the perceptron learning rule and convergence theorem [9] to the case of temporal sequences. We then analyse a number of simple examples to illustrate how the presence of activity traces in the input layer enables a time-summating network to handle temporal features such as ordering and coarticulation effects.

Another consequence of these activity traces is that there is a build up of correlations between the local fields of the output neuron generated by an input sequence. In section 3 we show how this may lead to an accumulation of errors along a sequence in the presence of noisy inputs. This motivates a modification of the perceptron-like learning algorithm of section 2 involving the introduction of a stability parameter η that guarantees a certain level of robustness to noise. Such a modification is similar to Gardner's generalization of perceptron learning to the problem of storing static patterns in attractor networks with finite basins of attraction [10]. We also discuss some results concerning the optimal capacity of a two-layer, time-summating network [11, 12].

Finally, in section 4 we consider the storage and retrieval of temporal sequences in a two-layer time-summating network. As shown in [7], such a network can learn to store a sequence $\{I(0), I(1), \ldots, I(R)\}$ by taking the desired output at time r, $r = 0, \ldots, R-1$, to be the next pattern in the sequence $I(r+1)$. If the network is successfully trained then the sequence may be retrieved by simply seeding the network with $I(0)$ and feeding back the resulting output $I(1)$ along a delay so that $I(1)$ is the input at the next time step, etc [7]. We analyse this network in terms of a set of independent time-summating perceptrons, and then apply the results of previous sections.

2. Associative learning of temporal sequences

Consider a two-layer neural network consisting of N time-summating input units, each with a linear output function, connected to a single output unit, which is taken to be a binary-threshold neuron. Suppose that some input sequence of length R, $\{I(1), \ldots, I(R)\}$, is presented to the network, where $I = (I_1, \ldots, I_N)$ is an N-component pattern vector that may be analogue or binary valued. The activation state $V_j(t)$ of the jth time-summating input neuron at time t along the sequence is given by

$$V_j(t) = k_j V_j(t-1) + I_j(t) = \sum_{r=1}^{t} k_j^{t-r} I_j(r) \tag{2.1}$$

where k_j is the decay-rate (with $k_j < 1$). It is assumed that the activation states are initialized to zero before the presentation of each input sequence. (We shall not address here the important question of how a network determines when a sequence ends.) Thus, each time-summating neuron maintains a decaying activity trace of previous inputs to that neuron. The decay term $k_i V_i(t-1)$ in equation (2.1) may be viewed as a positive feedback along a delay line of weight k_i; this should be contrasted with models in which the output of the neuron is fed back rather than the activation state. The output of the network is given by

$$f(t) = \theta(V(t) - h) \qquad V(t) = \sum_j w_j V_j(t) \tag{2.2}$$

where w_j is the weight of the connection from the jth neuron to the single output neuron (see figure 1), h is a fixed threshold and $\theta(x) = 1$ if $x > 0$ and 0 otherwise. The input-output relationship of the network is then $\{I(t);\ t = 1, \ldots, R\} \rightarrow \{f(t);\ t = 1, \ldots, R\}$.

An interesting issue for associative learning, in the case of temporal sequences, is whether one takes the desired output of the network to be a single fixed pattern as in 'static' networks or a sequence of patterns. The simplest choice is to require a response

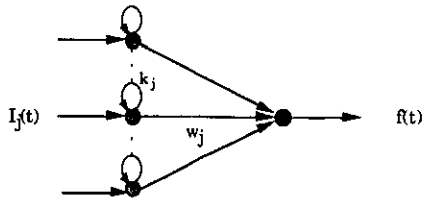

Figure 1. Two-layer feedforward network with an input layer of time-summating neurons.

only when the entire sequence has been presented, i.e. to specify a desired $f(R)$. This corresponds to a classification task in which the sequence is labelled by the output produced after presentation of the whole sequence. Thus, in speech recognition the input could be a sequence of phonemes and the output could be the word category to which this sequence belongs. On the other hand, for certain applications, especially in the area of control, the full output response of the network is needed, e.g. such an output could be a sequence of control commands in response to a sequence of sensor inputs. Moreover, even in the case of speech recognition, the additional information detailing how the response of a network varies in time may enhance the classification process. This is illustrated by Watrous and Shastri [5], who apply a two-layer time-summating network (referred to as a *temporal flow model*) to the problem of distinguishing between the words *go* and *no*. They define the target function of the single output neuron to be the ramp function $r(t)=0.5(1+t/R)$ if the input is *go*, say, and $1-r(t)$ if the input is *no*. In other words, the output is required to vary linearly from an initial value of 0.5 to the final value 1.0 or 0.0. (A more effective choice for the target function is an exponential function [13].)

In the case of the network of figure 1, we shall specify the desired output in terms of a sequence of binary patterns $\{\sigma(1),\ldots,\sigma(R)\}$. Suppose that the network is required to learn p input-output mappings $\{I^\mu(t);\ t=1,\ldots,R\}\rightarrow\{\sigma^\mu(t),\ t=1,\ldots R\},\ \mu=1,\ldots,p$. We shall reformulate this problem in terms of a classification task to which the perceptron learning theorem [9] may be applied. First, define a new set of inputs of the form

$$\tilde{I}_j^\mu(t)=\sum_{r=1}^{t} k_j^{t-r} I_j^\mu(r) \tag{2.3}$$

i.e. $\tilde{I}^\mu(t)$ is the output of the input layer at time t. Divide the pR inputs $\tilde{I}^\mu(t)$, $\mu=1,\ldots,p,\ t=1,\ldots,R$, into two sets F^+ and F^- where $\tilde{I}^\mu(t)\in F^+$ if $\sigma^\mu(t)=1$ and $\tilde{I}^\mu(t)\in F^-$ otherwise. Learning then reduces to the problem of finding a set of weights $\{w_j,\ j=1,\ldots,N\}$ such that the sets F^+ and F^- are separated by a single hyperplane in the space of inputs $\tilde{I}^\mu(t)$—linear separability. In other words, the weights must satisfy the pR conditions

$$\begin{aligned} &\sum_{j=1}^{N} w_j\tilde{I}_j^\mu(t)>h+\delta \qquad \text{if } \tilde{I}^\mu(t)\in F^+ \\ &\sum_{j=1}^{N} w_j\tilde{I}_j^\mu(t)<h-\delta \qquad \text{if } \tilde{I}^\mu(t)\in F^-. \end{aligned} \tag{2.4}$$

The perceptron convergence theorem [9] for the time-summating network of figure 1 may be stated as follows. Suppose that the weights are updated according to the perceptron learning rule. That is, at each iteration choose an input $\tilde{I}^\mu(t)$ from either

F^+ or F^- and update the weights according to the rule

$$w_j \to w_j + (\sigma^\mu(t) - \theta(w \cdot \tilde{I}^\mu(t) - h))\tilde{I}_j^\mu(t). \tag{2.5}$$

If there exists a set of weights that satisfy equation (2.4) for some $\delta > 0$, then the perceptron learning rule (2.5) will arrive at a solution of equation (2.4) in a finite number of time steps—independent of N. (Note that, in equation (2.5), t acts as an index for a particular input of the set $F^+ \cup F^-$; it does not refer to the iteration step of the learning algorithm.)

The above result implies that a two-layer time-summating neural network can learn the set of mappings $\{I^\mu(t);\ t = 1, \ldots, R\} \to \{\sigma^\mu(t),\ t = 1, \ldots, R\}$, $\mu = 1, \ldots, p$ provided that the associated classes F^+ and F^- are linearly separable. We shall illustrate this with a simple example for $N = 2$. Assume for simplicity that $k_1, k_2 = k$. Define the vectors

$$A = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.6}$$

and consider the mappings $AB \to 10$ and $BA \to 00$. This is essentially an ordering problem since the pattern A produces the output 1 or 0, depending respectively on whether it precedes or proceeds the pattern B. (Thus, it could not be solved by a standard perceptron.) Using equation (2.3) we introduce the four vectors

$$\tilde{A}(1) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \tilde{A}(2) = \begin{pmatrix} k \\ 1 \end{pmatrix} \qquad \tilde{B}(1) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \tilde{B}(2) = \begin{pmatrix} 1 \\ k \end{pmatrix}. \tag{2.7}$$

It is clear that the sets $F^+ = \{\tilde{A}(1)\}$ and $F^- = \{\tilde{A}(2), \tilde{B}(1), \tilde{B}(2)\}$ are linearly separable (figure 2(a)) and, hence, that the network can learn the above mappings. (On the other hand, the mappings $AB \to 11$ and $BA \to 00$ cannot be linearly separated by a single line (figure 2(b)), and a three-layer time-summating network is required.)

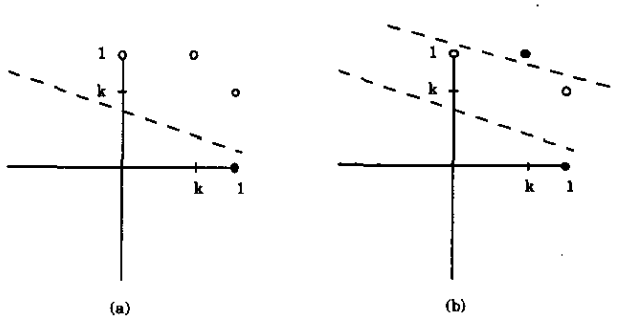

Figure 2. Example of (a) separable and (b) non-separable sets F^+ and F^- associated with the sequences of input–output mappings defined in the text. Points in F^+ and F^- are denoted by ● and ○, respectively.

In the previous example we assumed that the input neurons have the same decay rate k. However, it is clear that the network can solve a wider range of classification tasks if the decay rates are allowed to be site-dependent. A simple illustration of this is given in figure 3, which describes the sets F^+ and F^- associated with the mapping $AAA \to 101$ where $A' = (11)$. In figure 3(a) the two sets are not linearly separable since $k_1 = k_2 = k$, whereas in figure 3(b) $k_1 \neq k_2$ are the two sets are now separable.

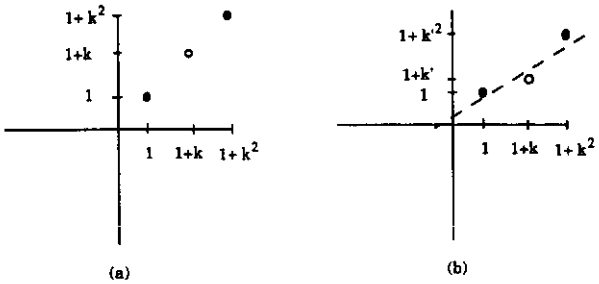

Figure 3. Illustration of the use of input neurons with different decay rates, $k \neq k'$, to solve a problem which is not linearly separable when $k = k'$.

Finally, note that the above approach can also be applied to the problem of temporal sequence classification. The network associates each input sequence with a class specified by the final output of the network $\theta(\tilde{I}^{\mu}(R)-h)$. The network may be trained to match its output with the correct class $\sigma^{\mu}(R)$ using the learning rule (2.5) with $t=R$. In this example, there is only a single contribution $\tilde{I}^{\mu}(R)$ to the set $F^{+}\cup F^{-}$ from each input sequence.

3. Temporal correlations and the accumulation of errors

So far we have shown how a time-summating network can solve certain tasks in the time domain by storing previous inputs in the form of activity traces within the input layer. Another consequence of the activity traces is that temporal correlations are set up between the local fields of the output neuron on presentation of an input sequence. To illustrate this point, consider sequences of patterns of the form $\{\boldsymbol{Y}(1),\ldots,\boldsymbol{Y}(R)\}$ where $\boldsymbol{Y}(t)=\boldsymbol{I}(t)+\boldsymbol{\zeta}(t)$, $\boldsymbol{I}(t)$ is fixed and $\boldsymbol{\zeta}(t)$ is a random pattern vector satisfying

$$\langle \zeta_i(t)\rangle = 0 \qquad \langle \zeta_i(t)\zeta_j(r)\rangle = \gamma\delta_{ij}\delta_{tr}. \tag{3.1}$$

Assuming for simplicity that $k_i=k$ for all $i=1,\ldots,N$, the mean and covariance of the corresponding local fields $V(r)=\sum_{s=1}^{r} k^{r-s}\boldsymbol{w}\cdot\boldsymbol{Y}(s)$ are given by

$$\langle V(r)\rangle = \sum_{s=1}^{r} k^{r-s}\boldsymbol{w}\cdot\boldsymbol{I}(s) = \boldsymbol{w}\cdot\tilde{\boldsymbol{I}}(r) \tag{3.2}$$

and

$$\operatorname{cov}[V(r),V(r')] = \gamma\left(\sum_{j=1}^{N} w_j^2\right)D_{rr'} \tag{3.3}$$

where

$$\mathbf{D}=\begin{pmatrix} 1 & k & k^2 & \ldots & k^{R-1} \\ k & 1+k^2 & k+k^3 & \ldots & \\ k^2 & k+k^3 & 1+k^2+k^4 & & \\ k^{R-1} & \ldots & \ldots & \ldots & \sum_{r=0}^{R-1} k^{2r} \end{pmatrix} \tag{3.4a}$$

or, in component form

$$D_{rr'} = k^{|r-r'|}\sum_{s=1}^{\min(r,r')} k^{2(s-1)}. \tag{3.4b}$$

The $R \times R$ temporal correlation matrix $\mathbf{D}$ is symmetric, has positive definite eigenvalues and unit determinant (see [12]). The inverse matrix $\mathbf{D}^{-1}$ has the simpler form

$$\mathbf{D}^{-1} = \begin{pmatrix} 1+k^2 & -k & 0 & \dots & 0 \\ -k & 1+k^2 & -k & \dots & 0 \\ 0 & -k & 1+k^2 & \dots & \\ & & & & -k \\ & & & -k & 1 \end{pmatrix} \tag{3.5a}$$

with components

$$D^{-1}_{rr'} = [1+k^2(1-\delta_{r,R})]\delta_{r,r'} - k(\delta_{r,r'+1} + \delta_{r+1,r'}). \tag{3.5b}$$

Hence, although there are no extrinsic correlations between patterns within an input sequence, intrinsic temporal correlations do arise between the sequence of activation states of the output neuron. Moreover, the variance of $V(r)$, determined by the diagonal term D_{rr}, satisfies

$$\text{var}(V(r)) = \gamma D_{rr} = \gamma \frac{1-k^{2r}}{1-k^2} \tag{3.6}$$

where we have set $\Sigma_j w_j^2 = 1$. Therefore, the variance is a monotonically increasing function of r that, for infinitely long sequences, approaches the asymptotic value $\gamma/(1-k^2)$ as $r \to \infty$. For values of the decay rate $k \sim 1$ this limiting value is large and may lead to an accumulation of errors. For example, assume that N is large so that the distribution of activation states $\{V(1), \dots, V(R)\}$ associated with the random sequences $\{\mathbf{Y}(1), \dots, \mathbf{Y}(R)\}$ is described by a multivariate Gaussian with mean and covariance given respectively by equations (3.2) and (3.3). Suppose that the underlying patterns $\boldsymbol{I}(r)$ should be mapped to the outputs $\sigma(r) = \pm 1$ for $r = 1, \dots, R$. Then the probability that the network produces an incorrect output at time r, on presentation of a noisy input sequence $\{\mathbf{Y}(1), \dots, \mathbf{Y}(R)\}$, is

$$\begin{aligned} \varepsilon(r) &= \int \mathrm{d}V(r)\theta(-V(r)\sigma(r)) \exp\left(-\frac{1}{2\gamma}\sum_{s,s'}(V(s)-\langle V(s)\rangle)D^{-1}_{ss'}(V(s')-\langle V(s')\rangle)\right) \\ &= H(\langle V(r)\rangle \sigma(r)/\sqrt{\gamma D_{rr}}) \end{aligned} \tag{3.7}$$

where $H(x)$ is an error function,

$$H(x) = \int_x^\infty \frac{\mathrm{d}z}{\sqrt{2\pi}} \mathrm{e}^{-z^2/2} \tag{3.8}$$

and we have switched to the spin output function $f(r) = \text{sign}(V(r) - h)$ for convenience. Equation (3.6) implies that the scale factor $1/\sqrt{D_{rr}}$ in equation (3.7) could lead to an accumulation of errors along a sequence when $k \sim 1$, $nb \cdot H(x)$ is a monotonically decreasing function of x. One cannot avoid this problem by taking $k \ll 1$, since, in order to disambiguate long sequences, it is necessary to have k as close to unity as possible. That is, to capture relationships between patterns of a sequence of length R, k^R should not be too small. A reasonable condition might be $k^R > 0.1$, which implies that $|\log k| < \log 10/R \sim 0$ for large R.

As it stands, the above network is not robust to noisy inputs when trained on noise-free pattern sequences using the learning rule (2.5). For if the network has successfully learned the mapping $\{\boldsymbol{I}(1), \dots, \boldsymbol{I}(R)\} \to \{\sigma(1), \dots, \sigma(R)\}$ then equation

(2.5) only ensures that $x(r) \equiv \sigma(r) w \cdot \tilde{I}(r) > 0$; we have set the threshold $h = 0$ for convenience. In particular, $x(r)$ may be sufficiently small such that, on presentation of a noisy sequence $\{Y(1), \ldots, Y(R)\}$, the probability of an error satisfies $\varepsilon(r) = H(x(r)/\sqrt{\gamma D_{rr}}) \approx H(0) = \frac{1}{2}$. Moreover, this feature is reinforced for long sequences when $k \approx 1$, since the scale factor $\sqrt{D_{rr}}$ considerably reduces the effective size of $x(r)$. (For a more detailed analysis of $\varepsilon(r)$, based upon the statistical-mechanical techniques of Gardner [10], see [14]). Therefore, to ensure a certain level of robustness to noise we introduce a stability parameter η, $\eta > 0$ and require that

$$\sigma(r) w \cdot \tilde{I}(r) > \eta \left(\sum_j w_j^2 \right)^{1/2}. \tag{3.9}$$

Then $\varepsilon(r)$ satisfies $\varepsilon(r) < H(\eta/\sqrt{\gamma D_{rr}})$, and it is now possible to reduce the upper bound for the probability of error by increasing η. (However, as in the case of static inputs, the number of patterns that may be learned without error decreases as η increases (see below and [12]).)

Equation (3.9) is similar in form to the fixed-point equation for the storage of a static pattern in an attractor network, with η guaranteeing a finite size for the basins of attraction [10]. As in [10], it is necessary to modify the perceptron-like learning algorithm (2.5) when $\eta > 0$. Suppose that the network is required to learn p mappings of the form $\{I^\mu(1), \ldots, I^\mu(R)\} \to \{\sigma^\mu(1), \ldots, \sigma^\mu(R)\}$, $\mu = 1, \ldots, p$. We introduce a mask of the form

$$\varepsilon^{\mu,r} = \theta(\eta \|w\| - \sigma^\mu(r) w \cdot \tilde{I}^\mu(r)) \tag{3.10}$$

where $\|w\| = (\Sigma_j w_j^2)^{1/2}$ and update the weights according to the rule

$$w_j \to w_j + \varepsilon^{\mu,r} \sigma^\mu(r) \tilde{I}_j^\mu(r). \tag{3.11}$$

For $\eta = 0$, equation (3.11) is equivalent to equation (2.5). The convergence of the algorithm may be established using a straightforward extension of Gardner's proof for fixed points [10]. Assume that a solution w^* exists such that for each $\mu = 1, \ldots, p$ and $r = 1, \ldots, R$,

$$\sigma^\mu(r) w^* \cdot \tilde{I}^\mu(r) > (\delta + \eta) \|w^*\| \tag{3.12}$$

where δ, $\eta > 0$. Let $w^{(n)}$ be the set of weights after n iterations of the learning algorithm and define $X^{(n)}$ by

$$X^{(n)} = \frac{w^{(n)} \cdot w^*}{\|w^{(n)}\| \, \|w^*\|}. \tag{3.13}$$

Using equations (3.10)–(3.12) and the Schwartz inequality, it may be shown that $X^{(n)}$ satisfies [10]

$$1 \geqslant X^{(n)} > \frac{\eta + \delta}{\eta + (\log n / n)[N/2(\eta + \delta)]}. \tag{3.14}$$

It follows from (3.14) that, in order for the upper bound of $X^{(n)}$ not to be violated, the algorithm must terminate in less than n_c time steps where $n_c / \log n_c = N/[2\delta(\eta + \delta)]$. Hence, the learning rule (3.11) is guaranteed to find a solution to equation (3.9) if at least one such solution exists.

As in the case of the standard perceptron [10], statistical-mechanical techniques may be used to analyse the performance of a two-layer, time-summating network in the thermodynamic limit [11, 12, 14]. One of the features that emerges from such

analyses is the non-trivial contribution from intrinsic temporal correlations, as characterized by the matrix **D** of equation (3.4). For example, consider the problem of learning αN mappings $\{S^\mu(t);\ t=1,\dots,R\}\to\{\sigma^\mu(t),\ t=1,\dots,R\}$, where $S_i^\mu(r)=\pm 1$ with equal probability and similarly for $\sigma^\mu(r)$. Assuming that $k_i=k$ and the weights are normalized such that $(\Sigma_{j\neq i} w_{ij}^2)^{1/2}=N$, the maximum number $\alpha_c^{(k)}(\eta)$ of sequences that may be learned without error satisfies [12]

$$1=\alpha_c^{(k)}(\eta)\left\langle \int \mathscr{D}t \sum_I \left(\Theta_I(t) \sum_{r,r'\in I} (t_r+\eta) D_{rr'}^{(I)} (t_{r'}+\eta)\sigma(r)\sigma(r') \right) \right\rangle_\sigma \tag{3.15}$$

where $\langle\dots\rangle_\sigma$ indicates averaging over the random output sequences $\{\sigma(1),\dots,\sigma(R)\}$, $\mathscr{D}t$ is the integration measure

$$\mathscr{D}t=\prod_{r=1}^{R}\frac{dt_r}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\sum_{r,r'}\sigma(r)t_r D_{rr'}^{-1}t_{r'}\sigma(r')\right) \tag{3.16}$$

and **D** is the matrix of equation (3.4); the elements of **D** determine the k-dependence of the optimal capacity. The first summation on the right-hand side of equation (3.15) is over all subsets $I\subset\{1,\dots,R\}$ and the $|I|\times|I|$ matrix $\mathbf{D}^{(I)}$ is defined by $\mathbf{D}^{(I)}\equiv[(D_{rs})|_{r,s\in I}]^{-1}$. The function $\Theta_I(t)$ restricts the integration over t to the domain $t_r\geq-\eta$ for all $r\in I$ and $t_r<-\eta$ otherwise, i.e. $\Theta_I(t)=\Pi_{r\in I}\theta(t_r+\eta)\Pi_{r'\notin I}\theta(-t_{r'}-\eta)$.

When $k=0$, the network is identical to a standard perceptron and, as expected, the optimal capacity is $\alpha_c^{(0)}(\eta)=\alpha_c(\eta)/R$, where

$$\alpha_c(\eta)=\left(\int_{-\kappa}^{\infty}\frac{dt}{\sqrt{2\pi}}(t+\eta)^2\,e^{-(1/2)t^2}\right)^{-1} \tag{3.17}$$

is the standard result for a perceptron [10], with $\alpha_c(\eta)$ a monotonically decreasing function of η such that $\alpha_c(0)=2/R$. A more interesting result is that [11], for $\eta=0$, the maximum capacity is independent of the decay parameter k with $\alpha_c^{(k)}(0)=2/R$. On the other hand, for $\eta>0$, $\alpha_c^{(k)}(\eta)$ is a monotonically decreasing function of k so that $\alpha_c^{(k)}(\eta)<\alpha_c(\eta)/R$. We conclude from this that the only effect of time-summating neurons, in the case of random, unbiased sequences, is to alter the effective size of the stability parameter η, leading to a smaller capacity and an enhanced robustness to noise. This result is consistent with the fact that the random input and output sequences are simple, in the sense that the probability of a pattern occurring more than once in the set of sequences approaches zero in the thermodynamic limit (see also section 4). Thus, the sort of temporal problems that require the presence of time-summating neurons (section 2) do not arise.

Another interesting example concerns the performance of a network when trained on noisy input sequences. Associate with each of the random sequences $\{S^\mu(t);\ t=1,\dots,R\}$ a set of noisy sequences that are generated according to the independent probabilities

$$\Pr[S_i(r)=S_i^\mu(r)]=(1\pm m)/2. \tag{3.18}$$

For large N, the probability of error is given by

$$\varepsilon(r)=\frac{1}{\alpha N}\sum_{\mu=1}^{\alpha N} H(\sqrt{a}\,\sigma^\mu(r)w\cdot\tilde{S}^\mu(r)/\sqrt{D_{rr}}) \tag{3.19}$$

where $a=m^2/(1-m^2)$; the derivation of equation (3.19) proceeds along similar lines to that of equation (3.7). Suppose that the network is trained by using equation (3.18)

to generate, for each μ, P example sequences $\{S^{\mu l}(r), r=1,\ldots,R\}$ and searching for a set of weights such that these noisy sequences are correctly classified,

$$\sigma^{\mu}(r)w\cdot\tilde{S}^{\mu l}(r)>0. \tag{3.20}$$

(This is analogous to the perceptron classification problem for static patterns that was considered in [15].) Define the average probability of error by

$$\bar{\varepsilon}(r)=\left\langle\!\!\left\langle \int \varepsilon(r)\,\mathrm{d}\mu(w)\right\rangle\!\!\right\rangle \tag{3.21}$$

where $\langle\!\langle\ldots\rangle\!\rangle$ indicates averaging over the distributions of input and output patterns, and the measure

$$\mathrm{d}\mu(w)=\frac{\Pi_j\,\mathrm{d}w_j\,\theta(\sigma^{\mu}(r)w\cdot\tilde{S}^{\mu l}(r))\delta(w^2-N)}{\int\Pi_j\,\mathrm{d}w_j\,\delta(w^2-N)} \tag{3.22}$$

determines the fractional volume in weight space of solutions to equation (3.20). For large P, $\bar{\varepsilon}(r)$ satisfies

$$\bar{\varepsilon}(r)=H(\sqrt{a}\,\lambda(r)/\sqrt{D_{rr}}) \tag{3.23}$$

where $\lambda(r)$ is the typical local field generated by the sequences $\{S^{\mu}(r), r=1,\ldots,R\}$ at time r. The average probability of error has a complicated dependence upon the decay parameter k. For, in addition to the explicit scale factor $1/\sqrt{D_{rr}}$, $\lambda(r)$ is an implicit function of off-diagonal elements of $\mathbf{D}$. In particular, the maximum number of examples P_c that may be classified without error, for fixed α, and the corresponding local field $\lambda_c(r)$ satisfy the saddlepoint equations of the free energy of the system at the critical point [14],

$$1=\alpha\sum_{r=1}^{R}\frac{\lambda_c(r)^2}{D_{rr}}+\alpha P_c\left\langle\int\mathscr{D}t\sum_I\Theta^I(t)\sum_{r,s\in I}D^{(I)}_{rs}(t_r-\sqrt{a}\,\lambda_c(r))(t_s-\sqrt{a}\,\lambda_c(s))\sigma(r)\sigma(s)\right\rangle_{\sigma}$$

$$\frac{\lambda_c(r)}{D_{rr}}=P_c\left\langle\int\mathscr{D}t\sum_{I:r\in I}\Theta^I(t)\sum_{s\in I}D^{(I)}_{rs}(t_s-\sqrt{a}\,\lambda_c(s))\sigma(r)\sigma(s)\right\rangle_{\sigma} \tag{3.24}$$

where $\Theta^I(t)=\Pi_{r\in I}\theta(t_r-\sqrt{a}\,\lambda(r))\Pi_{r'\notin I}\theta(\sqrt{a}\,\lambda(r')-t_{r'})$ and $\mathscr{D}t$ satisfies equation (3.16). It may be shown that the associated error $\bar{\varepsilon}(r)$ at the critical point is a monotonically increasing function of k, whereas P_c decreases with k. (A more detailed analysis will be presented elsewhere [14].)

4. Storage and retrieval of temporal sequences

In sections 2 and 3 we studied the associative learning of temporal sequences in feedforward networks. Another important issue is the storage and retrieval of temporal sequences and, in particular, the generation of a sequence of patterns given only the first few elements of the sequence. One of the most direct ways to store (periodic) binary sequences is to use an autoassociative network. Consider, for example, a fully connected network of N binary-threshold neurons and denote the output of the ith neuron at time t by $S_i(t)=\pm1$. The dynamics of the network is described by the equations (assuming synchronous update)

$$S_i(t+1)=\mathrm{sign}\left(\sum_{j\neq i}w_{ij}S_j(t)-h_i\right) \tag{4.1}$$

where w_{ij} is the connection from neuron j to neuron i and h_i is the threshold (which we shall set to zero for convenience). Since the total number of possible states of the network is finite (equal to 2^N), the dynamics of equation (4.1) is recurrent in the sense that the system returns to a state previously visited within a finite number of time steps. It follows that the long-term dynamics, in the absence of noise, is cyclic.

However, there are a number of problems with the above implementation: (*a*) The network can only store sequences in which there are no repeated states within a single cycle (simple sequences). Complex sequences, on the other hand, contain repeated states, which results in an ambiguity as to the successor of each of these states. (This is illustrated in figure 4.) The disambiguation of these sequences requires some form of memory extending over several time steps of a sequence. (*b*) Although systematic learning algorithms for the storage of temporal sequences in binary networks have been developed (e.g. see [16]), they tend to be rather cumbersome; hence the weights are often 'sculpted by hand'. (*c*) The network spends a single time step in each of the states of the sequence. Hence, there is no natural distinction between cycle states and transient states, which suggests that the entire repeated cycle should be considered as a single cognitive event rather than as a sequence of events.

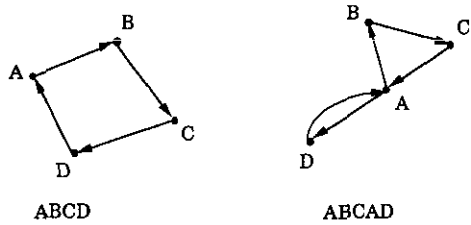

Figure 4. Example of a simple sequence *ABCD*... and a complex sequence *ABCAD*.... In the latter case there is an ambiguity concerning the successor of pattern *A*.

Problem (*c*) has motivated a different approach to temporal sequence storage based upon the framework of attractor networks [17]. A standard method for storing static patterns is to use a Hopfield network [18]. This is a binary network in which the dynamics is asynchronous; at each time step a single neuron i is chosen at random and its state updated according to equation (4.1). If the weights are symmetric, $w_{ij} = w_{ji}$, then there exists a Liapunov or energy function E which decreases along trajectories [18], with $E = -\Sigma_j w_{ij} S_i S_j/2$ (for zero thresholds h_i). Thus, the network converges to a stable fixed point corresponding to one of the minima of E. A set of p random unbiased patterns $\{\xi_i^\mu, i = 1, \ldots, N\}$, $\mu = 1, \ldots, p$ may be stored by taking the connection weights to be of the Hebbian form [18],

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu \tag{4.2}$$

so that each of the vectors $\boldsymbol{\xi}^\mu$ becomes a fixed point of the dynamics (assuming that the effects of cross-talk are negligible). Pattern $\boldsymbol{\xi}^\mu$ is recalled whenever the initial state of the network, $\boldsymbol{S}(0)$, lies within its particular basin of attraction.

The extension of attractor networks to the case of temporal sequence storage essentially involves the introduction of an antisymmetric contribution to the connection weights so that equation (4.2) becomes [8]

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu + \frac{\lambda}{N} \sum_{\mu=1}^{q} \xi_i^{\mu+1} \xi_j^\mu. \tag{4.3}$$

The effect of the first term on the right-hand side of equation (4.3), denoted $w_{ij}^{(s)}$, is to stabilize the pattern currently activated, whereas the second term, $w_{ij}^{(a)}$, induces transitions from one pattern to the next along the sequence (of length q). (The persistence or stabilization of each pattern of a stored sequence provides the mechanism for distinguishing between these patterns and transient states.) However, as it stands, such a strategy is too sensitive to the choice of value for λ; if λ is too small then the patterns remain completely stable, whereas if λ is too large then transitions occur too quickly. This problem may be avoided by taking the asymmetric weights $w_{ij}^{(a)}$ to have a slow dynamic response so that the activation state of the ith neuron at time t is given by

$$V_i(t) = \sum_{j \neq i} w_{ij}^{(s)} S_j(t) + \sum_{j \neq i} w_{ij}^{(a)} \int_0^\infty dt' \, C(t-t') S_j(t'). \tag{4.4}$$

More elaborate versions of such networks, which allow the storage of complex sequences, have also been developed [19]. It is interesting to note that equation (4.4) contains a sum over previous input activity that is similar in form to the expression for the activation state of the two-layer time-summating network, equation (2.2). Indeed, the two are equivalent if we set

$$C(t-t') = \sum_{r=1}^{t} \delta(t-t'-r) k^{r-1}. \tag{4.5}$$

One of the limitations of the above attractor networks is that the Hebbian-like learning rules adopted are suboptimal even for problems which are effectively linearly separable. A simple illustration of this is the result that, in the case of random, unbiased patterns, the storage capacity for the Hopfield model is $p = 0.14N$ [17] whereas the maximum possible storage capacity is $2N$ [10, 20]; in order to realize the optimal capacity it is necessary to use some form of perceptron-like learning algorithm [10]. Similarly, the requirement that each pattern of a sequence should be stabilized implies that the capacity for storage of sequences of random patterns in an attractor network has an upper bound of $0.14N/R$, where R is the sequence length. Therefore, we shall consider an alternative approach to the storage and retrieval of temporal sequences which uses, with some modifications, the perceptron learning rule for a two-layer time-summating network analysed in sections 2 and 3. (An earlier version of this approach was presented in [7].)

A schematic diagram of the learning phase is shown in figure 5. The input layer consists of N time-summating neurons with a linear output function and the output layer consists of N binary-threshold neurons. All neurons in the input layer are

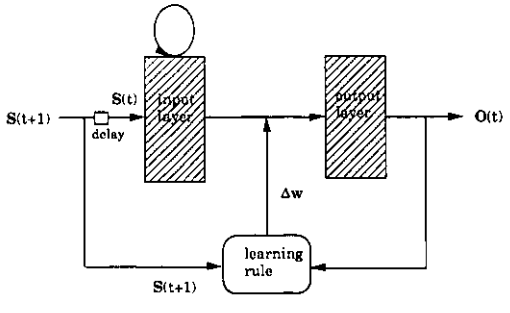

Figure 5. Schematic diagram of the learning phase of a two-layer time-summating network being trained to store a temporal sequence $\{S(t),\ t = 0, \ldots, R\}$.

connected to all neurons of the output layer. (This corresponds to the condition that an attractor network is fully connected.) Given an input sequence $\{S(t),\ t=0,\ldots,R\}$, the output of the ith neuron in the output layer at time t is

$$O_i(t)=\operatorname{sign}\left(\sum_{j\neq i} w_{ij}\sum_{r=0}^{t} k_j^{t-r}S_j(r)\right). \tag{4.6}$$

The network stores the sequence $\{S(t),\ t=0,\ldots,R\}$ by learning to output the pattern $S(t+1)$ on presentation of the patterns $\{S(0),\ldots,S(t)\}$, for $t=0,\ldots,R-1$. This may be achieved using a perceptron learning rule of the form (2.5), with the desired output at time t equal to the next pattern of the input sequence $S(t+1)$:

$$w_{ij}\to w_{ij}+(S_i(t+1)-O_i(t))\tilde{S}_j(t) \qquad \tilde{S}_j(t)=\sum_{r=0}^{t} k_j^{t-r}S_j(r). \tag{4.7}$$

The recall phase consists of feeding the output of the network back to the input layer. If the network is seeded with the first pattern of the sequence, then the full sequence will be generated (see figure 6). Note that the inclusion of time-summating neurons in the input layer allows the disambiguation of complex sequences due to the presence of an activity trace over previous inputs. Also note that, in this particular scheme, each sequence is uniquely labelled by the first pattern of the sequence. Thus, one cannot simultaneously store the sequences *AAB* and *AAC*, say. However, it is a simple matter to extend the system such that sequences are recalled by seeding the network with the first q patterns of each sequence.

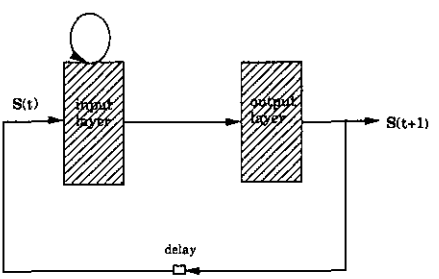

Figure 6. Recall of a temporal sequence by a two-layer time-summating network.

A convenient way to view the network of figures 5 and 6 is in terms of a set of N independent time-summating perceptrons, since the results of sections 2 and 3 may then be applied. Suppose that the network is required to store p pattern sequences $\{S^\mu(t),\ t=0,1,\ldots,R\}$. For each such sequence the ith perceptron must learn the R mappings, $\tilde{S}^\mu(t)\to S_i^\mu(t+1)$, $t=0,\ldots,R-1$. Following section 2, we divide the Rp inputs $\tilde{S}^\mu(t)$ into two sets F_i^+ and F_i^- where $\tilde{S}^\mu(t)\in F_i^+$ if $S_i(t+1)=1$ and $\tilde{S}^\mu(t)\in F_i^-$ otherwise. Temporal sequence storage then reduces to the problem of finding a set of weights $\{w_{ij},\ j\neq i\}$ such that the sets F_i^+ and F_i^- are separated for each $i=1,\ldots,N$ (cf equation (2.4)). By the perceptron convergence theorem, the learning algorithm (4.7) will find one such solution if it exists. In other words, the network is optimal with regards the storage of simple and complex sequences, provided that the corresponding sets F_i^+ and F_i^- are linearly separable. Note, however, that if F_i^+ and F_i^- are not linearly separable then it is necessary to introduce one or more hidden layers. Although the network may be trained using back-error-propagation [4], there is no longer any guarantee of convergence, so that the learning may not be optimal.

A simple example of a linearly separable problem is the storage of the two pattern sequences *AAB* and *BBA* where $A^t = (1, -1)$ and $B^t = (-1, 1)$. The corresponding sets F_1^+ and F_1^- associated with the first output neuron are shown in figure 7. We see that these sets are linearly separable provided that the decay rates of the two input neurons are different, $k \neq k'$. (A similar result holds for the sets F_2^+ and F_2^-.) This example also illustrates how the time-summating network of figure 5 can store a complex sequence.

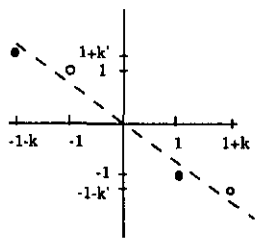

Figure 7. The sets F_1^+ and F_1^- corresponding to the sequences *AAB* and *BBA* where $A^t = (1, -1)$ and $B^t = (-1, 1)$.

As in section 3, we may extend the perceptron algorithm (4.7) to the case of a non-zero stability parameter η to ensure a certain degree of robustness. In order that the network can store the set of sequences $\{S^\mu(t), t = 0, \ldots, R\}$, $\mu = 1, \ldots, p$, we require that the weights w_{ij} satisfy the conditions (cf equation (3.9))

$$S_i^\mu(t+1)\left(\sum_{j\neq i} w_{ij} \sum_{r=0}^{t} S_j^\mu(r) k_j^{t-r}\right) > \eta\left(\sum_{j\neq i} w_{ij}^2\right)^{1/2}. \tag{4.8}$$

The learning rule corresponding to equation (3.10) is

$$w_{ij} \to w_{ij} + \varepsilon_i^{\mu,t} S_i^\mu(t+1)\tilde{S}_j^\mu(t) \tag{4.9}$$

where

$$\varepsilon_i^{\mu,t} = \theta\left[\eta\left(\sum_{j\neq i} w_{ij}^2\right)^{1/2} - \sum_{j\neq i} w_{ij} S_i^\mu(t+1)\tilde{S}_j^\mu(t)\right]. \tag{4.10}$$

The convergence of equation (4.9) then follows from section 3, since the network may once again be considered as a set of N independent time-summating perceptrons labelled by the site index $i = 1, \ldots, N$. Similarly, the maximum storage capacity of the network in the thermodynamic limit is given by equation (3.15). In particular, for $\eta = 0$, the storage capacity is $2N/R$.

Note that the analysis of optimal storage capacity simplifies considerably for a sparsely connected network in which the number of connections between the output layer and each neuron in the input layer is $O(\log N)$ [11]. For this network, the contribution from off-diagonal elements of $\mathbf{D}$ may be neglected and the capacity increases, rather than decreases, as a function of k when $\eta \neq 0$. That is, equation (3.15) reduces to the much simpler expression

$$\alpha_c^{(k)}(\eta) = \frac{1}{R}\sum_{r=1}^{R} \alpha_c^{(0)}(\eta/\sqrt{D_{rr}}) > \alpha_c^{(0)}(\eta) \qquad \eta > 0. \tag{4.11}$$

The inequality in equation (4.11) follows from the fact that $\alpha_c(\eta)$ is a monotonically decreasing function of η, and $D_{rr} > 1$. Hence, for a sparsely connected network, increasing k reduces the effective size of η.

The optimal storage capacity for temporal sequences has also been studied by Bauer and Krey using a Gardner-type analysis [21]. However, their mechanism for storing temporal sequences is based on a distribution of synaptic delays, rather than a slowly decaying activation state, and there is little overlap with the present work. In [21] the constraints analogous to (4.8) are

$$S_i^\mu(t+1)\left(\sum_{j\neq i} w_{ij}S_j^\mu(t-\tau_{ij})\right) > \eta \tag{4.12}$$

where τ_{ij} is the synaptic delay associated with connection (i, j). Moreover, each pattern of a sequence is required to persist for a number of time steps Δ. The main consideration in [21] is the effect of different choices of τ_{ij} on the optimal storage capacity.

One possible disadvantage of our scheme for storing and retrieving temporal sequences (figures 5 and 6) is that there is no intrinsic mechanism for distinguishing between patterns corresponding to stored sequences and arbitrary patterns (cf problem (c) discussed at the beginning of section 4). Therefore, one might need to supplement the network with some form of novelty detector. Finally, note that it is possible to combine the various approaches discussed in this section by taking the output layer of figure 5 to be an attractor network [22]. In this particular scheme a pattern from the time-summating input layer sets the initial state of the output layer; the neurons of the output layer then relax to some fixed point corresponding to a stored pattern, which is subsequently fed back to the input layer, and the whole process repeated. There are two distinct sets of weights; the interlayer connections encode transitions between one pattern and the next within a sequence, whereas the intralayer connections stabilize the patterns which constitute the various sequences. The presence of time-summating neurons in the input layer, with the possible addition of one or more hidden layers, allows complex sequences to be stored. Note, however, that careful account would need to be taken of the various timescales of the system. (We hope to explore this further elsewhere.)

5. Discussion

In this paper we have shown how a two-layer time-summating neural network may be trained to associate and store temporal sequences using perceptron-like learning algorithms. The presence of the time-summating input layer enables the network to handle temporal features such as ordering and coarticulation effects and to disambiguate complex sequences. However, the effects of intrinsic temporal correlations and the possible accumulation of errors must be carefully taken into account when analysing the performance of such networks. There are number of extensions of this work:

(i) The development of learning algorithms that determine the decay rates k_i as well as the weights between the input and output layers. Since the k_is may be viewed as weights associated with additional feedback lines, they may be determined by a straightforward extension of back-error-propagation [23].

(ii) The use of statistical-mechanical techniques to analyse the critical capacity of a time-summating network with a range of decay rates k_i and complex input sequences. (Recall that the random sequences considered in section 3 are simple, since the probability of a repeated pattern occurring within a finite sequence approaches zero in the thermodynamic limit.)

(iii) The analysis of the storage and classification of temporal sequences in a three-layer neural network with a time-summating hidden layer. This would involve a number of features. First, the generalization of the geometric approach based on linear separability which was used to investigate ordering and coarticulation effects in section 2. Second, the application of statistical-mechanical techniques recently developed for standard multilayer networks [24].

References

[1] Rumelhart D E and McClelland J L 1986 *Parallel Distributed Processing* (Cambridge, MA: MIT Press)
[2] Lang K J, Waibel A H and Hinton G E 1990 *Neural Networks* **3** 23
[3] Lyon R F and Mead C 1988 *IEEE Trans. Acoust., Speech, Signal Processing* **36** 1119
[4] Mozer M C 1987 *Complex Systems* **3** 349
[5] Watrous R L and Shastri L 1987 *Proc. IEEE 1st Int. Conf. on Neural Networks (San Diego, CA, June 1987)* vol II, p 619
Jordan M I 1987 *Proc. 8th Ann. Conf. of the Cognitive Science Society (Hillsdale, NJ, 1987)* p 531
[6] Stornetta W S, Hogg T and Huberman B A 1987 *Neural Information Processing Systems* ed D Z Anderson (New York: American Institute of Physics)
[7] Taylor J G and Reiss M 1991 *Neural Networks* **4** 773
[8] Kleinfeld D 1986 *Proc. Natl Acad. Sci. USA* **83** 9469
Sompolinsky H and Kanter I 1986 *Phys. Rev. Lett.* **57** 2861
[9] Minsky M and Papert S 1969 *Perceptrons* (Cambridge, MA: MIT Press)
[10] Gardner E 1988 *J. Phys. A: Math. Gen.* **21** 257
[11] Taylor J G 1991 *Int. J. Neural Syst.* **2** 47
[12] Bressloff P C and Taylor J G 1992 *J. Phys. A: Math. Gen.* **25** 833
[13] Morgan D P and Scofield C L 1991 *Neural Networks and Speech Processing* (Dordrecht: Kluwer)
[14] Bressloff P C 1991 Learning temporal sequences from examples in a time-summating network *Preprint* GEC-Hirst
[15] Hansel D and Sompolinsky H 1990 *Europhys. Lett.* **11** 687
[16] Guyon I, Personnaz L, Nadal J P and Dreyfus G 1988 *Phys. Rev.* A **38** 6365
[17] Amit D J 1989 *Modeling Brain Function* (Cambridge: Cambridge University Press)
[18] Hopfield J J 1984 *Proc. Natl Acad. Sci. USA* **81** 3088
[19] Herz A V M, Sulzer B, Kuhn R and van Hemmen J L 1988 *Europhys. Lett.* **7** 663; 1989 *Biol. Cybern.* **60** 457
[20] Cover T M 1965 *IEEE Trans. Electron. Comput.* **EC-14** 326
[21] Bauer M and Krey U 1991 *Z. Phys.* B **84** 131
[22] Reiss M and Taylor J G 1991 *Proc. 2nd IEE Conf. on Artificial Neural Networks (Bournemouth: November 1991)* p 129
[23] Bressloff P C 1992 *Mathematical Studies of Neural Networks* (Amsterdam: Elsevier) in press
[24] Meir R and Domany E 1988 *Phys. Rev.* A **37** 608
Barkai E, Hansel D and Kanter I 1990 *Phys. Rev. Lett.* **65** 2312