Dynamic Proximity of Spatiotemporal Patterns

David Horn
School of Physics and Astronomy
Raymond and Beverly Sackler
Faculty of Exact Sciences
Tel Aviv University, Tel Aviv 69978, Israel
Email: horn@post.tau.ac.il

Gideon Dror
Department of Computer Science
The Academic College
of Tel-Aviv-Yaffo
Tel Aviv 64044, Israel
Email: gideon@mta.ac.il

Brigitte Quenet
Lab. d'Electronique
Ecole Superieure de Physique
et Chimie Industrielles
Paris 75005, France
Email: brigitte.quenet@espci.fr

***Abstract*— Recurrent networks can generate spatiotemporal neural patterns that may look quite chaotic. Nonetheless a proximity measure between these patterns may be defined through comparison of the synaptic weight matrices that generate them. Following the Dynamic Neural Filter (DNF) formalism we demonstrate this concept by comparing teacher and student recurrent networks. We define an SVM variant of DNF in order to specify the optimal weight matrix and to deal with noise.**

I. Introduction

A recent paper [1] has introduced the concept of Dynamic Neural Filters (DNF), recurrent networks projecting input space into spatiotemporal behavior. The one-step dynamics of a binary recurrent network of N neurons is defined by

$$n_i(t+1) = H(h_i(t+1)) = H(\sum_j w_{ij} n_j(t) + b_i) \qquad (1)$$

where w_{ij} is the synaptic coupling matrix and b_i are the biases of the system [1] . H is the Heaviside step function taking the values 0 for negative arguments and 1 for positive ones. The initial condition is defined by the null state $n_i(0) = 0$.

For a DNF of N neurons with a randomly chosen synaptic matrix (e.g. $w_{ij} \sim \mathcal{N}(0,1)$) and small biases ($b_i \sim \mathcal{N}(0, 1/\sqrt{N})$) the dynamics of Eq. (1) lead to very complex spatiotemporal sequences, as will be demonstrated below.

An interesting problem is the reconstruction of the network from a given spatiotemporal sequence. This can be resolved [1] by a perceptron algorithm, recaptured below, and will be studied numerically in the next section. The given series has to obey a condition that stems from linear separability. This may be guaranteed under certain conditions (see in subsection B below) but is easily obtained if a teacher DNF is employed to create the sequence. This is the method we use in section 2 where we study an $N = 40$ example.

A. The Perceptron Algorithm

Here we recapture a straightforward algorithm for building the DNF from a given sequence [1]. Let us start by defining, for each neuron i, an $N+1$ dimensional vector of perceptron weights $\vec{w}^i$

$$(\vec{w}^i)_j = w_{ij} \quad \text{for } j = 1, \cdots N \qquad (\vec{w}^i)_{N+1} = b_i. \qquad (2)$$

[1]In [1] we have used $b_i = R_i - \theta_i$, putting emphasis on the inputs that the neurons receive. For simplicity we use only one parameter here.

Let us define vectors $\vec{x}^i(t)$, for each neuron i, as follows

$$(\vec{x}^i(t))_j = n_j(t)(2n_i(t+1) - 1) \quad \text{for } j = 1, \cdots N \ , \qquad (3)$$

$$(\vec{x}^i(t))_{N+1} = (2n_i(t+1) - 1).$$

The vectors $\vec{x}^i(t)$ represent the state of all neurons that form the input to neuron i, weighted by a positive or negative sign depending on whether the target neuron i at the next time step is 1 or 0 respectively. With these definitions, all constraints of the problem at hand can be written as T perceptron inequalities

$$\vec{w}^i \cdot \vec{x}^i(t) > 0 \quad \text{for all } t = 0, 1, \cdots T-1. \qquad (4)$$

Now one can use the perceptron learning rule [2], [3]

$$\Delta\vec{w}^i = \eta \vec{x}^{i,k}(t) H(-\vec{w}_i \cdot \vec{x}^{i,k}(t)) \qquad (5)$$

while iterating the system, time and again, over all T states. The Heaviside function guarantees that $\vec{w}^i$ gets modified at a given iteration by the vectors $\vec{x}^i(t)$ that do not satisfy the inequality. This algorithm converges [2] if the system of inequalities is soluble.

B. Network Size

It is straightforward [1] to derive an estimate of the size N of the network required to generate a sequence of length T. Each neuron has to satisfy T inequalities in an $N+1$ dimensional space. Hence, strict matching for any set of such data requires (see, e.g., [2])

$$\mathcal{A}: \qquad N \geq T - 1. \qquad (6)$$

In the large N limit one can apply the Cover result[4], saying that a solution may be found for

$$\mathcal{B}: \qquad T \leq 2(N+1) \qquad \text{or} \qquad N \geq \frac{1}{2}(T-2). \qquad (7)$$

This would be the case when the sequence is constructed of random states. Clearly, if the sequence is generated by a (teacher) recurrent network, we expect generalization to set in at some stage, where a fixed value of N may describe the sequence for very many time steps.

The question of the minimal size of a recurrent network, and its use as a measure of complexity of data, is discussed in [1].

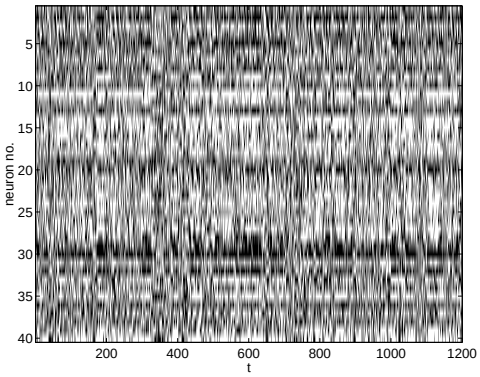

Fig. 1. A raster plot of an $N = 40$ DNF.

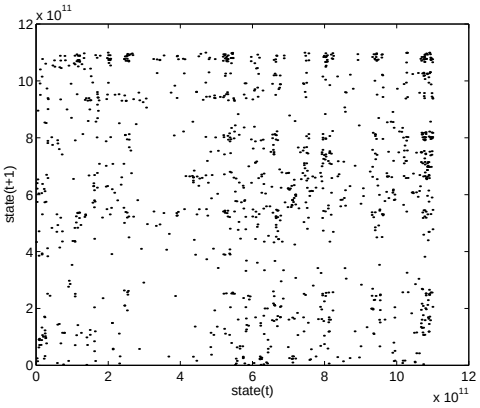

Fig. 2. Return map of an $N = 40$ spatiotemporal sequence.

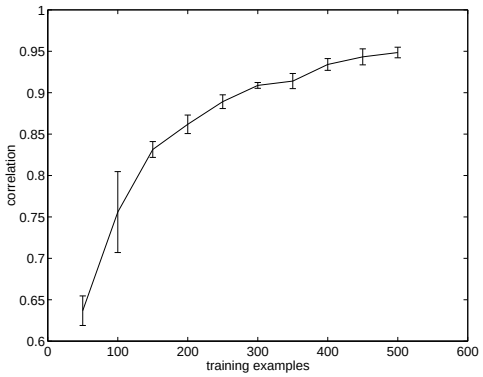

Fig. 3. The correlation between teacher and student DNF in the $N = 40$ problem, using the perceptron algorithm on 10 different teacher-student pairs.

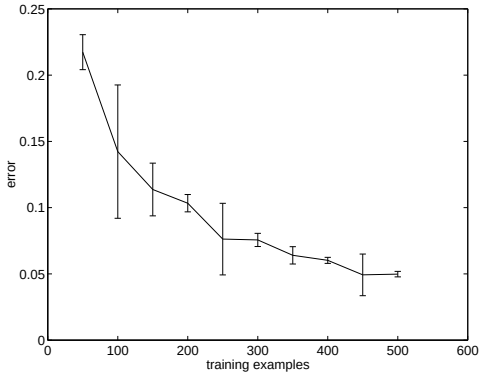

Fig. 4. Convergence of the errors of student networks on one-step prediction tasks.

II. Numerical studies

We run a network of $N = 40$ neurons with $w_{ij} \sim \mathcal{N}(0,1))$ and $b_i \sim \mathcal{N}(0, 1/\sqrt{N})$. An example of a resulting raster plot of this system is shown in Fig. 1. This is the kind of spatiotemporal pattern we wish to study. To each time step of this system there corresponds a spatial state (activity of all N neurons) that can be represented by the number $1 + \sum_{i=1}^{N} n_i 2^{i-1}$ where n_i, the activity of the ith neuron, is either 0 or 1. Thus we can generate a return map of these states, shown in Fig. 2. The disorder in this map is characteristic of these sequences. Checking 1200 time steps, we find that the sequence never repeats any state, i.e. it belongs to a cycle that is larger than 1200.

To recapture the structure of the DNF we use the perceptron algorithm. Thus the DNF that served to generate the sequence may be regarded as a teacher network training DNF students. The results of this training are shown in Fig. 3 in the form of correlations of the generalized $\mathbf{w}$ matrix, i.e. averages of the correlations of all vectors $\vec{w}^i$ of Eq. 2 between teacher and student recurrent networks. The training converges nicely after a few hundred steps. The error bars represent standard deviations over 10 different student-networks, trained on 10 different sequences of the type shown in Figs. 1 and 2. In Fig. 4 we present average prediction errors of the 10 networks, comparing one-step predictions of students to the teacher on all neurons. These errors converge rapidly as shown in Fig. 4.

Thus we see that the given temporal sequence of Fig. 1 defines, through the perceptron algorithm, a quite unique $\mathbf{w}$ matrix. Its uniqueness depends on the size of the training set, i.e. the number of time steps of the given sequence. The smaller this size, the wider will be the range of $\mathbf{w}$ that can reproduce a given sequence. One should notice that, in any case, the matrix $\mathbf{w}$ is defined only up to a set of N scale transformations. As can be seen from Eq. 1, every factor multiplying any $\vec{w}^i$ of Eq. 2, leads to the same dynamics. This arbitrariness of scale is factored out once we calculate correlations as in Fig. 3.

It is important to realize that sequences may be very uncorrelated in their spatiotemporal structure and yet may be very close in their generating $\mathbf{w}$s. Consider a sequence of the type shown in Figs. 1 and 2. Generating a $\mathbf{w}$ from its first 500 time steps, and another $\mathbf{w}$ from the next 500 steps, we obtain correlations close to 1 between these two $\mathbf{w}$s, as expected from Fig. 3. Yet the two spatiotemporal patterns are completely uncorrelated as far as their neural realization is concerned, as expected from Fig. 2. In other words, while Hamming distance shows no proximity the dynamics are as close as they can be! This exemplifies the power of the DNF as a tool for defining proximity between spatiotemporal patterns.

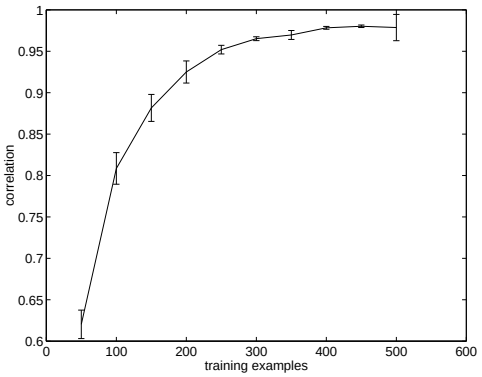

Fig. 5. The correlation between teacher and student DNF using the SVM learning algorithm.

III. The Optimal Network

The analysis of the previous section leads to the question if, for a given sequence, there exists an optimal choice of $\mathbf{w}$. Optimality, or stability toward small (noisy) change of data, can be obtained by using the maximal margin approach of SVM [5], [6]. This is an alternative to the perceptron algorithm discussed in the Introduction.

A teacher DNF produces a sequence that is linearly separable. The data that the student network is presented with consists of all input vectors $\vec{n}(t)$, with t varying from 0 to $T-1$, and the output (target) values are specified by $\vec{n}(t+1)$. The SVM formulation of the resulting $\mathbf{w}$ consists of

$$\vec{w}^i = \sum_L \alpha_L^i y_L^i \vec{n}^L \tag{8}$$

where $\vec{n}^L$ are all states that appear in the sequence, and a few positive α_L^i select out of them the support vectors appropriate for the ith neuron problem. The coefficients y_L^i are the appropriate targets, i.e. if $\vec{n}^L$ is the input to neuron i at time t then $y_L^i = 2n_i(t+1) - 1$.

We have employed the SVM algorithm to solve the same problem of Fig. 3, where we have used the perceptron algorithm. The results are shown in Fig. 5. Although the results of the SVM algorithm are slightly better, their general trend is the same as in the perceptron case. The error bars represent the standard deviations of 10 different teacher and student networks. Also here the correlations start out low, because, whereas the student finds an optimal solution the teacher is not optimal. In fact, a DNF with a given $\mathbf{w}$, has a family of different optimal solutions as function of the length T of the sequence that it generates. Only in the limit $T \to \infty$ they converge, as shown in Fig. 5.

IV. Hidden Neurons

Sequences or spatiotemporal patterns that are not generated by a DNF may not be amenable to DNF representations that are limited to the same number N of observed neurons. There may be two sources of disagreement: 1. Occurrence of contradictory repetitions, i.e. $\vec{n}(t_1) = \vec{n}(t_2)$ while $\vec{n}(t_1+1) \neq \vec{n}(t_2+1)$. 2. Occurrence of XOR-like configurations that violate the linear separability condition that is implied by the dynamics of Eq. 1. Both problems can be resolved by introducing hidden neurons [1], thus effectively increasing N to some new value N', while the observed N neurons continue to represent the observed sequence. No algorithm is known that specifies a learning procedure guaranteeing a minimal N'. Note that here one has not only to learn the large synaptic matrix but one has also to choose the values $n_i(t)$ of the hidden neurons during the time steps of the available sequence. In general we know that for any spatiotemporal sequence of length T a representation like that exists under the conditions discussed in subsection B of the Introduction.

An alternative resolution of the 2nd problem, i.e. accommodating data that do not obey linear separability, is to make use of the kernel formulation of SVM [5], [6]. This can be obtained by a suitable generalization of Eq. 8, searching for solutions of the type

$$n_i(t+1) = H(\sum_L \alpha_L^i y_L^i K(\vec{n}^L, \vec{n}(t)) + b_i) \tag{9}$$

where the kernel function K replaces the Euclidean scalar product $\vec{w}^i \cdot \vec{n}(t)$. Appropriate kernels can be powers $K(\vec{a}, \vec{b}) = (\vec{a} \cdot \vec{b} + 1)^p$ or Gaussians $K(\vec{a}, \vec{b}) = e^{\frac{(\vec{a}-\vec{b})^2}{2\sigma^2}}$ [5], [6]. Once the separability of a series is resolved, Eq. 9 can be used as the definition of the resulting recurrent support vector network (RSVN).

There exists however still the first problem, that of contradictory repetitions, when the same input vector, occurring at different time steps, leads to different outputs, i.e. when y_L^i is ill defined. When such a situation occurs one has to invoke hidden neurons also here. It is however straightforward to decide on their minimal number since all that is needed is to guarantee the disappearance of contradictory repetitions.

V. The Noisy Spatiotemporal Sequence

An interesting case is that of a spatiotemporal sequence that is DNF generated but is subject to a noisy channel, inverting some fraction of its bits. The question is if, given such a sequence, one can rebuild the original $\mathbf{w}$.

Clearly some of the errors will just mislead the learning algorithm, but others may introduce violations of linear separability. The latter pose a problem for the perceptron algorithm, but can be readily handled by the SVM formalism by introducing slack variables leading to an upper bound on $0 \leq \alpha_L^i \leq C$ [5], [6]. This is an alternative to introducing kernels or hidden neurons and, in the scenario of a noisy DNF, can end up correlating well with the original $\mathbf{w}$.

In Fig. 6 we display the effect of noise on the teacher-student task of recovering the synaptic matrix $\mathbf{w}$. We employed the same methodology as described in previous experiments, using the SVM algorithm with a constant slack parameter $C = 1$. As can be seen, this system works quite well, leading to very small errors even for noise of order 1/40, i.e. one expected flip in each state of $N = 40$ neurons. The error bars reflect the standard deviation of ten trials of teacher-student sessions. The number of training samples was held fixed at 250.

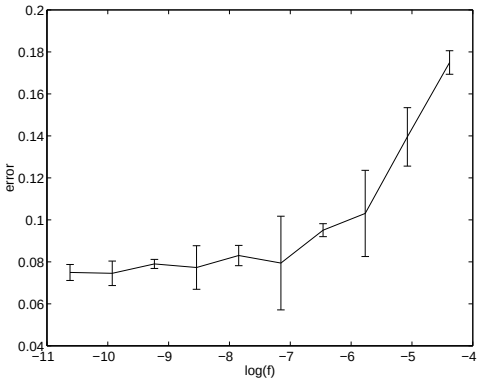

Fig. 6. Error of student performance in an $N = 40$ problem as function of the error in its training sequence of length 250. The highest error value corresponds to 1/40, i.e. one flip on average for each state of the trained system.

VI. Conclusion

The DNF serves as a general framework for the generation of binary spatiotemporal patterns. $N = 40$ is a large enough system to demonstrate very large cycles (over 1000), i.e. almost chaotic patterns. Although such patterns, and even different sections of the same patterns, look quite different from one another, there exists a dynamical proximity measure in terms of the synaptic matrices $\mathbf{w}$ that generate them.

In our numerical studies we have demonstrated such relationships. Each given spatiotemporal pattern may be regarded as a teacher and used for training student networks. We have demonstrated the correlations between $\mathbf{w}$s of teacher and student networks using two algorithms: the perceptron algorithm of [1] and an SVM algorithm suggested here. The latter may serve to define an optimal $\mathbf{w}$ in the sense of providing the largest margin in any linearly separable classification algorithm.

The spatiotemporal patterns generated by DNF recurrent networks obey linear separability. This points to a weakness of the formalism: Given a spatiotemporal sequence that does not obey these conditions, one has to invoke hidden neurons in order to embed it in a DNF. The optimal procedure for such embedding with hidden neurons is still lacking. SVM allows one to overcome linear separability by introducing a kernel approach, however hidden neurons may not be avoided if contradictory repetitions occur in the original sequence.

References

[1] B. Quenet and D. Horn, 2003. The Dynamic Neural Filter: A Binary Model of Spatiotemporal Coding. *Neural Computation* **15** 309–329.

[2] Hertz, J., Krogh, A. & Palmer, R., G., 1991. *Introduction to the Theory of Neural Computation.* Addison Wesley Pub. Comp.

[3] F. Rosenblatt, 1962. *Principles of Neurodynamics.* New York: Spartan.

[4] T. M. Cover, 1965. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE trans. Elect. Comp.* **14** 326-334.

[5] V. Vapnik 1995. *The Nature of Statistical Learning Theory*, Springer, New-York.

[6] C. J. C. Burges 1998. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 2(2).