PERGAMON

Neural Networks 12 (1999) 237–245

Neural Networks

Contributed article

Embedding recurrent neural networks into predator–prey models

Yves Moreau[a,*], Stéphane Louiès[b], Joos Vandewalle[a], Léon Brenig[b]

[a]*Katholieke Universiteit Leuven, Department of Electrical Engineering, Kardinaal Mercierlaan 94, B-3001 Leuven, Belgium*
[b]*Université Libre de Bruxelles, Boulevard du Triomphe, B-1050 Brussels, Belgium*

Received 23 February 1998; accepted 20 May 1998

Abstract

We study changes of coordinates that allow the embedding of ordinary differential equations describing continuous-time recurrent neural networks into differential equations describing predator–prey models—also called Lotka–Volterra systems. We transform the equations for the neural network first into quasi-monomial form (Brenig, L. (1988). Complete factorization and analytic solutions of generalized Lotka–Volterra equations. *Physics Letters A, 133*(7–8), 378–382), where we express the vector field of the dynamical system as a linear combination of products of powers of the variables. In practice, this transformation is possible only if the activation function is the hyperbolic tangent or the logistic sigmoid. From this quasi-monomial form, we can directly transform the system further into Lotka–Volterra equations. The resulting Lotka–Volterra system is of higher dimension than the original system, but the behavior of its first variables is equivalent to the behavior of the original neural network. We expect that this transformation will permit the application of existing techniques for the analysis of Lotka–Volterra systems to recurrent neural networks. Furthermore, our results show that Lotka–Volterra systems are universal approximators of dynamical systems, just as are continuous-time neural networks.

Keywords: Continuous-time neural networks; Equivalence of dynamical systems; Lotka–Volterra systems; Predator–prey models; Quasi-monomial forms

1. Introduction

Although major advances have been made, the field of application of dynamical system theory to recurrent neural networks still has much uncharted territory. Cohen and Grossberg studied a large class of competitive systems (which include many types of neural networks) (Cohen and Grossberg, 1983) with a symmetric interaction matrix for which they were able to construct a Lyapunov function guaranteeing global convergence. Hirsch considered cascades of networks and proved a cascade decomposition theorem (Hirsch, 1989) giving conditions under which convergence of the individual networks in isolation guarantees the convergence of the whole cascade. For systems with inputs, Sontag, Sussman and Albertini have studied their property from the point of view of nonlinear control system theory (Albertini and Sontag, 1993; Sontag and Sussman, 1997). But for the analysis of many properties of recurrent networks, simple computational tools are still missing.

Recently, we have started to investigate how changes of coordinates allow us to find equivalences between different neural networks. We have shown (Moreau and Vandewalle, 1997) how to use linear changes of coordinates to compute a transformation that maps a continuous-time recurrent neural network with a hidden layer onto a neural network without a hidden layer and with an extra output map. In the present paper, we will show how to embed different types of neural networks (having the hyperbolic tangent or the logistic sigmoid as activation function) into predator–prey models (also called Lotka–Volterra systems) of the form:

$$\dot{z}_i = \lambda_i z_i + z_i \sum_{j=1}^{m} M_{ij} z_j, \text{ with } j = 1, \ldots, m.$$

The Lotka–Volterra system will be of higher dimension than the original system but its n first variables will have a behavior equivalent or identical to that of the original system (if n was the number of variables of the original system).

Such a Lotka–Volterra representation of neural networks is of interest for several reasons. Lotka–Volterra systems are a central tool in mathematical biology for the modeling of competition between species and a classical subject of dynamical system theory (MacArthur, 1969; May and Leonard, 1975). They have therefore been the object of intense scrutiny. These systems have simple quadratic nonlinearities (not all quadratic terms are possible in each equation), which might be easier to analyze than the

* Corresponding author; e-mail: yves.moreau@esat.kuleuven.ac.be

0893-6080/99/$ - see front matter
PII: S0893-6080(98)00120-8

sigmoidal saturations of the neural networks. We thus expect that this representation will allow us to apply methods and results from the study of Lotka–Volterra systems to neural networks. In particular, we think about applying conditions for the global asymptotic stability of an equilibrium developed for quasi-monomial systems and Lotka–Volterra systems (Gouzé, 1990; Redheffer and Walter, 1984) to neural networks. Also, early work by Grossberg (1978) exploited one of the changes of variables that we present here to study convergence and oscillations in generalized competitive networks and generalized Lotka–Volterra systems. Furthermore, since we know that dynamical neural networks can approximate arbitrary dynamical systems (Sontag, 1992; Funahashi and Nakamura, 1993) for any finite time because of their equivalence with dynamical single hidden layer perceptrons (Moreau and Vandewalle, 1997), our result also serves as a simple proof that Lotka–Volterra systems enjoy the same approximation property; and we might want to investigate them further from that point of view.

We have structured this work as follows. First, we will consider the simple case of a dynamical perceptron. We will transform it into what we call a quasi-monomial form in a first step. This form will allow us to further transform the system into a Lotka–Volterra system. Second, we will consider the case of the dynamical perceptron with self-connections—that is, where each neuron has a linear connection to itself. We will perform the same transformations, first into a quasi-monomial form, and then into a Lotka–Volterra system. Finally, we will show that the same transformations apply to much more general types of continuous-time recurrent neural networks with hyperbolic tangent or logistic activation functions. In Appendices A and B, we describe in detail the structure of the quasi-monomial forms for the dynamical perceptron and the dynamical perceptron with self-connections. In Appendix C, we clarify what the exact constraints on the activation function are and we show that both the hyperbolic tangent and the logistic sigmoid satisfy these.

2. The dynamical perceptron

We consider first an n-dimensional neural network of the following form, which we call a dynamical perceptron:

$$\dot{x} = \vec{\sigma}(Ax + a_0), \tag{1}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, $a_0 \in \mathbb{R}^n$, and $\vec{\sigma}$ is defined as $\vec{\sigma}(x) = (\sigma(x_1), \sigma(x_2), \ldots, \sigma(x_n))^T$ with σ a continuous sigmoidal function from $\mathbb{R}$ to $\mathbb{R}$ such that $\lim_{x\to-\infty} \sigma(x) = a$ and $\lim_{x\to\infty} \sigma(x) = b$ ($a, b \in \mathbb{R}$ and $a < b$).

We consider this type of system because its simplicity makes our calculations easier to follow, yet it permits us to illustrate the essential points of our method. In this paper, we do all developments for the special case of $\sigma(x) = \tanh(x)$. Another possibility is the use of the logistic sigmoid $\sigma(x) = 1/(1 + e^{-x})$. We will later discuss what constrains the choice of the activation function. For the dynamical perceptron Eq. (1), the choice of the logistic sigmoid would be meaningless, since the vector field would then have positive components everywhere and the topology of the flow would be trivial (the hyperbolic tangent guarantees, on the other hand, the existence of an equilibrium). For the types of networks we describe in the following sections, this restriction to the hyperbolic tangent does not apply. But since the developments depend on the choice of the activation function, we limit ourselves to the hyperbolic tangent; all developments can be done similarly for the logistic sigmoid.

We first apply the change of coordinates $y = Ax + a_0$ (often called Sigma–Pi transformation (Grossberg, 1988)) and get the following equation:

$$\dot{y} = A\sigma(y).$$

As long as A is invertible (A is $n \times n$), we can directly revert the change of coordinates. Moreover, if A is singular, we in fact have an r-dimensional dynamics where r is the rank of the matrix A. Indeed, if we look at two different trajectories $\bar{x}(t)$ and $\tilde{x}(t)$ of the system $\dot{x} = \sigma(Ax + a_0)$ having the property that their initial conditions $\bar{x}_0$ and $\tilde{x}_0$ are such that $A\bar{x}_0 = A\tilde{x}_0$, the initial conditions $\bar{y}_0 = A\bar{x}_0 + a_0$ and $\tilde{y}_0 = A\tilde{x}_0 + a_0$ for the system $\dot{y} = A\sigma(y)$ are equal. The equality of these initial conditions implies that $\bar{y}(t) = \tilde{y}(t)$ for all times. But the vector field of the initial system is $\dot{x} = \sigma(Ax + a_0) = \sigma(y)$, thus $\dot{\bar{x}}(t) = \dot{\tilde{x}}(t)$ for all times. The equality of these derivatives implies that the trajectories $\bar{x}(t)$ and $\tilde{x}(t)$ through $\bar{x}_0$ and $\tilde{x}_0$ always remain parallel. Since the kernel of A is of dimension $n - r$, we see that the dynamics is in fact r-dimensional. If we define $\delta(x_0) = x_0 - A^{\dagger}Ax_0$ where $A^{\dagger}$ is the Moore–Penrose pseudo-inverse of A, then $x(t) = A^{\dagger}(y(t) - a_0) + \delta(x_0)$ with the initial condition $y_0 = Ax_0 + a_0$.

From here on, it will be necessary to look at the equations, coordinate by coordinate, so we have

$$\dot{y}_i = \sum_{j=1}^{n} A_{ij}\sigma(y_j), \text{ with } i = 1, \ldots, n. \tag{2}$$

Now, we decide to apply the change of coordinates $z_i = \sigma(y_i)$; we get

$$\dot{z}_i = \sigma'(y_i)\dot{y}_i. \tag{3}$$

It is here that we make use of our special choice of activation function, $\sigma(y) = \tanh(y)$. The essential property of the activation function we need is that we can express its derivative as a polynomial of the activation function:

$$\sigma'(y) = 1/\cosh^2(y) = 1 - \tanh^2(y) = 1 - \sigma^2(y). \tag{4}$$

Substituting Eqs. (4), (2) in Eq. (3), we have

$$\dot{z}_i = (1 - z_i^2) \sum_{j=1}^{n} A_{ij} z_j. \tag{5}$$

We see that this form begins to resemble the Lotka–Volterra model except for the factor $(1 - z_i^2)$ instead of z_i. The system is therefore polynomial of degree three. However, the vector field of the system has now become a linear combination of products of powers of the variables, which is what we call a quasi-monomial form.

Example. We will follow our developments by the use of an example to clarify the procedure. We look at the case $n = 3$. The equations of the recurrent neural network are

$$\begin{cases} \dot{x}_1 &= \tanh(A_{11}x_1 + A_{12}x_2 + A_{13}x_3 + A_{10}) \\ \dot{x}_2 &= \tanh(A_{21}x_1 + A_{22}x_2 + A_{23}x_3 + A_{20}) \\ \dot{x}_3 &= \tanh(A_{31}x_1 + A_{32}x_2 + A_{33}x_3 + A_{30}). \end{cases}$$

We can check that the equations become, after the affine and sigmoidal changes of coordinates,

$$\begin{cases} \dot{z}_1 &= (1 - z_1^2)(A_{11}z_1 + A_{12}z_2 + A_{13}z_3) \\ \dot{z}_2 &= (1 - z_2^2)(A_{21}z_1 + A_{22}z_2 + A_{23}z_3) \\ \dot{z}_3 &= (1 - z_3^2)(A_{31}z_1 + A_{32}z_2 + A_{33}z_3). \end{cases} \tag{6}$$

3. Transformation into a quasi-monomial form

By quasi-monomial form, we mean a dynamical system that we can write as

$$\dot{z}_i = z_i \Big(\sum_{j=1}^{r} \bar{A}_{ij} \prod_{k=1}^{n} z_k^{B_{jk}}\Big), \text{ with } i = 1, \ldots, n. \tag{7}$$

The number of different products of powers of the variables is r and we define $\bar{A} = \left[\bar{A}_{ij}\right] \in \mathbb{R}^{n \times r}$ and $B = \left[B_{jk}\right] \in \mathbb{R}^{r \times n}$. It is important to note that the exponents B_{jk} are not restricted to integer values, hence the name quasi-monomial form. Now, to reach this form, we write Eq. (5) as

$$\dot{z}_i = z_i \Big[\sum_{j=1}^{n} A_{ij}(z_i^{-1} z_j - z_i z_j)\Big]. \tag{8}$$

We then need to collect all possible monomials and use their exponents to construct the matrix B and collect all the coefficients of the monomials correspondingly to form the matrix $\bar{A}$. We perform this computation in a systematic fashion in Appendix A and illustrate it in the following example. Brenig (1988) has previously shown that we can transform such a quasi-monomial form into a Lotka–Volterra system.

Example (continued). To illustrate this somewhat obtuse notation (7), we take again the case $n = 3$. Expanding the expression in Eq. (6), we get

$$\begin{cases} \dot{z}_1 &= z_1(A_{11} - A_{11}z_1^2 + A_{12}z_1^{-1}z_2 - A_{12}z_1z_2 + A_{13}z_1^{-1}z_3 + A_{13}z_1z_3) \\ \dot{z}_2 &= z_2(A_{21}z_1z_2^{-1} - A_{21}z_1z_2 + A_{22} - A_{22}z_2^2 + A_{23}z_2^{-1}z_3 - A_{23}z_2z_3) \\ \dot{z}_3 &= z_3(A_{31}z_1z_3^{-1} - A_{31}z_1z_3 + A_{32}z_2z_3^{-1} - A_{32}z_2z_3 + A_{33} - A_{33}z_3^2). \end{cases}$$

We see that there are $2n^2 = 18$ terms, but only $r = (3n^2 - n + 2)/2 = 13$ different monomials. We discuss in Appendix A how to compute the exact number of monomials. The matrix B is then

$$B = \begin{pmatrix} 2 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 & 1 & 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 2 & 0 & 1 & 0 & 1 & -1 & -1 & 0 \end{pmatrix}^T;$$

while the matrix $\bar{A}$ is

$$\bar{A} = \begin{pmatrix} -A_{11} & -A_{12} & -A_{13} & 0 & 0 & 0 & A_{12} & A_{13} & 0 & 0 & 0 & 0 & A_{11} \\ 0 & -A_{21} & 0 & -A_{22} & -A_{23} & 0 & 0 & 0 & A_{21} & A_{23} & 0 & 0 & A_{22} \\ 0 & 0 & -A_{31} & 0 & -A_{32} & -A_{33} & 0 & 0 & 0 & 0 & A_{31} & A_{32} & A_{33} \end{pmatrix}.$$

How should we interpret these matrices? If we look at the first row of B, we have a row vector (2,0,0) (note that the notation above uses a transposition); and if we look at the first column of $\bar{A}$, we have a column vector $(-A_{11},0,0)^T$. This column indicates that the expression for the system contains a monomial of the form $z_1^2 z_2^0 z_3^0 = z_1^2$ and that it appears in the first equation with a coefficient of $-A_{11}$ and is absent from the second and third equations. Checking this notation for the whole matrices, we indeed have $\dot{z}_i = z_i(\sum_{j=1}^{13} \bar{A}_{ij} \prod_{k=1}^{3} z_k^{B_{jk}})$, with $i = 1,2,3$.

4. Transformation into a Lotka–Volterra system

The whole transformation relies now on a simple trick: we look at each product of powers of the variables that appear in the quasi-monomial form as a new variable of our dynamical system. With r the number of different monomials in the quasi-monomial form, we define the supplementary variables $z_{n+1}, \ldots, z_{n+r}$ as

$$z_{n+k} = \prod_{l=1}^{n} z_l^{B_{kl}}, \text{ with } k = 1, \ldots, r.$$

If we now look at the system of $n + r$ variables $\dot{z}_i$, taking into account these new definitions, we have for the first n variables

$$\dot{z}_i = z_i \sum_{j=1}^{r} \bar{A}_{ij} z_{n+j}, \text{ with } i = 1, \ldots, n.$$

Because of our special definition of the new variables, many simplifications occur in the expression of their time derivative, as seen in Eq. (9). We first use the formula for the derivative of a product (10). Then, we try to reconstruct the original variables since they are almost preserved by the derivation Eq. (11); the necessary correction leads to the appearance of a logarithmic derivative Eq. (12), which, in turn, transforms the exponent of the monomials into multiplicative coefficients Eq. (13). Lastly, the remaining correction produces the new expression of the vector fields of the first n variables, except for the disappearance of the multiplicative variable at the front (14). Writing these computations down, we have, for the r supplementary variables,

$$\dot{z}_{n+j} = (\prod_{k=1}^{n} z_k^{B_{jk}})' \text{ with } j = 1, \ldots, r \tag{9}$$

$$= \sum_{l=1}^{n} (\prod_{\substack{k=1 \\ k \neq l}}^{n} z_k^{B_{jk}})(z_l^{B_{jl}})' \tag{10}$$

$$= \sum_{l=1}^{n} z_{n+j} / z_l^{B_{jl}} (z_l^{B_{jl}})' \tag{11}$$

$$= z_{n+j} \sum_{l=1}^{n} (\ln(z_l^{B_{jl}}))' \tag{12}$$

$$= z_{n+j} \sum_{l=1}^{n} y B_{jl} \dot{z}_l / z_l \tag{13}$$

$$= z_{n+j} \sum_{l=1}^{n} B_{jl} \sum_{m=1}^{r} \bar{A}_{lm} z_{n+m} \tag{14}$$

$$= z_{n+j} \sum_{m=1}^{r} ((\sum_{l=1}^{n} B_{jl} \bar{A}_{lm}) z_{n+m}). \tag{15}$$

So, we can regroup all these terms by defining the $(n + r) \times (n + r)$ matrix M as

$$M = \begin{pmatrix} O^{n \times n} & \bar{A} \\ O^{r \times n} & B\bar{A} \end{pmatrix}$$

since $\sum_{l=1}^{n} B_{jl} \bar{A}_{lm} = (B\bar{A})_{jm}$; and the total system then becomes of the form

$$\dot{z}_i = z_i \sum_{j=1}^{n+r} M_{ij} z_j, \text{ with } i = 1, \ldots, n + r,$$

which is precisely an $n + r$-dimensional Lotka–Volterra system!

Supposing we were to have a solution $z(t) = (z_1(t), z_2(t), \ldots, z_{n+r}(t))^T$, we could then retrieve the behavior of the original system by transforming back the n first variables (if the matrix A is invertible):

$$x(t) = A^{-1}(\vec{\sigma}^{-1}((z_1(t), \ldots, z_n(t))^T) - a_0).$$

If the matrix is not invertible, we can apply the argument about the rank of the matrix A from Section 2 to retrieve the solution.

5. The dynamical perceptron with self-connections

One criticism that we can formulate about the dynamical perceptron is that its global behavior is often similar to the behavior around its fixed point. Although we can obtain complex dynamics by a careful choice of parameters, this is a delicate operation. The dynamics of such systems appear to be essentially limited. One way to obtain complex oscillatory behaviors more easily is to add a proportional self-connection to each variable:

$$\dot{x}_i = -\lambda_i x_i + \sigma(\sum_{j=1}^{n} A_{ij} x_j + a_{0i}). \tag{16}$$

The reason we can obtain oscillatory behaviors more easily is that we can choose A to make one of the fixed points unstable and then choose the λ_i large enough so that the behavior of the system always remain bounded. A further remark is that this kind of system can have multiple isolated equilibria, while the previous system had to have either a single equilibrium or a connected infinity of equilibria. The dynamics of the systems with inhibitory connections is thus richer. Also, the activation function for this system is no longer limited to the hyperbolic tangent; in Appendix C, we discuss other possible choices and conclude that we could use the logistic sigmoid as well.

We will embed this system into a Lotka–Volterra system. The first step is to define the following auxiliary variables:

$$w_i = \sigma(\sum_{j=1}^{n} A_{ij} x_j + a_{0i}).$$

Looking at the derivative of w, we then have the following dynamics for the system of x and w variables:

$$\dot{x}_i = w_i - \lambda_i x_i$$

$$\dot{w}_i = (1 - w_i^2) \sum_{j=1}^{n} A_{ij} (w_j - \lambda_j x_j).$$

This dynamics is, essentially, in quasi-monomial form and we can rewrite it as:

$$\dot{x}_i = x_i(-\lambda_i + x_i^{-1} w_i)$$

$$\dot{w}_i = w_i \sum_{j=1}^{n} (-\lambda_j A_{ij} x_j w_i^{-1} - \lambda_j A_{ij} x_j w_i + A_{ij} w_i^{-1} w_j + A_{ij} w_i w_j).$$

In Appendix B, we show how to write the matrices $\bar{A}$ and B

for the quasi-monomial form of this system. Again, we simply have to carefully collect all the exponents of the monomials in the matrix B and all their coefficients in the matrix $\bar{A}$. After this transformation into quasi-monomial form, the transformation of Section 4 directly applies in the same fashion except that the variable x is directly part of the Lotka–Volterra system and does not need to be retrieved by a change of coordinates (the cost of this advantage is a larger Lotka–Volterra system than for the dynamical perceptron).

6. Embedding a dynamical multi-layer perceptron into a Lotka–Volterra system

The transformations we have shown for the dynamical perceptron and the dynamical perceptron with self-connections apply also to the dynamical Multi-Layer Perceptron (MLP). To avoid an unnecessary clutter of notation, we will take the specific example of a two-hidden layer continuous-time recurrent MLP:

$$\dot{x} = \sigma(C\sigma(B\sigma(Ax + a_0) + b_0) + c_0)$$

where we have n states, p hidden units in the first layer, q hidden units in the second layer, and the activation function σ is the hyperbolic tangent. Thus, we find that $x(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{q\times p}$, $C \in \mathbb{R}^{n\times q}$, $a_0 \in \mathbb{R}^p$, $b_0 \in \mathbb{R}^q$, $c_0 \in \mathbb{R}^n$. The argument applies to any kind of MLP by doing the same type of computations again.

If we write this, coordinate by coordinate, we get

$$\dot{x}_i = \sigma(\sum_{l=1}^{q} C_{il}\sigma(\sum_{k=1}^{p} B_{lk}\sigma(\sum_{j=1}^{n} A_{kj}x_j + a_{0k}) + b_{0l}) + c_{0i}),$$

with $i = 1,\ldots,n$.

We define the following vector of p new variables: $y = \sigma(Ax + a_0)$. Again, we want to discuss the rank of the matrix A. If the matrix A is of rank n, then we can uniquely retrieve x from $y = \sigma(Ax + a_0)$. If the rank r of the matrix A is inferior to n, we have the same argument as in Section 2: two initial conditions $\bar{x}_0$ and $\tilde{x}_0$ such that $A\bar{x}_0 = A\tilde{x}_0$ will have the same dynamics in the new variables $\bar{y}(t) = \tilde{y}(t)$ for all times. Thus, $\bar{x}(t) = \sigma(C\sigma(B\bar{y}(t) + b_0) + c_0) = \tilde{x}(t)$ for all times. Therefore, the trajectories $\bar{x}(t)$ and $\tilde{x}(t)$ through $\bar{x}_0$ and $\tilde{x}_0$ always remain parallel. Since the kernel of A is of dimension $n - r$, we have that the dynamics is in fact r-dimensional. The dynamics of the variables $y_i = \sigma(\sum_{j=1}^{n} A_{ij}x_j + a_{0i})$ is thus

$$\dot{y}_i = \sigma'(\sum_{j=1}^{n} A_{ij}x_j + a_{0i})(\sum_{j=1}^{n} A_{ij}\dot{x}_j)$$

$$= (1 - \sigma^2(\sum_{j=1}^{n} A_{ij}x_j + a_{0i}))(\sum_{j=1}^{n} A_{ij}\dot{x}_j)$$

$$= (1 - y_i^2)(\sum_{j=1}^{n} A_{ij}\sigma(\sum_{l=1}^{q} C_{jl}\sigma(\sum_{k=1}^{p} B_{lk}y_k + b_{0l}) + c_{0j})).$$

We can remark that the form of the dynamics of y closely resembles that of the dynamics of x, but that the first hidden layer of the MLP has disappeared. This simplification encourages us to define, in the same fashion as before, a new vector of q variables: $z = \sigma(By + b_0)$. The dynamics of the variables $z_i = \sigma(\sum_{k=1}^{p} B_{ik}y_k + b_{0i})$ is thus

$$\dot{z}_i = \sigma'(\sum_{k=1}^{p} B_{ik}y_k + b_{0i})(\sum_{k=1}^{p} B_{ik}\dot{y}_k) = (1 - z_i^2)(\sum_{k=1}^{p} B_{ik}\dot{y}_k)$$

$$= (1 - z_i^2)(\sum_{k=1}^{p} B_{ik}(1 - y_k^2)(\sum_{j=1}^{n} A_{kj}\sigma(\sum_{l=1}^{q} C_{jl}z_l + c_{0j}))).$$

Enthusiastically, we pursue further by defining a new vector of n variables: $w = \sigma(Cz + c_0)$. The dynamics of the variables $w_i = \sigma(\sum_{l=1}^{q} C_{il}z_l + c_{0i})$ is thus

$$\dot{w}_i = \sigma'(\sum_{l=1}^{q} C_{il}z_l + c_{0i})(\sum_{l=1}^{q} C_{il}\dot{z}_l) = (1 - w_i^2)(\sum_{l=1}^{q} C_{il}\dot{z}_l)$$

$$= (1 - w_i^2)(\sum_{l=1}^{q} C_{il}(1 - z_l^2)(\sum_{k=1}^{p} B_{lk}(1 - y_k^2)(\sum_{j=1}^{n} A_{kj}w_j))).$$

Now, substituting the definitions for y,z,w in the previous equations and noting that, because of the same definitions, $\dot{x} = w$, we get the following system:

$$\dot{x}_i = w_i$$

$$\dot{y}_i = (1 - y_i^2)(\sum_{j=1}^{n} A_{ij}w_j)$$

$$\dot{z}_i = (1 - z_i^2)(\sum_{k=1}^{p} B_{ik}(1 - y_k^2)(\sum_{j=1}^{n} A_{kj}w_j))$$

$$\dot{w}_i = (1 - w_i^2)(\sum_{l=1}^{q} C_{il}(1 - z_l^2)(\sum_{k=1}^{p} B_{lk}(1 - y_k^2)(\sum_{j=1}^{n} A_{kj}w_j))).$$

The dynamics of the first n variables of this $(n + p + q + n)$-dimensional system (with appropriate initial conditions: $x_0 = x_0$, $y_0 = \sigma(Ax_0 + a_0)$, $z_0 = \sigma(By_0 + b_0)$, $w_0 = \sigma(Cz_0 + c_0)$!) will be identical to the dynamics of the original system. Further, we can write this system in quasi-monomial form:

$$\dot{x}_i = x_i(x_i^{-1}w_i)$$

$$\dot{y}_i = y_i(\sum_{j=1}^{n} A_{ij}(y_i^{-1}w_j - y_iw_j))$$

$$\dot{z}_i = z_i(\sum_{k=1}^{p}\sum_{j=1}^{n} B_{ik}A_{kj}(z_i^{-1}w_j - y_k^2z_i^{-1}w_j - z_iw_j + y_k^2z_iw_j))$$

$$\dot{w}_i = w_i(\sum_{l=1}^{q}\sum_{k=1}^{p}\sum_{j=1}^{n} C_{il}B_{lk}A_{kj}(1 - y_k^2)(1 - z_l^2)(w_i^{-1} - w_i)w_j).$$

We see that we have written the ordinary differential equation for each variable as a product of the variable itself by a linear

combination of products of the variables, which is precisely the definition of the quasi-monomial form.

A last remark is that the convergence of x implies the convergence of the enlarged system (x,y,z,w) since $\dot{x} = 0$ implies $\dot{y} = 0$, $\dot{z} = 0$, and $\dot{w} = 0$. The main difference is that, to check the convergence of the original system, we only need to check that all the trajectories of the enlarged system starting on the manifold of initial conditions $m(x_0) = (x_0,\sigma(Ax_0+a_0),\ \sigma(B\sigma(Ax_0+a_0) + b_0),\ \sigma(C\sigma(B\sigma(Ax_0 + a_0) + b_0) + c_0))$ converge.

The transformation into a Lotka–Volterra system proceeds exactly in the same way as in Appendices A and B. However, writing down the matrices $\bar{A}$ and B is a lengthy and tedious procedure and does not bring any further insight. We therefore do not perform it here. This computation will be helpful only when we use a computer algebra system to analyze the resulting Lotka–Volterra system.

7. Discussion and conclusion

An important corollary of this embedding of continuous-time recurrent multi-layer perceptrons into higher-dimensional Lotka–Volterra systems is that we can use Lotka–Volterra systems to approximate arbitrarily closely the dynamics of any finite-dimensional dynamical system for any finite time. The argument for this universal approximation property is exactly the same as that given by Funahashi and Nakamura (1993) which says that, for any given finite time, the trajectories of two sufficiently close dynamical systems remain close (this statement is true only for a finite time); this statement is based on the classical use of the Gronwall inequality for dynamical systems. It could therefore be interesting to use Lotka–Volterra systems as approximators for dynamical systems.

The restriction presented in Appendix C on the choice of the activation function (which is, in practice, limited to the hyperbolic tangent or the logistic sigmoid) is theoretically strong: there is no guarantee that systems with different choices of activation function have qualitatively the same behavior. Indeed, Sontag and Sussman (1997) have shown that controllability properties of systems with inputs could differ, even for different types of monotone increasing sigmoids. However, the limitation is fairly weak in the sense that the possible activation functions are the ones most commonly used and that we have universal approximation properties for dynamical systems.

We have presented techniques to embed different types of neural networks (dynamical perceptrons, dynamical perceptrons with self-connections, and multi-layer perceptrons) into predator–prey (Lotka–Volterra) systems. These techniques relied on looking at the dynamics of the output of every neuron in the network and using the special properties of the derivative of the hyperbolic tangent or logistic sigmoid activation function to write the system as what we called a quasi-monomial form—that is a dynamical system where the vector field is a linear combination of products of powers of the state variables. Once we had this quasi-monomial form, we looked at the dynamics of these products of powers of the state variables to obtain directly a Lotka–Volterra system. Our expectation is that these techniques will be useful to migrate techniques for the analysis of Lotka–Volterra systems into the field of dynamical neural networks—for example, techniques to study convergence or stability properties of dynamical systems, or discover first integrals of motion. This is our next objective.

Acknowledgements

The work of Yves Moreau and Joos Vandewalle has been supported by several institutions: the Flemish Government, GOA-MIPS (Model-based Information Processing Systems); the FWO Research Community, ICCoS (Identification and Control of Complex Systems); the Belgian State, DWTC—Interuniversity Poles of Attraction Programme (IUAP P4-02 (1997–2001): Modeling, Identification, Simulation and Control of Complex Systems. Léon Brenig and Stéphane Louies are grateful to the CATHODE ESPRIT Working Group (WG #24490) for scientific and financial support. They would also like to thank A. Figueiredo and T. M. da Rocha from the Physics Department of the University of Brasilia for fruitful scientific discussions.

Appendix A Transformation of the dynamical perceptron into a quasi-monomial form

We derive here the structure of the matrices $\bar{A}$ and B of the quasi-monomial form for the dynamical perceptron in a systematic fashion. Again, we write the equations as

$$\dot{z}_i = z_i[\sum_{j=1}^{n} A_{ij}(z_i^{-1}z_j - z_i z_j)]. \tag{17}$$

We see that the expression between brackets contains $2n$ monomials; for the n differential equations, we have a total of $2n^2$ terms. There are two types of monomials: $z_i z_j$ and $z_i^{-1}z_j$. The first type of monomial can take two different forms: z_i^2 (when $i = j$) or $z_i z_j$ with $j > i$ (when $i \neq j$). There are n monomials of the first form and $n(n-1)/2$ monomials of the second form. The second type of monomials can also take two different forms: 1 (when $i = j$) or $z_i^{-1}z_j$ (when $i \neq j$). There is one monomial of the first form and $n(n-1)$ monomials of the second form. Summing up all the contributions, we have a total of $(3n^2 - n + 2)/2$ different monomials to share among $2n^2$ terms. Carefully collecting all these terms into row vectors of exponents to form the matrix B and all their corresponding coefficients into column vectors to form the matrix $\bar{A}$, we can write these matrices as

$$B = [C^{(1)^T}C^{(2)^T}\ldots C^{(n)^T}D^{(1)^T}D^{(2)}\ldots D^{(n)^T}O^{1\times n^T}]^T, \tag{18}$$

$$A = [S^{(1)}S^{(2)}\ldots S^{(n)}R^{(1)}R^{(2)}\ldots R^{(n)}V], \tag{19}$$

where we define the sub-matrices $C^{(i)}$, $D^{(i)}$, $O^{1\times n}$, and $P^{(i)}$, $R^{(i)}$, V as follows. The matrix $O^{n\times m}$ is a matrix of zeros of dimension $n \times m$ and the matrix $I^{n\times m}$ is a diagonal matrix of ones of dimension $n \times m$. The matrix $C^{(i)}$ represents all the monomials of the form z_i^2 and z_iz_j (with $j > i$) and thus the dimension of the matrix depends on the index i with the matrix having $n - i + 1$ rows and n columns. We can write it as

$$C^{(i)} = \left(\begin{array}{c|c|c} \begin{matrix} 0 & \ldots & 0 \end{matrix} & 2 & \begin{matrix} 0 & \ldots & 0 \end{matrix} \\ \hline & 1 & \\ O^{(n-i)\times(i-1)} & 1 & I^{(n-i)\times(n-i)} \\ & \vdots & \\ & 1 & \end{array}\right).$$

Each of its rows is associated with the corresponding column of the matrix $P^{(i)}$, which collects all the coefficients of the monomials z_i^2 and z_iz_j (with $j > i$). Such a matrix again has variable dimension with n rows and $n - i + 1$ columns. We can write it as

$$P^{(i)} = \left(\begin{array}{c} O^{(i-1)\times(n-i+1)} \\ \hline \begin{matrix} -A_{ii} & -A_{i(i+1)} & -A_{i(i+2)} & \ldots & -A_{in} \\ 0 & -A_{(i+1)i} & 0 & \ldots & 0 \\ \vdots & 0 & -A_{(i+2)i} & & \vdots \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & 0 & \ldots & -A_{ni} \end{matrix} \end{array}\right).$$

Turning back to the matrix of exponents B, we collect all the exponents of the monomials of the form $z_i^{-1}z_j$ (with $j \neq i$) in the matrix $D^{(i)}$ which always has $n - 1$ rows and n columns:

$$D^{(i)} = \left(\begin{array}{c|c|c} I^{(i-1)\times(i-1)} & -1 & O^{(i-1)\times(n-i)} \\ \hline & \vdots & \\ O^{(n-i)\times(i-1)} & \vdots & I^{(n-i)\times(n-i)} \\ & -1 & \end{array}\right).$$

We collect the coefficients of these monomials in the corresponding columns of $R^{(i)}$, which also has $n - 1$ rows and n columns:

$$R^{(i)} = \left(\begin{array}{c} O^{(i-1)\times(n-1)} \\ \hline \begin{matrix} A_{i1} & \ldots & A_{i(i-1)} & A_{i(i+1)} & \ldots & A_{in} \end{matrix} \\ \hline O^{(n-i)\times(n-1)} \end{array}\right).$$

We are now only left with the constant monomial (which we represent by the row vector $O^{1\times n}$) and its coefficients, which we collect in the column vector

$$V = [A_{11},\ldots,A_{nn}]^T.$$

Taking all this notation into consideration, we can represent the matrices A and B according to Eqs. (18), (19).

Example. We now present the different matrices for the case of a four dimensional system. This clarifies how we have collected the different terms in the matrices $\bar{A}$ and B. The sub-matrices are $C^{(1)}$, $C^{(2)}$, $C^{(3)}$, $C^{(4)}$ and $D^{(1)}$, $D^{(2)}$, $D^{(3)}$, $D^{(4)}$ for B:

$$C^{(1)} = \left(\begin{array}{c|ccc} 2 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right), \quad C^{(2)} = \left(\begin{array}{c|ccc} 0 & 2 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right),$$

$$C^{(3)} = \left(\begin{array}{cc|cc} 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right), \quad C^{(4)} = \left(\begin{array}{ccc|c} 0 & 0 & 0 & 2 \end{array}\right);$$

$$D^{(1)} = \left(\begin{array}{c|ccc} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{array}\right), \quad D^{(2)} = \left(\begin{array}{c|c|cc} 1 & -1 & 0 & 0 \\ \hline 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{array}\right),$$

$$D^{(3)} = \left(\begin{array}{cc|c|c} 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ \hline 0 & 0 & -1 & 1 \end{array}\right), \quad D^{(4)} = \left(\begin{array}{ccc|c} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{array}\right).$$

We have decomposed the matrix A into the sub-matrices $P^{(1)}$, $P^{(2)}$, $P^{(3)}$, $P^{(4)}$, $R^{(1)}$, $R^{(2)}$, $R^{(3)}$, $R^{(4)}$, and V:

$$P^{(1)} = \left(\begin{array}{c|ccc} -A_{11} & -A_{12} & -A_{13} & -A_{14} \\ \hline 0 & -A_{21} & 0 & 0 \\ 0 & 0 & -A_{31} & 0 \\ 0 & 0 & 0 & -A_{41} \end{array}\right), \quad P^{(3)} = \left(\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ -A_{33} & -A_{34} \\ \hline 0 & -A_{43} \end{array}\right),$$

$$P^{(2)} = \left(\begin{array}{c|cc} 0 & 0 & 0 \\ \hline -A_{22} & -A_{23} & -A_{24} \\ 0 & -A_{32} & 0 \\ 0 & 0 & -A_{42} \end{array}\right), \qquad P^{(4)} = \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ -A_{44} \end{array}\right);$$

$$R^{(1)} = \left(\begin{array}{ccc} A_{12} & A_{13} & A_{14} \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right), \quad R^{(2)} = \left(\begin{array}{ccc} 0 & 0 & 0 \\ \hline A_{21} & A_{23} & A_{24} \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right),$$

$$R^{(3)} = \left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline A_{31} & A_{32} & A_{34} \\ \hline 0 & 0 & 0 \end{array}\right), \quad R^{(4)} = \left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline A_{41} & A_{42} & A_{43} \end{array}\right);$$

$$V = (A_{11}, A_{22}, A_{33}, A_{44})^T.$$

Appendix B Transformation of the dynamical perceptron with self-connections into a quasi-monomial form

We recall that for Eq. (16), the quasi-monomial form is

$$\dot{x}_i = x_i(-\lambda_i + x_i^{-1} w_i)$$

$$\dot{w}_i = w_i \sum_{j=1}^{n} (-\lambda_j A_{ij} x_j w_i^{-1} - \lambda_j A_{ij} x_j w_i + A_{ij} w_i^{-1} w_j + A_{ij} w_i w_j). \tag{20}$$

To use the quasi-monomial formalism in the same way as in the previous section, we need to unify our formalism to use a single set of variables. We therefore define the following variable $u = (x,w)^T$:

$$u_i = x_i, \text{ with } i = 1, \ldots, n$$

$$u_{n+i} = w_i, \text{ with } i = 1, \ldots, n.$$

If the matrices $\bar{A}^{(\lambda)}$ and $B^{(\lambda)}$ contain, respectively, all the coefficients and all the exponents of the different monomials, we can write the dynamics of u in quasi-monomial form:

$$\dot{u}_i = u_i \left(\sum_{j=1}^{m} \bar{A}_{ij}^{(\lambda)} \prod_{k=1}^{2n} u_k^{B_{jk}^{(\lambda)}} \right).$$

To build the matrices $\bar{A}^{(\lambda)}$ and $B^{(\lambda)}$, we have to look at all the monomials in Eq. (20). If we first focus on the terms $w_i^{-1} w_j$ and $w_i w_j$, which already appeared in Section 3, we see that we have exactly the same contribution in the matrix $B^{(\lambda)}$ as in the case of the dynamical perceptron. We have the same cancelation of the factors of the terms in $w_i^{-1} w_j$ or $w_i w_j$ as in Section 3. They will contribute to the matrix $B^{(\lambda)}$ with a first sub-matrix equal to B. Notice that the left part of the matrix $B^{(\lambda)}$ is for powers of x while the right part is for powers of w. The next blocks take into account, respectively, the monomials of the form $x_i^{-1} w_i$, $x_i w_j^{-1}$, and $x_i w_j$. None of these terms ever add up or cancel out, which explains the plain diagonal and column structure of these sub-matrices. We have

$$B^{(\lambda)} = \left(\begin{array}{c|c} O^{r\times n} & B \\ \hline -I^{n\times n} & I^{n\times n} \\ \hline I^{n\times n} & -K^{(1)} \\ \vdots & \vdots \\ I^{n\times n} & -K^{(n)} \\ \hline I^{n\times n} & K^{(1)} \\ \vdots & \vdots \\ I^{n\times n} & K^{(n)} \end{array}\right),$$

where we define $K^{(i)}$ as

$$K^{(i)} = \left(\begin{array}{c|c|c} O^{n\times(i-1)} & \begin{array}{c} 1 \\ \vdots \\ 1 \end{array} & O^{n\times(n-i)} \end{array}\right).$$

Similarly, we collect all the coefficients of the monomials in $\bar{A}^{(\lambda)}$. We place the contributions relating to the equations for the x variables in the top part of the matrix while we place the contributions for w in the bottom part of the matrix. The contributions for the monomials $w_i^{-1} w_j$ and $w_i w_j$ are exactly the same as before for the variables w; the first bottom sub-matrix is therefore equal to A. These monomials have no influence on the variables x and the first top sub-matrix is therefore equal to zero; except in the case where $w_i^{-1} w_j = 1$ corresponding to the $-\lambda_i$ terms in the equations for x, which generate a column of $-\lambda_i$ in the otherwise empty matrix Q. Correspondingly, the other sub-matrices take into account the coefficients of the monomials of the form $x_i^{-1} w_i$, $x_i w_j^{-1}$, and $x_i w_j$. We thus have

$$\bar{A}^{(\lambda)} = \left(\begin{array}{c|c|ccc|ccc} Q^{n\times r} & I^{n\times n} & O^{n\times n} & \ldots & O^{n\times n} & O^{n\times n} & \ldots & O^{n\times n} \\ \hline A & O^{n\times n} & -L^{(1)} & \ldots & -L^{(n)} & L^{(1)} & \ldots & L^{(n)} \end{array}\right),$$

where we define Q and $L^{(i)}$ as

$$Q = \left(\begin{array}{c|c} O^{n\times(r-1)} & \begin{array}{c} -\lambda_1 \\ \vdots \\ -\lambda_n \end{array} \end{array}\right), \qquad L^{(i)} = \left(\begin{array}{c} O^{(i-1)\times(n-1)} \\ \hline A_{i1}\lambda_1 \quad \ldots \quad A_{in}\lambda_n \\ \hline O^{(n-i)\times(n-1)} \end{array}\right).$$

Appendix C Choice of the activation function

We chose the activation function σ as $\sigma(x) = \tanh(x)$ for simplicity. But we are not directly limited to this activation function. The essential property that we used was the fact that $\sigma'(x) = 1 - \sigma^2(x)$. For the logistic sigmoid $\sigma(x) = 1/(1 + e^{-x})$, we have $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ and we can do the calculations again without difficulty. The condition that the activation function has to satisfy is that its derivative be a sum of quasi-monomials of itself. If we do not require saturation at plus and minus infinity, we can use $\sigma(x) = e^x$, but the system is not globally Lipschitz and the solution might blow-up in finite time, which is an annoyance. A square function $\sigma(x) = x^2$ suffers from the same problem. The hyperbola $\sigma(x) = 1/x$ gives $\sigma'(x) = -1/\sigma^2(x)$, but then the vector field is not even bounded, which is even worse. In general, we could consider the condition $\sigma'(x) = L(\sigma(x))$, where $L(.)$ is a finite Laurent series $L(y) = (1/y^m)\sum_{k=0}^{n} a_k y^k$, or even more generally, a Puiseux series. For the Laurent series, the solution of the differential equation is given by separation of variables $\frac{\sigma^m(x)}{L(\sigma(x))} d\sigma(x) = dx$. We can then expand the term $\sigma^m(x)/L(\sigma(x))$ as a polynomial plus a sum of simple fractions, and integrate them. This integration produces a rational function plus a sum of logarithmic terms and arctangent terms and $\sigma(x)$ is found by inverting this relation. This inversion will not lead to an explicit solution in general. If we further require that the vector field be globally Lipschitz to guarantee existence of the trajectories for all times, the possible solutions become scarce. From the most frequently used activation functions, we have been able to identify only the hyperbolic tangent and the logistic sigmoid as satisfying these conditions.

References

Albertini, F., & Sontag, E. D. (1993). For neural networks function determines form. *Neural Networks, 6*, 975–990.

Brenig, L. (1988). Complete factorization and analytic solutions of generalized Lotka–Volterra equations. *Physics Letters A, 133* (7-8), 378–382.

Cohen, M. A., & Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13* (5), 815–826.

Funahashi, K. I., & Nakamura, Y. (1993). Approximation of dynamical systems by continuous-time recurrent neural networks. *Neural Networks, 6*, 801–806.

Gouzé, J.-L. (1990). Transformation of polynomial differential systems in the positive orthant. (Tech. Rep. INRIA-1308). Institut National de Recherche en Informatique et en Automatique: Unité de Recherche INRIA-Sophia Antipolis, France.

Grossberg, S. (1978). Decisions, patterns, and oscillations in nonlinear competitive systems with applications to Volterra–Lotka systems. *Journal of Theoretical Biology, 73*, 101–130.

Grossberg, S. (1988). Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Networks, 1* (1), 17–61.

Hirsch, M. W. (1989). Convergent activation dynamics in continuous-time networks. *Neural Networks, 2*, 331–349.

MacArthur, R. H. (1969). Species packing, or what competition minimizes. *Proceedings of the National Academy of Sciences USA, 54*, 1369–1375.

May, R. M., & Leonard, W. J. (1975). Nonlinear aspects of competition between three species. *SIAM Journal of Applied Mathematics, 29*, 243–253.

Moreau, Y., & Vandewalle, J. (1997). When do dynamical neural networks with and without hidden layer have identical behavior? (Tech. Rep. ESAT-SISTA TR97-51). Katholieke Universiteit Leuven: Department of Electrical Engineering, Belgium.

Redheffer, R., & Walter, W. (1984). Solution of the stability problem for a class of generalized Volterra prey–predator systems. *Journal of Differential Equations, 52*, 245–263.

Sontag, E. D. (1992). Neural nets as system models and controllers. In *Proceedings Seventh Yale Workshop on Adaptive and Learning Systems*, pp. 73–79.

Sontag, E. D., & Sussman, H. J. (1997). Complete controllability of continuous-time recurrent neural networks. *Systems and Control Letters, 30*, 177–183.

Yves Moreau is a Research Assistant with the F.W.O.-Vlaanderen

Joos Vandewalle is Full Professor at the K.U. Leuven.