Penalty function approach to recurrent neural network dynamics

Edoardo Milotti*
Dipartimento di Fisica dell'Università di Trieste and Istituto Nazionale di Fisica Nucleare, Sezione di Trieste, Via Valerio 2, I-34127 Trieste, Italy
(Received 3 January 1997)

The Hopfield dynamics for recurrent neural networks minimizes a certain quadratic form on the unit hypercube. I show here how the dynamical system can be derived from a standard method of optimization theory. I use the method to give a precise meaning to nonsymmetric interactions, and I discuss the possibility of introducing other types of dynamics. [S1063-651X(97)11407-6]

PACS number(s): 87.10.+e, 02.60.Pn

I. INTRODUCTION

There are many connections between statistical mechanics and optimization theory, and new ones are discovered all the time. For instance, Barahona [1] recently showed how Ising problems can be solved numerically with a variant of linear programming. Neural networks belong to both disciplines and so it should not be surprising to find even deeper connections here. This paper deals with the dynamics of recurrent neural networks and its purpose is to show that the Hopfield dynamics [2,3] can be derived from a standard method of optimization theory. I use this method to understand both the origin of the algorithmic stability of different kinds of network dynamics and the dynamics associated with asymmetric interactions. I also discuss the possibility of introducing other types of dynamics.

The Hopfield dynamics minimizes a certain quadratic form on the $\{-1,1\}$ N-dimensional hypercube; i.e., it corresponds to a problem of constrained optimization.

Now consider the general problem of minimizing a continuous function $f_0(x_1, \ldots, x_N)$ subject to a set of constraints of the form $\{g_k(x_1, \ldots, x_N) \leqslant 0\}_{k=1,N}$: While there are several good numerical algorithms to minimize an unconstrained function, after the introduction of the constraints these methods cannot be used as they stand and the problem usually becomes much harder. There are two straightforward ways to convert the problem back to unconstrained [4]: In the barrier function method the function $f_0(x_1, \ldots, x_N)$ is replaced by the auxiliary function

$$f_0(x_1, \ldots, x_N) + \mu \sum_{k=1}^{k=N} \phi(g_k(x_1, \ldots, x_N)), \qquad (1)$$

where $\phi(y)$ is a function of one variable that is both continuous and non-negative over $\{y : y < 0\}$ and such that $\lim_{y \to 0^-} \phi(y) = +\infty$, and μ is a non-negative parameter.

In the penalty function method the function $f_0(x_1, \ldots, x_N)$ is replaced instead by the auxiliary function

$$f_0(x_1, \ldots, x_N) + \mu \sum_{k=1}^{k=N} \max[0, g_k(x_1, \ldots, x_N)]. \qquad (2)$$

In both methods a numerical solution is found for the unconstrained minimum of the auxiliary function for a given μ, which is then progressively decreased. The barrier function method has the advantage of having an auxiliary function with continuous derivatives, but it needs a starting point inside the feasible region. The penalty function method has no such requirement, but the auxiliary function has a discontinuous derivative on the boundary of the feasible region.

Here I use a variation of both methods: Let $\mathcal{F}_\beta(x)$ be a non-negative continuous function defined for all real x, so that it is a monotonically increasing function of β for all x, $\mathcal{F}_\beta(0) = 1$ for all β, and such that, when $\beta \to +\infty$, $\mathcal{F}_\beta(x) \to 0$ if $x < 0$ and $\mathcal{F}_\beta(x) \to +\infty$ if $x > 0$. One such function is $\mathcal{F}_\beta(x) = e^{\beta x}$, where β is a positive real number. Then replace $f_0(x_1, \ldots, x_N)$ with the auxiliary function

$$f_\beta(x_1, \ldots, x_N) = f_0(x_1, \ldots, x_N) + \sum_{k=1}^{k=N} \mathcal{F}_\beta(g_k(x_1, \ldots, x_N)). \qquad (3)$$

This function gives a small penalty inside the feasible region and a large penalty outside, it is everywhere continuous, and it does not require a starting point inside the feasible region. As β is made larger and larger the penalty inside the feasible region becomes vanishingly small, while outside it grows faster and faster. This modified penalty function method will be used now to rederive the Hopfield dynamics.

II. DERIVATION OF THE DYNAMICS FOR $\{-1,1\}$ INTEGER PROGRAMMING PROBLEMS

In general $\{-1,1\}$ integer programming problems one tries to find the vertex of the $\{-1,1\}$ N-dimensional hypercube that minimizes a certain function $f_0(x_1, \ldots, x_N)$. If we relax the problem and search instead for solutions that satisfy the constraints $-1 \leqslant x_k \leqslant 1$, we can take the penalty function

$$\sum_{k=1}^{k=N} G_k(x_k; \beta), \qquad (4)$$

where $G_k(x;\beta)$ are even, concave-up functions, with continuous derivatives, and such that $G_k(\pm 1;\beta) = 1$, $G_k(x;\beta) \to 0$ for $\beta \to +\infty$ if $|x| < 1$, and $G_k(x;\beta) \to +\infty$ for $\beta \to +\infty$ if $|x| > 1$ (see Fig. 1). If $F_k(x;\beta) =$

*Electronic address: milotti@trieste.infn.it

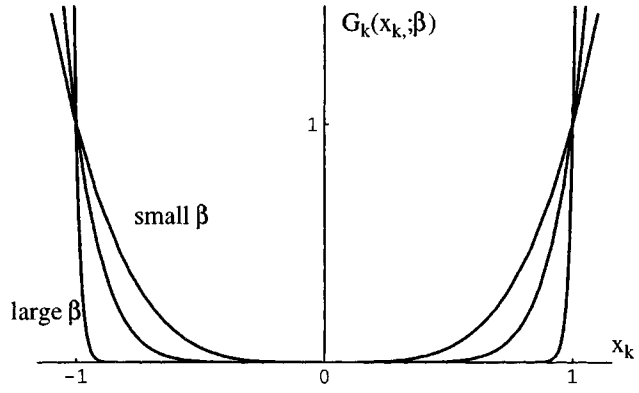

FIG. 1. Graphical representation of the functions $G_k(x_k,\beta)$: As β grows the G_k's look more and more like square wells.

$(d/dx)G_k(x;\beta)$, then $F_k^{-1}(x;\beta)$ is an increasing sigmoid function (see Fig. 2). The auxiliary function for the minimization problem is

$$f_\beta(x_1,\ldots,x_N)=f_0(x_1,\ldots,x_N)+\sum_{k=1}^{k=N} G_k(x_k;\beta), \quad (5)$$

so that one must solve the system of equations

$$\frac{\partial}{\partial x_k} f_\beta(x_1,\ldots,x_N)=\frac{\partial}{\partial x_k} f_0(x_1,\ldots,x_N)+F_k(x_k;\beta)=0, \quad (6)$$

which can also be rearranged in the form

$$x_k=F_k^{-1}\left(-\frac{\partial}{\partial x_k} f_0(x_1,\ldots,x_N);\beta\right). \quad (7)$$

So, for instance, if we take the linear function

$$f_0(x_1,\ldots,x_N)=\sum_{k=1}^{k=N} (a_k x_k+b_k), \quad (8)$$

then the *unique* solution is guaranteed to lie on a vertex [5], and

$$x_k=-F_k^{-1}(a_k;\beta)\underset{\beta\to+\infty}{\longrightarrow}-\mathrm{sgn}(a_k). \quad (9)$$

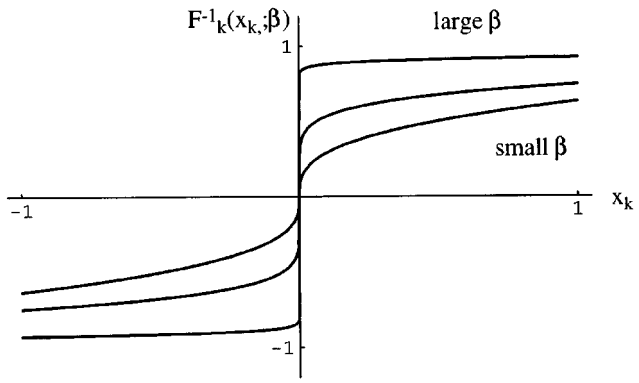

FIG. 2. Graphical representation of the functions $F_k^{-1}(x_k,\beta)$: As β grows the F_k^{-1}'s approach a step function.

In general formula (7) is not closed, but rather can be used as a recursive formula for x_k:

$$x_k(m+1)=F_k^{-1}\left\{-\frac{\partial}{\partial x_k} f_0(x_1(m),\ldots,x_N(m));\beta\right\}. \quad (10)$$

If we let $F_k(x;\beta)=(1/\beta)\mathrm{arctanh}x$, then

$$G_k(x;\beta)=\frac{1}{\beta}\left[x\,\mathrm{arctanh}x+\frac{1}{2}\ln(1-x^2)\right] \quad (11)$$

(which satisfies the conditions given above in a rather more restrictive way, but is still a useful penalty function), and we obtain the conventional form

$$x_k(m+1)=\tanh\left\{-\beta\frac{\partial}{\partial x_k} f_0(x_1(m),\ldots,x_N(m))\right\}. \quad (12)$$

Now if we take

$$f_0(x_1,\ldots,x_N)=\frac{1}{2}\sum_{i,j=1}^{N} T_{ij}x_i x_j+\sum_{j=1}^{N} I_j x_j, \quad (13)$$

with $T_{ij}=T_{ji}$ and $T_{ii}=0$ and where I_j is an external input, we recover the discrete Hopfield dynamics [3]:

$$x_k(m+1)=F_k^{-1}\left(-\sum_{j=1}^{j=N} T_{kj}x_j(m)+I_k;\beta\right). \quad (14)$$

Other combinatorial problems can be treated with the formalism introduced here. For instance, the maximum clique problem (see [6] for a review) can be shown to be equivalent to the problem of minimizing the quadratic form x^TAx so that $x\in\{0,1\}^N$ and $A=A_G-I$, where A_G is the adjacency matrix of a graph G. A simple auxiliary function for this minimization problem is

$$f_n(x_1,\ldots,x_N)=\sum_{j,k=1}^{N} a_{jk}x_j x_k+\sum_{k=1}^{N} G_k(x_k;\beta); \quad (15)$$

then, proceeding as before, one easily obtains the limiting form of the discrete dynamics

$$x_k(m+1)=\frac{1}{2}\left\{1-\mathrm{sgn}\left[\sum_{j=1}^{N} (a_{jk}+a_{kj})x_j(m)\right]\right\}, \quad (16)$$

and the solution of the maximum clique problem corresponds to one of the stable states of Eq. (16).

III. ALGORITHMIC STABILITY

Now let us see how all this can shed some light on two old problems in Hopfield theory, namely, algorithmic stability of the iterative solutions and the meaning of nonsymmetric T's, and let us consider the algorithmic stability problem first.

The update formula (14) can be implemented both in a synchronous and in an asynchronous way: It turns out that the synchronous updates have the tendency to produce oscillations more frequently than the asynchronous updates ([7]

and references therein). Moreover, it is quite clear from the discussion above that the usual recursive formula (14) has no special feature apart from being a very simple and obvious choice: We could as well take Eq. (6) and write

$$F_k(x_k;\beta)=-\frac{\partial}{\partial x_k}f_0(x_1,\ldots,x_N)=-\left(\sum_{j=1}^{j=N}T_{kj}x_j+I_k\right), \tag{17}$$

and then, assuming that each x_k may change by small amounts (so that it is not forced to lie on or near a hypercube vertex) and looking again for a recursive solution, we obtain

$$F_k(x_k(t+\Delta t);\beta)\approx F_k(x_k(t);\beta)+\frac{dF_k(x_k(t);\beta)}{dt}\Delta t =-\left(\sum_{j=1}^{j=N}T_{kj}x_j(t)+I_k\right) \tag{18}$$

or, equivalently,

$$-\frac{dF_k(x_k(t);\beta)}{dt}\Delta t=F_k(x_k(t);\beta)+\sum_{j=1}^{j=N}T_{kj}x_j(t)+I_k, \tag{19}$$

which is just the continuous dynamics defined in [3].

From the discussion above we see that the different ruggedness of the three types of dynamics is probably due to the different step sizes, which are such that

stepsize(continuous update)

$<$stepsize(async. discrete update)

$<$stepsize(sync. discrete update),

since in the continuous dynamics the variables may change by small amounts (just like in the ''interior point methods'' of optimization theory [8]), while in both kinds of discrete dynamics they must jump nearly from vertex to vertex of the hypercube. However, the step size is usually smaller for the asynchronous update, since in this case the jump is always to one adjacent vertex, while in the case of a synchronous (or parallel) update, the jump may bring the system to a faraway nonadjacent vertex.

IV. NONSYMMETRIC DYNAMICS

Now let us turn to matrices T which are not symmetric: Then, using Eq. (13) once again, instead of Eq. (14), we obtain

$$x_k(m+1)=F_k^{-1}\left(-\frac{1}{2}\sum_{j=1}^{j=N}(T_{kj}+T_{jk})x_j(m)+I_k;\beta\right), \tag{20}$$

so that the resulting dynamics is still symmetric; however, if we take

$$f_0(x_1,\ldots,x_N;y_1,\ldots,y_N)=\frac{1}{2}\sum_{i,j=1}^{N}T_{ij}x_iy_j+\sum_{j=1}^{N}(I_jx_j+J_jy_j), \tag{21}$$

where I_j and J_j are permanent external inputs, then the auxiliary function is

$$f_\beta(x_1,\ldots,x_N;y_1,\ldots,y_N)=f_0(x_1,\ldots,x_N;y_1,\ldots,y_N)+\sum_{k=1}^{k=N}G_k(x_k;\beta), \tag{22}$$

and the set $\{x_1,\ldots,x_N;y_1,\ldots,y_N\}$ that minimizes Eq. (22) must solve the system of equations

$$\left(\frac{1}{2}\sum_{j=1}^{N}T_{jk}y_j+I_k\right)+F_k(x_k;\beta)=0, \tag{23}$$

$$\left(\frac{1}{2}\sum_{j=1}^{N}T_{kj}x_j+J_k\right)+F_k(y_k;\beta)=0. \tag{24}$$

Therefore, we obtain the recursive formulas

$$x_k(m+1)=F_k^{-1}\left(-\frac{1}{2}\sum_{j=1}^{N}T_{jk}y_j(m)+I_k;\beta\right), \tag{25}$$

$$y_k(m+1)=F_k^{-1}\left(-\frac{1}{2}\sum_{j=1}^{N}T_{kj}x_j(m)+J_k;\beta\right). \tag{26}$$

A nonsymmetric dynamics which minimizes a Liapunov function is thus possible for two sets of paired variables: These variables may represent the state of a system at ''odd'' and ''even'' times, or they may actually be related to a split system.

V. OTHER KINDS OF DYNAMICS

In the previous sections I have often assumed that the extrema of the function f_0 lie on the vertices of the hypercube. Obviously this is not always the case, and with the assumed relaxation of the constraints the method may lead to extrema that are located *inside* the hypercube. However, it is possible to modify the method so that only the surface or the vertices of the hypercube are selected by the penalty function; for instance, the penalty function

$$\sum_{k=1}^{k=N}\cosh\left[n\left(\sum_{k=1}^{k=N}x_k^{2n}-1\right)\right] \tag{27}$$

selects the surface of the hypercube, while

$$\sum_{k=1}^{k=N}[(x_k^2-1)^2+1]^n \tag{28}$$

selects the vertices of the hypercube (the penalty function parameter n is in both cases a positive integer).

Penalty functions such as these do not lead to recursive formulas like Eq. (14), and do not have a limiting dynamics: To see the difficulties involved consider the penalty function (28), so that the condition for the extrema becomes

$$\frac{\partial}{\partial x_k} f_0(x_1, \ldots, x_N) = -4nx_k(x_k^2-1)[(x_k^2-1)+1]^{n-1}. \tag{29}$$

Now a limiting dynamics can no longer be defined, since the product $(x_k^2-1)[(x_k^2-1)+1]^{n-1}$ approaches the form $0\times\infty$ as $n\to\infty$. Geometrically the reason is that now, as n grows, the allowed regions (the neighborhoods of the vertices of the hypercube) become disconnected.

ACKNOWLEDGMENT

I wish to thank Dr. Marco Budinich for useful discussions.

[1] F. Barahona, Phys. Rev. B **49**, 12 864 (1994).
[2] J. J. Hopfield, Proc. Natl. Acad. Sci. USA **79**, 2554 (1982).
[3] J. J. Hopfield, Proc. Natl. Acad. Sci. USA **81**, 3088 (1984).
[4] M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming, Theory and Applications* (Wiley, New York, 1979), Chap. 9.
[5] V. Chvatal, *Linear Programming* (Freeman, New York, 1983).
[6] P. M. Pardalos and J. Xue, J. Glo. Opt. **4**, 301 (1994).
[7] C. M. Marcus and R. M. Westervelt, Phys. Rev. A **40**, 501 (1989).
[8] B. Jansen, C. Roos, and T. Terlaki, Delft University of Technology Report No. 95-45 (unpublished).