Revista Mexicana de Física **42**, *No. 5 (1996) 810–825*

Stable criticality in a feedforward neural network*

A. CECCATTO, H. NAVONE

Instituto de Física Rosario, Universidad Nacional de Rosario
Bv. 27 de febrero 210 bis, 2000 Rosario
Provincia de Santa Fe, República Argentina

AND

HENRI WAELBROECK

Instituto de Ciencias Nucleares, UNAM
Apartado postal 70-543, 04510 México, D.F., México
e-mail: hwael@roxanne.nuclecu.unam.mx

Recibido el 6 de diciembre de 1995; aceptado el 17 de abril de 1996

ABSTRACT. How do learning processes escape from local optima? Doing so requires an exploration of the landscape at a range of the order of the landscape correlation length —a "long jump" in synapsis space. This brings up a dilemma: because of the high dimensionality of this space, the probability that a *random* long jump lead to a better optimum is nearly zero. We conjecture that "intelligent" coarse-grained learning operators emerge as a consequence of a self-organization process in neural systems, as follows. The presentation of a single new data vector stimulates the recall of other vectors, each of which generates a small displacement in synapsis space. The sum of these displacements constitutes a coarse-grained learning event. Although long jumps are occasionally needed to escape from local optima, they should be the exception rather than the rule. This leads us to propose a neural network model where the recall process self-organizes to a critical state and one has a power-law distribution in the number of data vectors recalled.

RESUMEN. ¿Cómo pueden los procesos de aprendizaje "escapar" de mínimos locales? Para ello se requiere de una exploración de la topografía de error a una escala del orden de la distancia de correlación —un "brinco grande" en el espacio de sinapsis. Esto nos lleva a un dilema: por la alta dimensionalidad de este espacio, la probabilidad de que un brinco grande *aleatorio* conduzca a un mejor óptimo es prácticamente cero. Proponemos una conjetura: que operadores de aprendizaje "inteligentes" emergen en sistemas neuronales como consecuencia de un proceso de auto-organización, como sigue. La presentación de un vector de datos estimula el recuerdo de otros vectores, cada uno de los cuales genera un pequeño desplazamiento en el espacio de sinapsis. La suma de tales desplazamientos constituye un operador de aprendizaje macroscópico. Aunque se requiera de brincos grandes ocasionalmente, para escapar de mínimos locales, el aprendizaje debe estar dominado por pequeños pasos destinados a encontrar los mínimos locales. Estas consideraciones sugieren un modelo de red neuronal donde el proceso de recuerdo de memorias se auto-organiza en un estado crítico, de tal manera que la distribución en el número de vectores recordados (tamaño de avalancha) sigue una ley de potencias.

PACS: 87.10.+e; 89.70.+c; 87.22.As

* This work is supported in part by CONACyT grant 400349-5-1714E and by the Association Générale pour la Coopération et le Développement (Belgium).

1. INTRODUCTION

The process of "learning" can be viewed as the evolution of an algorithm to perform certain tasks. The performance can be measured by a "fitness function", which evaluates how well the tasks are accomplished. When the algorithm is executed by a complex system, such as a neural network or gene expression mechanisms, this fitness function is a rugged landscape [1] on the space of algorithms.

One distinguishes two types of learning operators: small-stepping operators generate displacements that are small in comparison with the landscape correlation length; their role in learning is to generate flows to a local optimum. Long-jump learning operators, in contrast, generate large displacements which may reach to other valleys in this landscape, with the purpose to seek out better optima.

It is not presently known whether biological learning systems make use of long-jump learning operators, although there is some evidence which points in this direction. The evolution from reptile to bird, for example, seems to imply escape from a local optimum. In the context of neural networks, the development of human motricity offers several examples: from lying to crawling, from crawling to walking and from there to riding a bicycle, all require several coordinated changes in motor control which are of no use if carried out independently, rather the contrary. We should stress that even though these examples appear to imply that long-jump learning operators are required to clear past a "barrier", not enough is known about neural and genetic systems to assert that there is no way to go around such barriers in small steps that are beneficial or at least close to neutral. As Bak has shown, one can model punctuated equilibria in learning processes without tunnelling through barriers [4]. In such models one has long jumps in algorithm space but these are not motivated by the purpose to escape from local optima.

This article is based on the premise that long-jump operators indeed play an important role in learning, whether or not they play a role in clearing fitness barriers. We will further speculate that these operators occur as a manifestation of self-organized criticality (SOC), as "avalanches" of small displacements. With this conjecture, there is no need to postulate the existence of a separate class of learning operators —rather, the long-jump operators would emerge as coarse-grained operators in an effective theory describing the learning process at a longer time scale.

Why self-organization and criticality? First of all, since a long jump by definition exceeds the landscape correlation length, learning by random long jumps would be no better than a random search in the space of algorithms. One cannot presently rule out that this is the way long-jump learning proceeds in biological systems —for example, it is difficult to estimate the number of failed mutations for every successful one since there is no fossil record of the failures— yet it is difficult to explain the observed efficiency of both genetic and neural learning without considering that the search in algorithm space is organized in some way. Since there are no external factors to carry out such an organizational task, one can only assume that a process of self-organization takes place. Self-organization of neural systems has been considered elsewhere, *e.g.*, in Kohonen's networks [2], and more recently by Stassinopoulos and Bak in a simple brain model [3]. The argument for criticality relies on the observation that stable systems are slow to change, while unstable ones tend to lose previously acquired beneficial traits, so learning would proceed most efficiently at the

boundary between these situations, *i.e.*, at the critical point. In terms of the distribution of sizes of learning events, the claim is that learning proceeds most efficiently when a majority of coarse-grained events are small but events of all sizes occur, including long jumps. Discrete dynamical systems which tend to a steady-state regime characterized by long-range correlations and a high algorithmic complexity are typically self-organized critical systems. In the case of genetic learning, there is evidence in the form of power-law distributions of various empirical correlation functions [4].

Clearly, the evidence for long-jump operators and for SOC is weak because the biological learning processes are complex and poorly understood. One of our aims in this work will be to lend partial support to these conjectures through numerical simulations with a model based on the feedforward neural network [5].

The proposal that long-jumps can be non-random, or "organized", is highly non-trivial. Since a long jump exceeds the landscape correlation length, it can lead to a better-than average fitness value only if the mechanism which generates the jump "knows" the structure of the landscape beyond its immediate vicinity. This knowledge can only arise from previous experiments, which lends further weight to our claim that long-jump learning operators *must be effective operators in a time coarse-grained theory.* In other words, the "intelligence" of the long-jump operators is an emerging property —from there to the conjecture that they are "avalanches" of smaller events is a relatively small step.

The claim that the long-jump learning operators can target regions of higher fitness indicates that there is an emergent topology in the coarse-grained theory that is distinct from the original one: the targeted regions are "nearby" in the sense that they can be reached in a single step. According to this point of view, it is not so much that long-jump operators must organize to be effective in spite of reaching beyond the landscape correlation length, but that these jumps appear to be "long" because they are being measured with the wrong metric.

Avalanches propagate through "near-neighbours", so to claim that a system enjoys the property of SOC one must be able to specify what is meant by "near neighbour"; this implies that there is a distance function in the space of training vectors. This in turn induces the mentioned "emergent topology" in algorithm space, as follows. According to our discussion above, a coarse-grained learning operator consists of an ordered sequence of presentations of training vectors. One can define the "norm" of such an operator as the sum of the distances between the successive training vectors. An avalanche involves near-neighbours, so the corresponding norm is small compared to a coarse-grained operator involving the same number of randomly-chosen training vectors.

The "emergent intelligence" must manifest itself in the choice of which training vectors are recalled in the avalanche process; *i.e., it is encoded in the distance function* on the space of training vectors. So this distance function must be allowed to self-organize via some *reinforcement* mechanism. Following Stephanopoulos and Bak's proposal that self-organization in the brain should follow from a "democratic reinforcement" mechanism, one would assume that the result of each coarse-grained learning step is evaluated and the distance between vectors that participated in the avalanche is reduced (increased) if the result was positive (negative). In this way, when a new data vector is presented the previously-viewed vectors that are recalled will be precisely those which can cooperate to realize a fruitful long-jump in algorithm space.

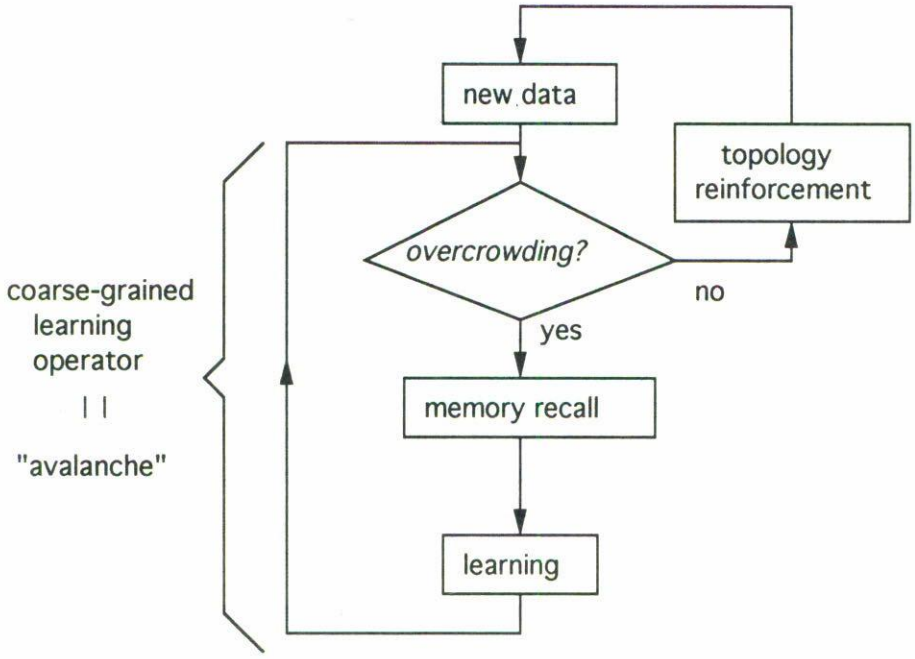

FIGURE 1. A model of learning is proposed, which allows for the emergence of intelligence through topological reinforcement. A learning avalanche propagates through near-neighbours in the space of data vectors; the organization of learning processes is then naturally related to the definition of what one means by "near-neighbour". Applying the principle of democratic reinforcement, points which cooperate in successful learning avalanches should be drawn closer together, and vice-versa, points which cooperate in failed learning attempts repel from each other. In this way, one expects the distance function to organize so that the presentation of a new training vector provokes the recall of previously-viewed vectors which are likely to help generate an intelligent long jump in algorithm space.

The proposal for neural learning which we have outlined in the previous paragraphs is summarized in the organigram of Fig. 1. When a new data vector is presented, one determines whether this forces a shift in the pile —a test referred to as "overcrowding", in analogy to sandpiles. As long as this occurs, the system proceeds to recall "near-neighbours" and test for continued overcrowding; every recall is accompanied by the corresponding learning iteration. When there is no further shifting, the overall result of the learning iterations is evaluated and a democratic reinforcement signal triggers an update of the distance function which defines the notion of "near-neighbour".

How does this proposal differ from that of Stephanopoulos and Bak? Rather than applying the principle of democratic reinforcement at the level of individual learning operators, our proposal is that it should be the guiding principle which allows for the emergence of "intelligence" in coarse-grained learning operators. We do not have any specific proposal on the nature of the small-step learning operators in biological neural networks, as very little is known about the biochemical processes involved. We stress that the mechanisms which we are proposing here are in principle applicable to any "microscopic" learning scheme —in the extreme case, the coarse-grained operators may end up being similar for

different schemes and the choice of fine-grained operators would then be irrelevant (within a universality class).

In this framework for learning one distinguishes two independent proposals: that of self-organized criticality, and that of democratic reinforcement applied to the structure of coarse-grained operators.

In this article, we will consider only the first of these two proposals: We will construct a system where the recall of previously-viewed vectors proceeds by "avalanches". Specifically, we will look for evidence for *self-organized criticality* [6] in a modification of the backpropagation learning algorithm for feedforward neural networks.

To this end, we will create a "critical pile" where previously-viewed data vectors are stored. Each displacement of a data vector in the pile is associated with a learning iteration. The introduction of a new data vector can provoke an avalanche in this pile, which propagates through a possibly large set of previously-viewed data. Thus, a single new data vector can force the system to recall previous experiences, and in the case of a large avalanche, provoke a long jump in synapsis space.

2. ERROR BACKPROPAGATION IN FEEDFORWARD NEURAL NETWORKS

In order to choose a specific model on which to test some of these ideas, we opted for the possibility of practical applications over biological realism. The model which will be discussed below is a generalization of a neural network model which has a track record of successes in a wide array of modelling tasks: the so-called "backpropagation network".

The backpropagation network is a parametric model $f_{\mathbf{w}}(\mathbf{x})$ of a map $y: \mathbb{R}^n \to (0,1)$, given by (Fig. 2)

$$f_{\mathbf{w}}(\mathbf{x}) = g\left(\sum_j w_j^{(2)} x_j^{(2)}\right),$$

where $g(x) = \frac{1}{1+e^{-x}}$ is the neural "transfer function" and $x_j^{(2)}$ is the activation of the so-called "hidden neurons", given by

$$x_j^{(2)} = g\left(\sum_i w_{ji}^{(1)} x_i\right).$$

The synaptic weights $w_{ji}^{(1)}$ and $w_j^{(2)}$ are free parameters, to be determined by the "learning" procedure.

In the language of the Introduction, the "algorithm space" is spanned by the weights, or "synapses"; the computer which executes the algorithm is the neural network running on a training set of input-output vectors, $\{(\mathbf{x}_k, y_k);\ k = 1, 2, \ldots, P\}$. The "fitness" is defined as the inverse of the mean squared error on this set,

$$\epsilon \sim \sum_k (y_k - f(\mathbf{x}_k))^2 .$$

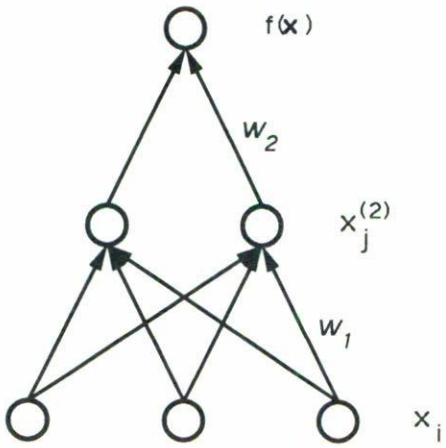

FIGURE 2. A feedforward neural network. The middle units, called "hidden neurons" receive an activation potential which is the weighted average of the inputs, and respond with an output given by the transfer function. Likewise for the output unit, only its activation potential is determined by the outputs of the hidden neurons.

The backpropagation learning algorithm [5] is a steepest–descent optimization method applied to this error function, where the corrections to the weights are computed one training vector at a time. One "epoch" of learning consists of a single presentation of each vector in the data set. A complete training process typically involves a large number of epochs.

We will focus for now on the fine-grained learning operator which consists of the presentation of a single data vector. For each training vector $\{\mathbf{x}_k, y_k\}$ one has a learning operator $\delta_\mathbf{k}$ whose action on the weights is proportional to the gradient of the error function evaluated at this point:

$$\delta_k w_j^{(2)} = -\eta(f_k - y_k)\, f_k(1 - f_k) x_j^{(2)},$$

$$\delta_k w_{ji}^{(1)} = -\eta(f_k - y_k)\, f_k(1 - f_k)\, w_j^{(2)}\, x_j^{(2)}(1 - x_j^{(2)}) x_i$$

(with the notation $f(\mathbf{x}_k) \equiv f_k$). The operators δ_k generate the fine-grained evolution of the weights. It is usual to apply these operators in random order. This strategy allows the system to escape from shallow local optima, as follows. If the weights are at a local optimum of the error averaged over the entire training set, they are generally not at a local optimum of the error averaged over a *proper subset* of training vectors. So if one considers the subset of m consecutive vectors in the random ordering of the training vectors, out of a total of N data vectors, training over this subset will generate a net movement in weight space. For $m \ll N$ this is essentially a random walk, so the random presentation of training vectors allows the learning algorithm to explore a region in weight space within a radius proportional to $\eta\sqrt{m}$ around the local optimum.

One can think of such a subset of m learning iterations as one coarse-grained learning operator —so the practice of presenting the training vectors in random order is seen to provide a class of medium-jump learning operators, *i.e.*, operators that generate displacements that are large compared to δ_k but small compared to the landscape correlation

length. As mentioned above, this allows the system to escape from "shallow" local minima, whose attracting radius is small compared to the landscape correlation length. For larger values of the learning rate these become long-jump operators, but then the convergence of the learning algorithm is lost. This should come as no surprise since, by definition, random long jumps in weight space lead to a random (uncorrelated) fitness.

The effective landscape at finite learning rate differs from the "bare" landscape in that it disfavors steep valleys: the expectation value of the fitness with the fluctuating weights reflects how quickly fitness deteriorates as one moves away from the optimum. At large values of the learning rate the effective force caused by a widening of the valley can dominate over that which reflects the slope at the bottom of the valley. Clearly, a better method to escape from local optima would be helpful.

Our proposal in this paper is to let the presentation of training vectors be given by the mechanics of a critical data pile, rather than a random process. In this way most iterations will consist in small or medium "jumps" in synapsis space, with large avalanches coming in as rare punctuations. Eventually, one would like to let the exploration away from the local optimum self-organize through a reinforcement loop, to reach the objective of an intelligent search —the "intelligence" would then manifest itself through the choice of particular subsets of training vectors which comprise the "avalanches"— however, the implementation of such a reinforcement loop lies beyond the scope of this work. In summary, criticality has to do with the size of a training subset, while the emergent topology (which we do not consider here) defines its structure.

We will close this section with a few facts about the structure of the landscape $\epsilon(\mathbf{w})$ and their significance to the backpropagation learning algorithm.

2.1. Compensation through different hidden neurons

As long as the activation potential of the hidden neurons is small, the transfer functions act as linear maps: for small x,

$$g(x) = \frac{1}{1+e^{-x}} \approx \frac{1}{2}\left(1+\frac{x}{2}\right).$$

In this regime, an increase in the product of the weights from a given input to the output through one hidden neuron can be compensated by an opposite change in the product of weights through another hidden neuron, with no change in the output. Away from the linear regime this sort of compensation is imperfect, but there is a remnant which makes for quasi-horizontal directions in the landscape. There is one such direction for every set of input neuron and other hidden neuron, so one concludes that most directions are approximately "irrelevant". At the bottom of an error valley, the relevant displacements in weight space are deleterious so learning proceeds mostly along the irrelevant directions.

2.2. Overlearning

For large values of the weights, the activation potential of the hidden neurons, which is the weighted mean of inputs, tends to be large in absolute value. Consequently, the neural

activation saturates at either 0 or 1, since $g(x) \to 1$ at $x \to \infty$ and $g(x) \to 0$ as $x \to -\infty$. This saturation of the hidden neurons can be avoided only by a careful balancing act, where very large positive and negative contributions from the various input neurons must balance to achieve a neural activation potential near zero. The output then becomes highly sensitive to small changes in the input vectors (a phenomenon known as *overlearning*), and the landscape becomes more and more "rugged" as one moves away from the origin in weight space.

2.3. Random walk in the irrelevant subspace

In practical applications one normally works with a large number of weights, typically of the order of 100. The number of "irrelevant" directions is similarly large, and learning proceeds along these directions mostly at random since the gradient of the error function is poorly defined, so once the system has reached the bottom of a valley learning proceeds as a random walk in the irrelevant subspace. This implies that the norm of the weight matrix grows as the square root of the number of epochs.

Combining these observations, one finds that the structure of the landscape is that of a network of winding valleys that become narrower and narrower as learning proceeds and one is driven away from the origin in weight space. One frequently finds that the learning procedure takes the network along a long path which eventually folds back on itself and returns to a point quite close to the original one, albeit separated from it by a steep barrier. There may or may not be local minima, depending on the function $y(\mathbf{x})$ and the neural architecture, but even in cases where local minima are not present it would be clearly advantageous if one could go through the steep barriers between valleys rather than following the long path along the irrelevant directions. Of course this wish is as old as the neural networks themselves, and we do not pretend to answer it here. Our aim is to investigate if and how long-jump operators can emerge as an avalanche of small backpropagation steps, as described in the Introduction.

3. A SANDPILE OF TRAINING VECTORS

Stable criticality is a property displayed by certain *forced many-body systems* with *non-linear near-neighbour interactions* and unrestricted energy escape through a *free boundary*. The forcing energy propagates through the system from the input site to the boundary, and the non-linear interactions imply that this propagation proceeds through a process of avalanches, rather than a steady flow. The criticality is an approximate concept, as the system is usually slightly subcritical right after an avalanche and the finite size of the system imposes a large-scale cutoff, but nevertheless it is reflected in the power-law distributions of avalanches and correlations at a certain scale.

The prototype of "self-organized criticality" is the sandpile model [7, 8]: A slow input of grains of sand at the top of the pile forces avalanches which involve non-linear friction forces between the grains of sand. If the pile is placed on a table, the edge of the table is the free boundary. When the pile reaches its asymptotic size, one finds that avalanches occur with a power-law distribution in the number of grains of sand involved.

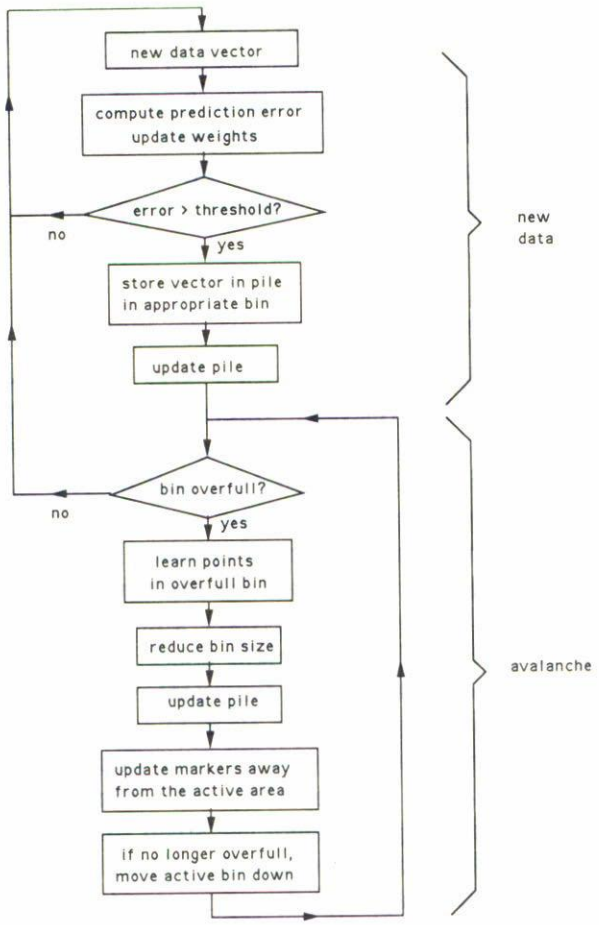

FIGURE 3. Organigram of the critical neural network model.

To design a critical backpropagation network, we will construct an analogy between grains of sand and training vectors. The tumbling of a grain of sand should be related to a learning iteration, so it is natural to assume that the "height" is the network prediction error for a training vector.

The algorithm is as follows (Fig. 3). Training vectors are presented one at a time; in each presentation the prediction error is calculated and the synaptic weights are updated according to the backpropagation algorithm. If the prediction error exceeds a tolerance threshold, then the training vector is stored in a pile which consists in a fixed number of error bins, equal to 100 in our simulations. These "bins" are specified by bounds on the prediction error, which we shall refer to as *markers*. Thus, bin number k will contain the data vectors for which the prediction error lies between $marker[k]$ and $marker[k+1]$.

Each error bin has a capacity limit; if it is exceeded, an "avalanche" is initiated which will stop only when all bins in the pile have returned to below the capacity limit. The avalanche proceeds as follows. All points in the overfull bin are presented for one learning iteration. The markers for the lower and upper bounds of this bin are then adjusted so as

to reduce the size of the bin, according to the algorithm

$$marker[k+1] \rightarrow marker[k+1] - \epsilon(marker[k+1] - marker[k]),$$
$$marker[k] \rightarrow marker[k] + \epsilon(marker[k+1] - marker[k]), \quad (1)$$

where $\epsilon = 0.01$ in our simulations. Note that both the learning iterations and the movements of the markers tend to reduce the occupancy of the overfull bin. We will refer to both processes together as an event, and refer to the bin in question as *active.*

Following an event, the weights of the neural network have been modified so it is necessary to recalculate the errors of all training vectors in the pile and reassign them to error bins.

If overfull bins are found either above the active bin or below its lower neighbour, it is assumed that the cause is the change in weights rather than the movement of one of the vectors in the active bin —indeed, one expects that the learning process should push one or more data vectors down to the bin immediately below the active one. The markers are adjusted by the algorithm (1) until there are no overfull bins except possibly for the active one or its lower neighbour.

One then checks the occupancy of the active bin. If it remains overfull, the process is repeated starting with one learning iteration for each training vector in this bin. If not, then one checks the occupancy of the next lower neighbour. If it in turn exceeds the capacity limit, the process described above is repeated for this new "active bin".

Thus, the "activity" proceeds downward one bin at a time until every bin's occupancy is below the capacity limit. In this entire process the lowest marker, which marks the "tolerance threshold" below which points are discarded from the pile, is kept fixed.

When the prediction error on a training vector falls below the tolerance threshold ($marker[0]$), this vector is considered to have been "learned" and is dropped from the pile. What value should this "tolerance threshold" take? For very large values of the threshold, training vectors will be easily "learned" and the pile will maintain a low occupancy, unlikely to lead to extensive avalanches. If, on the other hand, the threshold is low, training vectors will accumulate while the network struggles to meet the accuracy demand; any new training vector then generates an avalanche which propagates through the entire pile, until extensive learning iterations on the lowest error bin finally succeed in reducing the error for one of the vectors to below the threshold. In-between these two extremes one expects to find a "critical" threshold where the distribution of avalanche sizes follows a power law.

The tolerance threshold is allowed to self-organize to the critical value, as follows. At every presentation of a new training vector, the threshold is lowered by

$$marker[0] \rightarrow marker[0] - 0.1 \times inf\{marker[0], marker[1] - marker[0]\} \quad (2)$$

After each avalanche, the marker is raised by this amount multiplied by 0.05 times the number of events. Thus, the threshold reaches an equilibrium value when the average avalanche involves 20 events.

Numerical experiments were carried out with a pile of 100 bins, each with the capacity to hold 10 training vectors. The network architecture was chosen with 6 input neurons, 8

hidden neurons and one output, and "learning" proceeded with 50,000 randomly-generated training vectors. The use of random training vectors is intended to focus on the issue of criticality rather than the structure of the landscape —in particular, in deterministic landscapes one often finds that learning proceeds by "jumps" independently of any emerging property, simply because the landscape includes steep dropoffs from one valley floor to a lower one. When training on random data, the individual (fine-grained) learning steps have comparable sizes.

The code was designed to keep a record of several characteristics of the avalanches:

1. The number of events in response to the presentation of each training vector (sizes of avalanches).
2. The number of times the activation of one bin eventually leads to the activation of the n'–th lower bin (spatial correlation).
3. The number of consecutive presentations of training vectors that do *not* lead to an avalanche, or "waiting time" (temporal correlation).

4. EVIDENCE FOR STABLE CRITICALITY

The subject of self-organized criticality has received much attention in the recent literature; it is mostly an empirical concept at this point, so we will attempt to be thorough and provide an extensive list of observations which can be expected at criticality.

1. Two time scales: one, slow, for the introduction of new energy in the system: we will call T the time for complete renewal of the system's energy. The other, fast, for the propagation of avalanches.
2. Power-law behaviour of the temporal correlation functions, with a cutoff at T.
3. Two clearly distinguished spatial scales: the range of near-neighbour interactions which separates individual motions in the avalanche, and the total size of the system, N.
4. Power-law distribution in sizes of avalanches, with a cutoff at N.
5. Power-law behaviour of the spatial correlation functions, with a cutoff at N.
6. Existence of a self-organizing parameter (*e.g.*, the slope of a sandpile).
7. Loss of the properties (2–5) above when the self-organizing parameter is forced away from its critical value.

In order to look for signs of self-organized criticality in our model of critical backpropagation, we ran the model described in the previous section on a set of 50,000 randomly-generated data vectors.

1. New data vectors are introduced one at a time, and any ensuing avalanche is completed before a new vector is presented. Thus, the time scale for the execution of an avalanche is $\tau = 1$ and that for complete renewal of the data vectors is equal to the average number of vectors in the pile, which turns out to be $T \approx 500$ (the maximum capacity of the pile would be $10 \times 100 = 1000$).

2. The distribution of waiting times between avalanches is given in a log-log plot in Fig. 4: this experiment does not provide convincing evidence of power-law behaviour in the temporal correlations.

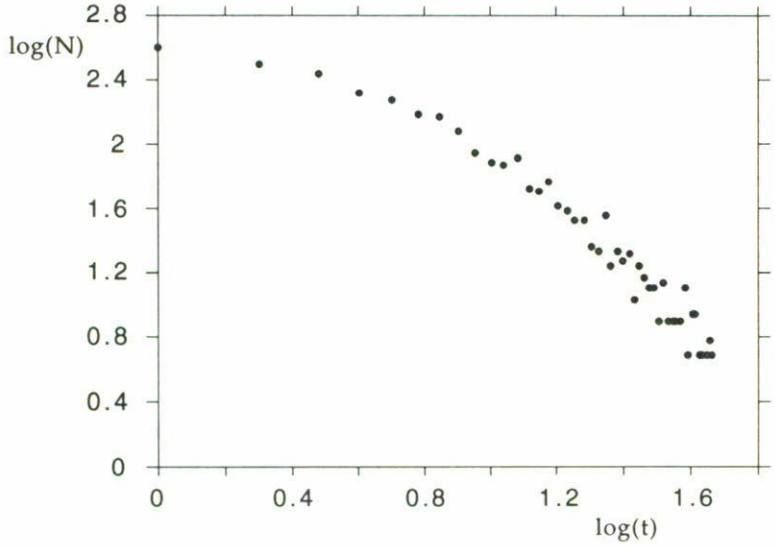

FIGURE 4. Log-log plot of the probability distribution of waiting times between avalanches. No linear region can be identified; this graph reflects instead an exponential dropoff of the probability of large waiting times.

3. The total size of the system is equal to the number of bins, or $N = 100$ in our case. The interactions are between neighboring bins only, by design.

4. The distribution of avalanche sizes is given in Figs. 5 and 6. In the first case, the total number of events is considered, while in the other one considers only the number of bins visited by the avalanche without regard to how many times the bin is visited. Both follow power law behaviour in the range $10^{0.2} < size < 10^{1.4}$, with critical exponents $\gamma_1 = 2.0$ (events), and $\gamma_2 = 2.75$ (bins visited).

5. The spatial correlation function is the probability that the activation of bin number k generates an avalanche which eventually activates another bin n positions lower. We find a power-law decay as a function of n, with a cutoff at the total number of bins (Fig. 7) and an exponent $\gamma_3 = 1.45$.

6. The self-organizing parameter is the "slope" of the markers, *i.e.*, the difference between the levels of neighbouring markers. As seen is Fig. 8, the markers settle at asymptotic error values determined by a linear function of the bin number. Thus,

$$marker[i+1] - marker[i] = \alpha_c,$$

where $\alpha_c = 0.00061$. The linear disposition of the markers is a consequence of the algorithm (1): the size of bins is reduced each time it becomes overfull, so the system converges to where each bin has is equally likely to become overfull. For random data, the distribution of errors is roughly flat in the low-error range, so the bins are equally likely to become overfull if they have the same size. Large errors are less frequent, so the bins near the top of the pile tend to be larger.

7. One can enforce a different slope by freezing the lowest marker. For large values of the lowest marker, $\alpha < \alpha_c$ and most points are learned with no avalanches or with only

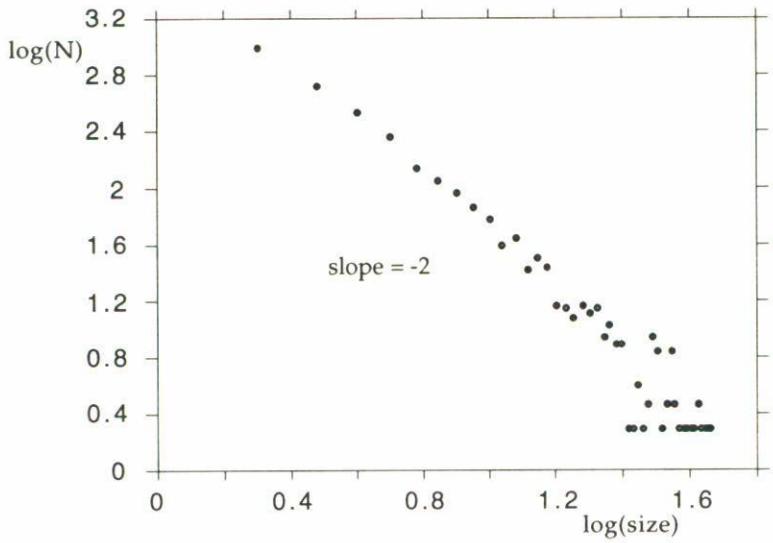

FIGURE 5. Power-law behaviour of the distribution of the sizes of avalanches. The "size" is the total number of backpropagation iterations which resulted from presenting a new training vector.

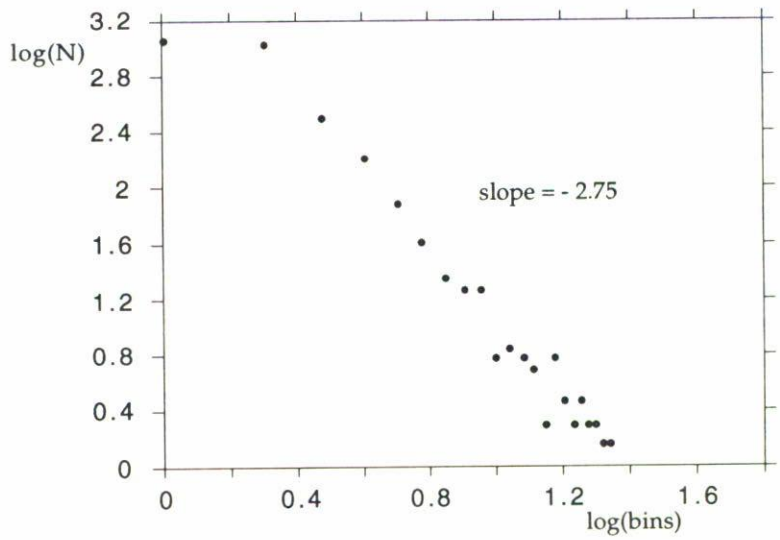

FIGURE 6. The number of bins visited by an avalanche, as the size, has a power-law distribution.

short ones. The distribution of avalanche sizes shows an exponential suppression of large events. *Viceversa*, with $\alpha > \alpha_c$ the system eventually reaches a state where it fails to learn the data vectors and iterates indefinitely on overfull bins.

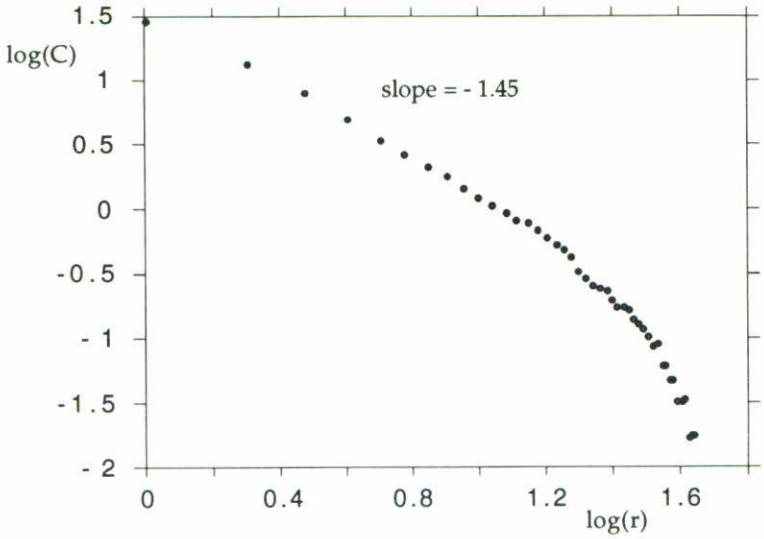

FIGURE 7. The correlation between the activity of bins decays as a power of their separation. The cutoff at large separations reflects the finite size of the sandpile.

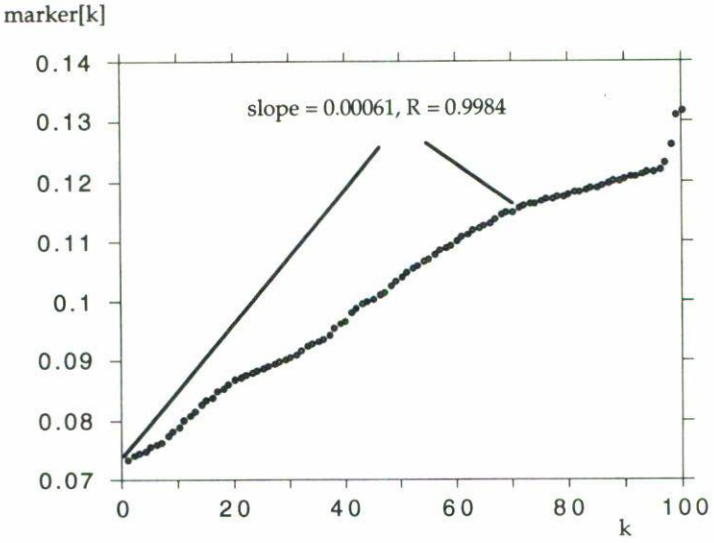

FIGURE 8. The markers which separate squared-error "bins" organize according to a critical slope, discarding edge effects at both extremities. If the markers are frozen at values which lead to a slope larger than this one, supercritical behaviour results. Viceversa, if the slope is forced to a lower value then the system becomes subcritical and one has an exponential cutoff of large avalanches.

5. CONCLUSIONS. APPLICATIONS IN ADAPTIVE SYSTEMS MODELLING?

We proposed a set of two conjectures on the role of long-jump operators in learning. Both are based on the hypothesis that *coarse-grained learning operators emerge as a consequence of a memory recall mechanism* whereby the presentation of a new data vector can stir up

previously viewed data and provoke a series of small learning steps. The first ansatz is that this recall mechanism self-organizes to a critical state, so that one has a power law in the *sizes* of the coarse-grained operators. The second one is that the principle of "democratic reinforcement" operates not at the microscopic level (fine-grained learning operators) but at the level of the coarse-grained operators, by specifying *which* of the previously viewed data vectors are recalled and in what order. In this way, in the struggle to understand neural processes we shift the focus away from the individual learning steps and towards the emergence of *structure* in the coarse-grained learning operators.

We proposed a toy model centered around a critical data pile. In this model, the data vectors are organized according to the level of error; overcrowding (many vectors with a similar error) triggers learning iterations which propagate through the pile as "avalanches". We described evidence that the pile self-organizes to a critical state.

One of the motivations for this work is the idea that organized avalanches have a role to play as a long-jump operator in the learning process. In order to test this idea in our code, we attempted training with a logistic map with a fluctuating gain parameter:

$$x_{n+1} = \mu x_n(1 - x_n),$$

where μ is switched at random with a probability $p = 0.1\%$, between the values

$$\mu_1 = 3.69, \mu_2 = 3.99.$$

To evaluate the usefulness of the critical learning scheme, we compared the results to a sliding-window training scheme. In these schemes, the learning algorithm is carried out over data vectors from the present time t to $t - \tau$, and τ is chosen to minimize average mean squared error. The results indicate that adaptation with the critical model proceeds roughly twice as fast as with sliding windows. More specifically, the number of new data vectors which must be presented before the mean squared error recovers half-way to the asymptotic fixed-μ value is half as much as with the optimal sliding window.

What happens to an organized pile when the system is changed? Since the new training vectors are not recognized by the neural network, the hidden neurons are found to saturate most of the time, one way or the other. The network thus produces an output which sees only a broad categorization of the input vector, so that many new training vectors are assigned the same output. This leads to an accumulation of most of the new training vectors in only a few bins, which quickly become overfull. In this way, avalanches are initiated on the new data and proceed until the system has learned the new data sufficiently well.

Thus, we have two of the ingredients required for a good model of adaptive systems: system changes are detected and can be recognized through the behaviour of the system, and they automatically trigger extensive learning with a certain bias towards the new data. The result (a factor of two drop in the adaptation time) is not very spectacular because the avalanches are not sufficiently organized to focus training accurately on the new data: a better definition of "near-neighbour" is required, so that avalanches propagate mainly on data vectors from the same system (same value of μ, in this case). If the distance function on the space of training vectors is allowed to self-organize to improve

the result of learning avalanches (by a "democratic reinforcement" procedure), one expects that a training vector from one of the two systems will be recognized to be the "near-neighbour" of other data vectors from the same system, so that a learning avalanche will recall training vectors only from the system currently producing the data. Work is underway to implement a reinforcement loop which should lead to the desired structure in the coarse-grained operators.

ACKNOWLEDGEMENTS

One of us (HW) would like to thank Ricardo García Pelayo for helpful discussions on the topic of stable criticality, and Chris Stephens for continuously stressing the importance of coarse-grained degrees of freedom in physics.

REFERENCES

1. P. Stadler, "Towards a Theory of Landscapes", in R. López Peña *et al.* (Eds.), *The Guanajuato Lectures on Complex Systems and Binary Networks*, Lecture Notes Series, Berlin, Springer-Verlag (1995).
2. T. Kohonen, in *Proceedings: World Congress on Neural Networks —San Diego*, New Jersey, Lawrence Erlbaum Associates, Inc., and INNS Press (1994).
3. D. Stassinopoulos and P. Bak, *Phys. Rev.* **D51** (1995) 5033.
4. K. Sneppen, P. Bak, H. Flyvbjerg and M.H. Jensen, *Proc. Natl. Acad. Sci. USA* **92** (1995) 5209.
5. J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the Theory of Neural Computation.* Redwood City, CA, Addison-Wesley (1991).
6. P. Bak and M. Paczuski, *Proc. Natl. Acad. Sci. USA* **92** (1995) 6689.
7. P. Bak, C. Tang and K. Wiesenfeld, *Phys. Rev.* **A 38** (1988) 364.
8. R. García-Pelayo, I. Salazar and W.C. Scieve, *J. Stat. Phys.* **72** (1993) 167.