THE MATHEMATICAL ROLE OF SELF-CONSISTENCY IN PARALLEL COMPUTATION

Paul Smolensky

Institute for Cognitive Science C-015
University of California, San Diego
La Jolla, CA 92093

Analysis of Emergent Properties of Neural Systems

One approach to the mind/body problem is to view the description of mind as a higher level description of brain; to view psychological principles as emergent properties of neural systems. Certainly before such a view can be scientifically tested, a better understanding of both brain and mind must be established. However enough is already known about each to make feasibility studies possible.

What methodology is capable of analyzing the emergent properties of large complex systems of interacting elements? One discipline where this job needs to be done is statistical physics, where large-scale properties of matter are derived mathematically from the principles believed to govern the interactions of molecular and sub-molecular constituents.

Is it possible to apply similar kinds of mathematical analysis to deduce emergent properties of neural systems? Although the principles governing neuronal interaction are by no means as well understood as those governing particles, models that abstract some of the characteristics of neural networks have been studied for some time. Hopfield (1982) has shown that with certain modifications, standard neural models can be analyzed with mathematics much like that of statistical physics, and emergent properties can be analyzed.

One of the central concepts in statistical physics is *temperature*. The utility of this concept in performing difficult computations has been shown by Kirkpatrick et. al. (1983). However the most important concept in statistical physics, as in all branches of physics, is that of *energy*. The meaning of "energy" in the computational context is not obvious; rather than a computational interpretation, Hopfield offered a general formula for the "energy" of a neural net while Kirkpatrick et. al. hand crafted "energy" formulae for their particular computations.

The application of statistical physics concepts to computation is now a rather active field of study (Hinton and Sejnowski, 1983; Hofstadter, 1983; Geman and Geman, 1983). To provide a solid foundation for this analysis, what is required in my opinion is *an interpretation of "energy" that establishes a deep connection between the formalism of statistical physics and the central problems of cognition.*

The help of David Rumelhart, Francis Crick, and other members of the UCSD Parallel Distributed Processing research group is gratefully acknowledged.

This research was supported by a grant from the System Development Foundation and by contract N00014-79-C-0323, NR 667-437 with the Personnel and Training Research Programs of the Office of Naval Research.

In this paper I will present the interpretation of "energy" that lies at the heart of a general computational approach I have been developing independently of the work of those interested in neural nets or in particular difficult computations. In this interpretation, *"energy" is a measure of the self-consistency of a computational state*. In place of the term "energy", which emphasizes the physical analogy, or the more technical term "Hamiltonian", which serves only to recall history and account for the physicist's notation H, I choose to foreground the measurement of self-consistency by using the term *harmony function*, denoted H. The general framework, *harmony theory*, is described in Smolensky (1984); an analysis of learning using this theory is begun in Smolensky (1983), and an application of the theory to modelling qualitative analysis of a simple electric circuit (with a discussion of the model's emergent properties) is described in Riley and Smolensky (1984). In this paper I will focus on the computational meaning of harmony, passing quickly over other aspects of the theory. The treatment will be very informal; for more formal presentations the reader is referred to the previously cited papers.

The Role of Harmony in Computation

Before considering how the harmony function is *defined*, we start with a discussion of how the harmony function is *used* during computation. The basic idea can be framed at a very general level. During computation, search for an answer is guided by a measure of "goodness" of possible answers: the harmony function H is that measure. The search is stochastic; the computation is a Monte Carlo random walk through the solution space under the guidance of H. The random walk is designed so that eventually, the probability at any moment of visiting a point p in the solution space is given by the *canonical distribution*:

$$\text{prob}(p) = Ne^{H(p)/T}$$

N is the constant needed to normalize the probabilities so that they sum to one. T is a global parameter that determines the spread in the probability distribution.

The canonical distribution is the only continuous relationship between H and probability that correctly treats the independence of components of a computation. The canonical distribution also happens to be the distribution on which most of statistical physics is based. (This is no coincidence, as the notion of independent subsystem in physics maps onto that of independent subcomputations.) There is an isomorphism that maps the harmony function into minus the Hamiltonian (energy) function, and T into temperature. This suggests calling T the *computational temperature* of the system.

In physics, the Hamiltonian determines what states are most probable: the states with lowest energy are most probable at all temperatures, and states of high energy have negligible probability except at high temperatures. In harmony theory, the harmony function determines what states are most probable: the states with highest harmony are most probable at all computational temperatures, and states of low harmony have negligible probability except at high temperatures. T can be thought of as setting the *scale* for what constitutes significant differences in harmony values. In fact, the ratio of probabilities of two states is $e^{\Delta H/T}$, where ΔH is the difference in harmony between the states. If this difference is small compared to T, the ratio of probabilities will be close to one; if ΔH is large compared to T, the state with higher harmony will be many times more probable.

The goal of the computation is to find the state of highest harmony. This means, in particular, that the state of next highest harmony should be much less likely. This requires that T be small compared to the harmony difference between the two highest levels of harmony.

We could simply set T to be such a low value and be done with it. However, this is not a practical search procedure. The Monte Carlo procedure will, if let run long enough, visit points with the probabilities given by the canonical distribution. However, the time required to reach this "thermal equilibrium" grows extremely rapidly as T is lowered. A more practical way of zeroing in on the state of highest harmony is to start with a high temperature and gradually lower it. Early in the search, only large harmony differences are significant, and the system quickly makes a crude cut at the problem, avoiding states of extremely low harmony. As the system cools down, smaller harmony differences become significant, and more and more states are avoided as the search focusses on states with harmonies close to the maximal value. If the cooling is done gently, the state of maximal harmony should be found in *much* less time than by giving T a constant low value.

The Relation of Harmony to the Environment

We have discussed a stochastic search technique that will find states of high harmony. But how do we design the function H so that the states with high H values give the correct solutions to problems? Now we must discuss the sense in which H measures self-consistency.

The "correct" answer to problems are often those that satisfy a set of rules. In the circuit analysis problem considered by Riley and Smolensky, for example, the rules are the physical laws of simple circuits. Any system that can correctly solve problems such as this must in some sense have a representation of the rules. In harmony theory, the rules are encoded in the harmony function. The question is, how are these rules encoded, and how can a system develop an appropriate harmony function through experience?

Of course most cognitive tasks are not as strictly governed by rules as is formal problem solving. Yet all cognition hinges on the *exploitation of regularities in the environment*, even if those regularities are less formal than Ohm's Law. Cognition enables organisms to do the *completion task*: take some limited information about the current state of their environment and make reasonable guesses about what else is likely to occur in the environment. That is, given *some* of the features that specify the environmental state, the organism can make reasonable guesses about missing features.

In harmony theory, the "rules" applied during the completion task are simply *statements that certain features can co-occur in the environment*. In the circuit application, for example, in place of a symbolic version of Ohm's Law, $V = IR$, there are many "rules" that each record a single combination of qualitative changes in V, I, and R that are consistent with the law. These "rules" can in fact be thought of as *memory traces* that might be left behind by individual experiences in the environment in which the regularities hold.

Here is the general idea of how to set up a harmony function for performing the completion task in a given environment. Imagine the system experiencing many encounters with the environment; each leaves many traces that each record some of the features that co-occurred. When partial information about the current state of the environment is given in a completion problem, the harmony of a possible completion of that information is *the overall consistency between that completion and the set of all traces*. To spell this out, we consider first how the traces are determined and then how the "overall consistency" is computed.

The traces can be produced automatically by simulating exposure to an environment, or they can be produced manually by the modeller. The latter technique was used in the circuit problem: each trace was chosen to be an allowed combination of qualitative changes in the circuit quantities appearing in a single circuit law. The automatic generation of traces is yet to be explored; the idea is

that traces would be produced in a random fashion (guided by the degree to which potential traces would enhance system harmony); the *statistical properties* of the resulting set of traces would then govern the emergent behavior of the system.

How is "the overall consistency between a completion and the set of all traces" computed? The idea here is that for each trace, a decision needs to be made whether the instance it recorded is relevant to the current situation or not. Borrowing the usage of schema theory, a match between part of a trace and a completed set of environmental features can cause the trace to become *active*. The "overall consistency" – the harmony – of a completion is the sum over all active traces of a measure h of the degree of match between the trace and the completion. A simple definition of h is the number of features in the completion that match the trace, minus the number that do not match. (A slightly more complicated definition of h was used in the circuit analysis model.)

There are now two kinds of variables used in the computation: features of the environmental state, and activation values for traces. The processing has two components: computing the harmony values of possible completions, and making corresponding random decisions about which completions to visit. Computation of the harmony value requires deciding which traces to activate, which requires computing the quality of match h between traces and the completion. Just as the Monte Carlo search is used to decide what completions to visit, it can be used to decide what traces to activate. So using the traces to define the harmony of completions leads naturally to extending the search space to include both environmental feature variables *and* trace activation values.

The Network Interpretation: A Computer Implementation

It is useful to represent the computation by a network like that shown in Figure 1, which shows a portion of the network for the circuit model. The activation variables are represented by nodes in the upper layer; each corresponds to a trace. The environmental feature variables are represented by nodes in the lower layer. There are connections between a trace variable and all the environmental features it incorporates. For simplicity all variables (nodes) are taken to have binary values: trace activation nodes have values *active* and *inactive*; environmental feature nodes have values *present* and *absent*.

The Monte Carlo search in this network representation proceeds as follows. Initially a high temperature T is chosen, all the traces are set inactive, the environmental features are permanently assigned their given values, and the remaining environmental feature variables are assigned random initial values. Then processing begins. A node is selected at random (but not one of the given features). Next the difference ΔH between the overall network harmonies that would result from the two possible values for the node is computed. This computation, it turns out, can in principle be performed in the node itself, for the only quantities needed are those to which the node is connected. Finally, the node randomly selects a new value, using as the ratio of probabilities for the two values $e^{\Delta H/T}$. The process of selecting a node and selecting a value for that node is iterated while the temperature T is gradually lowered according to some schedule.

The repeated selection of nodes and assignment of new values can be viewed (following Hopfield) as the asychronous processing of processors located at the nodes and running in parallel. The relation between this parallel processing network and those considered by Hopfield and Hinton and Sejnowski is that the harmony model has a special architecture: there are two classes of nodes, and connections between but not within the two classes. The formula for harmony turns out to be minus that for Hopfield's network "energy", taking into account the special architecture and the numerical assignments *active* $= 1$, *inactive* $= 0$; *present* $= 1$, *absent* $= -1$.

Comments on Neural Implementation

Since harmony theory is computationally- rather than neurally-inspired, the relation between the harmony network and neural networks has not been developed. However the close resemblance of the harmony network to Hopfield's neural network might suggest that harmony nodes correspond to neurons, so a brief comment is appropriate. While it does not seem unreasonable in principle to identify environmental feature nodes with neurons, it is *not* reasonable to to identify trace nodes with neurons. Indeed, I imagine that each trace is distributed over the synapses of the neurons corresponding to the environmental features involved in that trace. "Activation" of the trace might correspond to a feedback-mediated rapid enhancement of the strengths of these synapses, as in von der Malsberg (1981). In this sense, even the activation of traces, a primitive operation in the theory as presently formulated, may be an emergent property of synaptic dynamics.

Even without a precise specification of the relation between harmony networks and neurons, harmony theory offers a mathematical framework within which to explore the emergence of mind from brain-like processing. The isomorphism between computation and statistical physics which it represents rests on the identification of self-consistency – harmony – as playing a central role isomorphic to that played by energy in physics.

References

S. Geman and D. Geman (1983). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. Manuscript.

G. E. Hinton and T. J. Sejnowski (1983). Analyzing cooperative computation. *Proceedings of the Fifth Annual Conference of the Cognitive Science Society*. Rochester, NY.

D. R. Hofstadter (1983). The architecture of Jumbo. *Proceedings of the International Machine Learning Workshop*. Monticello, IL.

J. J. Hopfield (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, 79, 2554-8.

S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi (1983). Optimization by simulated annealing. *Science*, 220, 671-80.

P. Smolensky (1983). Schema selection and stochastic inference in modular environments. *Proceedings of the National Conference on Artificial Intelligence*. Washington, DC.

P. Smolensky (1984). Harmony theory: thermal parallel models in a computational context. Manuscript.

M. S. Riley and P. Smolensky (1984). A parallel model of (sequential) problem solving. Submitted to the Sixth Annual Conference of the Cognitive Science Society. Boulder, CO.

C. von der Malsburg (1981). The correlation theory of brain function. Internal report 81-2. Department of Neurobiology, Max Planck Institute for Biophysical Chemistry, Gottingen, W. Germany.

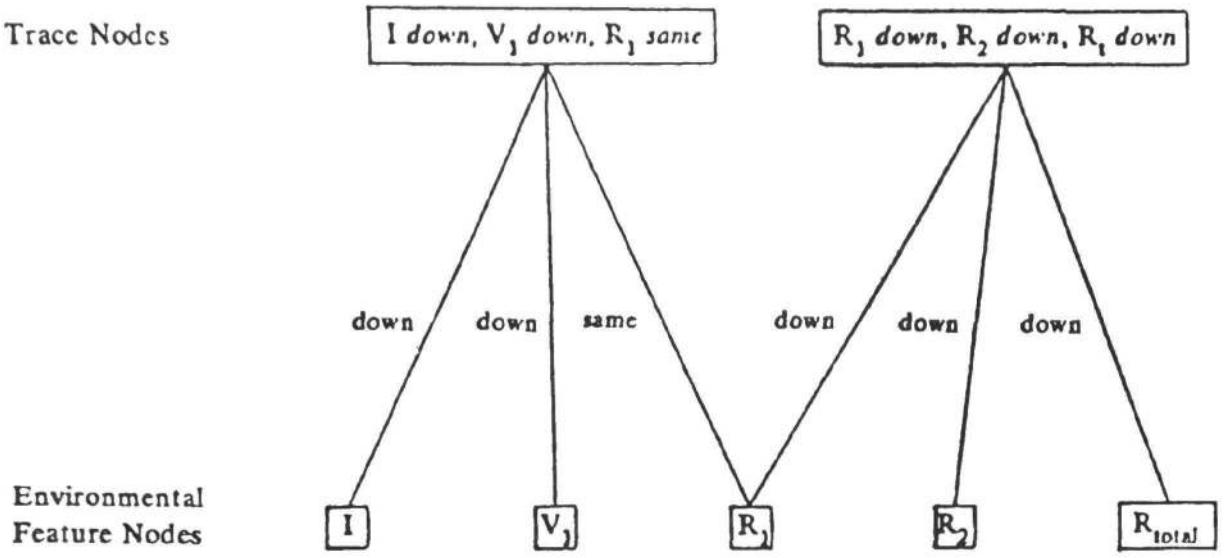

Figure 1. A portion of the network representation of the circuit analysis model (from Riley an Smolensky). [The values *up*, *down*, *same* for environmental features (circuit variable changes) are actually represented by using two binary nodes for each variable.]