JOURNAL OF COMPLEXITY **4**, 193–215 (1988)

# On the Capabilities of Multilayer Perceptrons

ERIC B. BAUM

*Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109*

Received April, 1987

What is the smallest multilayer perceptron able to compute arbitrary and random functions? Previous results show that a net with one hidden layer containing $N - 1$ threshold units is capable of implementing an arbitrary dichotomy of $N$ points. A construction is presented here for implementing an arbitrary dichotomy with one hidden layer containing $\lceil N/d \rceil$ units, for any set of $N$ points in general position in $d$ dimensions. This is in fact the smallest such net as dichotomies which cannot be implemented by any net with fewer units are described. Several constructions are presented of one-hidden-layer nets implementing arbitrary functions into the $e$-dimensional hypercube. One of these has only $\lfloor 4N/d \rfloor \lceil e/\lfloor \log_2(N/d) \rfloor \rceil$ units in its hidden layer. Arguments based on a function counting theorem of Cover establish that any net implementing arbitrary functions must have at least $Ne/\log_2(N)$ weights, so that no net with one hidden layer containing less than $Ne/(d \log_2(N))$ units will suffice. Simple counts also show that if the weights are only allowed to assume one of $n_g$ possible values, no net with fewer than $Ne/\log_2(n_g)$ weights will suffice. Thus the gain coming from using real valued synapses appears to be only logarithmic. The circuit implementing functions into the $e$ hypercube realizes such logarithmic gains. Since the counting arguments limit below only the number of weights, the possibility is suggested that, if suitable restrictions are imposed on the input vector set to avoid topological obstructions, two-hidden-layer nets with $O(N)$ weights but only $O(\sqrt{N})$ threshold units might suffice for arbitrary dichotomies. Interesting and potentially sufficient restrictions include (a) if the vectors are binary, i.e., lie on the $d$ hypercube or (b) if they are randomly and uniformly selected from a bounded region. © 1988 Academic Press, Inc.

## 1. INTRODUCTION

Since the seminal work of McCulloch and Pitts (1943), many researchers have studied the computational power and learning abilities of feed-forward "neural" networks of linear threshold units. The proof that the perceptron learning algorithm was able to learn any linear separation from examples (Papert, 1961; Rosenblatt, 1962) generated considerable excitement. Gamba and collaborators experimented with feedforward networks having two layers of perceptrons (Gamba *et al.*, 1961; Borsellino and Gamba, 1961). An extensive rigorous literature on threshold logic, generated primarily in the mid sixties, will be reviewed in more detail shortly. Interest declined when the limitations of single layers nets were made explicit (Minsky and Papert, 1969), but was revived by the discovery of the Boltzman machine (Ackley *et al.*, 1985) and back propagation (Werbos, 1974; Parker, 1985; LeCun, 1985; Rumelhart *et al.*, 1986). These algorithms have been experimentally shown capable of learning some complicated tasks on multilayer networks. They have been tested on a variety of highly structured, toy problems such as recognizing symmetries (Rumelhart *et al.*, 1986) as well as on some (perhaps relatively simple) real world problems (Sejnowski and Rosenberg, 1987; Gorman and Sejnowski, 1987; Werbos and Titus, 1978). Little, however, is rigorously known about the capabilities of back propagation and the Boltzman machine as learning algorithms.

For a function to be learnable by a given net, of course, there must be some choice of weights for which the net realizes the function. This paper addresses the question: what is the smallest net which can realize an arbitrary function? That is, we are given a set $S$ of $N$ vectors in $d$ dimensions, and a function $F$ into the $e$-dimensional hypercube; i.e., $F: S \rightarrow \{1, -1\}^e$. When $e = 1$, an important special case, $F$ is called a dichotomy. (So, for example, $\mathbf{s} \in S$ might represent sensory data about some object, and the dichotomy $F(\mathbf{s})$ might be $+1$ if the object were a tree and $-1$ otherwise.) We desire to implement $F$ on a multilayer perceptron.

Multilayer perceptrons are defined as follows. They have an input layer of $d$ units. They then have one, two, or more successive layers of intermediate units, and a layer of $e$ output units.[1] Each unit's output will be connected to the input of each unit in the next layer, and a synaptic weight $w^l_{ij}$ will be associated with the connection of the output of the $j$th unit in layer $l$ to the input of the $i$th in layer $l + 1$ (see Fig. 1). Except for the input layer, all neurons are linear threshold units. That is, the output $v^{l+1}_i$ of the $i$th unit in the $(l + 1)$th layer is computed by $v^{l+1}_i = \theta(\Sigma_j w^{l+1}_{ij} v^l_j - t^{l+1}_i)$, where $v^l_j$ is the output of the $j$th unit in the $l$th layer, $t^{l+1}_i$ is the threshold of the $i$th unit in the $(l + 1)$th layer, and $\theta(x) = +1$ for $x \geq 0$ and $-1$ for $x < 0$. Each such unit is thus associated with a hyperplane (defined by $\{u_j : \Sigma_j w_{ij} u_j = t_i\}$) and gives output $+1$ if the input vector $\mathbf{v}$ is on the positive side of the hyperplane and $-1$ if $\mathbf{v}$ is on the other side.

In computation the input units are set equal to the values of the components of $\mathbf{s}$, for some $\mathbf{s} \in S$. The values of the units are computed layer by

[1] I will refer to a network with $l$ internal layers as an $l$-hidden-layer network, or sometimes simply an $l$ layer network. Thus a two layer network actually has an input layer and an output layer in addition to two intermediate or "hidden" layers. I will occasionally refer to the units as "neurons."

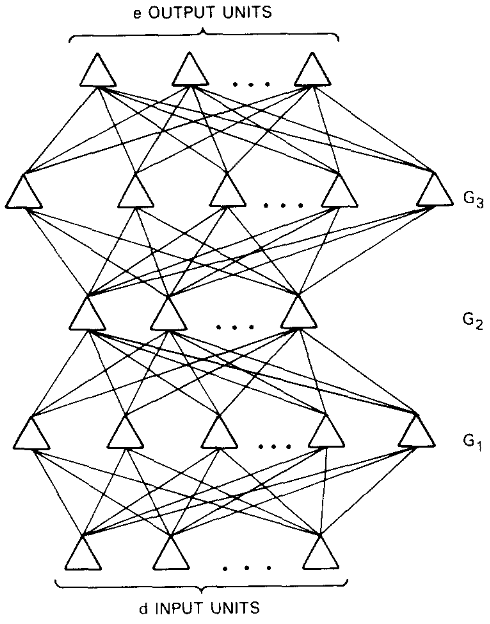

FIG. 1. A three-hidden-layer network.

layer until the output units are computed. The goal is that for each $\mathbf{s} \in S$, the value of the output units should equal $F(\mathbf{s})$. When this is true, the net *computes* or *realizes* the function $F$. A net is said to be *large enough* to compute a function $F$ if there is some choice of synaptic weight values for which the net computes $F$. Whether or not a net realizes $F(\mathbf{s})$, when a vector $\mathbf{s}$ is presented at the inputs, I call the vector of first hidden layer unit values, $\mathbf{v}^1$, the *first layer representation* of $\mathbf{s}$ (or sometimes the *first layer image*), and similarly for the $l$th layer. A set of $d$-dimensional vectors is said to be in *general position* if no subset of $d$ or fewer vectors are linearly dependent[2]; i.e., there are no "accidental" degeneracies.

There is an extensive literature on the capabilities of threshold logic, including both feedforward and feedback circuits. Recent intense interest in the storage capacity of threshold nets with feedback was stimulated by Hopfield's construction of a model for associative memory (Hopfield, 1982). (See, for example, McEliece *et al.* (1987) or Abu-Mostafa and St. Jacques (1985).) Recently Baum *et al.* (1986) have constructed feedforward nets with considerably better capacity and retrieval capabilities than known feedback designs for associative memory. The present paper

[2] Equivalently, no set of $d + 1$ points lie on a $(d - 1)$-dimensional hyperplane. The difference arises from the fact that, to consider "points" as vectors we implicitly subtract the origin.

studies the capacity of feedforward nets when no demands are imposed that the storage be associative. Small errors in the input to the nets described in this paper could lead to large variations in the output.

The literature on feedforward threshold logic includes, for example, Cover (1965), Nilsson (1965), Minsky and Papert (1969), Dertouzas (1965), Muroga (1979), Lewis and Coates (1967), and Hu (1965). Much of the early literature (but not all, see, e.g., Chap. 6 of Nilsson) focused on single threshold units. The input to such threshold units was frequently taken to be either **s** itself, or some vector $\phi(\mathbf{s})$ whose components $\phi_i$ are polynomials of the components $s_i$ (e.g., Cover, 1965; Nilsson, 1965). What I call one-hidden-layer nets can be considered nets of this type where, however, the $\phi_i$ are themselves linear threshold functions. Many of the results in the present paper come from varying the input weights to the first hidden layer. In contrast, many previous results considered the $\phi$ as fixed and varied only the output weights. The back propagation learning algorithm uses nets very similar to multilayer perceptrons (Rumelhart *et al.*, 1986; Werbos, 1974; LeCun, 1985; Parker, 1985).

Cover (1965) gave a count of the number of dichotomies implementable by a single threshold unit. This result will be described and applied in Section 3. Cover also showed that a threshold unit with $M$ weights has probability $\frac{1}{2}$ of implementing a random dichotomy on $2(M + 1)$ random vectors, and that for large $M$, it has probability rapidly approaching 1 of implementing a random dichotomy on fewer vectors and probability approaching 0 on more vectors. The extension of this result to nets with hidden layers is an interesting open question. (Lower bounds on the size of multilayer nets having probability $\frac{1}{2}$ of implementing a random dichotomy can be trivially obtained from the results in Section 3 of this paper. I do not, however, know of any sharp upper bounds.)

Nilsson (1965) showed that a multilayer perceptron with only one hidden layer containing $N - 1$ nodes was capable of computing an arbitrary dichotomy. My main result, presented in Section 2, improves this by constructing a net with one hidden layer containing[3] $\lceil N/d \rceil$ units capable of realizing an arbitrary dichotomy on an arbitrary set of $N$ points in general position in $\Re^d$. I also give examples of dichotomies which cannot be realized on any multilayer perceptron with fewer units. In this sense, the net given is optimal.

Section 4 constructs one-hidden-layer nets capable of realizing arbitrary functions into the $e$ hypercube. These nets are of nearly minimal size, as is seen from counting arguments of Section 3.

The counts of Section 3 are based on Cover's result and yield lower bounds on the size of multilayer networks capable of computating arbi-

[3] The notation $\lceil x \rceil$ is used for the smallest integer greater than or equal to $x$. $\lfloor x \rfloor$ denotes the greatest integer less than or equal to $x$.

trary or random functions. Lower bounds can be obtained both for networks with real valued synaptic weights and for networks where the weights are limited to only a finite number $n_g$ of possible values. Such a limited gray scale seems necessary in practice. The bounds for nets with limited gray scale and for real valued weights differ only by logarithmic factors. The construction of Theorem 3 in Section 4 shows that such logarithmic gains can be achieved.

The other side of achieving such logarithmic gains over fixed gray scale nets, of course, is that finely tuned weights must be used. To achieve the near optimal performance of Theorem 3 at least requires using a gray scale of order $\log(N)$. Similarly the construction of Theorem 1 could require finely tuned weights. This consideration might limit the application of the construction in some practical cases. Finding storage algorithms which attempt to minimize the gray scale for a given, near optimal capacity is an interesting open question which I have not considered.

The counts of Section 3 do allow substantial gains on the number of neurons (although not the number of weights) necessary for implementing arbitrary and random functions by using nets with two hidden layers provided topological restrictions are imposed on the set $S$ to avoid the obstructions described in Lemma 1. Potentially interesting restrictions are discussed in Section 5. As yet, however, I have not been able to prove these restrictions sufficient to allow small two-hidden-layer nets to compute arbitrary functions of the restricted vector set.

It is important to emphasize that this paper studies the size of nets capable of *realizing arbitrary* or *random* functions. Fast algorithms are presented which realize such functions on near minimal nets. The question of how to *learn* a *structured* function from examples is, however, only tangentially considered.

Abu-Mostafa has recently remarked that the problem of pattern recognition and other problems typically solved far better by people than by machines may be of a very unstructured nature[4] (Abu-Mostafa, 1987a, 1987b; Abu-Mostafa and Psaltis, 1987). To recognize a tree, for instance, may require storing a huge amount of information of different cases of trees and different attributes of trees. Possibly in many practical problems after a few symmetries are factored out, the remaining information will have no low complexity description, i.e., will be essentially random. It is also perhaps worth remembering that an important feature of Samuel's celebrated checker program was its unstructured data blank of previously encountered positions (Samuel, 1959, 1967). Abu-Mostafa (1987a, 1987b) has further emphasized that neural networks may be particularly effective for computing and learning random problems.

[4] My discussion of this point should not be construed as endorsing this pessimistic position. In fact, I maintain an open mind.

Many studies of learning algorithms, on the other hand, have emphasized highly structured problems which allow generalization (Rumelhart *et al.*, 1986). This implies that the map $F(S)$ has a low complexity description and may be implementable using a net much smaller than the lower bounds given by the counting arguments of Section 3 (or the topological bound of Section 2). The output of such a net applied to an input vector not in the set $S$ may provide a useful "generalization" of the map $F$. The back propagation and Boltzman machine algorithms for learning require the user to supply a net capable of capturing the function to be realized. In Section 3, I comment on appropriate size and shape nets for implementing and learning structured, nonrandom data. See also the final paragraph of Section 6.

## 2. Implementing Dichotomies of Real Values Vectors

This section concerns layered networks capable of implementing arbitrary dichotomies of a set $S$ of $N$ points in general position in $E^d$, Euclidean $d$-dimensional space. The case $N \gg d$ is of particular interest.

Lemma 1. *Any net capable of arbitrary dichotomies must have at least $N/d$ units in its first hidden layer.*

*Proof.* To realize a particular dichotomy $F$, the net must at least map each point in the set $\{S^+\} = \{s \in S : F(s) = +1\}$ to a first layer representation different from the first layer representation of any point in the set $\{S^-\} = \{s \in S : F(s) = -1\}$. Two points are mapped to different representations if and only if they are separated by at least one hyperplane associated with one of the hidden layer neurons. Consider the line segments connecting each point in $\{S^+\}$ to its nearest neighbor in $\{S^-\}$ and vice versa. There are up to $N$ such line segments (not $N/2$ since the nearest neighbor of the nearest neighbor of a point $s_1$ will generally be a third, different point). A hyperplane exists in $d$ dimensions which cuts any $d$ line segments, but in general no hyperplane can be constructed which cuts more than $d$ specific segments. Thus in general it will require at least $N/d$ hyperplanes to separate $S^+$ from $S^-$.

A two-dimensional example of this obstruction is shown in Fig. 2.

More generally, to construct an example in $d$ dimensions, consider the set of points to lie along the smooth curve parametrized by $(t, \exp(t), \exp(2t), \ldots, \exp((d-1)t))$ for $0 \leq t \leq 1/d$. Let the points in $S^+$ lie at $\{t = (2i-1)/Nd : i = 1, N/2\}$, and let the points in $S^-$ lie at $\{t = (2i)/Nd : i = 1, N/2\}$. This set of points is in general position. Any hyperplane can cut only $d$ of the links along the curve. Thus fewer than $N/d$ hyperplanes cannot separate this dichotomy. Q.E.D.

This lemma indicates that it may be fruitful to impose stronger condi-

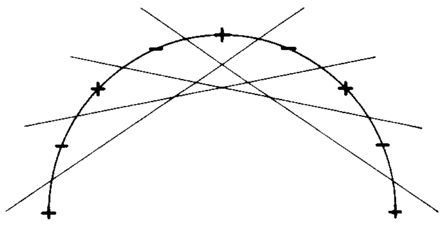

FIG. 2. A $d = 2$ example of a dichotomy which cannot be realized without $N/d$ units in the first hidden layer. The points lie on a circle and alternate between positive (represented by $+$) and negative (represented by $-$) in the dichotomy. No line can cut the circle more than twice so $N/2$ lines are necessary to separate points in $\{S^+\}$ from points in $\{S^-\}$.

tions than "general position" on the placement of the vectors in $S$. Such conditions are discussed in Section 5. If we impose topological conditions which avoid the lower bound of Lemma 1 we may be able to compute arbitrary dichotomies with far fewer neurons. Lower bounds arising from counting arguments, presented in the next section, would still be valid. I will now show that the lower bound of Lemma 1 can in fact already be achieved by a one-hidden-layer network.

THEOREM 1. *A one-hidden-layer net with $\lceil N/d \rceil$ internal units can compute an arbitrary dichotomy on N d-dimensional vectors in general position.*

*Proof.* Let $N^+$ be the size of $S^+$ and $N^-$ the size of $S^-$, and assume without loss of generality that $N^- \geq N/2 \geq N^+$. Construct $\lceil N^+/d \rceil$ hyperplanes each containing $d$ points in $S^+$, not contained by any of the other hyperplanes, and no points in $S^-$. (This is easy as I can choose a hyperplane which contains any $d$ points. Since the points in $S$ are in general position, such a hyperplane does not contain $d + 1$ points in $S$.) Shifting each plane infinitesimally parallel to itself construct $\lceil N^+/d \rceil$ pairs of planes such that no points in $S^-$ lie in the slice between the two hyperplanes in a pair and such that every point in $S^+$ is contained in such a slice. A two-dimensional example of the slicing produced is shown in Fig. 3.

Each such slice is characterized by two units $v_1$ and $v_2$ such that any point in the slice gives $v_1 = +1$, $v_2 = -1$, and any point not in the slice gives $v_1 = v_2$. Thus by using $2\lceil N^+/d \rceil \leq \lceil N/d \rceil$ neurons with connections to the output $+1$ from unit 1 and $-1$ from unit 2 for each pair, the output unit gets an input of 0 for any point in $S^-$ and 2 for any point in $S^+$. Thresholding at 1 implements the dichotomy. Q.E.D.

Notice that I am able to construct the weights to the output unit independent of the particular dichotomy. Only the input weights are adjusted to implement a given dichotomy.

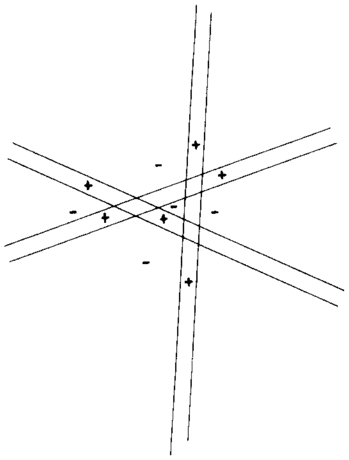

FIG. 3. An example of the slicing used in Theorem 1. Thin slices contain $d$ (in this case $d = 2$) positive points (represented by $+$) and no negative points (represented by $-$).

I have been able to implement the minimum possible net using only one hidden layer. Thus there is no need in this case to consider nets with more than one layer.

## 3. LOWER BOUNDS FROM COUNTING ARGUMENTS

This section derives lower bounds on the size of multilayer networks from simple counting arguments. I compute an upper bound on the number of distinct networks of a given size, and compare this to the total number of functions into the $e$ hypercube, $2^{Ne}$. The bounds so generated will be slightly weaker than the bound of Lemma 1; however, they will be important in the next section when maps into the $e$-dimensional hypercube are considered and particularly in Section 5 when I consider topological conditions which avoid the bound of Lemma 1. These arguments also give lower bounds on the size of nets able to implement (with high probability) random as opposed to arbitrary functions. I also remark on the appropriate shape of nets for implementing highly structured, nonrandom mappings, and on the power of real versus binary valued synapses.

Synapses whether electronic or biological are generally restricted by practical limitations on precision and range to some small number of gray levels, say $n_g$. If such a restriction on the weights $w^l_{ij}$ is assumed, it is immediate that $N_C$, the number of connections in the network, must be greater than $Ne/\log_2(n_g)$ for a net sufficiently large to implement any function, and $N_C > (Ne + \log_2(\alpha))/\log_2(n_g) \sim Ne/\log_2(n_g)$ for any net large enough to implement a fraction $\alpha$ of the possible functions. This of course holds not only for multilayer perceptrons but for nets of any fixed topology, including nets which allow feedback.

I will extend this counting argument to the case of real valued weights by using a function counting theorem of Cover (1965), and will find that in this case $N_C \gtrsim Ne/\log_2(N)$. These counting arguments can yield only lower bounds on the size of the networks required, as it might be that many of the functions are redundant. Simple nets without hidden layers but with real valued weights can in fact implement more dichotomies by a factor of $O(\log_2(N))$ than nets with binary valued weights. The next section describes a net with one hidden layer which also performs better by a factor of $\log_2(N)$ than a similar net with limited gray scale synapses could. Thus it is sometimes possible to realize a gain by a factor of $\log_2(N)$ if real valued synapses are employed. It has previously been shown (McEliece *et al.*, 1987), however, that the extension from binary to real valued weights only changes the capacity of the Hopfield Model by a factor of $\pi/2$.

THEOREM 2. (Cover, 1965). *There are $C(N, d)$ homogeneously linearly separable dichotomies of N points in general position in Euclidean d space, where*

$$C(N, d) = 2\sum_{k=0}^{d-1} \binom{N-1}{k}.$$

A dichotomy is *homogeneously* linearly separable if it is separable by a hyperplane passing through the origin. The linear separations of $N$ points in $d$ dimensions become homogeneously linearly separable in $d + 1$ dimensions. The number of linear separations of $N$ points is less than or equal to the number of linear separations of $N$ points in general position. Thus the number of linear separations in $d$ dimensions is less than the number of homogeneous linear separations in $(d + 1)$ dimensions. I thus easily obtain the following

COROLLARY 1. *The number of linearly separable dichotomies of N points in d dimensions, for $N \geq 3d$, is less than $4N^d/d!$.*

The number of maps of $N$, $G_1$-dimensional vectors into the $G_2$-dimensional hypercube which are implementable by linear threshold functions is, by definition, equal to the $G_2$ power of the number of linearly separable dichotomies of $N$ points in $G_1$ dimensions. By the corollary this is less than or equal to $(4N^{G_1}/G_1!)^{G_2}$. Counting layer by layer gives a bound on the number of mappings computable by a net with $d$-dimensional inputs, $e$-dimensional outputs, and $k$ layers with $G_i$ units in the $i$th hidden layer of

$$\frac{(4^{\Sigma_{i=1,k}G_i})N^{N_C}}{(d!)^{G_1}(\Pi_{i=1,k-1}(G_i!)^{G_{i+1}+1})(G_k!)^{e+1}},$$

where $N_C = dG_1 + G_1G_2 + \cdots + G_ke$. (A factor of $\Pi_{i=1,k}G_i!$ in the denominator came from the observation that relabeling the internal units within a row does not change the representations realizable by the next row.) Comparing this bound to $2^{Ne}$, the number of mappings of $N$ $d$-dimensional vectors into the $e$ hypercube, I establish

PROPOSITION 1. *No feedforward net of the type considered can compute an arbitrary map from N d-dimensional vectors into the e hypercube unless it has* $N_C \geq Ne/\log_2(N)$.

COROLLARY 2. *A one hidden layer net can never compute arbitrary functions on any set* $\{S\}$ *of N vectors in d dimensions unless it has at least* $G = Ne/((e + d)\log_2(N))$ *internal units.*

COROLLARY 3. *No multilayer perceptron can ever compute an arbitrary function, no matter how many levels it has, unless it has* $O(\sqrt{Ne/\log_2(N)})$ *units.*

This implies that under appropriate topological conditions on $S$ to avoid the obstruction which led to Lemma 1, it might be possible for a two layer net with only $O(\sqrt{N/\log_2(N)})$ units to realize an arbitrary function on $N$ vectors. I will discuss some possible topological conditions and the possibility of constructing small two level nets to compute arbitrary functions in Section 5. It is worth remarking, however, that were one able to compute an arbitrary function using a two layer net with only $O(\sqrt{N})$ units, this corollary would then imply that there was little interest in considering nets with more layers for the purpose of realizing random or arbitrary functions.

What is the geometry of a minimal net for a given highly structured problem? (We might reasonably define a function as being "highly structured" if it can be realized by a net much smaller than that necessary to realize a random function.) Back propagation and the Boltzman machine do not attempt to answer this question. Rather a human supplies a geometry and these algorithms simply vary the weights in a (perhaps vain) attempt to solve the problem. My arguments give estimates on the fraction of functions realizable by a given size net. It is entirely possible that a two-hidden-layer feedforward net able to compute $N_A$ different functions, and a three layer net also able to compute $N_A$ functions are in fact able to compute very different sets of functions. Characterizing these sets is an interesting question. With no notion of how to answer this question, one finds the two layer net as likely able to compute the given function as the three layer net.

Usually in applying learning algorithms, one is interested in "generalizing" from the training set $S$ to apply the net generated to new input vectors. Generalization in this context implies that the map $F(S)$ has a low complexity description and may thus be realizable using a net much smaller than the lower bounds given by the counting arguments of this section or the topological bound of Section 2. Some importance may reasonably be attached to the output of such a net when applied to an input vector not in the set $S$. Indeed a heuristic definition of the "best generalization" of the given function $F$ to inputs not in $S$ is given by the shortest algorithm which represents $F$ on $S$. The question of whether a given algorithm is the shortest representing a given function is in principle undecidable (Kholmogorov, 1965; Chaitin, 1974), so that one typically chooses some heuristic for finding a relatively terse algorithmic description.

The appropriate shape net for *learning* from examples may depend on the learning algorithm and might be rather different than the smallest net capable of representing a given set of examples. If we succeed in implementing the examples with a net much smaller than the lower bounds of Proposition 1 and the corollaries, we can hope to have achieved a useful generalization. On the other hand, larger nets with other bottlenecks to information flow can also give interesting generalizations. Such a net might for instance have a very large first hidden layer, a very very narrow second hidden layer, and a very large third hidden layer. The purpose of the large first and third layers might be to allow a lot of freedom for learning, or for representing complicated mappings, while the second bottleneck layer, with $G_2$ units, forced the net to classify inputs into no more than $2^{G_2}$ different patterns, so that any new, untrained stimulus will be classified into one of these categories. Back propagation on such a net might be a reasonable heuristic for generalization. The counting arguments presented do not yield the limitations restricting such a net.

In constructing multilayer networks that solve problems many authors, and particularly Hinton and Sejnowski (1986), have stressed the importance of finding good internal representations of the data. In the example discussed above the representation at the bottleneck layer might have to be particularly well chosen. The necessity of finding "good" internal representations, sometimes called the "credit assignment problem" has been said to separate "hard" from "easy" learning tasks.

For the cases considered in the previous section and the next section, one may fix the connections on one level and simply adjust the connections on the other level. No gain is possible by using an algorithm able to adjust the weights on both levels simultaneously. This might well extend to the cases considered in Section 5, where the possibility is discussed of imposing topological conditions on $S$ which allow very small multilayer nets to represent arbitrary functions. In a two-hidden-layer network, with $G_1 \sim G_2 \sim \sqrt{N} \gg d$, essentially all the connections lie in the intermediate layer of synapses. It is reasonable to hope that it might suffice to vary these in solving a random problem. For a crafted problem, where the set $S$ has a description of low algorithmic complexity, it may be possible to find an internal representation of the data which allows $G_1$ say to be of order $\log(N)$, or smaller. Such a description would require tuning the weights on several levels. By definition neither such a description nor a sensible notion of "generalization" exists for random problems.

## 4. General Functions of Real Valued Vectors

This section extends the results of Section 2 to arbitrary mappings of $N$ vectors in general position in $d$-dimensional space into the $e$-dimensional hypercube (i.e., the set of all $e$-dimensional vectors all of whose components are either $+1$ or $-1$). Lemma 1 of course applies so that at least $N/d$ units are necessary in the first layer. Only one-hidden-layer networks are considered in this section, as I do not know how to improve on these for this problem. I will, however, give constructions which are close to the bounds imposed by Corollary 2.

Proposition 2. *A one-hidden-layer net with N internal units can represent an arbitrary mapping into the e hypercube.*

*Proof.* One may map $N$ $d$-dimensional vectors into $N$ $N$-dimensional vectors in general position by simply slicing the space with hyperplanes orthogonal to any axis, say the $\hat{z}$ axis, so that the $d$-dimensional vector with the highest $z$ component is mapped to the $N$ vector $(+1, +1, \ldots, +1)$, the $d$ vector with the next highest $z$ component is mapped to $(+1, +1, \ldots, +1, -1)$, and so on to the last vector which is mapped to the $N$ vector all of whose components are $-1$. (If two $d$-vectors have the same $z$ component, one will need to infinitesimally rotate the planes to separate them.)

A separating hyperplane can implement an arbitrary dichotomy on $N$ vectors in general position in $N$-dimensional space. The mapping into the $e$ hypercube is the same as realizing $e$ dichotomies, and can thus be realized by this net. Q.E.D.

Note that this net can be learned by first constructing the first layer as outlined in the proof and then learning the second layer of synapses by the perceptron learning algorithm (Papert, 1961; Rosenblatt, 1962).

PROPOSITION 3. *A one-hidden-layer net with $2\lfloor e/2\rfloor\lceil(3N)/(4d) + 3\rceil + 2(e/2 - \lfloor e/2\rfloor)\lceil N/d\rceil$ internal units can represent an arbitrary mapping into the $e$ hypercube.*

*Proof.* It is apparent that a one-layer net with $e\lceil N/d\rceil$ internal units can perform an arbitrary mapping by using $e$ copies of the net of Theorem 1. To gain a factor of $\frac{3}{4}$ I will have to compress this net. I will consider first the case of $e = 2$ and realize this with $2\lceil(3N)/(4d) + 3\rceil$ units. The proposition will then follow for general $e$ by using $[e/2]$ copies of this net, plus one net like that of Theorem 1 if $e$ is odd.

Let $\{S_{++}\}$ be the set of vectors mapping into $(+1, +1)$ and call the size of this set $N_{++}$, and respectively for $\{S_{+-}\}$, $N_{+-}$, $\{S_{-+}\}$, etc. Recall from the proof of Theorem 1 that I may pass a plane through *any d arbitrary* points in $S$, and no other points in $S$, and then, by shifting this plane slightly parallel to itself find two units representing a slice containing only these vectors. Thus let the first two units represent a slice containing only $d$ vectors all in set $\{S_{++}\}$. That is, units 1 and 2 will have the same value for all input vectors except for $d$ vectors in $\{S_{++}\}$, when unit 1 will have value $+1$ and unit 2 will have value $-1$. The weights on the synapses to both output units from intermediate unit 1 will be taken to be $+1$ and from intermediate unit 2 will be $-1$. These two units act in consort to give total input into the output units of 0 if the input vector is not within the slice they recognize and $+2$ if it is. Continuing in this fashion I may recognize the entire set $\{S_{++}\}$ with $2\lceil N_{++}/d\rceil$ units. Similarly a pair of units can recognize $d$ vectors in the set $\{S_{+-}\}$ and collectively provide input $+2$ to output unit 1 and $-2$ to output unit 2 if the vector is one of these $d$ vectors and 0 to both output units otherwise. In this way, using $2(\lceil N_{++}/d\rceil + \lceil N_{+-}/d\rceil + \lceil N_{-+}/d\rceil)$ units I may provide input greater than 1 to the output units whenever they should turn on and less than or equal to zero otherwise. Biasing at 1 thus realizes the map.

This net has realized the map with about $(\frac{3}{4})N$ pairs of units. To get an exact upper bound, realize that we could instead have used pairs of units recognizing any three of the four sets $\{S_{++}\}$, $\{S_{-+}\}$, $\{S_{+-}\}$, $\{S_{--}\}$. With appropriate output weights from the pairs of units recognizing each set and appropriate output unit biases, the mapping can thus be realized with $2\lceil N/d + 3\rceil - \lceil\max(N_{++}, N_{+-}, N_{-+}, N_{--})/d\rceil \leq 2\lceil(3N)/(4d) + 3\rceil$ units, and the proposition follows. Q.E.D.

THEOREM 3. *A one-hidden-layer net with $G = \lfloor 4N/d\rfloor\lceil e/\lfloor\log_2(N/d)\rfloor\rceil$ intermediate units is capable of computing an arbitrary mapping.*

*Proof.* I will consider the output bits $b$ at a time (rather than 2 at a time as done in the proof of Proposition 3) and choose $b$ to minimize the total number of internal units. Consider first the case where $e > \log_2(N/d)$. I will use pairs of units which recognize up to $d$ vectors in one of the $2^b$ sets

$\{S_{+++\cdots+}\}$, $\{S_{-++\cdots+}\}$, . . . , $\{S_{---\cdots-}\}$ (where there are $b$ indices on each $S$). If there were no more than $d$ vectors in each one of these sets, I could thus implement the entire mapping into these $b$ bits with $2^{(b+1)}$ intermediate units. The optimal case would in fact be if there were exactly $d$ vectors in each of these sets, when I would have $2^b d = N$. If I choose $b = \lfloor \log_2(N/d) \rfloor$, however, some of the sets may have more than $d$ vectors and one must add extra pairs of units for the overflow. In the worst case, fewer than $2^b$ of the pairs of units recognize exactly one vector each, in which case the entire set of $N$ vectors is recognized with less than $2^{(b+2)} \leq \lfloor 4(N/d) \rfloor$ intermediate units.

To represent all $e$ output bits I need no more than $\lceil e/b \rceil$ such nets in parallel, so that I obtain a bound of $\lfloor 4N/d \rfloor \lceil e/\lfloor \log_2(N/d) \rfloor \rceil$ internal units. When $e \leq \log_2(N/d)$, I will simply let $b = e$, and realize the mapping with $\lfloor 4N/d \rfloor$ internal units. Q.E.D.

Notice that for any fixed number of gray levels $n_g$, and for sufficiently large $N$, this realizes an arbitrary mapping with fewer than $Ne/((e + d)\log_2(n_g))$ units (the lower bound with gray level $n_g$), so that one can indeed gain by utilizing real valued synapses in a multilayer net. For $N \gg d \gg e > \log_2(N)$, I have achieved, up to a factor of 4, the minimum net allowed by Corollary 2.

## 5. Multilayer Networks

The result of Section 2 was disappointing in that we were prevented by a topological obstruction from realizing arbitrary functions on nets as small as allowed by the counting arguments of Section 3. For one-hidden-layer nets the constructions of Theorems 1 and 3 are close to the best allowed by counting arguments. For example, Theorem 1 uses $G_1 = N/d$ hidden units to compute a dichotomy, while the counting arguments allow nets with only $G_1 = N/(d + 1)\log_2(N)$. However, if we impose conditions on $S$ which avoid the obstruction, two-hidden-layer nets might in principle get by with only $O(\sqrt{N/(d + 1)\log_2(N)})$ hidden units. Clearly we are interested in asking for natural conditions on $S$ which avoid this obstruction, and then in asking whether two layer nets are able to compute arbitrary functions with $O(\sqrt{N})$ units, or if not for a $k$ such that $k$-layer nets with $O(\sqrt{N})$ units suffice.

Another motivation for answering this question is to get a more accurate count of the number of distinct functions computable by nets of a given size and depth. The counting arguments of Section 3 give upper bounds. When fewer functions are computable, e.g., when the count says that $2^{Ne}$ functions are computable but in fact there are functions which are not computable, this can only be because of duplication; i.e., some functions are computed by several nets. A better count would improve estimates of the reliability of generalization by a given net and give better heuristics for deciding what net to use in order to learn a certain function from examples.

A necessary condition to avoid the limitation of Lemma 1 and realize a dichotomy $F$ with only $M < N/d$ units in the first hidden layer is that the set $S^+$ be separable from the set $S^-$ by use of only $M$ hyperplanes. Here by *separated* I merely mean that the first layer image of every vector in $S^+$ be distinct from the first layer image of every vector in $S^-$. Similarly, to realize a function $F$ we need separation in the sense that every two input points whose image under $F$ is different must have different first layer images.

Although general position is not a strong enough condition to allow separability, separability will occur in several important cases. If the points in $S$ are chosen randomly and uniformly in some compact region in $\Re^d$, say the unit cube, then they will almost always be separable by $O(dN^{2/d})$ planes. Indeed using $M$ planes to form a regular grid we can chop the unit cube into $(M/d)^d$ regions, so that the probability of having two points in the same region will be about $\binom{N}{2}/(M/d)^d$, and will be small for $M \gg dN^{2/d}$.

Another very important case is when the vectors $s \in S$ are binary, that is they lie on the $d$-dimensional hypercube. The corners of the $d$ hypercube can be separated using the $d$ threshold units which map each corner into its coordinate representation, i.e., by the $d$ hyperplanes through the origin orthogonal to the $d$ coordinate axes. For the rest of this section I will discuss the case where the $N$ points in the set $S$ are constrained to lie on the corners of the $(1, -1)$ hypercube.

Of course the set $S$ cannot now be assumed to be in general position. Only a small set of points can exist in general position on the hypercube. Indeed no set of $2d$ points in general position exist on the $d$ hypercube, since either at least $d$ of them must lie on the $d - 1$-dimensional subcube with last coordinate 1 or at least $d$ of them must lie on the subcube with last coordinate $-1$. Thus Theorem 1 does not apply in this case, and the smallest 1 level net that I am sure can compute an arbitrary dichotomy on $N$ points on the $d$ hypercube contains $N$ units in its internal level. The proof of Proposition 3 extends to this case—we can always map the points on the $d$ hypercube into $N$ points in general position on the $N$ hypercube and then can compute an arbitrary map into the $e$ hypercube with a one-hidden-layer net.

One would now like to construct a two layer net which computes an arbitrary dichotomy of $N$ vectors on the $d$ hypercube using only $O(\sqrt{N})$ units. One approach to this problem is to construct a first layer mapping $N$

vectors on the $d$ cube into $N$ vectors on a cube of dimension $O(\sqrt{N})$ for which a weakened version of "general position" holds. Such a "weakened version" must be weak enough to be obtainable, but strong enough to allow the technique of Theorem 1 to be applied to generate the second layer. If it were possible to give a first layer which maps $N$ arbitrary vectors on the $d$ hypercube into the $\alpha\sqrt{N}$ hypercube so that no $\beta\sqrt{N}$ image vectors are linearly dependent, for some numbers $\alpha$ and $\beta$, then the method of Theorem 1 would allow an arbitrary dichotomy to be computed by a net with $\alpha\sqrt{N}$ units in its first layer and $\sqrt{N}/\beta$ units in its second layer. For $\alpha > \beta$ this hypothesis seems plausible. Indeed as is shown in Appendix A, almost every set of $N$ randomly chosen vectors on the $\sqrt{N}$ hypercube will not have any linearly dependent subset of size $\sqrt{N}/3$. It remains an open question, however, whether it is always possible to generate a set without linearly dependent subsets by using linear threshold functions.

A weaker hypothesis is that for any two sets of vectors $S^+$ and $S^-$ on the $d$ hypercube, there exist a first layer mapping $S = S^+ \cup S^-$ into the $\alpha\sqrt{N}$ hypercube such that the images of the points in $S^+$ can be partitioned into no more than $\beta\sqrt{N}$ subsets having the property that the space spanned by the points in any subset does not contain any point in the image of $S^-$. Manifestly this result would allow one to find hyperplanes intersecting either $\alpha\sqrt{N}$ points in the image of $S^+$, or all the points in one subset, but none of the points in the image of $S^-$, so that again the method of Theorem 1 would construct a two layer network computing an arbitrary dichotomy. (Notice, however, that with this weaker hypothesis one could not use the compression scheme of Theorem 3 to compress maps into the $e$-dimensional hypercube.) Again, however, I have been unable to establish this plausible hypothesis.

Before beginning this investigation I mistakenly believed that if you chose a set of $N$ vectors on the $d$ hypercube by choosing each component randomly and independently with probability $\frac{1}{2}$ of being $+1$ and probability $\frac{1}{2}$ of being $-1$, then a similarly randomly chosen set of $\sqrt{N}$ hyperplanes through the origin would map these vectors to uncorrellated image vectors. Straightforward calculation of the variance shows this is not true. I remark on this because in a previous preprint (Baum *et al.*, 1986) we heuristically hypothesized that random feature detectors of this type might be able to increase the capacities of associative memories. I do not now see how to make such a scheme work. Rosenblatt (1962) and other early investigators also favored randomly chosen input feature detectors. Closer analysis may well indicate, however, that random feature detectors are not very efficient. Randomly chosen feature detectors will only rarely separate highly correlated input vectors and in a large set of even randomly chosen vectors there will be many pairs which are highly correlated.

In the face of my inability thus far to construct multilayer nets using simple threshold units, I have been driven to consider multilayer nets composed of more complicated objects: periodic threshold units. So far as I am aware, these are not utilized by the nervous system, but they are easily implementable in hardware and may be quite powerful computationally so there is some motivation for considering them. A periodic threshold unit takes value $v_i = \theta(\sin(u_i p_i/2\pi + t_i))$, where $u_i$ is the input $u_i = \sum_j w_{ij} v_j$ as for a normal threshold unit, $p_i$ is the period, and $t_i$ is a phase. I have previously remarked on the utility of using periodic activation functions in back propagation (Baum, 1986). See also (Lapedes and Farber, 1987). If the weights are restricted to be 1 or 0, the period is 2, and the input vectors $S$ lie on the hypercube, I call the units mod$_2$ *units*. The mod$_2$ units could be built out of $\log_2(N)$ simple threshold units. Multilayer nets of periodic threshold units will be discussed in a separate publication (E. B. Baum and L. E. Baum, in preparation).

## 6. Summary

The Boltzman machine (Ackley *et al.*, 1985) and the Hopfield model of associative memory (Hopfield, 1982) have stimulated a resurgence in interest in circuits of threshold units. These models use circuits with feedback. More recently, however, the back propagation algorithm (Werbos, 1974; Parker, 1985; LeCun, 1985; Rumelhart *et al.*, 1986), which uses a circuit much like a layered perceptron, has aroused more interest than the Boltzman machine and feedforward networks have been proposed (Baum *et al.*, 1986) which seem superior to known feedback networks for associative memory. This strongly motivates elucidation of the capabilities of multilayer perceptrons.

This paper has presented results on the size of multilayer perceptrons necessary to compute arbitrary functions into the $e$ hypercube on given sets of $N$ $d$-dimensional vectors. Previously it was known that an arbitrary dichotomy (i.e., $e = 1$) could be realized by a net with one hidden layer containing $N - 1$ units. I have constructed a one-hidden-layer net with $\lceil N/d \rceil$ units in the hidden layer that implements an arbitrary dichotomy on vectors in general position. Furthermore, this construction was shown to be minimal by giving examples of dichotomies which cannot be computed by any multilayer perceptron with fewer units or weights.

Constructions which implement arbitrary functions were also given. For example, a net was constructed with one hidden layer containing $\lfloor 4N/d \rfloor \lceil e/\lfloor \log_2(N/d) \rfloor \rceil$ units that implements an arbitrary function. This net was shown to be near minimal by counting arguments.

Indeed, by applying a function counting theorem of Cover (1965), it was shown that no feedforward net can compute an arbitrary function unless it has at least $Ne/\log_2(N)$ weights. Simple counts also showed that, if the number of values allowed each weight is restricted to $n_g$, the net requires at least $Ne/\log_2(n_g)$ weights. Thus the possible gain from using real valued weights seems to be $O(\log_2(N))$. The net described above realizes this gain.

This degree of optimization, on the other hand, necessarily requires using at least $O(\log_2(N))$ possible weight values. For some sets $S$ of input vectors, the optimized algorithm for implementing dichotomies will also require finely tuned weights. This consideration may limit the usefulness of these algorithms for practical circuits. Similarly, I have not constrained the algorithms to be robust to errors in the input vectors, nor have I considered algorithms useful for generalization in that they implement a given function on a circuit which is near minimal for that particular function. The counts do give bounds on the number of functions implementable by a circuit of any particular size and shape. This may be useful in estimating whether a particular circuit is likely to be able to implement a particular function.

The bounds obtained by counting arguments show merely that the number $N_C$ of weights must be sufficiently large to implement arbitrary functions. As $N_C$ weights can be obtained with one hidden layer of size $O(N_C)$ or two hidden layers of size $O(\sqrt{N_C})$, it is of great interest to ask which functions can be implemented by multilayer circuits with only $O(\sqrt{N})$ units. For arbitrary sets of functions in general position, it was shown that the first hidden layer itself needs $N/d$ units so that no gain is possible by using multilayer circuits. However, if the vectors are constrained to lie on the $d$ hypercube the obstructions which lead to this result vanish. Alternatively these obstructions will almost never be present if the input vectors are chosen randomly and uniformly over some region in Euclidean space. It is an interesting open question in both these cases to construct multilayer nets with $O(\sqrt{N})$ neurons which realize random or arbitrary dichotomies. It was remarked that if one could give a first layer mapping any set of $N$ vectors on the $d$ hypercube (or alternatively randomly chosen in a bounded region) into the $\alpha\sqrt{N}$ hypercube so that no $\beta\sqrt{N}$ image vectors are linearly dependent, for some numbers $\alpha$ and $\beta$, then such multilayer nets could be constructed. Appendix A shows that such sets of vectors exist, but I do not know how to construct such a mapping using threshold functions. Note that if arbitrary functions can be accomplished by a two layer net of size $O(\sqrt{N})$, the counting bounds will allow no substantial improvement by using more than two hidden layers.

If a set of functions $F$ includes all possible dichotomies on a set $S$ of points, then $S$ is said to be *shattered* by F. The *Vapnik–Chernovenkis dimension* of $F$ is defined as the smallest $N$ such that no set $S$ of size $N + 1$

is shattered by $F$. Blumer *et al.* (1986) have proved remarkable theorems which classify learnable concepts with the Vapnik–Chernovenkis (V–C) dimension and rigorize Occam's Razor.[5] Indeed, they show that if a set of positive and negative examples of a Boolean function, chosen under an arbitrary but fixed probability distribution over the set of examples, can be captured by a simple enough formula, then a good generalization to future examples produced in the same way will almost certainly be obtained. Here the formula used to represent the examples is taken from some set $H$ of hypothesis functions, and the "simplicity" of the formula is characterized by the V–C dimension of $H$. My Theorem 1 shows that the V–C dimension of a class of $N_C$ connection multilayer perceptrons is at least $N_C$ and Proposition 1 shows that the V–C dimension is no greater than $N_C \log_2(N_C)$. Using these results, Baum and Haussler (in preparation) will characterize how small a net should be used in learning a given set of examples, if valid generalization is desired. This line of argument will, for example, bear directly on the appropriate net to use when applying back propagation.

## Appendix A

In this appendix I present a proof due to H. Rumsey (1988, personal communication) of:

PROPOSITION 4. *For large $N$, almost every set of $N$ vectors on the hypercube of dimension $\sqrt{N}$ has no linearly dependent subset of $\sqrt{N}/3$ vectors.*

*Comment.* As discussed in the text, if every set of $N$ vectors on the $d$ hypercube could be mapped into the $\sqrt{N}$ hypercube by linear threshold functions in such a way that there is no linearly dependent subset of $\sqrt{N}/3$ vectors, then a two-hidden-layer perceptron with $O(\sqrt{N})$ neurons could implement an arbitrary dichotomy. Proposition 4 is evidence of the plausibility of this open conjecture.

*Proof.* Consider an $N \times k$ matrix $M$ of independently selected equally likely $\pm 1$'s. We wish to estimate the probability $p_j$ that some $j$-rowed subset of $M$ is linearly dependent. $p_j$ can be bounded from above by counting the expected number of dependent subsets of $M$ with $m \leq j$ rows. We can count dependent rows of $M$ by the following criteria:

(1) the $m$ rows are dependent but of rank $m - 1$,

(2) all $m$ rows are used in the linear relation.

[5] Informally, Occam's Razor is the philosophical principle that one should prefer the simplest hypothesis which explains any given set of facts.

We will later sum over $m$ from 2 to $j$ to bound $p_j$. Observe

$$\begin{aligned}
&\text{Prob}(m \text{ rows satisfy (1) and (2)})\\
&= \text{Prob}(\text{there is an } m-1 \times m-1 \text{ independent submatrix})\\
&\times \text{Prob}(\text{the defined relation has nonzero coefficients})\\
&\times \text{Prob}(\text{the remaining } k-m \text{ columns satisfy the relation})\\
&\qquad \le 1 \times 1 \times (c/\sqrt{m})^{k-m}, \qquad \text{(A1)}
\end{aligned}$$

where $c$ is a constant. Here we have simply bounded the first two probabilities by 1, and the bound on the last probability will be demonstrated shortly.

Let $E_m$ be the expected number of $m$-rowed subsets of $M$ which satisfy (1) and (2). We have

$$E_m \le \binom{N}{m}(c/\sqrt{m})^{k-m}.$$

Using Sterling's formula we estimate

$$\log(E_m) \sim m\log(N/m) - \frac{k-m}{2}\log(m),$$

where we have taken $k$, $N$, and $m$ large with $m/N$ small. Now take $N = k^\gamma$, $m = \delta k$ for fixed $\gamma$, $\delta$, and large $k$. Then

$$\log(E_m) \sim \delta k(\gamma - 1)\log(k) - \frac{k(1-\delta)}{2}\log(k) + O(k\log(k)).$$

So if

$$\delta(\gamma - 1) - \frac{1-\delta}{2} < 0, \qquad \text{(A2)}$$

$E_m$ is exponentially small. Clearly

$$p_j \le \sum_{m\le j} E_m,$$

and as long as Eq. (A2) is satisfied $p_j$ is exponentially small.

Taking $\gamma = 2$ we see $\delta = \frac{1}{3}$ satisfies Eq. (A2); i.e., taking $k = \sqrt{N}$, which is the case of interest, we see, as advertised, that almost every set of $N$

randomly chosen vectors on the $\sqrt{N}$ hypercube have no subset of $\sqrt{N}/3$ linearly dependent vectors.

It only remains to demonstrate the bound of Eq. (A1). Let $h$ be a linear relation (hyperplane) with integer coefficients. Let $n_i$ be the number of coefficients with value $\pm i$, and let $s$ be the largest coefficient (in absolute value). The probability that a random $\pm 1$ vector is on the hyperplane is given by

$$P = E(\cos^{n_1}(\theta)\cos^{n_2}(2\theta)\ \ldots\ \cos^{n_s}(s\theta)), \tag{A3}$$

where the expectation value is over uniform $\theta \in [0, 2\pi]$. Equation (A3) may be demonstrated by expanding $\cos(t\theta) = (e^{it\theta} + e^{-it\theta})/2$. The probability of being on the hyperplane is explicitly the number of $m$ dimensional vectors with components $\pm 1$ having inner product 0 with the linear relation and this quantity in turn is explicitly the constant term in $\Pi_{i=1,s}\cos^{n_i}(n_i\theta)$.

The probability in Eq. (A3) may now be bounded using Holder's inequality,

$$\begin{aligned} P &= E(\cos^{n_1}(\theta)\cos^{n_2}(2\theta)\ \ldots\ \cos^{n_s}(s\theta)) \\ &\leq \{E(|\cos^{n_1p_1}(\theta)|)\}^{1/p_1}\{E(|\cos^{n_2p_2}(2\theta)|)\}^{1/p_2}\ \ldots\ \{E(|\cos^{n_sp_s}(s\theta)|)\}^{1/p_s}, \end{aligned} \tag{A4}$$

where $p_1, \ldots, p_s > 0$ and $1/p_1 + 1/p_2 + \ldots 1/p_s = 1$.

Assume $n_1 + n_2 + \cdots + n_s = m$; i.e., the coefficients in $h$ are all nonzero. We get a bound on $P$ by taking $p_1n_1 = m, p_2n_2 = m, \ldots, p_sn_s = m$ which of course satisfies the constraint $\Sigma 1/p_l = \Sigma n_l/m = 1$. Plugging this into Eq. (A4), we find

$$P \leq \{E(|\cos(\theta)|^m)\}^{1/p_1}\ \ldots\ \{E(|\cos(s\theta)|^m)\}^{1/p_s} = E(|\cos(\theta)|^m).$$

Now $E(|\cos(\theta)|^m) \leq E(\cos^{m'}(\theta))$ where $m' = m$ if $m$ is even and $m' = m - 1$ if $m$ is odd. So

$$P \leq E(\cos^{m'}(\theta)) = 2^{-m'}\binom{m'}{m'/2} \sim \sqrt{2/\pi m}, \tag{A5}$$

where we used Sterling's formula. Equation (A5) is the bound used in Eq. (A1). Q.E.D.

## Acknowledgments

I thank L. E. Baum for helpful discussions, H. Rumsey for supplying the proof presented in Appendix A, and the referee for suggesting stylistic improvements. The research reported in this paper was performed by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Strategic Defense Initiative Organization, Innovative Science and Technology Office and the National Aeronautics and Space Administration. This work was performed as part of JPL's Center for Space Microelectronics Technology.

## References

ABU-MOSTAFA, Y. S. (1987a), Number of synapses per neuron, *in* "Analog VLSI and Neural Systems" (C. A. Mead, Ed.), Addison–Wesley, Reading, MA.

ABU-MOSTAFA, Y. S. (1987b), Random problems, *J. Complexity,* in press.

ABU-MOSTAFA, Y. S., AND PSALTIS, D. (1987), Optical neural computers, *Sci. Amer.* **256,** No. 3, 88–95.

ABU-MOSTAFA, Y. S., AND ST. JACQUES, J. (1985), Information capacity of the Hopfield model, *IEEE Trans Inf. Theory* **IT31,** 461–464.

ACKLEY, D. H., HINTON, G. E., AND SEJNOWSKI, T. J. (1985), A learning algorithm for Boltzmann machines, *Cog. Sci.* **9,** 147–169.

BAUM, E. B. (1986), Generalizing back propagation to computation, *in* "Neural Networks for Computing" (J. Denker, Ed.), pp. 47–53, AIP Conf. Proc. 151, Snowbird, UT.

BAUM, E. B., AND BAUM, L. E., in preparation.

BAUM, E. B., AND HAUSSLER, D., in preparation.

BAUM, E. B., MOODY, J., AND WILCZEK, F. (1986), Internal representations for associative memory, preprint NSF-ITP-86-138; *Biol. Cybernetics,* to appear.

BLUMER, A., EHRENFEUCHT, A., HAUSSLER, D., AND WARMUTH, M. (1986), Classifying learnable geometric concepts with the Vapnik–Chernovenkis dimension, *in* "Proc. 18th ACM Symp. on Theory of Computing," pp. 273–282, Assoc. Comp. Mach., New York.

BORSELLINO, A., AND GAMBA, A. (1961), An outline of a mathematical theory of PAPA, *Nuovo Cimento Suppl. 2* **20,** 221–231.

CHAITIN, G. (1974), Information theoretic computational complexity, *IEEE Tran. Inf. Theory* **IT-20,** 10–15.

COVER, T. M. (1965), Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Trans Electron. Comput.* **EC-14,** 326–334.

DERTOUZAS, M. (1965), "Threshold Logic: A Synthesis Approach," MIT Press, Cambridge, MA.

GAMBA, A., GAMBERINI, L., PALMIERI, G., AND SANNA, R. (1961), Further experiments with PAPA, *Nuovo Cimento Suppl. 2* **20,** 112.

GORMAN, P., AND SEJNOWSKI, T. J. (1987), Learned classification of sonar targets using a massively parallel network, *in* "Workshop on Neural Network Devices and Applications," JPLD-4406, pp. 224–237.

HINTON, G. E., AND SEJNOWSKI, T. J. (1986), Learning and relearning in Boltzmann machines, *in* "Parallel Distributed Processing" (D. E. Rumelhart, and J. L. McClelland, Eds.), Vol. 1, pp. 282–317, MIT Press, Cambridge, MA.

HOPFIELD, J. J. (1982), Neural networks and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. Sci. U.S.A.* **79,** 2554–2558.

HU, S.-T. (1965), "Threshold Logic," Univ. of California Press, Berkeley.

KHOLMOGOROV, A. (1965), Three approaches for defining the concept of information quantity, *Inf. Transmission* **1,** 3–11.

LAPEDES, A., AND FARBER, R. (1987), Nonlinear signal processing using neural networks: Prediction and system modelling, Los Alamos preprint LA-UR-87-2662.

LE CUN, Y. (1985), A learning procedure for asymmetric threshold networks, *Proc. Cognitiva* **85,** 599–604.

LEWIS, P. M., AND COATES, C. L. (1967), "Threshold Logic," Wiley, New York.

MCCULLOCH, W. A., AND PITTS, W. (1943), A logical calculus of the ideas immanent in neural nets, *Bull. Math. Biophys.* **5,** 115–137.

MCELIECE, R. J., POSNER, E. C., RODEMICH, E. R., AND VENKATESH, S. S. (1987), The capacity of the Hopfield associative memory, *IEEE Trans. Inf. Theory* **IT33.**

MINSKY, M., AND PAPERT, S. (1969), "Perceptrons, an Introduction to Computational Geometry," MIT Press, Cambridge, MA.

MUROGA, S. (1979), "Logic Design and Switching Theory," Wiley, New York.

NILSSON, N. J. (1965), "Learning Machines," McGraw–Hill, New York.

PAPERT, S. (1961), Some mathematical models of learning, *in* "Proceedings, 4th London Symp. on Information Theory" (C. Cherry, Ed.), Academic Press, New York.

PARKER, D. B. (1985), "Learning Logic," MIT Tech. Report TR-47, Center for Computational Research in Economics and Management Science.

ROSENBLATT, F. (1962), "Principles of Neurodynamics," Spartan, New York.

RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, G. E. (1986), Learning internal representations by error propagation, *in* "Parallel Distributed Processing" (D. E. Rumelhart and J. L. McClelland, Eds.), Vol. 1, MIT Press, Cambridge, MA.

SAMUEL, A. (1959), Some studies in machine learning using the game of checkers, *IBM J. Res. Dev.* **3,** 211–229.

SAMUEL, A. (1967), Some studies in machine learning using the game of checkers, Part II, *IBM J. Res. Dev.* **11,** 601–617.

SEJNOWSKI, T. J., AND ROSENBERG, C. R. (1987), NET Talk: A parallel network that learns to read aloud, *Complex Syst.* **1,** 145–168.

WERBOS, P. (1974), "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Harvard University dissertation.

WERBOS, P., AND TITUS, J. (1978), An empirical test of new forecasting methods derived from a theory of intelligence: The prediction of conflict in Latin America, *IEEE Trans. Syst. Man Cybernetics* **SMC-8,** No. 9, 657–666.