PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Concepts In Distributed Systems

Anderson, James, Golden, Richard, Murphy, Gregory

James A. Anderson, Richard M. Golden, Gregory L. Murphy, "Concepts In Distributed Systems," Proc. SPIE 0634, Optical and Hybrid Computing, (13 February 1986); doi: 10.1117/12.964018

Event: Optical and Hybrid Computing, 1986, Leesburg, United States

SPIE.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020 Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

Concepts in Distributed Systems

James A. Anderson, Richard M. Golden, and Gregory L. Murphy

Department of Psychology, Brown University, Providence, RI 02912

Abstract

We describe a parallel, distributed, associative model based on Hebbian modification of connection strengths between simple elements. Some extensions to the simple model are described and interpretations of the model as a gradient descent algorithm and as maximizing a probability density function are given. The model is applied to concept formation, since most versions of this modelling approach form equivalence classes of inputs that act like much like psychological concepts. Some computer simulations of concept-like behavior are described. Some kinds of computation can be performed effectively with these techniques.

Introduction

We have developed a series of parallel, distributed, associative models that are capable of what we feel are some interesting kinds of computation. This paper is composed of two parts. The first part is a summary of some of the analytic results on this modelling approach. The second part is a description of the concept forming behavior of these models. This part also contains examples of the somewhat non-traditional techniques that can be used to retrieve information from the system.

The linear autoassociator

We view model neurons as linear integrators that compute weighted sums of the firing frequencies of other neurons in the system. The firing frequencies are weighted by a collection of synaptic weights that specify how the neurons are interconnected. Information in such systems is represented by an N-dimensional state vector, $\underline{x}$, where the ith element of the state vector corresponds to the deviation of the firing rate of the ith neuron in the system from its spontaneous firing frequency. The real-valued synaptic weights represent the degree of synaptic coupling between individual neurons in the system and may be conveniently arranged within a matrix referred to as A. Thus, the A matrix specifies how a particular collection of N neurons are connected. The response of this collection of neurons to a neural activation pattern $\underline{x}$ is obtained by multiplying the matrix A by the vector $\underline{x}$ to obtain the response $A\underline{x}$. The strength of the response $A\underline{x}$ is defined as the magnitude of $A\underline{x}$, while the identity of the response $A\underline{x}$ is defined as the unit vector in the direction of $A\underline{x}$.

For simplicity, consider the special case where the real-valued A matrix is symmetric. In this case, the A matrix is diagonalizable and will possess a set of real orthogonal eigenvectors. Thus, the system will respond strongly to incoming activity patterns that can be represented as linear combinations of the eigenvectors associated with the largest eigenvalues of the A matrix, and will respond weakly to other types of activity patterns. To generalize these ideas to the case where there are no symmetry restrictions upon A it is useful to define the following 'energy' function which is simply the dot product of the response $A\underline{x}$ with the incoming activity pattern $\underline{x}$:

$$E(\underline{x}) = (-1/2)\underline{x}^T A\underline{x} \qquad (1)$$

where $\underline{x}^T$ denotes the transpose of $\underline{x}$.

When $E(\underline{x})$ is small, the system will respond strongly to the activity pattern $\underline{x}$. On the other hand, if $E(\underline{x})$ is large, the system will respond weakly to $\underline{x}$.

A gradient descent algorithm is an algorithm designed to find a state vector $\underline{x}$ such that $\underline{x}$ minimizes a particular performance function, $C(\underline{x})$. In particular, let a state vector $\underline{x}(k)$ be updated to obtain a state vector $\underline{x}(k+1)$ according to the following rule:

$$\underline{x}(k+1) = \underline{x}(k) - \eta \text{ GRAD } C(\underline{x}(k)) \qquad (2)$$

where η is a positive step constant and GRAD $C(\underline{x}(k))$ is the gradient of some cost function evaluated at $\underline{x}(k)$. This type of updating rule is called a gradient descent algorithm since the vector $\underline{x}(k+1)$ is obtained by moving away from $\underline{x}(k)$ in the direction of steepest descent (Duda and Hart, 1973). A well-known problem with such algorithms is that it is possible to

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

become stuck in local minima.

If we compute the gradient of $E(\underline{x})$ in the special case where A is real and symmetric, then we obtain $A\underline{x}$ which is simply the response to our original linear system. Thus, the linear autoassociator in this special case, may be viewed as a 1-step gradient descent algorithm that minimizes the value of $E(\underline{x})$.

The Brain-State-in-a-Box (BSB) model

The 'Brain-State-in-a-Box' or BSB model (Anderson et al., 1977) is a simple extension of the linear autoassociator that is also related to the energy function. The BSB model operates by amplifying an incoming activation pattern (state vector) using positive feedback until many of the neurons in the system have obtained their maximum or minimum firing rates. More formally the BSB model is defined as follows:

$$\underline{x}(k+1) = S(\underline{x}(k) + A\underline{x}(k)) \quad (3)$$

where $\underline{x}(k)$ is the state of the system at time interval k and the function S is a linearized sigmoidal function. In particular, $S(a) = a$ for $|a| \leq F$, $S(a) = +F$ for $a \geq F$, and $S(a) = -F$ for $a \leq -F$. The sigmoidal function S constrains the state vector x to remain within an origin-centered hypercube. When every element of a state vector has obtained either its maximum or minimum value, that state vector is said to be a hypercube corner.

A Liapunov function for a dynamic system is simply a scalar-valued function of the state vector $\underline{x}$ that can be used to prove theorems about the asymptotic stability of the system. Luenberger (1979) provides an introduction to such functions. A particular property of a Liapunov function that will be important in the following development, however, is that if $\underline{x}(k)$ is the state of a dynamic system at time k, then a Liapunov function $L(\underline{x})$ has the property that $L(\underline{x}(k+1)) \leq L(x(\underline{k}))$. It can be shown (Golden, 1986a) that the energy function in the first equation is a Liapunov function for the discrete-time dynamical system specified by the BSB model. The similarities between the BSB model and a gradient descent algorithm designed to minimize $E(\underline{x})$ should also be noted.

Probabilistic intepretation of learning and recall

The gradient descent and Liapunov approaches to the analysis of neural network models provide insights into the behavior of these models at the algorithmic level. A third perspective upon these systems, however, is also possible. In particular, the behavior of many of the neural network models discussed in the literature can be viewed within a probabilistic framework. Reformulating the information retrieval problem in this manner is appealing for several reasons. First, such an approach provides a unified theory of a large class of algorithms. Second, the goal of this class of algorithms can be stated clearly and concisely. And third, this type of framework provides new insights into the analysis and design of content-addressable memory systems in general.

Assuming a binary-valued vector data structure, the fundamental problem of content-addressable memory may be formulated as follows. Given some unusual or 'improbable' vector $\underline{x}(0)$, construct a more probable interpretation. More formally, we can search for a local maximum of some probability density function $P(\underline{x})$ in the vicinity of $\underline{x}(0)$. The probability density function $P(\underline{x})$ indicates the frequency of occurrence of a stimulus vector $\underline{x}$ within the environment. Within this framework, we can view a broad class of neural network models as specific gradient ascent algorithms that maximize $P(\underline{x})$, while some popular neural network learning algorithms are viewed as procedures that are estimating the general form of $P(\underline{x})$.

Recalling information by maximizing a probability density function

A large class of deterministic continuous-time neural network models (Hopfield, 1984), Hopfield's (1982) discrete neural model, the BSB neural model (Golden, 1986a), and the linear autoassociator have all been demonstrated to be gradient descent algorithms that minimize $E(\underline{x})$ over $\underline{x}$ (Equation 1). In fact, the function $E(\underline{x})$ may be shown to be a Liapunov function for these neural network models if the models are viewed as dynamical systems (Hopfield, 1984). The above class of models will be referred to as autoassociative network models that minimize energy.

Consider the following probability density function:

$$P(\underline{x}) = k\mathrm{EXP}\{(1/2)\underline{x}^T A\underline{x}\} \quad (4)$$

where EXP{} is the exponential function and k is a normalization constant. The gradient of $P(\underline{x})$ with respect to $\underline{x}$ is calculated as follows:

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

$$\mathrm{GRAD}(P(\underline{x})) = - \mathrm{GRAD}(E(\underline{x}))P(\underline{x}). \qquad (5)$$

Equation (5) states that an algorithm that moves along a path in the state vector space that decreases the value of E($\underline{x}$) most rapidly (i.e., the gradient of E($\underline{x}$)), is also moving along the path that increases the value of P($\underline{x}$) most rapidly. Moreover, the multiplicative factor P($\underline{x}$) in (5) modulates the step-size. Thus, when the state vector $\underline{x}$ is 'improbable' (i.e., P($\underline{x}$) is small), the step-size will be small. But when the state vector is 'probable' (i.e., P($\underline{x}$) is large), the step-size will be large. Finally note that since P($\underline{x}$) is a monotonically decreasing function of E($\underline{x}$) and since $E(\underline{x}(k+1)) \leq E(x(\underline{k}))$, an autoassociative model that minimizes 'energy' is also maximizing P($\underline{x}$). In psychological terms, such models are dynamically constructing 'more probable' interpretations of the initial state vector $\underline{x}(0)$.

Learning is estimating the form of the probability density function

In this section, the Hebbian and autoassociative Widrow-Hoff learning rules are viewed as algorithms that are trying to find 'reasonable' forms for the probability density function P($\underline{x}$). Our definition of a 'reasonable' form for P($\underline{x}$) will be a density function that obtains local maxima that correspond to equivalence classes of stimuli that have been learned by the system. An alternative procedure for estimating the form of P($\underline{x}$) might involve maximizing a maximum likelihood performance function (see for example, Ackley et al., 1985). Still, this definition of reasonable will lead to some useful new interpretations of the behavior of the classical Hebbian and autoassociative Widrow-Hoff learning rules.

Hebbian learning

A popular learning rule is the Hebbian learning rule that states that the change in the synaptic weight connecting the ith and jth neurons in the system is directly proportional to the product of the activities of the ith and jth neurons. In particular, at each learning trial, a member of the training stimulus set is randomly selected and used to update the synaptic connectivity matrix. Anderson et al. (1977) have noted that the expected value of the synaptic connectivity matrix in this case is a scalar multiple of the stimulus covariance matrix provided that the expected value of $\underline{x}$ is the zero vector. Consider the special case where the stimulus set consists of K orthogonal vectors where K is less than the dimensionality of the vector space. The stimulus covariance matrix is then spanned by K orthogonal eigenvectors associated with a single degenerate eigenvalue. Any hypercube corner in this K-dimensional subspace is therefore an eigenvector associated with the single eigenvalue of A. Given this observation, it is easy to show (Golden, 1986a), that the set of hypercube corners in this K-dimensional subspace are global minima of E($\underline{x}$) and therefore global maxima of P($\underline{x}$). Finally, note that the set of orthogonal hypercube corners that are learned by the system are members of this K-dimensional subspace. Note, however, if the K stimulus vectors are not orthogonal, then the stimulus vector set members do not necessarily obtain global maxima of P($\underline{x}$).

Autoassociative Widrow-Hoff learning

A more accurate learning rule is the autoassociative Widrow-Hoff learning scheme (Anderson, 1983). The autoassociative Widrow-Hoff learning rule is defined as follows:

$$A(t+1) = A(t) + \eta(\underline{x} - A(t)\underline{x})\underline{x}^T \qquad (6)$$

where A(t) is the value of the matrix at learning trial t, $\underline{x}$ is a stimulus vector that is used to update the matrix of synaptic weights at learning trial t, and η is a positive step constant. Typically, a subset of the matrix elements in A are set equal to zero before learning begins and remain equal to zero throughout the learning process to model a nervous system that, realistically, has limited connectivity.

It can be shown that the Widrow-Hoff learning rule is a type of gradient descent algorithm that searches for an A that minimizes the average Euclidean distance between A$\underline{x}$ and $\underline{x}$. In particular, the average is taken with respect to the probability distribution of the stimulus vectors taught to the system (Widrow, 1971). Qualitatively then, the autoassociative Widrow-Hoff learning rule may be viewed as searching for an A matrix that will result in a function

$$P(\underline{x}) = k\mathrm{EXP}\{\underline{x}^T A\underline{x}\}$$

whose local maxima are associated with the members of the training stimulus set (see Golden, 1986b, for additional details). Note, however, that this learning rule is not a

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

strict gradient descent procedure, since only a single member of the stimulus set is used to update the matrix at each learning trial. Assuming, however, that the step constant is very small, then such a learning rule approximates a true gradient descent very closely.

Generality of the quadratic probability density function

The large class of autoassociative neural network models that we have considered are all maximizing the quadratic probability density function given in Equation (4). This suggests, in conjunction with current speculation about the existence of Hebbian (pair-wise) synapses in the nervous system, that the form of the density function in (4) might be a reasonable one to assume from a neurophysiological perspective. From a computational perspective, however, it should be clear that certain probability distributions of the binary state vector $\underline{x}$ might exist that simply can not be adequately represented using Equation (4).

As an example of this statement, consider a random vector $\underline{x}$ that takes on the value $\underline{x}_1=(1,-1)$ with probability 0.75 and $\underline{x}_2=(-1,1)$ with probability 0.25. We now show that the probability density function (4) cannot adequately represent this particular probability distribution of the random vector $\underline{x}$. Using Equation (4), we have the following system of equations in terms of the elements of the A matrix, where a_{ij} refers to the ijth element of the A matrix.

$$0.75 = P(\underline{x}_1) = k\mathrm{EXP}\{\underline{x}_1^T A\underline{x}_1\} \qquad (7)$$

$$0.75 = k\mathrm{EXP}\{(1)(1)a_{11} + (1)(-1)a_{12} + (-1)(1)a_{21} + (-1)(-1)a_{22}\}$$

$$0.25 = P(\underline{x}_2) = k\mathrm{EXP}\{\underline{x}_2^T A\underline{x}_2\} \qquad (8)$$

$$0.25 = k\mathrm{EXP}\{(-1)(-1)a_{11} + (-1)(1)a_{12} + (1)(-1)a_{21} + (1)(1)a_{22}\}$$

This is an inconsistent system of equations, and a probability density function of the form in (4) can not be found (see Minsky and Papert, 1969, for a discussion of related problems). This observation suggests that the following 'coding theorem' might be useful.

A coding theorem

At least one two-layer neural network model exists that specifies a particular quadratic probability density function of the form described in Equation (4). Moreover, this probability density function can adequately represent any arbitrary probability distribution of binary-valued stimulus vectors that may occur in the model's environment. The first layer of the neural network contains no more than 2^N perceptrons, while the second layer of the neural network (which is specified by no more than 2^N synaptic weights) is an autoassociative network model that minimizes 'energy.'

To motivate the proof, consider again the problem where the vector $\underline{x}_1 = (1,-1)$ occurs with probability 0.75 and the vector $\underline{x}_2 = (-1,1)$ occurs with probability 0.25 within some environment. A nonlinear mapping from this two-dimensional vector space to a higher dimensional (three-dimensional) vector space is now defined. In particular, a specific vector $\underline{x} = (x_1,x_2)$ is mapped into a specific vector $\underline{y} = (x_1,x_2,x_1x_2)$. Thus, $\underline{x}_1 = (1,-1)$ maps into $\underline{y}_1 = (1,-1,-1)$, and $\underline{x}_2=(-1,1)$ maps into $\underline{y}_2=(-1,1,-1)$. If we were to attempt to solve for the nine matrix weights that assign the probability 0.75 to the vector $\underline{y}_1$ and the probability 0.25 to the vector $\underline{y}_2$ using Equation (4), we would be successful.

Finally note that a single layer consisting of three perceptrons can be used to make the required transformation from the $\underline{x}$ vector space to the $\underline{y}$ vector space. The first two perceptrons simply map the first and second elements of $\underline{x}$ into the first and second elements of $\underline{y}$. The third perceptron can be made, by appropriately selecting its threshold, to function as a logical AND gate that generates the value 1 only when both elements of $\underline{x}$ are equal to 1, and a -1 otherwise. The output of this third perceptron is precisely the output that is required to generate the third element of the $\underline{y}$ vector.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

A proof of the coding theorem

We begin the proof by forming a state vector $\underline{y}$ whose first element is a constant, whose second N elements are identical to the state vector $\underline{x}$, and whose remaining elements correspond to the 'higher-order' correlations between the elements of $\underline{x}$. Such a mapping can be easily carried out by a single layer of perceptrons that are functioning as AND gates. For example, to represent an element in the $\underline{y}$ vector that represents the correlation between K elements in the $\underline{x}$ vector, it is only necessary to select the threshold of a simple perceptron such that the perceptron works like a K-input AND gate (see Figure 1).

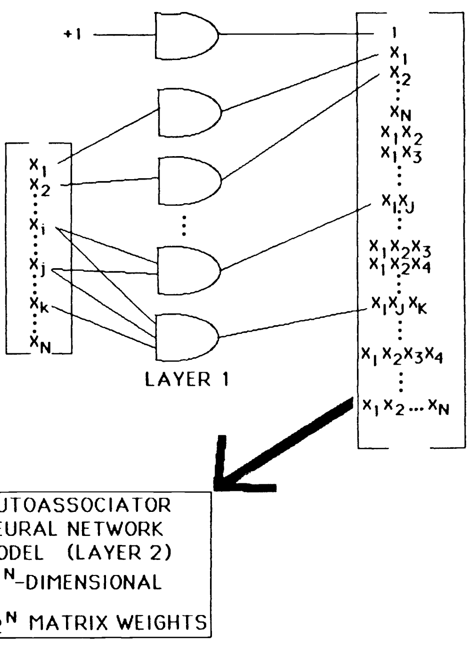

Figure 1. A two-layer neural network that can represent any arbitrary probability density function. Only a small subset of the perceptrons (AND gates) in layer 1 and their interconnections with layer 2 are displayed for the purpose of clarity.

If the vector $\underline{y}$ is now substituted into the quadratic probability density function in Equation (4), the resulting density function will still be quadratic in terms of the elements of the $\underline{y}$ vector but the density function will be of the following form in terms of the elements of the $\underline{x}$ vector:

$$P(\underline{y}) = k\mathrm{EXP}\{Q(\underline{y})\} \qquad (9)$$

$$Q(\underline{y}) = a_0 + \Sigma a_i x_i + \Sigma\Sigma a_{ij} x_i x_j + \Sigma\Sigma\Sigma\, a_{ijk} x_i x_j x_k + \dots + a_{12\dots n} x_1 x_2 \dots x_n$$

where a_0, a_i, a_{ij}, a_{ijk}, etc. are the parameters of the density function $P(y)$ and specify the synaptic connectivity A matrix, and the x_i are elements of the state vector $\underline{x}$.

Besag (1974; see Appendix 1) has demonstrated that any arbitrary probability density function of binary-valued vectors can be represented by a probability density function of this form. Moreover (see Appendix 1), many of the coefficients in Equation (9) are not required and may be set equal to zero. In particular, 2^N coefficients are required.

Concepts in distributed models

There are two related but distinct ways of explaining simple concepts in connectionist models. First, there are prototype forming systems which involve taking a kind of average during the act of storage. We will describe how this occurs in a simple linear associative system. Second, there are models which explain concepts as related to attractors in a dynamical system. We will develop some ideas for concept formation in a non-linear system in this section using this approach.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

There is a third way, which involves the construction during the learning process of very selective elements (model neurons) and lets the very selective elements stand for concepts. Such 'grandmother cell' forming systems do not agree with the psychological or physiological data. The neural representation of even simple concepts must involve changes in activities of many simple elements -- i.e. it is a distributed code. Certainly, there are many moderately selective elements in both a learning system and in the brain, but interesting psychological behavior corresponds to a good many more than one neuron changing its firing rate. (See Anderson and Mozer, 1981).

The first two concept models are closely related formally. In these comments we will briefly describe the two approaches and the relation between them and then present some computer simulations showing how the suggested connections function in practice.

Human concepts

At a general level, it is easy to explain what a concept is: A concept is a mental representation or rule that picks out a class of objects or events. We will discuss two aspects of simple human concept formation: representation of a category by a prototype and hierarchical structure.

Representation of a simple concept by a prototype or best example, is associated with the name of Rosch (1975, 1978). Exemplars are judged to be category members by their closeness to the prototype. For example, sparrows or robins are close to the prototypical bird and are therefore 'good' or typical birds, but vultures or penguins are not, even though they are biologically just as good birds. Considerable evidence demonstrates that people process items differently, depending on how typical they are. Subjects tend to have faster reaction times to prototypical items, judge them as 'better' representatives of a name, and will frequently produce them as examples of the category.

Another important aspect of human concepts is their hierarchical structure. Concepts seem to be organized in a nested fashion: robin-bird-animal-living thing or rocking chair-chair-furniture-artifact, for example. This structure allows people to economically represent information about increasingly general categories (Collins and Quillian, 1969). So, the facts people know about birds apply to robins, larks, crows, etc., and the facts people know about animals apply to birds, mammals, reptiles, etc. These hierarchies allow us to draw inferences about objects. Information about all animals is not repeated with each animal, but is stored as true about the animal concept in general. We will give a simple example of a hierarchy in the simulation to be described.

Our fundamental modelling assumption, common to all neural network models, is that information is carried by the pattern or set of activities of many neurons in a group of neurons. This set of activities carries the meaning of whatever the nervous system is doing. We represent these sets of activities as state vectors. Concepts, percepts, or mental activity of any kind are similar if their state vectors are similar. The component values in the state vectors might correspond to the activities of moderately selective neurons.

Prototype formation

The simple linear model presented in the first part will form prototypes as part of the storage process. Let us make the coding assumption that the activity patterns representing similar stimuli may be represented as correlated state vectors. This means the inner product between two similar patterns is large. Now consider the case where the model has made the association $\underline{f} \rightarrow \underline{g}$ using a simple linear associator. With an input pattern $\underline{f}'$ then

$$\text{(output pattern)} \propto \underline{g}\underline{f}^T\underline{f}'$$

If $\underline{f}$ and $\underline{f}$ are not similar, their inner product is small. If $\underline{f}$ is similar to $\underline{f}'$ then the inner product will be large. The model responds to input patterns based on similarity to $\underline{f}$.

Consider a situation in which a category contains many similar items. Here, a set of similar activity patterns (representing the category members) becomes associated with the same response, for example, the category name. It is convenient to discuss such a set of vectors with respect to their mean. Let us assume the mean is taken over all potential members of the category. Specifically, consider a set of correlated vectors $\{\underline{f}\}$, with mean vector $\underline{p}$. Each individual vector in the set can be written as the sum of the mean vector and an additional noise vector, $\underline{d}^{(i)}$, representing the deviation of that vector from the mean, that is,

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

$$\underline{f}^{(i)} = \underline{p} + \underline{d}^{(i)} .$$

If there are n different patterns learned and all are associated with the same response, $\underline{g}$, the final connectivity matrix will be

$$A = \sum_{i=1}^{n} \underline{g}\, \underline{f}^{(i)T}$$

$$= n\, \underline{g}\, (\underline{p}^{T} + \sum_{i=1}^{n} \underline{d}^{(i)T})$$

Suppose that the term containing the sum of the noise vectors is small. In that case, the connectivity matrix is approximated by

$$A \propto n\, \underline{g}\, \underline{p}^{T} .$$

The system behaves as if it had repeatedly learned only one pattern, $\underline{p}$, proportional to n times the mean of the set of vectors it was exposed to. Under these conditions, the simple association model extracts a prototype just like a signal averaging program. In this respect, the distributed memory model behaves like a psychological prototype model, because the most powerful response will be to the pattern $\underline{p}$, which it may never have seen.

However if the sum of the $\underline{d}$'s is not relatively small, as might happen if the system only sees a few patterns from the set and/or if $\underline{d}$ is large, the response of the model will depend on the similarities between the novel input and each of the learned patterns, that is, the system behaves like a psychological exemplar model. (An exemplar model does not extract a prototype, but rather represents a category in terms of specific category members it remembers. See Smith and Medin, 1981, for details.)

There is a kind of tension between exemplar storage and prototype formation in these models, with the balance between them dependent on the parameters and the details of the system. There is little agreement in the psychological literature as to which type of model is better. It is an advantage of the connectionist concept model described here that it allows for intermediate cases between pure storage of exemplars and formation of prototypes. These models can do both simultaneously, or act as one or the other. Where a particular case falls on the continuum depends on the exact parameters used and the detailed structure of the exemplars.

Knapp and Anderson (1984) applied this model to the formation of simple 'concepts' composed of nine randomly placed dots, a classic concept formation experiment (Posner and Keele, 1968). Using a straightforward realization of the linear model presented above, Knapp and Anderson were able to give a reasonable fit to a set of experimental data.

Implications of error correction for concept formation

The simplest kind of concept formation, the averaging prototype extractor, is strongly affected by error correction. In this discussion, let us only consider the autoassociative system. Suppose we are learning a number of examples, as before, then

$$A = \sum_{i=1}^{n} \underline{f}^{(i)}\, \underline{f}^{(i)T}$$

$$= n\, (\underline{p} + \sum_{i=1}^{n} \underline{d}^{(i)T})\, (\underline{p}^{T} + \sum_{i=1}^{n} \underline{d}^{(i)T})$$

If the $\underline{d}$ terms are relatively small, then the final matrix is approximately

$$A \propto n\, \underline{p}\, \underline{p}^{T}$$

implying that

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

$$A \underline{p} \propto n \underline{p}$$

That is, an eigenvector of A will be close to the prototype $\underline{p}$ with eigenvalue proportional to n. Since the examples are removed from the prototype by a distance, we have a system that (in the limit of small distortions) responds best to the average, that is, there is a response peak, in terms of length of output vector, at the prototype location. There is also a strong dependency of the eigenvalue (a measure of the strength of response) on the number of presentations. This is psychologically sensible for small n, but if one was to consider the enormous relative frequencies of presentation of words, say, this result becomes unreasonable. It would predict that frequent words would completely dominate the response of the lexicon, since the relative retrieval strengths of different words, based purely on frequency, could potentially be in the tens of thousands. (For example, in the Francis and Kucera, 1982, sample of one million words, 11 words occur more frequently than 10,000 times, whereas more than 30,000 words occur fewer than 5 times. Clearly, the frequency disparity in natural language is enormous.)

When we apply the Widrow-Hoff error correction algorithm (Equation 6), the picture changes dramatically. Let us assume our error correction has worked perfectly, so that given any example,

$$Af^{(i)} = f^{(i)}$$

In this case all the members of the stimulus set become eigenvectors and have eigenvalue 1. Clearly, the system responds well to examples. What has happened to response to the prototype?

Since all the eigenvalues associated with the stimlus set are 1, the set of eigenvectors representing the stimulus set is degenerate. This set of eigenvectors forms a subspace where every vector in the subspace is an eigenvector of A. One way to show the structure of the learned stimulus set is to consider what would happen if we presented a sum of exemplars to the system. (The sum would be found by simply summing the vectors of each exemplar and dividing by the number of exemplars.) In this system, any sum of exemplars (which includes the average, the prototype in the previous system) is an eigenvector. The exact geometry can be involved, but if such a solution can be found by the matrix, it will have this property.

Therefore all sums of exemplars are 'equally good' and will give the same amplitude of response, forming a subspace. (In the simple prototype model, the more exemplars involved in the sum, the more the sum approximates the prototype, and the better the prototype would be in generating a response.) Newly presented exemplars lying outside the subspace will also give a response, based upon their projection onto the subspace. Internal to the subspace, the response to exemplars is not 'peaky' but 'flat' with no effect of frequency of presentation. Therefore, the 'prototype' in a psychological sense perhaps can be interpreted as a region, not as a single vector. Corresponding to intuition, the actual average, the prototype in the simpler system, will be toward the center of this subspace. Note the similarity of this effect to the 'learning subspace' technique of pattern recognition (Kohonen, 1984).

One technical point: None of our simulations (or the analysis of error correction) assumes the matrix is symmetric. Our simulations are 50% connected. There is no reason to require the eigenvectors be orthogonal or real, and in fact this is seen in the computation of the eigenvectors in the simulations. We have not investigated the geometry of this system in much detail, but this study is underway. The BSB model is nonlinear. However, because the BSB model is linear for small signals, we can make realistic predictions about the behavior of the system from the above analysis.

Retrieval

Assume we have an autoassociative matrix that has learned a set of vectors $\{\underline{f}\}$. We start by assuming we have some information, and want more information to be given to us by the system. Starting information is represented by a vector constructed according to rules used to form the original vectors, except missing information is represented by zeros. If we have no information to start with, generally nothing significant can be retrieved. The BSB algorithm is applied to the initial state, and a final state is obtained. This final state will be the output of the system. The final state can be interpreted according to the rules used to generate the stimuli. The BSB model fills in missing information based on its past experiences using the interconnections represented in the matrix.

General description of simulations

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

In the specific examples of state vector generation that we will use for the simulations to follow, letters, words and sets of words are coded as concatenations of the bytes in their ASCII representation. A parity bit is present. Zeros are replaced with minus ones. For example, an 's', ASCII 115, is represented by -1 1 1 1 -1 -1 1 1 in the state vector. A 200 dimensional vector would represent 25 alphanumeric characters. This is a 'distributed' coding because a single letter or word is determined by a pattern of many elements. In the outputs from the simulations, the underline, '_', stands for an uninterpretable character whose amplitude is below an interpretation threshold. The output strings displayed are only those characters of which the system is 'very sure' because the element values that generate output characters were all above a high threshold.

All the BSB simulations described here are 200 dimensional (25 characters). The autoassociative matrix, constructed using the Widrow-Hoff technique (Eq. 6) is 50% connected, that is, half the elements (in these simulations, 100 of them), randomly chosen, are set identically to zero. The matrix is not symmetric. During learning, items to be associated are chosen randomly from the stimulus set.

Eigenvectors

We have pointed out before (Anderson et al., 1977, Anderson and Mozer, 1981) the importance of the eigenvectors in analysis. We proposed that the eigenvectors with large positive eigenvalues were related to the 'distinctive features' of the stimulus set, or, as we called them in several papers, the 'macrofeatures,' were related directly to eigenvectors of the connectivity matrix.

We constructed a stimulus set to check some of the properties we expected the eigenvectors to show in the learning system. The stimuli are given in Table 1. These stimuli are constructed from three different parts, with varying degrees of redundancy. The right-most part of the state vector is unique to a particular stimlus. The left-most part is the same in six stimuli. The set of stimuli has a pronounced 'hierarchical' structure in that segments of the state vector to the right determine the segments to their left. The system, therefore, is learning about vehicle and furniture concepts, with a mix of generic and specific information.

Table 1

Stimuli in Learned Set

F[1]. VehiclesAirplaneJetPlane_	F[7]. FurniturChairs Ostuffed_
F[2]. VehiclesAirplaneX15Rockt_	F[8]. FurniturChairs Diningrm_
F[3]. VehiclesAirplanePropplne_	F[9]. FurniturChairs Swivelch_
F[4]. VehiclesNewcars Porsche _	F[10]. FurniturBgtablesWorkshop_
F[5]. VehiclesNewcars Mercedes_	F[11]. FurniturBgtablesKitchen _
F[6]. VehiclesNewcars Chevrolt_	F[12]. FurniturBgtablesClassrom_

We showed that the error correction procedure applied to a set of exemplars, will tend to make the exemplars eigenvectors of the system matrix, with an eigenvalue of 1. Without error correction, the eigenvalues can be very large. We would expect the eigenvalue spectrum of the largest eigenvalues to contract toward 1 as error correction proceeds. This effect is clearly seen in Table 2, where the ratio of the largest to the tenth largest eigenvalue goes from about 12 with no error correction to 1.01 after 800 learning trials using error correction.

The actual eigenvectors, presented in Table 3, are a little harder to understand. One obvious early tendency is to have an eigenvector associated with the most recently presented stimulus. For example, after 50 presentations, the largest eigenvector is very close to F[3], which was the last stimulus presented. As learning progresses, this trend becomes less pronounced. The eigenvectors are clearly stimulus-like, but in detail, they can differ significantly. The most frequently occurring pattern segments (Vehicles, Furniture) appear most clearly in the eigenvectors.

Note that more and more eigenvectors become meaningful as learning progesses. The eigenvector with the tenth largest eigenvalue does not become stimulus-like until 200 presentations. The system is gradually structuring its eigenvectors for most accurate recall or, one might say, developing a more useful set of vector valued 'features' (macrofeatures). However the actual eigenvectors are still not exactly the same as the stimuli, even though retrieval is virtually perfect at 800 presentations.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

Table 2

Ratio of Largest Eigenvalue to Tenth Largest Eigenvalue

Presentations	Largest e.v.	10th Largest	Ratio
12	3.830	0.331	11.57
50	1.000	0.374	2.67
100	1.000	0.555	1.80
200	1.000	0.765	1.31
400	1.000	0.928	1.08
800	1.000	0.986	1.01

There were 12 stimuli in an autoassociative system. The matrix was 50% connected. Except for the first point in the table, (only one presentation of each stimulus) the Widrow-Hoff learning rule was used.

Table 3

Characteristic Eigenvectors

Number Pres.	Interpretation	Rank	Eigenvalue
12	VuxmidesBetabpg!Cmtscdod_	1	3.830
	#CIi7K#?-#r6mA¨EX####iLW_	2	1.475
	#K##srok#uTmd=#0xŸ&#^´pl_	10	0.331
50	VehiclesAirplanePropplne_	1	1.000
	VehiclesAirplaneP2/PXoje_	3	0.988
	BurniturNw5cc<urChmvr_l|_	5	0.843
	SgpfidusLmf0oz P->[kkp,_	10	0.374
100	FurniturBfeir¨!2GM`kossm_	1	1.000
	FurniturAh`x¨a,eP2-\lmzl_	3	0.994
	VehiclesAhkx|c*%LezIler#_	5	0.983
	T5H#7hR#qxb8\C#elrvy#P#´_	10	0.555
200	FurniturChairs 0stuffdd_	1	1.000
	FurniturChairs Ls|uffde_	3	0.999
	FuznkturAhax¨s($LexImer+_	5	0.978
	VehiclesAhbx|a.eLnvyrrue_	10	0.765
400	FurniturChairs 0stuffed_	1	1.000
	FurniturNdeksrl 0gpggfer_	3	1.000
	FurniturChairs Dinilgro_	5	0.996
	FurniturNfeks¨12P+;Nkiz0_	10	0.928
800	FurniturChairs 034Uffet_	1	1.000
	VehiclesLgvrml_wI,0Cyqm3_	3	1.000
	FurniturChairs Uzkmfnqh_	5	1.000
	FurniturAhax¨s($T++*i{ir8_	10	0.986

The eigenvectors were interpreted with the most 'meaningful' sign. The threshold of all interpretations was 0 for normalized eigenvectors.

Using the hierarchy

Up to this point, we have been teaching the model concepts at one level of abstraction. However, as discussed earlier, people do not only divide the world up this way--they also use concepts at different levels of abstraction, from very specific (e.g., antique Ford) to very general (e.g., physical object). Often, these concepts are arranged hierarchically, and we have structured the stimulus set in Table 1 the same way. For example, "a swivel chair is a chair is furniture" is one line in the hierarchy.

Retrieval respects the hierarachy. If we put in one field, e.g., 'Airplane,' superordinate information (to its left) is usually reconstructed correctly, in whole or part, whereas subordinate information (to its right), which is ambiguous, is extremely slow to be reconstructed and sometimes nonsense (i.e., a meaningless string of characters).

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

If the eigenvectors representing the stimuli are nearly degenerate, then, as shown earlier, sums of stimuli should give about the same response as the learned stimuli. This is easy to check and agrees with theory. If 'JetPlane' is input, the recall process reconstructs 'Vehicles' after 16 iterations and 'Airplane' after 15 iterations. The sum of 'JetPlane' and 'X15Rockt' gives 'Vehicles' after 17 iterations and 'Airplane' after 16 iterations. Other combinations give essentially the identical pattern: sums (i.e., new exemplars contained within the subspace of the old exemplars) are within one or two iterations of the effectiveness of learned patterns.

Some more complex computational combinations can be implemented with the system. Suppose we want to know what 'JetPlane' and 'Chevrolt' have in common. They are clearly neither 'Airplanes' or 'Newcars' but both are 'Vehicles'. If we put in the sum of 'JetPlane' and 'Chevrolt,' the system reconstructs 'Vehicle' after 20 iterations. There is no reponse in the second field after 50 iterations. As another example, the sum of 'Ostuffed' and 'Workshop' gives 'Furniture' after 17 iterations and no response in the second field after 50 iterations.

Table 4 gives several examples. Note the indications of a 'prototype effect'. The sum of all six examples of 'Vehicles' and the sum of all six examples of 'Furnitur' gives a faster reconstruction time than any single example. These examples also constructed more specific category names, 'Airplane' and 'Bgtables', which do not fit half the members of the sum, but took a long time to do so. A careful study confirms, in the non-linear system, the frequently found pattern of faster response to the prototype seen in the linear system (Anderson and Murphy, 1986).

These examples and others, which almost always work in the expected way, indicate how appropriate use of commonalities can automatically raise or lower the level of computation in a concept hierarchy.

Table 4

Examples of Computations with a Hierarchy

Input	Result	
JetPlane	Vehicles (16)	Airplanes (15)
JetPlane + X15Rockt	Vehicles (17)	Airplanes (16)
JetPlane + X15Rockt + Propplne	Vehicles (16)	Airplanes (15)
JetPlane + Chevrolt	Vehicles (20)	-- (50)
Mercedes	Vehicles (19)	Newcars (19)
Mercedes + Porsche	Vehicles (21)	Newcars (20)
Mercedes + X15Rockt	Vehicles (23)	-- (50)
Diningrm	Furnitur (15)	Chairs (14)
Diningrm + Ostuffed	Furnitur (19)	Chairs (19)
Diningrm + Workshop	Furnitur (17)	-- (50)
Diningrm + Kitchen	Furnitur (17)	Bgtables (37)
JetPlane + X15Rockt + Propplne + Porsche + Mercedes + Chevrolt	Vehicles (15)	Airplane (33)
Ostuffed + Diningrm + Swivelch + Workshop + Kitchen + Classrom	Furnitur (13)	Bgtables (25)

The positions of the input strings in the state vector can be seen by referring to the stimulus set in Table 1. The number in parentheses is the number of iterations required to completely reconstruct the characters shown. '--' means nothing was successfully reconstructed.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

Redundancy

Although we were primarily interested in the eigenvector structure the system developed, during testing we observed an interesting phenomenon. The segment in the rightmost position completely determines the other components, that is, if the right most postion is 'Porsche ' then the stimulus must be F[5]. But during retrieval, the rest of the information is retrieved more accurately and more quickly if some redundant information is added. This effect was strong, universal, and the explanation is theoretically obvious. As a typical example, if 'JetPlane' was placed in the right most position of the state vector, which uniquely determines the stimlus, F[1], it took 16 iterations to reconstruct the two missing positions. If a redundant 'Airplane' was added to the second position, it took 7 iterations to reconstruct F[1].

Although this particular simulation reconstructed all the missing segments correctly, when errors were present, they were greatly reduced with redundant information. In a more difficult, but related simulation using a similar structure, there were 90% errors of reconstruction if only the rightmost positions were filled, while, if a redundant position to the left was filled, the error rate dropped to 15%.

This result is interesting because frequent use of redundancy seems common in much human cognition (Garner, 1974). A traditional explanation of the effect of redundancy is that it increases the signal-to-noise ratio of the signal. In the neural models an effect related to the retrieval mechanism occurs as well: redundant information greatly increases accuracy and speed even though signal-to-noise ratio of the signal is effectively infinite (i.e., we know exactly what the signal is). By using redundancy we increase the signal-to-noise ratio and speed of memory retrieval rather than of the signal. We know that the typical elephant foot is big and slow moving. Yet, in terms of vivid writing, phrase two, highly redundant in terms of information content, might be preferred:

S[1]. The foot of the elephant squashed the watermelon.
S[2] The massive, slowly moving foot of the elephant squashed the watermelon.

Retrieval techniques: a simulation

We will discuss concept oriented retrieval techniques with examples from a reasonably complex data base. This data base contains information about military history, birds, and animals, a somewhat arbitrary selection corresponding to the kind of unstructured mess typical of most real world conceptual associative networks.

The stimulus set is given in Table 5. This set is purely autoassociative. The 200 dimensional state vectors represent 25 characters. The state vector is divided into five five character fields. Inspection will show a good deal of structure in the state vectors. For example, there is a hierarchy involving 'Wr' including 'Tnk's, 'Plns' and Civil War soldiers. There is also a separate hierarchy involving birds and animals that is connected to the military hierarchy because of the tendency to give weapons the names of dangerous animals. It is this kind of analogical connection that is the despair of AI programmers and the delight of cognitive scientists.

Table 5.

(Stimulus set is autoassociative)

F[1].	Wr WeapnKill Maim $$$$$	F[13]	WrPlnMosqtTwoEnWood GrBrt
F[2].	Wr HumanSoldrFightBrave	F[14].	WrPlnSptfrOneEnFightGrBrt
F[3].	WrCivBtwenStats19thCUSA	F[15].	WrPlnHurcnOneEnFightGrbrt
F[4].	WrCivShrmnSoldrGenrlUnion	F[16].	WrPlnFW190OneEnFightGermn
F[5].	WrCivGrantSoldrGenrlUnion	F[17].	WrPlnME109OneEnFightGermn
F[6].	WrCivLee SoldrGenrlConfd	F[18].	AnimlTigerCarniLargeDangr
F[7].	WrTnkWeapnGuns ArmorDangr	F[19].	AnimlPnthrCarniBlackDangr
F[8].	WrTnkTiger88mm.EffctGermn	F[20].	Bug MosqtInsctIrritFlyng
F[9].	WrTnkPnthr88mm.HeavyGermn	F[21].	PeaceDove Bird OliveFlyng
F[10].	WrTnkShrmnMedimM4 USA	F[22].	SprngRobinBird WingsFlyng
F[11].	WrTnkLee EarlyM3 USA	F[23].	Copy MckbrBird WingsFlyng
F[12].	WrPlnWeapnGuns BombsFlyng	F[24].	Animl Bird WingsFlyng

These simulations were done using a 50 percent connected 200 dimensional autoassociative system. The matrix was constructed using the Widrow-Hoff procedure, randomly presenting members of the stimulus set for 2,500 learning trials. The character '_' in a vector corresponds to eight zeros at that location. Final state vectors are constructed using the BSB procedure.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

Disambiguation by context

Distributed models in general are effective disambiguators by context (see Anderson and Murphy, 1986; Kawamoto, 1985). Looking at the stimulus set, we see that 'Dangr' appears both in the context of 'Wr' (F[7]) and 'Animl' (F[18] and F[19]). Putting in the two partial state vectors, the expected disambiguation occurs:

Wr __________________Dangr → WrTnkWeapnGuns ArmorDangr

and

Animl _______________Dangr → AnimlTigerCarniLargeDangr

Disambiguation by addition

A particularly exotic form of disambiguation can be accomplished by a technique that is partially analyzed above. Note that the stimulus set contains three Civil War generals, Sherman, Grant and Lee. Two of these (Sherman and Lee) are the names of tanks in the data base. If we put in the average of the vectors corresponding to the names of the generals associated with tanks, we might be able to retrieve the data associated with tanks. Similarly, if we put in the average containing a general only associated with the Civil War (Grant), then we might be able retrieve the data associated with Civil War, as the most common association. This actually works:

```
_____Shrmn________________
          +                 →  WrTnkLee  EarlyM3   USA
_____Lee   ________________

_____Shrmn________________
          +                 →  WrCivGrantSoldrGenrlUnion
_____Grant________________
```

The twist

One simple way of chaining associations in an autoassociative system involves intervention by a controller. There must be a control structure which is willing and able to intervene actively in the retrieval process if the system is not giving the right answers. This assumption obviously requires some complex, somewhat ill-defined machinery that we will not discuss further now. However, successful use of such memories will require a whole set of heuristic retrieval techniques of a somewhat arbitrary kind. A few speculations about such techniques are given below. There are other ways to chain associations, involving direct realization of network structures by building heteroassociation into the system. An example is given in Anderson (1986).

If we assume the controller has the ability to selectively blank portions of a retrieved state vector and then apply the memory again, we get a whole series of chained associates of a complex nature, which we called a 'twist. One uses the 'twist' technique by putting in a probe bit of information, say 'AB'. (Let us assume the state vector is partioned into five fields, as it is in this example.) Then a twisted chain might be:

```
AB___
        →
              ABCDE

              ___DE
        ←
GHIDE
```

Note the controller intervenes by blanking ABC and then rentering the matrix with DE. An example from the data base would be a chain to retrieve data about different kinds of tanks, but with a somewhat arbitrary method:

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

```
_____Pnthr__________ → WrTnkPnthr88mm.HeavyGermn

(Replace all positions with zero except WrTnk.)

WrTnk_______________ → WrTnkShrmnMedimM4   USA
```

Common features

One of the strongest predictions of distributed concept systems is that commonly occurring features should be retrieved more quickly than features that are rare or atypical. See the previous discussion in this paper for a more careful discussion of this important point, as well as Knapp and Anderson (1985). If we put in a state vector containing only 'Bird' it retrieves 'Wings' and 'Flying' very quickly (after 16 iterations) but takes over 100 iterations to retrieve the name of a particular bird. Items typical of a concept name are retrieved rapidly and more specific items are retrieved slowly.

War and peace

It is often difficult to predict exactly what will happen in a linked system such as our set of state vectors. A number of variations on this theme are possible. For example, using the techniques described above, it is quite difficult to get from War to Peace, but relatively easy to get from Peace to War.

It is assumed that there is a controller who blanks parts of the state vector after each step. Note that it would be essentially impossible to describe how this state vector was moving around a semantic network, though there is actually a strong hierarchical structure in the information initially stored. Of course this example is merely a demonstration of non-traditional retrieval techniques that make quite good sense in the context of using a distributed memory.

```
                          War → Peace

Wr    ____________________ → Wr   WeaponKillMaim $$$$$

Wr___Weapn_______________ → WrTnkWeapnGuns ArmorDangr

____________________Dangr → AnimlPnthrCarniBlackDangr

Animl     _______________ → Animl     Bird WingsFlyng

__________Bird _____Flyng → ____eDo___Bird W__g_Flyng (re-zero)

____eDo___Bird _____Flyng → PeaceDove Bird OliveFlyng


                          Peace → War

Peace____________________ → PeaceDove Bird OliveFlyng

_____     Bird___________ → Animl     Bird WingsFlyng

Animl____________________ → AnimlTigerCarniLargeDangr

_____Tiger_______________ → WrTnkTiger88mm.EffctGermn
```

We have used a 'generic' place holder (five blanks) in a couple of places.

We should point out that if we put in the sum of Peace and War, we can get two different results, depending on the exact codings:

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

```
Wr ____________________
          +              → PeaceDove Bird OliveFlyng
Peace _________________
```

But, alas

```
Wr     ________________
          +              → Wr   HumanSoldrFightBrave
Peace _________________
```

Acknowledgements

This work is sponsored by the National Science Foundation under grants BNS-82-14728 and BNS-83-15145 administered by the Memory and Cognitive Processes section. We are grateful to David Cooper for suggesting that even deterministic neural network models that minimize 'energy' might be reinterpreted within a probabilistic framework.

Appendix 1

In this section, the generality of the following probability density function is considered:

$$P(\underline{X}) = Z\,EXP[Q(\underline{X})] \qquad (1)$$

$$Q(\underline{X}) = a_0 + \Sigma a_i x_i + \Sigma\Sigma a_{ij} x_i x_j + \Sigma\Sigma\Sigma a_{ijk} x_i x_j x_k + \ldots a_{12\ldots n} x_1 x_2 \ldots x_n$$

where the summations are over all $x_i, x_j, \ldots x_k$ such that $1 \leq i < j < k \leq n$,

Z is a normalization constant, and the x_i are the elements of the vector $\underline{X}$.

We demonstrate that the alpha coefficients of the Q function in (1) may be selected in such a manner as to generate any arbitrary probability density function of binary-valued vectors. We also demonstrate that any set of alpha coefficients generates a valid probability density function. The proof of this statement is due to Besag (1974) and is restated here, in a more appropriate form, for convenience.

Let the function $P(\underline{X})$ be an arbitrary discrete probability density function of the 2^N possible N-dimensional vectors in the vector space. In fact, each possible value of $\underline{X}$ corresponds to a corner of an N-dimensional hypercube.

Following Besag (1974), the $Q(\underline{X})$ function is defined as follows:

$$Q(\underline{X}) = LN[P(\underline{X})/P(\underline{C})] \qquad (2)$$

where $\underline{C}$ is an arbitrary binary vector such that $P(\underline{C})$ is not equal to zero and LN[] denotes the natural logarithm. For $Q(\underline{X})$ to be defined, therefore, $P(\underline{C})$ must be greater than zero. Fortunately, however, since any binary vector in the space may be used this problem can always be resolved. For convenience, a vector $\underline{D}$ is now defined that corresponds to the corner of the hypercube that is directly opposite $\underline{C}$.

Let c_i be the ith element of the vector $\underline{C}$ and let d_i be the i<u>th</u> element of the vector $\underline{D}$. Now consider the following equation:

$$Q(\underline{X}) = b_0 + \Sigma b_i(x_i - c_i) + \Sigma\Sigma b_{ij}(x_i - c_i)(x_j - c_j) + \ldots b_{12..n}(x_1 - c_1)\ldots(x_n - c_n) \qquad (3)$$

where the summations are over all $x_i, x_j, \ldots x_k$ such that $1 \leq i < j < k \leq n$

and where $b_0 = Q(\underline{C})$. Also let

$$\underline{X}_i = (c_1, c_2, \ldots c_{i-1}, d_i, c_{i+1}, \ldots c_n)$$

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

$$\underline{X}_{ij} = (c_1, c_2, \ldots c_{i-1}, d_i, c_{i+1}, \ldots c_{j-1}, d_j, c_{j+1}, \ldots c_n).$$

Now note that the substitution of $\underline{X}_i$ into (3) yields

$$Q(\underline{X}_i) = b_0 + b_i(d_i - c_i)$$

which gives us the formula for b_i:

$$b_i = [Q(\underline{X}_i) - b_0]/(d_i - c_i)$$

Next, substitute $\underline{X}_{kl}$ (k not equal to l) into (3) to obtain

$$Q(\underline{X}_{kl}) = b_0 + \Sigma b_i(d_i - c_i) + b_{kl}(d_k - c_k)(d_l - c_l)$$

which gives us the following formula for b_{kl} (since b_0 and the set of b_i are now known).

$$b_{kl} = [Q(\underline{X}_{kl}) - b_0 - \Sigma b_i(d_i - c_i)]/[(d_k - c_k)(d_l - c_l)].$$

In a similar fashion, the remaining coefficients of the Q expansion in (3) may be calculated. Collecting the terms in (3) we see that the form of (3) is identical to (1).

To summarize, any arbitrary $P(\underline{X})$ can be represented in terms of $Q(\underline{X})$ using Equation 2, and any arbitrary $Q(\underline{X})$ may be represented in terms of a set of appropriate alpha coefficients. Moreover, the converse result holds. Any specific set of values for the alpha coefficients corresponds to a valid discrete probabilty density function $P(\underline{X})$. To see this, take the sample space as the set of all possible realizations of $\underline{X}$ such that $P(\underline{X}) > 0$, and note that $0 < P(\underline{X}) \leq 1$, $\Sigma P(\underline{X}) = 1$, and that any realization $\underline{X}$ may be considered to be an elementary event.

References

Ackley, D. A., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. Cognitive Science, 9, 147-169.

Anderson, J.A. (1970). Two models for memory organization using interacting traces. Mathematical Biosciences, 8, 137-160.

Anderson, J. A. (1983). Cognitive and psychological computation with neural models. IEEE Transactions on Systems, Man, and Cybernetics, 5, 799-815.

Anderson, J.A. (1986). In E. Bienenstock, F. Fogelmann, and G. Weisbuch (Eds.), Cognitive capabilities of a parallel system. Disordered Systems and Biological Organization. Berlin: Springer.

Anderson, J.A. & Murphy, G.L. (1986) Psychological concepts in a parallel System. In A. Lapedes, N. Packard, and B. Wendroff (Eds). Evolution, Games, and Learning. New York: North-Holland.

Anderson, J.A., Silverstein, J.W., Ritz, S.A., & Jones, R.S. (1977). Distinctive features, categorical perception, and probability learning: Some applications of a neural model. Psychological Review, 84, 413-451.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. Journal of the Royal Statistical Society, Series B, 36, 192-236.

Collins, A. & Quillian, M. (1969). Retrieval times from semantic memory. Journal of Verbal Learning and Verbal Behavior, 8, 241-248.

Duda, R. O., & Hart, P. E. (1973). Pattern classification and scene analysis. New York: John Wiley.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

Francis, W. N., & Kucera, H. (1982). Frequency analysis of English usage: Lexicon and grammar. Boston: Houghton Mifflin.

Garner, W. R. (1974). The Processing of Information and Structure. Potomac, MD: Erlbaum.

Golden, R. M. (1986a). The 'brain-state-in-a-box' neural model is a gradient descent algorithm. Journal of Mathematical Psychology. 30, 73-80.

Golden, R. M. (1986b). Modelling causal schemata in human memory: A connectionist approach. Ph.D. dissertation, Department of Psychology, Brown University, Providence, RI.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences USA, 79, 2554-2558.

Hopfield, J. J. (1984). Neurons with graded response have collective properties like those of two-state neurons. Proceedings of the National Academy of Sciences USA, 81, 3088-3092.

Kawamoto, A. (1985). Dynamic processes in the (re)solution of lexical ambiguity. Ph.D. dissertation, Department of Psychology. Brown University, Providence, RI.

Knapp, A.G. & Anderson, J.A. (1984). Theory of categorization based on distributed memory storage. Journal of Experimental Psychology: Learning, Memory, and Cognition., 10, 616-637.

Kohonen, T. (1977). Associative memory. Berlin: Springer.

Kohonen, T. (1984). Self organization and associative memory. Berlin: Springer.

Luenberger, D. G. (1979). Introduction to dynamic systems: Theory, models, and applications. New York: John Wiley & Sons.

Minsky, M., & Papert, S. (1969). Perceptrons: An introduction to computational geometry. Cambridge, MA: MIT Press.

Posner, M.I. & Keele, S.W. (1968). On the genesis of abstract ideas. Journal of Experimental Psychology, 77, 353-363.

Rosch, E. (1975). Cognitive representations of semantic categories. Journal of Experimental Psychology: General, 104, 192-233.

Rosch, E. (1978). Principles of categorization. In E. Rosch & B.B. Lloyd (Eds.), Cognition and categorization. Hillsdale, NJ: Erlbaum.

Smith, E. E., & Medin, D. L. (1981). Categories and concepts. Cambridge, MA: Harvard University Press.

Widrow, B. (1971). Adaptive filters. In R.E. Kalman & N. DeClaris (Eds.) Aspects of network and system theory. New York: Holt, Rinehart, and Winston.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Dec 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use