A More Biologically Plausible Learning Rule Than Backpropagation Applied to a Network Model of Cortical Area 7a

Pietro Mazzoni,[1,2] Richard A. Andersen,[1] and Michael I. Jordan[1]

[1] Department of Brain and Cognitive Sciences and [2] Harvard-MIT Division of Health Sciences and Technology, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Area 7a of the posterior parietal cortex of the primate brain is concerned with representing head-centered space by combining information about the retinal location of a visual stimulus and the position of the eyes in the orbits. An artificial neural network was previously trained to perform this coordinate transformation task using the backpropagation learning procedure, and units in its middle layer (the hidden units) developed properties very similar to those of area 7a neurons presumed to code for spatial location (Andersen and Zipser, 1988; Zipser and Andersen, 1988). We developed two neural networks with architecture similar to Zipser and Andersen's model and trained them to perform the same task using a more biologically plausible learning procedure than backpropagation. This procedure is a modification of the Associative Reward–Penalty (A_{R-P}) algorithm (Barto and Anandan, 1985), which adjusts connection strengths using a global reinforcement signal and local synaptic information. Our networks learn to perform the task successfully to any degree of accuracy and almost as quickly as with backpropagation, and the hidden units develop response properties very similar to those of area 7a neurons. In particular, the probability of firing of the hidden units in our networks varies with eye position in a roughly planar fashion, and their visual receptive fields are large and have complex surfaces. The synaptic strengths computed by the A_{R-P} algorithm are equivalent to and interchangeable with those computed by backpropagation. Our networks also perform the correct transformation on pairs of eye and retinal positions never encountered before. All of these findings are unaffected by the interposition of an extra layer of units between the hidden and output layers. These results show that the response properties of the hidden units of a layered network trained to perform coordinate transformations, and their similarity with those of area 7a neurons, are not a specific result of backpropagation training. The fact that they can be obtained by a more biologically plausible learning rule corroborates the validity of this neural network's computational algorithm as a plausible model of how area 7a may perform coordinate transformations.

An important element of information processing in the nervous system appears to be the collective behavior of large ensembles of neurons. The study of the emergent properties of these networks has been an important motivation behind the development of artificial neural network models whose architecture is inspired by the biological wiring of nervous systems, containing a large number of simple computational units extensively connected to one another. It is the hope of many neuroscientists that these models will elucidate, at least at an abstract level, some of the basic principles involved in information handling by the nervous system and thus perhaps provide a theoretical framework within which to formulate experimental questions.

One of the best examples of this type of approach so far is a neural network model of area 7a of the primate's posterior parietal cortex developed by Zipser and Andersen (1988; Andersen and Zipser, 1988). From lesion and single-cell recording studies in primates, it appears that area 7a is concerned with the representation of spatial locations in a head-centered reference frame (for a review, see Andersen, 1989). This representation is distributed over a group of neurons that are sensitive to both the position of the eyes in the orbits and the location of visual stimuli on the retinas. Other neurons in area 7a respond to either eye position or visual stimuli alone and are presumed to provide the inputs from which the visual/eye-position neurons extract the craniotopic representation. The latter neurons have very large retinotopic visual receptive fields, and their response to eye position interacts nonlinearly with the visual signals. Although the majority of area 7a neurons maintain the same retinotopic receptive fields for different eye positions, the magnitude of the visual response varies with angle of gaze. Holding the retinal location of a visual stimulus constant and varying eye position (Fig. 1*a*), Andersen and his colleagues found that these neurons' overall firing rate (visual plus eye position components) varied roughly linearly with changes in horizontal and vertical eye position (Fig. 1*b*). The response profiles for varying eye position were called "gain fields," and a majority (78%) of area 7a cells had planar or largely planar gain fields (Fig. 1*c*; Andersen et al., 1985; Andersen and Zipser, 1988; Zipser and Andersen, 1988).

Zipser and Andersen (1988) designed a computer-

Cerebral Cortex July/Aug 1991;1:293–307; 1047–3211/91/$4.00

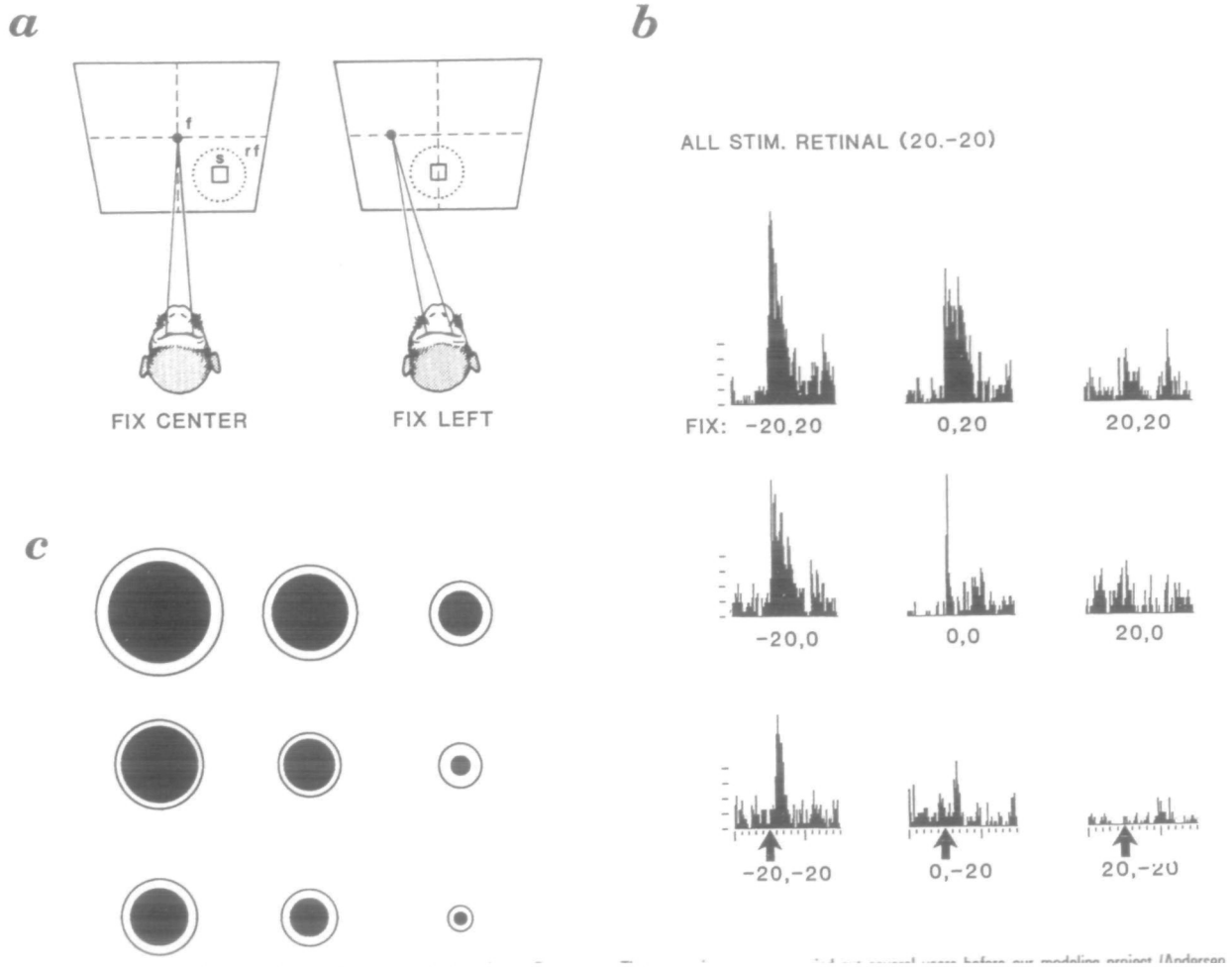

Figure 1. *a*, Experimental method of measuring spatial gain fields of area 7a neurons. These experiments were carried out several years before our modeling project (Andersen et al., 1985, 1987). The monkey faces a projection screen in total darkness and is trained to fixate on a point, *f*, at one of nine symmetrically placed locations on a projection screen with his head fixed. The stimulus, *s*, is always presented at the same retinal location, at the peak of the retinal receptive field, *rf*. The stimulus consists of 1°- or 6°-diameter bright spots flashed for 500 msec. *b*, Peristimulus time histograms of neuronal activity recorded from a particular area 7a neuron, arranged in the same relative positions as the corresponding fixation spots. The *arrows* indicate the time of visual stimulus onset. The characteristics of the response to the visual stimulus at the various angles of gaze constitutes the neuron's eye position gain field. Parts *a* and *b* were adapted from Andersen et al. (1985). *c*, graphic representation of the gain field in *b*, introduced by Zipser and Andersen (1988). The diameter of the *thin outer circle* is proportional to the total response evoked by the stimulus. The width of each *annulus* represents the contribution to the total response due to eye position alone and is measured as the background activity recorded 500 msec before the stimulus onset. The *dark inner circle* represents the visual contribution to the response, and its diameter is computed by subtracting the background activity from the total response. This representation shows that this neuron's gain field is roughly planar in a direction up and to the left.

simulated neural network with an input layer, a layer of internal or hidden units, and an output layer. The input layer consisted of two groups of units with properties similar to those of area 7a neurons sensitive to either eye position or visual stimulus alone. The output layer coded for head-centered location in an abstract format independent of eye position and was used to generate error signals to train the network. The network was trained to perform the coordinate transformation from retinotopic to craniotopic reference frames using the backpropagation procedure (Rumelhart et al., 1986a). The striking result of these simulations was that in the process of learning the hidden units developed response properties very similar to those of the area 7a neurons that seem to encode spatial location—specifically, a roughly planar modulation of visual response by eye position, and large complex receptive fields. This result suggested that area 7a neurons, as an ensemble, can in fact provide information for the abstract representation of space, and that these neurons' properties can be generated by a supervised learning paradigm.

Backpropagation networks can learn to perform a computation without explicit "knowledge," using only error signals from the environment as cues to improve its performance, in a paradigm referred to as "supervised learning" (Hinton, 1987). This type of training scheme has conferred upon neural networks a stronger element of biological plausibility than many previous models of brain function that relied on precompiled rules and symbol processing. Moreover, although various supervised learning algorithms for one-layer networks were described long before backpropagation (Widrow and Hoff, 1960; Minsky and Papert, 1969), the discovery of the backpropagation algorithm (Werbos, 1974; Parker, 1985; Rumelhart et al., 1986a), which can be applied to more powerful multilayer networks composed of nonlinear units, is in large part responsible for the recent increase in interest in neural network models. Despite the bio-

logical plausibility of supervised learning, however, the implementation of backpropagation in the nervous system would require mechanisms such as the retrograde propagation of signals along axons and through synapses and precise error signals that are different for each neuron, which are not accepted as likely candidates for learning processes in the brain. To this end, Zipser and Andersen (1988) emphasized that it was the solution that was of interest in their model and not the method by which this solution was learned. They speculated that other, more biological learning procedures might generate a solution to the coordinate transformation task similar to that which resulted from backpropagation learning. It was therefore important to ask how crucial backpropagation is for the development of the hidden units' properties in a model like Zipser and Andersen's.

We addressed precisely this question in our study. We trained two neural networks with architectures similar to the Zipser and Andersen model using a supervised learning paradigm that is more plausible from a biological perspective than backpropagation. The algorithm we used, which is a variant of the Associative Reward–Penalty ($A_{R\text{-}P}$) algorithm for supervised learning tasks introduced by Barto and Jordan (1987), trains a neural network using a global reinforcement signal broadcast to all the connections in the network. We found that our networks can indeed be trained by these algorithms to perform the coordinate transformation task, and that the hidden units acquire response properties very similar to those of area 7a neurons, as in the Zipser and Andersen model. A second layer of hidden units can be interposed between the original hidden layer and the output layer without affecting the properties developed by units in the first hidden layer. Furthermore, all of our networks perform the correct transformation on pairs of eye and retinal position never encountered before; that is, they generalize appropriately. A less detailed report of results from one of the $A_{R\text{-}P}$ networks has recently been published (Mazzoni et al., 1991).

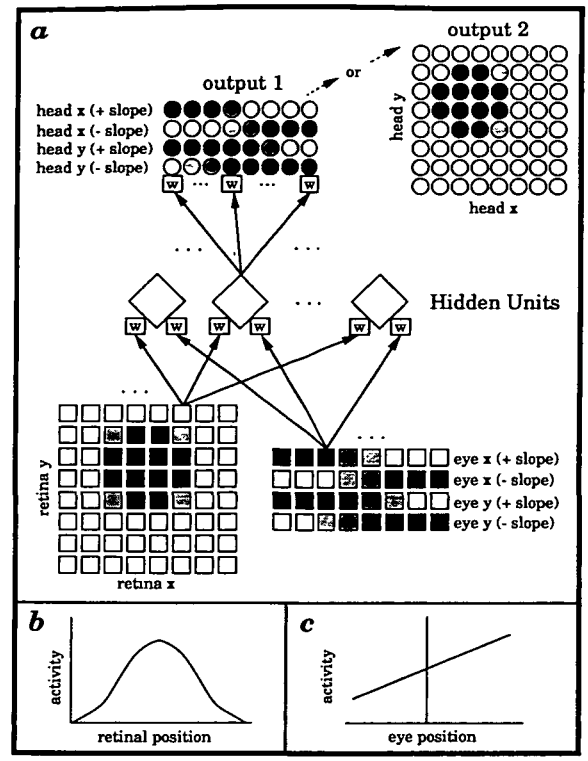

Figure 2. *a*, Structure of the Mixed $A_{R\text{-}P}$ network. The network is composed of three layers of computing units: input units (encoding retinal location of stimulus and eye position), hidden units, and output units (encoding head-centered location of the visual stimulus). Retinal position of the visual stimulus is encoded topographically by an 8 × 8 array of input units, each with a gaussian receptive field (described in *b*). The remaining 32 input units code for eye position in a linear fashion (see *c*), with two groups of eight units encoding horizontal gaze angle (with positive and negative slopes), and two groups of eight units for vertical angle. Units in the output layer code for head-centered coordinates in a monotonic format (*output 1*) similar to the eye position input, or in a gaussian format (*output 2*) similar to the retinal input. The hidden units are binary stochastic elements, while the output units are deterministic logistic elements (see Fig. 3). Shading is proportional to unit activity in the input and output layers. Connection weights are indicated by *w*. *b*, Angle-coding function of the retinal input units. Each unit's activity level is a gaussian function of the retinotopic *x*- and *y*-coordinates of the visual stimulus, one $1/e$ width of 15°, and spaced 10° apart from that of its neighboring units. *c*, Angle-coding function for the eye position units. Each unit codes for the *x* or *y* eye position linearly. The slopes and intercepts for each unit were assigned randomly within ranges observed for area 7a neurons.

Materials and Methods

Model Networks

We devised two types of networks that we trained to map visual targets to head-centered coordinates, given any arbitrary pair of eye and retinal positions. The basic architecture of these networks is similar to that of Zipser and Andersen's model.

Mixed $A_{R\text{-}P}$ Network

We call the first network the *Mixed $A_{R\text{-}P}$ network* (Fig. 2*a*) because its hidden and output layers are trained by different learning rules (see Training, below). It is composed of three layers of computing units: an input, a hidden, and an output layer. The network has a fully connected feedforward architecture, meaning that every unit in each layer sends a signal to every unit in the next layer through an individual connection strength or weight (w), so that signals propagate in one direction from the input toward the output layer. The input layer consists of two groups of units (Fig. 2*a*, squares), one coding for the retinal location of the visual stimulus, and the other for the position of the eyes in the orbits. These units encode the external input by transforming an angular position into a value between 0 and 1, which is then sent to the hidden units. Retinotopic locations are represented by 64 visual units arranged in an 8 × 8 array, each with a gaussian receptive field (Fig. 2*b*) with peak at 10° from its neighbors' and $1/e$ width of 15°, producing a uniform topographic representation of the retina. Eye position is coded by four sets of eight units representing horizontal and vertical eye coordinates with positive and negative slopes, for which activation is a linear function of eye angle (Fig. 2*c*). These representation formats were modeled according to characteristics of area 7a neurons established in previous studies (Andersen et al., 1985; Andersen and Zipser, 1988; Zipser and Andersen, 1988) and are the same

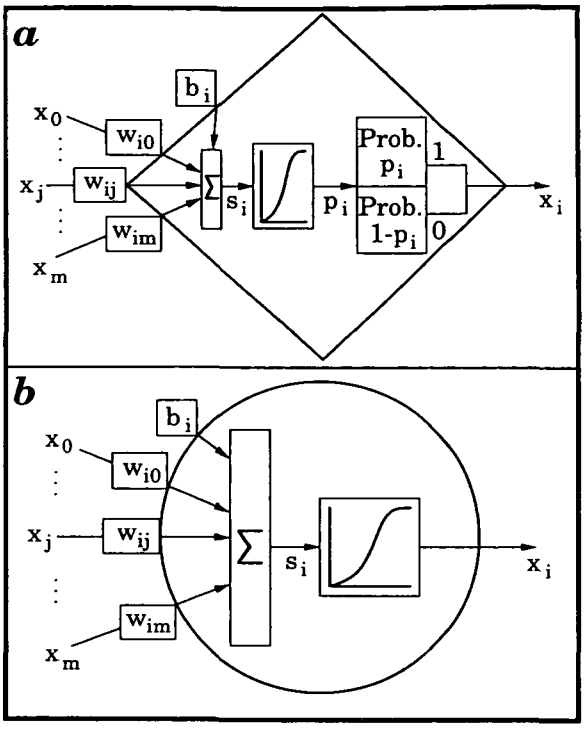

Figure 3. *a*, Binary stochastic element. This neurally inspired computing element performs a weighted sum of its synaptic inputs (x_0 through x_m) by multiplying each input by a synaptic weight (w_0 through w_m), which can be positive or negative, and adding these products. This sum (s) is used to compute a value (p) between 0 and 1 from the logistic function $[1/(1 + \exp(-s))]$. The element then produces an output of 1 with probability p, and an output of 0 with probability $1 - p$. We used this element in the hidden layers of all our networks, as well as in the output layer of the All $A_{R\text{-}P}$ network. *b*, Deterministic logistic element. This unit computes a weighted sum of its input in the same manner as the binary stochastic element. This sum is also passed through the logistic function, but the resulting value (x) is the unit's output itself. The unit therefore produces a continuous output between 0 and 1, which is determined exactly by the weighted sum of its inputs. This is the element that was used in the hidden and output layers of the Zipser and Andersen model. We used it only in the output layer of the Mixed $A_{R\text{-}P}$ network.

as those used in the input layer of the Zipser and Andersen model.

The hidden layer (Fig. 2*a*, diamonds) is so described because it is not "visible" (i.e., directly connected) to external agents acting at the input or at the output. The type of computational unit making up this layer is the *binary stochastic element* (Fig. 3*a*). This probabilistic element performs a weighted sum (s) of its inputs and passes it through the logistic function[1] to obtain a value (p) between 0 and 1. This value is then used as the probability of producing an output equal to 1. The output is 0 with probability $1 - p$. This type of computing element is "neurally inspired," in the sense that it incorporates some well-established features of neurons. In such an analogy, the inputs correspond to synaptic inputs from other neurons, the connection weights represent synaptic strengths (with inhibitory synapses implemented as negative weights), and the weighted input represents the intracellular potential. The probability of firing is analogous to a neuron's rate of action potential firing, and changes in the total weighted input affect the unit's probability of firing in a manner similar to how changes in the intracellular potential affect a neuron's firing rate. This hidden layer differs from that of the Zipser and Andersen network in that the units of the latter were deterministic logistic elements (described below). The number of hidden units in the Mixed $A_{R\text{-}P}$ network, as well as in all the networks described below, varied from two to eight.

The hidden units project in turn to the output layer, which encodes the craniotopic location that is the vector sum of the positions encoded by the retinal and eye position inputs. The units in the output layer (Fig. 2*a*, circles) are *deterministic logistic elements* (Fig. 3*b*). Like the binary stochastic units, they too perform a weighted sum of their inputs and pass it through the logistic function. In the deterministic logistic element, however, this value between 0 and 1 is the unit's output itself. The output, therefore, is continuous and precisely predictable from the input. In the analogy with the neuron, this continuous output would correspond to the firing rate. The outputs of the output units encode head-centered location in one of two possible formats: a "monotonic" representation analogous to the eye position input, containing any number of units from 2 to 32 (Fig. 2*a*, output 1), and a "gaussian" representation similar to that of the retinal input, with a number of units ranging from 4 to 64 (Fig. 2*a*, output 2). In the monotonic representation, the activity of the output units increases for more peripheral locations of the visual target with respect to the head, regardless of eye position. The gaussian format units fire for visual stimuli appearing within limited receptive fields in head-centered coordinates. We used either representation interchangeably, as this did not seem to affect our results. A physiological correlate of the monotonic representation could be a motor signal to the extrinsic eye muscles (Zipser and Andersen, 1988; Goodman and Andersen, 1989), while the gaussian format would be more like a receptive field for a mental representation of craniotopic space. These output representations are similar to those of the Zipser and Andersen model.

All $A_{R\text{-}P}$ Network

The second type of network we studied is the *All $A_{R\text{-}P}$ network* (Fig. 4*a*), so-called because all of its connections are adjusted by the $A_{R\text{-}P}$ algorithm (see Training, below). This network's input and hidden layers are identical to those of the Mixed $A_{R\text{-}P}$ network. The output layer, however, is composed of binary stochastic units like the hidden layer. It, too, encodes craniotopic location in one of two alternative formats. Due to the binary nature of the output units, we devised output formats for the All $A_{R\text{-}P}$ network such that the collective activity of the output units codes for discrete adjacent regions of space, instead of continuously varying spatial locations. In the "binary-monotonic" format, four triplets of units divide craniotopic space into 16 regions by giving an output of 1 if the x (or y) head-centered coordinate is greater (or less) than −40°, 0°, or +40° (Fig. 4*b*). This format is analogous to the monotonic format of the Mixed $A_{R\text{-}P}$ network. The binary counterpart of the continuous gaussian output is the "binary-gaussian" format, in which four units have overlapping receptive fields centered

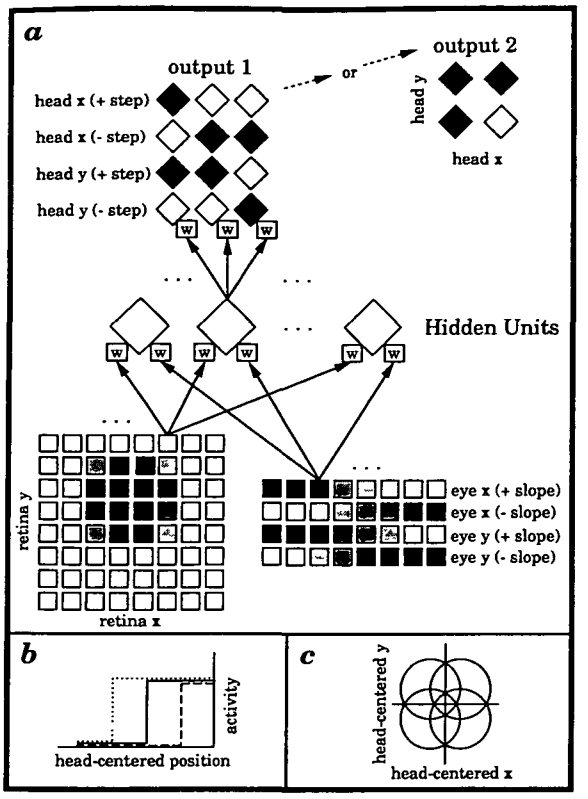

Figure 4. *a*, Structure of the All $A_{R\text{-}P}$ network. The input and hidden layers are the same as for the Mixed $A_{R\text{-}P}$ network (Fig. 2). The output layer is composed of binary stochastic units (Fig. 3*a*). These code for locations in craniotopic space by dividing the latter into discrete regions according to one of two formats (*output 1* and *output 2*) described in *b* and *c*. *b*, Binary-monotonic format for the All $A_{R\text{-}P}$ network. Each unit transforms an output value between 0 and 1 into an angle via a step function. An output of 0 corresponds to all angles less (positive step) or greater (negative step) than a given "cutoff" angle, and an output of 1 codes for all angles greater or less, respectively, than the cutoff angle. We use step here to indicate the direction along which the step function changes from 0 to 1. Typically there are four sets of three units each, for horizontal and vertical coding with positive and negative step. Within each set, the cutoff angles for the three units are −40°, 0°, and 40°. Only the functions for one positive-step triplet of units are plotted. *c*, Binary-gaussian format. Four units divide craniotopic space into 13 regions using overlapping circular binary receptive fields. Each unit outputs a 1 for a position within a 100°-radius circle centered at one of the four positions (±60°, ±60°).

at (±60°, ±60°), such that each unit's output is one if the spatial position is within 100° of its center (Fig. 4*c*). This format divides craniotopic space into 13 regions by virtue of the overlap of the output units' receptive fields. The number of units in both types of binary output format may be increased to improve the output's spatial resolution. We did not examine the parameter of number of output units systematically, as it did not produce significantly different network behaviors.

Other Networks

In addition to the two three-layer networks just described, we studied the behavior of two similar four-layer networks. These consist of a Mixed $A_{R\text{-}P}$ network and an All $A_{R\text{-}P}$ network, each with an extra layer of hidden units between the first layer of hidden units and the output layer. This extra layer, like the original hidden layer, is also composed of binary stochastic units. We did this to see whether the response properties developed by the units in the hidden layer of

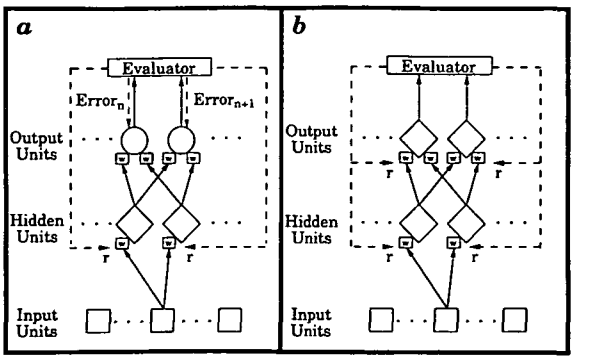

Figure 5. *a*, Learning scheme for the Mixed $A_{R\text{-}P}$ network. Training proceeds in two phases that are repeated sequentially. First, a pair of retinal and eye positions is presented at the input layer. The signal propagates forward (*solid arrows*) to the network's upper layers along connections strengths that initially have random or 0 values. The network's output is evaluated by some external agent, and two types of signals are fed back to the network (*broken arrows*). One is a vector error signal that consists of the differences between the actual and desired outputs for the output units, and is sent to the output units. The other is a scalar payoff signal (r) between 0 and 1 that is sent to the hidden units. In the second phase, the connection weights are adjusted using the error and payoff signals. The output units adjust their weights according to the delta rule, while the hidden units adjust them by the S'-model $A_{R\text{-}P}$ learning rule. The backpropagation network used for reference was trained by the standard backpropagation algorithm described by Rumelhart et al. (1986a). *b*, Learning scheme for the All$_{R\text{-}P}$ network. This is identical to Mixed $A_{R\text{-}P}$ learning described in *a*, except that the scalar payoff signal r is broadcast to all the units in the hidden and output layers, and all the weights are adjusted by the S'-model $A_{R\text{-}P}$ rule. The network therefore employs only the scalar payoff signal for feedback information on its performance, and no error vector is required.

the three-layer networks depended on a direct connection with the output layer.

For comparison purposes, we also set up a backpropagation network identical to the Mixed $A_{R\text{-}P}$ network described in Figure 2*a*, except for its hidden units, which are deterministic logistic elements and not binary stochastic elements.

Training

We trained our networks to perform the coordinate transformation task in a supervised learning paradigm. In supervised learning, the network starts out with all connections and biases set at 0, or at some set of small random values if the training algorithm cannot break the initial symmetry (the $A_{R\text{-}P}$ algorithm we used can handle both cases). An input pattern is presented to the network's input layer, which propagates a signal to the following layers (Fig. 5, solid arrows). The output layer thus produces a "guessed" output based on the initial set of connections. This output is compared to the correct output pattern for that particular input, and an error is computed and fed back to the network (Fig. 5, broken arrows). The values of all the network's weights and biases are then adjusted by a learning rule so that at the next presentation of the same pattern the error is, at least on the average, smaller than before. This procedure is repeated until the error is reduced to a value below a desired level.

For our task, the input pattern is a signal for the retinal location of a visual stimulus paired with one for the current eye position. The desired output pattern is one that codes for a head-centered location

that is the vector sum of the retinal and eye positions. The error signal is computed externally to the network. To draw an analogy with how an animal may learn the coordinate transformation task, the input pattern would correspond to a visual stimulus seen with the eyes at a known angle of gaze (sensed by proprioceptive or corollary discharge pathways). The animal may then make a movement toward the stimulus, and any metric of performance, such as visually detected inaccuracies, could be used to generate an error signal.

The network is trained by being repeatedly presented with a finite number of patterns forming a chosen training set, the connection weights being adjusted after each pattern presentation. We used two types of pattern sets to train the networks. One is a set of random pairs of retinal locations and eye positions so that the desired output associated with each input is an arbitrary location in head-centered space. In the analogy with the learning animal, learning with this set would correspond to looking at various stimuli in the visual field at various angles of gaze. The other type of training set consists of input patterns for which the eye position is chosen randomly, while the retinal location is computed so that the vector sum of the two inputs is one of a few chosen craniotopic locations. The resulting training set contains a few fixed spatial locations, each represented by a large number (at least 10) of retinal and eye positions that add vectorially to it. For an animal, this type of training corresponds to looking at an object fixed in space with the eyes in various orbital orientations. This training set was used to see how well the network generalized to new locations in space once it had been trained on a few fixed ones.

We devised two variants of the supervised learning procedure for $A_{R\text{-}P}$ networks of Barto and Jordan (1987) to adjust the weights of our networks. The essence of this algorithm is the $A_{R\text{-}P}$ learning rule. Every binary stochastic element in a given network receives a scalar *reinforcement* (or payoff) signal, r, whose value, in the supervised learning paradigm, depends on how close the current output is to the desired output. Specifically, r assumes a value between 0 and 1, with 0 indicating maximum error in the output (i.e., every unit that should be firing is not, and vice versa), and 1 corresponding to optimal performance (no error in the computed head-centered position). The weights of the input connections on each binary stochastic element are then adjusted in such a way as to maximize the value of this payoff. Using the notation of Figure 3*a*, where x_i represents the output of the ith unit in the network, p_i its probability of firing, and w_{ij} the connection weight for its input from the jth unit, the equation for updating the weights on a binary stochastic unit is

$$\Delta w_{ij} = \rho r(x_i - p_i)x_j + \lambda\rho(1 - r)(1 - x_i - p_i)x_j, \quad (1)$$

where Δw_{ij} denotes the change in the value of the connection strength w_{ij} after each pattern presentation, and ρ and λ are constant parameters representing the learning rate. As shown in Figure 3*a*, each unit also has a constant input, or bias (b_i). The value of this bias is also adjusted by the rule in Equation 1, setting $x_j = 1$. Typical values for the parameters in this equation were 0.3 for ρ and 0.01 for λ. We will describe this equation in more detail in the Discussion. The value of r is computed, externally to the network, as

$$r = 1 - \epsilon, \quad (2)$$

where ϵ is a measure of the current error at the output layer. In our model, ϵ is computed as the nth root of the absolute value of the output units' error averaged over the number of output units:

$$\epsilon = \left\{\frac{1}{K}\sum_{k=1}^{K}|x_k^* - x_k|\right\}^{1/n}, \quad (3)$$

where k indexes the K output units in the network, x_k^* is the desired output of the kth unit in the output layer, x_k is its actual output, and n is a constant. Values for n ranged from 2 to 6. This expression for ϵ is different from the one used by Barto and Jordan (1987), who computed ϵ as the sum of the squares of the output units' errors. Both expressions give a quantity nonlinearly related to the absolute value of the output units' errors, but our expression is more sensitive to small errors (a given unit's absolute error, $|x_k^* - x_k|$ is always less than or equal to 1). Barto and Jordan referred to their learning algorithm as the "S-model $A_{R\,P}$ rule," borrowing terminology from learning automata theory. In order to distinguish our modification of this rule from the original one, we refer to our training algorithm as the *S′-model $A_{R\text{-}P}$ rule*.

We used the S′-model $A_{R\text{-}P}$ rule to adjust the weights of all the binary stochastic units in our networks. This includes all the weights in the All $A_{R\text{-}P}$ networks and the weights between the input and hidden layers of the Mixed $A_{R\,P}$ network. The output units of the Mixed $A_{R\text{-}P}$ network, being deterministic units with continuous output, were trained by the delta rule for output units (Rumelhart et al., 1986a). This rule adjusts the weights of each output unit according to

$$\Delta w_{ij} = \alpha[(x_k^* - x_k)f'(s_k)]x_j, \quad (4)$$

where x_k indicates the kth output unit, x_k^* is the desired output of the kth unit, α is a scalar learning rate, f' is the derivative of the logistic function with respect to the unit's net input s_k, and x_j is the output of one of the hidden units "presynaptic" to it. Typical values for α were between 0.5 and 2.5. This learning rule also is the basis for the backpropagation algorithm and is used in identical form to train the output units of backpropagation networks.

Results

Learning

All the networks described above learned to perform the coordinate transformation task to any desired accuracy. Figure 6 shows the general behavior during training of the two $A_{R\,P}$ networks studied and compares them to that of a backpropagation network, with

identical architecture, learning from the same training set. We plot here the absolute value of the output units' error, averaged over the number of output units and the number of patterns in the training set, versus the number of presentations of the complete training set. The $A_{R\text{-}P}$ networks produce learning curves with much more jitter than the curve for backpropagation training, due to the stochastic nature of their hidden units and to the type of error signal used in $A_{R\text{-}P}$ training (see Discussion). All three networks, however, produce curves with similar envelopes, and the times required for convergence are comparable. For the backpropagation network, which has a continuous output, the error decreases monotonically (Fig. 6*a*), while for the All $A_{R\text{-}P}$ network, which has a binary output, the error follows a noisy path down to 0 and spends increasingly more time there, flickering occasionally to the value of the output's smallest resolvable angle (Fig. 6*c*). The error for the Mixed $A_{R\text{-}P}$ network is also noisy, because this network's hidden units are stochastic. It assumes, however, a continuous range of values (Fig. 6*b*), because the output units are logistic elements. Similar training curves were obtained for both monotonic and gaussian formats. Neither algorithm had serious problems with local minima (i.e., getting stuck at suboptimal solutions).[2]

Response Properties

We studied the response properties developed by the hidden units during training in the same manner as Zipser and Andersen did for their model, except we plot the units' probability of firing (which is a continuous variable) instead of its instantaneous output (which is binary). The probability of firing can be thought of as equivalent to the firing rate and thus equivalent to the continuous output in the Zipser and Andersen model. In other words, over a number of repeated presentations of a given input pattern, the frequency with which a binary unit's output is 1 encodes a continuous value, which can be conceived as a firing rate.

An interesting feature of area 7a neurons is that the visual and the eye position contributions to their overall response interact nonlinearly. For a constant retinal stimulus position, the total response is not composed of a constant visual response to which an independently varying amount of activity is added as the eye position changes. As Figure 7*a* and the work of Andersen and Zipser (1988; Zipser and Andersen, 1988) show, the visual and eye position components can vary simultaneously, in either the same or opposite directions, or to different degrees with eye position (see Andersen and Zipser, 1988, for a more detailed analysis of area 7a gain fields). When examined after training, the hidden units of both types of $A_{R\text{-}P}$ networks displayed gain fields similar to those of area 7a neurons (Fib. 7*b,c*), as well as the same type of variety. The overall response of the hidden units, moreover (Fig. 7, thin circles), was always roughly planar along vertical and horizontal eye positions. This result was found in 78% of spatially tuned area 7a neurons (Andersen and Zipser, 1988). When

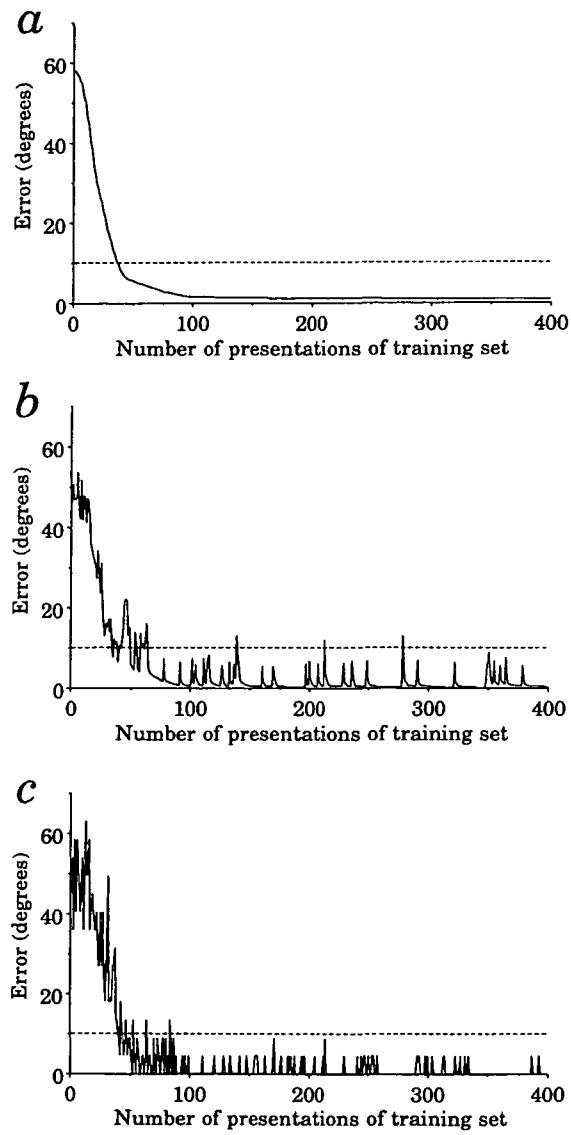

Figure 6. Learning curves for the various networks studied. The error plotted is the absolute value of the difference between the network's expected and actual output, averaged over the units in the output layer and over the patterns in the training set. This average error corresponds approximately to the radial distance between the craniotopic location encoded by the output layer and the correct one (given by the sum of the retinal and eye position vectors). The *broken line* is a scaling reference of 10° corresponding roughly to the resolution of the visual input. A three-layer network architecture with three hidden units was used in *a–c*. The training set consisted of 12 random inputs coding for four spatial locations. A two-unit monotonic output format was used, which provided for easy conversion of the error values from unit activation levels to angular coordinates of craniotopic space. The training set was chosen small for better visualization of network behavior. The error for the binary output units of the All $A_{R\text{-}P}$ network was converted to degrees using the same linear activation function as for the other two networks. *a*, Backpropagation training; *b*, Mixed $A_{R\text{-}P}$ training; *c*, All $A_{R\text{-}P}$ training.

a second hidden layer of binary stochastic units was added to either the Mixed $A_{R\text{-}P}$ or All $A_{R\text{-}P}$ network, both networks learned to perform the task, and the units in the first hidden layer still developed planar gain fields similar to those of area 7a (Fig. 7*d*; only All $A_{R\text{-}P}$ shown).

It is worth noting that when studied in more detail, that is, when sampled over a larger range of eye positions, the gain fields produced by $A_{R\text{-}P}$ (as well as backpropagation) training are not exactly planar, but

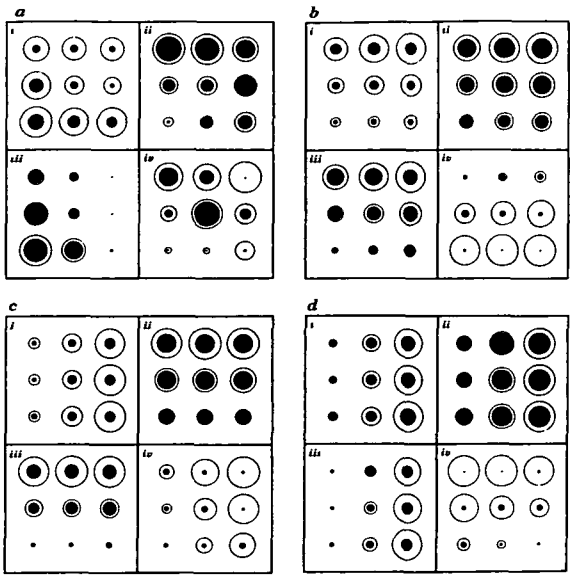

Figure 7. Eye position gain fields for area 7a neurons and for the model networks' hidden units. Gain fields for four different neurons (*a*) or hidden units (*b–d*) are shown in each case. The *nine circles* in each *box* are a set of responses sampled at nine different eye positions, with the retinal location of the stimulus held constant. As described in Figure 1, the *thin outer circles* represent the total activity (normalized), the *dark inner circles* are proportional to the contribution of the visual stimulus to the total response, and the *white annuli* are the background activity due to eye position. By "activity" we mean frequency of firing for area 7a neurons and probability of firing for the networks' hidden units. The spacing between eye positions is 20° for area 7a neurons. It is 20° for all the networks' hidden units, except for the two gain fields in the bottom left of *c* and *d*, for which the spacing is 40°. *a*, Area 7a neurons; *b*, Mixed $A_{R,P}$ network; *c*, All $A_{R,P}$ network; *d*, All $A_{R,P}$ network with two layers of hidden units.

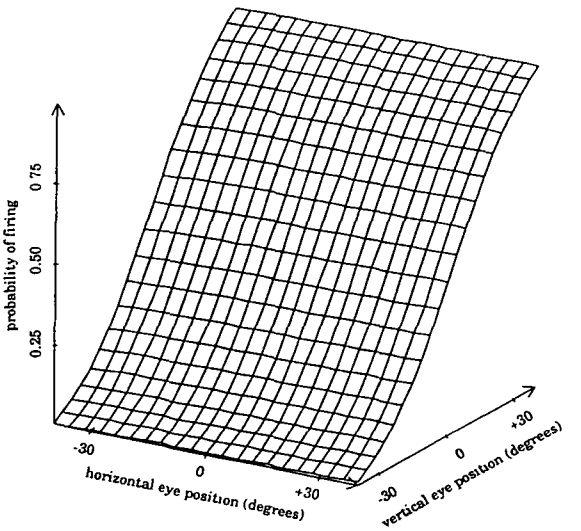

Figure 8. Probability of firing (total response alone) of a hidden unit from a Mixed $A_{R,P}$ network, sampled over a continuous range of eye positions. The gain field is planar over a wide range of eye positions. Note the slight saturation effects (flattening) at the edges of the eye position field.

roughly sigmoidal along one direction of eye position (Fig. 8). In other words, the overall responses are approximately planar within a range of eye positions and are flat outside this range. It turns out that this range is determined by the most eccentric eye positions on which the network was trained. The unit whose gain field is shown in Figure 8, for example, was part of a network trained with eye positions between −40° and 40° (horizontally as well as vertically), and it developed a gain field approximately planar over this range along the x direction (there were other units in the network with similar gain fields oriented along the y direction). This result shows that the hidden units learn to interpolate for eye positions between those in the training set, but they do not learn to extrapolate to eye positions outside this range. We believe that this is a direct consequence of using a sigmoidal probability function (or input–output function in the case of the deterministic logistic element) for the hidden units. The gain fields of area 7a neurons may also flatten outside a certain range of eye positions, producing a sigmoidal shape like that in Figure 8. At present, the recording data available are too noisy, and perhaps too limited in range of eye positions, to distinguish between a simple plane and a sigmoid for the gain fields.

There was also a qualitative similarity between the visual receptive fields developed by the network's hidden units and those of area 7a neurons, as shown in Figure 9. The most striking feature of these neurons' receptive fields is their size, which extends to diameters of 80° (Fig. 9*a*). This feature is reproduced by our model networks (Fig. 9*b,c*). Another feature is the complexity of these receptive fields' surfaces, characterized by one or more smooth peaks of various eccentricities, which sets area 7a neurons apart from those of many other visual areas. The networks' hidden units also display a similar complexity in their receptive fields, although such a comparison can be qualitative at best. As was the case for the gain fields, the addition of an extra hidden layer did not significantly affect the types of receptive fields developed by units in the first hidden layer (Fig. 9*d*; only All $A_{R,P}$ shown).

The response properties of the $A_{R,P}$ networks' hidden units are similar not only to those of area 7a neurons, but also to those of hidden units of networks trained by backpropagation to compute coordinate transformations. These response properties were described by Zipser and Andersen (1988; Andersen and Zipser, 1988). We were able to reproduce them also in a backpropagation network with the same number of units and the same training set as the $A_{R,P}$ networks (Fig. 10). This similarity suggests that the S′-model $A_{R,P}$ rule and backpropagation compute similar solutions (i.e., sets of network connection strengths) to the coordinate transformation problem.

Comparison of Solutions

The solutions found by S′-model $A_{R,P}$ training and by backpropagation are not just similar in the qualitative sense depicted in Figures 7, 9, and 10. In fact, for a given training set, we found that the set of weights trained by the $A_{R,P}$ algorithm may be transferred to a backpropagation network (with continuous output hidden units) without any appreciable reduction in the accuracy of the network's response to that training pattern, and vice versa (Fig. 11). This is true for both

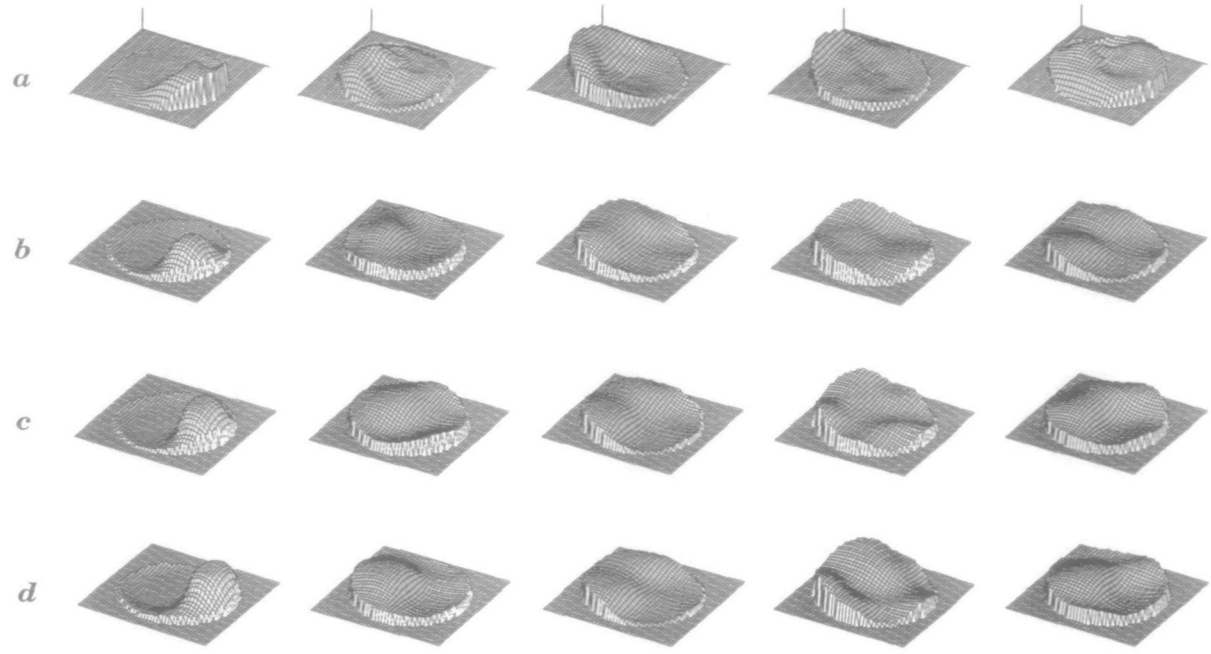

Figure 9. Visual receptive fields of area 7a neurons and networks' hidden units. As in Figure 7, the variables sampled are firing rate for area 7a neurons (*a*) and probability of firing for the hidden units (*b–d*). *a*, Area 7a neurons' receptive fields. Each was sampled at seventeen points in a 40°-radius circle, and a smooth surface was obtained by gaussian interpolation (adapted from Zipser and Andersen, 1988). *b*, Mixed $A_{R\text{-}P}$ network. *c*, All $A_{R\text{-}P}$ network. *d*, All $A_{R\text{-}P}$ network with two layers of hidden units.

versions of our networks (Mixed and All $A_{R\text{-}P}$) and for networks with one and two hidden layers, as long as the output format is the same for the $A_{R\text{-}P}$ and backpropagation networks being compared. The individual values of the weights are *not* the same after training by the three different procedures, but the overall structure of these weights is such that the two algorithms' solutions to the coordinate transformation problem are functionally equivalent for the various network structures.

Generalization

A property that is often exhibited by artificial neural networks trained by examples is the ability to generalize from those examples, that is, to produce the correct output when presented with input patterns that were not in the training set. This property is of great theoretical and practical importance, as it demonstrates that the model network has not merely learned to associate patterns in the training set with their correct outputs on an individual basis, but has

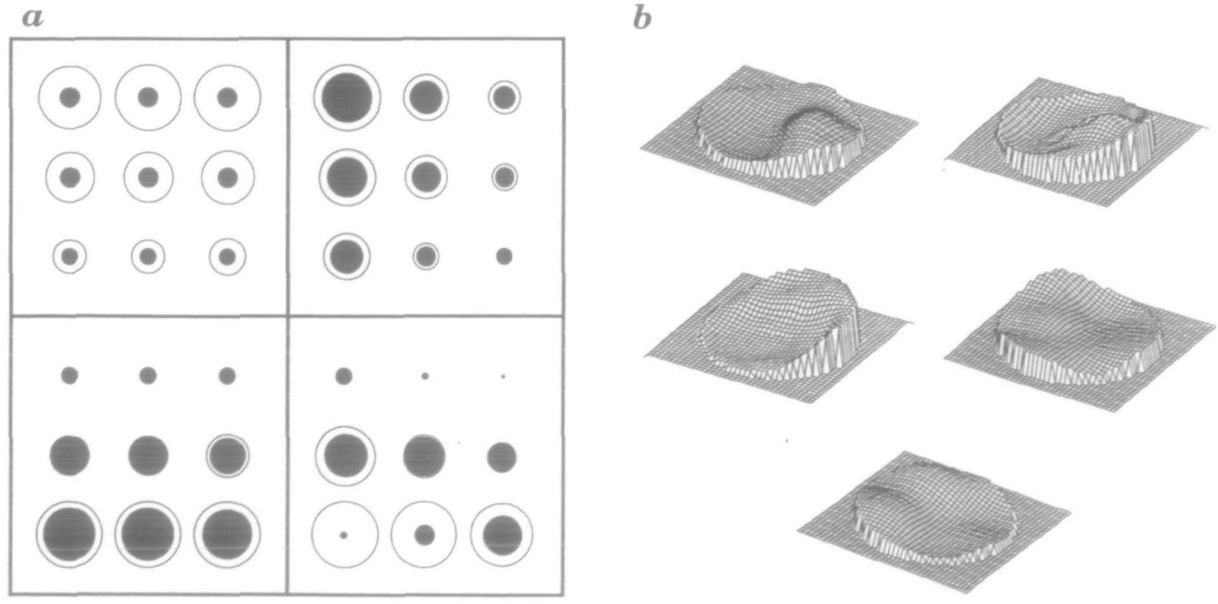

Figure 10. Response properties of hidden units in a backpropagation-trained network. Shown are representative gain fields (*a*) and the receptive fields (*b*) recorded from the hidden units of a backpropagation network after it was trained to perform the coordinate transformation. Note the similarity between these response properties and those of the hidden units of $A_{R\text{-}P}$-trained networks (Figs. 7, 9). Two of the receptive fields in *b* (*top right* and *middle left*) are adapted from Zipser and Andersen (1988).

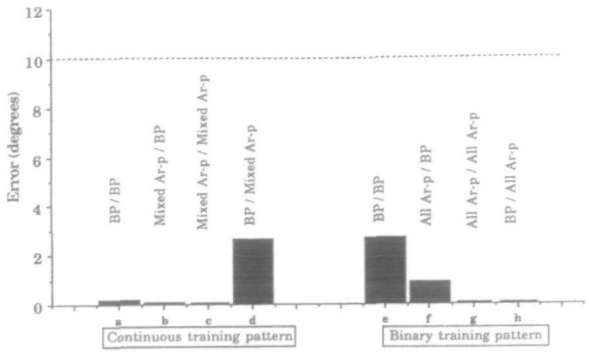

Figure 11. Error values (as defined for Fig. 6) produced by various networks when trained sets of connection strengths were swapped among them. Three networks [backpropagation (*BP*), *Mixed Ar-p*, and *All Ar-p*] were first trained to a given accuracy. The trained weights were then exchanged among the different networks, and the error upon presentation of the training set was recorded. The value of 10° is again used as a scale reference (see Fig. 6). The average error for all untrained networks was around 60°. The error values do not show a significant change when backpropagation training is replaced by Mixed or All $A_{R,P}$ training, and vice versa, showing that the solutions found by the different algorithms are functionally equivalent. *a*, Backpropagation-trained weights tested on backpropagation network; *b*, Mixed $A_{R,P}$-trained weights tested on backpropagation network; *c*, Mixed $A_{R,P}$-trained weights tested on Mixed $A_{R,P}$ network; *d*, Backpropagation-trained weights tested on Mixed $A_{R,P}$ network; *e*, Backpropagation-trained weights tested on backpropagation network (binary output format of All $A_{R,P}$ network used in the training set); *f*, All $A_{R,P}$-trained weights tested on backpropagation network; *g*, All $A_{R,P}$-trained weights tested on All $A_{R,P}$ network; *h*, Backpropagation-trained weights tested on All $A_{R,P}$ network.

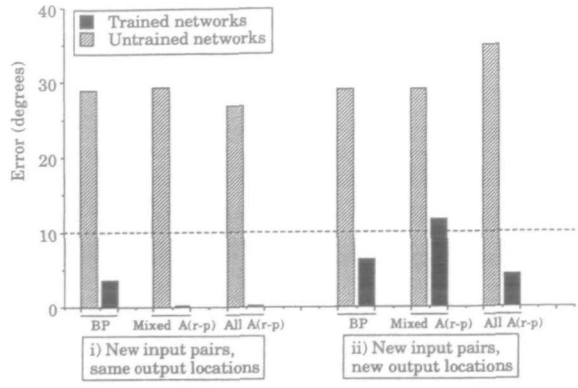

Figure 12. Generalization properties. Three-layer networks were trained by the three different algorithms on a set of 40 random input pairs. The nets were then tested on a new pattern set, and the average absolute error (as defined for Fig. 6) was recorded (*solid bars*). The error for this testing set was also recorded for the untrained networks for reference (*hatched bars*). In the test for the recognition of the training output locations (*left*), the testing set consisted of 40 new random inputs that coded for the same four spatial locations as in the training set. This tested whether the nets had really learned to add the eye position and retinal location vectors to obtain the resulting craniotopic location, and not just formed an associative memory storage of the training set. In the test for generalization to new locations (*right*), the testing set consisted of random inputs that coded for 40 new random head-centered locations. This tested for the networks' ability to generalize the vector addition operation to new targets. Note that the error bars labeled *Mixed* $A_{R,P}$ were obtained (for the second task only) by transferring a set of Mixed $A_{R,P}$-trained weights on a deterministic network and testing for generalization using this network. The reason for doing this was that the Mixed $A_{R,P}$ network cannot perform the second generalization task because there are too few hidden units to produce new locations in the continuous output format. The All $A_{R,P}$ network does not have this problem because the binary output format codes for regions of space and not for unique locations. *BP*, backpropagation.

learned to perform the transformation implied in the training examples. In our task, this mapping is the addition of two position vectors.

We tested two extensively trained networks for two types of generalization abilities. One is the ability to perform the correct vector addition of new, random input patterns that code for the same output locations as the training set. As shown in Figure 12 (left), all three networks performed this task extremely well. The other generalization task required the trained networks to give the correct output for input patterns coding for new output locations (Fig. 12, right), which is a more difficult task. Although all networks produced some error, this was still considerably less than for the untrained nets, indicating that the networks generalized to a considerable extent.

Discussion

Choice of the Learning Algorithm

The choice of the algorithm used to update the network's connection strengths was the central issue of our study. There exist a number of procedures to change the weights of a network in order to maximize some measure of performance in a supervised learning paradigm (for review, see Barto, 1985; Hinton, 1987; Lippmann, 1987; Anderson and Rosenfeld, 1988; McClelland and Rumelhart, 1988). An important class of such learning algorithms consists of those that maximize the performance measure by following its gradient (i.e., the direction of its maximum increase) with respect to the weights and adjusting them accordingly, arriving at the set of weights that produces the optimal output for every pattern in the training set. Backpropagation is an important and powerful algorithm belonging to this class. It has been used to train networks to perform such disparate tasks as pronouncing written English text (Sejnowski and Rosenberg, 1986) and detecting explosives in passengers' luggage at airports (Shea and Lin, 1989). It is also the training procedure used by Zipser and Andersen to teach their network to perform the coordinate transformation task (Zipser and Andersen, 1988). Powerful as it is, however, backpropagation suffers from the problem mentioned above of not being easily implementable in biological neuronal hardware. Central to the backpropagation algorithm are (1) the feedback of detailed error signals that are specific for each output unit, (2) the retrograde propagation of these signals from the output units back to the hidden and input units, and (3) the adjustment of synaptic strengths using global network information, that is, information about the activities and errors of units removed from the synapse whose strength is adjusted. These requirements represent major hurdles to envisioning backpropagation as a plausible model of learning in biological neural networks even to only a rough approximation, a concern that has been expressed by some of the discoverers of this algorithm (Rumelhart et al., 1986b). Possible solutions to this problem have been proposed, which require, for example, specialized connections carrying the error signals for each unit in the network (Hecht-Nielsen, 1989), or symmetrical feedback pathways with connection strengths identical to those in the forward network (Parker, 1985; Zipser and Ru-

melhart, 1990). Besides being rather unconvincing from a biological perspective, however, these complicated mechanisms detract quite a bit from the simplicity of structure and process that makes parallel neural networks so appealing as models of nervous system function.

We chose the $A_{R\text{-}P}$ learning algorithm because it uses the same abstract principles involved in supervised learning as backpropagation, but with specific processes that are more plausible for implementation in neurobiological hardware. In particular, in the $A_{R\text{-}P}$ algorithm, (1) the feedback signal is a single scalar value computed from the output units' average error; (2) this signal is the same for all units in the network, and as such it does not require backwards propagation along network connections; and (3) synaptic strength is adjusted using the payoff signal and information about the activity of the presynaptic and postsynaptic unit only. We will return to these three features shortly.

How the Network Learns

The abstract principle used by backpropagation is *gradient descent,* the minimization of an error measure by following the negative of its gradient with respect to the weights. A priori there are no conceptual or experimental obstacles to envisioning the brain using this principle, too, given a plausible performance measure. The S′-model $A_{R\text{-}P}$ algorithm, as implemented in both of our network classes, also makes use of this general principle. While the backpropagation algorithm, however, computes the exact value of the error's gradient for a given input pattern, the S′-model $A_{R\text{-}P}$ rule computes only an estimate of that gradient (Williams, 1986, 1987; Barto and Jordan, 1987). Units trained by the $A_{R\text{-}P}$ rule do not have the detailed information about the error vector and the state of other units, which is necessary to compute the exact gradient and which backpropagation units obtain through nonbiological pathways. Due to the random noise in their output, however, $A_{R\text{-}P}$ units can "jitter" their activity during learning so as to get an estimate of how the noise in activity affects the payoff they receive, which in turn allows them to estimate the direction in weight space along which to change their weights in order to increase reinforcement.

While this method allows $A_{R\text{-}P}$-trained units to adjust their weights properly using only locally available information, it is more random in its search for a solution than backpropagation. These differences are obvious in the learning curves of Figure 6. Backpropagation's precise computation of the performance gradient tells the algorithm the exact manner in which to change the weights so that the error is monotonically decreased, resulting in the smooth curve of Figure 6*a*. As this curve shows, the error falls quickly to a value below the resolution of the gaussian units in the retinal input (10°), and then continues to decrease much more slowly as the output is refined to match the training signal. The curves for $A_{R\text{-}P}$ learning (Fig. 6*b,c*) follow an envelope very similar to the backpropagation curve, but they are much noisier. The noise is due to the randomness of the $A_{R\text{-}P}$ units' output. In order to learn, the $A_{R\text{-}P}$ element adjusts its weights so that its net input drives its probability of firing toward one of the flat regions of the sigmoid function, thus effectively decreasing the randomness of its output. The decrease of the $A_{R\text{-}P}$ units' jitter as learning proceeds is reflected in decreasing noise on the learning curves.

Biological Plausibility of the $A_{R\text{-}P}$ Algorithm

As we mentioned above, one crucial requirement for our choice of a learning algorithm was a greater plausibility of biological implementation than the backpropagation algorithm. We must point out at the outset, however, that $A_{R\text{-}P}$ networks were not designed as literal models of biological neural nets (Barto, 1985, 1989). Because of the poor knowledge we have of the precise mechanisms of information processing used by nervous systems, the most useful connection between artificial and biological neural networks is presently limited to the description of abstract processes in simplified models and the investigation of the *possibility* of implementation in the biological hardware. In other words, the $A_{R\text{-}P}$ element was not designed by collecting scattered known facts of neurobiology and molding them into a computationally interesting unit capable of supervised learning, but rather as a simple, "neurally inspired" element with a few theoretically motivated features that give it interesting learning abilities. We will discuss biological plausibility, therefore, in its literal sense of suggesting that the abstract computing processes performed by the $A_{R\text{-}P}$ unit during learning are more in keeping with possible neural mechanisms proposed and partly demonstrated by experimental neuroscientists than the mechanisms used by backpropagation networks.

The first and most salient element of $A_{R\text{-}P}$ models, which aligns them with many neurobiological and psychological models of learning, is the scalar reinforcement signal, r. This has the attractive features of being computed from an average value of the error of the output units and of being transmitted as a single value to all the $A_{R\text{-}P}$-trained units in the network equally. This error could also be detected, for example, as a function of the angular difference between an object in space and the end position of a reaching arm movement or a saccade toward that object. After successful training, this difference would be nil and reinforcement would be maximal. The reinforcement signal could thus be computed by a part of the nervous system that monitored the animal's behavior with very little information about the activity of area 7a neurons. The fact that a single value is valid for All $A_{R\text{-}P}$ units implies that only one connection is necessary from the reinforcement computing region to area 7a. In the backpropagation algorithm, on the other hand, the output units' errors must be kept as separate components as they are fed back to the network to adjust individual weights.

A single scalar value is easier to transmit to a group of neurons than an error signal with specific multiple components. Evidence that the nervous system may

use such signals already exists. For example, the nucleus basalis sends a widespread system of cholinergic connections to several cortical areas, and the signals involved appear to be related to behavioral choices and reward (Richardson et al., 1988). The reinforcement signal required by our model, of course, would not have to be distributed on such a wide scale. The signal could carry information about a specific motor task, for example, the accuracy of a saccade to a target, and thus be used only by one or a few ensembles of neurons in area 7a. Because a single signal, however, would be valid for an entire group of neurons, there would be no need to propagate it through special pathways to specific units in the network of interest. This feature of the $A_{R\text{-}P}$ algorithm is more attractive than backpropagation's requirement for the retrograde propagation of error signals along specific pathways.

Besides not having to carry specific information to train individual neurons, the reinforcement signal used in our networks has the advantage that it can be independent of any coordinate system. In backpropagation, the "teacher" signal must code for the correct head-centered locations as a vector in a craniotopic reference frame. The $A_{R\text{-}P}$ algorithm, on the other hand, computes its feedback signal from the average of the output error's absolute value (Eqs. 2, 3), which is a single number that can be derived from the comparison of retinotopic as well as craniotopic positions.

Another "biological" feature of learning by $A_{R\,P}$ units is the use of information that is locally available to the element itself at its individual synapses, in a fashion reminiscent of Hebbian learning. The weight-adjusting equation for the ith $A_{R\,P}$ unit (Eq. 1) consists of the sum of two terms, each assigning the "reward" and the "penalty" portions, respectively, of the learning rule. These terms consist of three components: (1) the payoff signal, r (and the corresponding penalty value, $1 - r$); (2) information regarding the current state of the unit ($x_i - p_i$); and (3) the output of the presynaptic element, x_j, directly available at the synapse that the jth unit makes onto the ith unit.

We have already discussed the first component. In the second, x_i is the unit's output (0 or 1), and p_i is the probability that the unit's output will be one given the current net input, which depends on the unit's weights. As mentioned above, this probability could also be interpreted as the rate at which the unit will fire given the present input. These two values, as well as x_j (component 3), are effectively available at the connection between the input unit and the given hidden unit. The $A_{R\text{-}P}$ rule therefore embodies one of the most important elements of Hebbian learning (Hebb, 1949), that is, the proportionality of a change in synaptic strength to both presynaptic and postsynaptic signals. Hebbian learning remains one of the most attractive mechanisms for synapse modification, both on theoretical (Linsker, 1989) and experimental grounds (Ito, 1984; Kelso et al., 1986; Sejnowski et al., 1989; Stanton and Sejnowski, 1989; Brown et al., 1990). This is in contrast to backpropagation, in which changes in strength at one connection require information about the activities and error signals at all the connections in every layer above that connection (Rumelhart et al., 1986a).

The Mixed $A_{R\text{-}P}$ network, as we have mentioned, uses the $A_{R\text{-}P}$ rule only to train its hidden units. Its output units are trained by the delta rule (Eq. 4). Although this is the same rule as is used for the output units in backpropagation training, this does not pose as many obstacles to biological implementation as the full backpropagation algorithm does. As shown in Figure 5*a*, the only extra information required by the delta rule, as compared to the $A_{R\text{-}P}$ rule, is an error vector from the external evaluator. This is needed to form an individual error signal for each output unit, whose weights are then adjusted by error correction. There is no requirement, however, for backpropagation of error or activity signals across synapses. In fact, the delta rule also has a Hebbian form at the output layer, again in the sense that all the information required to adjust a connection's strength is available at the synapse. In Equation 4 two terms are multiplied, the first of which (in square brackets) contains "postsynaptic" information, while the other is the activity of the "presynaptic" unit.

The last feature that adds some biological flavor to the $A_{R\text{-}P}$ unit is the probabilistic nature of its output. The unpredictability of the exact firing rate produced by a neuron for any given presentation of a certain input has long been recognized as a feature of nerve cells (see, e.g., Tolhurst et al. 1983; Tolhurst, 1989; Vogels et al., 1989). In fact, this stochastic aspect of activity is one of the reasons neurophysiologists usually present data as summed histograms of several trials (Sejnowski, 1981). This is a feature that is not included in the deterministic units of backpropagation networks. The binary stochastic element's output is not simply noisy. It has a variance that is an increasing function of the mean probability of firing. The variability of spike trains recorded from cortical neurons also exhibits this statistical property (Vogels et al., 1989).

Many discussions of this aspect of neural activity have emphasized the difficulties it creates, such as the requirement it may impose on certain types of sensory information to be distributed over populations of neurons (Tolhurst, 1989). In our model, however, the intrinsic variability of the $A_{R\text{-}P}$ units' responses to input signals is essential for the learning process. It allows the network to jitter its weights around the current set of values, thus sustaining the search for a better solution. The noise provides the algorithm with the means of obtaining information about local variations in reinforcement. By making successive incremental adjustments to the weights, the algorithm converts these local variations into an estimated gradient of the reinforcement signal. In this manner the noise compensates, in a sense, for the scarcity of information contained in the scalar payoff signal. The stochastic aspects of the model, therefore, are not mere demonstrations of robustness to noise. The $A_{R\,P}$ rule demonstrates, rather, how a computational unit's out-

put variance can be used to achieve learning in a network that receives less than optimal feedback information.

Simulation Results

The basic results of this study corroborate the validity, from a physiological perspective, of parallel networks with distributed representations as models of area 7a. We have shown that the $A_{R\text{-}P}$ algorithm can train a neural network to perform the same coordinate transformation task as that performed by Zipser and Andersen's model. We also found that the solution discovered by this algorithm is equivalent to that found by backpropagation. As was established in Zipser and Andersen's analysis (Zipser and Andersen, 1988), this solution gives hidden unit response properties (planar gain fields and large visual receptive fields) very similar to those of area 7a neurons presumed to code for spatial location. These response properties, therefore, are not a specific result of the backpropagation training procedure. The set of connections strengths computed by the $A_{R\,P}$ algorithm, moreover, is not a unique one imposed by the network's architecture. Other solutions, not involving planar gain fields or large receptive fields, can be constructed that would work for the training sets we used. It is striking, then, that $A_{R\text{-}P}$ and backpropagation produce this particular algorithm for computing coordinate transformations, and not any other.

In a more detailed analysis of the model, we have shown that a second layer of hidden units can be added to the network without changing the response properties of the first hidden layer, and that the model networks are indeed capable of generalizing their coordinate transformation abilities to new input patterns. Both these results strengthen the physiological significance of this model architecture. The former implies that relay elements—an important and ubiquitous feature of brain architecture—are not an obstacle to learning and allow similar solutions to develop. The latter establishes the important point that these model networks are indeed learning to perform the coordinate transformation task. They do not merely act as content-addressable memories, associating each input pattern in the training set with its correct output individually, but rather they are capable of abstracting from the training examples the transformation common to them, in this case vector addition, and applying it successfully to new pairs of retinal and eye positions. This property has been observed before in parallel networks with very few hidden units in the hidden layer compared to the input layer (Cottrell et al., 1987).

The number of training iterations required for convergence by $A_{R\text{-}P}$ and backpropagation were comparable for networks and training sets of the size we used. We have not examined in our study the issue of how the $A_{R\text{-}P}$ algorithm behaves for networks with considerably larger numbers of hidden units and training locations. From previous experience with this learning rule (Barto and Jordan, 1987), learning should be significantly slower for such networks. It may be possible, however, to make the algorithm more resistant to scale changes, for example, by using a topographically more specific reinforcement signal. Our use of a single scalar feedback signal could thus be viewed as a worst-case scenario that does not exclude more specialized signals that may be used by biological systems.

Future Directions

It would be desirable to modify the $A_{R\text{-}P}$ algorithm so that it could train networks with continuous-output hidden units. Actually, any algorithm capable of performing gradient descent using a scalar reinforcement signal to train continuous-output units would be acceptable. Such algorithms are currently being developed (e.g., Gullapalli, 1988), and it would be a natural continuation of this work to try to apply them to networks modeling area 7a. The major hurdles in these algorithms involve the theoretical details of simultaneously updating the mean and variance of multiparametric distributions required by continuous stochastic units. The present form of these algorithms is similar to that of the $A_{R\text{-}P}$ procedure for binary units. It is conceivable that the extension of the concepts of supervised $A_{R\text{-}P}$ learning to networks with continuous output units will be a natural refinement that should not drastically change the types of solutions obtained.

Conclusion

We have shown that (1) the $A_{R\text{-}P}$ algorithm can train neural networks to compute coordinate transformations; (2) the convergence times for small networks are comparable to those obtained by backpropagation training; (3) in the process of learning this computation, the hidden units develop gain fields and receptive fields qualitatively similar to those of area 7a neurons; (4) the solutions are equivalent to those computed by backpropagation; and (5) these networks generalize appropriately. We have also pointed out a number of features of the $A_{R\text{-}P}$ algorithm that bring it closer than backpropagation to what is known about biological learning. We must emphasize again that the focus of our interest at this point is not in how literally $A_{R\,P}$ networks reproduce individual neurophysiological processes. It is rather the fact that these algorithms form a family of training procedures that yield similar functional representations when applied to a class of parallel distributed networks, and that they can do so using mechanisms not excluded, and perhaps suggested, by neurophysiological evidence.

These results represent a step toward establishing the physiological validity of the architecture and general learning principles of the model of area 7a introduced by Zipser and Andersen. They show that physiological properties can arise from a more plausible learning algorithm than backpropagation, thus suggesting that the detailed processes by which neuronal ensembles learn may play only a secondary role in their ultimate collective behavior. Abstract optimization principles, such as gradient descent, may in-

stead be more important determinants of neuronal learning strategies, and it would be worthwhile to pursue such hypotheses with further theoretical and experimental studies.

Notes

1. The logistic function, which is a type of "squashing" function, has a sigmoidal shape and maps real-valued inputs into the interval 0 to 1, according to $f(s) = 1/(1 + \exp(-s))$. In our networks, s is the sum of the unit's inputs weighted by the corresponding connection strength, plus a bias.

2. The frequency of local minima was around 5% for backpropagation and approximately 1% for the A_{R-P} algorithm, in approximately 200 different simulations. One likely reason for the rather high frequency of local minima for backpropagation is the small number of hidden units in the network. The A_{R-P} networks were less affected by this parameter, mainly because the unit's output noise improved the network's chances of escaping local minima.

This work was supported by ONR Grant N00014-89-J-1236 and NIH Grant EY05522 to R.A.A., by a grant from the Siemens Corporation to M.I.J., and by NIH Medical Scientist Training Program Grant 5T32GM07753-10 to P.M. We thank Sabrina J. Goodman for helpful discussion and for providing several computer programs.

Correspondence should be addressed to Dr. Richard A. Andersen, Department of Brain and Cognitive Sciences, E25-236, Massachusetts Institute of Technology, Cambridge, MA 02139.

References

Andersen RA (1989) Visual and eye movement functions of the posterior parietal cortex. Annu Rev Neurosci 12: 377–403.

Andersen RA, Zipser D (1988) The role of the posterior parietal cortex in coordinate transformations for visual-motor integration. Can J Physiol Pharmacol 66:488–501.

Andersen RA, Essick GK, Siegel RM (1985) Encoding of spatial locations by posterior parietal neurons. Science 230:456–458.

Andersen RA, Essick GK, Siegel, RM (1987) Neurons of area 7a activated by both visual stimuli and oculomotor behavior. Exp Brain Res 67:316–322.

Anderson JA, Rosenfeld E, eds (1988) Neurocomputing. Cambridge, MA: MIT Press.

Barto AG (1985) Learning by statistical cooperation of self-interested neuron-like computing elements. Hum Neurobiol 4:229–256.

Barto AG (1989) From chemotaxis to cooperativity: abstract exercises in neuronal learning strategies. In: The computing neuron (Durbin RM, Miall RC, Mitchison GJ, eds), pp 73–98. New York: Addison-Wesley.

Barto AG, Anandan P (1985) Pattern recognizing stochastic learning automata. IEEE Trans Sys Man Cybern 15:360–375.

Barto AG, Jordan MI (1987) Gradient following without backpropagation in layered networks. Proc IEEE Int Conf Neural Networks 2:629–636.

Brown TH, Kairiss EW, Keenan CL (1990) Hebbian synapses: biophysical mechanisms and algorithms. Annu Rev Neurosci 13:475–511.

Cottrell GW, Munro P, Zipser D (1987) Image compression by back-propagation: an example of extensional programming. San Diego: University of California, Institute for Cognitive Science, ICS Report 8702.

Goodman SJ, Andersen RA (1989) Microstimulation of a neural-network model for visually guided saccades. J Cogn Neurosci 1:317–326.

Gullapalli V (1988) A stochastic learning algorithm for learning real-valued functions via reinforcement feedback. Amherst: University of Massachusetts, COINS Technical Report 88-91.

Hebb DO (1949) The organization of behavior. New York: Wiley.

Hecht-Nielsen R (1989) Theory of the backpropagation neural network. Proc Int Joint Conf Neural Networks 1: 593–605.

Hinton GE (1987) Connectionist learning procedures. Pittsburgh: Carnegie-Mellon University, Technical Report CMU-CS-87-115.

Ito M (1984) The cerebellum and neural control. New York: Raven.

Kelso SR, Ganong AH, Brown TH (1986) Hebbian synapses in hippocampus. Proc Natl Acad Sci USA 83:5326–5330.

Linsker R (1989) How to generate ordered maps by maximizing the mutual information between input and output signals. Neural Comput 1:402–411.

Lippmann RP (1987) An introduction to computing with neural nets. IEEE ASSP Mag 4:4–22.

Mazzoni P, Andersen RA, Jordan MI (1991) A more biologically plausible learning rule for neural networks. Proc Natl Acad Sci USA 88:4433–4437.

McClelland JL, Rumelhart DE (1988) Explorations in parallel distributed processing. Cambridge, MA: MIT Press.

Minsky ML, Papert S (1969) Perceptrons. Cambridge, MA: MIT Press.

Parker DB (1985) Learning logic. Cambridge: MIT, Center for Computational Research in Economics and Management Science, Technical Report TR-47.

Richardson RT, Mitchell SJ, Baker FH, Delong MR (1988) Responses of nucleus basalis of Meynert neurons in behaving monkeys. In: Cellular mechanisms of conditioning and behavioral plasticity (Woody CD, Alkon DL, McGaugh JL, eds), pp 161–173, New York: Plenum.

Rumelhart DE, Hinton GE, Williams RJ (1986a) Learning internal representations by error propagation. In: Parallel distributed processing: explorations in the microstructure of cognition, Vol 1 (Rumelhart DE, McClelland JL, PDP Research Group), pp 318–362. Cambridge, MA: MIT Press.

Rumelhart DE, Hinton GE, Williams RJ (1986b) Learning representations by back-propagating errors. Nature 323: 533–536.

Sejnowski TJ (1981) Skeleton filters in the brain. In: Parallel models of associative memory (Hinton GE, Anderson JA, eds), pp 189–212, Hillsdale, NJ: Erlbaum.

Sejnowski TJ, Rosenberg CR (1986) NETtalk: a parallel network that learns to read aloud. Baltimore: The Johns Hopkins University, Department of Electrical Engineering and Computer Science, Technical Report JHU/EECS-86/01.

Sejnowski TJ, Chattarji S, Stanton PK (1989) Introduction of synaptic plasticity by Hebbian covariance in the hippocampus. In: The computing neuron (Durbin RM, Miall RC, Mitchison GJ, eds), pp 105–124. New York: Addison-Wesley.

Shea PM, Lin V (1989) Detection of explosives in checked airline baggage using an artificial neural system. Proc Int Joint Conf Neural Networks 2:31–34.

Stanton PK, Sejnowski TJ (1989) Associative long-term depression in the hippocampus induced by Hebbian covariance. Nature 339:215–218.

Tolhurst DJ (1989) The amount of information transmitted about contrast by neurons in the cat's visual cortex. Visual Neurosci 2:409–413.

Tolhurst DJ, Movshon JA, Dean AF (1983) The statistical reliability of signals in single neurons in cat and monkey visual cortex. Vis Res 23:775–785.

Vogels R, Spileers W, Orban GA (1989) The response variability of striate cortical neurons in the behaving monkey. Exp Brain Res 77:432–436.

Werbos PJ (1974) Beyond regression: new tools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University.

Widrow B, Hoff ME (1960) Western Electronic Show and Convention record, Pt 4, Adaptive switching circuits, pp 96–104. New York: Institute of Radio Engineers.

Williams RJ (1986) Reinforcement learning in connectionist networks: a mathematical analysis. San Diego: University of California, Institute for Cognitive Science, Technical Report ICS 8605.

Williams RJ (1987) A class of gradient-estimating algorithms for reinforcement learning in neural networks. Proc IEEE Int Conf Neural Networks 2:601–608.

Zipser D, Andersen RA (1988) A backpropagation programmed network that simulates response properties of a subset of posterior parietal neurons. Nature 331:679–684.

Zipser D, Rumelhart DE (1990) Neurobiological significance of new learning models. In: Computational neuroscience (Schwartz E, ed), pp 192–200. Cambridge, MA: MIT Press.