TOP-DOWN TEACHING ENABLES TASK-RELEVANT CLASSIFICATION WITH COMPETITIVE LEARNING †

Virginia de Sa Dana Ballard

Computer Science Department
University of Rochester
Rochester, NY 14627-0226
(email:[desa,dana]@cs.rochester.edu)

Abstract

Unsupervised competitive learning classifies patterns based on similarity of their input representations. As it is not given external guidance, it has no means of incorporating task-specific information necessary for classifying based on semantic similarity. This report describes a method of augmenting the basic competitive learning algorithm with a top-down teaching signal. This teaching signal removes the restriction inherent in unsupervised learning and allows high level structuring of the representation while maintaining the speed and biological plausibility of a local Hebbian style learning algorithm. The function of the teaching input is illustrated geometrically and examples, using this algorithm in small problems, are presented. This work supports the hypothesis that cortical back-projections are important for the organization of sensory traces during learning.

1 Introduction

Supervised learning algorithms, of which back-propagation [17] is the most used, are well known for their ability to attack problems which we cannot solve explicitly. Given a set of input output pairs they are able to learn an appropriate mapping. However, there are several technical problems with back-propagation type algorithms[10]. They are not well suited for tasks where the input pattern space changes over time and are often slow to learn, particularly with many hidden units. Also the semantics of the algorithm are poorly understood and it is not biologically plausible, restricting its usefulness as a model of neural learning.

Unsupervised learning algorithms such as competitive learning [18, 10] and Kohonen Feature Mapping [11] are based on a form of Hebbian learning [9]. This is a local learning algorithm in that all information necessary to calculate weight changes at a connection are available at that connection. This results in more biologically plausible learning with increased learning speed and adaptability. The problem with these algorithms is that they are restricted in their applicability: Not given any desired output, they are unable to incorporate task-specific information and are limited to learning how to best represent the input data.

There have been attempts to combine the benefits of both supervised and unsupervised learning algorithms. One variant of the associative reward-penalty algorithm (A_{R-P})[1] can be seen as a biologically plausible approximation of back-propagation[2] but is unfortunately extremely slow. Hybrid methods designed to attack the speed problem learn faster [10] but, as they usually have an unsupervised layer followed by one or more supervised layers [10, 7, 14], they are limited to applications that map similar inputs to similar outputs and in most cases their biological plausibility is still in doubt.

The algorithm we develop combines the general applicability and utility of supervised learning algorithms with the speed, simplicity and biological plausibility of unsupervised algorithms. It works by augmenting the basic competitive learning algorithm with a teaching signal which allows task relevant information to guide the development of synaptic connections. This basic idea has been suggested before [18, 6, 15] but we present a more elegant, complete treatment. We illustrate the method via its solutions to the XOR problem and on small scale problems in pattern recognition. The final section argues that the resulting learning algorithm is biologically plausible and agrees well with what is known about biological learning.

2 Background

Competitive learning is a neural network learning algorithm for categorizing inputs. "Neurons" (or units) compete to fire for particular inputs and then learn to respond better to the inputs that activated them by

†This research was supported by a grant from the Human Frontier Science Program and by a Canadian NSERC 1967 Science and Engineering Scholarship.

0-7803-0559-0/92 $3.00 © 1992 IEEE

adjusting their "weights" (or synaptic connections) in a Hebb-like [9] way [1]. The network is usually a 2 layer feed-forward network with full forward connections. Input patterns $\vec{\xi}$ are applied to the neurons in the first layer, one component of the input vector ξ_i for each input neuron i. The neurons in the output layer are divided into one or more winner-take-all (WTA) groups. Neurons within each WTA group "compete" to respond for the pattern. The "winning" neuron in group k for the μ^{th} pattern $\vec{\xi}^{\mu}$ is defined to be the neuron with index j_k^* and incoming vector of weights $\vec{w}_{j_k^*}$ such that

$$||\vec{w}_{j_k^*} - \vec{\xi}^{\mu}|| \leq ||\vec{w}_j - \vec{\xi}^{\mu}|| \text{ for all j in group k} \quad (1)$$

The winning neuron in each WTA group then updates its weights according to:

$$\Delta w_{i,j_k^*} = \eta(\xi_i^{\mu} - w_{i,j_k^*})$$

where w_{i,j_k^*} represents the weight from input neuron i to the winning output neuron in group k. η is a learning parameter that reflects the malleability of the network and is usually a decreasing function of time. (If the network has more than two layers the input pattern for the next layer is the output of the previous layer which consists of all 0's except at the winning neurons which are usually assigned a value of 1.)

Kohonen Feature Mapping [11] is similar except that there is only one output neuron WTA group and it has a specified topology (usually a 2-D grid). Nearby neurons update their weights with a distance dependent attenuated version of the input pattern. This results in nearby neurons responding to similar patterns. The weight update formula in this case is

$$\Delta w_{i,j} = \Lambda(j, j^*)\eta(\xi_i^{\mu} - w_{i,j}) \text{ for all j}$$

where $\Lambda(j, j^*)$ is a monotonic decreasing function of $||r_j - r_{j^*}||$ where r_j gives the position of the neuron with index j in the topology within its layer.

3 Top-down Teaching

In competitive learning, different neurons become responsive to different input patterns. If there are more patterns than output neurons within a WTA group, then similar patterns (where similarity is defined as in equation 1) will map to the same output neuron. In this way competitive learning algorithms learn to group similar inputs without a supervisory signal.

This natural grouping, however, is a disadvantage in tasks that require the mapping of dissimilar inputs to similar outputs. In order to achieve classification based on semantic similarity as opposed to the similarity of the inputs, it is necessary to provide more information. This can be done with a "teaching" input which tells the network which dissimilar inputs should be mapped to the same output. The purpose of the teaching input is to bias the activations of the neurons so that patterns from the same class are more likely to activate the same neurons. Once a neuron "wins" on a pattern it is more likely to win again. In this way we can train the neurons to respond to patterns from the same semantic class so that even with removal of the teaching input they will respond appropriately.

Our goal is to have X output neurons separate X pattern classes— a different output neuron activated for input patterns from each class. However, as shown in Section 3.1.1 a layer of X competitive neurons can only separate X classes if they are pairwise linearly separable. As we are dealing with non-linearly separable problems we require a hidden layer with extra neurons.

We would like to connect the teach neurons to both the hidden and output neurons but this would result in infeasible connectivity from the teaching neurons. This paper shows that the teaching signal can reach all non-input neurons if we connect it directly to the output layer neurons and allow backward propagation of the teaching signal through the use of backward projections from the output to the hidden layer. This results in a cleaner, more efficient solution [4] than achieved by connecting the teach input to the hidden layer as in [18]. The architecture is shown in Figure 1. It should be emphasized that, unlike the back-propagation algorithms, the projections that propagate the teach information backwards have the same structure (they are actual connections) as the forward projections [2].

[1] The connection between a pre- and post-synaptic cell is strengthened when they exhibit correlated firing.

[2] This idea that back-projections can help organize the representation of incoming information to allow it to be more useful has been suggested as a role of back-projections in the neocortex [15].

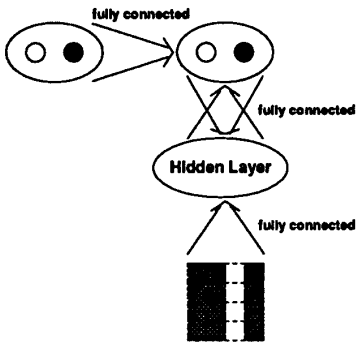

Figure 1: Our architecture has the teach input connecting to the output layer and feedback projections from the output layer to the hidden layer. The neurons in the output and hidden layers perform competitive learning.

3.1 Geometric Interpretation

In a feedforward network the weight vector of a neuron contains one weight from each neuron in the previous layer. In particular, neurons in the hidden layer can be represented as a point in the input space (where the input space is simply the vector space of dimensionality $Length(\xi)$). The competitive learning algorithm can be thought of as trying to position the weights in the input space in such a way that they minimize the distance between the inputs and their closest weight vector over the distribution of input patterns.

By adding recurrent connections we increase the dimensionality of the weight space of the hidden layer neurons. The algorithm still minimizes the distance between the incoming patterns and the weight vectors but this time the space is higher dimensional as the incoming patterns have been augmented with the recurrent signal. This signal is actually feedback from the output neurons but we shall see in Section 3.2 that the output neurons are forced by the teach neurons (thus the output signal is just a scaled version of the teach signal) and hence we can think of the recurrent signal as coming directly from the teach neurons.

These extra teaching dimensions influence the positions of the weight vectors by pulling them to the centres of clusters of augmented inputs. Thus if images which we consider similar are given similar values in the added teaching dimensions these augmented patterns will cluster. The weight vectors will be pulled to the centres of clusters of augmented inputs allowing for similarly taught images to activate the same neuron. During testing, when only the forward projecting weights are relevant, we can observe the results by "projecting" the weights back to the original (feedforward) input space. Thus the purpose of the backward projections is to move the values of the forward weights so that the appropriate classification is achieved in the absence of teaching input. These ideas will be concretely illustrated in the next section.

3.1.1 The XOR Problem

The XOR problem being a 2-D problem nicely illustrates the effect of the teaching input. Consider a network as above with only one output unit, one teach unit and two input units. The number of hidden layer units is varied from 4 to 2 as discussed below. The input patterns are from the two sets of vectors {[-1,-1], [1,1]} and {[1,-1],[-1,1]}. The goal is to have the output neuron respond positively for input from one of the input sets and negatively for input from the other set. This output coding is different than the usual competitive code mentioned earlier but was chosen so that the weights of the hidden units would be 3-dimensional and easily visualized. The activation update and weight update rule for the output layer were changed appropriately to reflect this change.

With four hidden layer neurons, feedback from the output to the hidden layer is not required. Each unit will become responsive to one of the input patterns. In this case the only need for a teach unit is to allow the output unit to make the correct positive/negative connections from the hidden layer units.

The problem becomes more interesting with 3 hidden layer units. In this case two patterns will have to

activate the same unit. In order for the problem to be solved it must be that two patterns with the same required output be grouped (i.e. [1, 1] with [-1, -1] or [1, -1] with[-1, 1]). That is, the two patterns that share a hidden neuron must be from the same class. However these patterns are more distant in the input space. Without teaching input the two patterns that activate the same unit will be patterns that are close in the input space. An example result is shown in Figure 2a) where the input patterns are represented by the small black dots and the projections of the weights onto the input space are represented by the larger, open circles. The dashed lines show the partition of the input space among the hidden layer neurons. The figure shows that inputs [1, 1] and [1, -1] share a hidden-layer neuron. With this weight distribution the problem cannot be solved.

The teaching input solves this problem by making the shared neuron be allotted to two patterns from the same class. Figure 2b) illustrates how this is accomplished. The teach input has forced the patterns apart along the third dimension. The patterns impinging on the hidden layer neurons are now from the 3-dimensional augmented input space. They are [-1, -1, K],[-1, 1, -K],[1, 1, K],[1, -1, -K] where $\pm$K is the activation of the output neuron. Closest patterns in the new 3-D space will be patterns of the same class as long as $|K| > 1$ [3] In this case the output weight vectors converge to positions such as those shown in the figure. Now when the teach signal is removed the hidden layer neurons receive zero activation from the output neurons and the problem is projected back to the original 2 dimensional input space. As shown in Figure 2c), the network still classifies the patterns correctly as no unit confuses inputs with different output values.

The case of 2 hidden layer units displays a limitation of the algorithm. With 2 hidden layer units and large enough K one neuron responds to each output class. However the weights of each neuron will tend to lie midway between the patterns of its class ([0,0,K] and [0,0,-K]). In this case when we remove the teach input both neurons have the same weights from the input units and we cannot separate the patterns appropriately. This is a direct result of the competitive approach— the input is represented in terms of which output neuron is activated. As one can consider the weight vectors of two neurons as points in space where the line midway between them partitions the input space between the two neurons, only linearly separable problems can be solved with 2 competitive neurons. Likewise a layer of X competitive neurons can only separate X classes if the classes are pairwise linearly separable.

3.2 The Algorithm

The architecture and corresponding algorithm are modeled after biological networks where activations are being continually propagated. The idea is that activation at the input (retinal stimulation) coexists with activation at the teach neurons (input from other systems). Propagation of activation proceeds in both directions [4] with continual Hebbian weight updates. In order to simulate the activation propagation on a digital computer it is convenient to assume synchronous dynamics and discrete time.

In order to present the update rules we define the following notation:

- x_i^j represents the activation of the ith neuron in layer j. The input, hidden, output and teach layers are respectively layers 0,1,2 and T.
- s_i^j are temporary variables.
- $w_{i,j}^{k,l}$ represents the weights of the connections. Superscripts k and l represent the layers of the pre- and post-synaptic neuron respectively and subscripts i and j represent the indices of the individual neurons within their respective layers.
- $\vec{w}_j^k$ represents the weight vector of incoming weights to neuron j in layer k.

The update rules then are (for k=1,2)

$$s_j^k(t) = \sum_i x_i^{k-1}(t-1) w_{i,j}^{k-1,k}(t-1) + \sum_i x_i^{k+1}(t-1) w_{i,j}^{k+1,k}(t-1).$$

[3] For convergence in a finite time it should be reasonably bigger than 1. Empirical tests showed 1.1 to be large enough.

[4] Also the winner-take-all (WTA) calculations are modelling lateral inhibitory circuits which will also be continually spreading activation.

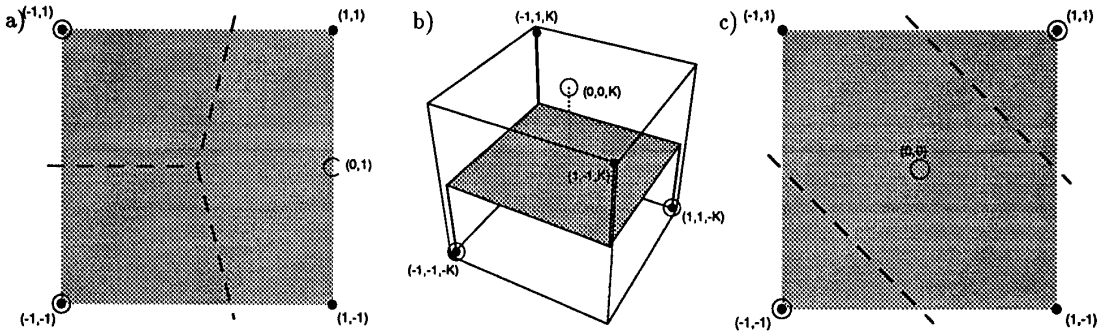

Figure 2: Input patterns are represented by the small dark dots, weights are represented by the larger open circles, and the dashed lines represent the partition of the input space among the neurons.
a) The patterns and weights for the XOR problem with 3 hidden-layer neurons and no teach input. The solution has allocated one weight to be shared between two of the closest patterns. The other two weights are allocated to the other patterns.
b) The patterns and weights for the XOR problem with 3 hidden-layer neurons and a teach input. The third dimension comes from the feedback from the output layer. The solution has allocated one weight to be shared between two neurons with the same feedback. The other two weights are allocated to the other patterns.
c) The result of removing the Teach input after training as in b).

$$x_j^k(t) = WTA(s_j^k(t)) \text{ for k=1,2}$$

where we have defined the teaching layer to be the highest layer (Layer 3) and the WTA function is defined such that:

$$WTA(s_j^k(t)) = \begin{cases} K(k) & \text{if } (2s_j^k - ||\vec{w}_j^k||^2) \geq (2s_i^k - ||\vec{w}_i^k||^2) \text{ for all i } [5] \\ 0 & \text{otherwise} \end{cases}$$

where $K(k)$ is a constant for each layer k.

The general weight update rules are (for k=1,2)

$$\Delta w_{i,j}^{k-1,k}(t) = \Lambda(j, j^{*k}(t))\eta(x_i^{k-1}(t-1) - w_{i,j}^{k-1,k}(t-1))$$

$$\Delta w_{i,j}^{k+1,k}(t) = \Lambda(j, j^{*k}(t))\eta(x_i^{k+1}(t-1) - w_{i,j}^{k+1,k}(t-1))$$

where the w's and x's are as above and $j^{*k}(t)$ is the index of the winning neuron in layer k at time t. Λ and η are as in Section 2. The choice of the Kohonen update rule in the current implementation is mainly to help in human visualization as shown in the next section.

In a general recurrent network, it is possible that the activation cycles will not converge. In order to avoid continual cycling we make the backward connections to the output from the teach layer significantly stronger than those from the hidden layer [6]. In this way the active neuron in the output layer is determined by the activation pattern of the teach layer — the hidden layer has no effect. This eliminates the recurrent dependency, between the output and hidden layer, removing the cycling problem

Although the forward connections from the hidden layer to the output layer are not effective during learning, they are important as they are being trained to produce the correct output in the absence of the teach input. The learning algorithm leads to relatively symmetric connections as the correlation based weight update rule strengthens the connections in both directions between the active neurons. This symmetry is important so that during the testing phase when the forward connections are the only active ones, (as the teach input and the activation of the higher layers is initially zero until activation reaches the higher layers)

[5] This condition is equivalent to Equation 1 (in Section 2).
[6] This is achieved by making the teach activation significantly stronger than K(1).

Figure 3: Example input patterns in the square/triangle experiment.

they are able to activate the correct neurons. The back-projecting weights have no extra effect during the testing phase [7].

3.3 Position invariant Recognition

The algorithm was tested on two small tasks in position invariant recognition of 2-D shapes. These problems are impossible for unsupervised competitive learning; different patterns in the same position have more overlap than the same pattern in a different position so the similarity metric is not able to separate based on pattern shape but only on position in the image. For both tasks, K(1) and K(2) were 1 and 2.5 respectively.

The first task was to distinguish horizontal from vertical lines. The lines were presented as positive activation in a 4×4 pixel array (an example input is shown in Figure 1). With 4 hidden layer neurons and no teaching input, each neuron became responsive to 1 horizontal and 1 vertical line [8] making it impossible for the output layer to distinguish between the two line types. With the addition of the teach input and recurrent connections each neuron became responsive to only horizontal or vertical lines and the output layer was able to respond differently for horizontal and vertical lines. With eight or more hidden layer neurons the recurrent connections were not necessary as each line could activate a different hidden layer neuron eliminating the potential for a neuron to confuse a horizontal and vertical line.

We also tested the algorithm on the problem of distinguishing squares from triangles. This problem is harder due to the considerable overlap of patterns from the two classes. Typical input patterns are shown in Figure 3. Again without the teach input and recurrent projections the neurons confuse the square and triangle inputs. An example of the weights of these neurons is shown in Figure 4. Squares and triangles in the appropriate positions have been drawn on top of the representation of the winning neuron for each pattern. Note that there are several neurons that won for both a square and a triangle. The addition of teach input and recurrent connections resulted in weight vectors such as those shown in Figure 5. In this case the neurons responding maximally to squares are disjoint from those responding maximally to triangles. This allows the task to be solved. In addition due to the topography of the Kohonen mapping [9] we can see that for the most part the "square" neurons are separated from the "triangle" neurons. This is another indication that the teach input was able to warp the grid of weight vectors in a perceptually relevant way.

4 Conclusions

We have seen that the addition of a teach input with recurrent projections greatly increases the uses of the competitive learning algorithm and may make it suitable for such tasks as position-invariant recognition. Although the algorithm has only been demonstrated on small problems, it was not specialized in any way for the above tasks. In fact given enough hidden layer neurons and full presentation of the data set any classification task could be learnt quickly using this algorithm. However to achieve generalization in real tasks the algorithm will have to be able to work with more layers. Future work will include modifying the algorithm to be appropriate for multi-layer learning.

The algorithm appears biologically plausible as it is essentially Hebb learning with lateral inhibition. The most obvious areas of implausibility are: the winner-take-all condition, the use of Equation 1 to determine the winning neuron, the different learning and testing procedures, and the teach input in general. The strict winner-take-all condition as currently implemented is biologically implausible and will be relaxed or softened [12] to more closely represent lateral connections in future studies. The other concerns are not as serious as:

[7] This agrees with the result from [5] mentioned in the Conclusions section.

[8] As horizontal and vertical lines have more pixels in common (and are thus more similar) than with lines of the same type.

[9] The relative positions of the winners in the 2D hidden layer grid give a crude idea of nearness in the original space— nearby weights on the 2D grid generally correspond to weights nearer in the original high-dimensional space.

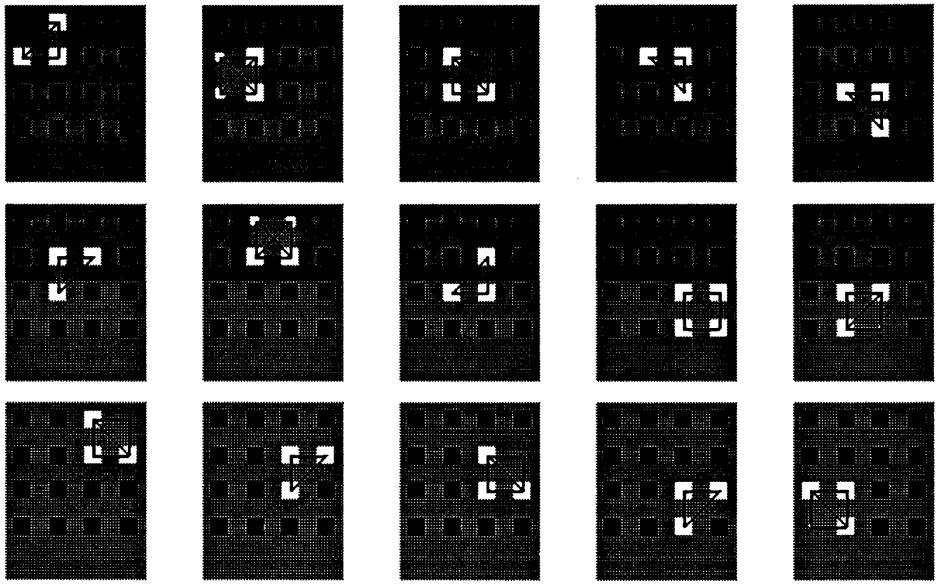

Figure 4: The weights of the hidden layer neurons with no teach input. Each large gray square represents a hidden layer neuron. The small squares represent the weights from the 4×4 input layer to these neurons. The size of these squares represents the magnitude of the connection; white/black represents positive/negative weights. Squares and triangles have been drawn in the appropriate positions on top of the representation of the winning neuron for the pattern.

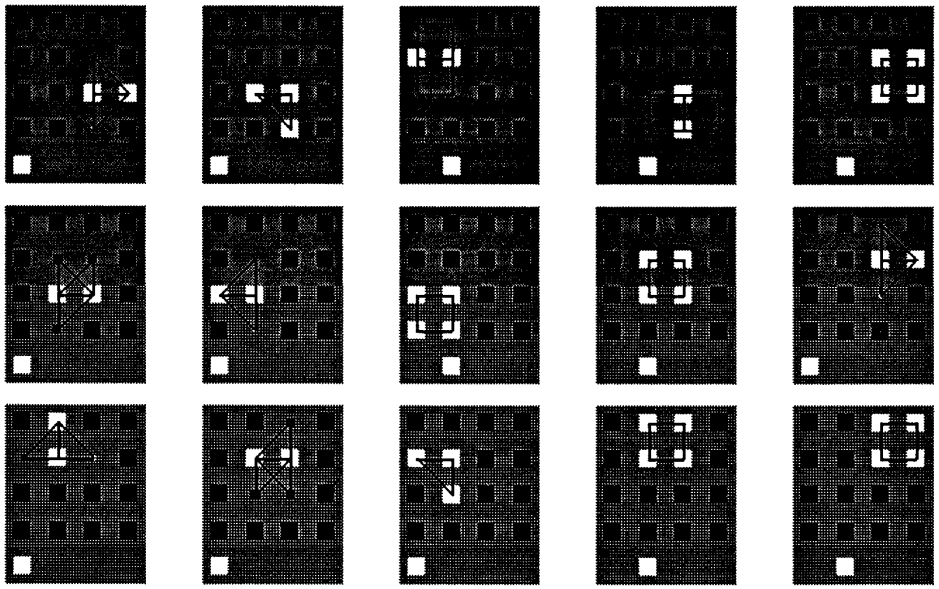

Figure 5: The weights of the hidden layer neurons after the addition of teach input and recurrent connections. Note the greater segregation of neurons responsive to squares from neurons responsive to triangles. The representation is as in the above Figure. The extra two weights represent the feedback from the two output neurons.

Equation 1 could be approximated by a neuron that moved its threshold proportional to its last maximal activation [4], neuromodulators could provide the learning/teaching signal [8], and we have recently shown [3] that the teach input can be replaced by presentation of correlated inputs to 2 or more connected networks.

Kohonen mapping algorithms have been shown to account well for the retinotopic maps observed in striate cortex [13]. It is possible that the addition of the teach input with recurrent connections may explain how higher areas achieve position invariant responses and object-centered representations [16]. This theory would also explain the purpose of the prevalent back-projections throughout cortex. One study [5] found that lesioning V2 had little affect on the response properties of cells in V1 indicating that back-projections, or at least the V2-V1 projection has little purpose during regular processing. In this report we have provided some support for the hypothesis that these back-projections are important for the organization of the incoming sensory stimuli during learning.

References

[1] A. G. Barto and P. Anandan. Pattern recognizing stochastic learning automata. *IEEE Transaction on Systems, Man, and Cybernetics*, 15:360–375, 1985.

[2] A. G. Barto and M. I. Jordan. Gradient following without back-propagation in layered networks. *Proceedings of the IEEE First Annual I.C.N.N.*, pages II–629–II–639, June 1987.

[3] V. R. de Sa and D. H. Ballard. Self-teaching through correlated input. Submitted to CNS*92.

[4] V. R. de Sa and D. H. Ballard. Top-down teaching enables non-trivial clustering via competitive learning. Technical Report 402, Dept. of Computer Science, University of Rochester, Nov. 1991.

[5] R. Eckhorn, R. Bauer, W. Jordan, M. Brosch, W. Kruse, M. Munk, and H. J. Reitburg. Coherent oscillations: A mechanism of feature linking in the visual cortex? *Biol. Cybern.*, 60:121–130, 1988.

[6] M. A. Fanty. *Learning in Structured Connectionist Networks.* PhD thesis, University of Rochester, Dept. of Computer Science, 1988. (Also available as UR CS TR 252).

[7] D. Fox, V. Heinze, K. Möller, S. Thrun, and G. Veenker. Learning by error-driven decomposition. In O. Simula, editor, *ICANN-91.* Elsevier Science Publishers, June 1991.

[8] M. E. Hasselmo, B. Anderson, and J. Bower. Selective cholinergic suppression of intrinsic but not afferent fiber synaptic transmission in rat piriform (olfactory) cortex. J. Neurophys. in press.

[9] D. O. Hebb. *The Organization of Behavior.* Wiley, 1949.

[10] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*, volume 1 of *Santa Fe Institute Studies in the Sciences of Complexity Lecture Notes.* Addison-Wesley, 1991.

[11] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[12] S. J. Nowlan. Maximum likelihood competitive learning. In *Advances in Neural Information Processing Systems 2*, pages 574–582. Morgan Kauffman, 1990.

[13] K. Obermayer, H. Ritter, and K. Schulten. A principle for the formation of the spatial structure of cortical feature maps. *Proc. Natl. Acad. Sci.*, 87:8345–8349, Nov. 1990.

[14] H. J. Ritter, T. M. Martinetz, and K. J. Schulten. Topology-conserving maps for learning visuo-motor-coordination. *Neural Networks*, 2:159–168, 1989.

[15] E. Rolls. The representation and storage of information in neuronal networks in the primate cerebral cortex and hippocampus. In Durbin, Miall, and Mitchison, editors, *The Computing Neuron*, chapter 8, pages 125–159. Addison-Wesley, 1989.

[16] E. T. Rolls. The representation of information in the temporal lobe visual cortical areas of macaques. In R. Eckmiller, editor, *Advanced Neural Computers*, pages 69–78. Elsevier Science Publishers, 1990.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–364. MIT Press, 1986.

[18] D. E. Rumelhart and D. Zipser. Feature discovery by competitive learning. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2, pages 151–193. MIT Press, 1986.