[Obermayer *et al.*, 1990] 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, November 1990.

[Ritter *et al.*, 1989] Helge J. Ritter, Thomas M. Martinetz, and Klaus J. Schulten, "Topology-Conserving Maps for Learning Visuo-Motor-Coordination," *Neural Networks*, 2:159–168, 1989.

[Rolls, 1989] Edmund 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.

[Rolls, 1990] Edmund 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.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," In David E. Rumelhart, James 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.

[Rumelhart and Zipser, 1986] D. E. Rumelhart and D. Zipser, "Feature Discovery by Competitive Learning," In David E. Rumelhart, James 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.

References

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

[Barto and Jordan, 1987] Andrew G. Barto and Michael 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.

[Eckhorn *et al.*, 1988] 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.

[Földiák, 1991] Peter Földiák, "Learning Invariance from Transformation Sequences," *Neural Computation*, 3(2):194–200, 1991.

[Fox *et al.*, 1991] Dieter Fox, Volker Heinze, Knut Möller, Sebastian Thrun, and Gerd Veenker, "Learning by Error-Driven Decomposition," In O. Simula, editor, *Proceedings of the International Conference on ARTIFICIAL NEURAL NETWORKS, ICANN-91.* Elsevier Science Publishers, June 1991.

[Graves *et al.*, 1987] Adreienne L. Graves, Yves Trotter, and Yves Frégnac, "Role of Extraocular Muscle Proprioception in the Development of Depth Perception in Cats," *Journal of Neurophysiology*, 58(4):816–831, 1987.

[Hasselmo *et al.*] M. E. Hasselmo, B.P. Anderson, and J.M. Bower, "Selective Cholinergic Suppression of Intrinsic but not Afferent Fiber Synaptic Transmission in Rat Piriform (Olfactory) Cortex," Journal of Neurophysiology in press.

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

[Hein, 1974] A. Hein, "Prerequisite for development of visually guided reaching in the kitten," *Brain Research*, 71:259–263, 1974.

[Hertz *et al.*, 1991] John Hertz, Anders Krogh, and Richard 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.

[Kohonen, 1982] Teuvo Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, 43:59–69, 1982.

[Miyashita, 1988] Yasushi Miyashita, "Neuronal correlate of visual associative long-term memory in the primate temporal cortex," *Nature*, 335(27):817–820, October 1988.

[Nowlan, 1990] Steven J. Nowlan, "Maximum Likelihood Competitive Learning," In *Advances in Neural Information Processing Systems 2*, pages 574–582. Morgan Kauffman, 1990.

at different places along the dendrites). [Hasselmo *et al.*] show that acetylcholine affects the synaptic connection strength of lateral but not feedforward connections in rat piriform cortex. Neuromodulators such as this could provide a means for neurons to give more or less weighting to the "teach" inputs. The neuromodulators or the teach inputs themselves could serve as a gate for learning ensuring that weight modification only occurs during times of appropriate teaching. It should also be noted that occasional misgating would not be disastrous.

The teaching input could be considered as input from other sensory modalities (or sub-modalities as in colour helping shape) possibly through the limbic system. Some of these modalities would be receiving nearly constant input and thus could serve as a teacher for others receiving changing stimuli. There is evidence that proprioceptive input is necessary for the development of visually guided reaching [Hein, 1974] and depth perception [Graves *et al.*, 1987]. Alternatively teaching input could come from temporally contiguous views. Miyashita in [Miyashita, 1988] has found neurons in macaque inferotemporal cortex that associated "visually distinct but temporally related" images. The article suggests that the strong limbic connections are important in this association.

Kohonen mapping algorithms have been shown to account well for the retinotopic maps observed in striate cortex [Obermayer *et al.*, 1990]. 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 [Rolls, 1990]. This theory would also explain the purpose of the prevalent back-projections throughout cortex. [Eckhorn *et al.*, 1988] 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.

Acknowledgements

We would like to thank Steven Nowlan and Leonidas Kontothanassis for their helpful comments and suggestions on earlier drafts.

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 [Nowlan, 1990] to more closely represent lateral connections in future studies. The other areas are discussed below.

We have used Equation 1 to determine the maximally responding neuron. However, as it it is not immediately clear how a neuron could compare an input activation pattern to its weight vector, this rule seems less biologically plausible than Equation 2. However as

$$||\vec{w}_j - \vec{\xi}|| = \sqrt{||\vec{w}_j||^2 + ||\vec{\xi}||^2 - 2(\vec{w}_j \cdot \vec{\xi})}$$

and the pattern vector $\vec{\xi}$ (here $\vec{\xi}$ refers to the augmented pattern vector of all inputs) is constant for all the neurons, minimizing $||\vec{w}_j - \vec{\xi}||$ is equivalent to maximizing $(\vec{w}_j \cdot \vec{\xi}) - 0.5 \times ||\vec{w}_j||^2$. Thus Equation 1 is similar to Equation 2 with the addition of a threshold term. This leaves open the question as to whether a neuron could modify its threshold proportionately to the norm of its weight. Consider the patterns for which neuron j wins. In these cases the $\vec{\xi}$ are very much like $\vec{w}$ and thus

$$\sum w_{i,j}\xi_i \approx ||w_j||^2.$$

Hence a neuron that moved its threshold according to its last maximal activation would be implementing a rule close to Equation 1. (Details on this rule will be available in a future technical report.) The above argument ignored the effect of the competitive interactions on the activations of the neurons. However if these interactions have little effect on the maximally activated neuron but function mainly to suppress non-winning neurons the argument will not be much affected. Thus Equation 1 may not be as implausible as it seems.

In biological learning systems there is no clear distinction between learning and testing. There will always be sensory information coming from other systems. Sometimes that information will be helpful and other times it won't. It is known that concentrations of neuromodulators vary greatly under different conditions. These changes in concentrations could affect the relative strengths of feedforward and feedback connections (as they occur

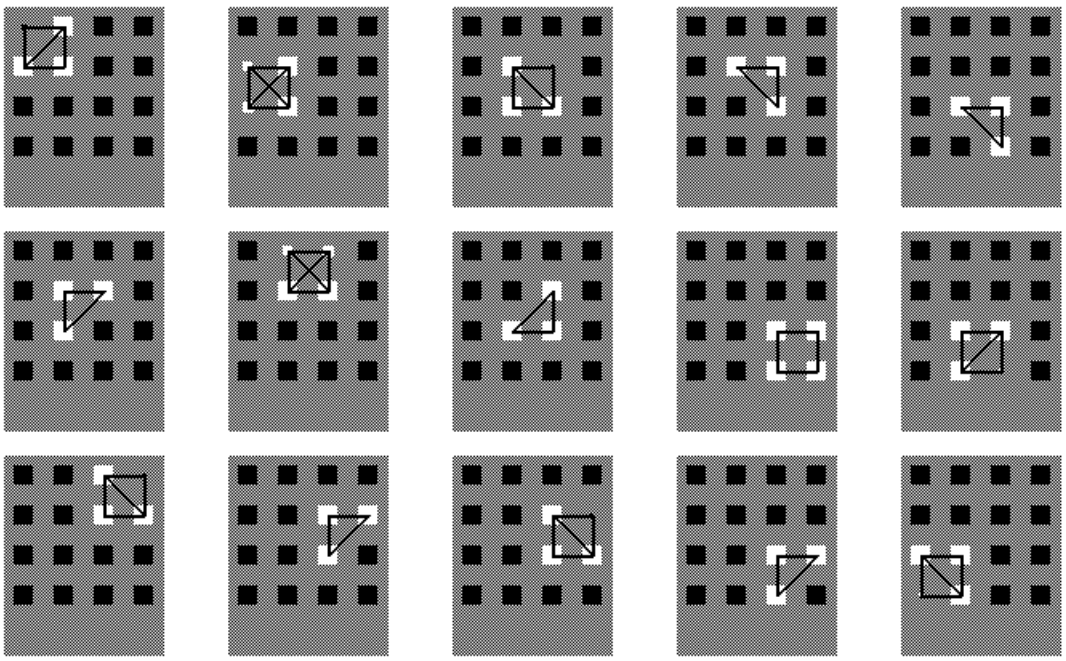

Figure 14: The weights of the hidden layer neurons with no teach input. Each large gray square represents a hidden layer neuron. The small squares represent the weights from the 4×4 input layer to these neurons. The size of these squares represents the magnitude of the connection; white/black represents 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.

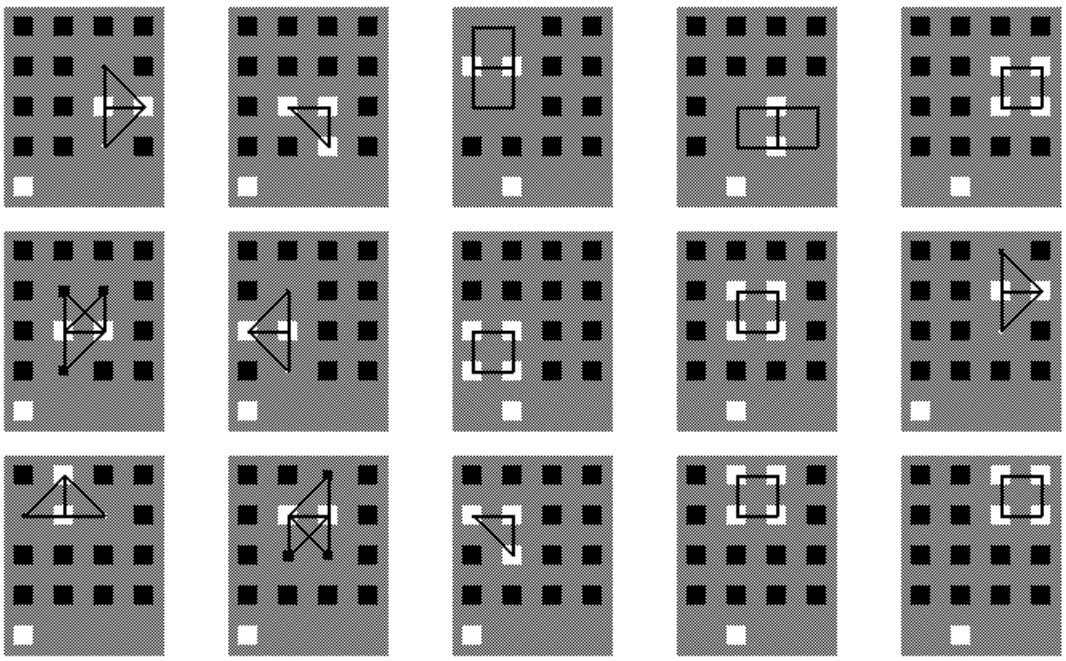

Figure 15: The weights of the hidden layer neurons after the addition of teach input and recurrent connections. Note the greater segregation of units responsive to squares from units responsive to triangles. The representation is as in the above Figure. The extra two weights represent the feedback from the two output units.

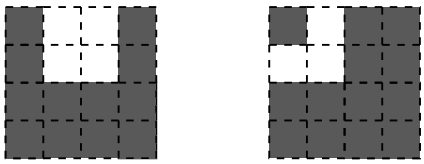

Figure 13: Example input patterns in the square/triangle experiment. These patterns were placed in all positions and in all orientations on the 4×4 grid with the exception that the right angle of the triangle occurred on one of the four inner squares. The dark squares represent input of -1 and the white input of +1.

3.5 Distinguishing Squares from Triangles

As a final test, we 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 13. For this task, a 16 neuron input layer, a 15 unit hidden layer (arranged in a 5×3 grid), 2 output units and 2 teach units were used. As in the line example, K(1) was 1, K(2) was 2.5. and the activation of the teaching units were set to [5,0] and [0,5] for triangles and squares respectively.

Kohonen feature mapping was useful in this case to help visualize the influence of the teaching feedback in determining the weight vectors of the hidden layer. We cannot directly visualize the weights in the 18 dimensional space as we could with the 3D weights in the XOR problem. However the topological constraint on the weight vectors results in neighbouring neurons developing similar weights. Therefore the relative positions of the winners in the 2D hidden layer grid will give a crude idea of nearness in the original space. That is nearby weights on the 2D grid generally correspond to weights nearer in the original high-dimensional space.

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 14. 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 15. In this case the units 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 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.

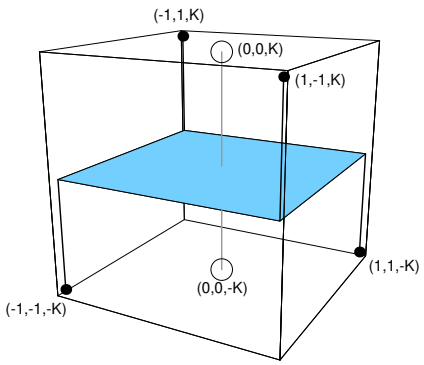

Figure 11: The patterns and weights for the XOR problem with 2 hidden-layer neurons and a teach input. The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The solution has allocated one weight for each output class.

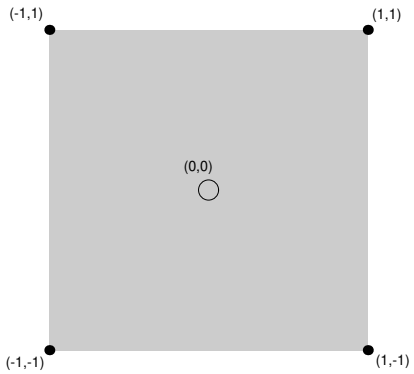

Figure 12: The Result of removing the Teach input after training as above. The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. When the recurrent connections are ignored the two neurons have the same weights so they are drawn superimposed.

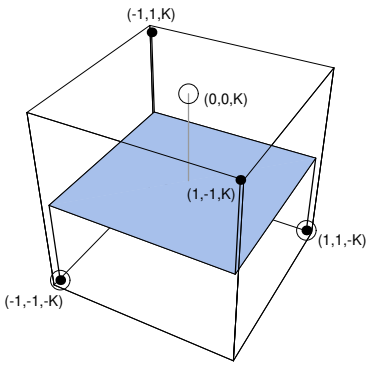

Figure 9: The patterns and weights for the XOR problem with 3 hidden-layer neurons and a teach input. The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The third dimension comes from the feedback from the output layer. The solution has allocated one weight to be shared between two neurons with the same feedback. The other two weights are allocated to the other patterns.

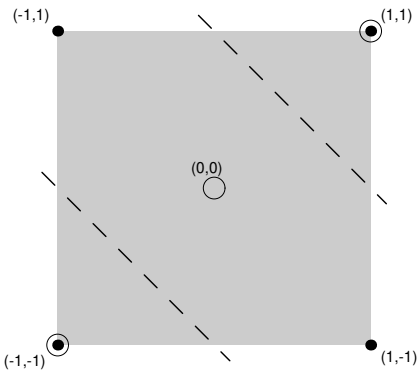

Figure 10: The result of removing the Teach input after training as above. The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The dashed lines indicate the partition of the input space among the 3 neurons.

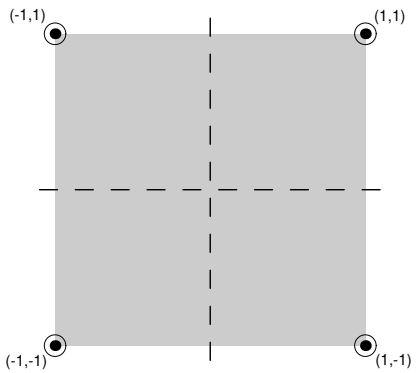

Figure 7: The patterns and weights for the XOR problem with 4 hidden-layer neurons. The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The solution has allocated one weight vector for each input pattern so the dots and circles are superimposed. The dashed lines indicate the partition of the input space among the four neurons.

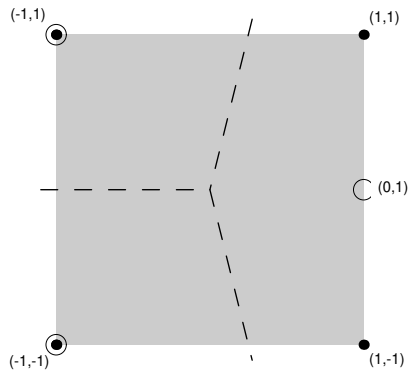

Figure 8: The patterns and weights for the XOR problem with 3 hidden-layer neurons and no teach input. The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The solution has allocated one weight to be shared between two of the closest patterns. The other two weights are allocated to the other patterns. The dashed lines indicate the partition of the input space among the three neurons.

represented in Figure 7 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. 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 8 where 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 9 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(2)],[-1, 1, -K(2)],[1, 1, K(2)],[1, -1, -K(2)]. Closest patterns in the new 3-D space will be patterns of the same class as long as $|K(2)| > 1$ [11] and thus $|||[-1,-1,K(2)]-[1,1,K(2)]|| < ||[-1,-1,K(2)]-[-1,1,-K(2)]||$ and the other analogous relations hold. 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 10, 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(2) one neuron responds to each output class. However the neurons will tend to lie midway between the patterns (see Figure 11). In this case when we remove the teach input both neurons have the same weights to the input units and we cannot separate the patterns appropriately (see Figure 12). This is a direct result of the competitive approach. The input is represented in terms of which output neuron is activated. As one cannot position two points (representing the weights of 2 neurons) in the 2D plane such that [-1, -1] and [1, 1] are closest to one point and [1, -1] and [-1, 1] are closest to another point we cannot do the XOR task with a hidden layer of 2 WTA neurons. As one can consider the two neurons as points in 2D where the perpendicular line midway between them partitions the input space among the two neurons, this is equivalent to saying that the XOR problem is not linearly separable.

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

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 as the output neurons are forced by the teach neurons the output signal is just a scaled version of the teach signal and thus for this section we can think of the recurrent signal as coming directly from the teach neurons.

The extra teaching dimensions influence the positions of the weight vectors. The weight vectors will be pulled 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 similar 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. These ideas will be concretely illustrated in the next section.

The XOR Problem

The XOR problem being a 2-D problem nicely illustrates the effect of the teaching input. Consider a network to solve the problem 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 the four vectors: [-1,-1], [-1,1], [1,-1] and [1,1]. In order for the weights of the hidden units to be 3-dimensional and easily visualized we chose a 1 dimensional output code (+/-). The teach input is 5 for the patterns [-1,1] and [1,-1] and -5 for the other two. As this output coding is slightly different from the [1,0]/[0,1] coding, the activation propagation and weight change rules for the output layer had to be changed. The calculation of $s^2(t)$ is as before but $x^2(t)$ is now calculated as

$$x^2(t) = WTA2(s^2(t))$$

where the WTA2 function is defined as

$$WTA2(s^2(t)) = \begin{cases} K(k) & \text{if } s^2 \geq 0 \\ -K(k) & \text{otherwise.} \end{cases}$$

Note that there is only one output neuron so the subscript on the above rules has been dropped.

The weight update rules are changed accordingly to

$$\Delta w_{i,1}^{1,2}(t) = \begin{cases} \Lambda(1,1)\eta(x_i^1(t-1) - w_{i,1}^{1,2}(t-1)) & \text{if } x^2(t) \geq 0 \\ \Lambda(1,1)\eta(-x_i^1(t-1) - w_{i,1}^{1,2}(t-1)) & \text{otherwise.} \end{cases}$$

The rule for $\Delta w_{i,j}^{3,2}(t)$ is exactly analogous.

With four hidden layer neurons, feedback from the teach input to the hidden layer is not required. Each unit will become responsive to one of the input patterns. This is

weights during testing. The biological plausibility of this is discussed in the Conclusions section.

3.3 Distinguishing Horizontal from Vertical Lines

The algorithm was tested on the same task used in [Rumelhart and Zipser, 1986] with three slight changes in the coding strategy. The input was changed from a 6×6 pixel array to a 4×4 pixel array[9]. The activation of the input units were coded as -1 and 1 as opposed to 0/1 and the teaching input was simply 2 extra inputs (instead of 36 extra). The activations of the teaching inputs were set to [5,0] and [0,5] for horizontal and vertical lines respectively. None of these changes affect the results of applying either algorithm [10]. K(1) and K(2) were assigned values of 1 and 2.5 respectively.

Another minor modification to the approach used in [Rumelhart and Zipser, 1986] is the use of the Kohonen feature mapping learning algorithm. This is a method of avoiding "dead units",i.e. neurons that don't win on any of the input patterns due to their initial random weight assignment putting them far from the input patterns [Hertz *et al.*, 1991]. Rumelhart and Zipser use a slightly different method. The use of the Kohonen variant of competitive learning is useful for human visualization of the effect of the teaching input in higher dimensions. This is shown in Section 3.5.

The results were similar to those of [Rumelhart and Zipser, 1986]. Without any teach input each neuron would become responsive to 1 horizontal and 1 vertical line. With the addition of the teach input and recurrent connections each neuron became responsive to only horizontal or vertical lines. With eight or more hidden layer neurons the recurrent projections 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.

3.4 Geometrical 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 $Length(\xi)$). As Mentioned in the Background section, 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 the 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

[9]Subsequently a 6×6 pixel array was tested with similar results. A value of 3 for K(2) was used to balance the greater number of input units.

[10]although the 6×6 input array allows for a greater probability of the two modules dividing the lines up differently as required in the Rumelhart and Zipser model

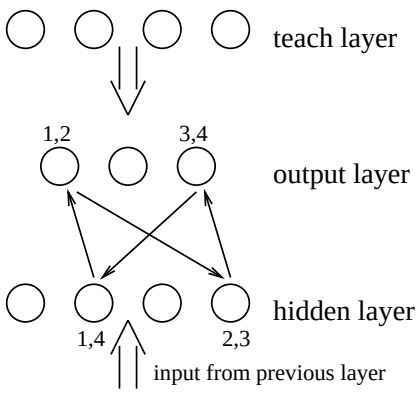

Figure 5: The problem with non-symmetric recurrent connections. The numbers represent the neurons that are active at each time step and the arrows represent the strongest connections from relevant neurons. The figure shows that if activation starts on the nodes marked 1 it will lead to activation of the nodes marked 2 in the next cycle followed by those marked 3 and then those marked 4 and then back to those marked 1 to start the cycle again. Only the dominant connections are shown.

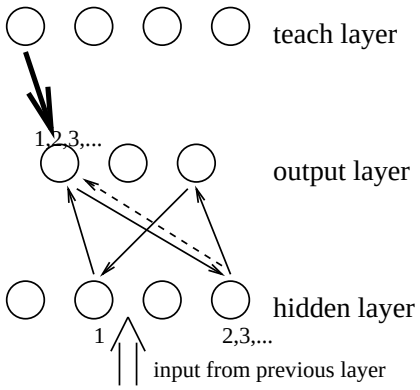

Figure 6: The cycling problem can be avoided by having stronger connections in one direction. In this case the strong connection from the teach layer forces the activation of one neuron in the output layer. This neuron together with the input layer activation will determine the active neuron in the hidden layer. As the active hidden layer neuron will not change the active output layer neuron, cycling is avoided.

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 reasons for choosing the Kohonen update rule in the current implementation are discussed later in Section 3.

In a general recurrent network, it is possible that the activation cycles will not converge. That is, the activation pattern may never become constant over time but continually cycle [6]. This could happen when the forward and backward projections are not symmetric so that activation of a neuron in layer 1 will lead to activation of a neuron in layer 2 but that neuron will lead to activation of a different neuron in layer 1. (Note that due to the winner-take-all nature only one neuron per layer is activated at each time). This problem is illustrated in Figure 5.

We avoid this problem by making the backward connections to the output from the teach layer significantly stronger than those from the hidden layer [7]. 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 as illustrated in Figure 6. In this figure the teach layer has forced activation of one of the neurons in the output layer (as shown by the thick arrow). The hidden layer activation pattern is determined by both the input layer and output layer activation patterns. Cycling is avoided because the hidden layer activation does not change the activation of the output or input layer neurons.

Thus during the learning stage the network is not really a recurrent network as the forward connections from the hidden layer to the output layer are not effective during learning. The forward connections are important however as they are being trained so that when the teach input is not present the correct output is still achieved. The training leads to relatively symmetric connections as the correlation based weight update rule strengthens the connections in both directions between the active neurons.

During the testing phase the teach input and the activation of the higher layers is initially zero so forward connections are the only active ones until activation reaches the higher layers. Due to the symmetry induced in the forward and backward weights during the learning phase the forward weights are able to activate the correct neurons. The back-projecting weights have no extra effect during the testing phase [8].

In order to not "unlearn" the network should learn (change its weights) only when the teaching signal is present. The current implementation achieves this by not modifying

[6] This is useful for dynamical applications such as trajectory generation, but is not what we want in our case.

[7] This is achieved by making the teach activation significantly stronger than K(1).

[8] This agrees with the [Eckhorn *et al.*, 1988] result mentioned in the Conclusions section.

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

$$s_j^1(t) = \sum_i x_i^0(t-1) w_{i,j}^{0,1}(t-1) + \sum_i x_i^2(t-1) w_{i,j}^{2,1}(t-1)$$

$$s_j^2(t) = \sum_i x_i^1(t-1) w_{i,j}^{1,2}(t-1) + \sum_i x_i^T(t-1) w_{ij}^{T,2}(t-1)$$

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

where 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.

If we define the teaching layer to be the highest layer (Layer 3) then the first two equations can be generalized to (for k=1,2)

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

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

[5] This condition is equivalent to Equation 1 (in Section 2). We do not use Equation 2 because this equation requires normalized weights. If we normalized the weights in this algorithm we would be normalizing over both forward and backward connections. However during the testing phase the backward connections are ineffective and thus the active weights are no longer normalized. Normalizing separately over forward and backward connections also will not work as this means the forward weights can not reach values such as (0,0). The biological plausibility of this rule is discussed in the Conclusions section.

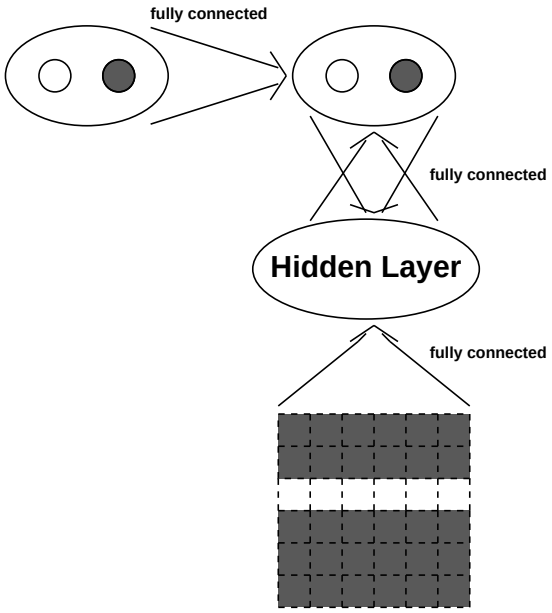

Figure 4: Our architecture has the teach input connecting to the output layer and feedback projections from the output layer to the hidden layer

The connections required in the top-down algorithm are

$$(I \cdot H) + [2 \cdot (H \cdot O)] + (T \cdot O).$$

Thus there is a connectivity saving of

$$I \cdot H + T(2H - O).$$

For larger problems in which many hidden layer neurons are required the savings with the top-down algorithm may be considerable.

Another benefit to this approach results in the ability to replace the external teaching signal by the presentation of many views of the same object close in time combined with persistent activation of the output neurons as in [Földiák, 1991]. This memory within the neurons could replace the external signal as all that is required is a biasing of the output representation so that views of the same object give the same or similar output. The system could again learn to associate different views from the same object.

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).

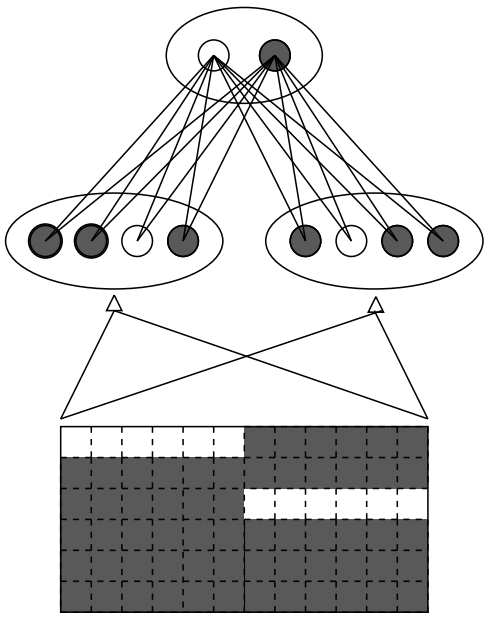

Figure 3: The architecture used for the horizontal/vertical line discrimination task in [RumelhartZipser86]. The hidden-layer neurons are grouped into 2 groups. There is competition only within each group.

an error that higher layers can not undo. Thus the teach input must be available to them to prevent them from grouping patterns from different classes.

A simple solution would be to connect the teach neurons to all but the input neurons but this would result in infeasible connectivity from the teaching neurons. This report shows that the backward propagation of the teaching signal can be accomplished through the use of backward projections from the output to the hidden layer. The proposed architecture is shown in Figure 4. 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 [3].

With this top-down algorithm the duplication of neurons in the hidden layer to introduce redundancy is not required. This results in networks with fewer neurons. The required connectivity is also reduced. If we let I represent the number of input-layer neurons, H the number of hidden-layer neurons required to segregate the patterns, T the number of teach neurons and O the number of output-layer neurons the number of connections in the Rumelhart and Zipser model are

$$[(I+T)\cdot 2H]+[2H\cdot O].$$

[3]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 [Rolls, 1989].

two vertical neurons. Thus a layer with more than 2 neurons is needed to perform the segregation.

The hidden layer however introduces another problem in that the output layer is not connected to the teaching input. Thus the output layer has no information to group the two horizontal neurons apart from the two vertical neurons. The output neurons at any one time see only a vector with one 1 and three 0's in it. These pattern vectors are all equidistant providing no information for appropriate grouping. To state it another way — There is no redundancy in the coding in the hidden-layer units. As the teach input does not reach the output layer these neurons are performing unsupervised learning which can not succeed without redundancy in the input data [Hertz *et al.*, 1991].

Rumelhart and Zipser addressed this problem by duplicating the hidden layer neurons to form two clusters and allowing competition only within each cluster (see Figure 3). Now the output neurons see a pattern with two 1's, one from each of the hidden layer groups. For a horizontal line one of the horizontal neurons from each group will be activated. This solution depends on the probability that the division of the horizontal and vertical lines will occur differently in the two groups. In this case different horizontal lines will activate different combinations of the "horizontal line neurons". The important point is that the occurrence of activation of all 4 horizontal neurons will have overlap with each other but not with the vertical neurons. Each of the horizontal lines will have hidden layer activations that differ in 0,2 or 4 bits from activations of all other horizontal lines but in 4 bits from all the vertical lines. Thus the patterns of activation caused by the horizontal lines will be grouped separately from those caused by the vertical lines allowing the output neurons to segregate horizontal from vertical lines.

Once the network is taught in this way, removing the teaching input still results in correct classification. The teach input is only necessary during the teaching phase [2]

3 Top-down Teaching

3.1 Problems with Bottom-up Teaching

The duplication solution, although it usually works for the example task, is hardly elegant. The main problem stems from the fact that the teaching input is not available to the output layer. We suggest a "top-down" approach that projects the teach input to the highest layer in the network.

However, if there are more input patterns than hidden layer neurons, it is important to have teaching information available to the hidden layer. This is because if the number of input patterns exceeds the number of hidden layer neurons some of the neurons must win on more than one pattern. If they win on patterns from different classes they will introduce

[2] This is under the assumption that the weights are not allowed to change during the testing (no teach) phase. This assumption is discussed later.

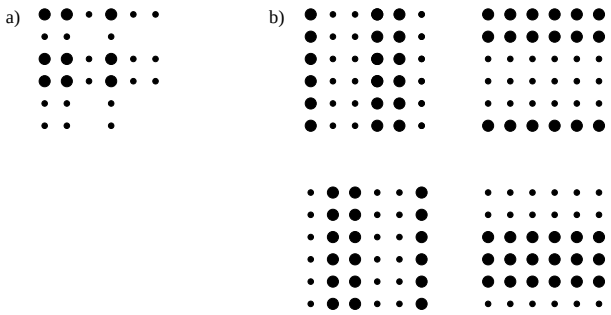

Figure 2:
a) The results of training without a teaching input (from [RumelhartZipser86]). This diagram shows the weights from a representative hidden unit from the hidden layer in Figure 1a. The size of the dots represents the strength of the connection between each of the 36 input neurons and this hidden neuron. This figure shows the typical result of training the network without a teaching input.
b) The results with the addition of the teaching input (from [RumelhartZipser86]). This diagram shows the result of training four hidden units with a teaching input. Each square represents 1 hidden unit and the size of the dots represent the strength of the connections from each of the input units.

the other horizontal lines (for the 0/1 coding the dot products are 1 and 0 respectively). Figure 2a shows one example of the weights of the hidden-layer neurons. The unit shown has become responsive to 3 horizontal and 3 vertical lines.

Rumelhart and Zipser represent the teaching input by doubling the number of input pixels where the new half contains a constant horizontal line with all the horizontal lines and a constant vertical line with all the vertical lines as shown in Figure 1b. The exact form of the teaching input, however, is not important. The important point is the presence of extra salient information which is the same for inputs that are to be classified the same. The purpose of the teaching input is to bias the activations of the hidden-layer neurons so that lines 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 lines of the same type so that even with removal of the teaching input they will respond appropriately. As the teaching signal is presented as additional input, we term this "bottom-up" teaching.

With this teaching stimulus the hidden layer was able to divide up the horizontal and vertical lines appropriately. Figure 2b shows a typical division of the input patterns with four hidden-layer units. (The display format is the same as that in Figure 2a except all 4 units are displayed.) Figure 2b also illustrates the need for the hidden layer. In the absence of the hidden layer the input would connect directly to the output and the information stored in the weights of the four hidden layer neurons would have to be combined in the two output neurons. However combining the weights of the two horizontal neurons would result in a neuron with the same weights as one resulting from the combination of the

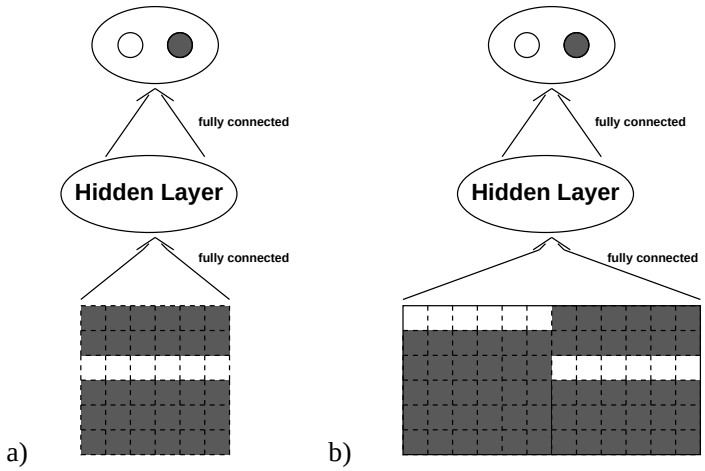

Figure 1:
a) The architecture first tried in [RumelhartZipser86]
b) The modified architecture that makes use of a teaching input (the extra 36 input neurons on the left hand side are the teaching input)

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.

Competitive learning algorithms learn to group similar inputs without a supervisory signal. However this 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 similar outputs. Rumelhart and Zipser [Rumelhart and Zipser, 1986] developed a supervised competitive technique to do this. Their algorithm learned a task not solvable with an unsupervised approach.

The task was to distinguish between horizontal and vertical lines in varied positions across an image. Their network has an input, hidden and output layer and full forward connections between the layers. The image is presented to the input layer as a 6×6 pixel map. There is one input neuron for each pixel. Lines are represented as six 1's on a background of 0's. The output layer has 2 neurons, one to represent each of the line classes. The architecture and an example input are shown in Figure 1a.

The teaching input is essential, as without it the hidden layer neurons group horizontal lines with vertical lines as in the input space horizontal lines are more similar to vertical lines than to the other horizontal lines. That is, the vector representation of a horizontal line has a greater dot product with the representation of a vertical line than with those of

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 clustering inputs. "Neurons" (or units) compete to fire for particular inputs and then learn to respond better to the inputs that activated them by adjusting their "weights" (or synaptic connections) in a Hebb-like [Hebb, 1949] 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 grouped into one or more clusters. (Unless otherwise specified we will be referring to networks with only one cluster to layer.) Neurons within each cluster "compete" to respond for the pattern. The "winning" neuron in cluster 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

$$||\vec{w}_{j_k^*} - \vec{\xi}^{\mu}|| \leq ||\vec{w}_j - \vec{\xi}^{\mu}|| \text{ for all j in cluster k} \tag{1}$$

For normalized weight vectors this condition is equivalent to the more easily calculated condition

$$\vec{w}_{j_k^*} \cdot \vec{\xi} \geq \vec{w}_j \cdot \vec{\xi}^{\mu} \text{ for all j in cluster k.} \tag{2}$$

The winning neuron in each cluster 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 cluster 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.)

The weight update results in the winning neurons becoming more selective for that input pattern. Within each cluster different neurons become responsive to different input patterns. The weights converge to vectors such that the distance between the patterns and their closest weight is minimized. If there are more patterns than output neurons within a cluster then similar patterns (where similarity of input patterns is defined analogously to equations 1 or 2) will map to the same output neuron. In this way the neurons partition the input space.

Kohonen Feature Mapping [Kohonen, 1982] is similar except that there is only one output neuron cluster 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.

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

1 Introduction

Supervised learning algorithms, of which back-propagation [Rumelhart *et al.*, 1986] is the most used, are well known for their ability to attack problems which we do not know how to solve explicitly. Given a set of input output pairs they are able to learn a mapping, often with appropriate generalization to other inputs. However, despite the successes with back-propagation type algorithms there are several technical problems with their use[Hertz *et al.*, 1991]. The algorithm is not well suited for tasks where the input pattern space changes over time and it is 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 [Rumelhart and Zipser, 1986; Hertz *et al.*, 1991] and Kohonen Feature Mapping [Kohonen, 1982] are based on a form of Hebbian learning [Hebb, 1949]. This is a local learning algorithm in that all information necessary to calculate weight changes at a neuron are available at that neuron. 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}) of [Barto and Anandan, 1985] can be seen as a biologically plausible form of back-propagation [Barto and Jordan, 1987]. Instead of back-propagating an error vector to each neuron, a global "correctness" scalar signal is broadcast to all neurons. Stochastic units are then used to approximate the back-prop algorithm. Unfortunately the approximation takes significant time making this algorithm extremely slow.

Other hybrid methods attack the speed problem. These usually have an unsupervised layer followed by one or more supervised layers [Hertz *et al.*, 1991; Fox *et al.*, 1991; Ritter *et al.*, 1989]. These networks learn faster [Hertz *et al.*, 1991] however they are limited to applications that map similar inputs to similar outputs and in most cases their biological plausibility is still in doubt.

Rumelhart and Zipser [Rumelhart and Zipser, 1986] developed a method of augmenting a competitive learning algorithm with extra "teaching" input to influence the competition that shapes the network. Their algorithm forms the basis for our work and will be discussed in the next section.

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 in a way that greatly extends its usefulness. A teaching signal allows task relevant information to guide the development of synaptic connections. We illustrate the method via its solutions to the XOR problem and on a small scale problem in object recognition. The final section argues

Top-down Teaching Enables Non-trivial Clustering via Competitive Learning

Virginia de Sa Dana Ballard

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 402

November 1991

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 useful 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. Examples, using this algorithm in small problems, are presented and the function of the teaching input is illustrated geometrically. This work supports the hypothesis that cortical back-projections are important for the organization of sensory traces during learning.

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