Methods for Learning Articulated Attractors over Internal Representations

David C. Noelle
(NOELLE@CNBC.CMU.EDU)
Center for the Neural Basis of Cognition
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Andrew L. Zimdars
(ZIMDARS@ANDREW.CMU.EDU)
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Abstract

Recurrent attractor networks have many virtues which have prompted their use in a wide variety of connectionist cognitive models. One of these virtues is the ability of these networks to learn *articulated attractors* — meaningful basins of attraction arising from the systematic interaction of explicitly trained patterns. Such attractors can improve generalization by enforcing "well formedness" constraints on representations, massaging noisy and ill formed patterns of activity into clean and useful patterns. This paper investigates methods for learning articulated attractors at the hidden layers of recurrent backpropagation networks. It has previously been shown that standard connectionist learning techniques fail to form such structured attractors over internal representations. To address this problem, this paper presents two unsupervised learning rules that give rise to componential attractor structures over hidden units. The performance of these learning methods on a simple structured memory task is analyzed.

Introduction

Connectionist attractor networks have been used to model many aspects of cognitive performance, involving domains as diverse as word naming (Plaut and McClelland, 1993), cognitive control (Cohen et al., 1996), and conscious awareness (Mathis and Mozer, 1995). Such networks have many virtues. Processing element activity evolves over time in complex ways in such networks, allowing them to capture various aspects of the dynamics of cognition. Learned recurrent connection weights often lend themselves to interpretation as soft constraints between representational elements, facilitating analysis of such models. One of the most interesting advantages of these networks, however, is the manner in which the learning of attractor basins can aid generalization of performance.

Attractor networks can learn to enforce "well formedness" constraints on representations, and this process of "cleaning up" patterns of activity can facilitate generalization (Mathis and Mozer, 1995). Such enforcement is implemented by the instantiation of a distinct stable fixed-point attractor for every possible well formed representation. It is important to note that this potentially combinatoric space of valid attractor basins need not be explicitly trained, but may arise in the interaction between trained patterns (Plaut and McClelland, 1993). When the dynamics of a network includes such a compositional space of meaningful attractors, arising from the interplay of trained patterns, we refer to the network as possessing *articulated attractors*.

Previous work has shown that standard connectionist learning techniques spontaneously give rise to such structured attractors when recurrent connections are present at the output layer of the network, but they *fail* to learn such attractors over units which do not receive a direct teaching signal (Noelle and Cottrell, 1996). In other words, networks with recurrent connections only at a hidden layer cannot learn articulated attractors from backpropagated error. This means that recurrent backpropagation networks cannot learn to actively maintain a componential representation without having the structure of that representation *explicitly* specified by a teacher or by the environment.

This result poses a problem for cognitive models which employ attractor networks as a form of working memory, since it shows that such networks cannot learn internal representations for a task and simultaneously learn to robustly remember those representations over time. Even models which involve the learning of attractors at an output layer may be challenged by this result, as the output representation of such models is often conceptualized as a learned internal representation in a larger cognitive system. For example, the recurrent phonological output layer of some word naming models (Plaut and McClelland, 1993) is trained with an explicitly structured teaching signal, despite the fact that phonology is thought to involve a learned internal coding scheme.

This paper discusses some preliminary efforts to discover connectionist learning methods which will give rise to articulated attractors over learned internal representations. Specifically, two unsupervised learning rules are presented, simulated, and analyzed.

A Structured Memory Task

To facilitate analysis, we focused on an extremely simple task. Each attractor network was to learn to act as a kind of working memory, maintaining a presented pattern of activation indefinitely, given only a brief initial exposure to that pattern. Furthermore, the networks were to discover regularities in the corpus of presented patterns, identify the structure of well formed patterns, and use that knowledge to "clean up" noisy patterns, thereby enforcing "well formedness" constraints. This task is depicted schematically in Figure 1. Note that the input pattern is made available to the network for the first few time steps only, requiring the network to both "clean up" and remember the pattern over time.

Specifically, each network was briefly presented with an encoding of a simple slot-filler structure. The network was to filter out any noise in this representation and continuously present the resulting clean pattern at the network's output, even after the input was removed. Thus, the network needed

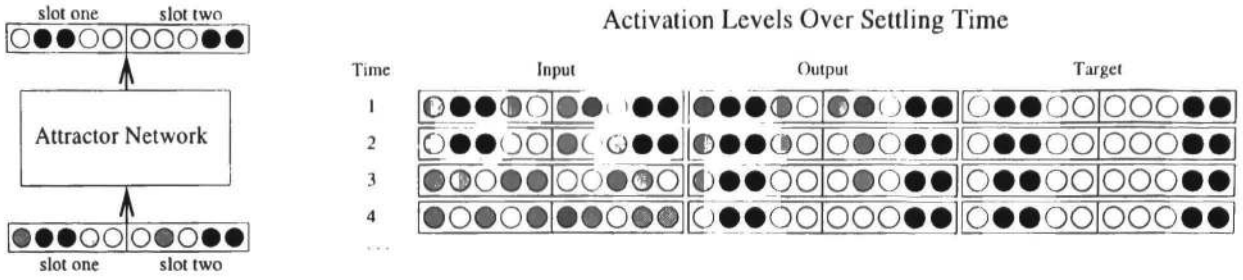

Figure 1: The Slot-Filler Structured Memory Task

to learn a distinct stable attractor for each valid slot-filler structure. Each input pattern represented a structure containing two slots, each holding exactly one of five distinct fillers. The contents of the slots were considered independent, with the specific filler in one slot in no way constraining the filler for the other. The fillers for each slot were encoded over a 5 element binary vector, resulting in a 10 element vector for the entire structure. Each of the five fillers was encoded as a unique pair of adjacent "on" elements.[1] Thus, well formed patterns included exactly two of the first five elements "on" and exactly two of the last five elements "on". With five possibilities for each slot, there were only $5^2 = 25$ "well formed" patterns out of the $2^{10} = 1024$ possible binary input vectors.

These simulation experiments focused on systematic generalization. The networks were to learn an attractor for every valid slot-filler structure, given training on only a fraction of the valid patterns. In order to investigate such generalization, each network was explicitly trained on some subset of the well formed input patterns. Once trained, each network was then tested on *all* valid slot-filler representations, and the number of attractors corresponding to these valid patterns was determined. The dynamic behavior of each trained network was also examined to locate any spurious attractors corresponding to ill formed patterns.

In order to generalize in a systematic way, the network needed to learn two interrelated properties of these input patterns. First, it needed to recognize that the patterns consisted of two *independent* slots. Second, the network needed to identify the structure of the filler patterns, constructing attractors for patterns involving two adjacent active units but not for other cases (e.g., 3 of the 5 units in a group active). It is important to note that a localist code (i.e., 1 of 5 units "on" for each filler) was *not* used here. Such a localist code would make the identification of the two independent slots very difficult. If such a code were used and the training set of the network left out but a single valid pattern — a single pair of slot fillers — the network would discover that the two units for that pair were perfectly anticorrelated in the training set and would hinder the formation of an attractor for the novel pattern involving that pair of fillers. Thus, for a network to discover the independence of the two slots by attending to the pairwise statistics of pattern element values, some form of coarse coding of fillers was needed.

Even with such coarse coding of fillers, a training set consisting of only a few valid patterns displays very little inherent structure. As training sets get larger, however, the underlying slot-filler structure of the input patterns becomes evident. In order to examine this dependence on training set size, networks were trained with varying sized collections of well formed patterns. Each training set contained at least five patterns, as this was the minimum number needed to present each filler pattern to the network at least once. The largest training set consisted of all 25 well formed patterns. The frequency of each filler value in each training set was balanced as much as was possible given the small size of the training sets. During training, zero mean independent Gaussian noise with 0.025 variance was added to each input element. Noise was resampled on every time step, and it persisted even after the input pattern was removed. Network output targets consisted of the "clean" patterns over the entire course of network settling, as shown in Figure 1. A settling period of 10 time steps was used during training, and 100 time steps were used during testing. In our initial simulations, the input pattern was presented for only a single time step.

Problems With Internal Representations

It has previously been shown that network learning techniques based on the backward propagation of error, such as *backpropagation through time (BPTT)* (Rumelhart et al., 1986), can learn appropriate articulated attractors for this structured memory task, but only if an external teaching signal is provided directly to the processing elements which are recurrently connected (Noelle and Cottrell, 1996). Consider the simulation results plotted on the left side of Figure 2. In this graph, the size of the training set is displayed along the horizontal axis, and the resulting number of valid attractors learned is specified vertically. If a given network failed to generalize, only learning attractors for the training set patterns, then data should fall along the displayed unit slope reference line. But this graph displays substantial generalization. Training sets consisting of 15 or more of the 25 valid slot-filler patterns resulted in networks which generalized to all 25 of them. The attractor network exhibiting this good generalization performance contained recurrent connections between the units of its output layer and was trained using BPTT. This means that the teaching signal strictly specified the structure of the activation vectors over which attractors were to form. This external structuring signal is not directly available to recurrent weights, however, when only hidden units are recurrently connected. This results in a failure to generalize when attractors are required to form over an internal representation — over the hidden layer of the network —

[1] Each group of five elements was conceptualized as forming a closed loop for the purposes of determining adjacency.

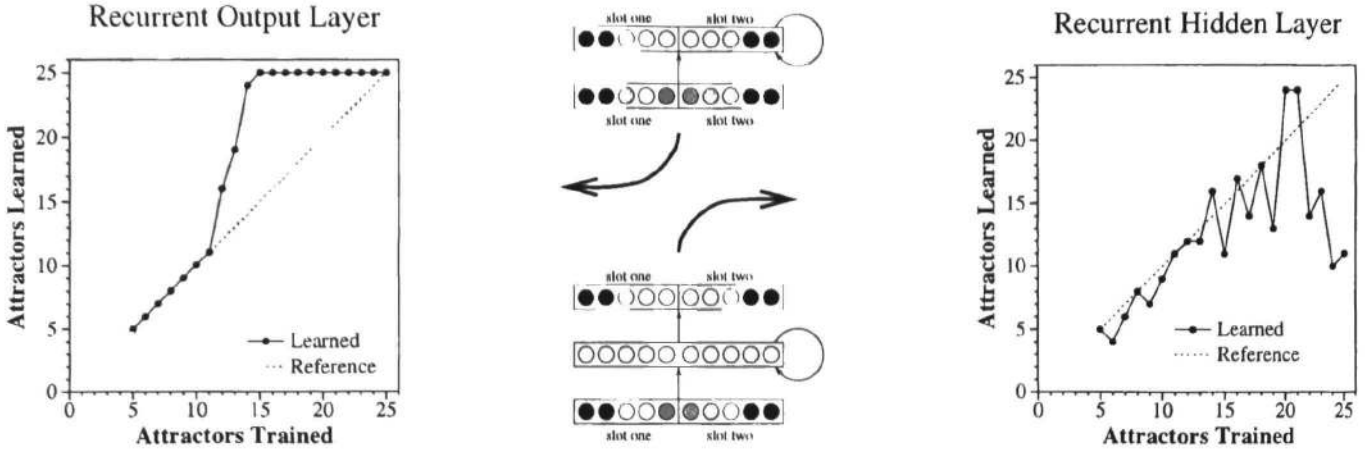

Figure 2: Generalization Performance of BPTT With Recurrence at the Output Layer vs. the Hidden Layer

as shown in the graph on the right side of Figure 2. Without the teaching signal directly enforcing a compositional structure on the recurrently connected layer of units, the network can barely construct stable attractors for the training patterns.

Detailed analyses of these poorly generalizing networks suggested four main reasons for their failures:

(1) *Localist Coding*. The network sometimes developed hidden layer representations for individual slot fillers which were approximately localist in nature. Even though the inputs and output targets were coarse coded, the hidden layer used a single unit to represent a given filler value. This introduced the previously mentioned problem of localist coding to the recurrent weights. The recurrent connections would learn to inhibit novel, yet valid, slot value pairs.

(2) *Small Weights Have Large Effects Over Time*. In typical training sets, each slot filler is somewhat anticorrelated with all of the fillers for the other slot, because individual fillers appear relatively rarely. This leads to small amounts of inhibition between the representations for fillers in opposite slots. This slight inhibition typically poses no problem for the maintenance of a pattern over the course of the 10 time steps that the network is allowed to settle during training, but extending the settling time beyond 10 steps (as is done during testing) allows the network to drift, slowly losing the activation pattern that it had maintained. In other words, the network does not actually learn stable attractors for patterns, but, instead, learns to drift away from well formed patterns slowly enough so as to not impact network error noticeably during the 10 settling time steps of training.

(3) *The Error Gradient Disappears Near Solutions*. Another reason that these networks fail to form stable attractors is that such attractors typically require a particular configuration of large weight values. Starting with small random weights, the connection strengths grow towards the needed values, but network error decreases as those weights are approached, causing the weights to change less and less. Thus, the network remains in a region of weight space corresponding to the "slow drift" strategy, never quite arriving at the magnitude of weights required for stable attractors.

(4) *Polarization*. The first behavior that these networks acquire is the reproduction of the input pattern across the output units during the initial time period when the input is still directly available. This occurs without much use of the recurrent weights. The hidden layer representation that forms early in training tends to be distributed and graded. The graded nature of this representation makes it hard to form the needed attractors. It is much easier to learn stable fixed-point attractors involving extreme activation values rather than involving activity levels in the linear range of the unit activation function (Noelle et al., 1997).

We sought to modify the learning procedures of our networks so as to alleviate these problems, and we found two very distinct and somewhat successful mechanisms for learning articulated attractors over internal representations.

Asymmetric Hebbian Learning

Our approach involves learning the recurrent connection weights using an unsupervised learning method. The idea is to restrict the pattern of connectivity at the hidden layer so as to promote the formation of attractors, and then learn the recurrent weights in a manner which is independent of the teaching signal being provided at the output of the network. A sketch of the network architecture used appears in Figure 3. Notice that the hidden layer in this architecture contained 20 units, and a new layer of 20 "interneurons" was added. In the network diagram, the thin solid arrows represent sparse random connectivity between the layers, with connection weights bound to be non-negative. The dashed lines specify one-to-one patterns of connectivity, with each hidden unit possessing a non-negative connection to itself and a fixed positive connection to its "partner" unit in the interneurons layer. The bold line from the interneurons layer to the hidden layer represents nearly complete interconnectivity, with weights *bound to be non-positive*. Each interneuron projected to all hidden units except for its hidden layer "partner". Thus, the architecture provided each hidden unit with a positive self connection to maintain it's own activity and a partner interneuron to inhibit other hidden units.

The feedforward connections, from input to hidden and hidden to output, were trained using the standard BPTT algorithm. The self connections on each hidden unit were also learned using the backpropagated error signal, as were the unit biases. The one-to-one connections between the hidden units and the interneurons had fixed weights, forcing

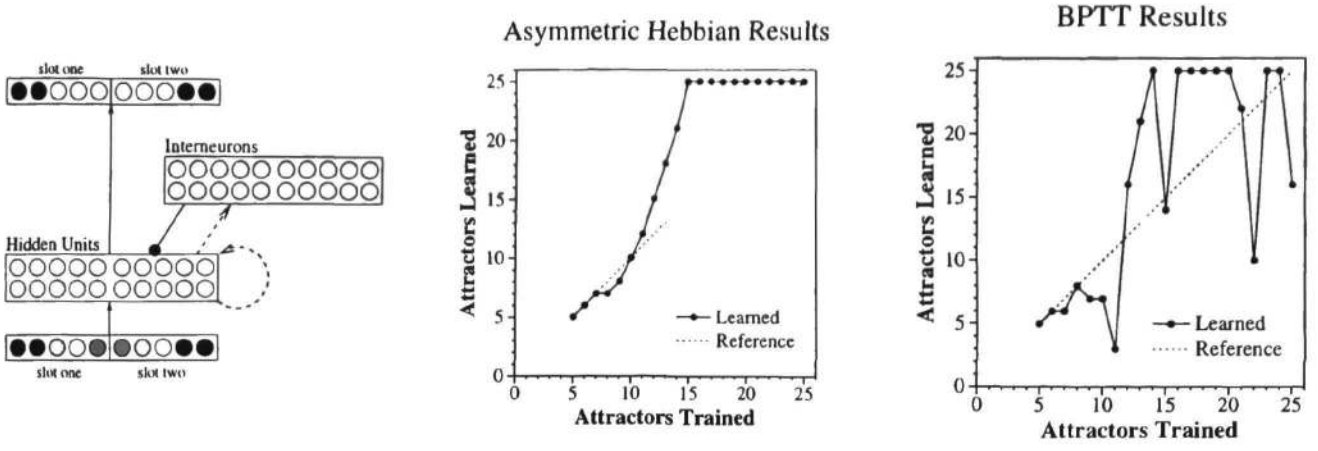

Figure 3: Network Architecture & Results for Asymmetric Hebbian Learning Technique Along With BPTT Control

the partner interneuron to come on whenever its hidden unit became active. The inhibitory connections projecting from the interneurons used the following new asymmetric Hebbian learning procedure:

- If the interneuron is in the upper half of its activation range, and the hidden unit it projects to is also highly active, then decrease the magnitude of the connecting weight by a large amount, proportional to the product of their activities.
- If the interneuron is highly active, and the hidden unit it projects to is not, then lower the connecting weight value slightly, proportional to the product of the interneuron activity and one minus the hidden unit activity.

This learning rule tends to zero out weights associated with pairs of hidden units that have been active together, while allowing hidden units that are never co-active to acquire negative connections between them, mediated by an interneuron.

This network architecture and learning technique avoids the problem of localist codes at the hidden layer (problem 1, above) by using sparse random patterns of connectivity between the input and hidden layers and between the hidden and output layers. Each unit in these layers received input from only five randomly selected units from the preceding layer. Since the inputs and targets were coarse coded, this pattern of connectivity encouraged the appearance of a similar coarse code at the hidden layer. By pushing weights strongly towards zero if a co-occurrence of activity is detected, this scheme also avoids the problem of small inhibitory weights (problem 2, above). The problem of the disappearing gradient (problem 3, above) is alleviated by using a Hebbian scheme which is insensitive to the backpropagated error. Lastly, the problem of developing polarized hidden layer representations (problem 4, above) is handled by initializing the hidden unit self weights to large values, encouraging the network to make use of the extreme hidden unit activations which naturally arise from such positive feedback.

Simulations of this learning mechanism were conducted with the self connection weights initialized to values uniformly sampled from $[5.5, 6.5]$, feedforward weights sampled from $[0.0, 0.2]$, the weights leading to the interneurons fixed at 6, and the weights from the interneurons initialized to zero. The unit biases were initialized to values sampled from $[-2.5, -3.5]$ and were bound to be non-positive. The bias values on the interneurons, however, were fixed at -3. All weights trained using BPTT, including the bias weights, used a learning rate of 0.1. The weights trained using the asymmetric Hebbian rule used a learning rate of 0.002 when raising weight values towards zero and a rate of 0.0001 when making weights more negative. A weight decay regularization, reducing each weight magnitude by 0.001% on each weight update, kept the Hebbian weights from growing without bound. All processing elements used the logistic activation function, bounding output activity between 0 and 1.

The results of these simulations are shown in the middle of Figure 3. Notice that the network successfully generalized to all 25 patterns for training sets of size 15 and larger. An analysis of the dynamics of these networks revealed more than these 25 stable fixed-point attractors, however. A total of 96 attractors appeared for ill formed input vectors, as well. These spurious attractors had an interesting structure, however. They all consisted of cases in which a group of five units encoding a slot had either one element active or no elements active. In other words, the network successfully learned that the two slots were independent, but came to treat both "no units on" and "one unit on" as valid filler values. In essence, the trained network treated the two adjacent active elements in every valid slot filler as independently modifiable features. This makes sense considering that the recurrent connections in this network could only encode independence (zero weight) or anticorrelation (negative weight).

One might wonder how much of the success of this network was driven by the asymmetric Hebbian learning algorithm and how much relied on the restricted architecture and weight initializations. To investigate this, networks which were identical in architecture and initial weight configuration were trained using BPTT. The weights from the hidden layer to the interneurons remained fixed, but the connections projecting from the interneurons were trained used a backpropagated error signal. The results of these simulations are shown in Figure 3, on the right. While this diagram appears to display good generalization for the larger training set sizes, it hides a multitude of sins. Every training set which resulted in attractors for all 25 well formed patterns also resulted in a

stable attractor for *any* of the 1024 possible binary vectors. It appears as if the network architecture and weight initialization scheme provided the means for retaining patterns over time, but the asymmetric Hebbian learning rule provided the means for discerning well formed patterns.

While this learning method generalizes much more effectively than any investigated method involving backpropagated error, it still has a number of problems. In an effort to avoid small negative weights, the learning rule ignores less than perfect correlations between unit activation levels. This forces the network to treat the components of coarse coded filler values as independent (zero weight between them) when they are not. One might imagine augmenting these networks with separate recurrent weights which are bound to be *non-negative*, hoping that these weights would capture the positive correlations between the components of filler representations. Unfortunately, the inclusion of such weights reintroduces the problem posed by small negative weights — such positive weights interfere with the formation of stable attractors. As settling time increases, such weights tend to result in virtually *all* of the hidden units becoming active. Another weakness of this asymmetric Hebbian learning strategy is that it requires the formation of an appropriate coarse code at the hidden layer. The feedforward connections must be tightly restricted in order to ensure that the elements of the input filler representation remain separated in the hidden layer representation. Thus, while this learning method allows articulated attractors to form over internal representations, it only works for a restricted class of such representations.

Competitive Inhibition Learning

In hopes of overcoming the need for tight restrictions on the feedforward weights, our attention shifted toward finding a method for learning the recurrent weights which could effectively handle localist encodings of filler values. As previously discussed, learning to dissociate the two slots is impossible with a localist code *if pairwise correlations in unit activity are used to make the dissociation.* A training set communicates more information than just pairwise statistics, however. If an input unit, A, has been seen co-active with another input element, B, then, under a localist coding scheme, these two elements must code for filler values in separate slots. If a third input element, C, is never seen co-active with either A or B, standard learning mechanisms would come to build inhibition between C and the other two elements. But we know, given a localist code, that C only participates in one slot, so it should only inhibit either A or B, but *not both*. Thus, by attending to these ternary relationships, we can start to identify alternative fillers for a single slot as opposed to novel pairs of fillers across slots.

For this new learning approach we used the same network architecture as before, with one exception. Since localist encodings of filler values were actually desired at the hidden layer in this case, sparse random connectivity in the feedforward connections was not needed. Instead, each unit in the input layer was connected to every hidden unit, and each hidden unit was connected to every output. Once again, the feedforward weights were bound to be non-negative and were trained using standard BPTT. As before, the one-to-one weights projecting to the interneurons were of fixed positive values. The inhibitory weights projecting from the interneurons were modified according to a competitive learning rule:

- If a hidden unit is in the upper half of its activation range, and an interneuron projecting to it is also highly active, then decrease the magnitude of the connecting weight by a large amount, proportional to the product of their activities.
- If a hidden unit is in the lower half of its activation range, find all highly active interneurons projecting to it and identify the interneuron which inhibits this hidden unit most strongly. This interneuron is the *winner* of the competition. Its weight is made more negative by a moderate amount, proportional to the product of the activity levels of the two units. Active interneurons which lose this competition have the magnitude of their weights decreased by a large amount, proportional to the product of activations.
- If a hidden unit is inactive, but no interneurons are strongly inhibiting it, it needs to join a new group of mutually inhibiting units. This is done by identifying the strongest weight to this hidden unit from the interneurons, and raising this weight towards zero while making all other weights slightly more negative.

The first part of this rule makes sure that only one hidden unit is active in any group of mutually inhibitory units. The second part further separates the slots by ensuring that a unit is never inhibited by units belonging to different slots. The third part keeps the network from creating too many separate groups of mutually inhibitory units by insisting that at least one unit be active in each group.

We found that standard BPTT training did not produce sufficiently strong positive self connections, so a different learning algorithm was used on the self weights. Whenever a hidden unit was in the upper half of its activation range and experienced a negative change in activity level, it effectively received an error signal driving the unit towards its maximum activation level, resulting in an increase in the self weight.

Simulations involving this learning mechanism initialized hidden unit self weights to values uniformly sampled from $[3.5, 4.5]$ and feedforward weights sampled from $[0.0, 0.2]$. The weights leading to the interneurons were fixed at 8, and the weights from the interneurons were initialized to zero. All unit biases were initialized to -2, though these biases were adapted via BPTT in all units except the interneurons. All weight values trained using BPTT used a learning rate of 0.1. The hidden unit self connections used a learning rate of 0.01. The competitive inhibition rule essentially set weights mediating between co-active hidden units to zero, used a learning rate of 0.01 for "winning" interneurons and a rate of 0.1 to reduce the magnitude of weights from "losing" units. When an inactive hidden unit failed to be inhibited by any interneuron, its strongest weight was raised with a learning rate of 0.1 while other weights were made more negative with a learning rate of 0.001. The weights projecting from the interneurons were bound to be no lower than -4. Lastly, to promote the use of a localist code at the hidden layer, the feedforward weights were subjected to the "weight elimination" procedure (Weigand et al., 1991) with a decay rate of 0.001.

Since this learning method was expected to thrive on localist encodings, the simpler localist encoding of slot fillers

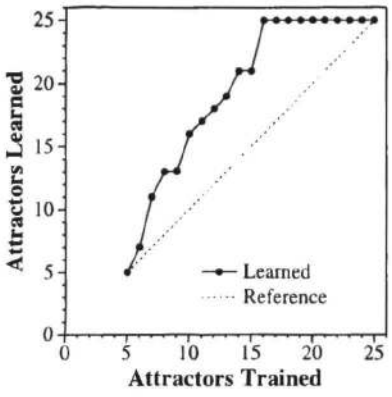

Figure 4: Results for Competitive Inhibition Learning

was used in the inputs and targets for these simulations. Each filler was encoded as one unit active out of the group of five. Also, the input pattern was presented for 4 initial time steps in these simulations rather than one. This aided the learning of strong self weights at the hidden layer.

The results of these simulations are shown in Figure 4. Notice that the networks generalized perfectly for training sets of size 16 and greater. As before, the networks overgeneralized, producing attractors for 11 ill formed patterns. Once again, these spurious attractors had an interesting structure, always involving a slot with no units active. The networks were able to identify the two independent slots, but came to treat "no units on" as a valid filler value.

While this learning method overcame the requirement for a coarse coding of slot fillers in the hidden layer, it introduced its own constraints on the internal representation in use. Specifically, the hidden layer representation of a slot filler had to be localist in nature. If there were multiple hidden units encoding a particular filler, only one of these units would be recruited to participate in the mutual inhibition between fillers for this slot. The unrecruited units would either come to participate in other groups of mutually inhibiting units, making generalization to novel combinations of slot fillers less likely, or they would "stand on their own", potentially encouraging overgeneralization.

Conclusions

We have presented two new methods for the learning of articulated attractors over internal representations. These techniques gain their strength in large part from the use of a restricted network architecture. These learning methods also benefit from being unsupervised — from basing weight updates on an inherent inductive bias rather than on a backpropagated error signal. Such a bias appears necessary for the formation of true stable attractors at hidden layers.

These two learning rules do not completely resolve the problem of hidden layer articulated attractors. Both methods are only successful when the network is constrained to develop certain kinds of internal representations. The asymmetric Hebbian learning method requires a coarse coding of fillers at the hidden layer, while the competitive inhibition method works best with a localist code. Future work will focus on modifying these learning rules so as to allow for more general distributed representations at the hidden layer.

It might be the case, however, that active maintenance of a pattern of activation requires some restricted coding scheme. Indeed, some researchers have argued that the working memory functions of dorsolateral prefrontal cortex require representations involving isolated components (Cohen et al., 1996). Perhaps the key to learning articulated attractors at a hidden layer, then, is learning an internal representation with an appropriate componential structure.

Acknowledgements

This work was supported, in part, by the NIH through a National Research Service Award (# 1 F32 MH11957-01) from the National Institute of Mental Health, awarded to the first author. We extend our thanks to James L. McClelland, Randy O'Reilly, and the members of the CMU *PDP Research Group* for their comments and suggestions. Thanks are also due to three anonymous reviewers for their helpful advice concerning the clear presentation of this research.

References

Cohen, J. D., Braver, T. S., and O'Reilly, R. C. (1996). A computational approach to prefrontal cortex, cognitive control and schizophenia: Recent developments and current challenges. *Philosophical Transactions of the Royal Society of London B*, 351:1515–1527.

Mathis, D. W. and Mozer, M. C. (1995). On the computational utility of consciousness. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances In Neural Information Processing Systems 7*, pages 11–18, Denver. MIT Press.

Noelle, D. C. and Cottrell, G. W. (1996). In search of articulated attractors. In Cottrell, G. W., editor, *Proceedings of the 18th Annual Conference of the Cognitive Science Society*, pages 329–334, La Jolla. Lawrence Erlbaum.

Noelle, D. C., Cottrell, G. W., and Wilms, F. R. (1997). Extreme attraction: On the discrete representation preference of attractor networks. In Shafto, M. G. and Langley, P., editors, *Preceedings of the 19th Annual Conference of the Cognitive Science Society*, page 1000, Stanford. Lawrence Erlbaum.

Plaut, D. C. and McClelland, J. L. (1993). Generalization with componential attractors: Word and nonword reading in an attractor network. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, pages 824–829, Boulder. Lawrence Erlbaum.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge, Massachusetts.

Weigand, A. S., Rumelhart, D. E., and Huberman, B. A. (1991). Generalization by weight-elimination with application to forcasting. In Lippman, R. P., Moody, J., and Touretzky, D. S., editors, *Advances In Neural Information Processing Systems 3*, pages 875–882, Denver. Morgan Kaufmann.