Adaptability of the Backpropagation Procedure*

Nathalie Japkowicz
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
nat@cs.dal.ca

Stephen José Hanson
Department of Psychology
Rutgers University
Newark, NJ 07102, USA
jose@psychology.rutgers.edu

Abstract

Possible paradigms for concept learning by feedforward neural networks include discrimination and recognition. An interesting aspect of this dichotomy is that the recognition-based implementation can learn certain domains much more efficiently than the discrimination-based one, despite the close structural relationship between the two systems. The purpose of this paper is to explain this difference in efficiency. We suggest that it is caused by a difference in the generalization strategy adopted by the Backpropagation procedure in both cases: while the autoassociator uses a (fast) bottom-up strategy, MLP has recourse to a (slow) top-down one, despite the fact that the two systems are both optimized by the Backpropagation procedure. This result is important because it sheds some light on the nature of Backpropagation's adaptative capability. From a practical viewpoint, it suggests a deterministic way to increase the efficiency of Backpropagation-trained feedforward networks.

1 Introduction

Possible paradigms for concept learning include discrimination and recognition. In feedforward neural networks, discrimination can be implemented by teaching a "pyramid"-shaped discrimination network to return a "1" or a "0" at the output layer according to the class of the training instance. Recognition, on the other hand, can be implemented using an autoassociator for learning how to reconstruct training instances of the concept class at the output layer.

A particular aspect of discrimination- versus recognition- based classification within the feedforward scheme is that the recognition-based system was shown to learn an interesting test problem much more efficiently than the discrimination-based system.[1] While this difference in efficiency may be caused by an increase or decrease in problem complexity (due to the change of representation in effect when going from one paradigm to the other), it may also be caused by specific phenomena associated with the Backpropagation procedure. In particular, Backpropagation may assume different modes of generalization—which could present variable degrees of efficiency—when faced with distinct situations.

The purpose of this paper is to find out whether, indeed, Backpropagation does resort to different generalization strategies when applied to such problems through the discrimination or recognition paradigm, or whether the difference in efficiency witnessed is solely caused by a change of representation.

The remaining sections of this work are organized as follows: Section 2 describes the two approaches to concept-learning and compares their efficiency on our test problem. Section 3 represents the core of this paper. It begins by discussing the experiments designed to analyze and compare the operation of the two systems, it then presents the results of this analysis, and it concludes with a discussion of our results. Section 4 summarizes our findings. Additional details can be found in [3].

*The research described in this paper was conducted at Rutgers University, Princeton University and the Ohio State University.

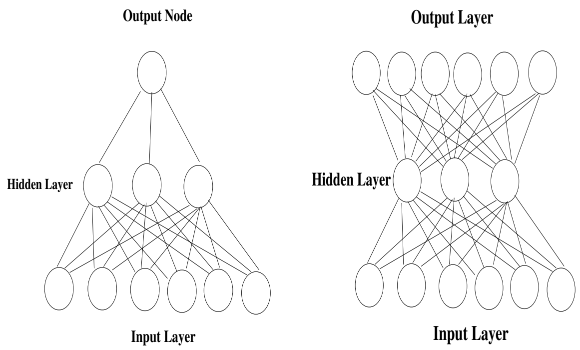
[1] Another interesting aspect of this dichotomy is that the recognition-based system can be significantly more accurate than the discrimination-based one (see [4], [3], [2]).

2 Discrimination versus Recognition

We begin this section by describing the two classification systems contrasted in this study and we then present the efficiency results they obtained on an abstract problem which displays nonlinearly-separable and multi-modal properties.

2.1 Systems Description

A formal definition of the concept-learning problem typically involves an input vector x, and a response vector y which are such that the pair (x, y) belongs to some unknown joint probability distribution, P. The goal of concept learning is to induce a function $f(x)$ from a set $(x_1, y_1), (x_2, y_2), \dots\ (x_N, y_N)$ of training examples, so that $f(x)$ approximates y. Multi-Layer Perceptrons are a particular type of neural networks that are capable of approximating, as closely as desired, any such continuous functional mapping f. In this paper, we restrict our attention to Multi-Layer Perceptrons with a single hidden layer, i.e., with two layers of weights. Two examples of such feedforward neural networks are given in Figure 1. In figure 1(a), the input layer contains 6 units, the hidden layer, 3 units, and the output layer, a single unit. Figure 1(b) is similar except for the output layer which contains 6 units. All the networks considered in this paper have nonlinearities in both their hidden and output layers and these nonlinearities are implemented by the Sigmoid function.



(a) MLP **(b) Autoassociator**

Figure 1: Examples of Feedforward Neural Networks: (a) The MLP Network; (b) The Autoassociator

The network of figure 1(a), which we refer to as "MLP" or "the MLP network", is the most commonly used network in the field of connectionist systems. It is typically used to implement a discrimination-based approach to classification. Using this scheme, each element y_i of y, the response vector, takes on a value of "1" or "0" which is interpreted as to mean that x_i, the input vector associated with y_i belongs or does not belong to the concept class of the problem. After training the network to approximate the function $f(x)$ from which the training examples are believed to have been generated, it is expected that, if the training set was appropriately designed, the network will be able to compute the appropriate label ("1" or "0") for any input vector of size d, the size of the input layer, even if that vector did not appear in the training set.

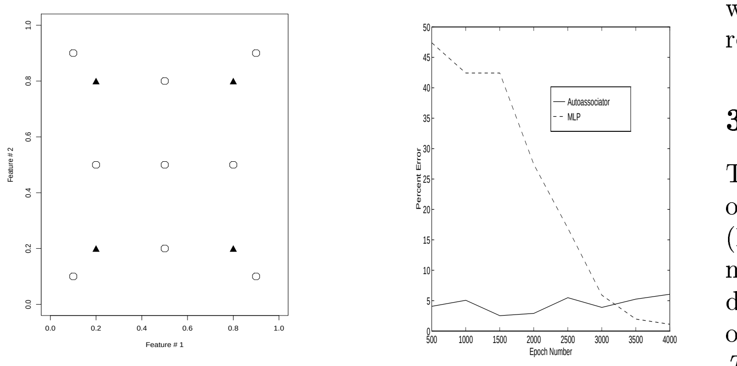
The network of figure 1(b) is not as universal as the MLP network, but it has been used previously to implement a compression scheme. As well, and this is the application of interest here, it can be used to implement a recognition approach to concept learning as proposed in [1]. In this approach, each response vector y_i of y is set to x_i, the corresponding input vector. In other words, the network is trained to reproduce the input at the output layer. Such a network is called an autoassociator. For concept-learning to take place, the network is trained to reconstruct concept data only and classification is performed on a new vector x_{Test} by comparing its *reconstruction error*[2] to a threshold, and assigning it to the concept class if the reconstruction error is smaller than this threshold, and to the other class otherwise. The idea behind this recognition-based classification scheme is that since the autoassociator is trained to compress and decompress examples of the concept class only, when tested on a novel data point, it will compress and decompress it appropriately if this example belongs to the concept class, but it will not do so appropriately if the example does not belong to the concept class.

2.2 Efficiency Results

The learning performance of the two systems just described was tested on instances of the problem depicted in Figure 2(a). In this domain—inspired by real-world domains (see [3])—, examples of the concept are represented by black triangles

[2] The reconstruction error corresponds to $\Sigma_{j=1}^{d}[x^j_{Test} - g(x^j_{Test})]^2$, where g is the function implemented by the autoassociator and x^j_{Test} and $g^j(x_{Test})$ represent input unit j and output unit j of the network, respectively.

while counter-examples are represented by white circles. Each triangle and circles, actually, represents the mean, μ, of a 50 point cluster normally distributed around it with a variance of $\sigma^2 = 0.01$.[3]



(a) Test Domain **(b) Classification Error Rates**

Figure 2: (a) Test domain used to compare MLP with the autoassociator; (b) Classification error rates obtained by the two systems on domain (a). MLP's performance is represented by a broken curve while the autoassociator's is represented by a full one.

The two systems were trained using the Backpropagation procedure with 16 hidden units (a capacity found to be optimal for the two systems) and the standard learning rate of 0.05 and momentum of 0.9. The average classification error rates they obtained over five trials are plotted in Figure 2(b) which displays the two system's learning curves recorded every 500 epochs for up to 4000 epochs.

Although the graph of figure 2(b) shows that for this particular domain, MLP is slightly more accurate than the autoassociator, it also indicates that the autoassociator converges to a solution much faster than MLP: while the autoassociator has already converged by the first epoch recorded in the graph (epoch 500), MLP takes over 3000 epochs to learn how to classify the data properly. The purpose of the next section is to explain this drastic difference.

[3] More precisely, Figure 2(a) shows that, the means of the concept components are located at points $(.2, .2)$, $(.2, .8)$, $(.8, .2)$, and $(.8, .8)$ and the means of the counter-concept components are located at points $(.5, .5)$, $(.1, .1)$, $(.1, .9)$, $(.9, .1)$, $(.9, .9)$, $(.5, .2)$, $(.2, .5)$, $(.5, .8)$, and $(.8, .5)$.

3 Experiments and Results

This section is divided into three parts. We first present the methodology we employed to explain the difference in efficiency between the two classification schemes; we then report our results; and we conclude the section with a discussion of these results.

3.1 Methodology

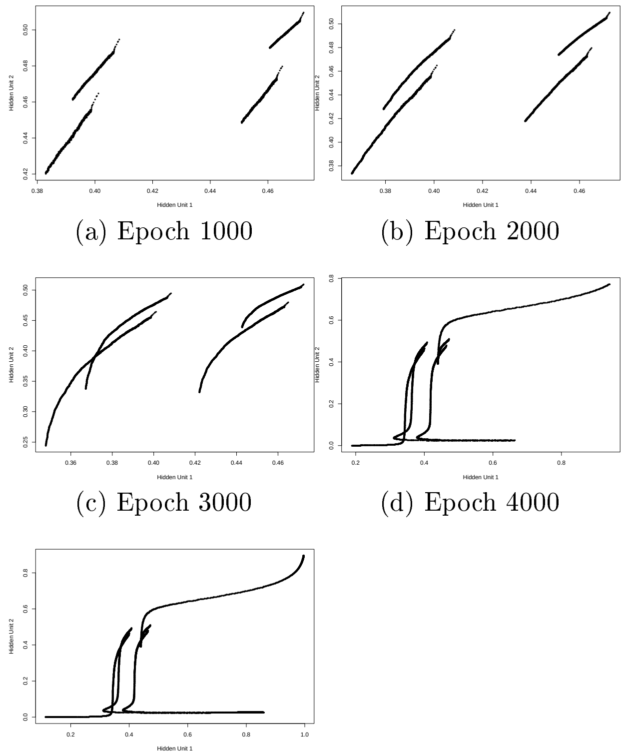
The methodology used in this study consists of building the Hidden Unit Activation Planes (HUAP) of the networks and plotting them. This methodology is inspired from the study of natural dynamical systems (e.g., the behavior of biological organisms) and uses elements of *Stability Analysis Theory.*

In more detail, the method consists of plotting the activation values of the hidden units for each input pattern as a function of one another. Since it is based on visualization, the method is restricted to feedforward networks of two hidden units and the x-axis of the HUAP records the activation value of the first hidden unit while its y-axis records that of the second. Within this two-dimensional space, if the HUAP is plotted at a single epoch, then each input pattern is represented by a point, but if it is plotted over several epochs, then each input pattern is represented by a path which reveals the evolution over time of the relative activation values of the two hidden units for that input. Convergence of the network is observed when the paths have stopped growing.

Like in stability analysis theory, global properties of the system can be established by observing the paths formed in a HUAP. In particular, the notions of *stable* and *unstable attractors (or fixed-points)*—where stable attractors refer to points towards which a trajectory converges indefinitely, and unstable attractors refer to points towards which a trajectory converges for some time and then diverges suddenly—will prove very useful.

3.2 Results

Because the HUAP method is restricted to networks of two hidden units and the domain of Figure 2(a) requires more than two hidden units both for discrimination and recognition, we applied the HUAP methodology to a simpler domain, namely, the *X-OR problem.* In this domain, each



(a) Epoch 1000 (b) Epoch 2000

(c) Epoch 3000 (d) Epoch 4000

(e) Epoch 5000

Figure 3: HUAPs for MLP. Note the difference in scale for each graph.

data point is composed of two features taking the values of "0" or "1" and an input is considered part of the concept class if one of its features is a "1" and the other, a "0", while it is part of the counter-concept class, otherwise.

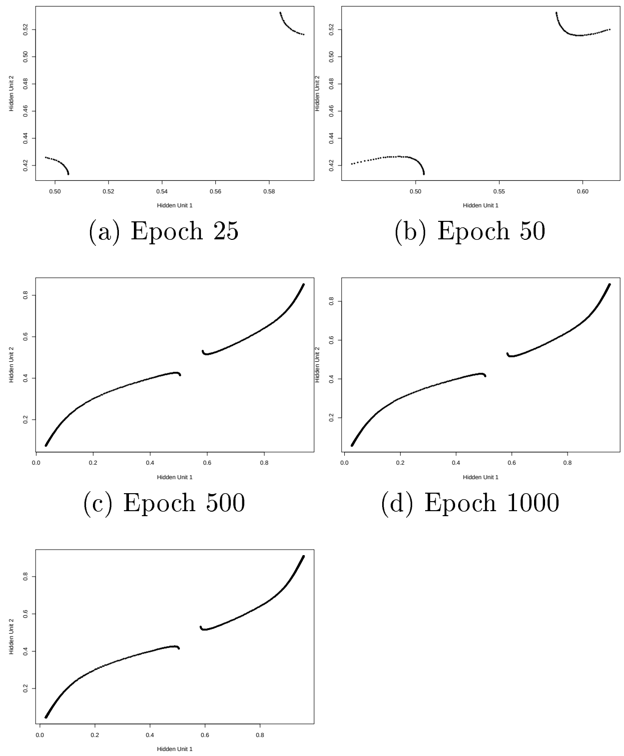
The results of monitoring the hidden unit activation values of the two systems on the X-OR domain are plotted in Figures 3 and 4. Figure 3 shows what happens on a 5000 epoch run of the MLP network. The figure is composed of 5 different graphs, each illustrating the state of the HUAP at a given epoch. Figure 4 is similar but for the autoassociator and on a run of 2000 epochs. In the graphs of Figure 3, four trajectories are visible since, as discussed in Section 2.1, the MLP network is trained on both the two concept and the two counter-concept examples contained in the X-OR domain. In the graphs of Figure 4, on the other hand, only two trajectories are visible since the autoassociator is only trained on the two concept examples of the X-OR domain.

Again, in each of these graphs, the trajectories displayed correspond to the change in state of the network in relation to a single data point as time progresses, and, therefore, each input datum is responsible for a different trajectory in that space.

The graphs of Figure 3 show that for at least 3000 epochs, the trajectories caused by each of the four data points input to the MLP network are parallel and tend towards the same general area. In other words, all input values are originally located in the *basin of attraction* of the same single attractor (i.e., the area influenced by a single stable attractor) and thus follow the same direction. By epoch 4000, however, the picture is very different. At this point, the single trajectory previously observed has divided into three separate trajectories, one tending towards the upper right corner of the HUAP, and the other two directed toward its lower left and right corners. More specifically, this plot shows that after epoch 3000, the input data to the MLP network have been divided into three sets: one set containing two input data of the same class and triggering a single hidden unit activation pattern, and another two sets, each containing a single input datum of the opposite class, and triggering different hidden unit activation patterns. The figure displayed for epoch 5000 is very similar to the one displayed for epoch 4000, showing that the network has reached a stable solution by epoch 4000. Translated into dynamical system jargon for neural networks, these plots suggest that the HUAP of the MLP network contains both unstable and stable attractors. It contains a single unstable attractor which corresponds to the point towards which all the trajectories were directed until some time between Epoch 3000 and 4000; and it contains three stable attractors which correspond to the three separate points towards which the trajectories tend after epoch 4000. The stable attractors are weaker than the unstable one (in terms of attraction capability) because they are noticed by the network after 3000 epochs have elapsed whereas the unstable attractor was immediately spotted.

The graphs of figure 4 display a very different pattern. They show that from very early on, the trajectories of the two patterns input to the autoassociator are differentiated. They both begin in the vicinity of the center of the HUAP and start by following an inward direction (Plot 4(a)), but then, quickly reverse course and tend towards the upper right and lower left corners of the HUAP (Plots 4(b), (c), (d) and (e)). The two trajectories are perfectly symmetrical and they tend, respectively, towards two of the furthest points in the

HUAP. As well, convergence occurs very quickly since by epoch 500, the trajectories have already reached their final destinations. Again, these plots suggest that the HUAP of the autoassociator contains both unstable and stable attractors. It contains one unstable attractor towards which the two trajectories gravitate during the first 25 epochs of the learning process and two stable attractors that are reached by the two trajectories as early as epoch 500. The unstable attractor is very weak relative to the stable ones, since it is abandoned very early on in favor of the stable ones.



(a) Epoch 25 (b) Epoch 50

(c) Epoch 500 (d) Epoch 1000

(e) Epoch 2000

Figure 4: HUAPs for the autoassociator. Note the difference in scale for each graph.

3.3 Discussion

In summary, our experiments have suggested that:

1. The HUAP of the MLP network contains a strong unstable attractor and three weaker stable attractors.

2. The HUAP of the autoassociator contains a weak unstable attractor and two strong stable attractors.

These results are interesting as they can help qualify the learning strategy of the two networks and suggest strategies for speeding up the Back-propagation procedure. In particular, based on our observations, it can be speculated that the MLP network uses a *top-down* or *hypothesis-driven* strategy in which, during the first phase of learning, it tries to group all the inputs (whether instances of the concept or counter-examples) in the same cluster. This can be seen from the fact that for up to Epoch 3000, the HUAP representation of every input point follows the same trajectory, meaning that the network attempts to assign the same output value to all the inputs, whether they belong to the same class or not. Because the destination of these trajectories is an unstable attractor, however, eventually, the trajectories separate and tend towards three different stable attractors. This corresponds to the creation of a second level in the top-down search of the generalization space which allows for the assignment of different output values to the various inputs. Although the autoassociator also starts by searching the space using a top-down strategy (in the first phase of learning which is directed towards a weak attractor), this strategy is quickly abandoned in favor of a *bottom-up* or *data-driven* strategy in which, rather than trying to group all the input data together, the system uses its resources to specify all the input data separately.

This result shows clearly the adaptability of the feedforward neural network which, rather than following rigidly the same search strategy, spontaneously selects the one it assesses to be the most promising given the domain configuration and the network's architecture. Remarkably enough, this phenomenon takes place in the absence of input from the user, and, most generally, without his or her knowledge.

Note that, in addition to their academic interest, our observations have practical relevance since they suggest that a plausible strategy for speeding up Backpropagation learning consists of weakening the unstable attractors present in the hidden unit activation plane. One way of weakening these attractors—at least the strategy used by the autoassociator or suggested by Munro in [5] for the MLP network—is to constrain the learning task by running the Backpropagation procedure on a network containing many output units.[4] The large number of constraints present in the learning task of such networks seems to guarantee that the strength of the unstable attractors is decreased. While apparently useful as far as efficiency is concerned, the effect of such a strategy on accuracy has yet to be documented.

4 Summary

Altogether, this paper has demonstrated that the learning strategies used by the MLP network and the autoassociator are different from one another. The difference in efficiency of the two methods is caused by the fact that the hidden units of the MLP network experience a long latency period during which they tend towards a strong attractor, whereas the autoassociator does not experience such a latency period. We suggest that this difference is caused by the fact that the MLP network and the autoassociator pursue different goals. While MLP attempts to learn the domain in a top-down fashion, the autoassociator uses a bottom-up approach. In the HUAP, this difference translates into the presence of strong unstable attractors in the first case and in their absence in the second. This result constitutes a first step towards the optimization of the time efficiency of a Backpropagation Neural Network, though an important task for the future will be to formalize these results and identify carefully the conditions under which a network will select one or the other strategy.

[4] This strategy is also reflected by the common practice, in the neural network community, of designing binary learning networks of two output units rather than restricting them to a single one (as was done in this study).

References

[1] Stephen J. Hanson and Judy Kegl. Parsnip: A connectionist network that learns natural language grammar from exposure to natural language sentences. In *Proceedings of the Ninth Annual Conference on Cognitive Science*, 1987.

[2] Nathalie Japkowicz. Are we better off without counter-examples? In *Advances in Intelligent Data Analysis (To appear)*, June 1999.

[3] Nathalie Japkowicz. *Concept-Learning in the Absence of Counter-Examples: An Autoassociation-Based Approach to Classification (in progress)*. PhD thesis, Rutgers University, 1999.

[4] Nathalie Japkowicz, Catherine Myers, and Mark Gluck. A novelty detection approach to classification. In *Proceedings of the Fourteenth Joint Conference on Artificial Intelligence*, pages 518–523, 1995.

[5] Paul Munro. Visualizations of 2-d hidden unit space. Technical Report LIS035/IS91003, School of Library and Information Science, University of Pittsburgh, 1991.