# Experiments with the Cascade-Correlation Algorithm

Jihoon Yang
and
Vasant Honavar

Technical Report # 91-16
July 1991; Revised August 1991

**Department of Computer Science**
**226 Atanasoff Hall**
**Iowa State University**
**Ames, IA 50011-1040, U.S.A.**

# Experiments with the Cascade-Correlation Algorithm

Jihoon Yang & Vasant Honavar
Department of Computer Science
Iowa State University
Ames, Iowa 50011. U.S.A.

**ABSTRACT**[1] [2] [3]

This paper describes a series of experiments with the *cascade-correlation algorithm* (CCA) and some of its variants on a number of real-world pattern classification tasks. Some of these experiments investigate the effect of different design parameters on the performance of CCA (in terms of number of training *epochs* and classification accuracy on the test data). Parameter settings that consistently yield good performance on different data sets are identified. The performance of CCA is compared with that of the backpropagation algorithm (BP) and the perceptron algorithm (PA). Preliminary results obtained from some variants of CCA and some directions for future work with CCA-like generative algorithms for neural networks are discussed. [4]

## 1 Introduction

Pattern classification and function approximation are two of the application areas for which several neural network architectures and learning algorithms have been proposed. The perceptron algorithm (PA) [Rosenblatt, 1958] is rather simple and performs quite well in learning to classify patterns that are *linearly separable*. The backpropagation algorithm (BP) [Rumelhart et al., 1986; Werbos, 1974; Parker, 1985] offers a way to learn arbitrarily complex decision boundaries using multi-layer *feed-forward* networks of *sigmoid* nodes. Such networks contain one or more layers of *hidden* nodes between the input and output layers. Despite reports of success on a number of interesting problems, BP can be excruciatingly slow. Several modifications for improving the learning speed have been proposed in the literature (e.g., *Quickprop* (QP) [Fahlman, 1988]). The choice of the number of hidden nodes can have a significant impact on the performance of BP networks. Yet the choice is usually a matter of trial and error.

*Generative* or *Constructive* learning algorithms [Honavar & Uhr, 1988; 1989; 1991; Ash, 1989; Fahlman & Lebiere, 1990] potentially offer means to adaptively determine the necessary network connectivity (thereby avoiding ad-hoc, a-priori, and often inappropriate choices) and the *weights* to solve specific function approximation and pattern classification tasks through supervised learning. However, such algorithms have to incorporate mechanisms for making a variety of decisions (e.g., when to add a new node to the network versus continuing to modify the weights associated with the existing nodes, where to add a node, the connectivity of the

[1]This work was partially supported by the Iowa State University College of Liberal Arts and Sciences
[2]This paper will appear in: *Proccedings of the Fourth UNB Artificial Intelligence Symposium*
[3]It incorporates substantial revisions to an earlier version of TR 91-16
[4]The authors wish to thank Dr. Scott Fahlman for his feedback on an early draft of this paper

added node, etc) [Honavar & Uhr, 1991]. While preliminary results appear to be encouraging [Honavar & Uhr, 1988; Fahlman & Lebiere, 1990], extensive comparative experimental studies with a range of real-world tasks are needed to identify the strengths and weaknesses of, and improved designs for generative algorithms.

This paper presents the results of a number of experiments with the *Cascade-correlation algorithm* (CCA) [Fahlman & Lebiere, 1990]. The performance of CCA and a variant are compared with that of PA and BP on a number of real-world data sets. The performance of CCA is analyzed with respect to a number of design parameters. Some insights concerning parameters that play a critical role in the performance of CCA, some suggestions for improved generative learning algorithms, and some directions for future research are offered.

## 2 Experimental Methodology

### 2.1 Data Sets Used

Four different data sets from the UC-Irvine machine learning data repository were used[5]: chess end games [Shapiro, 1987], audiological disorders [Bareiss, 1988], soybean disease [Reinke, 1984] (The choice of these data sets was influenced by the availabilty of published results of detailed empirical studies with PA and BP [Shavlik et al., 1990; Mooney et al., 1989]), and iris plants database [Fisher, 1936]. The encoding of input and output patterns was exactly identical to that used in their experiments: Each possible value of each input attribute is represented by a different input node; Each output category is represented by a distinct output node; A node activation of 1 denotes the presence of a particular attribute-value (in the case of input nodes) or the category to which the input pattern is assigned (in the case of output nodes). On each run of the algorithm with each of the data sets two thirds of the examples were chosen randomly for training and the rest were used for testing.

The *chess* data set consists of 3196 examples of a "king and rook vs. king and pawn" end game which can be classified into two categories - *win* and *not win*. Each example has 36 attributes (of which one is 3-valued and the rest are 2-valued). Thus, the original data set was encoded with 73 input nodes and 2 output nodes. Because of the large number of examples, 591 examples were selected randomly for each run of the algorithm.

The *audiology* data set consists of 200 examples[6] of cases from the Baylor College of Medicine. 87 nodes were used to represent the input patterns and 24 to represent the output categories.

The *soybean* data set consists of examples of soybean diseases. Several versions of this data set exist. To facilitate direct comparison with the results reported in [Shavlik et al., 1990; Mooney et al., 1989], we used their data set consisting of 594 examples in our experiments[7]. The input patterns were encoded using 208 input nodes and the outputs with 17 output nodes.

The *iris* data set consists of 150 examples of iris plants database[8]. Each example in this

[5]We are grateful to Dr. Pazzani of UC-Irvine for making the data sets available

[6][Shavlik et al., 1990; Mooney et al., 1989] used 226 examples

[7]We are grateful to Dr. Mooney of UT-Austin for making the data available

[8]We are grateful to Dr. Phillips of University of Queensland, Australia for making an on-line version of this data set available

data set is described by four attributes which we encoded using a distributed binary coding using 22 input nodes. The examples belong to one of three species, which were represented by 3 output nodes.

As mentioned earlier, two thirds of data were chosen at random for training and the rest were used for testing on each run. For any given parameter setting, the program was run 10 times (each with a different random initial weight setting). The mean and standard deviation of the observed performance are reported for each such run. The correctness of classification was determined by comparing the highest output with the desired output. Training was terminated if the network classifies more than 99.5% of training data or or the number of epochs exceeded 5000.

## 2.2 Algorithms and Implementation Details

This section briefly summarizes the learning algorithms used in the experiments described in this paper. PA [Rosenblatt, 1958] performs a gradient descent in weight space to minimize the errors in a single layer network (no hidden layers). BP [Rumelhart et al., 1986; Parker, 1985; Werbos, 1974] performs gradient descent in weight space to minimize the errors in a multi-layer network of sigmoid nodes. The details of particular BP and PA implementations are given in [Mooney et al., 1989; Shavlik et al., 1990]. In their study, the learning rate was set to 1.0 for PA; and for BP, the learning rate was set to 0.25 and the momentum was set to 0.9 and 10% of the total number of input and output nodes was used in the hidden layer.

CCA, like several other generative learning algorithms [Honavar & Uhr, 1988; 1989; 1991], performs an error-guided (heuristic) search in the space of network topologies as well as weights. It solves the problem of determination of adequate network topology as follows: It starts with input and output nodes only and trains the connections. If the error is not decreasing fast enough during a sequence of training epochs (as determined by the *patience* parameter, as weights are modified using QP (a faster variant of BP - see below), a pool of candidate nodes is generated. Each node in the pool is connected to all the existing nodes in the network. Each candidate node is trained in parallel again using QP. After a short period of evaluation, the node from the candidate pool whose output exhibits the maximum correlation with the network error is added to the network and the rest of the nodes in the pool are discarded. Once a node is added to the network, its input weights are frozen and only the weights on the output links continue to be altered so as to reduce the error until the error levels off at which point a new pool of candidates is generated. This process is repeated until the error is reduced to the desired level.

In QP [Fahlman, 1988] for each weight, the previous and current error slopes and the weight-change between the points at which these slopes were measured are used to determine a parabola. Then, during weight update, the QP jumps to the minimum point of the parabola. The formula for updating the weights at time $t$ is

$$\Delta w(t) = \frac{S(t)}{S(t-1) - S(t)} \Delta w(t-1)$$

where $S(t)$ and $S(t-1)$ are the current and previous error slopes, respectively. New weights are computed by adding the gradient descent term ($\varepsilon$ times the current slope) to the above formula with two exceptions: if the current slope is non-zero and opposite in sign from the previous slope, ignore the gradient descent term, and if the current slope is in the same sign but in the same or larger size, use some *maximum growth factor* times previous change instead of the above formula.

Fahlman [1988] suggests the following techniques to improve QP's performance.

- *Split Epsilon*: The learning rate for each node is set to be inversely proportional to its fan-in.
- *Sigmoid-prime Function*: To prevent the derivative of the sigmoid function at each node from becoming 0, an offset (0.1) is added to the sigmoid function.
- *Hyperbolic Arctangent Error Function*: To accelerate the learning, the hyperbolic arctangent function which magnifies the large error differences between desired and actual output values is used.

In addition, several parameters can affect the behavior of the algorithm.

- *Learning Rate*: As in PA and BP, this determines the rate of weight change.
- *Maximum Growth Factor*: No weight change is allowed to be greater than maximum growth factor times the previous weight change in magnitude.
- *Weight Range*: This specifies the range of values for randomly generated initial weights.
- *Weight-decay*: To avoid any possible floating-point overflow in changing of weights, this term is added to the slopes computed.
- *Candidate Pool*: Candidates in this pool are trained simultaneously and the best is selected and added to the network.
- *Input/Output Patience*: If the network does not improve during this amount of epochs, stop tuning.

In our experiments with CCA, we used the C code written by Scott Crowder of Carnegie Mellon University (with some necessary modifications) [9].

## 3 Experimental Results

### 3.1 Parametric Study of CCA

This section describes the results of our experiments exploring the space of user-specified parameters of the CCA algorithm. There are essentially two classes of parameters used in the CCA - those that are used to specify the QP weight-modification algorithm, and those that control the generation of new nodes (candidate pool size and patience).

[9]We wish to thank Dr. Fahlman and Mr. Crowder of CMU for making the CCA code available

### 3.1.1 Effect of QP Parameters

Given the large number of parameters used in QP (and heavy computational requirements of each run of the algorithm), an exhaustive study of all possible combinations of parameters was impractical. We therefore performed a sequence of exploratory runs sampling the parameter space and used these initial results to guide the choice of particular parameter ranges for closer examination. A large number of experiments were conducted with parameters that were in the vicinity of those that gave good results during the exploratory runs (iris data set was not used in this phase of the study):

- Chess: Learning rate was varied from 0.2 to 4.0 in steps of 0.2; maximum growth factor from 1.5 to 2.5 in steps of 0.25; weight range from 0.5 to 1.5 in steps of 0.5.
- Audiology and Soybean: Learning rate was varied from 1.4 to 4.0 in steps of 0.2; maximum growth factor from 2.0 to 2.5 in steps of 0.25; weight range from 0.5 to 1.5 in steps of 0.5.

The split epsilon technique (see above) was used for all data sets. Both runs with and without the hyperbolic arctangent error function and the sigmoid-prime offset of 0.1 were conducted for chess and audiology data sets. For soybean data, hyperbolic arctangent error function and sigmoid-prime offset of 0.1 were used in all the runs. When the hyperbolic arctangent error function was used, $\pm 17.0$ was used for $\pm\infty$. Both the size of the candidate pool and the patience were set to 8 for all runs with parameter settings mentioned above. The input weight decay and output weight decay were set to 0.0 and 0.0001, respectively. Finally, symmetric activation function (range -0.5 to +0.5) was employed.

The results of our experiments are summarized below:

- Use of hyperbolic arctangent error function and sigmoid-prime offset generally gave better results.
- Smaller weight ranges generally gave better results.
- Larger maximum growth factor generally gave better results.

The best performance was observed with hyperbolic arctangent error function and sigmoid-prime offset, small weight ranges (0.5), large maximum growth factor (2.5 for chess data and 2.25 for audiology and soybean data), and learning rate between 3.0 and 4.0.

### 3.1.2 Effect of Candidate Pool Size and Patience

The effects of *candidate pool size* and the *patience* parameters were studied by fixing the rest of the parameters to values in the neighborhood of those that yielded the best results as summarized above. Runs were made with three different candidate pool sizes (4, 8, 12) and the patience values (5, 8, 11, 14). Generally, better results were obtained with larger candidate pool sizes. However, the value of patience parameter did not appear to have an observable impact on the results (at least over the range that was examined).

### 3.1.3 Remarks on the Results of Parametric Study

CCA uses QP which incorporates a large number of design parameters. The interaction of various parameters of QP and CCA can be rather complex. Therefore the results presented here should be taken with a grain of salt. Additional studies with a range of increasingly complex tasks are needed to come up with definitive statements on the sensitivity of the performance of CCA with respect to different parameters (and their combinations).

## 3.2 Performance of CCA, PA and BP Compared

This section summarizes the performance of CA on chess, audiology, soybean and iris data sets and compares them with the results for PA and BP. The results for PA and BP with chess, audiology and soybean data sets are from [Shavlik et al., 1990; Mooney et al., 1989]. The results reported for iris data are based on our own experiments (using several different numbers of hidden units in the case of BP). The parameter settings used with these runs correspond to those that gave reasonably good observed performance in the parametric study described above. The results are averaged over 10 runs and the mean and the standard deviation are given in Figure 1.

| *Domain* | *Algorithm* | *Training Epochs* | *Test Correctness (%)* | *# of Hidden Nodes* |
|---|---|---|---|---|
| **Chess** | PA | $3,460.0 \pm 1,910.0$ | $93.9 \pm 2.2$ | 0.0 |
| | BP | $4,120.0 \pm 1,770.0$ | $96.3 \pm 1.0$ | 8.0 |
| | CCA | $164.3 \pm 29.3$ | $95.4 \pm 1.2$ | $1.1 \pm 0.3$ |
| **Audiology** | PA | $9.6 \pm 0.5$ | $73.5 \pm 3.9$ | 0.0 |
| | BP | $2,880.0 \pm 1,980.0$ | $77.7 \pm 3.8$ | 11.0 |
| | CCA | $69.1 \pm 46.5$ | $73.5 \pm 4.3$ | $0.3 \pm 0.5$ |
| **Soybean** | PA | $11.9 \pm 0.7$ | $92.9 \pm 2.1$ | 0.0 |
| | BP | $158.0 \pm 175.0$ | $94.1 \pm 2.5$ | 23.0 |
| | CCA | $17.3 \pm 1.9$ | $97.0 \pm 1.3$ | 0.0 |
| **Iris** | PA | $5000.0 \pm 0.0$ | $31.0 \pm 5.4$ | 0.0 |
| | BP | $4507.8 \pm 1075.3$ | $89.2 \pm 5.9$ | 1.0 |
| | | $4821.8 \pm 563.5$ | $91.2 \pm 2.9$ | 2.0 |
| | | $3987.9 \pm 1733.7$ | $91.0 \pm 3.0$ | 3.0 |
| | | $838.5 \pm 1508.2$ | $93.2 \pm 2.5$ | 4.0 |
| | | $1757.7 \pm 2249.5$ | $92.6 \pm 2.5$ | 5.0 |
| | | $2159.6 \pm 2450.2$ | $92.2 \pm 2.6$ | 6.0 |
| | CCA | $84.9 \pm 58.9$ | $92.6 \pm 1.4$ | $0.5 \pm 0.5$ |

**Figure 1: Performance of CCA, BP, and PA Compared**

Note that soybean and audiology data sets are linearly separable but chess and iris data set are not. The number of training epochs needed by the CCA was on the average, 1-2 orders of magnitude lower than that for BP. Furthermore, CCA generated far fewer hidden nodes than those used in the experiments by [Shavlik et al., 1990; Mooney et al., 1989] as well as in our experiments with iris data. CCA needed somewhat larger number of training epochs than PA on audiology and soybean data but PA took significantly larger number of training epochs than CCA on chess and iris data (presumably because of oscillatory behavior on data that is not linearly separable).

CCA achieved classification accuracy (on test data) that was somewhat worse than that of BP on chess and audiology data but somewhat better than that of BP on soybean data. On iris data, CCA achieved a little worse classification accuracy than BP with 4 hidden nodes but achieved somewhat better classification accuracy with other BP architectures. CCA performed considerably better than PA on chess, soybean and iris data and no worse than PA on audiology data.

## 3.3 Performance of Some Variants of CCA

This section describes the results of our experimets with several variants of CCA. In particular, the results obtained from choosing two different error metrics and node selection strategies are compared; and the effect of allowing the weights on the input links of recruited nodes to change is examined.

### 3.3.1 Choice of Error Metrics and Node Selection Strategies

Two different error metrics were used in measuring the classification accuracy of the algorithm: the first (error metric I) uses the magnitude of error between the desired and actual outputs to determine if an example is correctly classified; the second (error metric II) counts an example as correctly classified if the node with the highest output corresponds to the correct category (In both cases, the weight changes are calculated using the magnitude of the error). With each of the error metrics, two different approaches to choosing a node from the candidate pool (to be added to the network) were tried: the first (like the original CCA) picks the node whose output has the highest correlation with the residual error; the second randomly picks a node from the candidate pool. For each data set, two parameter settings - one with values in the neighborhood of those that gave the best results in the parametric study and the other with values arbitrarily chosen - were used.

The results on chess, audiology and iris data have almost the same characteristics while those on soybean data are different. The number of epochs for training with error metric I was larger than that with error metric II on all data sets with the exception of soybean data.

The number of training epochs needed was substantially smaller when the degree of correlation with the residual error was used as the basis for the selection of a node from the candidate pool than when a candidate node was chosen at random. The difference between the two variants of CCA was even more conspicuous in runs that generated a large number of hidden units. The difference in classification accuracy between the two variants of CCA was negligible on all the data sets. It must be noted however, that this result might be an artefact resulting from the relatively simple decision surfaces needed to classify the data sets used (as indicated by the relatively small number of hidden units generated when "good" parameters were used in the runs). Additional studies with data sets requiring sufficiently complex decision surfaces are needed to establish the validity of this observation.

Figure 2 summarizes these results (the entries indicate the values averaged Over 10 runs for a given parameter setting for each error metric; The first row of entries for each data set corresponds to the results obtained with an *arbitrary* setting of QP parameters whereas the second row corresponds to the results obtained with *good* QP parameter settings as determined from the parametric study of section 4.1; #H denotes the number of hidden units).

| *Error Metric* | *Domain* | *With Correlation* | | | *Without Correlation* | | |
|---|---|---|---|---|---|---|---|
| | | *Epochs* | *% Correct* | *# H.* | *Epochs* | *% Correct* | *# H.* |
| **I** | **Chess** | 335.3 | 95.7 | 2.2 | 431.2 | 95.1 | 2.9 |
| | | 309.8 | 95.8 | 2.1 | 354.1 | 95.6 | 2.4 |
| | **Audiology** | 519.0 | 72.1 | 4.4 | 724.9 | 76.3 | 6.0 |
| | | 170.5 | 78.5 | 1.0 | 230.3 | 75.0 | 1.4 |
| | **Soybean** | 17.9 | 47.3 | 0.0 | 17.9 | 47.3 | 0.0 |
| | | 17.6 | 61.9 | 0.0 | 17.6 | 61.9 | 0.0 |
| | **Iris** | 120.7 | 91.6 | 0.8 | 186.9 | 90.8 | 1.2 |
| | | 131.9 | 90.8 | 0.8 | 166.8 | 90.6 | 1.1 |
| **II** | **Chess** | 214.0 | 96.0 | 1.4 | 213.5 | 96.7 | 1.4 |
| | | 175.5 | 95.4 | 1.1 | 233.4 | 96.8 | 1.5 |
| | **Audiology** | 366.1 | 74.0 | 2.8 | 407.2 | 74.1 | 3.2 |
| | | 43.7 | 73.5 | 0.0 | 43.7 | 73.5 | 0.0 |
| | **Soybean** | 37.1 | 98.5 | 0.1 | 40.4 | 98.7 | 0.1 |
| | | 22.5 | 97.5 | 0.0 | 22.5 | 97.5 | 0.0 |
| | **Iris** | 125.2 | 92.6 | 0.8 | 127.0 | 92.6 | 0.8 |
| | | 84.9 | 92.6 | 0.5 | 89.0 | 92.8 | 0.5 |

**Figure 2: Using Different Error Metrics With and Without Correlation**

### 3.3.2 Unfreezing the Input Link Weights on the Hidden Nodes

CCA algorithm as described in [Fahlman & Lebiere, 1990] modifies only the weights on the output links on the hidden nodes; The weights on the input links to each hidden node are frozen once the node is added to the network. Two variants of CCA were tried to examine the effect of not freezing the weights on the input links to the hidden nodes in the network. The first variant (Variant I) always trains incoming weights as well as outgoing weights. The second (Variant II) freezes the weights on the input links (as does the original CCA) until the network learns to classify most (in our experiments, 95%) of the training data correctly; then it unfreezes the previously frozen weights and allows them to change according to the BP algorithm.

Again, two different parameter settings (one in the vicinity of that which gave the best results in the parametric study (case I) and the other arbitrarily chosen (case II)) were tried in the runs with chess, audiology, and iris data sets (soybean data set was not used in these experiments because it is linearly separable and requires no hidden units as verified from the results reported above). As we can see in Figure 3, the number of training epochs needed decrease (but an epoch now includes training weights on both the input as well as the output links on the hidden units which is computationally more expensive than training only the weights on the output links of the hidden units) when the weights on the input links of the hidden units were allowed to change after their addition to the network. However, the accuracy of classification on the test data was found to be generally worse than the original CCA on chess and audiology data for both the variants. However the results on the iris data set were better than the original CCA in the case of variant II - at least when good parameter settings were used for QP.

It is difficult to say whether or not unfreezing the weights on the input links offers any advantages over the original CCA algorithm without carrying out additional experiments. It appears reasonable to expect variant II to yield somewhat better performance than the original CCA on account of allowing the weights on the input links to change after the network had arrived at weight values that are close to the target values (as indicated by the classification accuracy of 95% on the training data).

Figure 3 summarizes these results (the entries indicate the values averaged over 10 runs for a given parameter setting; #H indicates the number of hidden ndoes; case I and case II denote runs with good QP parameter settings and runs with arbitrarily chosen QP parameter settings).

| *Domain* | *Method* | *Case I* | | | *Case II* | | |
|---|---|---|---|---|---|---|---|
| | | *Epochs* | *% Correct* | *# H.* | *Epochs* | *% Correct* | *# H.* |
| **Chess** | Original | 235.5 | 95.5 | 1.7 | 195.7 | 95.9 | 1.2 |
| | Variant I | 235.5 | 92.3 | 1.7 | 195.7 | 90.2 | 1.2 |
| | Variant II | 235.5 | 89.3 | 1.7 | 195.7 | 84.3 | 1.2 |
| **Audiology** | Original | 366.1 | 74.0 | 2.8 | 301.8 | 74.1 | 2.5 |
| | Variant I | 212.2 | 54.9 | 1.5 | 262.4 | 69.6 | 2.2 |
| | Variant II | 382.5 | 61.6 | 2.8 | 262.4 | 71.6 | 2.2 |
| **Iris** | Original | 252.0 | 95.4 | 3.1 | 84.9 | 92.6 | 0.5 |
| | Variant I | 236.5 | 97.0 | 3.1 | 99.4 | 92.2 | 0.6 |
| | Variant II | 236.5 | 97.0 | 3.1 | 99.4 | 92.2 | 0.6 |

**Figure 3: CCA and Variants that Modify Input Weights on Hidden Units Compared**

## 4 Summary and Discussion

This section summarizes the results of our experiments with CCA and some of its variants and identifies some directions for further research.

### 4.1 Comparison of CCA with BP and PA

The results of our experiments show that CCA learns on the average, 1-2 orders of magnitude faster than BP (in terms of the number of training epochs needed) with minimal degradation in the accuracy of classification attained on the test data. Also, the resulting networks were substantially smaller than in the case of BP. However, CCA incorporates a large number of design parameters that have to be set by the user. Our best results were obtained using parameter settings that were found to give good results after a large number of exploratory runs. This might be worth the effort involved if the goal is to find relatively small networks that perform the task well (for say, later implementation in hardware).

## 4.2 Parameter Settings

There are essentially two classes of user-specifiable parameters in CCA: those that are used to specify the QP weight-modification algorithm, and those that control the generation of new nodes (e.g., candidate pool size and patience).

Given the large number of parameters used by QP, and the heavy compuatational requirements of each run, an exhaustive exploration of the parameter space was impractical. We had to resort to a sampling of the parameter space followed by a detailed study of parameter settings in the vicinity of those that gave the best results during the initial exploration. This can be impractical in situations in which fast learning is the primary goal: the time taken up by the exploratory runs needed to identify good parameter settings might more than offset the reduction in learning time obtained with such parameter settings - unless relatively domain-independent parameter settings can be found. On all the data sets used in our study, we were able to obtain relatively good performance over a certain narrow range of QP parameters. However, the question as to whether relatively domain-independent QP parameter settings exist can be settled only through extensive empirical studies on a wide range of complex real-world data sets.

It is not at all clear to what extent the performance gains exhibited by CCA over BP are to be attributed to the improved weight-modification algorithm (QP) used by CCA as opposed to the generative aspect of CCA (i.e., its ability to add potentially useful nodes to the network as needed). It is probably worth investigating CCA-like generative learning algorithms that use a small set of parameters. Indeed, variants of CCA that replace QP with a simpler weight modification algorithm (using fewer user-specifiable parameters than QP) is probably worth examining.

Our preliminary results indicate that the generative component of CCA performs better as the size of the candidate pool is increased. This is to be expected because the larger the size of the candidate pool, the greater is the likelihood of finding a node that is well-tuned to reduce the residual error. The effect of patience parameter was not as clear. Over the range that we examined, the exact setting of this parameter did not appear to have a significant impact on the performance on any of the data sets. Whether this result would be valid on more complex data sets remains to be seen.

## 4.3 Some Variants of CCA

CCA is an instance of *generative learning* algorithms which dynamically alter the network topology as well as the weights on links so as to minimize some error metric. The space of possible designs for such algorithms is quite large [Honavar & Uhr, 1991] but only a few designs have been investigated empirically to date. The preliminary results of experiments with some simple variants of CCA that are presented in this paper suggest the need for a systematic study of the consequences of different design decisions (e.g., how a node is chosen for addition to the network; how the weights are modified, regulatory mechanisms that guide the search for appropriate network topologies, etc.) on the performance of generative learning algorithms.

## 4.4 Some Directions for Further Research

This study was motivated by the need for a thorough empirical analysis of generative learning algorithms such as CCA to determine their effectiveness on a range of real-world problems. The results of our experiments encourage further investigation of this class of algorithms. Extensive empirical studies are needed to guide the development of better learning algorithms. This task is made difficult by the non-availability of non-proprietary data sets from real-world domains that are complex enough to adequately challenge generative learning algorithms (note that all the data sets used in this study required the generation of relatively small numbers of hidden units). To (at least partially) circumvent this problem, we are developing artificial data sets with known properties.

Generative learning can be viewed as a heuristic search through the space of network topologies (and weights). Heuristics that can be used to constrain this search to useful regions of the space to ensure efficient learning in particular applications (e.g., 2-and-3-dimensional object recognition) are being investigated [Honavar & Uhr, 1988; 1989; 1991]. Comparison of the performance of generative learning algorithms with well-known theoretical bounds (e.g., those based on the VC dimension of the hypothesis space) is also of considerable theoretical as well as practical interest. Experiments are currently underway to address these and related questions.

## References

[1] Ash, T. (1989) Dynamic Node Creation in Backpropagation Networks. *ICS Report #8901, Institute for Cognitive Science, University of California at San Diego*, La Jolla, CA.

[2] Bareiss, E. R. (1988) PROTOS: A Unified Approach to Concept Representation, Classification and Learning. *Ph.D. Thesis, Department of Computer Science, University of Texas*, Austin, TX.

[3] Fahlman, S. E. (1988) Faster-Learning Variations on Back-Propagation: An Empirical Study. *Proceedings of 1988 Connectionist Models Summer School*, Touretzky, D., Hinton, G and Sejnowski T. (Eds.), San Mateo, CA, Morgan Kaufmann, pp. 38-51.

[4] Fahlman, S. E. and Lebiere, C. (1990) The Cascade-Correlation Learning Architecture. *Technical Report #CMU-CS-90-100, School of Computer Science, Carnegie Mellon University*, Pittsburgh, PA.

[5] Fisher, R. A. (1936) The Use of Multiple Measurements in Taxonomic Problems. *Annual Eugenics, 7, Part II*, pp. 179-188.

[6] Honavar, V. and Uhr, L. (1988) A Network of Neuron-Like Units that Learns to Perceive by Generation as well as Reweighting of its Links. *Proceedings of 1988 Connectionist Models Summer School*, Touretzky, D., Hinton, G and Sejnowski T. (Eds.), San Mateo, CA, Morgan Kaufmann.

[7] Honavar, V. and Uhr, L. (1989) Brain-Structured Connectionist Networks that Perceive and Learn, *Connection Science 1*, pp. 139-159.

[8] Honavar, V. and Uhr, L. (1991) Generative Learning Structures and Processes for Generalized Connectionist Networks. *Technical Report #91-02, Department of Computer Science, Iowa State University*, Ames, IA.

[9] Mooney, R., Shavlik, J., Towell, G. and Gove, A. (1989) An Experimental Comparison of Symbolic and Connectionist Learning Algorithms. *Proceedings of Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI.

[10] Parker, D. B. (1985) Learning-Logic. *Technical Report TR-47, MIT, Center for Computational Research in Economics and Management Science*, Cambridge, MA.

[11] Reinke, R. (1984) Knowledge Acquisition and Refinement Tools for the ADVISE Meta-Expert System. *M.S. Thesis, University of Illinois at Urbana-Champaign*, IL.

[12] Rosenblatt, F. (1958) The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review 65*, pp. 386-408.

[13] Rumelhart, D. E., Hinton, G. E. and Williams, J. R. (1986) Learning Internal Representations by Error Propagation. *Parallel Distributed Processing, Vol. I*, Rumelhart, D. E. and McClelland, J. L. (Eds.), MIT Press, Cambridge, MA.

[14] Shapiro, A. (1987) *Structured Induction in Expert Systems*, Addison Wesley, Reading, MA.

[15] Shavlik, J. W., Mooney, R. J. and Towell, G. G. (1990) Symbolic and Neural Learning Algorithms: An Experimental Comparison. *Technical Report #955, Computer Sciences Department, University of Wisconsin-Madison*, Madison, WI.

[16] Werbos, P. (1974) Beyond regression: New Tools for Prediction and Analysis in Behavioral Sciences. *Ph.D. Dissertation, Harvard University*, Cambridge, MA.