An Experimental Comparison of Symbolic and Connectionist Learning Algorithms

Raymond Mooney
Computer Sciences
University of Texas
Austin, TX 78712

Jude Shavlik
Computer Sciences
University of Wisconsin
Madison, WI 53706

Geoffrey Towell
Computer Sciences
University of Wisconsin
Madison, WI 53706

Alan Gove
Computer Sciences
University of Texas
Austin, TX 78712

Abstract

Despite the fact that many symbolic and connectionist (neural net) learning algorithms are addressing the same problem of learning from classified examples, very little is known regarding their comparative strengths and weaknesses. This paper presents the results of experiments comparing the ID3 symbolic learning algorithm with the perceptron and back-propagation connectionist learning algorithms on several large real-world data sets. The results show that ID3 and perceptron run significantly faster than does back-propagation, both during learning and during classification of novel examples. However, the probability of correctly classifying new examples is about the same for the three systems. On noisy data sets there is some indication that back-propagation classifies more accurately.

1. Introduction

Both symbolic and connectionist (or neural network) learning algorithms have been developed; however, there has been little direct comparison of these two basic approaches to knowledge acquisition. Consequently, despite the fact that symbolic and connectionist learning systems frequently address the same general problem, very little is known regarding their comparative strengths and weaknesses.

The problem most often addressed by both connectionist and symbolic learning systems is the inductive acquisition of concepts from examples. This problem can be briefly defined as follows: given descriptions of a set of examples each labelled as belonging to a particular class, determine a procedure for correctly assigning examples to these classes.[1] Within symbolic machine learning, numerous algorithms have been developed for learning decision trees [Quinlan86] or logical definitions [Michalski80] from examples, both of which can be used to classify subsequent examples. These algorithms have been tested on problems ranging from soybean disease diagnosis [Michalski80] to classifying chess end games [Quinlan83]. Within connectionism, several algorithms have been developed for training a network to respond correctly to a set of examples by appropriately modifying its connection weights [Rosenblatt62, Rumelhart86]. After training, a network can be used to classify novel examples. Connectionist learning algorithms have been tested on problems ranging from converting text to speech [Sejnowski87] to evaluating moves in backgammon [Tesauro88].

Connectionist and symbolic systems for learning from examples generally require the same input, namely, classified examples described as feature vectors. In addition, their performance is generally evaluated in the same manner, namely, by testing their ability to correctly classify novel examples. Despite this situation, until this conference there were no published experiments that directly compared the two approaches on "real world" data. There have been several experimental comparisons of different symbolic learning systems (e.g., [Rendell89]); however, none of these experiments included connectionist algorithms.

This paper presents the results of experiments comparing the performance of the ID3 symbolic learning algorithm [Quinlan86] with both the perceptron [Rosenblatt62] and back-propagation [Rumelhart86] connectionist algorithms. All three systems are tested on several large data sets from previous symbolic and connectionist experiments, and their accuracy on novel examples and run-time performance are measured. One surprising result is that the perceptron learning algorithm, which has well-known limitations, performs quite well. In general, its accuracy and run-time are comparable to ID3's. Although the back-propagation algorithm generally takes one to two orders of magnitude more time to train than the other two algorithms, its accuracy on novel examples is always comparable or superior to that of the other algorithms.

2. Experimental Methodology

This section motivates the choice of the particular algorithms used in the study and describes the data sets used to test them. Next, the versions of the systems used are discussed and the method for choosing appropriate input encodings and parameter settings is described. Finally, the

* This research was partially supported by the University of Texas at Austin's Department of Computer Sciences, by a grant from the University of Wisconsin Graduate School, and by a graduate fellowship to one of the authors (A.G.) supported by the Army Research Office under grant ARO-DAAG29-84-K-0060.

[1] In the connectionist literature, this problem is frequently referred to as *supervised* or *associative* learning.

exact experiments conducted and data collected are reported.

2.1. Choice of Algorithms

In order to experimentally compare connectionist and symbolic algorithms for learning from examples, the ID3, perceptron, and back-propagation algorithms are chosen as representative algorithms. The choice of these particular algorithms is based on several factors.

ID3 is chosen because it is a simple and popular symbolic algorithm for learning from examples. It has been extensively tested on a number of large data sets and is the basis of several commercial rule induction systems. In addition, ID3 has been augmented with techniques for handling noisy data and missing information [Quinlan86]. Finally, in experimental comparisons with other symbolic learning algorithms, ID3 generally performs about as well or better than the competition [Rendell89].

As is well known, the perceptron learning procedure (one of the first "connectionist" learning algorithms) is incapable of learning concepts that are not linearly separable [Minsky67]. Despite this fact, early results indicated that it performed quite well on data sets used to test other learning systems. For example, it performed well on a popular data set for diagnosing soybean diseases [Reinke84], which was found to be linearly separable. Consequently, perceptron is included to determine to what extent this simple algorithm can handle real-world data.

Over the past several years, a few connectionist learning algorithms have been developed that overcome the restrictions of the perceptron. Back-propagation (also called the generalized delta rule) [Rumelhart86] is currently the most popular of these procedures. Its ability to train "hidden units" allows it to learn concepts that are not linearly separable. Back-propagation has proven more efficient than learning procedures for Boltzmann machines and has been tested on several large-scale problems [Sejnowski87, Tesauro88]. Consequently, it is chosen to represent the new connectionist learning algorithms.

2.2. Data Sets Used

Four different data sets are used to test the three algorithms. Three have been previously used to test different symbolic learning systems and one has been used to test back-propagation.

The three data sets previously used with symbolic learning systems describe soybean diseases [Reinke84], chess end games [Shapiro87], and audiological disorders [Bareiss88]. These data sets involve, respectively: 50, 36, and 58 features; 289, 591, and 226 examples; 17, 2, and 24 categories.

The NETtalk data set [Sejnowski87] is a training set for converting text to speech. It consists of a 20,012 word dictionary with corresponding phonetic pronunciation. Each letter of each word and the surrounding three letters on each side form a training example and each phoneme/stress pair constitutes a category. A seven letter window is insufficient to uniquely identify the sound/stress pair attributable to the central letter. As a result, the data can be considered to be noisy. While this is not a problem for back-propagation, it is a problem for the standard ID3 algorithm. To handle noise in ID3, the *chi-square* technique suggested by Quinlan [Quinlan86] is used with this data set.

Unfortunately, the full dictionary is too extensive to tractably analyze in a manner equivalent to the other domains. Instead, a smaller data set is extracted by looking at the 1000 most common English words (as reported in [Kuchera67]) and keeping those that appear in the NETtalk dictionary. This intermediate data set is further pruned by only keeping the examples involving "A" sounds. This produces 444 examples that fall into 18 sound/stress categories. This data set is called *NETtalk-A*.

A single test is also done for each algorithm using the full dictionary as a test set. The subset of the 1000 most common English words that appear in the dictionary is the training set. This set - called *NETtalk-full* - contains 808 words, which produce 4,259 examples classified into 114 categories. The test set involves roughly 143,000 examples.

2.3. Implementation Details

Both ID3 and perceptron are implemented in Common Lisp and although some attention was paid to efficiency, the code was not carefully optimized. All experiments are run on Sun 4/110's with 8 Mbytes of memory.

Two versions of ID3 that learn to discriminate multiple categories are used. For the noisy NETtalk data, a chi-square version is used with a confidence level of 99% [Quinlan86]. In the other domains, a no-noise version of ID3 is used. Experiments with the noise-handling version of ID3 on the noise-free domains indicated that the no-noise version consistently performed better.

The version of perceptron used in the experiments includes cycle-detection. Once a set of weights repeats, the algorithm stops, indicating that the data is not linearly separable. The *perceptron cycling theorem* [Minsky67] guarantees this will eventually happen if the data is not linearly separable. (However, due to simulation time restrictions, the perceptron is also stopped if it cycles through the training data 5000 times.) One perceptron is trained for each possible category to distinguish members of that category from all other categories. A test example is classified by passing it through all the perceptrons and assigning it to the category whose perceptron's output exceeds its threshold by the largest amount.

The version of back-propagation used in the experiments is the C code supplied with the third volume of the PDP series [McClelland87]. The learning rate is set to 0.25 in an attempt to avoid local minima. The momentum term is set at 0.9. Networks contain one output bit for each category and the number of hidden units is 10% of the total number of input and output units, a number which was empirically found to work well. When testing, an example is assigned to the category whose output unit has the highest value. Training terminates when the network correctly classifies at least 99.5% of the training data or when the number of passes through the data (i.e., the number of

epochs) reaches 5000.

Examples are represented as bit vectors rather than standard feature vectors. For each possible value of each feature there is a a bit which is on or off depending on whether the example has that value for that feature. Normally, exactly one bit in the set of bits comprising a single feature will be on. However, if the feature is missing, then all the bits in the set are off, and no special processing for missing features need be done.

This binary representation was initially used only on the connectionist algorithms which require binary inputs. However, the binary encoding was found to consistently improve the classification performance of ID3 as well, and therefore is used for all of the learning algorithms. ID3's improved performance with the binary encoding may be due to several factors. First, since every feature has only two values, the binary encoding eliminates the gain criterion's undesirable preference for many-valued features [Quinlan86]. Second, it allows for more general branching decisions such as *red* vs. *non-red* instead of requiring nodes to branch on all values of a feature. This may help overcome the *irrelevant values problem* [Cheng88] and result in more general and therefore better decision trees. However, since the binary encoded examples have more features, ID3's run time is increased somewhat under this approach.

2.4. Description of Experiments

Each data set (except for NETtalk-full) is separated into a collection of training and testing sets. After each system processes a training set, its performance is measured on the corresponding test set. To reduce statistical fluctuations, results are averaged over ten different training and testing sets produces by randomly placing two-thirds of the examples for each category in the training set and the others in a corresponding test set. Again to reduce statistical fluctuations, each run of back-propagation uses a different seed random number which determines the initial network weights.

During training, CPU time, the number of cycles through the training data, and the final classification correctness for the training data are measured. ID3 only processes the data once, while back-propagation and perceptron repeatedly process the data until one of the stopping criteria described above is met. During testing, CPU time and correctness on the testing data are measured.

Due to computational resource limitations, NETtalk-full is not converted into ten randomized data sets. Rather, the 808 most common words are processed by each system and the results tested on the full dictionary (minus the 808 training words). Because of the size of the training set, training restrictions are required. Back-propagation is limited to 100 epochs and perceptron is only allowed to cycle a maximum of 100 times on each category before terminating.

3. Experimental Results

Figures 1 and 2 contain the experimental results. The first figure reports the training times of the three systems (normalized to the time taken by ID3). Correctness on the test data is reported in figure 2. The actual numbers, their standard deviations, and several other statistics are given in [Shavlik89]. The means of all 40 training times (everything but NETtalk-full) are 144 seconds (ID3), 373 seconds (perceptron), and 21,000 seconds (back-propagation). For correctness, the means are 81.0% (ID3), 79.4% (perceptron), and 83.6% (back-propagation). For the single run with NETtalk-full, the correctness figures are 53.5% (ID3), 51.5% (perceptron), 60.7% (back-propagation) and the training times are 6,350 seconds (ID3), 10,300 seconds (perceptron) and 168,000 seconds (back-propagation).

A relevant statistical test for these problem results is a two-way analysis of variance. This test is designed to determine the source of variation in the observed correctness over each of the four domains. The two way analysis of variance returns two values: the likelihood that the variation can be explained by the different training sets, and the likelihood that the variation can be explained by differences between training methods.

For NETtalk-A and soybeans (where back-propagation performs best) and chess (where ID3 performs best), it is possible to conclude at the 0.5% level (i.e., with 99.5% confidence) that the variation in observed correctness results from the difference in training methods rather than random error. Results are less conclusive for audiology (where back-propagation out-performs the others). There it is only possible to conclude with 95% confidence that the training method is the source of the variation.

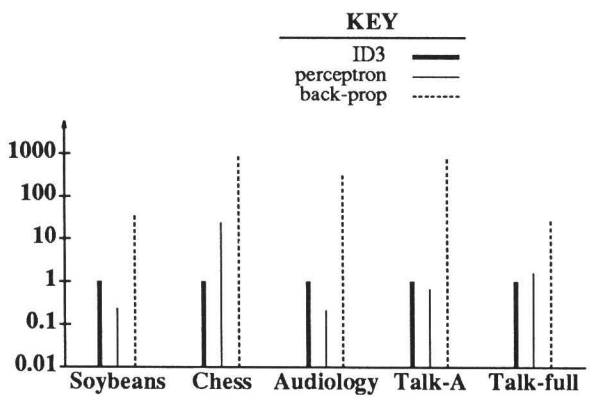

Figure 1. Relative Training Times of the Three Algorithms (scaled to ID3)

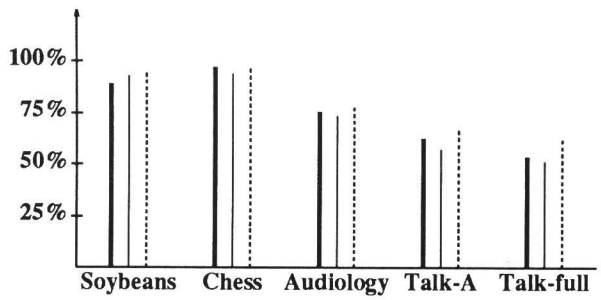

Figure 2. Correctnesses of the Three Algorithms on Test Data

4. Discussion of Results

The results indicate that the three systems performed remarkably similarly with respect to classification of novel examples. However, ID3 and perceptron train much faster than back-propagation. There is some indication that back-propagation works better on the noisy NETtalk data.

4.1. Perceptron Performance

One surprising result of these experiments is how well the simple perceptron algorithm performs. The perceptron was largely abandoned as a general learning mechanism about twenty years ago because of its inherent limitations, such as its inability to learn non-linearly-separable concepts [Minsky67]. Nevertheless, it performed quite well in these experiments. Except on the NETtalk data, the accuracy of the perceptron is hardly distinguishable from the more complicated learning algorithms; and even on the NETtalk data, it performs about as well as ID3. In addition, it is very efficient. On all but the chess data and NETtalk-full, perceptron's training time is less than ID3's.

These results indicate the presence of a large amount of regularity in the training sets chosen as representative of data previously used to test symbolic learning systems. The categories present in both the soybean and audiology data are linearly separable for all ten randomly chosen training sets. The two categories in the chess data are linearly separable for four of the ten training sets and almost linearly separable on the rest (average correctness on the training set is 97.5%). Despite the fact that the data sets are large and represent real categories, their regularity makes them relatively "easy" for even simple learning algorithms like the perceptron.

One possible explanation for the regularity in the data is that it is reflecting regularity in the real-world categories. In other words, members of a real category naturally have a great deal in common and are relatively easy to distinguish from examples of other categories. Another possible explanation is that the features present in the data have been very carefully engineered to reflect important differences in the categories. For example, formulating features for chess end games which were appropriate for learning required considerable human effort [Quinlan83]. The actual explanation is probably a combination of these two important factors.

Regardless of the reason, data for many "real" problems seems to consist of linearly separable categories. Since the perceptron is a simple and efficient learning algorithm in this case, using a perceptron as an initial test system is probably a good idea. If a set of categories are not linearly separable, the *perceptron cycling theorem* [Minsky67] guarantees that the algorithm will eventually repeat the same set of weights and can be terminated. In this case, a more complicated algorithm such as ID3 or back-propagation can be tried. The *perceptron tree error correction procedure* [Utgoff88a] is an example of such a hybrid approach. This algorithm first tries the perceptron learning procedure and if it fails splits the data into subsets using ID3's information-theoretic measure and applies itself recursively to each subset.

4.2. Equivalence of Classification Accuracy

In general, all three learning systems are remarkably similar with respect to classifying novel examples. Despite the obvious differences between decision trees and connectionist networks, their ability to accurately represent concepts and correctly classify novel instances appears to be quite comparable. Data from other recent experiments support this general conclusion [Fisher89, Weiss89]. Learning curves presented in [Fisher89] suggest that that ID3 performs better on relatively small amounts of data; however, this is an artifact of how incremental training was performed in back-propagation. If back-propagation is always run to "convergence" this difference disappears [Shavlik89]. Fisher and McKusick also found that once back-propagation converges, it consistently attains a correctness a few percentage points above ID3. However, they did not use binary input encodings for ID3, consequently ID3 and back-propagation were given different encodings of the inputs. We found that using the binary encoding for ID3 can improve correctness by a few percentage points and may account for some of the difference observed by Fisher and McKusick.

One possible explanation for the similarity in performance is that most reasonable procedures which correctly classify a particular set of training instances (possibly allowing for some noise) are about equally likely to classify a novel instance correctly. Another possible explanation is that the inductive biases inherent in connectionist and symbolic representations and algorithms reflect implicit biases in real categories equally well. For example, all three systems share some form of an "Occam's Razor" bias and, at least to some degree, prefer simpler hypotheses. However, for perceptron and back-propagation, the complexity of the hypothesis is constrained by the user who must initially select an appropriate network for the problem. Unlike most symbolic learning systems which explicitly search for a simple hypothesis, connectionist systems simply search for a correct hypothesis which fits into a user-specified network. Although both generally use hill-climbing search to guide learning, connectionist systems hill-climb in "correctness space" while symbolic systems hill-climb in "simplicity space." An interesting development would be a connectionist learning algorithm which explicitly tries to learn simple networks, for example by eliminating unnecessary hidden units and/or slowly adding hidden units as they are needed [Honavar88]. Such a system would help ease the burden of having to initially specify an appropriate network for a problem.

4.3. Performance on the NETtalk Data

In addition to being by far the largest data set, the NETtalk data set is the only one that involves noise. On NETtalk-A and especially NETtalk-full, back-propagation performs better than perceptron and the noise-handling version of ID3. Although it is not clear actually which properties of the NETtalk data lead to back-propagation's superior performance, one hypothesis is that back-propagation's ability to perform in the presence of noisy training data is better than ID3's. Experiments on the effect of noise on

both ID3 and back-propagation tend to support this hypothesis [Fisher89, Shavlik89].

In [Sejnowski87], the sound/stress outputs in NETtalk-full are encoded for back-propagation as a sequence of 26 bits (as opposed to one bit for each of the 114 categories). The first 21 bits are a distributed encoding of the 51 phonemes contained in the dictionary. Each bit is a unary feature describing one of: position of the tongue in the mouth, phoneme type, vowel height, or punctuation. The remaining five bits form a local encoding of the five types of stresses used in the dictionary. A final category is obtained by choosing the one that makes the smallest angle in feature space with the 26 output bits. (This is the "best guess" metric used in [Sejnowski87].)

Sejnowski's setup was repeated with the NETtalk-full data, using 120 hidden units. Running this for 30 epochs (as in [Sejnowski87]) takes 180,000 seconds. Correctness is 95% on the training set and 71% on the testing set. (These numbers are the best of five runs. Due to the random initial weights, twice after 30 training epochs back-propagation's correctness on the both the training and test sets were less than 10%.) This is significantly better than using the original output encoding of one bit per category, which results, after 100 epochs and 168,000 seconds, in correctnesses of 84% percent on the training set and 61% on the testing set. Further investigation has shown that the distributed output encoding can also be used to improve the performance of ID3; however, not to same the extent that it improves back-propagation's performance [Shavlik89].

4.4. Slowness of Back-Propagation

Although back-propagation performs about as well or better than the other systems at classifying novel examples, it consistently takes a lot more time to train. Averaged across all four data sets, back-propagation takes about 500 times as long to train as ID3. (Testing in back-propagation takes about 10 times longer.) These factors would probably increase if one optimized the ID3 code, which is not coded efficiently compared to the C version of back-propagation.

One obvious way back-propagation can be made faster is to exploit its intrinsic parallelism. The networks used in these experiments contained an average of about 175 units. Consequently, assuming one processor per unit and perfect speedup, the training time for back-propagation could possibly be made competitive with ID3's. However, ID3 is a recursive divide-and-conquer algorithm and therefore also has a great deal of intrinsic parallelism. In addition, in perceptron each output bit is learned independently, one simple source of parallelism for this method. Comparing the training time for parallel implementations of all three algorithms would be the only fair way to address this issue.

4.5. Additional Issues

There are several other differences between the three systems and between symbolic and connectionist approaches in general. For one, given a collection of input/output training pairs, ID3 and perceptron can be directly run. On the other hand, in order to run back-propagation, a network architecture must be chosen, currently much more of an art than a science. Not only must the number of hidden units be chosen, but the number of hidden layers must also be specified. In addition, the learning rate and the momentum term must be specified. Performance may depend greatly on the initial randomly-selected weights and several runs with different initial weights may be necessary to get a good final result. Finally, a criterion for stopping training must be chosen. Our experience has shown that if parameters are inappropriately set or if initial weights are unfavorable, back-propagation may fail to converge efficiently. However, it should also be mentioned that many symbolic learning systems have parameters which must be appropriately set to insure good performance [Rendell89].

Another issue is the human interpretability of the acquired rules. Symbolic learning can produce interpretable rules while networks of weights are harder to interpret. However, large decision trees can also be very difficult to interpret [Shapiro87]. Finally, connectionist models are neurally-inspired, while symbolic models are not. Hence, connectionist models may shed more light on human neurophysiology.

5. Conclusions

A current controversy is the relative merits of symbolic and connectionist approaches to artificial intelligence. Although, symbolic and connectionist learning systems often address the same task of inductively acquiring concepts from classified examples, their comparative performance has not been adequately investigated. In this paper, the performance of a symbolic learning system (ID3) and two connectionist learning systems (perceptron and back-propagation) are compared on four real-world data sets. These data sets have been used in previous experiments. Three in symbolic learning research (soybeans, chess, and audiology) and one in connectionist research (NETtalk). Experimental results indicate that ID3 and perceptron run significantly faster than does back-propagation, both during learning and during classification of novel examples. The probability of correctly classifying new examples is about the same for the three systems, although there is indication that back-propagation classifies more accurately on noisy complex data sets.

Follow-up experiments reported in [Shavlik89] have confirmed that back-propagation handles both noise and missing feature values more reliably than either ID3 or perceptron. Although on some data sets all three systems degrade equally as noise or missing data is introduced, back-propagation shows significantly less degradation on particular data sets. Another issue is how well the algorithms learn as a function of the amount of training. Perhaps some methods perform better with relatively little training data while others perform better on large training sets. In [Shavlik89] "learning curves" are presented which indicate that relative amount of training is not a distinguishing factor. A third issue is incremental learning without requiring the complete storage of all previous examples.

Incremental versions of ID3 have been proposed [Schlimmer86, Utgoff88b] and back-propagation can be performed incrementally by processing each new example some number of times and then discarding it. Comparison of various incremental approaches is an area for future research, as is investigation of parallel approaches. Further investigation will hopefully lead to a better understanding of the relative strengths and weaknesses of the symbolic and connectionist approaches to machine learning.

Acknowledgements

The authors would like to thank the following people for supplying data sets: Bob Stepp and Bob Reinke for the soybean data; Ray Bareiss, Bruce Porter, and Craig Wier for the audiology data which was collected with the help of Professor James Jerger of the Baylor College of Medicine; Rob Holte and Peter Clark for Alen Shapiro's chess data; and Terry Sejnowski for the NETtalk data. Elizabeth Towell assisted with the analysis of variance. Rita Duran, Wan Yik Lee, and Richard Maclin contributed to the implementations.

References

[Bareiss88] E. R. Bareiss, "PROTOS: A Unified Approach to Concept Representation, Classification and Learning," Ph.D. Thesis, Department of Computer Science, University of Texas, Austin, TX, 1988. (Available as Technical Report AI88-83.)

[Cheng88] J. Cheng, U. M. Fayyad, K. B. Irani and Z. Qian, "Improved Decision Trees: A Generalized Version of ID3," *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor MI, June 1988, pp. 100-106.

[Fisher89] D. H. Fisher and K. B. McKusick, "An Empirical Comparison of ID3 and Back-propagation," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989.

[Honavar88] V. Honavar and L. Uhr, "A Network of Neuron-Like Units that Learns to Perceive by Generation as Well as Reweighting of its Links," in *Proceedings of the 1988 Connectionist Models Summer School*, G. E. Hinton, T. J. Sejnowski and D. S. Touretzky (ed.), Morgan Kaufmann, San Mateo, CA, 1988.

[Kuchera67] H. Kuchera and W. N. Francis, *Computational Analysis of Modern-Day American English*, Brown University Press, Providence, RI, 1967.

[McClelland87] J. L. McClelland and D. E. Rumelhart, *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*, MIT Press, Cambridge, MA, 1987.

[Michalski80] R. S. Michalski and R. L. Chilausky, "Learning by Being Told and Learning from Examples: an Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis," *Policy Analysis and Information Systems 4*, 2 (June 1980), pp. 125-160.

[Minsky67] M. L. Minsky and S. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1967.

[Quinlan83] J. R. Quinlan, "Learning Efficient Classification Procedures and their Application to Chess End Games," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, T. M. Mitchell (ed.), Tioga Publishing Company, Palo Alto, CA, 1983.

[Quinlan86] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning 1*, 1 (1986), pp. 81-106.

[Reinke84] R. Reinke, *Knowledge Acquisition and Refinement Tools for the ADVISE Meta-Expert System*, M.S. Thesis, University of Illinois at Urbana-Champaign, 1984.

[Rendell89] L. A. Rendell, H. H. Cho and R. Seshu, "Improving the Design of Similarity-Based Rule-Learning Systems," *International Journal of Expert Systems 2*, 1 (1989), pp. 97-133.

[Rosenblatt62] F. Rosenblatt, *Principles of Neurodynamics*, Spartan, New York, 1962.

[Rumelhart86] D. E. Rumelhart, G. E. Hinton and J. R. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing, Vol. I*, D. E. Rumelhart and J. L. McClelland (ed.), MIT Press, Cambridge, MA, 1986, pp. 318-362.

[Schlimmer86] J. C. Schlimmer and D. Fisher, "A Case Study of Incremental Concept Induction," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 496-501.

[Sejnowski87] T. J. Sejnowski and C. R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text," *Complex Systems 1*, (1987), pp. 145-168.

[Shapiro87] A. Shapiro, *Structured Induction in Expert Systems*, Addison Wesley, Reading, MA, 1987.

[Shavlik89] J. W. Shavlik, R. J. Mooney and G. Towell, "Symbolic and Neural Net Learning Algorithms: An Experimental Comparison," Technical Report, Department of Computer Science, University of Wisconsin, Madison, WI, 1989.

[Tesauro88] G. Tesauro, "Connectionist Learning of Expert Backgammon Evaluations," *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor MI, June 1988, pp. 200-206.

[Utgoff88a] P. E. Utgoff, "Perceptron Trees: A Case Study in Hybrid Concept Representations," *Proceedings of the National Conference on Artificial Intelligence*, St. Paul, MN, August 1988, pp. 601-606.

[Utgoff88b] P. E. Utgoff, "ID5: An Incremental ID3," *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor MI, June 1988, pp. 107-120.

[Weiss89] S. M. Weiss and I. Kapouleas, "An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989.