An Algorithm to Evaluate Instance Representations

Sharad Saxena

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003.

COINS Technical Report 91-21
February 22, 1991

Abstract

The ability to generalize from examples depends on the algorithm employed for learning and the *instance representation* that describes the examples to the learning algorithm. This paper describes ACR, an algorithm to compare instance representations. Given a learning algorithm, a set of examples, and alternative instance representations for the examples, ACR identifies the instance representation that will enable the learning algorithm to produce the most accurate hypothesis. It works by estimating the minimum number of bits with which the learning algorithm can express the examples using the different representations. Experiments with two learning algorithms, including a well-known algorithm to build decision trees, show that ACR can correctly identify the best representation for a variety of tasks, including handwriting recognition.

1 Introduction

The ability to generalize from examples is central to the study of machine learning. Generalization depends on the algorithm employed for learning and the *instance representation* that describes the examples to the learning algorithm. Many efforts have been devoted to the design of learning algorithms, for instance Quinlan (1986), Rumelhart et. al. (1986), Utgoff (1988). However, the effect of the instance representation on generalization is poorly understood. This paper addresses the *representation evaluation problem*, which is:

Given: (a) A set of examples and alternative instance representations for the examples, (b) an algorithm for learning from examples.

Determine: The instance representation that enables the learning algorithm to produce a hypothesis that makes the most accurate predictions.

Section 3 describes an Algorithm to Compare Representations, called ACR, that solves this problem for a class of learning algorithms and tasks.

For example, consider the problem of learning to recognize handwritten characters. In the *pixel representation*, Casey & Nagy (1984) embed the characters in a grid and represent them by the listing the grid squares through which the pen passes. In the *fourier representation*, Arakawa et. al. (1978) track the pen position, and represent a character by the first few Fourier coefficients of the resulting sequence. Tappert et. al. (1990) list other representations. This diversity of representations used by different people indicates that the best instance representation is not always obvious. Section 4 reports ACR's ranking of the above representations for ID3, a well-known algorithm to construct decision trees (Quinlan, 1986), along with results of representation comparison for other problems.

A solution to the representation evaluation problem has the following applications. First, in automatic rule extraction from examples, only the hypothesis from the best instance representation need be used for classifying the future instances. The hypotheses produced from other representations can be rejected safely, alleviating the danger of making predictions with less accurate hypotheses. Second, while learning incrementally from a large database of labeled examples, one can begin learning in all the representations and stop learning in representations that are identified as worse than others. Third, *constructive induction* algorithms modify the given instance representation, or create new instance representations. They need some method to determine whether their actions have produced representations that result in more accurate hypotheses. An algorithm to rank instance representations can form a test component of a constructive induction algorithm that searches the space of representations in a generate-and-test manner.

ACR is based on the observation that modeling and data compression are duals of each other. A good model for the data enables short encodings to be reserved for the more likely data items. Dually, the models that permit the data to be compressed the most, capture best the underlying data generating machinery. Hence, ACR measures for each available instance representation, the ability of the learning algorithm to compress the examples. The representation that enables the examples to be compressed the most is considered to be the best. Section 2 gives formal and informal justifications for this view that generalization can be achieved by data compression.

The main characteristics of ACR are:

1. Representations are ranked based on the highest predictive accuracy a learning algorithm can achieve with the given set of examples. The ranking is *not* based on a single sample from the examples, which may be unrepresentative, or may be otherwise unsuitable because the learning algorithm underfits or overfits the sample.

2. ACR uses all the available examples to rank the representations. The examples are not divided into train and test sets; as would be required if one were to rank representations by estimating the error rate of the hypotheses produced with the different representations on an independent test set. This becomes important when only a small or moderate number of examples are available. In these situations, to get accurate error rate estimates one normally has to use *resampling* techniques, which are computationally more expensive than ACR.

3. Bad representations are identified quickly. The effort required to differentiate two good representations depends on how much better one representation is than the other.

The disadvantage of ACR is that in order to prove its correctness we need to make certain assumptions about the learning algorithms and the learning tasks. These assumptions are described below as they arise. However, in practice we have found ACR to be quite effective in ranking representations. It has been applied to rank instance representations of ten tasks for ID3, and eight tasks for the *Perceptron Tree* learning algorithm, which constructs decision trees that can have perceptrons at the leaves (Utgoff, 1988). In all the cases to which it has been applied, ACR has correctly ranked the instance representations whenever there is an appreciable difference in the accuracy of the hypotheses produced with the different representations.

2 Data Compression for Modeling

Informally, the data compression view can be justified by considering learning to be a process of information extraction. If one imposes an ordering on the elements of the domain of a function, then the information required to learn the function is the string that lists the value of the function on each of the domain elements. Call this string the *output string* of the function. The amount of information required for learning is the number of bits that it takes to specify the output string (Abu-Mostafa, 1986). If the output string is compressible, then it can be specified with fewer bits. By measuring the effectiveness of the learning algorithm in compressing the output strings that result from different instance representations, one can identify the instance representation that minimizes the amount of information needed by the learning algorithm to learn the corresponding function. Maciejowski (1979), Watanabe (1985), and Rendell (1986) have also pointed out the relationship between data compression and generalization.

Formally, Blumer et. al.'s (1987) *Occam's razor* result from learnability theory shows that in order to produce with high likelihood a hypothesis that is a good approximation of the function being learned, it suffices to find a hypothesis that can be expressed with much fewer bits than the number of bits required to enumerate the examples. Therefore, learning algorithms that generalize by producing compact hypothesis consistent with a set of examples are more likely to generalize better with instance representation that increase their ability to compress the examples. Rissanen's (1978,1989) *minimum description length principle* justifies the data compression view when one is not learning functions but modeling probabilistic phenomena. The application of *algorithmic information theory* to inductive inference justifies the data compression view when any Turing machine can be considered as a hypothesis (Li & Vitányi, 1989).

3 An Algorithm to Compare representations

The key idea of ACR is to determine for each instance representation, the minimum number of bits required to specify a consistent hypothesis for the examples. By the Occam's Razor result, the instance representation that permits a consistent hypothesis to be specified with the fewest bits will result in a hypothesis that makes the most accurate hypothesis. Call the number of bits required to specify an object is called its *codelength.*

To estimate the minimum codelength required to specify a consistent hypothesis, ACR observes the growth in the codelength as random samples of increasing size are drawn from the available examples and presented to the learning algorithm. From a particular sample, the given learning algorithm produces a hypothesis that is correct on some examples and incorrect on others. To make this hypothesis consistent, one imposes an arbitrary order on the examples and records a 0 for all examples for which the hypothesis is correct and an 1 otherwise, along with the appropriate correction. The list marking the position of errors is called the *error vector.* For instance, suppose the target function is the logical AND of $x1$ and $x2$, and the hypothesis is $x1$. For a lexicographic ordering of the examples ($01 < 10$), the error vector is [0, 0, 1, 0]. If by drawing a different sample the hypothesis changes to $x2$, the error vector becomes [0, 1, 0, 0]. The presence of error at a particular position in the error vector is indicated by a random variable whose value depends on the hypothesis. For a given ordering of the examples, the hypothesis produced by the learning algorithm augmented with the location and the value of the corrections is a consistent hypothesis. The codelength required to specify this hypothesis is the sum of the codelength of the original hypothesis and the codelength required to specify the corrections. The description of the procedure for ordering the examples is a fixed overhead, which can be ignored in the comparisons. This procedure is identical to Rissanen's (1989) and Wallace & Freeman's (1989) two part encoding.

As more examples are presented, the learning algorithm may produce a hypothesis that is correct on a larger number of examples. Fewer errors decreases the codelength of the errors. However, it may increase the codelength needed to specify the hypothesis. One can be sure that the minimum value of codelength has already been observed if it can be established that the codelength cannot decrease further. This can happen in two ways:

1. There are no regularities in the examples that can be utilized by *any*

learning algorithm to reduce the total codelength.

2. The given learning algorithm cannot further utilize any remaining regularities. Consequently, the total codelength increases for hypotheses constructed with a larger number of examples.

The next two subsections describe tests to detect these conditions. Given these tests, the algorithm to compare two representations $R1$ and $R2$ is:

1. In a *run*, draw random samples of increasing size from the examples available in each instance representation and present them to the given learning algorithm. For each sample, determine the codelength of the resulting consistent hypothesis. Conduct the two tests mentioned above and terminate the run if it is established that the codelength cannot decrease any more.

2. Conduct multiple runs to ensure that representations are not ranked based on unrepresentative samples. Conduct runs until it is established with a desired confidence that one representation is better than another.

Specifically, a run returns the pair $(v1, v2)$, which is $(1, -1)$ if $R1$ is better, $(-1, 1)$ if $R2$ is better and $(1, 1)$ otherwise. If in repeated runs the decision about the better representation changes then $v1 - v2$ is equally likely to be 1, -1 or 0. A large number of positive or negative values indicates that one representation is better than another. After each run, a ***sign test*** determines whether there is a significantly greater proportion of positive values or negative values seen so far (Gibbons, 1971). The hypothesis of equal likelihood is rejected if there is a large number of values of one type, and the probability of observing so many values of one type by chance is small, 0.01 in our experiments. More than two representations are ranked by ranking all pairs.

In effect, ACR solves a number of small learning problems in order to estimate the minimum codelength. The cost of a run is the sum of the time taken by the learning algorithm to construct a hypothesis from the sample, time required classify the N examples, and $O(NlogN)$ steps to perform the two tests mentioned above. This is a saving because, firstly one does not have to build a hypothesis for all sample sizes in order to determine the one that produces the most accurate model. Secondly, one does not have to resort to expensive resampling methods to get a reliable estimate of the accuracy of the hypotheses produced for a sample size (Weiss & Kulikowski, 1991).

3.1 A test to determine that the codelength will not decrease

Intuitively, the idea of this test is that if there is no structure present in the examples, then by taking more examples to form the hypotheses, one cannot reduce the total codelength further. Any decrease in the codelength of the error vector will be accompanied by at least an equal increase in the codelength required to describe the hypothesis.

To detect that the codelength will not decrease further, we make three assumptions. First, the learning algorithm always produces a hypothesis consistent with the examples presented to it, and in general, a hypothesis consistent with a set E of examples cannot be described with fewer bits than the number of bits required to specify a hypothesis formed from a subset of E. Second, the learning task is such that small additions to a set of examples produce hypotheses that are "close" to the original hypothesis: in the sense that if the size of the new hypothesis is increased by r bits then one can produce the new hypothesis by specifying r bits of corrections to the original hypothesis. Third, the random variables marking the errors in the error vector are normally distributed, and that the joint distribution of these random variables is the same throughout the error vector.

We need the above assumptions to prove ACR's correctness. It may be difficult to ascertain that these assumptions hold for the specific learning algorithm and task at hand. However, in practice, we have found the test described below to be very effective in determining that the total codelength will not decrease below a certain value. The empirical success of this test can be seen as either indicating that above assumptions are satisfied for the cases studied here; or, as indicating that this test is a good *heuristic* for establishing that the total codelength will not decrease further.

The test is based on Shannon's (1948) noiseless coding theorem for any coding method that permits multiple items to be encoded by concatenating codes for individual items. By this theorem, if the errors are statistically independent, then, on average, the codelength of the error vector cannot be less than the product of its length and its entropy. Call this the *ideal codelength* of the errors.

Suppose after observing a set of examples E the errors become independent; then by adding more examples to E the total codelength cannot be decreased below the sum of the codelength required to describe the hypothesis formed from E and the ideal codelength of the error vector. Otherwise, by describing changes to the hypothesis formed from E and the new error vector, one can recover the old error vector from fewer bits than its ideal codelength. This follows from the first and second assumptions above. Thus, the minimum total codelength observed until the errors become independent is a lower bound on the total codelength that can be obtained using any hypothesis formed from a set of examples containing E.

The normality assumption lets the error-vector to be tested for independence based on its *auto-correlation function*, which measures the correlation between random variables denoting the errors separated by different distances, or *lags*. The auto-correlation coefficient at lag k is the correlation between errors separated by a distance of k. The independence test is based on the fact that for normally distributed, zero mean random variables, zero auto-correlation for all non-zero lags implies independence. To test for independence of finite sequences, Chatfield (1984) recommends looking at the first few lags, and to consider an auto-correlation coefficient *significant* if it falls outside a desired confidence interval around zero. Furthermore, to consider a sequence of random variables independent if the number of significant coefficients is below a threshold determined by the confidence interval. In our experiments, one fourth of the coefficients are analyzed and the error vector is considered independent if less than 5% of the coefficients fall outside a 95% confidence interval around zero.

3.2 A test to determine that the codelength is increasing

It could happen that though there are some regularities in the examples, either the language used to describe the hypothesis is inadequate to express them, or the learning algorithm is not able to find a hypothesis that expresses them. A symptom of this situation is that the total codelength increases as hypotheses are constructed from samples of larger size. Assume that there is no "critical" number of examples after which the ability of the learning algorithm to generalize increases. Now if one establishes that there is a statistically significant trend that the total codelength increases as hypotheses are constructed from samples of larger size, then the minimum seen so far is a lower bound on the total codelength required to express the examples. If there is a critical sample size, and it is known, then the behavior of the total codelength should be observed with sample sizes greater than the critical size.

A *non-parametric* test of the correlation between two variables, called

Kendall's rank test, is used to determine whether total codelength increases with sample size (Gibbons, 1971; Kendall, 1962). The idea behind this test is that if there is no relationship between the number of examples and the codelength, then as the number of examples increases the codelength is equally likely to increase or decrease. However, if one observes that codelength increases with sample size, then one can determine the likelihood of getting such an observation by chance. The hypothesis that codelength does not depend on sample size is rejected and the hypothesis that it increases with sample size is accepted, if the probability of getting the observed increase in codelength with the sample size is below a desired value: 0.01 in our experiments.

4 Experiments

This sections reports the results of applying ACR to rank representations for ID3 (Quinlan, 1986). The experimental methodology is the following:

1. The available examples are randomly partitioned into two sets. 90% of the examples form the *training set*, used for predicting the better representation. 10% of the examples form the *test set*, used for validating the prediction[1]. Quinlan & Rivest's (1989) method gives the codelength of a decision tree and the codelength of an error vector.

2. Random samples of different sizes drawn from the training set and presented to the learning algorithm. The fraction of the test set correctly classified by the hypothesis produced by the learning algorithm is recorded as the *accuracy* of the hypothesis. ACR's prediction is considered correct if the representation predicted to be the best produces a hypothesis that has higher accuracy than the hypotheses produced by the other representations.

The following tasks were considered:
The Clumps Problems: This is a class of problems introduced by Denker et. al. (1987) to illustrate the effect of instance representation on learning. The task is to learn to recognize various patterns of contiguous blocks of black pixels, or black *clumps*, in an one dimensional visual field, which is bordered by two white pixels. In the *pixel representation*, the visual field is represented

[1]The partitioning of the instances is just for experimental purposes. It is not required by ACR, which predicts the better representation for any given set of examples.

Problem name	ACR's prediction
Two-or-more-clumps	Edge better than Pixel
Two-clumps	Edge better than Pixel
tic4	Movs better than Locs
tic1	Locs better than Movs
Handwriting-1	Pixel better than Cross
Handwriting-2	Fourier better than Pixel

Table 1: Ranking the instance representations for ID3.

by the color of each pixel. The *edge representation* marks the transition between a black and a white pixel. The two problems considered here are the *two-or-more-clumps* and the *two-clumps* problem.

Tic-Tac-Toe problems: These are six sets of boards from the game of Tic-Tac-Toe. The two problems reported here are *tic1*, the set of legal boards on which the player on move can win, and *tic4*, the set of legal boards on which the player on move can prevent the opponent from setting up a fork— that is, prevent a win by the opponent in 2-ply. The *location-contents representation (Locs)* enumerates the contents (X, O or blank) of each location of the board, locations being numbered left to right, top to bottom. In the *move-sequence (Movs)* representation, one numbers the moves 1 to 9 and gives the location of each move, denoting by a '?' moves not yet made.

Handwriting recognition: The examples are a set of numerals handwritten on a computer several times by different people using a mouse. The pixel representation and the fourier representation are as described before. Here, the fourier representation is the first five fourier coefficients of the sequences of X and Y coordinates of the pen position. Another representation is a simplification of the *cross representation*, which represents a character by combining the representation of the different strokes that form the character. Here, one embeds a stroke in a box, divides the box into predetermined regions, and represents a stroke by the starting region and number of times it crosses various regions (Newman & Sproull, 1979). Our simplification is to consider each character as a single stroke.

Table 1 shows ACR's ranking of the representations for the above problems. Figure 1 shows the accuracy of the decision trees produced by ID3 with the different representations. The accuracies are averages over multiple runs and the differences are significant at the 99% level via the t-test. Observe that in

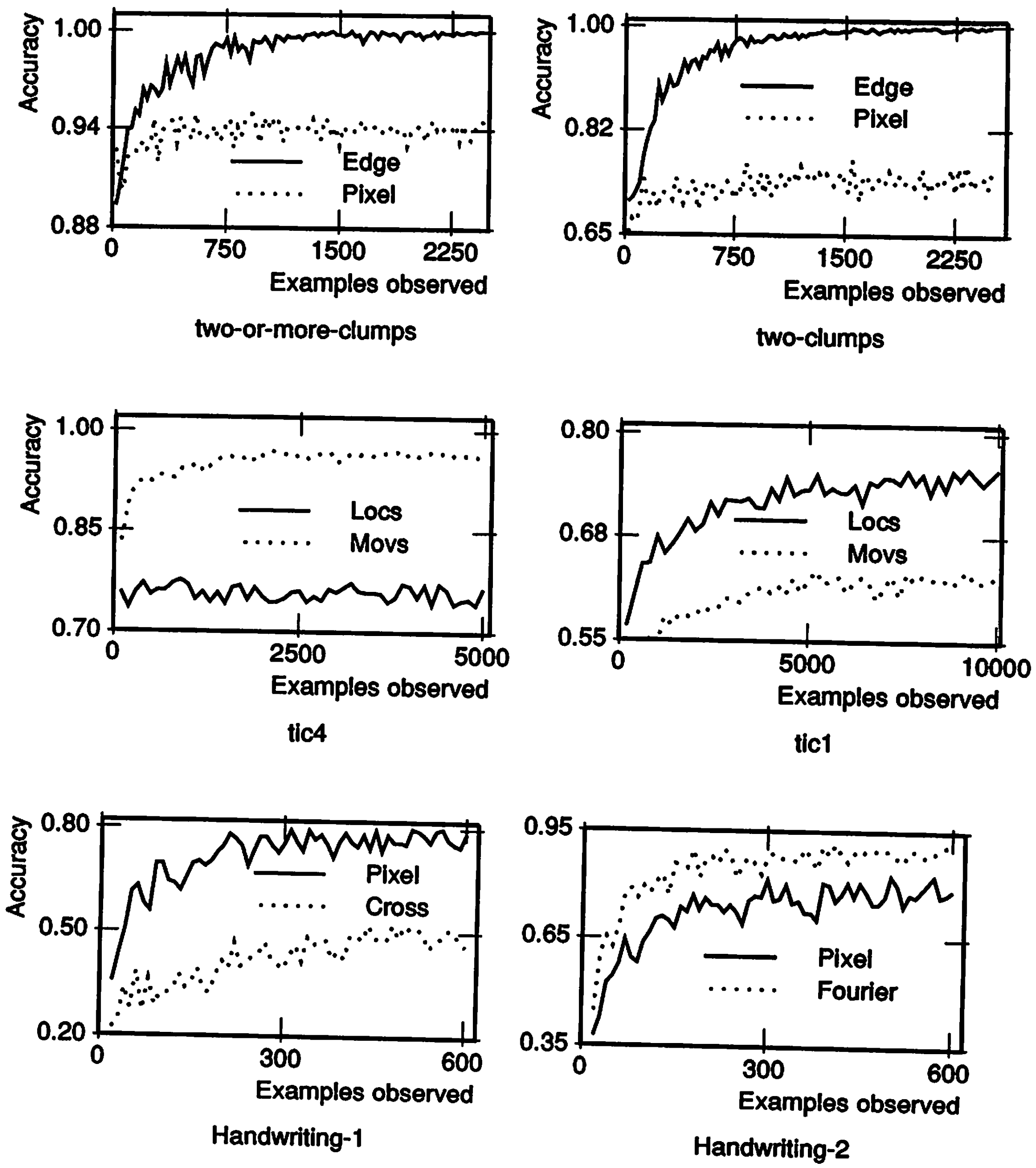

Figure 1: The accuracy of ID3 on the test sets.

every case the representation predicted to be better by ACR results in decision trees that are more accurate in classifying the test set.

These results, along with the results obtained for perceptron trees and for the other four Tic-Tac-Toe problems (Saxena, 1991), show that ACR ranks representations correctly whenever there is a significant difference in the accuracies of the hypotheses produced by the different representations. We take this as empirical support for concluding that ACR is effective in ranking representations for the ID3 and the Perceptron Tree learning algorithms.

5 Related Approaches

Very few techniques for representation evaluation are known. The methods of feature selection from *statistical pattern recognition* are not directly applicable. These methods try to select features based on how well they discriminate a given set of examples (Kittler, 1986). However if each representation describes a function, as is the case here, then any given set of examples can be perfectly discriminated, by remembering them if needed, and they would be considered equally good by these criteria.

ACR is closest in spirit to the work of Gao & Li (1989) on handwriting recognition. They want the best interval for recording the successive positions of the pen. They report that the best accuracy on a test set is obtained for the interval that minimizes the sum of the codelength required to store a set of instances in a database and the codelength for the corrections when these characters are classified by their matching scheme. Our perspective on their work is that rather than using description length to choose the best model, by changing the feature selection interval they change the instance representation. By taking this view, we have been able to generalize it so that we now have an algorithm to compare representations for any task and learning algorithm that satisfies our assumptions. Also in ACR, rather than deciding the better representation based on a single sample, which could be unrepresentative or otherwise unsuitable because the learning algorithm under or overfits it, the sample size is systematically increased to estimate the minimum codelength.

6 Conclusions

The main contribution of this paper is ACR, an algorithm to rank the available representations of a task based on how well the given algorithm for learning from examples will generalize with the various representations. Experiments with ACR on a variety of tasks, including handwriting recognition, show that it ranks the instance representations correctly whenever there is an appreciable difference between the accuracy of the hypotheses produced with the various representations. To prove ACR's correctness, we need four assumptions about the learning algorithms and the learning tasks. However, its success in ranking representations for the ID3 and Perceptron Trees algorithms indicates that it is a useful tool for studying representations.

This paper is a further explication of the data compression view of generalization. ACR is an application of results from learnability theory. We believe that the information extraction view of learning will result in a better understanding of the problems that arise in learning from examples. In particular, it can help in addressing the issues that arise in the design and analysis of good instance representations. Determining why a learning algorithm is unable able to compress the examples can give ideas about improving the instance representation.

Acknowledgements

This paper is based on work supported by the National Science Foundation under the Grant No. IRI-8619107 and by the ONR, through an University Research Initiative under contract number N00014-86-K-0764. Professors Paul Utgoff, Andy Barto, and Dave Barrington gave many valuable suggestions during this research. Discussions with Jamie Callan, Richard Yee, Margie Connell, David Haines, Tom Fawcett, Carla Brodley, and Jeff Clouse have helped to improve this work.

References

Abu-Mostafa, Y. S. (1986). The complexity of information extraction. *IEEE Transactions on Information Theory, 32*, 513-525.

Arakawa, H., Odaka, K., & Masuda, I. (1978). On-line recognition of handwritten characters—Alphanumerics, Hiranga, Katakana, Kanji. *Proceedings of the Fourth International Joint Conference on Pattern Recognition* (pp. 810-812).

Blumer, A., Ehrenfeuct, A., Haussler, D., & Warmuth, M. K. (1987). Occam's Razor. *Information Processing Letters, 24*, 377-380.

Casey, R. G., & Nagy, G. (1984). Decision tree design using a probabilistic model. *IEEE Transactions on Information Theory, 30*, 93-99.

Chatfield, C. (1984). *The Analysis of Time Series: an Introduction (Third Edition)*. New York: Chapman and Hall.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Hopfield, J., Howard, R., & Jackel, L. (1987). Large automatic learning, rule extraction, and generalization. *Complex Systems, 1*, 877-922.

Gao, Q., & Li, M. (1989). An application of minimum description length principle to online recognition of handprinted alphanumerals. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. Detroit, Michigan: Morgan Kaufmann.

Gibbons, J. D. (1971). *Nonparametric Statistical Inference*. New York: MaGraw Hill.

Kendall, M. G. (1962). *Rank Correlation Methods*. New York: Hafner Publishing Company, Inc.

Kittler, J. (1986). Feature selection and extraction. In T. Z. Young, & K. S. Fu (Eds.), *Handbook of pattern recognition and image processing*. New York: Academic Press.

Li, M., & Vitányi, P. M. B. (1989). Inductive reasoning and Kolmogorov complexity (Preliminary Version). *Proceedings of the Fourth Annual IEEE Structure in Complexity Theory Conference* (pp. 165-185).

Maciejowski, J. M. (1979). Model discrimination using an algorithmic information criterion. *Automatica, 15*, 579-593.

Newman, W. M., & Sproull, R. F. (1979). *Principles of interactive computer graphics (second edition)*. New York: MacGraw Hill.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81-106.

Quinlan, J. R., & Rivest, R. L. (1989). Inferring decision trees using the minimum description length principle. *Information and Computation, 80*, 227-248.

Rendell, L. (1986). A general framework for induction and a study of selective induction. *Machine Learning, 1*, 177-226.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica, 14*, 465-471.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry.* New Jersey: World Scientific.

Rumelhart, D. E., Hinton, G. E., & Williams, R.J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition.* Cambridge, MA: MIT Press.

Saxena, S. (1991). *Data Compression for Modeling* (Doctoral thesis, in preparation). Amherst, MA: University of Massachusetts, Computer and Information Science Department.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*, 379-423.

Tappert, C. C., Suen, C. Y., & Wakahara, T. (1990). The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*, 787-808.

Utgoff, P. E. (1988). Perceptron trees: A case study in hybrid concept representations. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 601-606). Saint Paul, MN: Morgan Kaufmann.

Wallace, C. S., & Freeman, P. R. (1989). Estimation and Inference by Compact Coding. *Journal of Royal Statistical Society, Series B, 49*, 240-265.

Watanabe, S. (1985). *Pattern recognition: Human and mechanical.* New York: Wiley & Sons.

Weiss, S. M., & Kulikowski, C. S. (1991). *Computer systems that learn.* Palo Alto: Morgan Kaufmann.