A Hybrid Genetic Algorithm for Classification

James D. Kelly, Jr.
Bolt Beranek and Newman, Inc.
10 Moulton Street
Cambridge, MA 02138
jkelly@bbn.com

Lawrence Davis
Tica Associates
36 Hampshire Street
Cambridge, MA 02139

Abstract

In this paper we describe a method for hybridizing a genetic algorithm and a k nearest neighbors classification algorithm. We use the genetic algorithm and a training data set to learn real-valued weights associated with individual attributes in the data set. We use the k nearest neighbors algorithm to classify new data records based on their weighted distance from the members of the training set. We applied our hybrid algorithm to three test cases. Classification results obtained with the hybrid algorithm exceed the performance of the k nearest neighbors algorithm in all three cases.

1 Introduction

There has been a great deal of progress recently in the development of automated classification techniques. Many of the new techniques have arisen from combinations of artificial intelligence and statistical classification approaches. In this paper we describe such an approach. We have developed an improvement on a classical technique, the k nearest neighbors algorithm, that we believe shows a good deal of promise. Our improvement involves using an artificial intelligence technique, a genetic algorithm, to enhance the performance of the classical algorithm.

The k nearest neighbors algorithm classifies a new instance by noting its distance from each member of a database of classified examples and assigning the new instance to the class of the majority of its nearest neighbors. This algorithm can be quite effective when the attributes of the data are equally important. It can be less effective when many of the attributes are misleading or irrelevant to the classification. The approach we describe here involves finding a vector of weightings of the attributes that make the distance measure more meaningful.

It is not a simple matter to find an optimal vector of attribute weightings. In this paper we show how to use a real-valued genetic algorithm to find vectors of attribute weightings that are good in the sense that they out-perform a vector composed of identical values.

Below we describe the nearest neighbors algorithm, the k nearest neighbors algorithm, the weighted k nearest neighbors algorithm, the genetic algorithm, our hybrid classifier, and the performance of our hybrid algorithm on three classification problems.

In one respect this work is similar to the technique of memory-based reasoning, described in [Stanfill and Waltz, 1986]. There the authors detail a method for reasoning about new instances which involves the weighting of attribute-value pairs based on their frequency of occurrence. Our algorithm learns attribute weights based on demonstrated classification importance. Both techniques are based on the need to vary attribute influence for classification. While the work of Stanfill and Waltz focuses on the development of a general reasoning technique, ours is concerned with the optimization of a classical statistical classification technique.

A technique for the analysis of attributes from the field of statistics is principal components analysis [Jolliffe, 1986]. The focus of principal components analysis, however, is on the variability of attribute values and variance may or may not relate to an attribute's classification utility.

2 The K Nearest Neighbors Classification Algorithm

The *nearest neighbor* classification algorithm (NN) is based on the idea that, given a data set of classified examples, an unclassified individual should belong to the same class as its nearest neighbor in the data set. The measurement of proximity, or similarity, between two individuals is a subject of much research (e.g., [Vosniadou and Ortony, 1989]). Although this is an important issue for classification, our research did not focus on this problem. Our implementations of nearest neighbor algorithms used the Euclidean distance metric, with which the distance between two data points i and j is computed as follows:

$$d_{ij} = \{\sum_{a=1}^{n}(X_{ia} - X_{ja})^2\}^{\frac{1}{2}} \qquad (1)$$

where X_{ia} is the value of the a^{th} attribute for the i^{th} datum.[1] The techniques we describe below for improving the performance of these algorithms will be effective no matter what distance metric is employed.

A common extension to the nearest neighbor algorithm is to classify a new instance by looking at its it *nearest neighbors* $(k > 1)$. Then the unclassified individual is assigned

[1]A preprocessing step for all data is the conversion to standard units (0 mean, unit standard deviation).

to the class of the majority of its k closest neighbors. There are various ways to adjudicate disputes among an instance's neighbors, the most basic being a simple vote among the k nearest neighbors. Our techniques should be effective regardless of the specific conflict resolution technique employed.

The k nearest neighbors algorithm (KNN) is simple, quick, and often effective. There are many cases in which its performance is at least as good as other, more sophisticated algorithms. It can go wrong, however, in cases in which the proportion of attributes that are significant in the classification process is small with respect to the number of attributes that are irrelevant or misleading. Given a vector of attributes, the KNN algorithm effectively treats each as equal in the process of classification. In terms we develop below, the KNN algorithm uses a weighting vector in which each attribute is equally weighted

3 Weighted Attributes and the KNN Algorithm

Assigning variable weights to the attributes of the instances before applying the KNN algorithm distorts the space, modifying the importance of each attribute to reflect its relevance for classification. In this way, closeness or similarity with respect to important attributes becomes more critical than similarity with respect to irrelevant attributes. Figure 1 graphically displays a simple example of the effect of attribute weighting. In this example an incorrect classification is corrected when the influence of an an attribute is decreased. (Consider the classification of the datum at the center of both diagrams if it were not known to be "unfilled circle". In the left diagram it is closest to the filled circle datum and thus would be classified incorrectly. The right diagram shows the effect of decreasing the influence of attribute Y from 1 to 0.5: classification of the middle datum based upon the nearest neighbor is now correct.) When attributes are appropriately weighted, the performance of the KNN algorithm will not be degraded It is possible that the optimal weight vector will consist of identical values, as is the case in the KNN algorithm. In this case the performance of the KNN and Weighted KNN algorithm (WKNN) algorithms will be identical. In all other cases, a WKNN algorithm with an optimal weight vector will outperform a KNN algorithm.

The distance metric we use for the WKNN is a slight variant of the Euclidean metric (Equation 1, above), where w_a are attribute weights:

$$d_{ij} = \{\sum_{a=1}^{n} w_a(X_{ia} - X_{ja})^2\}^{\frac{1}{2}} \qquad (2)$$

We also extended the mechanism in which the k nearest neighbors determine the class of the target datum. As there is an ordering of neighbors from 1 to k based on calculated distances, each of the k neighbors should not necessarily have an equal influence in classification decisions. It is likely that the vote of the closest neighbor should have more influence than the k^{th} neighbor. Thus, in addition to associating weights with attributes for use in the calculation of inter-datum distances, we also associate weights with the

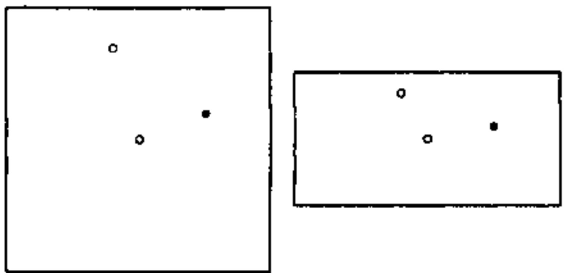

Figure 1: Example of the Effect of Attribute Weighting.

k nearest neighbors for use in the actual class determination. The voting strength V for a class i is calculated by:

$$V_i = \sum_{j=1}^{k} \begin{cases} W_j & \text{if } C_j = i \\ 0 & \text{if } C_j \neq i \end{cases} \qquad (3)$$

where W_j is the weight associated with the j^{th} nearest neighbor and C_j is the j^{th} neighbor's class. The class of the focus datum is determined to be that class with the largest V.

Although the performance of the WKNN algorithm will nearly always exceed the performance of a KNN algorithm given an optimal vector of attribute weights, an optimal vector of attribute weights is difficult to find. One method is to manually set weights. In addition to the difficulties involved in such a process, the manual setting of weights is inherently biased and may preclude the discovery of unforeseen relationships. Brute force exploration of the search space is another method, however it can be quite time-consuming. If there are n instance attributes and we allow only 5 values for each weight, then there are 5^n possible vectors to consider. In the biocriteria database described below, there were 17 attributes. Exhaustive test of the weight vectors for this problem, even considering only five values for each weight, would have involved the consideration of 5^{17} vectors and taken months of computer time.

These considerations show why the KNN algorithm, which requires no training time, is often employed: it generates good solutions quickly. These considerations also show why variably-weighted solutions have not in general been employed: finding high-performance weight vectors can take a tremendous amount of time.

In the next two sections we describe an approach to learning attribute weight vectors. Our technique uses a genetic algorithm to leam vectors of variable weights. While the genetic algorithm does not necessarily find the optimal weight vector for a problem, it does generate vectors which produce levels of classification performance superior to that obtained with the KNN algorithm.

4 Genetic Algorithms

Genetic algorithms are optimization and machine learning algorithms based loosely on processes of biological evolution. John Holland created the genetic algorithm field, and

[Holland, 1975] is the first major genetic algorithm publication.

Interest in genetic algorithms has increased recently in conjunction with an increase in interest in other algorithms based on natural processes, including simulated annealing and neural networks. For further source material on genetic algorithms the reader is referred to any of the recent proceedings of genetic algorithm conferences, to David E. Goldberg's textbook on genetic algorithm theory [Goldberg, 1989], or to Lawrence Davis's book on the application of genetic algorithms to optimization problems [Davis, 1991].

Put simply, genetic algorithms (GAs) solve optimization problems by manipulating a population of *chromosomes,* encoded solutions to the problem. Each chromosome is assigned a *fitness* that is related to its success in solving the problem. Given an initial population of chromosomes, a genetic algorithm proceeds by choosing chromosomes to serve as parents and then replacing members of the current population with new chromosomes that are (possibly modified) copies of the parents. The process of reproduction and population replacement goes on until a stopping criterion (the achievement of a performance target or the usage of an allotted amount of CPU time, for instance) has been met.

Genetic algorithms have two important features that underlie their success. The first is their employment of an algorithmic equivalent of natural selection. When chromosomes are chosen as parents during the reproduction process, the probability that a given chromosome will be chosen is biased in accord with its fitness. Thus, the fittest chromosomes (those that solve the problem best) will tend to have more children than the less fit ones. The use of fitness-based reproduction generally leads to an improvement in the population as a genetic algorithm runs. The second feature is the use *of mutation* and *crossover* operators during reproduction. Mutation operators cause children to differ from their parents through the introduction of localized change. Crossover operators create children that combine chromosomal matter from two parents. The production of high-performance chromosomes can be greatly speeded up with crossover working to combine subparts of good solutions from multiple parents on a single child.

There is a good deal of art and theory to account for the success of genetic algorithms in solving hard optimization problems. For discussions of these matters, the reader is referred to the aforementioned source material on genetic algorithms.

The discovery of effective weight vectors for a WKNN algorithm is a hard optimization problem with a very large search space. This is just the sort of problem that genetic algorithms have been shown to be good at, and so it seemed to us that the hybridization of a GA with a WKNN algorithm was a likely candidate for a high-performance classification algorithm.

The genetic algorithm we used differed from those that have been employed in more classical studies in the field. In Holland's work and in that of his students the encoding of numbers on chromosomes is done in binary notation. A growing body of research ([Goldberg, 1990] and [Davidor, 1990], for instance) suggests that problems like this one, in which good solutions tend to occur close to each other instead of being distributed periodically over the search space, may be more efficiently solved with genetic algorithms that use real-valued chromosomes. We employed real-valued chromosomes here.

The real-number genetic algorithm we used is an early version of one of the genetic algorithm programs which accompanies [Davis, 1991]. It is a genetic algorithm applicable to a wide variety of real-number optimization problems. The parameter settings and operator set were not tailored to the WKNN domain. It is possible that domain-based heuristics could be added to the suite of genetic algorithm operators, particularly in the biocriteria test set described below. We did not add such heuristics in this stage of our research, preferring to use the more generic version of a genetic algorithm.

5 The GA-WKNN Algorithm

The GA-WKNN algorithm combines the optimization capabilities of a genetic algorithm with the classification capabilities of the weighted k nearest neighbors algorithm. The goal of the algorithm is to leam an attribute weight vector which improves KNN classification. Specifics of the GA-WKNN algorithm are:

- Chromosomes are vectors of real-valued weights. Each chromosome is a vector of decimal numbers between 0 and 1 inclusive. A vector value is associated with each classification attribute and one is associated with each of the k neighbors. Thus the length of the vector is the number of attributes plus k. The initial population of chromosomes in each run of the GA-WKNN algorithm was randomly generated.
- Chromosomes are evaluated by iterating through each data set element and classifying each datum by using its associated weights in Equation 2, to determine the k closest neighbors, and then in Equation 3, to make the class determination. These computations can be used in a number of ways to rank chromosomes; we used two evaluation functions in the experiments reported here:
    1. Number of Misclassifications. This method sums the number of data which were assigned to an incorrect class by the GA-WKNN algorithm. Thus, for any chromosome an evaluation of 0 is optimal. Chromosome X is ranked higher than chromosome Y if the number of misclassifications it generates is lower.
    2. Multiple Value Ranking. Among chromosomes which generate equal numbers of misclassifications there are finer degrees of difference. This method orders such chromosomes by additionally considering 1) the number of k neighbors which are of the same class *(k same neighbors), 2)* the total distance to the *k same neighbors,* 3) the number of k neighbors which are not of the same class *(k different neighbors),* and 4) the total distance to the *k different neighbors.* While there are many ways to use these factors to rank chromosomes, we settled on the following cascaded method, which ranks chromosome X higher than chromosome Y if:

$$\begin{array}{lll} nm_X & < & nm_Y \quad \textit{or;} \\ \\ nm_X & = & nm_Y \quad \textit{and} \\ nks_X & \geq & nks_Y \quad \textit{and} \\ dks_X & < & dks_Y \quad \textit{and} \\ dkd_X & > & dkd_Y \quad \textit{or;} \\ \\ nm_X & = & nm_Y \quad \textit{and} \end{array}$$

$$(dks_X/nks_X + dkd_X/nkd_X) < (dks_Y/nks_Y + dkd_Y/nkd_Y)$$

Where :

nm	=	*total number of misclassifications,*
nks	=	*total number of k same neighbors (maximum equals k $\times$ number of examples),*
nkd	=	*total number of k different neighbors,*
dks	=	*total distance to the k same neighbors, and,*
dkd	=	*total distance to the k different neighbors.*

In the experiments reported here, the GA-WKNN algorithm learns a single weight vector which is used to discriminate all classes of data. A natural extension of our technique is to train a vector of weights for each class and assign new instances to the class whose weight vector produces the closest neighbors.

6 Experimentation

The primary focus of our experimentation was a comparison of the classification performance of the GA-WKNN algorithm to that of the KNN algorithm. A secondary issue was a comparison of the two chromosome ranking functions. Many empirical studies of classification (e.g., [Weiss and Kapouleas, 1989]) have compared the performance of a broad range of techniques, including the KNN algorithm, and our results can be interpolated into the results of such studies.

In this section we discuss the specific GA-WKNN operating parameter settings used in our experiments, our data sets and testing methods, and our results.

6.1 GA-WKNN parameters

Each run of the genetic algorithm maintained a population of size 50. The runs terminated after 600 individuals had been produced. The algorithm used the steady-state without duplicates reproduction technique [Whitley, 1988], [Syswerda, 1989].

The operators used were *uniform crossover, average values, real number mutation,* and two varieties of *real number creep,* one with a large creep range and one with a smaller creep range. The parameter settings of these operators were as follows. Real number mutation replaced a field on a chromosome with a 10% probability. The new number was randomly generated from the interval between 0 and 1. The first creep operator *(large creep)* altered a chromosome field with 10% probability. The amount of the alteration was a randomly generated number between 0 and 0.25 in magnitude. The second creep operator *(small creep)* altered a chromosome field with 5% probability. The amount of the alteration was a randomly generated number between 0 and 0.1 in magnitude. The creep operator altered values up or down with equal probability.

Only one of these operators was involved in any reproduction event. The number of parents used and children created in a reproduction event depended on the operator employed. The uniform crossover and average values operators used two parents. Uniform crossover produced two children and average values produced one. The other three operators, real number mutation and large and small creep, used one parent and produced one child. The relative probabilities that these operators would be selected for use in a reproduction event was held constant at 30%, 6%, 20%, 29%, and 15%, respectively, over the course of the run. These values had been found to perform well over a range of real number optimization problems during the research reported in [Davis, 1991].

Fitness was assigned in the following way. The population was rank ordered using one of the two ranking functions described above. Then each member was assigned a fitness from the series $[1000c^0, 1000c^1, 1000c^2, \ldots, 1000c^{50}]$, except that where any of these values fell below 1 it was replaced by 1. The value of c was not held constant during the runs. At the beginning of each run c was set to 0.95. At the end, c was set to 0.7. Intermediate values were the result of interpolating between 0.95 and 0.7. The effect of this technique is to produce mild pressure in favor of the best population members when the run begins. The curve steepens over the course of the run as the population of solutions converges on similar individuals. The steeper curve increases the selection pressure, causing the algorithm to focus more and more on the best individuals in the population.

While we experimented with various k values throughout this research, we did not do an exhaustive search for optimal settings. For the sake of consistency we set k = 3 during our experiments.

6.2 Testing

The GA-WKNN algorithm was tested using three data sets, one of which, biocriteria, is previously only described in State of Ohio Environmental Protection Agency (EPA) documents [Ohio EPA, 1987].

- IRIS The iris data set, Fisher's classical test data [Fisher, 1936], contains attributes of three types of iris plants. Each of the 150 examples in this data set is described by four attributes.
- GLASS This data set consists of attributes of glass samples taken from the scene of an accident.[2] Each of the 214 examples is a member of one of six classes. There are nine attributes.
- BIOCRITERIA This data set contains biological measurements, taken at surface water sites by the State of Ohio EPA, and the associated pollution impact type. It contains 463 examples, with 17 attribute values associated with each, such as number of sunfish species and total number of deformities, eroded fins, and/or lesions present on all species. The eight decision classes

[2]Collected by B. German of the Home Office Forensic Service at Aldermaston, Reading, Berkshire in the UK.

represent principal pollution impact types such as complex municipal/industrial pollution and combined sewer overflows. The data has a high noise level, due in part to sampling methods and the fact that most sites are impacted in multiple ways. In addition, naturally-occurring events such as droughts and floods impact the attributes of water sites and may mask the effect of human pollution.[3]

We used the *cross validation* error estimation technique, described in [Breiman *et al*„ 1984]. Each data set was divided into five partitions. The only constraint on otherwise random partitioning was that classes be represented equally in each partition. We generated five training/test sets for each data set. Four-fifths of the data were used for training and the remaining fifth was used for testing. Thus no training example was used as a test example in our experiments. For each data set we experimented with misclassification ranking versus multiple value ranking. As a result, we ran two different GA-WKNN algorithms for five different partitions of each of the three data sets, for a total of thirty different experiments.

63 Results and Discussion

Figure 2 summarizes our results. Column 1 contains the benchmark performance of the k nearest neighbors algorithm (i.e., each attribute weighted equally). Each cell contains the test set classification error rates, averaged across the five partitions.

Our primary result is that the GA-WKNN algorithm outperforms the KNN algorithm on these data. In all six comparisons (KNN vs. GA-WKNN for both ranking functions for all three data sets) the GA-WKNN algorithm has a lower test set error rate than KNN. For random events such results have an occurrence probability of less than *2%.* The genetic algorithm has indeed found weighting vectors that, while " not optimal, nonetheless produce better classification performance given the parameter settings above. Allowing the genetic algorithm to spend more time on the data improves its performance, although the amount of improvement falls off as the amount of additional time increases. This is an interesting feature of the genetic algorithm that is different from many other classification techniques: one can allot the algorithm whatever optimization resources one has available, with the expectation of obtaining better results the more resources one allots.

A secondary result is that in two-thirds of the trials the multiple value ranking function outperforms the number of misclassifications ranking function. While this result is not statistically significant, we believe that this occurs because the multiple value ranking function provides a gradient in cases in which two chromosomes produce the same number of misclassifications. We found that genetic algorithms ranked by the number of misclassifications tended to wander aimlessly when the best population members generated equal numbers of mismatches. Genetic algorithms using the multiple value ranking function tended to improve long

The collection and analysis of such biological data is a branch of environmental research which is focused on the development of standards for biological integrity, to be used in conjunction with more standard chemical assessments.

	KNN	GA-WKNN	
		Ranking Function:	
		Number of Misclassi-fications	Multiple Value
IRIS	0.1000	0.0600	0.0733
GLASS	0.4154	0.3972	0.3785
BIOCRITERIA	0.6371	0.6328	0.6198

Figure 2: Classification Error Rates.

after genetic algorithms using the number of misclassifications ranking method were stalled. The multiple value function takes longer to compute than the number of misclassifications function, but the difference is not great and performance improvement may warrant using the more time-consuming technique.

For experimental purposes an identical stopping criterion was used for each training trial (the completion of 600 reproduction/evaluation cycles). Iris, the smallest data set in terms of number of attributes and number of instances, generated the best results (i.e., largest decrease in KNN error rate by GA-WKNN). While the rate of performance improvement decreases as more cycles are allotted, one conjecture supported by the comparison of the Iris results with those from the other two data sets is that an optimal stopping criterion is related to the data set size; as the number of data points (attributes x instances) increases, there must be a corresponding increase in the resources allocated to the genetic algorithm.

Our experiments were run on a Symbolics Ivory-based Lisp machine. Rough operating times were on the order of ten minutes for the smallest data set and one hour for the largest (biocriteria). The minimization of operating time was not a goal of this experimentation and certain software optimizations will decrease operating times. But even the aforementioned times are reasonable for many applications and not a prohibitive factor to the use of the GA-WKNN algorithm.

Finally, although the GA-WKNN algorithm improves on KNN performance in each case, the amount of improvement is incremental. However, improvements are consistently yielded, and this fact demonstrates the potential of the algorithm. For problems in which the k nearest neighbors classification technique outperforms other techniques, we have shown that additional performance increments can be obtained relatively cheaply. For many applications such performance improvements can provide a critical increment of success.

7 Conclusions

We have described an algorithm that hybridizes the classification power of KNN algorithms with the search and optimization power of the genetic algorithm. The result is an algorithm that requires computational capabilities above that of the KNN algorithm, but achieves improved classification performance in a reasonable time. We anticipate that extensions to the research will improve the algorithm's performance and there are a number of issues that we plan to address in further work, including:

- Alternative distance/similarity metrics;
- Alternative K values;
- Formal characterization of the example sets on which this algorithm outperforms other classification techniques such as ID3 and CART,
- Other techniques for learning real-valued weight vectors;
- The correspondence of the learned weights to the feature selection/variable selection problem;
- Class-based derivation of attribute weight vectors;
- Incorporation of domain knowledge into the genetic algorithm in the form of heuristic operators; and
- Performance issues.

Several of these issues will be addressed in [Kelly and Davis, 1991], including the use of a rotation parameter, in addition to the scaling described here, and a comparison to ID3 classification results.

Acknowledgements

Discussions with Ken Anderson, Albert Boulanger, Herb Gish, William Salter, and Gilbert Syswerda contributed significantly to the evolution of the ideas expressed in this paper. Charles T Walbridge of the United States Environmental Protection Agency suggested the application of genetic algorithms to the biocriteria domain. Chris Yoder and Edward Rankin of the State of Ohio Environmental Protection Agency provided the biocriteria data and feedback on classification results. Finally, the authors thank Bolt Beranek and Newman, Inc., Systems and Technologies Division for supporting this research.

References

[Breiman *et al,* 1984] L. Breiman, J. Freidman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wads worth, Monterrey, CA, 1984.

[Buntine, 1990] W. L. buntine. A *Theory of Classification Rules.* Ph. D. Thesis, School of Computing Science, University of Technology, Sydney, 1990.

[Davidor, 1990] Y. Davidor. *Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization.* World Scientific Publishing Co., Singapore, 1990.

[Davis, 1987] L. Davis (editor). *Genetic Algorithms and Simulated Annealing.* Pitman, London, 1987.

[Davis, 1989] L. Davis. Adapting Operator Probabilities in Genetic Algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms,* Morgan Kaufman, San Mateo, CA, 1989.

[Davis, 1991] L. Davis (editor). *The Handbook of Genetic Algorithms.* Van Nostrand Reinhold New York, 1991.

[Duda and Hart, 1973] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[Everitt, 1974] B. Everitt *Cluster Analysis.* Heinemann, London, 1974.

[Fisher, 1936] R.A. Fisher. Multiple Measurements in Taxonomic Problems. *Annals of Eugenics,* VII:179-188, 1936.

[Goldberg, 1989] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[Goldberg, 1990] D.E. Goldberg. Real-coded GAs, Virtual alphabets, and Blocking. ILL1GAL Report 90001, Dept. of General Engineering, University of Illinois, Urbana-Champaign, Illinois, September 1990.

[Holland, 1975] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[Jolliffe, 1986] I.T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, New York, 1986.

[Kelly and Davis, 1991] J. Kelly and L. Davis. Extensions to a Genetic Algorithm/K Nearest Neighbors Classification Algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithms,* Morgan Kaufman, San Diego, CA, 1991, (Forthcoming).

[Ohio EPA, 1987] Ohio Environmental Protection Agency. Biological Criteria for the Protection of Aquatic Life. Ohio EPA, Division of Water Quality Monitoring and Assessment, Vols. 1,11,111, 1987.

[Stanfill and Waltz, 1986] G. Stanfill and D. Waltz. Toward Memory-Based Reasoning. *Communications of the ACM,* 29(12):1213-1228, 1986.

[Syswerda, 1989] G. Syswerda. Uniform Crossover in Genetic Algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms,* Morgan Kaufman, San Mateo, CA, 1989.

[Vosniadou and Ortony, 1989] S. Vosniadou and A. Ortony (editors). *Similarity and Analogical Reasoning.* Cambridge University Press, Cambridge, 1989.

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

[Whitley, 1988] D. Whitley. GENITOR: A Different Genetic Algorithm. In *Proceedings of the Rocky Mountain Conference on Artificial Intelligence,* Denver, CO, 1988.