Noise-Tolerant Instance-Based Learning Algorithms

David W. Aha and Dennis Kibler*
Department of Information and Computer Science
University of California, Irvine, CA 92717
aha@ics.uci.edu kiblerOics.uci.edu

Abstract

Several published reports show that instance-based learning algorithms yield high classification accuracies and have low storage requirements during supervised learning applications. However, these learning algorithms are highly sensitive to noisy training instances. This paper describes a simple extension of instance-based learning algorithms for detecting and removing noisy instances from concept descriptions. This extension requires evidence that saved instances be significantly good classifiers before it allows them to be used for subsequent classification tasks. We show that this extension's performance degrades more slowly in the presence of noise, improves classification accuracies, and further reduces storage requirements in several artificial and real-world database applications.

1 Introduction

Instance-based learning (IBL) algorithms have several notable characteristics. They employ simple representations for concept descriptions, have low incremental learning costs, have small storage requirements, can produce concept exemplars on demand, can learn continuous functions [Kibler *et al.*, 1989], and can learn non-linearly separable categories, IBL algorithms have been successfully applied to such varied tasks as speech recognition [Bradshaw, 1987], handwritten letter identification [Kurtzberg, 1987], the cart-and-pole problem [Connell and Utgoff, 1987], power load forecasting [Jabbour *et al.*, 1987], and thyroid disease diagnosis [Kibler and Aha, 1987]. However, they are highly sensitive to noise.

The ability to tolerate noise is a necessity for robust, practical learning methods. Algorithms should demonstrate graceful degradations in performance when presented with noisy data. Pruning methods, based on tests of statistical significance, were developed to allow decision tree learning algorithms to tolerate noisy data [Quinlan, 1986, Niblett and Bratko, 1986]. Similar noise-tolerant methods have been developed for rule learning algorithms [Clark and Niblett, 1987].

*This work was partially supported by a grant from the Hughes AI Research Center, Malibu, CA.

This paper introduces an extension for IBL algorithms, also based on a form of significance testing, that identifies and eliminates noisy concept description instances. We show that the resulting algorithms' classification accuracies degrade linearly with linear increases in noise in an artificial domain and improves classification performance on several complicated domains. We also compare the performances of our algorithms with previous IBL algorithms and an extension of ID3 [Quinlan, 1986] that tolerates noise.

2 Instance-Based Learning Algorithms

IBL algorithms induce neither rules, decision trees, nor other types of abstractions. Instead, instance-based concept descriptions are represented solely by a set of instances. In this paper, each instance is represented by a set of attribute-value pairs - a point in the *instance space*. IBL algorithms incrementally derive their concept descriptions from a sequence of training instances. Classifications are made with respect to the concept description's extension, which is derived with respect to a similarity function and a classification function.

2.1 A Framework for IBL Algorithms

More precisely, all IBL algorithms consist of the following three components:

1. *Similarity Function:* Given two normalized instances, this yields their numeric-valued *similarity.*
2. *Classification Function:* Given an instance *i* to be classified and its similarity with each saved instance, this yields a *classification* for *i.* (In this paper, a classification is expressed as a concept name.)
3. *Memory Updating Algorithm:* Given the instance being classified and the results of the other two components, this updates the set of saved instances and their classification records.

Each instance is *normalized* to ensure that attributes are assigned equal importance by the similarity function. Assuming that attributes have equal importance is not necessarily correct, but it is a fair approach without prior knowledge of relative attribute saliencies.

All IBL algorithms in this paper define the similarity of two instances as the negation of their Euclidean distance in the instance space. Comparisons of different similarity functions is left for future research. These

Table 1: Names and components of the eight IBL algorithms. Asterisks denote those algorithms for which noise-tolerant extensions are used in this paper (NTGrowth, NT K-nn Growth, & NT Disjunctive Spanning).

Memory Updating Function	Classification Function	
	nearest neighbor	k-nearest neighbor
Saves All Instances	Proximity	K-nn
Instance-Filtering	Growth*	K-nn Growth*
Instance-Averaging	Disjunctive Spanning*	–

Table 2: The Growth IBL algorithm: Deriving concept description C from training set T.

- Initialize C to the singleton set of T's first instance
- $\forall$ subsequent training instances $t \in T$:
  1. Find the nearest neighbor n of t in C
  2. IF (t is classified correctly by n)
     THEN discard t
     ELSE add t to C

algorithms also employ the same algorithm for tolerating missing attribute values. Calculating the similarity of two instances involves computing their pairwise attribute-value differences. If either value of a pair is missing, then they are assumed to be maximally different from each other.

The IBL algorithms described in this paper employ either the *nearest neighbor* or *k-nearest neighbor* classification function. The former classifies an instance as being a member of the same concept as its most similar instance. The latter does the same, but takes a majority vote among its *k* most similar instances (we set *k* to 3).

2.2 A Family of IBL Algorithms

The eight IBL algorithms described in this paper, summarized in Table 1, differ primarily in their memory updating functions. The simplest IBL algorithms (Proximity and K-nn) save all training instances. Since most real-world domains exhibit regularities that make thern amenable to storage-reducing algorithms, the remaining IBL algorithms save only misclassified training instances (which are assumed to contain additional concept boundary information). Storage-reducing IBL algorithms differ in how they update memory when a training instance is correctly classified. *Instance-filtering* algorithms discard correctly classified instances while *instance-averaging* algorithms replace the classifying instance with an average of it and the instance being classified. The Growth (Table 2) and Disjunctive Spanning [Bradshaw, 1987] IBL algorithms are identical except that the former performs instance-filtering while the latter does instance-averaging. We will also experiment with the same instance-filtering variant of the K-nn algorithm and noise-tolerant versions for the three storage-reducing algorithms (NTGrowth, NT Disjunctive Spanning, and NT K-nn Growth).

Table 3: NTGrowth IBL algorithm: Deriving concept description C from training set T.

- Initialize C to the singleton set of T's first instance
- $\forall$ subsequent training instances $t \in T$:
  1. Find the nearest *acceptable* neighbor n of t in C
  2. IF (t is classified correctly by n)
     THEN discard t
     ELSE add t to C
  3. Update the classification records of all instances in C at least as similar to t as n
  4. Drop from C those instances that appear to be noisy

There are several reasons to experiment with these eight algorithms. Variants of the Growth algorithm have recorded excellent classification accuracies and small storage requirements in several task domains [Hart, 1967, Kurtzberg, 1987, Kibler and Aha, 1987]. Bradshaw [1987] reported similar results for his Disjunctive Spanning algorithm. However, both algorithms are sensitive to noise. We wanted to extend these storage-reducing algorithms to tolerate noise. Hence we also experimented with NTGrowth and NT Disjunctive Spanning.

Analyses of the k-nearest neighbor algorithm suggest that it is a more accurate classifier than the nearest neighbor algorithm [Duda and Hart, 1973]. However, the noise-sensitive K-nn instance-filtering algorithm (K-nn Growth) should also benefit from the noise-tolerant extension. Thus, this study also includes three IBL algorithms that use the k-nearest neighbor classification function (K-nn, K-nn Growth, and NT K-nn Growth).

3 The Noise-Tolerant Extension

The NTGrowth algorithm (Table 3) is a noise-tolerant extension of the Growth algorithm. NT Disjunctive Spanning and NT K-nn Growth are similar extensions of their respective algorithms. These noise-tolerant algorithms differ from their respective storage-reducing algorithms in three respects:

1. they maintain classification records for all saved instances (i.e., the number of correct and incorrect classifications of subsequent training instances),
2. only those saved instances with *significantly* good classification records are acceptable for use in subsequent classification tasks, and
3. the noise-tolerant algorithms discard those saved instances that appear to be noisy (i.e., those instances whose classification performance is poor after several classification attempts).

For each training instance t, classification records are updated for all saved instances that are at least as similar as t's most similar *acceptable* neighbor.[1]

During the initial stages of training, none of the saved instances are acceptable. In order to more closely rnimic: the behavior of the algorithms when at least one instance is acceptable, only a randomly chosen number of most similar saved instances' classification records are updated.

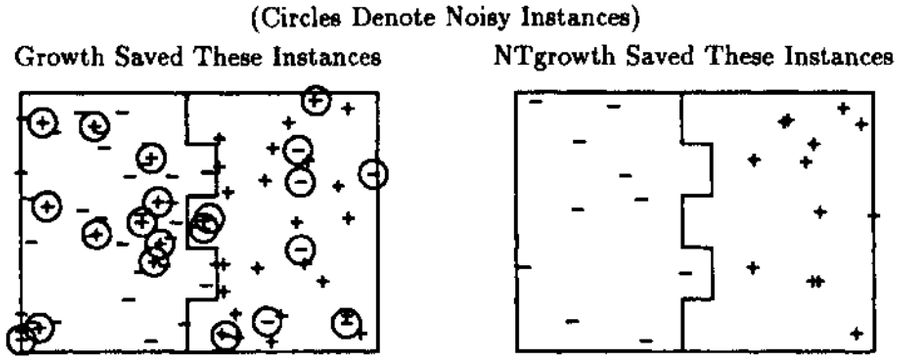

Figure 1: Growth and NTGrowth saved these instances when applied to 250 training instances with 10% noise. (NTGrowth accepted no noisy instances!) This instance space has 2 numeric attributes. The boundary between the positive and negative disjuncts is the rectilinear line shown in this instance space.

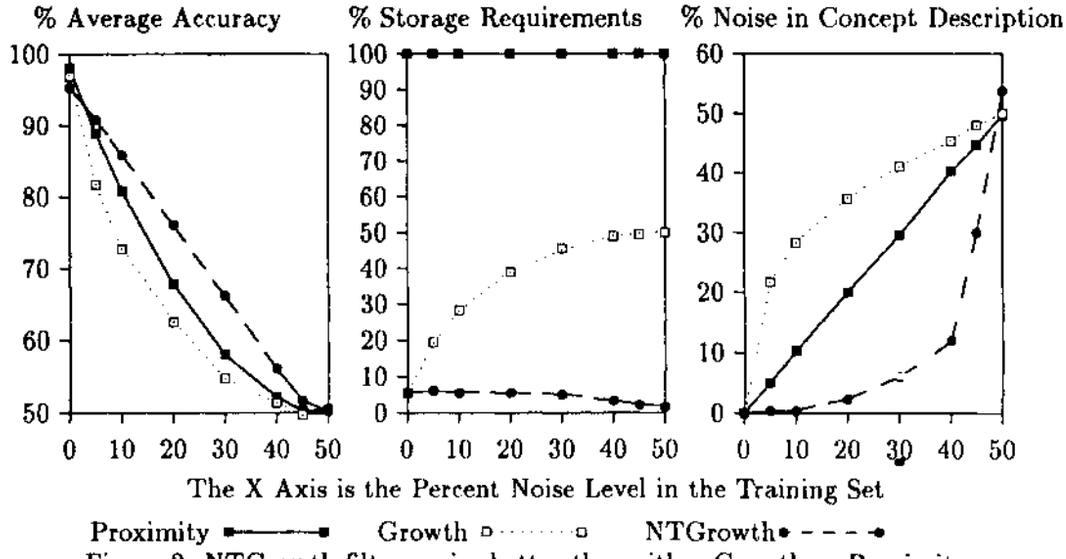

Figure 2: NTGrowth filters noise better than either Growth or Proximity.

These extensions employ a significance test to decide whether saved instances are acceptable, noisy, or neither. Instances are acceptable if their classification accuracy is statistically significantly greater than their class' observed frequency and dropped if it is statistically significantly less. Confidence intervals are constructed around both the instance's current accuracy and its class' current frequency. If the accuracy interval's lowest value is greater than the class frequency interval's greatest, then the instance is accepted. Similarly, instances are dropped when their accuracy interval's highest value is less than their class frequency interval's lowest. Confidence intervals are constructed using formula 5.5-4 in [Hogg and Tanis, 1983, page 296].

We designed the extensions to make it difficult for an instance to be accepted by employing a high (90%) confidence for acceptance. However, we selected a lower (75%) confidence level for dropping since we would like to drop those instances with even moderately poor classification accuracies.

4 Benefits of the Extension

We trained Growth and NTGrowth on 250 randomly drawn instances from a 2-dimensional instance space containing one concept. Figure] reveals which instances were saved by each algorithm when each training instance's class was mislabeled with a probability of 10%. In this trial, Growth saved 74 instances, 21 of which were mislabeled. The saved noisy instances invariably recorded poor classification accuracies on subsequent training instances. Since NTGrowth accepts only those instances with significantly good classification accuracies, we expected it to distinguish the noisy instances from those with good classification records.

In fact, Figure 1 shows that NTGrowth successfully filtered noisy instances from the concept description. All instances accepted by NTGrowth had good classification accuracies (at least a 70% accuracy in this case). NTGrowth's performance in this example was typical. Averaged over 50 trials, 28.3% of Growth's saved instances were noisy while only a mere 0.5% of NTGrowth's accepted instances were mislabeled (Figure 2).

Table 5: Percent average accuracy/percent average storage requirements results (over 50 trials). Storage results are not given for Frequency and C4.

Algorithm	Database Number					
	1	2	3	4	5	6
Frequency	10.0/-	33.3/-	54.1/-	63.9/-	54.8/-	24.8/-
Proximity	72.2/100	74.7/100	76.2/100	59.5/100	91.7/100	31.6/100
Growth	63.1/43.5	70.3/31.8	70.5/29.9	56.6/36.0	90.7/11.4	28.2/73.0
NTGrowth	72.0/28.7	74.7/14.1	77.9/7.4	78.6/7.5	91.9/7.5	35.4/15.0
Disjunctive Spanning	61.5/44.6	72.4/29.3	71.3/28.3	58.4/35.9	92.4/9.4	26.2/73.5
NT Disjunctive Spanning	71.4/27.7	76.6/12.0	78.7/7.2	73.1/12.3	92.9/6.4	34.6/15.4
K-nn	60.5/100	77.6/100	79.2/100	64.9/100	86.2/100	28.9/100
K-nn Growth	52.6/53.7	72.0/31.2	71.1/30.4	59.4/33.5	86.4/14.6	26.3/75.1
NT K-nn Growth	36.1/17.0	77.8/18.6	79.4/11.0	80.8/11.5	85.2/12.2	24.0/7.4
C4	68.1/-	71.1 /-	75.4/-	75.4/-	94.5/-	36.9/-

Table 4: Database characteristics.

Database Name	#	Train Size	Test Size	Number Attributes	Number Classes
LED Display	1	200	500	7	10
Waveform	2	300	500	21	3
Cleveland	3	250	53	13	2
Hungarian	4	250	44	13	2
Voting	5	350	85	16	2
Tumor	6	250	89	17	22

We experimented with the Proximity, Growth, and NTGrowth algorithms on this same instance space. The results (averaged over 50 trials per noise setting) are summarized in Figure 2. Our purpose was to discover how these algorithms' performances and concept descriptions degrade with increasing amounts of noise, which was varied from 0% to 50%. The three dependent variables were classification accuracy, storage requirements (number of instances in concept descriptions), and the quality of the concept descriptions (the percentage of concept description instances that were mislabeled).

1. *Classification Accuracy:* While the three algorithms performed equally well with no noise, NTGrowth's accuracy degraded more slowly (linearly) with increasing noise levels.

2. *Storage Requirements:* The Proximity algorithm saved all training instances. Growth's storage requirements were much lower, asymptoting towards 50%. However, NTGrowth's were significantly lower than Growth's and asymptoted towards zero. This was expected: since none of the saved instance's accuracies were significantly good at high noise levels, NTGrowth accepted only a few of them into the concept description.

3. *Concept Description Quality:* The percentage of noisy instances in the Proximity algorithm's concept description increased linearly with the noise level. Growth's percentage of noise in the concept description rose far more quickly. However, NTGrowth's filtering effect drastically slowed the influx of noisy instances into its concept description.

In summary, the noise-tolerant extensions assume that the classification records of noisy instances will distinguish them from non-noisy instances. Noisy instances will have poor classification accuracies because their nearby neighbors in the instance space will invariably have other classifications.

5 Experiments and Results

NTGrowth's performance degraded more slowly than the other two algorithms in these and other experiments with artificial domains. This encouraged us to test the noise-tolerant extensions on six more challenging domains to see if these benefits recur during more practical applications. For comparison purposes, C4, Quinlan's modification of ID3 [Quinlan, 1986] that performs pruning, was also tested. The databases' characteristics are given in Table 4. The average results are summarized in Table 5. (We have also included the benchmark algorithm Frequency, which always guesses the class with the highest frequency. This algorithm provides a comparative measure of the other algorithm's utilities.) The instances chosen for the disjoint training and test sets were always randomly selected from the databases.

The LED Display and Waveform domains [Breiman *et al.*, 1984] are artificial domains with large amounts of noise (each LED attribute value has a 10% chance of being noisy and all Waveform attribute values contain an added noise factor). For both domains, the three noise-tolerant extensions easily outperformed their respective unextended algorithms. (However, NT K-nn Growth required 500 LED training instances to reach a

64.7% accuracy.) They also recorded equally good classification accuracies and incomparably lower storage requirements than their respective all-instance saving algorithms (Proximity and K-nn).

The Cleveland and Hungarian databases consist of cardiological records recorded at the Cleveland Clinic Foundation and Hungarian Institute of Cardiology respectively. These domains contain a great deal of noise; Detrano [1988] reported that his discriminant analysis method for predicting heart disease resulted with accuracies of approximately 75%. The NT algorithms again significantly outperformed the other algorithms.

The voting domain contains small amounts of noise. Therefore the payoff of the noise-tolerant algorithms was smaller than in more noisy domains. Finally, while NTGrowth and NT Disjunctive Spanning performed well on the primary tumor domain, NT K-nn Growth required more training instances to perform well.

In summary, the noise-tolerant extensions of both the Growth and Disjunctive Spanning algorithms always recorded higher classification accuracies and lower storage requirements than their ancestor algorithms. Also, their classification accuracies were always as good or better than Proximity's. However, while NT K-nn Growth always recorded low storage requirements, the average learning curves generated from these experiments indicate it is a much slower learner than its ancestor algorithms. Only when given enough instances was it able to achieve accuracies as good as or better than the K-nn and K-nn Growth algorithms.

NTGrowth, NT Disjunctive Spanning, and C4 recorded the most consistently high classification accuracies among the ten algorithms (i.e., NTGrowth was within 3% and the others within 8% of the highest accuracy recorded for each database). This indicates that these two noise-tolerant IBL algorithms should perform well in a large number of database applications.

6 Advantages and Limitations

We suspect that instance-based and decision tree learning algorithms can learn the same classes of concepts, namely those whose disjuncts have shapes in instance space that can be described by piecewise linear approximations [Kibler and Aha, 1988]. However, IBL algorithms have certain behavioral advantages. First, most decision tree and rule learning algorithms can form only hyper-rectangular approximations of concept boundaries in instance space. After extensive training, these hyper-rectangular partitions form more detailed approximations that closely resemble IBL's more general piecewise linear approximations. Decision tree algorithms probably learn more slowly when the concepts' boundaries are not aligned with the attribute dimensions' axes. In fact, two studies have reported that IBL algorithms learn faster than decision tree algorithms [Shepard, 1983, Aha, 1989]. An extension of the CART decision tree learning algorithm employs linear combinations of attributes (i.e., perceptrons) at its nodes to form piecewise linear approximations [Breiman *et* a/., 1984, Section 5.2]. However, this extension is expensive, requiring an explicit search for the best hyperplane separator in the instance space for each node's instances. IBL algorithms appear to be more convenient for learning some types of concepts.

Second, IBL algorithms are cost-effective incremental learning methods, requiring only $0(|I| \times \backslash A \backslash)$ attribute examinations to update the concept description for a single instance, where $\backslash I \backslash$ is the size of the training set and instances are described with $\backslash A \backslash$ attributes. (IBL storage-reducing algorithms require significantly fewer attribute examinations.) In comparison, both C4 and an incremental variant, ID4 [Schlimmer and Fisher, 1986], require at most $0(|J| \times \backslash A \backslash^2)$ attribute examinations to incorporate a single instance. IDS, another incremental variant, [Utgoff, 1988] requires only $0(|A|^2)$ examinations plus $0(|A_i| \times \backslash Aj \backslash \times |A|)$ additional examinations for each pullup, where IAil and $\backslash Aj \backslash$ are the sizes of the two attribute domains involved in ID5's pullup process. However, this includes neither the costs for recursive pullups, numeric-value partitioning, nor pruning.

Finally, Aha [1989] indicated that decision tree algorithms build huge trees when learning a set of concept descriptions that have different sets of relevant attributes. Aha presented Bloom, an extension of NTGrowth that addresses this problem by learning the relative attribute relevancies and building an independent concept description for each concept. (This allows Bloom to represent independent, overlapping, graded, and non-exhaustive concept descriptions.) Therefore, a better decision tree strategy may be to build a separate tree for each concept (or set of related concepts).

C4 recorded higher classification accuracies than NTGrowth in only two of the six experiments. However, the noise-tolerant IBL algorithms do not summarize their concept descriptions. We are currently working on solutions to this problem, including combining the instance-based and decision tree approaches, which would allow the IBL algorithms to (1) generate concise concept description summaries and (2) exploit an instance-indexing hierarchy so that similarities are computed only for a priori-known similar instances, thus further reducing incorporation (and classification) costs. The decision tree algorithm would benefit from reduced storage and incremental learning costs. Case-based reasoning indexing schemes (e.g., [Bareiss and Porter, 1987]) may likewise assist the IBL approach, while our noise-tolerant algorithms could assist in judging the utility of cases. However, our noise-tolerating methods don't distinguish noisy instances from exceptions. While our methods might detect noisy-looking cases, further analyses should be made when this distinction is critical.

7 Conclusions

This paper described a noise-tolerant extension for instance-based learning algorithms. We showed that the NTGrowth algorithm's performance degrades more gracefully, in the presence of noise, than does the performance of previous instance-based algorithms. In addition, the noise-tolerant extensions recorded lower storage requirements and higher classification accuracies than previous instance-based algorithms on several domains (both artificial and real-world). These gains occurred be-

cause the noise-tolerant extensions decreased the number of noisy instances used in classification decisions.

The key contribution of this paper was the introduction of a simple voting method, combined with a statistical test, to assist in the detection and removal of noisy instances from concept descriptions. This method, like the pruning of decision trees [Quinlan, 1986, Niblett and Bratko, 1986] and the testing of the quality of CN2's complexes [Clark and Niblett, 1987], is based upon a simple significance test. Our method, which tolerates noise by gathering evidence of correctness before employing information for classification decisions, is a representation-independent technique. An amalgamation of IBL algorithms with those that yield compilations (in the forms of rules or decision trees) should result with a superior learning algorithm having lower updating costs, lower storage requirements, and the ability to present concept descriptions concisely.

Acknowledgements

Thanks to Peter Clark, Doug Fisher, Wayne Iba, Fat Langley, Haym Hirsh, Jeff Schlimmer, Jim Wogulis, David Ruby, Marc Albert, and Stephanie Aha for their valuable comments on previous drafts of this paper. We would like to thank M. Zwitter and M. Soklic (Institute of Oncology, Ljubljana, Yugoslavia) for donating the primary tumor database, Robert Detrano (V.A. Center, Long Beach, CA) for the Cleveland database, and Andras Janosi (Hungarian Institute of Cardiology, Budapest) for the Hungarian database. These are among the 38 databases available from the U.C.I. Repository of Machine Learning Databases.

References

[Aha, 1989] D. W. Aha. Incremental, instance-based learning of independent and graded concept descriptions. In *Sixth International Workshop on Machine Learning,* Detroit, MI, 1989. Morgan Kaufmann.

[Bareiss and Porter, 1987] E. R. Bareiss and B. W. Porter. Protos: An exemplar-based learning apprentice. In *Fourth International Workshop on Machine Learning,* pages 12-23, Irvine, CA, 1987. Morgan Kaufmann.

[Bradshaw, 1987] G. Bradshaw. Learning about speech sounds: The nexus project. In *Proceedings of the Fourth International Workshop on Machine Learning,* pages 1-11, Irvine, CA, 1987. Morgan Kaufmann.

[Breiman *et al,* 1984] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees.* Wads worth International Group, Belmont, CA, 1984.

[Clark and Niblett, 1987] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning,* 3:261-285,1987.

[Connell and Utgoff, 1987] M. E. Connell and P. E. Utgoff. Learning to control a dynamic physical system. In *Proceedings of the Sixth National Conference on Artificial Intelligence,* pages 456-460, Seattle, WA, 1987. Morgan Kaufmann.

[Duda and Hart, 1973] It. 0. Duda and P. E. Hart. *Pattern classification and scene analysis.* John Wiley & Sons, Menlo Park, CA, 1973.

[Hart, 1967] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory,* 13:515-516,1967.

[Hogg and Tanis, 1983] R. V. Hogg and E. A. Tanis. *Probability and statistical inference.* Macmillan Publishing Co., Inc., New York, NY, 1983.

[Jabbour *et al.,* 1987] K. Jabbour, J. F. V. Riveros, D. Landsbergen, and W. Meyer. ALFA: Automated load forecasting assistant. In *Proceedings of the 1987 IEEE Power Engineering Society Summer Meeting,* San Francisco, CA., 1987.

[Kibler and Aha, 1987] D. Kibler and D. W. Aha. Learning representative exemplars of concepts: An initial case study. In *Proceedings of the Fourth International Workshop on Machine Learning,* pages 24-30, Irvine, CA, 1987. Morgan Kaufmann.

[Kibler and Aha, 1988] D. Kibler and D. W. Aha. Comparing instance-averaging with instance-filtering learning algorithms. In *Proceedings of the Third European Working Session on Learning,* pages 63-80, Glasgow, Scotland, 1988. Pitman Publishing.

[Kibler *et al.,* 1989] D. Kibler, D. W. Aha, and M. Albert. Instance-based prediction of real-valued attributes. *Computational Intelligence,* May 1989.

[Kurtzberg, 1987] J. M. Kurtzberg. Feature analysis for symbol recognition by elastic matching. *I.B.M. Journal of Research and Development,* 31:91 95, 1987.

[Niblett and Bratko, 1986] T. Niblett and I. Bratko. Learning decision rules in noisy domains. In M. A. Bramer, editor, *Research and Development in Expert Systems III.* Cambridge University Press, Brighton, England, 1986.

[Quinlan, 1986] J. R. Quinlan. Induction of decision trees. *Machine Learning,* 1:81-106, 1986.

[Schlimmer and Fisher, 1986] J. C. Schlimmer and D. Fisher. A case study of incremental concept induction. In *Proceedings of the Fifth National Conference on Artificial Intelligence,* pages 496-501, Philadelphia, PA, 1986. Morgan Kaufmann.

[Shepard, 1983] B. Shepard. An appraisal of a decision tree approach to image classification. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* pages 473-475, Karlsruhe, West Germany, 1983. William Kaufmann.

[Utgoff, 1988] P. Utgoff. ID5: An incremental ID3. In *Proceedings of the Fifth International Conference on Machine Learning,* pages 107-120, Ann Arbor, MI, 1988. Morgan Kaufmann.