Detecting and Removing Noisy Instances from Concept Descriptions

David W. Aha
Dennis Kibler

Irvine Computational Intelligence Project
Department of Information and Computer Science
University of California, Irvine, CA 92717

Technical Report 88–12

5 May 1988
Revised: 12 December 1988

Submitted to IJCAI 1989 as "Noise-Tolerant Instance-Based Learning Algorithms"

Noise-Tolerant Instance-Based Learning Algorithms

David W. Aha   Dennis Kibler

Department of Information & Computer Science
University of California, Irvine
Irvine, CA 92717 U.S.A.
(714) 856-8779
aha@ics.uci.edu   *kibler@ics.uci.edu*

Requested Reviewing Area: Learning (B2)

Abstract

Several published results show that instance-based learning algorithms record high classification accuracies and low storage requirements when applied to supervised learning tasks. However, these learning algorithms are highly sensitive to training set noise. This paper describes a simple extension of instance-based learning algorithms for detecting and removing noisy instances from concept descriptions. The 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 databases.

1 Introduction

Instance-based learning (IBL) algorithms have several notable characteristics, including representational simplicity, low incremental learning costs, small storage requirements, the ability to learn continuous functions (Kibler & Aha, in press), the ability to learn non-linearly separable categories, and the ability to produce concept exemplars on demand. 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 & Utgoff, 1987), power load forecasting (Jabbour, Riveros, Landsbergen, & Meyer, 1987), and thyroid disease diagnosis (Kibler & Aha, 1987). However, instance-based learning algorithms are highly sensitive to noise.

The ability to tolerate noise is an important requisite for robust, practical learning methods. Algorithms should demonstrate graceful degradations in performance when presented with noisy data. Pruning methods, based upon tests of statistical significance, have been developed for tolerating noisy data in decision tree learning algorithms (Quinlan, 1986; Niblett & Bratko, 1986). Clark & Niblett (1987) subsequentially described a similar method for tolerating noise in CN2, their rule learning algorithm.

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.

2 Instance-Based Learning Algorithms

IBL algorithms induce neither rules, decision trees, nor other types of abstractions during learning. Instead, instance-based representations of concept descriptions consist solely of a set of instances. (We will assume in this paper that each instance is represented with a set of n attribute-value pairs.) IBL algorithms incrementally derive their concept descriptions from a sequence of training set 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 Instance-Based Learning Algorithms

More precisely, all IBL algorithms contain the following three components:

1. *Similarity Function:* Given two normalized instances, this yields their *similarity*, expressed numerically.
2. *Classification Function:* Given an instance i to be classified and its similarity with each of the saved instances, 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 possibly some relevant statistics concerning them).

Each instance is *normalized* to ensure that all attributes are assigned equal classification importance by the similarity function. While assuming that attributes have equal importance is not necessarily correct, it is a fair approach when given no prior knowledge of relative attribute saliencies.

All IBL algorithms in this paper define similarity as the negation of Euclidean distance between two (normalized) instances. Comparisons of different similarity functions is left for future research. All our IBL 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 one of two classification functions, namely the *nearest neighbor* or *k-nearest neighbor* algorithms. 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 algo-

Table 1: Names and characterizations of the eight IBL algorithms. Asterisks denote those algorithms for which noise-tolerant extensions are used in the paper (e.g., NTgrowth).

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*	–

rithms (Proximity and K-nn) save all training instances. However, since most real-world domains exhibit regularities that make them amenable to storage-reducing algorithms, the remaining IBL algorithms save only those instances that are misclassified during training (which are assumed to contain additional concept boundary information). Storage-reducing IBL algorithms in turn are partitioned by how they update memory when the current training instance is correctly classified. *Instance-filtering* IBL algorithms discard correctly classified instances while *instance-averaging* IBL algorithms replace the instance which performed the classification with an average of it and the instance being classified. The Growth algorithm (Table 2) is an example instance-filtering algorithm. The Disjunctive Spanning algorithm (Bradshaw, 1987) is an otherwise identical instance-averaging algorithm. We will also be experimenting with the same instance-filtering variant of the K-nn algorithm and noise-tolerant versions for each of the three storage-reducing algorithms (signified with the prefix **NT** in front of each algorithm name).

Our choice for experimenting with this set of eight algorithms was motivated by several factors. Variants of the Growth algorithm have recorded excellent classification accuracies and small storage requirements in several task domains (Hart, 1967; Kurtzberg, 1987; Kibler & Aha, 1987). Similarly, Bradshaw (1987) reported impressive results concerning an application of the Disjunctive Spanning algorithm to a speech recognition problem. However, our experiments indicate that both algorithms are sensitive to noise. Our interest is in extending the 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 & Hart, 1973). However, the K-nn instance-filtering algorithm (K-nn Growth) is also sensitive to noise and should benefit from the noise-tolerant extension. Thus our present study also includes three IBL algorithms that use the k-nearest neighbor classification function (K-nn, K-nn Growth, and NT K-nn Growth).

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
∀ subsequent training instances $t$ in training set $T$:
    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 $C$
```

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 training set $T$:
    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
```

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). The noise-tolerant algorithms differ from their respective storage-reducing algorithms in three respects.

1. First, they maintain classification records for all saved instances (i.e., number of correct and incorrect classifications of subsequent training instances).
2. Second, only those saved instances with *significantly* good classification records are acceptable for use in subsequent classification tasks.
3. Third, 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]

The noise-tolerant extensions employ a significance test for determining 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 their accuracy is statistically significantly less. Confidence intervals are constructed around both the instances' current accuracy and their class' current frequency. If the accuracy interval's lowest (highest) value is greater than (less than) the class frequency interval's greatest (lowest) value, then the instance is accepted (dropped).[2]

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 value for dropping since we would like to drop those instances with even moderately poor classification accuracies.

[1]During the initial stages of training, none of the saved instances are acceptable. In order to more closely mimic the behavior of the algorithms when at least one instance is acceptable, only a subset of the saved instances' classification records are updated. In these cases, the most similar r instances' classification records are updated, where r is a randomly selected integer between 1 and the number of instances saved.

[2]The confidence interval is constructed from formula 5.5-4 in (Hogg & Tanis, 1983, pp. 296).

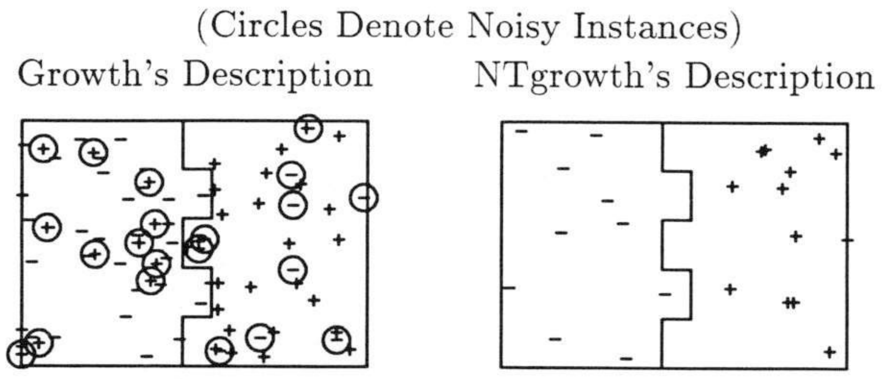

Figure 1: Growth and NTgrowth produced these concept descriptions when applied to 250 training instances with 10% noise. (NTgrowth accepted no noisy instances!) The instance space has 2 numeric attributes. The boundary between the positive and negative disjuncts is the rectilinear line shown in the instance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

4 Advantages of the Noise-Tolerant Extension

We applied the Growth and NTgrowth algorithms to a training set containing 250 randomly drawn instances from a 2-dimensional instance space containing two concepts. Figure 1 reveals which instances were saved by each algorithm when each of the training instances' class was mislabeled with a probability of 10%. In this trial, Growth saved 74 instances, 20 of which were mislabeled. An inspection of the saved noisy instances showed that they invariably recorded poor classification accuracies on subsequent training instances. Since the NTgrowth algorithm 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 reveals that NTgrowth successfully prevented noisy instances from entering the concept description. An inspection confirmed that all the instances accepted by NTgrowth had good classification accuracies (in this case, at least 70%). While NTgrowth's performance in this example might seem better than expected, it was actually typical. Averaged over 50 trials, 28.3% of Growth's saved instances were noisy. However, only a mere 0.5% of NTgrowth's accepted instances were likewise mislabeled.

We conducted a set of experiments with the Proximity, Growth, and NTgrowth algorithms on this same instance space. A summary of the results (averaged over 50 trials per noise setting) is displayed in Figure 2. The purpose was to discover how these algorithms' performances and concept descriptions degraded 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

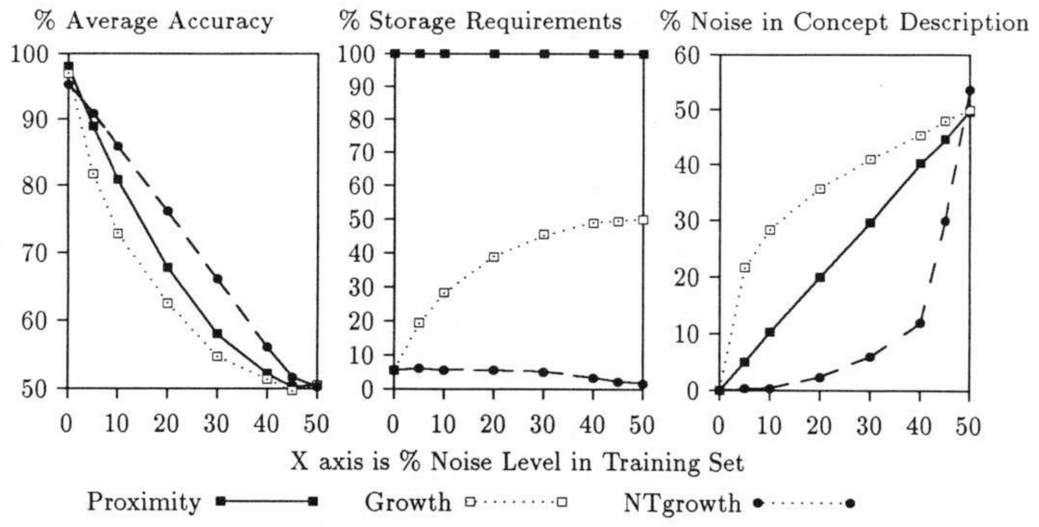

Figure 2: NTgrowth is a better noise filter than Growth.

well with 0% noise, NTgrowth's accuracy degraded linearly with the noise level while the other algorithms' accuracies degraded more quickly.[3]

2. *Storage Requirements*: The Proximity algorithm saved all training instances. As expected, the Growth algorithm's storage requirements were much lower, asymptoting towards 50%. However, NTgrowth's was 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 small numbers 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 descriptions.

5 Experiments and Results

The NTgrowth algorithm'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 in order to see if the benefits seen here pay off in practical applications. For comparison purposes, C4, a descendant of ID3 (Quinlan, 1986), was also applied to the databases. A summary of the database characteristics is given in Table 4. The results (averaged over 50 trials) are summarized in Table 5. (We have also included, for comparison purposes, the benchmark algorithm "Frequency" that always guesses the class with the highest frequency in the database. This algorithm also provides a comparative measure of the domain's difficulty.) In each case, the instances chosen

[3]We found that NTgrowth's parameters could be tailored to yield significantly slower degradations in accuracy.

Table 4: Database characteristics.

Database Name	Database #	Training Size	Test Size	# Attributes	# 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
Primary Tumor	6	250	89	17	22

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

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
DS	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 DS	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/–

for the training and test sets were randomly selected from the databases. Furthermore, we ensured that that training and test sets were disjoint.

The LED Display and Waveform domains (Breiman, Friedman, Olshen, & Stone, 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 a noise factor that is added to the original value). 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 are known to 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 their ancestor algorithms.

The voting domain contains only small amounts of noise. Therefore the payoff of the noise-tolerant algorithms was smaller than in more noisy domains. Finally, while the NTgrowth and NT Disjunctive Spanning algorithms performed well on the primary tumor domain, there were not enough training instances to allow the NT K-nn Growth algorithm to perform well.

In summary, the noise-tolerant extensions of 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 result indicates that these two noise-tolerant IBL algorithms should perform well in a large number of database applications.

6 Advantages and Limitations of IBL Algorithms

We believe that IBL and decision tree (or rule) learning algorithms are both highly similar and complementary. We showed (Kibler & Aha, 1988) that the Growth algorithm can learn any concept whose disjuncts have shapes in instance space that are piecewise approximable. We believe that this is the same general class of concepts that decision tree algorithms can learn. The two types of algorithms can also be extended to support probability-based classifications, polythetic prediction tasks (i.e., predict all attribute values rather than only the class), and multiclass learning tasks (in which classes are not necessarily disjoint). However, each method has its own advantages.

First, IBL algorithms are a more natural choice for tolerating concept drift. Since they retain and attend to only a small subset of the training examples, they can detect shifts in a concept's boundary (definition) more quickly than can those decision tree algorithms which save all the instances (e.g., ID5). Second, IBL algorithms are more cost-effective incremental methods. While they require $O(|I|^2 \times |A|)$ attribute examinations for a training set with $|I|$ instances and $|A|$ attributes per instance, only $O(|I| \times |A|)$ examinations are required to update the concept description for a single instance. Furthermore, since they greatly reduce storage requirements, their actual number of attribute examinations is much lower. In comparison, C4 and ID5 (Utgoff, 1988) require at most $O(|I|^2 \times |A|^2)$ and $O(|I| \times |A|^2)$ attribute examinations respectively to update the decision tree for a single instance. Finally, IBL algorithms are not constrained to forming hyper-rectangular partitions of instance space (as are several rule and decision tree algorithms).

In terms of classification accuracy, C4 outperformed NTgrowth in only two of the six experiments. However, the noise-tolerant IBL algorithms have several limitations. We haven't yet applied them to non-numeric attributes, they are sensitive to irrelevant attributes, and they do not summarize their concept descriptions. We are currently working on solutions to these problems. This includes an amalgamation with the decision tree approach, which allows for the generation of concise concept description summaries.

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 complicated domains (some artificial and some real-world). These gains occurred because the noise-tolerant extensions decreased the number of noisy instances partaking 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 & Bratko, 1986) and the testing of the quality of CN2's complexes (Clark & Niblett, 1987), is based upon a simple significance test. This 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, higher tolerability of concept drift, and the ability to present concept descriptions concisely.

Acknowledgements

We would like to thank the following donators: M. Zwitter and M. Soklič (Institute of Oncology, Ljubljana, Yugoslavia) for the primary tumor database, Robert Detrano (V.A. Center, Long Beach, CA) for the Cleveland database, Andras Janosi (Hungarian Institute of Cardiology, Budapest, Hungary) for the Hungarian database, and Jeff Schlimmer (Carnegie-Mellon University, Pittsburgh, PA) for the database concerning the 1984 U.S. congressional voting records. These are among the 34 databases available from the U.C.I. Repository of Machine Learning Databases.

References

Bradshaw, G. (1987). Learning about speech sounds: The NEXUS project. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 1–11). Irvine, CA: Morgan Kaufmann.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Belmont, CA: Wadsworth International Group.

Clark, P., & Niblett, T. (1987). Induction in noisy domains. In *Progress in Machine Learning - Proceedings of the Second European Working Session on Learning* (pp. 11–30). Bled, Yugoslavia: Sigma Press.

Connell, M. E., & Utgoff, P. E. (1987). Learning to control a dynamic physical system. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 456–460). Seattle, WA: Morgan Kaufmann.

Detrano, R., M.D. (1988). International application of a new probability algorithm for the diagnosis of coronary artery disease. Unpublished Manuscript.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* Menlo Park, CA: John Wiley & Sons.

Hart, P. E. (1967). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory, 13*, 515–516.

Hogg, R. V., & Tanis, E. A. (1983). *Probability and statistical inference.* New York, NY: Macmillan Publishing Co., Inc.

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

Kibler, D., & Aha, D. W. (1987). Learning representative exemplars of concepts: An initial case study. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 24–30). Irvine, CA: Morgan Kaufmann.

Kibler, D., & Aha, D. W. (1988). Comparing instance-averaging with instance-filtering learning algorithms. In *Proceedings of the Third European Working Session on Learning* (pp. 63–80). Glasgow, Scotland: Pitman Publishing.

Kibler, D., & Aha, D. W. (in press). Instance-based prediction of real-valued attributes. To be published in *Computational Intelligence.*

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

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

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

Stanfill, C. (1987). Memory-based reasoning applied to English pronunciation. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 577–581). Seattle, WA: Morgan Kaufmann.

Utgoff, P. (1988). ID5: An Incremental ID3. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 107–120). Ann Arbor, MI: Morgan Kaufmann.