Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms[*]

David B. Skalak
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
skalak@cs.umass.edu

February 6, 1994

Abstract

With the goal of reducing computational costs without sacrificing accuracy, we describe two algorithms to find sets of prototypes for nearest neighbor classification. Here, the term "prototypes" refers to the reference instances used in a nearest neighbor computation — the instances with respect to which similarity is assessed in order to assign a class to a new data item. Both algorithms rely on stochastic techniques to search the space of sets of prototypes and are simple to implement. The first is a Monte Carlo sampling algorithm; the second applies random mutation hill climbing. On four datasets we show that only three or four prototypes sufficed to give predictive accuracy equal or superior to a basic nearest neighbor algorithm whose run-time storage costs were approximately 10 to 200 times greater. Finally, we explain the performance of the sampling algorithm on these datasets in terms of a statistical measure of the extent of clustering displayed by the target classes.

keywords: instance-based learning

[*]This work was supported in part by the Air Force Office of Scientific Research under contract 90-0359.

1 Introduction

The classical nearest neighbor algorithm has a probably deserved reputation for being computational expensive. Run time costs for the basic algorithm are high: in order to determine the class of a new instance, its similarity to every instance in memory is assessed. Retaining all instances in primary memory entails also storage costs. Our goal in this paper is to demonstrate how two algorithms that rely on random sampling and local search can reduce the cost of nearest neighbor classification at run-time by reducing the number of prototypes retained. While the term "prototype" has many meanings, we use it to refer to a member of a set of actual instances whose similarity distance to a new data item is computed to determine which of those instances is its nearest neighbor. While the basic nearest neighbor algorithm treats all instances as prototypes, we show that it is possible to maintain or even improve nearest neighbor classification accuracy on out-of-sample data by selecting only a small handful of instances as prototypes. In order to further reduce costs in space and time, we also show how local search can be applied to limit the features used in the nearest neighbor computation.

Reducing the number of instances used for nearest neighbor retrieval has been researched by the pattern recognition and instance-based learning communities for some time, where it is sometimes called the "reference selection problem." Approaches to the problem have included storing misclassified instances (*e.g.*, the Condensed Nearest Neighbor algorithm [Hart, 1968], the Reduced Nearest Neighbor algorithm [Gates, 1972], IB2 [Aha, 1990]); storing typical instances [Zhang, 1992]; storing only training instances that have been correctly classified by other training instances [Wilson, 1972]; exploiting domain knowledge [Kurtzberg, 1987]; and combining these techniques [Voisin and Devijver, 1987]. Other systems deal with reference selection by storing averages or abstractions of instances. For discussions of these approaches, see [Aha, 1990].

Aha's research [1990] has shown that classification accuracy can be improved by limiting the number of prototypes and by weighting features. See for example the comparable or superior performance of Aha's instance-filtering algorithm IB3 over the baseline nearest neighbor algorithm IB1[Aha, 1990]. Aha's IB4 algorithm also improves performance by learning attribute relevance weights, resulting in a substantial increase in classification accuracy in some domains with irrelevant attributes. The research presented in this paper extends previous work on reference selection by showing that in some domains decreasing the number of prototypes can be pushed quite far indeed — that only several well-selected prototypes can give good classification accuracy. We further extend previous work by demonstrating that two algorithms that rely primarily on random techniques can perform the task of choosing a few salient prototypes.

The intuition that a small number of prototypes can achieve comparable or superior predictive accuracy is based on two speculations. (1) Noisy instances will not be used as prototypes; and (2) since each prototype may represent in a sense an additional degree of freedom for the classifier, limiting the number of prototypes may avoid overfitting the training data. This hypothesis may be tantamount to the assumption that a learning bias in favor of simpler models is appropriate for the data sets we use [Schaffer, 1993].

Many previous approaches to selecting prototypes are instance-filtering techniques, where each member of the dataset is examined in turn and some screen is used to sift the elements that should be retained in the emerging concept description. Since our goals are to decrease as far as possible the number of prototypes used and to select prototypes that *together* classify well, our approach starts from an *a priori* specification of prototype set size and tries to construct an accurate prototype set of that cardinality. This approach has the obvious advantage of forcing the examination of very small sets of prototypes. However, clamping the number of prototypes does limit the complexity of the problem we address. While we do not pursue in this paper the question of how many prototypes to include, we do appeal to a clustering index in Section 4 that might also be applied to this important problem.

Two algorithms are applied to select prototypes and features used in a nearest neighbor algorithm: (1) a Monte Carlo technique, which chooses the most accurate of a sample of random prototype sets; and (2) a random mutation hill climbing algorithm (*e.g.,* [Mitchell and Holland, 1993]), which searches for sets of prototypes with demonstrably good classification power. In previous work we described a genetic algorithm approach to finding prototypes [Skalak, 1993]. This paper follows the theme of using adaptive search techniques to find sets of prototypes, but uses somewhat less computationally intensive algorithms to locate them.

We next describe the underlying nearest neighbor algorithm and then introduce a Monte Carlo method (MC1) and two applications of random mutation hill climbing algorithms (RMHC-P and RMHC-PF1). Finally, we offer evidence that the degree of clustering of the dataset is a factor in determining how well a simple sampling algorithm will work.

1.1 The nearest neighbor algorithm

To determine the classification accuracy of a set of prototypes, a 1-nearest neighbor classification algorithm is used [Duda and Hart, 1973]. The similarity function used in this nearest neighbor computation is straightforward and relies on equally-weighted features.

In a pre-processing step, all feature values are linearly scaled from 0 to 100. Extreme feature values are squashed by giving a scaled value of 0 (100) to any raw value that is less (greater) than three standard deviations from the mean of that feature computed across all instances. Scaling in this way is designed to limit the effect of outlying values. In the pre-processing step, missing feature values are (naively) instantiated with the median value of that feature across all instances. To compute the similarity distance between two scaled instances, we use the Manhattan ("city block" or L_1) distance metric. The prototype with the smallest such similarity distance to a test instance is its 1-nearest neighbor. As usual, a instance is considered correctly classified by a prototype set if the instance's class equals the class of the prototype that is its 1-nearest neighbor taken from the prototype set. The ReMind case-based reasoning development shell has previously incorporated a similar algorithm for similarity assessment [Cognitive Systems, Inc., 1990].

1.2 Baseline storage requirements and classification accuracy

Four databases from the UCI machine learning repository were used: Iris, Cleveland Heart Disease (binary classes), Breast Cancer Ljubljana, and Soybean (small database). All but the Soybean database were chosen in part because results using various algorithms were compiled from the research literature by Holte [1993], providing convenient touchstones for comparison.

As a basis for comparison for the results we will present in this paper, we computed the baseline classification accuracy of the nearest neighbor algorithm, using all the training cases as prototypes and all the features, and five-fold cross validation. (Folds of equal size were used in the five-fold cross validation, necessitating that a residue of fewer than five instances may be left out of any fold.) The average accuracy over the five folds is given in Table 1. We also give for reference the best nearest neighbor algorithm results collected by Holte [1993].

Dataset	**Storage**	**NN-best reported**	**1-NN (CV)**
Iris	100% (150)	96.0%	92.0%
Cleveland	100% (303)	79.4%	76.3%
Breast Cancer	100% (286)	65.3%	67.4%
Soybean	100% (57)	(not reported)	100.0%

Table 1: Storage requirements (with number of instances in each data set), best classification accuracy for a nearest neighbor algorithm collected by Holte [1993], and classification accuracy computed using five-fold cross validation with the 1-nearest neighbor algorithm used in this paper.

However, direct comparison with the published results may be improvident in that different validation techniques, similarity metrics, and values of k of the k-nearest neighbor algorithms were used in the experiments[1]. In this research, we did not try to optimize k or experiment with different similarity metrics, and we applied five-fold cross validation.

For general reference, since C4.5 [Quinlan, 1993] is a benchmark learning algorithm, we also include in Table 2 classification accuracies on the four datasets using C4.5, taken both from the compilation by Holte and from our own five-fold cross validation runs using unpruned and pruned trees generated by C4.5.

Dataset	**C4.5** (reported)	**C4.5** (CV-unpruned)	**C4.5** (CV-pruned)
Iris	93.8%	93.3%	93.3%
Cleveland	73.6%	68.0%	71.6%
Breast Cancer	72.0%	67.8%	72.4%
Soybean	97.5%	95.6%	95.6%

Table 2: C4.5 classification accuracies collected by Holte [1993], and C4.5 classification accuracies computed using five-fold cross validation for unpruned and pruned trees.

The somewhat smaller C4.5 classification accuracies computed using five-fold cross validation may be explained by noting each fold applies only 80% of the instances in training, and therefore overestimates the true error of the algorithm.

[1]For example, on the Cleveland heart disease data, an average of several runs of Aha's IB3 algorithm yielded 80.1% correct using a five-fold cross validation regime. Personal communication.

2 The Algorithms

2.1 Monte Carlo (MC1)

As a general matter, Monte Carlo methods provide approximate solutions to mathematical and scientific problems through repeated stochastic trials. The results of independent trials are combined in some fashion, usually averaged [Sobol', 1974]. The method has been applied to many problems in such domains as numerical integration, statistical physics, quality control and particle physics.

The algorithm described in this section, called MC1, is a simple application of repeated sampling of the data set, where sampling is done with replacement. The algorithm is simple. It takes three input parameters: k (k-nearest neighbors), m (the number of prototypes, which is the sample size), and n (the number of samples taken). These parameters are fixed in advance. For all the experiments presented in this paper, $k = 1$ and $n = 100$. For all but the Soybean data ($m = 4$, since there are four classes) $m = 3$.

The Monte Carlo MC1 classification procedure can be summarized as follows.

1. Select n random samples, each sample with replacement, of m instances from the training set.
2. For each sample, compute its classification accuracy on the training set using a 1-nearest neighbor algorithm.
3. Select the sample(s) with the highest classification accuracy on the training set.
4. Classify the test set using as prototypes the samples with the highest classification accuracy on the training set. If more than one sample has the same highest classification accuracy on the training set, select a classification randomly from those given by the samples with the highest accuracy on the training set[2].

We assume that the data set has been divided for experimental purposes into a training set and test set. In a real application, all previously seen instances would be used as training instances. We report in Table 3 the storage requirements (the percentage of members of the training set that were retained) and the classification accuracy of the MC1 algorithm, applying five-fold cross validation.

These experiments show that MC1, a very simple approach based on random sampling, does quite well on these four data sets, with a large reduction in storage. An approach to prototype selection based on random mutation search is described next.

[2]We report average accuracy on the test set of the reference sets that display the highest predictive accuracy on the training set.

Dataset	Storage	MC1	1-NN
Iris	2.0%	93.7%	92.0%
Cleveland	1.0%	80.7%	76.3%
Breast Cancer	1.0%	71.9%	67.4%
Soybean	8.5%	99.1%	100.0%

Table 3: Storage requirements, average MC1 classification accuracy and average baseline 1-nearest neighbor classification accuracy using five-fold cross validation.

2.2 Random mutation hill climbing

2.2.1 The algorithm (RMHC)

Random mutation hill climbing is a local search method that has a stochastic component [Papadimitriou and Steiglitz, 1982]. The basic random mutation hill climbing algorithm (RMHC) is as described in [Mitchell and Holland, 1993]:

1. Choose a binary string at random. Call this string *best-evaluated.*
2. Mutate a bit chosen at random in *best-evaluated.*
3. Compute the fitness of the of the mutated string. If the fitness is greater[3] than the fitness of *best-evaluated*, then set *best-evaluated* to the mutated string.
4. If the maximum number of iterations have been performed return *best-evaluated*; otherwise, go to Step 2.

The general approach is to use a bit string to represent a set of prototypes, and in some experiments, a collection of features. The intuitive search mechanism is that the mutation of the bit vector changes the selection of instances in the prototype set or toggles the inclusion or exclusion of a feature from the nearest neighbor computation. The fitness function used for all the RMHC experiments is the predictive accuracy of a set of prototypes (and features) using the 1-nearest neighbor classification algorithm described in Section 1.

2.2.2 RMHC to select prototype sets (RMHC-P)

For this experiment, the bit vector encodes a set of m prototypes in a straightforward way. Each prototype set is encoded as a binary string, which is conceptually divided into m substrings, one for each of the m prototypes, where each substring encodes an index into the training cases stored as an array. The length of the binary string encoding a prototype set is the number of bits required to represent the largest index of a case in the training set, multiplied by the number of prototypes, $\lceil log_2 r \rceil * m$ where r is the number of cases in the training set. In each experiment, the training set size was 80% of the case-base size.

[3]In the version of RMHC used in the experiments below, an equal fitness does not result in the replacement of best-evaluated with the mutated string.

The RMHC algorithm was applied to this representation, searching for a set of three prototypes with superior predictive power. The fitness function was the classification accuracy on the training set composed of the three prototypes used as prototypes. In the experiments reported here the algorithm was run for 100 mutations[4]. The results after only 100 mutations — a very small number for this type of approach — are presented for two reasons. Informally, we did not observe an increase in classification accuracy in many experiments by running the algorithm longer, although more experimentation is required to establish this result. Second, such a small amount of search supports a *sub rosa* hypothesis of this paper, that small sets of prototypes with good classification accuracy are denser in the space of sets of prototypes than generally believed, since so little search is required to locate them. Average classification accuracy and storage requirements for five-fold cross validation are given in Table 4.

Database	**Storage**	**RMHC-P**	**1-NN**
Iris	2.0%	94.0%	92.0%
Cleveland	1.0%	82.3%	76.3%
Breast Cancer	1.0%	68.4%	67.4%
Soybean	8.5%	100.0%	100.0%

Table 4: Storage requirements, average classification accuracy for the prototypes selected by RMHC-P, and average baseline 1-nearest neighbor classification accuracy.

The results show a comparable or improved classification accuracy[5] of the RMHC-P algorithm over using all the training instances as prototypes. In particular, the results on the Cleveland database appear to be better than previously published results with only a very small percentage of stored instances used.

2.2.3 Select prototypes and features simultaneously (RMHC-PF1)

Finally, experiments wer performed to determine if the RMHC algorithm could select features as well as prototypes. Further reduction in computational costs would result from a reduction in the number of features that have to be taken into consideration in the nearest neighbor similarity computation. We used RMHC to select prototypes and features simultaneously, using a representation that records both the prototypes and features in a single vector.

To use RMHC to select features, we used a simple characteristic function bit vector representation, one bit for each feature used in the instance representation. The i-th bit records whether to use the corresponding i-th feature from some fixed presentation of the features: 1 to include the feature in the similarity distance computation, 0 to exclude it. Thus, for the Cleveland database, there are 13 features, and a 13-bit vector the features. RMHC used a fitness function that classified the training set using the prototypes as prototypes and only taking into account the features that are specified by the bit vector.

[4]To accelerate the algorithm, the calculated fitness of each prototype set was cached by memoizing [Abelson *et al.*, 1985] the evaluation function so that the predictive accuracy of a set of prototypes (and features) need only be computed once for each fold of the cross validation. Nonetheless, retrievals of previously cached evaluations are counted in the number evaluations reported.

[5]Statistical significance of all results is summarized in Section 7.

The bit vectors used in the previous experiments for prototypes and the features bit vector were concatenated in this representation. At each iteration only one bit was mutated, and it was left to the random bit mutation procedure whether that bit fell within the "prototypes sub-vector" or the "features sub-vector". We did not attempt to alter the bias stemming from the different relative lengths of the prototype set vectors and feature vectors, so that for this experiment it was more likely that a mutation changed the set of prototypes than the feature set.

For consistency of experimental presentation, the algorithm was again run, starting with a random set of features and prototypes, for 100 evaluations only, notwithstanding the increased size of the search space. Classification accuracy and storage requirements for the five-fold cross validation and three prototypes are given in Table 5.

Database	**Storage**	**RMHC-PF1**	**1-NN**
Iris	1.2%	94.7%	92.0%
Cleveland	0.6%	81.0%	76.3%
Breast Cancer	0.5%	72.3%	67.4%
Soybean	3.8%	97.8%	100.0%

Table 5: Storage requirements and average classification accuracy and for the selection of prototypes and features by RMHC-PF1, with average 1-nearest neighbor baseline classification accuracy.

In general, about half of the total number of features were used (Table 6), cutting run-time storage costs in half. For example, an average across the five folds of 2.4 features out of 4 were used for Iris classification. Petal-width and petal-length appear to be useful predictive features (as is known), since they were selected as features in five and four partitions, respectively. Sepal-length is apparently not useful, since it was selected for no partition. An average of 7.6 features of 13 were used for classification in the Cleveland data set. The ca feature was used in all five partitions; cp, exang, and thal were applied in four; restecg in none.

Database	**No. Total Features**	**Average No. Selected Features**
Iris	4	2.4
Cleveland	13	7.6
Breast Cancer	9	4.8
Soybean	36	16.4

Table 6: Total number of features and average number of selected features to select prototypes and features by RMHC-PF1.

3 Discussion

Table 7 summarizes the mean predictive accuracy of the preceding experiments.

Except for the small Soybean dataset, the storage requirements for all of the algorithms were on the order of 1% of the training instances, except for the baseline nearest neighbor

Database	1-NN	MC1	RMHC-P	RMHC-PF1
Iris	92.7 (100.0)	93.7 (2.0)	94.0 (2.0)	94.7 (1.2)
Cleveland	74.0 (100.0)	80.7†(1.0)	82.3* (1.0)	81.0* (0.6)
Breast Cancer	65.5 (100.0)	71.9†(1.0)	68.4 (1.0)	72.3* (0.5)
Soybean	100.0 (100.0)	99.1 (8.5)	100.0 (8.5)	97.8 (3.8)

Table 7: Summary of average classification accuracy (%) from five-fold cross validation for the experiments presented in this paper to select prototoypes and features. Storage requirements, in percentage of the training set are given in parentheses. The symbol "†" represents statistically significant improvement over the baseline 1-nearest neighbor algorithm (1-NN) at the 0.1 confidence level; "*", significance at the 0.05 confidence level. A two-sample t-test for statistical significance assuming equal population variances was used.

algorithm, which used 100% of the training examples. The general lesson is that a reduction in storage costs of one or two orders of magnitude from a standard nearest neighbor algorithm that uses all instances can be achieved on some of these data sets together with a statistically significant increase in computational accuracy.

While some of the nominally better results may not be statistically significant[6] , these experiments at least show that using three or four prototypes and possibly a proper subset of the features performs statistically as well as using the entire training set and all the features. Thus the accuracies of the algorithms presented in this paper are statistically comparable to a standard nearest neighbor approach with much smaller computational expense.

To determine the practicality of using random algorithms in "real-world" situations, the worst-case behavior of these algorithms should be investigated, particularly of the Monte Carlo algorithm, which displayed high variance in some tests. The results presented here represent average predictive accuracies on out-of-sample data. However, given the random basis for these algorithms, the probability of catastrophic failures of these methods should be determined. One direction for future research is to characterize the databases for which these very simple Monte Carlo or RMHC approaches will work. (For research that attempts to characterize the qualities of different data sets, see [Quinlan, 1993, Aha, 1992, Rendell and Cho, 1990] We make some initial steps in this direction in the following section.

4 When will Monte Carlo sampling work?

It is clear that such naive sampling techniques will not always work, although their limits need to be determined experimentally and theoretically. (One avenue of research is to determine whether random sampling of sets of prototypes can itself provide a measure of the predictive complexity of a database.) The success of a sampling algorithm appears to depend partly on the distribution of instances in the dataset and the geometric structure of the concepts to be learned.

One possible explanation for the successful results from sampling presented in this paper is

[6]A test for the analysis of variance (single factor) for each dataset reveals that we cannot reject (at the 0.05 confidence level) the null hypothesis that all of the predictive accuracy means are equal.

that the data sets used exhibit well-defined, widely spaced classes[7] in feature space, classes that exhibit a high degree of "internal coherence" and "external isolation." The intuition is that such an ideal separation of classes moots the selection of a prototype, since any instance in an isolated class may give perfect accuracy via a nearest neighbor algorithm.

Characterizing clusters and determining the "optimal" number of clusters in a data set are classical problems, and many indicators have been proposed, usually under the heading *stopping rules*, used to determine where to terminate a hierarchical clustering algorithm [Milligan and Cooper, 1985]. In a recent empirical examination of stopping rules, the Calinski-Harabasz Index (sometimes, the "Index") was the best performing of 30 procedures [Milligan and Cooper, 1985, Calinski and Harabasz, 1974]. The Index is defined as $[traceB/(k-1)]/[traceW/(n-k)]$ where n is the total number of data instances and k is the number of clusters in a possible clustering. B is the between cluster sum of squares and cross product matrices, and W is the pooled within cluster sum of squares and cross product matrices from multivariate statistics.

We have applied this index to determine how well the classes — regarded as clusters — are separated within each dataset. We performed a set of experiments to determine the effect of class isolation and cohesion, measured by the Calinski-Harabasz Index, on the performance of the MC1 Monte Carlo sampling algorithm. Our hypothesis was that as the Calinski-Harabasz Index increased, which entails greater class cohesion and external isolation, the performance of MC1 would also increase.

For each of the four datasets, we performed a 10-fold cross validation. The MC1 algorithm was run on each of the resulting 10 training sets and its classification accuracy was determined on the corresponding test set. Since classification accuracy is computed on the test set, we compute the Calinski-Harabasz Index of each pair of classes appearing in each of the 10 test sets. We report the results in Figures 1, 2 and 3, where we show the relationship between the minimum Calinski-Harabasz Index (taken over all the pairs of classes in a data set) and the classification accuracy on out-of-sample test sets[8]. The minimum Index was used as the independent variable under the hypothesis that the worst separation between a pair of classes would dominate the classification accuracy. We would have preferred to compare the Index of each data set and attempted to show how the classification accuracy varies with the Calinski-Harabasz Index, but it is unclear to what extent Index values from distinct data sets are comparable.

The results show a clear tendency for MC1 to perform better when the cluster index is higher (good class separation) and worse when the cluster index is lower (poor separation). While the results are not conclusive, they do present a basis for additional experiments to determine the range of applicability of Monte Carlo approaches.

5 Related research

In a recent article Holte [1993] demonstrated that a very simple type of classification rule is sufficient to perform quite good classification on a number of commonly used databases.

[7]We thank Paul Utgoff for this suggestion.

[8]Due to the uniformly high performance on the Soybean data, we omit the analysis of that dataset.

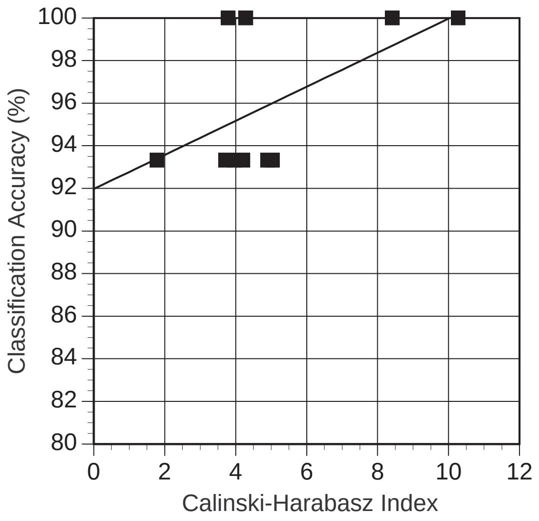

Figure 1: Classification accuracy vs. Calinski-Harabasz Index on Iris data

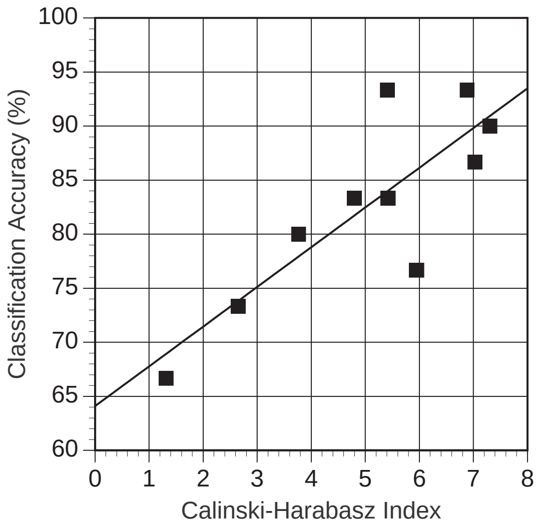

Figure 2: Classification accuracy vs. Calinski-Harabasz Index on Cleveland Heart Disease data

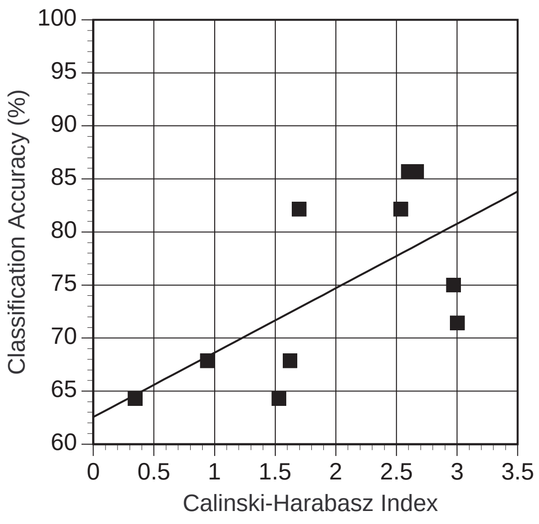

Figure 3: Classification accuracy vs. Calinski-Harabasz Index on Breast Cancer data

Our results complement that work by showing how weak methods dependent on random sampling search techniques work very well on two such databases. While Holte used decision trees for concept representation and applied several simple inductive learning algorithms, we instead use sets of prototypical instances as partial concept descriptions for a simple nearest neighbor classification algorithm.

Protos [Bareiss, 1989] is a good example of a case-based reasoning system that relies on case prototypes for classification. An exemplar that is successfully matched to a problem has its prototypicality rating increased. The prototypicality rating is used to determine the order in which exemplars are selected for further, knowledge-based pattern matching. Protos is an intelligent classification assistant and the prototypicality rating may be increased based in part on the actions of the supervising teacher. The ReMind case-based reasoning development shell [Cognitive Systems, Inc., 1990] also incorporates a facility for the user to create prototypes to further index a case base.

Genetic algorithm classification systems have been created by Vafaie and De Jong [1992] and by Kelly and Davis [1991] to select features by learning real-valued weights for features in a data set and by DeJong and Spears [1991] to learn conceptual classification rules. Tan and Schlimmer [1990] have shown how features for nearest-neighbor retrieval may be selected where the determination of feature values has a computational price.

6 Conclusion

We have found these results surprising. Using very simple stochastic algorithms these experiments show that significant reductions in storage of cases and features can be achieved on the datasets examined without decreasing nearest neighbor classification accuracy, and in some instances actually improving it.

7 Acknowledgments

This research was supported in part by the Air Force Office of Scientific Research, contract 90-0359. For assistance of various sorts I thank E. Rissland, P. Utgoff, D. Aha, C. Cardie, C. Brodley, T. Fawcett, J. Callan, R. Holte, M.T. Friedman, E. Riloff, S. Anderson and an anonymous reviewer from the Tenth International Conference on Machine Learning, 1993. The principal investigator for the compilation of the Cleveland heart disease data was Robert Detrano, M.D., Ph.D., V.A. Medical Center, Long Beach and Cleveland Clinic Foundation. The breast cancer data was originally obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing that data.

References

[Abelson *et al.*, 1985] Abelson, H.; Sussman, G.; and Sussman, J. 1985. *Structure and Interpretation of Computer Programs.* MIT Press, Cambridge, MA.

[Aha, 1990] Aha, D. W. 1990. *A Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Evaluations.* Ph.D. Dissertation, Dept. of Information and Computer Science, University of California, Irvine.

[Aha, 1992] Aha, D. W. 1992. Generalizing from Case Studies: A Case Study. In *Proceedings of the Ninth International Conference on Machine Learning*, Aberdeen, Scotland. Morgan Kaufmann. 1–10.

[Bareiss, 1989] Bareiss, E. R. 1989. *Exemplar-Based Knowledge Acquisition.* Academic Press, Boston, MA.

[Calinski and Harabasz, 1974] Calinski, T. and Harabasz, J. 1974. A Dendrite Method for Cluster Analysis. *Communications in Statistics* 3:1–27.

[Cognitive Systems, Inc., 1990] Cognitive Systems, Inc., 1990. Case-Based Retrieval Shell, User's Manual V. 3.17. Cognitive Systems, Inc.

[DeJong and Spears, 1991] DeJong, K. A. and Spears, W. M. 1991. Learning Concept Classification Rules Using Genetic Algorithms. In *12th International Joint Conference on Artificial Intelligence*, Sydney, Australia. International Joint Conferences on Artificial Intelligence. 651–656.

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

[Gates, 1972] Gates, G. W. 1972. The Reduced Nearest Neighbor Rule. *IEEE Transactions on Information Theory* 431–433.

[Hart, 1968] Hart, P. E. 1968. The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory (Corresp.)* IT-14:515–516.

[Holte, 1993] Holte, R. C. 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning* 11:63–90.

[Kelly and Davis, 1991] Kelly, J. D. J. and Davis, L. 1991. Hybridizing the Genetic Algorithm and the K Nearest Neighbors Classification Algorithm. In *Proceedings, Fourth International Conference on Genetic Algorithms*, San Diego, CA. Morgan Kaufmann, San Mateo, CA. 377–383.

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

[Milligan and Cooper, 1985] Milligan, G. W. and Cooper, M. C. 1985. An Examination of Procedures for Determining the Number of Clusters in a Data set. *Psychometrika* 50:159–179.

[Mitchell and Holland, 1993] Mitchell, M. and Holland, J. H. 1993. When Will a Genetic Algorithm Outperform Hill-Climbing? Technical report, Santa Fe Institute.

[Papadimitriou and Steiglitz, 1982] Papadimitriou, C.H. and Steiglitz, K. 1982. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, NJ.

[Quinlan, 1993] Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.

[Rendell and Cho, 1990] Rendell, L. and Cho, H. 1990. Empirical Learning as a Function of Concept Character. *Machine Learning* 5:267–298.

[Schaffer, 1993] Schaffer, C. 1993. Overfitting Avoidance as Bias. *Machine Learning* 10:153–178.

[Skalak, 1993] Skalak, D. B. 1993. Using a Genetic Algorithm to Learn Prototypes for Case Retrieval and Classification. In *Proceedings of the AAAI-93 Case-Based Reasoning Workshop (Technical Report WS-93-01)*, Washington, D.C. American Association for Artificial Intelligence, Menlo Park, CA.

[Sobol', 1974] Sobol', I. M. 1974. *The Monte Carlo Method.* The University of Chicago Press, Chicago, IL.

[Tan and Schlimmer, 1990] Tan, M. and Schlimmer, J. C. 1990. Two Case Studies in Cost-Sensitive Concept Acquisition. In *Proceedings, Eighth National Conference on Artificial Intelligence*, Boston, MA. AAAI Press, Menlo Park, CA. 854–860.

[Vafaie and DeJong, 1992] Vafaie, H. and DeJong, K. 1992. Genetic Algorithms as a Tool for Feature Selection in Machine Learning. In *Proceedings of the 1992 IEEE Int. Conf. on Tools with AI*, Arlington, VA (Nov. 1992). IEEE. 200–203.

[Voisin and Devijver, 1987] Voisin, J. and Devijver, P. A. 1987. An application of the Multiedit-Condensing technique to the reference selection problem in a print recognition system. *Pattern Recognition* 5:465–474.

[Wilson, 1972] Wilson, D. 1972. Asymptotic Properties of Nearest Neighbor Rules using Edited Data. *Institute of Electrical and Electronic Engineers Transactions on Systems, Man and Cybernetics* 2:408–421.

[Zhang, 1992] Zhang, J. 1992. Selecting Typical Instances in Instance-Based Learning. In *Proceedings of the Ninth International Machine Learning Workshop*, Aberdeen, Scotland. Morgan Kaufmann, San Mateo, CA. 470–479.