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

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. We briefly investigate how random mutation hill climbing may be applied to select features and prototypes simultaneously. 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.

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 also entails 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" [Dasarathy, 1991]. From one research perspective, this problem is a part of the general learning problem of determining the effect of the number and quality of training instances on predictive accuracy. 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 second 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 data set 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 data set is a factor in determining how well our 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. The ReMind case-based reasoning development shell has previously incorporated a similar data pre-processing method [Cognitive Systems, Inc., 1990]. 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, an 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.

1.2 Baseline storage requirements and classification accuracy

Four databases from the UCI machine learning repository [Murphy and Aha, 1994] 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 might be left out of every fold.) The average accuracy over the five folds is given in Table 1. For general reference, since C4.5 [Quinlan, 1993] is a benchmark learning algorithm, we also include in Table 1 classification accuracies on the four data sets from our own five-fold cross validation runs using pruned trees generated by C4.5 with its default option settings.

Table 1: Storage requirements (with number of instances in each data set) and classification accuracy computed using five-fold cross validation with the 1-nearest neighbor algorithm used in this paper and pruned trees generated by C4.5.

Data Set	Storage	1-NN	C4.5
Iris	100% (150)	93.3%	93.3%
Cleveland	100% (303)	74.3%	71.6%
Breast Cancer	100% (286)	65.6%	72.4%
Soybean	100% (47)	100.0%	95.6%

In general, direct comparison with published results may be improvident in that different validation techniques, similarity metrics, and values of k of the k-nearest neighbor algorithms almost surely will have been used. In this research, we did not try to optimize k or experiment with different similarity metrics.

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 $n = 100$. We choose $m = 3$ for all but the Soybean data set, where $m = 4$, since there are four classes.

The Monte Carlo MC1 classification procedure can be summarized as follows. We assume that the data set has been divided for experimental purposes into a training set and test set.

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 these best-performing samples[1].

In a real application, all previously seen instances would be used as training instances. We report in Table 2 the storage requirements (the percentage of members of the data 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. A discussion of the statistical significance of all results is postponed to Section 3. In Section 4 we try to characterize one of the

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

Table 2: Storage requirements, average MC1 classification accuracy and average baseline 1-nearest neighbor classification accuracy using five-fold cross validation.

Data Set	Storage	MC1	1-NN
Iris	2.0%	93.5%	93.3%
Cleveland	1.0%	80.7%	74.3%
Breast Cancer	1.0%	72.6%	65.6%
Soybean	8.5%	99.1%	100.0%

factors that will determine when random sampling will provide good accuracy. An approach to prototype selection based on random mutation search is described next.

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 by 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 strictly greater 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 on the training data 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 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 data set, multiplied by the number of prototypes, $\lceil log_2 r \rceil \cdot m$ where r is the number of cases in the data set. So, for example, the number of bits used to encode a set of three Iris prototypes was $\lceil log_2 150 \rceil \cdot 3 = 24$.

The RMHC algorithm was applied to this representation, searching for a set of three (four for the Soybean data set) prototypes with superior predictive power. The fitness function was the classification accuracy on the training set of a 1-nearest neighbor classifier that used each set of prototypes as reference instances. In the experiments reported here the algorithm was run for 100 mutations[2]. 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 might be expected for these data sets, since little search is required to locate them. Further, the accuracy of the final prototype set returned by the algorithm does appear to be better than the random prototype set used as the starting point. A random starting prototype set yielded an average accuracy (across the five partitions) on the test set of 68.0% for the Iris data set, 54.7% for the Cleveland data, 61.4% for the Breast Cancer data, and 66.7% for the Soybean data. Average classification accuracy for the final prototype set and storage requirements for five-fold cross validation are given in Table 3.

Table 3: Storage requirements, average classification accuracy for the prototypes selected by RMHC-P, and average baseline 1-nearest neighbor classification accuracy.

Database	Storage	RMHC-P	1-NN
Iris	2.0%	93.3%	93.3%
Cleveland	1.0%	82.3%	74.3%
Breast Cancer	1.0%	70.9%	65.6%
Soybean	8.5%	97.8%	100.0%

The results show a comparable or improved classification accuracy 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[3].

2.2.3 Select prototypes and features simultaneously (RMHC-PF1)

Finally, experiments were 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 considered in the nearest neighbor similarity computation. We used RMHC-PF1 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 ith bit records whether to use the corresponding ith 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 represents the features.

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 to specify whether that bit fell within the "prototypes sub-vector" or the "features sub-vector." We did not attempt to alter any bias stemming from the different relative lengths of the prototype set vectors and feature vectors. RMHC-PF1 used a fitness function that classified the training set using the encoded prototypes and only taking into account the features that are specified by the bit vector.

For consistency of experimental presentation, the algorithm was again run for 100 evaluations only, notwithstanding the increased size of the search space, starting with a random set of features and prototypes. The random starting prototype and features set yielded an average accuracy (across the five partitions) on the test set of 51.3% for the Iris data set, 61.6% for the Cleveland data, 55.4% for the Breast Cancer data, and 51.1% for the Soybean data. Classification accuracy and storage requirements for the five-fold cross validation are given in Table 4. If a data set has P instances, each containing F features, and the RMHC-PF1 algorithm yields p prototypes and f features, the storage requirements reported are pf/PF. We assume a uniform-cost, and not a log-cost, model of storage costs in reporting these figures.

Table 4: Storage requirements and average classification accuracy for the selection of prototypes and features by RMHC-PF1, with average 1-nearest neighbor baseline classification accuracy.

Database	Storage	RMHC-PF1	1-NN
Iris	1.2%	94.7%	93.3%
Cleveland	0.6%	80.7%	74.3%
Breast Cancer	0.6%	72.3%	65.6%
Soybean	3.9%	97.8%	100.0%

[2] 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.

[3] D. Aha reported an average of several runs of the IB3 algorithm yielded 80.1% correct on the Cleveland heart disease data using a five-fold cross validation regime. Personal communication.

In general, about half of the total number of features were used (Table 5), 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, since they were selected as features in five and four partitions, respectively. *Sepal-length* is apparently not useful, since it was not selected in any 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. Descriptions of these features may be found in [Murphy and Aha, 1994].

Table 5: Number of features in the original instance representation and average number features selected by RMHC-PF1.

Database	Total Features	Average Features
Iris	4	2.4
Cleveland	13	7.6
Breast Cancer	9	4.8
Soybean	36	16.4

3 Discussion

Table 6 summarizes the mean predictive accuracy of the preceding experiments.

Except for the small Soybean data set, the storage requirements for all of the algorithms were on the order of 1% of the training instances, except for the baseline nearest neighbor 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[4], 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 would be 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 take 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 data set and the geometric structure of the concepts to be learned.

In particular, one possible explanation for the successful results from sampling presented in this paper is that the data sets used exhibit well-defined, widely spaced classes[5] 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 an 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 $[trace\, B/(k-1)]/[trace\, W/(n-k)]$ where n is the total number of data instances and k is the number of clusters in a possible clustering[6]. 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 data set. 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

[4] In interpreting the significance results in the table, note that the t-test considers the mean and variance of the underlying cross validation data, but that only the mean percentage accuracy is reported here. In general, MC1 displayed higher variance than the other algorithms. A test for the analysis of variance (single factor) for each data set also reveals that we cannot reject at the 0.05 confidence level the null hypothesis that all of the predictive accuracy means are equal.

[5] We thank Paul Utgoff for this suggestion.

[6] The *trace* of a square matrix is the sum of its diagonal elements.

Table 6: Summary of average classification accuracy (% correct) from five-fold cross validation for the experiments presented in this paper to select prototypes and features. Storage requirements, in percentage of the data set are given in parentheses. The symbol "†" denotes 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.

Database	**1-NN**	**MC1**	**RMHC-P**	**RMHC-PF1**
Iris	93.3	93.5 (2.0)	93.3 (2.0)	94.7 (1.2)
Cleveland	74.3	80.7 (1.0)	82.3* (1.0)	80.7†(0.6)
Breast Cancer	65.6	72.6†(1.0)	70.9 (1.0)	72.3* (0.6)
Soybean	100.0	99.1 (8.5)	97.8 (8.5)	97.8 (3.9)

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 data sets, 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. From the standpoint of using the Index to predict the suitability of a sampling technique for a given data set, the Index value on the training set might better reflect the "true" clustering displayed by the entire data set. Experimental results that apply the Index to the training set have not been conclusive to date, however.

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 test set) and the classification accuracy on out-of-sample test sets[7]. 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. Also, Index values vary by orders of magnitude on the training and test sets, and the large apparent variation in Index values across partitions is small on a relative basis.

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). Tests for significance of the regression trendlines were confirmed by an ANOVA F-test, where all were found significant at the 0.05 confidence level (Iris: $F = 6.2$; Cleveland: $F = 17.6$; Breast Cancer: $F = 5.4$). While the results are not conclusive, they do present a basis for additional experiments to determine the range of applicability of Monte Carlo approaches.

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

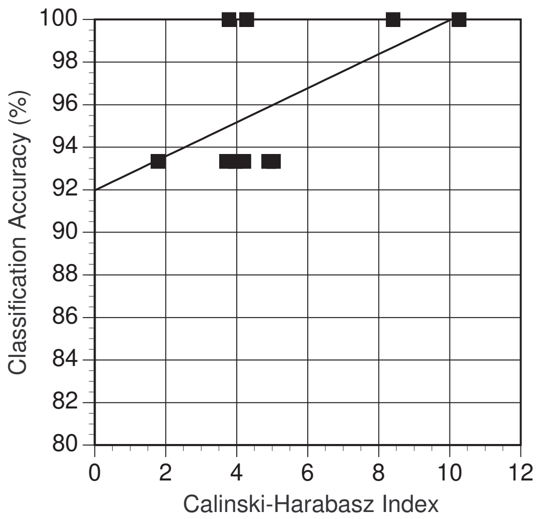

Figure 1: Classification accuracy vs. Calinski-Harabasz Index on Iris data

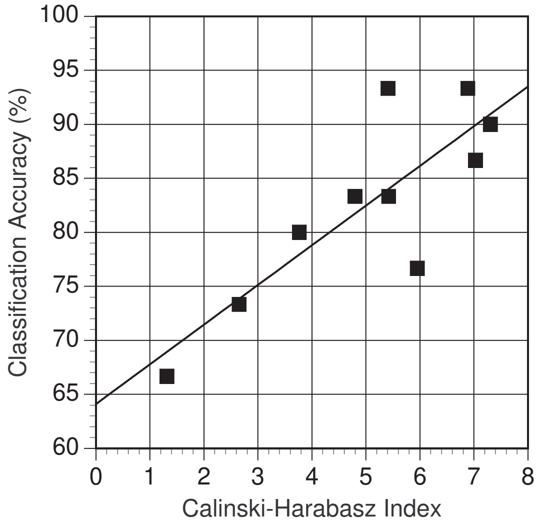

Figure 2: Classification accuracy vs. Calinski-Harabasz Index on Cleveland Heart Disease data

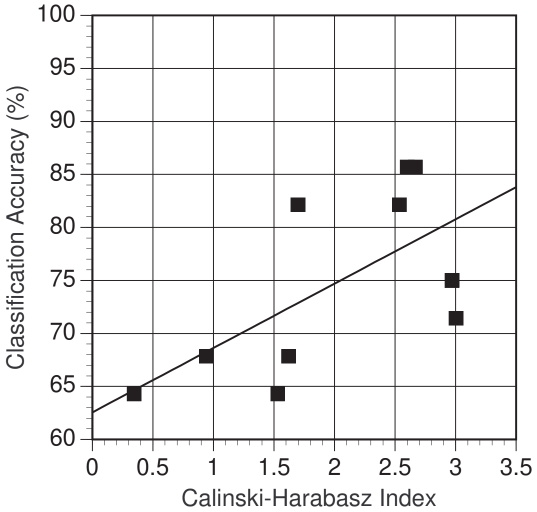

Figure 3: Classification accuracy vs. Calinski-Harabasz Index on Breast Cancer data

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. 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.

While this paper investigates the coupling between prototypes and features, the classical problem of feature subset selection has received much attention from machine learning researchers (*e.g.,* [Almuallim and Dietterich, 1991; Kira and Rendell, 1992; Moore and Lee, 1994; Langley and Sage, 1994]). John, Kohavi and Pfleger[1994] identify the need for "wrapper" models of selection, which incorporate the target inductive algorithm into the selection of features. The RMHC-PF1 procedure is an instance of a wrapper algorithm that wraps around the target nearest neighbor method, for example.

Caruana and Freitag[1994] compare five algorithms that perform hillclimbing in feature space where the target algorithm is C4.5, and the task is a difficult scheduling task. The authors also present an ingenious caching scheme appropriate to decision trees to speed up hillclimbing. (In contrast, the caching scheme used by the programs presented here was a simple memoizing technique that relied on exact match of function arguments.)

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 data sets examined without decreasing nearest neighbor classification accuracy, and in some instances actually improving it. This paper also has provided evidence for the hypothesis that the average accuracy of a simple method of finding prototypes by sampling increases with the internal coherence and external isolation of the classified data, as measured by a classical clustering index developed by Calinski and Harabasz.

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, three anonymous reviewers, 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 Ljubljana data were 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: Mathemat-*

ical, 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.

[Almuallim and Dietterich, 1991] Almuallim, H. and Dietterich, T.G. 1991. Learning with Many Irrelevant Features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Cambridge, MA. MIT Press. 547–552.

[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.

[Caruana and Freitag, 1994] Caruana, Rich and Freitag, Dayne 1994. Greedy Attribute Selection. In *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ. Morgan Kaufmann.

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

[Dasarathy, 1991] Dasarathy, Belur V. 1991. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, CA.

[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.

[John *et al.*, 1994] John, George; Kohavi, Ron; and Pfleger, Karl 1994. Irrelevant Features and the Subset Selection Problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ. Morgan Kaufmann.

[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.

[Kira and Rendell, 1992] Kira, Kenji and Rendell, Larry A. 1992. The Feature Selection Problem: Traditional Methods and a New Algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, Cambridge, MA. MIT Press. 129–134.

[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.

[Langley and Sage, 1994] Langley, P. and Sage, S. 1994. Oblivious Decision Trees and Abstract Cases. In *Proceedings of the AAAI-94 Case-Based Reasoning Workshop (to appear)*, Seattle, WA. American Association for Artificial Intelligence, Menlo Park, CA.

[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.

[Moore and Lee, 1994] Moore, Andrew W. and Lee, Mary S. 1994. Efficient Algorithms for Minimizing Cross Validation Error. In *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ. Morgan Kaufmann.

[Murphy and Aha, 1994] Murphy, P. M. and Aha, D. W. 1994. UCI repository of machine learning databases. For information contact ml-repository@ics.uci.edu.

[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.