Feature Selection for Case-Based Classification of Cloud Types: An Empirical Comparison

David W. Aha
Navy AI Center
Naval Research Laboratory
Washington, DC 20375
aha@aic.nrl.navy.mil

Richard L. Bankert
Marine Meteorology Division
Naval Research Laboratory
Monterey, CA 93943
bankert@nrlmry.navy.mil

Abstract

Accurate weather prediction is crucial for many activities, including Naval operations. Researchers within the meteorological division of the Naval Research Laboratory have developed and fielded several expert systems for problems such as fog and turbulence forecasting, and tropical storm movement. They are currently developing an automated system for satellite image interpretation, part of which involves cloud classification. Their cloud classification database contains 204 high-level features, but contains only a few thousand instances. The predictive accuracy of classifiers can be improved on this task by employing a feature selection algorithm. We explain why non-parametric case-based classifiers are excellent choices for use in feature selection algorithms. We then describe a set of such algorithms that use case-based classifiers, empirically compare them, and introduce novel extensions of backward sequential selection that allows it to scale to this task. Several of the approaches we tested located feature subsets that attain significantly higher accuracies than those found in previously published research, and some did so with fewer features.

Motivation

Accurate interpretation of maritime satellite imagery data is an important component of Navy weather forecasting, particularly in remote areas for which there are limited conventional observations. Decisions regarding tactical Naval operations depend on their accuracy. Unfortunately, frequent personnel rotations and lack of training time drastically reduce the level of shipboard image interpretation expertise. Therefore, the Naval Research Laboratory (NRL) in Monterey, CA is building an expert system named SIAMES (Satellite Image Analysis Meteorological Expert System) that aids in this task (Peak & Tag, 1992). Inputs to SIAMES, including cloud classification, are not yet fully automated; the user must manually input details on cloud types. Bankert (1994) addressed this problem. Given a database containing 204 continuous features describing ten classes of clouds, he used a standard feature selection technique and then applied a probabilistic neural network (PNN) (Specht, 1990; Welch et. al, 1992) to classify cloud types.[1] Current plans include using this network in SIAMES.

This paper extends the work reported by Bankert (1994). We argue for the use of case-based classifiers in domains with large numbers of features. We show they can locate smaller feature subsets leading to significantly higher accuracies on this task. We begin by describing the cloud classification task and Bankert's study. Next, we explain the benefits of using case-based classifiers in feature selection, and argue why such approaches should outperform the approach previously used. We then describe a set of feature selection algorithms that we tested on the cloud classification task. Finally, we report our results, and discuss them in the context of related work.

This paper has three primary contributions. First, we show further evidence that feature selection algorithms should use the classifier itself to evaluate feature subsets. Second, we show that a case-based algorithm is a particularly good classifier in this context. Finally, we introduce a method that allows the backward sequential selection algorithm to scale to large numbers of features.

Cloud Classification Task

Bankert (1994) describes the cloud classification task. The original data were supplied and expertly labeled by Dr. C. Wash and Dr. F. Williams of the Naval Postgraduate School. They were in the form of four advanced very high resolution radiometer (AVHRR) local area coverage (LAC) scenes with size 512×512 pixels. An additional 91 images were labeled independently by four NRL Monterey experts under the direction of Dr. C. Crosiar. The scenes were not confined to a specific location nor time of year. Sample areas of 16x16 pixels that were covered by at least 75% of

---

[1] PNN uses a standard three-layer network of nodes and a Bayesian classification strategy to select the class with the highest value of the a posteriori class probability density function.

a particular cloud type were expertly labeled for each image. A total of 1633 cases were extracted using this procedure, where each case was labeled according to one of ten cloud classes.

The features used to describe each case consist of spectral, textural, and physical measures for each sample area. Examples of each group respectively include such features as maximum pixel value, gray level entropy of the image, gray level distribution of adjacent pixels in the horizontal dimension, and cloud-top temperature. Latitude was included as a final feature. All features are defined over a continuous range of values.

The space of feature subsets (i.e., the power set of all features) for this task is $2^{204} = 6.4 \times 10^{60}$. Bankert (1994) used a standard feature selection approach from the pattern recognition literature. This reduced the number of features used from 204 to 15. He then applied PNN (Specht, 1990; Welch et al., 1992), which has a 10-fold cross validation (CV) accuracy of 75.3% on the initial feature set. Its accuracy improves to 78.9% when using the selected 15 features.

Feature Selection

The objective of feature selection is to *reduce the number of features used to characterize a dataset so as to improve an algorithm's performance on a given task.* The task studied in this paper is classification. Our objectives are to maximize classification accuracy and minimize the number of features used to define the cases. Feature selection algorithms input a set of features and yield a subset of them.

Several knowledge-intensive CBR algorithms have been used to perform feature selection (e.g., Ashley & Rissland, 1988). However, domain specific knowledge is not yet available for the cloud classification task. This prevents us from using explanation-based approaches for indexing and retrieving appropriate features (e.g., Cain, Pazzani, & Silverstein, 1991; Ram, 1993). Furthermore, the same set of features are used to describe each case in this case base, their values have been pre-computed, and no further inferencing is required to access these values. Therefore, we do not address the cost of evaluating features (Owens, 1993). Thus, this study was restricted to using knowledge-poor feature selection approaches (e.g., Devijver & Kittler, 1982), although we plan to work with knowledge-based techniques for feature extraction in the future.

Feature selection algorithms have three components:

1. **Search algorithm** This searches the space of feature subsets, which has size 2^d where d is the number of features. Points in this space are called *states*.

2. **Evaluation function** This inputs a state and outputs a numeric evaluation. The search algorithm's goal is to maximize this function.

3. **Classifier** This is the target algorithm that uses the final subset of features (i.e., those found by the search algorithm to have the highest evaluation).

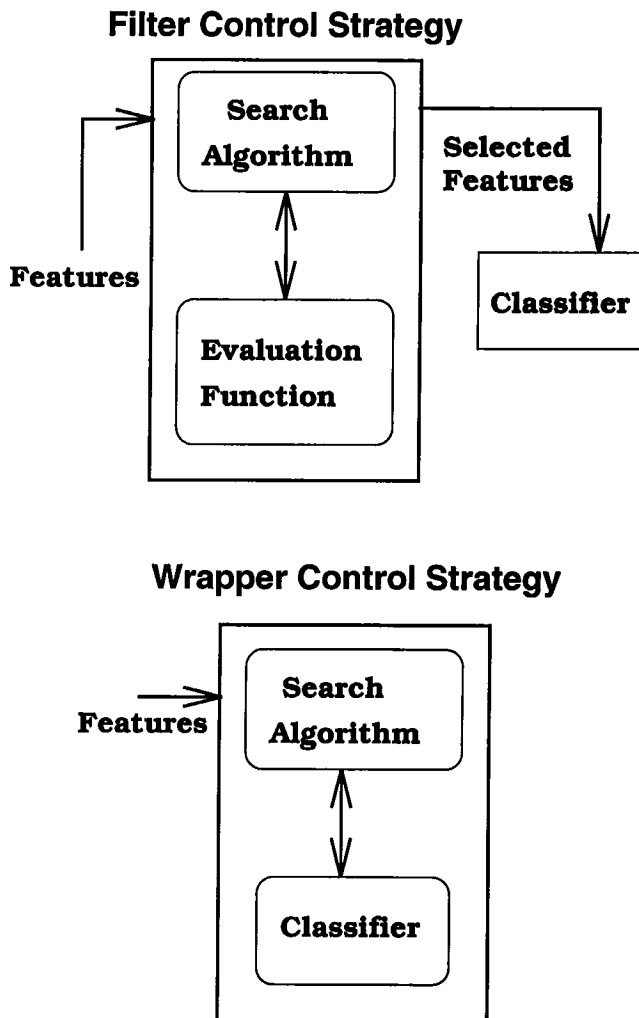## Filter Control Strategy

Wrapper Control Strategy

Figure 1: Common Control Strategies for Feature Selection Approaches

Typically, these components interact in one of the two ways described in Figure 1. Either the search and evaluation algorithms locate the set of features before the classifier is consulted, or the classifier is itself the evaluation function. We adopt the terms *filter* and *wrapper* control strategies, respectively, from John, Kohavi, and Pfleger (1994), who argued that the wrapper strategy is superior because it avoids the problem of using an evaluation function whose bias differs from the classifier. Our empirical results strongly support this claim.

Since many feature subsets are evaluated during the feature selection search, classifiers that require manual tuning of their parameters (e.g., PNN) cannot be used as the evaluation function. Since Bankert (1994) chose PNN as his classifier, he used a different function to evaluate states. We hypothesize that better re-

107

sults can be obtained by using a wrapper model. Non-parametric case-based classifiers are excellent choices for feature selection classifiers because they often obtain good accuracies and can also be used as evaluation functions. Thus, we used IB1 (Aha, Kibler, & Albert, 1991) in our studies for these two purposes; it is an implementation of the nearest neighbor classifier.

Doak (1992) identifies three categories of search algorithms: exponential, sequential, and randomized. Exponential algorithms have exponential complexity in the number of features. Sequential search algorithms have polynomial complexity; they add or subtract features and use a hill-climbing search strategy. Randomized algorithms include genetic and simulated annealing search methods. Sequential search algorithms seem most appropriate for our task because exponential search algorithms (e.g., FOCUS (Almuallim & Dietterich, 1991) are prohibitively expensive and randomized algorithms, unless biased as in (Skalak, 1994), tend to yield larger subsets of features than do sequential strategies (Doak, 1992). Sequential algorithms performed comparatively well in Doak's study. Thus, we chose to use them in our study. We have not yet investigated strategies that combine algorithms from different categories (e.g., Doak, 1992; Caruana & Freitag, 1994).

The most common sequential search algorithms for feature selection are variants of *forward sequential selection* (FSS) and *backward sequential selection* (BSS). FSS begins with zero attributes, evaluates all feature subsets with exactly one feature, and selects the one with the best performance. It then adds to this subset the feature that yields the best performance for subsets of the next larger size. This cycle repeats until no improvement is obtained by extending the current subset. BSS instead begins with all features and repeatedly removes the feature that, so removed, the maximal performance increase results. Doak (1992) reported that variants of BSS outperformed variants of FSS, probably because BSS evaluates the contribution of a feature when all others are available, whereas FSS suffer when features individually don't contribute to performance but do so when jointly considered with other features (i.e., they interact).

BSS's computational expense prevents it from being easily applied to our task. It would require approximately 21,000 applications before it could find a subset of size found using Bankert's approach. Each application would involve an application of IB1, whose costs are $O(F \times I^2)$, where I is the number of instances and F is the number of features. Therefore, we modified BSS (see below). Because of this modification, it is not obvious whether it would perform as well as FSS in our study.

A Framework of Algorithms

In our experiments, we focussed on the four categories of feature selection algorithms shown in Table 1. The

Table 1: Four Types of Feature Selection Algorithms

Control Strategy	Search Algorithm
Filter	FSS
Filter	BSS
Wrapper	FSS
Wrapper	BSS

rest of this section describes the algorithms we applied.

Filter Control with FSS

Bankert's (1994) approach is a member of the first of these four categories. He used FSS with the Bhattacharya class separability index as the evaluation function - it measures the degree to which classes are separated and internally cohere. Many such indices have been proposed and evaluated (e.g., Devijver & Kittler, 1982). They are usually less computationally expensive than the classifier and are nonparametric, which allows them to be used without requiring manual parameter tuning. However, if their bias does not match the bias of the classifier, then they can lead to suboptimal performance.

Filter Control with BSS

The second category combines a filter control strategy with BSS search. Since it is difficult to apply BSS to a task with 204 features due to its computational complexity, we introduce BEAM, a novel extension of BSS that allows it to work with larger numbers of features.

BEAM is somewhat similar to Doak's (1992) random generation plus sequential selection (RGSS) algorithm. They both begin with a feature subset found through random selection of features, and they both use a form of sequential feature selection after locating this initial subset. After choosing the initial set of features, RGSS uses FSS and then BSS in the hope of overcoming the drawbacks of both approaches. These sequential search algorithms suffer from beginning with extreme subsets of features (i.e., none and all features respectively). Thus, it makes sense to begin with some other subset of features, to extend them with FSS, and then use BSS since it has proven itself to perform well on many tasks. Doak (1992) reports good performance with RGSS on a heart disease database with 75 features.

BEAM differs from RGSS in three ways. First, instead of beginning with a random selection of features, BEAM randomly samples the feature space for a fixed number of iterations and begins with the best-performing feature subset found during those iterations. BEAM's performance in informal experiments without this more careful selection of an initial feature set was much lower.

Second, BEAM is biased in the initial number of features it selects. For the cloud classification task, BEAM initially sampled only in the space of 25 features among the 204 available. This is required because the space of features is large, and we know from Bankert's (1994) study that a small number of features (i.e., 15) significantly improves predictive accuracy on this task.

Third, BEAM employs a beam search (hence its name) during BSS. It maintains a fixed-sized queue of the best-performing states it has visited along with a list of the features in it that have already been removed by BSS for evaluation. The states on the queue are ordered by decreasing accuracy, and the queue is updated each time a state is selected and evaluated.

A parameter to BEAM determines whether a state evaluation consists of evaluating all subsets of one smaller number of features (*full evaluation*) or only a single feature subset (*greedy evaluation*). When using greedy evaluation, a random element is used to select both which state on the queue to examine next and which feature to remove from it. The probability that a state is chosen is a decreasing function of its location in the queue. A state in location i of a queue of size n is selected with probability

$$P(i) = \frac{n - i - 1}{\sum_{j=1}^{n} j}$$

Let L be the subset of features in the selected state, with features F, that have been removed for evaluation. The next feature selected from this state is chosen randomly according to a uniform distribution from $F - L$.

BEAM's algorithm is summarized in Figure 2. The subfunctions have been previously explained except for update_queue, which during greedy evaluation adds f to L, removes the selected state from (and reduces the number of states in) the queue when $F = L$, and properly inserts the state with features $F - \{f\}$ in the queue if its evaluation (i.e., 10-fold CV accuracy) is among the best n whose features have not yet been exhaustively removed by BSS for evaluation.
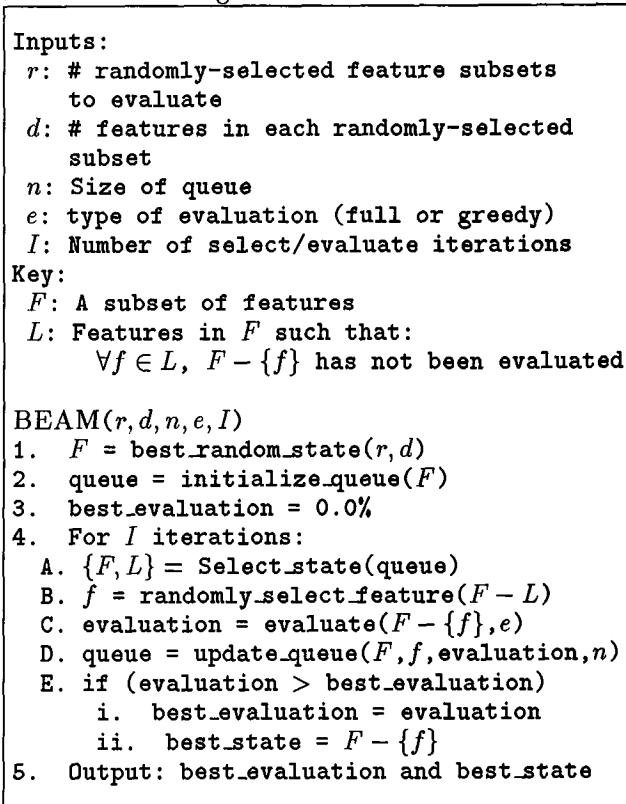
For this category, we selected as BEAM's evaluation function a separability index that is similar to the one used by Bankert and performed best in comparison with many other indices (Skalak, 1994). A full evaluation function was used with a queue size of 1.[2]

Wrapper Control with FSS

For this category, we combined FSS search with IB1 as both the evaluation function and classifier. We also tested a second variant of this combination in which we used BEAM, but substituted FSS for BSS. When

---

[2]Since a larger queue could contain feature subsets of different sizes, and since most separation indices are sensitive to feature subset size, we decided to not experiment with the greedy evaluation function for this category.

Figure 2: BEAM: A Biased Variant of BSS for Feature Selection with Large Numbers of Features

```
Inputs:
  r: # randomly-selected feature subsets
     to evaluate
  d: # features in each randomly-selected
     subset
  n: Size of queue
  e: type of evaluation (full or greedy)
  I: Number of select/evaluate iterations
Key:
  F: A subset of features
  L: Features in F such that:
       ∀f ∈ L, F - {f} has not been evaluated

BEAM(r, d, n, e, I)
1.  F = best_random_state(r, d)
2.  queue = initialize_queue(F)
3.  best_evaluation = 0.0%
4.  For I iterations:
    A. {F, L} = Select_state(queue)
    B. f = randomly_select_feature(F - L)
    C. evaluation = evaluate(F - {f}, e)
    D. queue = update_queue(F, f, evaluation, n)
    E. if (evaluation > best_evaluation)
         i.  best_evaluation = evaluation
         ii. best_state = F - {f}
5.  Output: best_evaluation and best_state
```

using the greedy evaluation function, we initialized BEAM with a randomly chosen subset of three features (i.e., the best among 25 subsets tested for that size), and used a queue size of 25. A queue size of one was used when using full evaluation, which began with the empty subset of features. In both cases, feature subsets were constrained to have size no more than 25.

Wrapper Control with BSS

The fourth category combines a wrapper control strategy with BSS search. We again used the BEAM algorithm, but in this case we used IB1 (Aha, Kibler, & Albert, 1991) as the evaluation function. Again, we tested BEAM using both the full and single state evaluation functions, and using queue sizes of one and 25 respectively.

Other Algorithms

We included two other case-based feature selection algorithms in our study. IB4 (Aha, 1992a) was chosen as an example of a classifier that hill-climbs in the space of real-valued feature weights. Several such algorithms exist (e.g., Wettschereck & Dietterich, 1994). Although these algorithms have performed well on some datasets, they tend to work best when features are either highly relevant or irrelevant. We hypothesized

that IB4 would not work well on the cloud classification task, which contains many partially-relevant features.

We also included Cardie's (1993) approach. She used C4.5 (Quinlan, 1993) to select which features to use in a case-based classifier and reported favorable results. This method has not previously been used on tasks involving more than a few dozen features, and some reports suggest that C4.5 does not always perform well when the features are all continuous (e.g., Aha, 1992b). Thus, we suspected that it may not perform well on the cloud classification task.

Empirical Comparisons

We examined two hypotheses concerning the sequential search framework described in the previous section:

1. The wrapper control strategy is superior to the filter control strategy for this task.

2. Variants of BSS tend to outperform variants of FSS for this task.

The first hypothesis is based on evidence that using the classifier itself as the evaluation function yields better performance on some tasks (e.g., Doak, 1992; Vafaie & De Jong, 1993; John, Kohavi, & Pfleger, 1994). Similar evidence exists for the second hypothesis (Doak, 1992). However, these hypotheses have not been previously investigated for tasks of this magnitude. We also have secondary hypotheses concerning the other two algorithms in our study:

3. Searching in the 2^d space is superior than searching in the $\Re^d$ space, at least when initially determining which features are needed to attain good performance on the cloud classification task.

4. C4.5 will not perform particularly well as a filter algorithm because this dataset contains only numeric-valued features.

Hypothesis three is based on the observation that $\Re^d$ is much larger than 2^d, which is itself large for our task. Thus, a hill-climbing algorithm such as IB4 will have difficulty locating a set of weights that yield good performance on this task. Finally, although C4.5 has performed well as a feature selection algorithm in previous studies, it has not been used for this purpose with numeric-valued data.

Baseline Results

When using all 204 attributes, IB1's 10-fold CV accuracy is 72.6% while PNN's accuracy is 75.3%. The most frequent concept in the database occurs for 15.4% of the cases. Both IB1 and PNN significantly increase accuracy, although PNN fares better here because IB1 does not perform well for high-dimensional tasks involving many partially relevant attributes (Aha, 1992a).

Table 2: Best and Average (10 runs for the nondeterministic algorithms) 10-fold CV Percent Accuracies (Acc) and Feature Set Sizes (Sz) for Feature Selection Algorithms on the Cloud Classification Task, where "B_s" and "B_f" Indicates Using BEAM with the Single and Full Evaluation Strategies Respectively

Control Strategy	Search Alg.	Best		Average	
		Acc	Sz	Acc	Sz
Random Selection		75.9	83	69.9	103.4
Filter	FSS	78.6	15	–	–
Filter	BSS	75.4	24	69.8	21.3
Wrapper	FSS	**88.0**	10	–	–
Wrapper	FSS B_s	87.2	25	83.1	24.9
Wrapper	BSS B_f	82.4	13	79.7	17.0
Wrapper	BSS B_s	**85.1**	5	81.0	9.1
IB4		73.3	204	–	–
C4.5		76.5	79.4	–	–

Comparison Results

Table 2 displays the best and average results when using the eight feature selection algorithms described in the previous section. Additionally, the first line refers to the results when randomly selecting subsets of features. The algorithms using the wrapper control strategy all used IB1 as their evaluation function, while the filter strategy algorithms used a separability index to evaluate states. IB1 was used as the classifier for all the algorithms.

The second line in Table 2 refers to using the subset found by Bankert (1994) and using IB1 rather than PNN as the classifier. This method is at a disadvantage because the bias of the index separation measure may not match the bias of the classifier. Thus, the set of features selected by the index measure may not yield comparatively high accuracies, as is true in this case.

The third line refers to using BSS search, a separation index, and the IB1 classifier. This approach also performed comparatively poorly. Our other results suggest this is due to using a separation index as the evaluation function.

Line four of Table 2 replaces Bankert's use of a separability index with IB1 as the evaluation function. This single change yields the highest accuracy in our study. Thus, this is strong evidence that using the classifier also as the evaluation function yields better performing algorithms for feature selection.

Line five reports a similar approach, but in this case each run was initialized with the best-performing subset of three randomly chosen features. BEAM was used (with FSS rather than BSS) with a queue size of 25. The sizes of feature subsets were constrained to not exceed 25. This algorithm found a subset with high accuracy but required a relatively larger number of features.

The sixth and seventh lines in Table 2 lists BEAM's results when using BSS as the search algorithm ($r = 25$, $d = 25$, and $I = 1000$) and IB1 as the classifier.[3] These settings have not been optimized. Line six refers to using a queue size of one and the full evaluation strategy. Line seven refers to using a queue size of 25 and the greedy evaluation strategy. This latter approach performed well; it located a subset containing only five features that attained an accuracy of 85.1%.

The eighth line in Table 2 refers to using IB4 (Aha, 1992a). Like most weight-tuning case-based algorithms, IB4 does not perform well unless each feature is either highly relevant or irrelevant. Its poor performance here is not surprising.

The final line refers to using C4.5 (Quinlan, 1993) to select features for IB1. Using C4.5's default parameter settings, it yields pruned trees that have a large number of features that do not deliver high predictive accuracies. We have not investigated whether alternative parameter settings will improve its performance.

Summary

The highest accuracy (i.e., 88.0%) was obtained when using the FSS search algorithm combined with IB1 as the evaluation function. A students one-tailed t-test confirmed that the accuracies obtained using the feature subset found by this approach, as tested on the ten folds, are significantly higher than the best accuracies obtained by using the other approaches at either the 95% confidence level or higher. In particular, using IB1 rather than a separability index for the evaluation function significantly improved performance when using either FSS (97.5% confidence level) or BSS (99.5%). Thus, this provides evidence for our hypothesis that using the classifier as the evaluation function improves performance.

The combination of BSS with IB1 as the evaluation function, as implemented in BEAM with the single evaluation option, also performed well. However, FSS with IB1 located a five-feature subset with even higher accuracy (i.e., 85.9%). Thus, we found no evidence that BSS outperformed FSS, but this may be due to the differences between BSS and its modification in BEAM.

As expected, the final two algorithms performed comparatively poorly in terms of accuracy (i.e., they were significantly lower than the accuracies recorded by the wrapper algorithms at either the 95% confidence level or higher). Furthermore, neither algorithm found small-sized feature sets in this study.

Related Work

Feature selection has its roots in pattern recognition and statistics and is addressed in several textbooks (e.g., Devijver & Kittler, 1982). Mucciardi

---

[3]The average 10-fold CV accuracy of fifty random selections of 25 features was 66.7%, while the same average for BEAM's initially selected feature sets was 74.6%.

and Gose's (1971) empirical study fostered much interest on heuristic approaches for feature selection that use filter control strategies. Cover and van Campenhout (1977) later proved that, under some assumptions, such heuristic algorithms can perform arbitrarily poorly. Yet exhaustive (exponential) search algorithms are prohibitively expensive for non-trivial tasks. Their seminal paper greatly influenced the pattern recognition community, and arguably reduced interest in heuristic approaches using filter strategies.

In contrast, several feature selection algorithms have been recently described in the AI literature. Many weight-tuning algorithms have been proposed (e.g., Aha, 1992a; Wettschereck & Dietterich, 1994), although it is not feasible to explore the space of real-valued weights in $\Re^d$ when d is large. Also, these algorithms work best when the features are either highly relevant or irrelevant, which doesn't appear to be true for the cloud classification task. However, it is possible that the notion of locally warping the instance space (e.g., Aha & Goldstone, 1992) can be used for feature selection (i.e., the relevance of features varies in a given instance space).

We suspect that FOCUS (Almuallim & Dietterich, 1991) and RELIEF (Kira & Rendell, 1992) would not perform well on this task since the former uses exhaustive search and the latter assumes that features are, again, either highly relevant or irrelevant. Vafaie and De Jong (1993) instead used a genetic algorithm to select features for a rule induction program. Like John, Kohavi, and Pfleger (1994), they found that filtering is inferior to the wrapper control strategy, although the database in their study again involved only a comparatively small number of features.

The results from Doak's (1992) survey suggest using feature selection algorithms similar to BEAM. Doak combined random with sequential searching, suggested using a beam search to guide BSS, and suggested that genetic search approaches can be improved by biasing them towards smaller-sized feature subsets. Skalak (1994) recently demonstrated that this approach is effective. BEAM extends some of these ideas by adding a strategy for selecting a good initial set of features, implementing a beam search, and introducing random elements in the state selection process, and may benefit from using a more intelligent method for selecting its initial feature subset.

Conclusions and Future Research

This paper focuses on improving predictive accuracy for a specific task: cloud classification. Properties specific to this task require the use of feature selection approaches to improve case-based classification accuracy. We explained the motivation for using sequential selection algorithms for this task, tested several variants, and introduced a novel modification of BSS named BEAM for problems with large numbers of features. Our results strongly indicate that, when using

these algorithms, the evaluation function should be the same as the classifier. This leads to significantly higher accuracies than those previously published for this task (Bankert, 1994).

We have not yet evaluated BEAM on other problems with large numbers of features to better analyze its benefits. While we anticipate that some of its properties will prove generally useful, we do not yet have evidence for such claims. Also, several of BEAM's design decisions have not been justified, and we plan to examine these decisions systematically in future research. Another goal is to integrate the capabilities of genetic search in BEAM, such as by using it whenever backward sequential selection depletes the queue of good feature subsets. Similarly, C4.5 can be utilized by selecting feature subsets from the power set of features it finds to be relevant for classification. Many other combinations might prove useful. A promising direction of future research should involve using domain-specific knowledge concerning feature relevance. Methods used in case-based reasoning should prove highly valuable once such domain expertise becomes available.

References

Aha, D. W. (1992a). Tolerating Noisy, Irrelevant, and Novel Attributes in Instance-Based Learning Algorithms. *International Journal of Man-Machine Studies, 36*, 267–287.

Aha, D. W. (1992b). Generalizing from case studies: A case study. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 1–10). Aberdeen, Scotland: Morgan Kaufmann.

Aha, D. W., & Goldstone, R. L. (1992). Concept learning and flexible weighting. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (pp. 534–539). Bloomington, IN: Lawrence Erlbaum.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning, 6*, 37–66.

Almuallim, H., & Dieterich, T. G. (1991). Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 547–552). Menlo Park, CA: AAAI Press.

Ashley, K. D., & Rissland, E. L. (1988). Waiting on weighting: A symbolic least commitment approach. In *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 239-244). St. Paul, MN: Morgan Kaufmann.

Bankert, R. L. (1994). Cloud classification of AVHRR imagery in maritime regions using a probabilistic neural network. To appear in *Journal of Applied Meteorology.*

Cain, T., Pazzani, M. J., & Silverstein, G. (1991). Using domain knowledge to influence similarity judgement. In *Proceedings of the Case-Based Reasoning Workshop* (pp. 191-202). Washington, DC: Morgan Kaufmann.

Caruana, R., & Freitag, D. (1994). Greedy attribute selection. To appear in *Proceedings of the Eleventh Inter-* national Machine Learning Conference. New Brunswick, NJ: Morgan Kaufmann.

Cardie, C. (1993). Using decision trees to improve case-based learning. In *Proceedings of the Tenth International Conference on Machine Learning* (pp. 25–32). Amherst, MA: Morgan Kaufmann.

Cover, T. M., & van Campenhout, J. M. (1977). On the possible orderings in the measurement selection problem. *IEEE Transactions on Systems, Man, and Cybernetics, 7*, 657–661.

Devijver, P. A., & Kittler, J. (1982). *Pattern recognition: A statistical approach.* Englewood Cliffs, NJ: Prentice-Hall.

Doak, J. (1992). *An evaluation of feature selection methods and their application to computer security* (Technical Report CSE-92-18). Davis, CA: University of California, Department of Computer Science.

John, G., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. To appear in *Proceedings of the Eleventh International Machine Learning Conference.* New Brunswick, NJ: Morgan Kaufmann.

Kira, K., & Rendell, L. A. (1992). A practical approach to feature selection. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 249–256). Aberdeen, Scotland: Morgan Kaufmann.

Mucciardi, A. N., & Gose, E. E. (1971). A comparison of seven techniques for choosing subsets of pattern recognition properties. *IEEE Transaction on Computers, 20*, 1023–1031.

Owens, C. (1993). Integrating feature extraction and memory search. *Machine Learning, 10*, 311-340.

Peak, J. E., & Tag, P. M. (1992). Towards automated interpretation of satellite imagery for navy shipboard applications. *Bulletin of the American Meterological Society, 73*, 995–1008.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* San Mateo, CA: Morgan Kaufmann.

Ram, A. (1993). Indexing, elaboration, and refinement: Incremental learning of explanatory cases. *Machine Learning, 10*, 201–248.

Skalak, D. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. To appear in *Proceedings of the Eleventh International Machine Learning Conference.* New Brunswick, NJ: Morgan Kaufmann.

Specht, D. F. (1990). Probabilistic neural networks. *Neural Networks, 3*, 109–118.

Vafaie, H., & De Jong, K. (1993). Robust feature selection algorithms. In *Proceedings of the Fifth Conference on Tools for Artificial Intelligence* (pp. 356–363). Boston, MA: IEEE Computer Society Press.

Welch, R. M., Sengupta, S. K., Goroch, A. K., Rabindra, P., Rangaraj, N., & Navar, M. S. (1992). Polar cloud and surface classification using AVHRR imagery: An intercomparison of methods. *Journal of Applied Meteorology, 31*, 405–420.

Wettschereck, D., & Dieterich, T. G. (1994). An experimental comparison of the nearest neighbor and nearest hyperrectangle algorithms. To appear in *Machine Learning.*