19

A Comparative Evaluation of Sequential Feature Selection Algorithms

David W. Aha[†] and Richard L. Bankert[††]

Navy Center for Applied Research in AI[†]
Naval Research Laboratory
Washington, D.C. 20375-5337
aha@aic.nrl.navy.mil

Marine Meteorology Division[††]
Naval Research Laboratory
Monterey, CA 93943
bankert@nrlmry.navy.mil

ABSTRACT Several recent machine learning publications demonstrate the utility of using feature selection algorithms in supervised learning tasks. Among these, *sequential feature selection* algorithms are receiving attention. The most frequently studied variants of these algorithms are *forward* and *backward sequential selection*. Many studies on supervised learning with sequential feature selection report applications of these algorithms, but do not consider variants of them that might be more appropriate for some performance tasks. This paper reports positive empirical results on such variants, and argues for their serious consideration in similar learning tasks.

19.1 Motivation

Feature selection algorithms attempt to reduce the number of dimensions considered in a task so as to improve performance on some dependent measures. In this paper, we restrict our attention to supervised learning tasks, where our dependent variables are classification accuracy, size of feature subset, and computational efficiency.

Feature selection has been studied for several decades (e.g., Fu, 1968; Mucciardi & Gose, 1971; Cover & van Campenhout, 1977). Several publications have reported performance improvements for such measures when feature selection algorithms are used (e.g., Almuallim & Dietterich, 1991; Doak, 1992; Kononenko, 1994; Caruana & Freitag, 1994; Skalak, 1994; Moore & Lee, 1994; Aha & Bankert, 1994; Townsend-Weber & Kibler, 1994; Langley & Sage, 1994). Feature selection algorithms are typically composed of the following three components:

1. **Search algorithm** This searches the space of feature subsets, which has size 2^d, where d is the number of features.
2. **Evaluation function** This inputs a feature subset and outputs a numeric evaluation. The search algorithm's goal is to maximize this function.
3. **Performance function** The performance task studied in this paper is classification.

[1] *Learning from Data: AI and Statistics V.* Edited by D. Fisher and H.-J. Lenz. ©1996 Springer-Verlag.

Given the subset found to perform best by the evaluation function, the classifier is used to classify instances in the dataset.

Doak (1992) identified three categories of search algorithms: exponential, randomized, and sequential. Exponential algorithms (e.g., branch and bound, exhaustive) have exponential complexity in the number of features and are frequently prohibitively expensive to use (i.e., they have complexity $O(2^d)$, where d is the number of features). Randomized algorithms include genetic and simulated annealing search methods. These algorithms attain high accuracies (Doak, 1992; Vafaie & De Jong, 1993; Skalak, 1994), but they require biases to yield small subsets. How this is best done remains an open issue. Sequential search algorithms have polynomial complexity (i.e., $O(d^2)$); they add or subtract features and use a hill-climbing search strategy. We are investigating frequently used variants of these algorithms (Aha & Bankert, 1994).

The most common sequential search algorithms for feature selection are *forward sequential selection* (FSS) and *backward sequential selection* (BSS), and we focus on these algorithms in this paper. 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 from extending the current subset. BSS instead begins with all features and repeatedly removes a feature whose removal yields the maximal performance improvement.

Doak (1992) reported that BSS frequently outperformed FSS, perhaps because BSS evaluates the contribution of a given feature in the context of all other features. In contrast, FSS can evaluate the utility of a single feature only in the limited context of previously selected features. Caruana and Freitag (1994) note this problem with FSS, but their results favor neither algorithm. Based on these observations, it is not clear whether FSS will outperform BSS on a given dataset with an unknown amount of feature interactions. Therefore, we study both in this paper.

Doak (1992) evaluated a beam search variant of FSS and noted that it frequently outperformed FSS. He predicted that a similar variant of BSS would outperform BSS. We investigate both variants in this paper.

This paper focuses on variants of BSS and FSS and their application to a sparse data set of particular interest to the Naval Research Laboratory's Marine Meteorology Division. We define a space of parameterized algorithms in which these algorithms are two of many possible instantiations. We show that feature selection improves the performance of a case-based classifier on this dataset, provide evidence for John, Kohavi, and Pfleger's (1994) conjecture that *wrapper* models outperform *filter* models (described in Section 19.3), and provide evidence against Doak's (1992) implication that BSS is always preferable to FSS. Finally, we show that variants of BSS and FSS can frequently outperform them, and argue that similar variants should always be tested when possible.

19.2 Cloud Pattern Classification Task

Feature selection algorithms hold the potential to improve classification performance for sparse datasets with many features and for datatsets where the features used to describe cases are not known to be highly relevant to the performance function (i.e., here, the

classifier). Both of these situations are true for the dataset we examined.

The cloud pattern classification dataset was provided by the Marine Meteorology Division of the Naval Research Laboratory[2] It is sparse; its number of cases is small (69) and they are each described by a comparatively large number (98) of features. Furthermore, these features were chosen under the assumption that they *may* provide useful information for classifying cloud patterns. However, no verification was performed to ensure that the chosen features are predictive of particular cloud patterns. Some of the features are known to be relevant for discriminating some cloud pattern classes, but their relevance for distinguishing all cloud patterns is unknown.

Thus, we expected this to be an ideal dataset for testing sequential selection variants that use classifiers which perform poorly in the presence of many features with potentially low relevance. It is well known that nearest neighbor algorithms perform poorly under such situations (e.g., Aha, 1992). However, such non-parametric classifiers are excellent choices for evaluation functions since manual parameter-tuning is not feasible whenever a large number of (different) feature subsets must be evaluated. Therefore, we chose IB1 (Aha, 1992) as the classifier; it is an implementation of the nearest neighbor classifier.

The features used to describe each instance consist of shape, size, spectral, and textural measures for each cloud pattern region. Examples of each group include such features as compactness, perimeter size, maximum pixel value, and cluster prominence, respectively. All features are defined over a continuous range of values. Bankert (1994a; 1994b) describes these features in more detail.

19.3 Framework

In our experiments, we investigated using FSS and BSS as the search algorithm, and we used IB1 as the classifier. We used both IB1 and the Calinski-Harabasz separability index measure as the evaluation function.[3] Our selection of these two evaluation functions was motivated by the hypothesis that *wrapper* models, which use the classifier itself as the evaluation function, outperform *filter* models, which do not. This was conjectured by John, Kohavi, and Pfleger (1994). Doak (1992) cited some informal evidence for this conjecture but did not describe a detailed analysis.

Three additional search-related variables were varied in our experiments that allowed us to test beam searching variants of FSS and BSS. These variables constrain how a beam search is conducted. Figure 19.3 describes our parameterizable framework, which we refer to as BEAM. Briefly, BEAM implements beam-search variants of the given search algorithm. A *queue* of *states* is maintained, where a state is a subset of features and an indication as to which subsets immediately derivable from it (i.e., the state's children) have not yet been evaluated. The evaluation of each state is maintained with it on the queue. Initialize_queue initializes the queue with either the complete set of features (for BSS) or the empty set (for FSS). Its evaluation is computed, and this is recorded as the initial *best* state. BEAM then loops until the queue is empty. The first step of

[2]Bankert (1994b) describes feature selection experiments with a variant of this dataset, but the differences in these datasets prevent a direct comparison.

[3]Milligan and Cooper (1985) found that the Calinski-Harabasz separability index performed best among 30 separability indices in their experiments.

```
Inputs:
 S: Search algorithm
 E: Evaluation function
 q: Size of the queue
 m: Number of states evaluated per iteration
 n: Number of subsets evaluated per state
Partial Key:
 queue: An ordered queue of states
 best: The best-performing state

BEAM(S, E, q, m, n)
1. queue = Initialize_queue(S, E)
2. best = Initialize_best(queue)
3. While (queue is not empty) do:
  A. states = Select_states(queue,m)
  B. evaluations = Evaluate_states(S,E,states,n)
  C. queue = Update_queue(states,evaluations,queue,q)
  D. best = Update_best(best,queue)
4. Output: best
```

FIGURE 1. BEAM: A Framework for Sequential Feature Selection

the loop stochastically selects m states from the queue, where states higher in the queue have a higher probability of being selected.[4] The next step evaluates n child states (i.e., subsets for BSS, supersets for FSS) derivable by search algorithm S from each of these m states using evaluation function E. The queue is then updated; states that have been exhaustively searched (i.e., all its children have been evaluated) by S are dropped, and newly-evaluated states that are among the current top q states are integrated into their appropriately ordered locations in the queue. Finally, the best state located so far is maintained in *best*. After the queue has been exhausted, this best state is output for use by the classifier. The user must provide the five inputs listed in Figure 1 (i.e., S, E, q, m, and n).

19.4 Empirical Comparisons

All experiments were run using a leave-one-out strategy. We report averages for classification accuracy, size of selected feature subset, and number of subsets evaluated. Averages were computed over ten runs for all the experiments in which IB1 was used as the evaluation function. Our implementation of the Calinski-Harabasz separability index measure is deterministic because ties rarely occur when it evaluates states (i.e., when used as an evaluation function, it outputs a continuous value). Ties occur more frequently when us-

[4]More specifically, this function selects ordered state i with probability $P(i) = (q - i + 1)/\sum_{j=1}^{q} j$, where q is the size of the queue.

TABLE 19.1. Best and Averages (Accuracy, Size of Selected Subset, and Number of Subsets Evaluated) for IB1 When Using FSS and BSS (with a Queue Size of One) for Two Different Evaluation Functions

Search	EvalFn	Best			Averages		
		Acc	Sz	#Eval	Acc	Sz	#Eval
IB1		62.3%	98	1	-	-	-
FSS	IB1	89.9%	6	4,852	88.1%	13.1	4,852.0
BSS	IB1	84.1%	18	4,699	79.9%	30.8	4,343.8
FSS	Index	66.7%	20	4,852	-	-	-
BSS	Index	68.1%	45	4,849	-	-	-

ing IB1 since BEAM uses it to evaluate states in a leave-one-out test on classification accuracy. Since tied states are randomly ordered (among themselves) on the queue, this can cause IB1 to behave differently on each run. Thus, Table 1 shows averages when IB1 is used as the evaluation function, but not otherwise.

The following subsections summarize our investigations of four hypotheses. Sections 4.1, 4.2, and 4.3 discuss results with the standard FSS and BSS algorithms, whereas Section 4.4 concerns their beam search variants.

19.4.1 Feature selection is appropriate for this dataset

Since this dataset is sparse (i.e., 98 features, only 69 instances) and several features have unknown and possibly low classification relevance, we expected feature selection to improve IB1's performance. Table 19.1 displays the results for IB1 and the average results for four feature selection algorithms (i.e., the FSS and BSS search algorithms were tested using both IB1 and the Calinski-Harabasz separability index as the evaluation function). Feature selection consistently increased accuracy and reduced feature set size. These results support this hypothesis.

19.4.2 The wrapper model is superior to the filter model

Two evaluation functions were compared: IB1 and the Calinski-Harabasz separability index. We expected that, since IB1 was also used as the classifier, using it as the evaluation function would yield superior results.

As shown in Table 19.1, the performance of the standard FSS and BSS algorithms (i.e., $q = 1$, $m = 1$, $n =$ all) is much lower when using the separability index measure rather than IB1 as the evaluation function. This supports our hypothesis. We believe the reason for this is that the separability index measure's bias is different than IB1's bias. Therefore, the index measure will not necessarily select a good-performing subset for IB1.

19.4.3 BSS *outperforms* FSS

Doak (1992) found that BSS often outperformed FSS in his studies, but he examined smaller numbers of features in his task. We were unsure which algorithm would yield better results.

The average accuracies for FSS were higher than for BSS, and their differences in accuracy were significant according to a one-tailed t-test ($p < 0.05$). FSS also found

TABLE 19.2. Best and Averages when using IB1 as the Evaluation Function and a Queue Size of 25

Search	m	n	Best			Averages		
			Acc	Sz	#Eval	Acc	Sz	#Eval
FSS	1	1	91.3%	20	57,122	87.8%	21.8	56,387.4
FSS	1	all	94.2%	22	110,058	89.3%	19.7	117,015.1
FSS	all	all	95.7%	16	114,850	94.2%	13.7	114,854.9
BSS	1	1	89.9%	12	2,871	86.8%	10.0	3,918.9
BSS	1	all	91.3%	10	8,031	86.5%	10.6	10,764.5
BSS	all	all	87.0%	11	110,881	84.1%	23.6	104,837.3

significantly ($p < 0.05$) smaller subset sizes but required significantly more evaluations ($p < 0.1$).

Table 19.2 summarizes the results for the two search algorithms when we varied the number of states selected per evaluation (m) and the number of subsets evaluated per selected state (n) while using a queue size (q) of 25. Although FSS's average accuracy is higher for all three variants, it is significantly higher only for the third variant ($p < 0.01$). FSS also yields significantly larger-sized subsets only for the first variant ($p < 0.1$), and evaluated significantly more subsets ($p < 0.05$) in all three cases.

The accuracy results differ from the majority of results found by Doak (1992), thus contradicting this hypothesis. We suspect that BSS is more easily confused by large numbers of features because the deletion of a single feature, though perhaps relevant, can have less effect in the presence of a larger number of features. We plan to explore this hypothesis further in systematic experimentation with artificially generated data. However, the general pattern found here suggests that FSS is preferred when the optimal number of selected features is small while BSS is preferred if otherwise. This also suggests that BSS's performance can be enhanced in these situations when it is biased towards using small-sized subsets of features. We show evidence for this in (Aha & Bankert, 1994). Similarly, in situations when most features are relevant, we expect that FSS' performance can be enhanced by biasing its search to start with large-sized subsets of features.

19.4.4 Beam search variants outperform standard FSS *and* BSS

Caruana and Freitag (1994) report evidence for this hypothesis when using combinations of these algorithms, but not for simple extensions of them. Doak (1992) found that a beam search version of FSS was preferable to it, but did not evaluate such extensions of BSS.

The accuracies for the beam searching variants in Table 19.2, for which the queue size was 25, were all higher than when the queue has size one (Table 19.1). These differences were significant for the third variant of FSS and all variants of BSS ($p < 0.05$). Furthermore, the BSS beam-search variants also significantly reduced the size of the selected feature subsets ($p < 0.1$). However, these sizes increased significantly for two of the FSS variants ($p < 0.05$). Finally, the number of evaluated subsets significantly increased for five of the six variants tested ($p < 0.005$), though this decreased significantly for BSS ($m = 1, n = 1$) ($p < 0.1$).

In summary, many beam search variants of the standard sequential feature selection approaches can significantly increase classification accuracies on this task. Simultaneously,

some can also significantly reduce the size of the subsets located, but they tend to significantly increase search requirements (i.e., the number of subsets evaluated).

19.5 Conclusions

This paper examines variants of forward and backward sequential feature selection algorithms for a cloud pattern classification task defined by a sparse dataset with numerous features. When using a nearest neighbor algorithm as the classifier, our results show that (1) feature selection improves accuracy on this task, (2) using the classifier as the evaluation function yields better performance than when using a separability index, (3) BSS does not always outperform FSS (contrary to some claims), and (4) beam search variants of these algorithms can improve accuracy, at least on this task.

We found similar results with other datasets and plan to discuss them in extensions of this paper. Additional studies showed that using random initial subset selection with a comparatively small number of features drastically reduces the amount of search performed by BSS without sacrificing accuracy.

Acknowledgements

Thanks to John Grefenstette, Paul Tag, Karl Branting, Pat Langley, Diana Gordon, Doug Fisher, and the reviewers for their comments and suggestions.

19.6 References

Aha, D. W. (1992). 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., & Bankert, R. L. (1994). Feature selection for case-based classification of cloud types: An empirical comparison. In D. W. Aha (Ed.) *Case-Based Reasoning: Papers from the 1994 Workshop* (Technical Report WS-94-01). Menlo Park, CA: AAAI Press.

Almuallim, H., & Dietterich, 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.

Bankert, R. L. (1994a). Cloud classification of AVHRR imagery in maritime regions using a probabilistic neural network. *Journal of Applied Meteorology*, *33*, 909–918.

Bankert, R., L. (1994b). Cloud pattern identification as part of an automated image analysis. *Proceedings of the Seventh Conference on Satellite Meteorology and Oceanography* (pp. 441–443). Boston, MA: American Meteorological Society.

Caruana, R., & Freitag, D. (1994). Greedy attribute selection. In *Proceedings of the Eleventh International Machine Learning Conference* (pp. 28–36). New Brunswick, NJ: 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.

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.

Fu, K. S. (1968). *Sequential methods in pattern recognition and machine learning.* New York: Academic Press.

John, G., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Machine Learning Conference* (pp. 121–129). New Brunswick, NJ: Morgan Kaufmann.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. In *Proceedings of the 1994 European Conference on Machine Learning* (pp. 171–182). Catania, Italy: Springer Verlag.

Langley, P., & Sage, S. (1994). Oblivious decision trees and abstract cases. In D. W. Aha (Ed.), *Case-Based Reasoning: Papers from the 1994 Workshop* (Technical Report WS-94-01). Menlo Park, CA: AAAI Press.

Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, *50*, 159–179.

Moore, A. W., & Lee, M. S. (1994). Efficient algorithms for minimizing cross validation error. In *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 190–198). New Brunswick, NJ: 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.

Skalak, D. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the Eleventh International Machine Learning Conference* (pp. 293–301). New Brunswick, NJ: Morgan Kaufmann.

Townsend-Weber, T., & Kibler, D. (1994). Instance-based prediction of continuous values. In D. W. Aha (Ed.), *Case-Based Reasoning: Papers from the 1994 Workshop* (Technical Report WS-94-01). Menlo Park, CA: AAAI Press.

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.