The Greedy Prepend Algorithm for Decision List Induction

Deniz Yuret[1] and Michael de la Maza[2]

[1] Koç University, Istanbul, Turkey, dyuret@ku.edu.tr
[2] Park Hudson Finance, Cambridge, MA 02139, USA

Abstract. We describe a new decision list induction algorithm called the Greedy Prepend Algorithm (GPA). GPA improves on other decision list algorithms by introducing a new objective function for rule selection and a set of novel search algorithms that allow application to large scale real world problems. GPA achieves state-of-the-art classification accuracy on the protein secondary structure prediction problem in bioinformatics and the English part of speech tagging problem in computational linguistics. For both domains GPA produces a rule set that human experts find easy to interpret, a marked advantage in decision support environments. In addition, we compare GPA to other decision list induction algorithms as well as support vector machines, C4.5, naive Bayes, and a nearest neighbor method on a number of standard data sets from the UCI machine learning repository.

1 Introduction

The Greedy Prepend Algorithm (GPA) is an induction system for decision lists. A decision list is an ordered list of rules where each rule consists of a pattern and a classification [1]. A new instance is classified by comparing it to the pattern of each rule, and the classification of the first matching rule is accepted. An example of a three rule decision list for the UCI house votes data set [2] is shown in Table 1.

Table 1. A three rule decision list for the UCI house votes data set

```
1. If adoption-of-the-budget-resolution = y
      and anti-satellite-test-ban = n
      and water-project-cost-sharing = y
   then democrat
2. If physician-fee-freeze = y
   then republican
3. If TRUE then democrat
```

To learn a decision list from a given set of training examples the general approach is to start with a default rule or an empty decision list and keep adding the best rule to cover the unclassified or misclassified examples. The new rules can be added to the end of the list [3], the front of the list [4], or other positions [5]. Other design decisions include the criteria used to select the "best rule" and how to search for it.

GPA works by prepending rules to the front of a growing decision list and distinguishes itself with its simple "best rule" definition. When a rule is added to a decision list, the increase in the number of correctly predicted training set instances is called the rule's *gain*. In GPA the best rule is defined to be the rule that has the maximum gain. Earlier decision list induction systems typically used variants of the Laplace preference function [6], which emphasizes the rule's individual accuracy rather than its contribution to the decision list as a whole. We show that the simple maximum gain rule results in comparable or better overall accuracy on a number of standard data sets.

By default, GPA uses an efficient admissible search algorithm (OPUS [7]) which is guaranteed to find the maximum gain rule. However, in real world problems with a large number of training instances and distinct feature values such exhaustive search may become prohibitively expensive. To cope with large scale problems GPA introduces two novel heuristic search algorithms: one that restricts the rule set, and another that restricts the search algorithm.

To illustrate the performance of the algorithm on large scale problems we present two applications of GPA, one to the secondary structure prediction problem in bioinformatics (Section 3) and the second to English part of speech tagging (Section 4). Comparisons with other decision list inducers on these problems were not feasible due to the scale of the problems. Thus we compare the classification accuracy of GPA to other decision list inducers and well-known concept learning algorithms on a representative set of problems from the UCI Machine Learning Repository (Section 5). Section 2 describes the GPA algorithm.

2 The Greedy Prepend Algorithm

To construct the decision list in Table 1, GPA first generates the default rule (rule #3) which assigns every instance the majority class in the training set. Next, candidates for rule #2 are searched, and the one that results in the highest gain in accuracy for the training set is selected. The whole decision list is constructed from the bottom up until no further improvement can be made. Note that, when the completed decision list is applied to classify an instance, rules are applied from the top (rule #1) to the bottom (rule #3).

The high-level Greedy Prepend Algorithm is given in Table 2. MAX-GAIN-RULE finds the rule that results in the largest increase in the training set accuracy when added to the beginning of a decision list. By default, an admissible search algorithm (OPUS [7]) is used to search all possible rules. For large scale problems where this approach is not feasible, either the search algorithm (see Section 3) or the set of rules (see Section 4) can be restricted. The selection of rules based

Table 2. Greedy Prepend Algorithm

```
GPA(data)
1  dlist ← NIL
2  default-class ← MOST-COMMON-CLASS(data)
3  rule ← [if TRUE then default-class]
4  while GAIN(rule, dlist, data) > 0
5        do dlist ← PREPEND(rule, dlist)
6           rule ← MAX-GAIN-RULE(dlist, data)
7  return dlist
```

on gain, and the heuristic search methods distinguish GPA from other decision list algorithms such as Prepend [4].

By default, GPA reserves 20% of its training set for validation. As more and more rules are added to the decision list, the performance on the validation set peaks, then starts dropping even though the training set performance is improved with every rule. GPA prevents overfitting by stopping the rule generation when the validation performance starts to drop.

The antecedents of the GPA rules can represent binary and nominal attributes naturally. Missing values do not pose a problem: an instance with a missing attribute will not match a rule that specifies a value for that attribute. For datasets with numeric attributes we use the MDL method of supervised discretization [8, 9]. We force the resulting attributes to be binary (i.e. less than, greater than conditions) so that GPA can construct arbitrary ranges using only two terms of the form `attr >= x1 and attr <= x2` in the antecedent.

3 Secondary Structure Prediction

In this section, we present an application of GPA to a large scale problem and illustrate the use of its heuristic search algorithm. Specifying the tertiary structure of a protein, the location of all of its constituent amino acids in Euclidean space, is key to understanding its function and is one of the outstanding unsolved problems in biology. Ideally, the tertiary structure could be determined from the primary structure, the linear sequence of amino acids that compose the protein. A step towards the tertiary structure is the secondary structure which classifies each amino acids into one of three groups: alpha helix, beta strand, and loop.

Secondary structure prediction has been studied for several decades and dozens of standard machine learning techniques have been applied to the problem. Chou [10] described a simple rule based method for predicting secondary structure. Levin [11] introduced the notion of using multiple sequence alignment to improve prediction. Rost [12] combined neural networks and multiple sequence alignment. Huang [13] discusses the limitations of many secondary structure prediction techniques.

Data Set: Our GPA evaluation for the secondary structure prediction problem is based on the recommendations and the data set described in [14]. The data set, CB513, consists of 513 non-homologous proteins that have been used in a number of studies. These proteins, along with their homologues, were used with seven fold cross validation to obtain our accuracy figures. The data set includes 1.7 million instances, nine attributes with 180 distinct values, and three classes.

Methodology: Two decision lists were built to perform secondary structure prediction. The sequence-to-structure list takes a window of 9 adjacent residue names as input and outputs a prediction of one of three secondary structure assignments (helix, strand, or loop) for the center residue. The structure-to-structure list takes a window of 19 adjacent secondary structure *assignments* from the first decision list as input, and outputs a possible correction for the center residue. During testing both decision lists are applied to the test protein and its homologues, and final assignments are determined by majority vote. This type of organization is common in the literature [12], although implemented using different learners like neural networks.

Search Algorithm: Learning disjunctions is critical in the secondary structure prediction domain so that the decision lists can capture regularities across classes of residues. However, searching the space of all rules with disjunctions is computationally infeasible and current search techniques, such as OPUS [7], cannot be used in this problem domain.

We implemented a heuristic technique that finds good disjunctive rules. The search uses randomly selected instances to gradually construct a candidate rule. If the instance is misclassified by the current decision list and matches the class of the candidate rule, the rule is generalized to include the instance. If the instance is correctly classified by the current decision list but the candidate rule changes its classification, the rule is specialized to exclude the instance. After every modification, the change in the gain is computed and the modification is accepted if there is an overall increase in the rule's gain. The search terminates after a prespecified number of modifications fail to improve the candidate rule. A candidate rule is constructed for each class, and the one with the highest gain is accepted to the decision list.

Table 3. Performance results for the set of CB513 proteins

Algorithm:	PHD	DSC	Predator	NNSSP	GPA
Accuracy:	72.3	69.1	69.0	71.7	69.2

Results: GPA achieves a classification accuracy of 69.2% on CB513. To put this number in perspective, always predicting the most common class (loop) gives a

prediction accuracy of 43%. Predicting the class based only on the identity of the center amino acid gives a prediction accuracy of 49%.

The secondary structure problem has been the subject of intense interest in computational biology and machine learning. Table 3 shows the classification accuracy of several of the leading algorithms on CB513. PHD is a neural network based system [12], DSC is a statistical technique [15], PREDATOR is a knowledge based system [16], and NNSSP is a nearest neighbor algorithm [17]. GPA gives a strong result and is the only one that produces human readable models.

4 Part of Speech Tagging

In this section, we present an application of GPA to a linguistic problem and illustrate the use of rule space restriction to cope with the large search space. Determining the part of speech of ambiguous words is a useful step in the analysis of natural language text. The sentence "Time flies like an arrow" can be interpreted in several different ways depending on the part of speech assignments. The word "time" can be a noun (time moves quickly just like an arrow does), a verb (measure the speed of flying insects like you would measure that of an arrow), or an adjective (a type of flying insect, "time-flies," enjoy arrows; compare "Fruit flies like a banana.") The problem of part of speech tagging is to decide what part of speech each word has depending on its context.

Data Set: We have used the Penn Treebank Wall Street Journal corpus [18] for evaluating the performance of GPA for part of speech tagging. The Penn Treebank uses a set of 45 distinct part of speech tags to annotate about a million words of newspaper text. The data set consists of one million instances with fifty attributes and half a million distinct attribute values. The data was split randomly intro training, validation, and test sets using a 4:1:1 ratio.

Methodology: To represent the context of a word to be tagged, we take a window of seven words centered around the target word and encode the following information about each one in the attributes:

1. The original word and its lowercase version.
2. The character types (upper case, numeric etc.) used in the word.
3. Prefixes and suffixes of the word up to four characters long.

Common domain specific optimizations were not performed on our GPA implementation: We did not use an explicit dictionary that lists the possible parts of speech for each word, or use previous part of speech predictions in the model, or train a separate model for the unknown words. Nevertheless GPA seems to extract the necessary information from the surface features of the surrounding words and achieve comparable accuracy to previously reported results.

The fact that these attributes are very redundant (e.g. the three character suffix of a word is correlated with its four character suffix) and mostly irrelevant

(no attribute filtering was performed) does not seem to prevent GPA from getting a good result. More formal analysis of the robustness of GPA in the presence of redundant and irrelevant attributes needs to be performed.

Search Algorithm: The large size of the search space precludes an exhaustive search for the maximum gain rule. Representing the suffixes and strings of each word encountered during training creates a large number of distinct values for some attributes. To cope with the large search space, we limited the search to rules that add a single attribute to one of the already accepted rules in the decision list. As a result, after the default rule, only single attribute rules are searched. After the first single attribute rule, all the two attribute rules which share the same first attribute, and all the remaining single attribute rules are searched. This way each rule is forced to specify an exception to a previously accepted rule that can be expressed by a single attribute.

Table 4. Performance results for part of speech tagging

Algorithm:	Post	Brill	Mxpost	GPA
Overall:	96.7	96.6	96.6	96.0
Unknown:	85.0	82.2	86.2	81.8

Results: Part of speech tagging is a much studied problem with a number of successful approaches. We present the results of three well known algorithms for comparison in Table 4. The "overall" column gives the overall accuracy, and the "unknown" column gives the accuracy on unknown words. Post is a Markov model based stochastic tagger [19], Brill's tagger uses transformation based learning [20], and Mxpost is based on a maximum entropy model [21].

5 UCI Machine Learning Repository

The UCI Machine Learning Repository [2] has a long and storied history in the machine learning community. Although its influence has waned in recent years, testing a new algorithm on a variety of UCI data sets serves as a sanity check on the algorithm's characteristics.

We present two sets of comparisons in this section. First, we compare the accuracy of GPA and the lengths of the decision lists it generates to other decision list induction algorithms analyzed in [4]. Then, we report results with a number of standard machine learning algorithms implemented in Weka [9].

Most UCI datasets are small enough to allow exhaustive search for the maximum gain rule. We therefore used the admissible OPUS algorithm [7] to find the best rule. On some problems a limit on the maximum number of terms in the antecedent of a rule was imposed for the sake of efficiency.

5.1 Comparison with other decision list algorithms

Webb [4] compares the accuracy and average list length of a number of decision list induction algorithms. Prepend builds decision lists from the bottom towards the top like GPA, but uses the Laplace preference function which emphasizes the individual accuracy of a rule. GPA, in contrast, uses the maximum gain function which only measures the increase in the performance of the whole decision list caused by the new rule. Append is a variation of the CN2 algorithm [3] which builds decision lists from the top towards the bottom. Append-p2 implements rule pruning. C4.5rules is a standard algorithm which generates rules from a decision tree [22].

Table 5 presents accuracy and average list length for GPA and Prepend on the 11 problems used in [4]. The GPA results were obtained repeating 5 fold cross validation 100 times. The number of terms in the antecedents were limited to 3 for the sake of time.

Table 5. Comparison of GPA and Prepend on accuracy and list length on some UCI data sets (standard deviations are given in parentheses)

	Accuracy		Length	
Data set	GPA	Prepend	GPA	Prepend
audio	74.55 (5.78)	77.2 (5.4)	25.3 (2.15)	10.7 (0.9)
breast	71.96 (4.28)	61.2 (5.6)	4.5 (3.75)	9.6 (1.1)
votes	95.10 (2.41)	94.9 (2.5)	4.1 (1.76)	5.0 (0.5)
lymph	76.60 (7.27)	84.3 (7.4)	7.7 (2.19)	5.1 (0.6)
monks-1	100.0 (0.0)	100.0 (0.0)	5.9 (1.14)	4.7 (0.5)
monks-2	64.94 (2.07)	99.1 (1.5)	64.1 (4.8)	15.8 (0.4)
monks-3	98.92 (0.88)	97.9 (1.2)	4.0 (0.0)	4.9 (0.7)
mushroom	100.0 (0.0)	100.0 (0.0)	8.0 (0.0)	4.0 (0.0)
soybean	89.59 (2.69)	83.6 (5.3)	26.0 (8.6)	20.7 (0.8)
tic-tac-toe	98.41 (0.92)	96.9 (1.0)	9.0 (0.0)	12.0 (0.8)
tumor	40.8 (4.32)	39.1 (5.3)	11.1 (6.67)	16.8 (1.6)

A two tailed z test at 0.05 significance level was used to compare the GPA results to the ones reported in [4]. Compared to Prepend, GPA had significantly higher accuracy in 5 problems and worse in 3 problems. The list lengths were shorter than prepend 5 times and longer 6 times. GPA exceeds the performance of the original Prepend in accuracy and matches it in length. Comparison with the other algorithms is summarized in Table 6. The number triples indicate the number of problems in which GPA won, drew, and lost at the 0.05 significance level. GPA seems to achieve significantly higher accuracy than append and append-p2 and build shorter decision lists than append and c4.5rules.

Table 6. Win:draw:loss numbers comparing GPA to other rule based algorithms

Algorithm	Accuracy	Length
prepend-ip2	5:3:3	5:0:6
append	6:3:2	7:2:2
append-p2	5:4:2	5:1:5
c4.5rules	4:6:1	8:1:2

5.2 Comparison with other learning algorithms

Table 7 summarizes GPA's performance versus naive Bayes, support vector machines (SMO), a nearest neighbor algorithm (IB5), C4.5 (J4.8), and a decision table learner (DTable) as implemented in the Weka tool [9]. All algorithms were run in their default settings. GPA was restricted to rules with at most three antecedent terms. Each of the 20 data sets were evaluated with five fold cross validation repeated 10 times. The results were compared with a two tailed paired t-test at 0.05 significance level.

The key result is that the performance accuracy of GPA is comparable to leading concept learning algorithms. For example, compared to support vector machines GPA is better ($p < 0.05$) on five data sets, worse ($p < 0.05$) on three data sets, and statistically indistinguishable on the rest.

Table 7. Comparison of GPA with other common learning algorithms. Standard deviations are given in parentheses. A (*) next to a result shows that it is statistically significantly worse than GPA and a (v) shows that it is significantly better than GPA. The last line gives the win:draw:loss numbers for each column.

Dataset	GPA	NBayes	SMO	IB5	J48	DTable
anneal	98.83(0.76)	86.80(2.41) *	97.37(1.10) *	96.97(1.30) *	98.62(0.76)	98.50(0.87)
audio	75.42(6.55)	71.37(4.44)	80.36(4.53)	60.67(5.22) *	77.57(4.77)	74.35(6.21)
balance	74.42(3.84)	89.97(1.31) v	87.82(1.96) v	87.60(1.70) v	78.11(2.49)	76.78(3.83)
breast	71.57(3.39)	73.21(5.00)	69.19(4.81)	74.02(3.52)	73.18(3.75)	71.47(4.80)
glass	69.27(6.59)	48.45(5.95) *	57.06(6.96) *	65.24(4.69)	67.45(7.13)	68.80(7.01)
heart-clev	79.24(5.81)	83.04(4.97)	83.40(4.37)	81.73(4.43)	76.24(4.80)	77.76(6.56)
heart-hung	78.47(4.24)	83.91(4.40) v	82.15(4.55)	81.84(4.68)	79.83(5.56)	79.77(4.68)
hepatitis	79.29(5.53)	83.42(5.09)	85.16(4.56)	84.32(4.88)	78.52(5.44)	78.77(6.22)
hypo	99.52(0.32)	95.33(0.48) *	93.58(0.29) *	93.26(0.40) *	99.50(0.26)	99.36(0.31)
iris	94.60(3.74)	95.47(3.55)	96.33(2.95)	96.13(3.04)	94.53(3.67)	93.73(3.27)
lymph	76.91(5.65)	82.71(7.02)	85.89(5.78) v	82.58(5.79)	76.11(6.53)	74.41(6.80)
monks-1	100.00(0.0)	74.62(2.75) *	74.64(2.77) *	98.44(1.37) *	97.73(3.80)	100.00(0.0)
monks-2	65.24(1.94)	62.33(2.54) *	65.72(0.22)	57.52(3.29) *	63.18(4.40)	65.72(0.22)
monks-3	98.92(0.80)	96.39(1.45) *	97.15(1.80)	97.71(1.29)	98.92(0.80)	98.92(0.80)
mushroom	100.00(0.0)	95.66(0.52) *	100.00(0.0)	99.98(0.05)	100.00(0.0)	100.00(0.0)
segment	94.44(1.15)	80.27(1.47) *	92.67(0.99) *	95.18(0.93)	96.47(1.14) v	91.60(1.71) *
sonar	70.76(6.68)	68.29(6.81)	76.39(5.86)	80.29(6.28) v	72.98(7.16)	70.74(7.15)
soybean	89.63(2.55)	92.65(1.92) v	93.03(1.61) v	89.66(2.18)	90.57(2.52)	85.78(2.76) *
tic-tac-toe	98.14(1.10)	70.26(2.91) *	98.33(0.84)	98.85(0.76)	83.98(2.26) *	77.40(2.89) *
tumor	40.62(4.49)	48.47(3.62) v	45.87(4.61)	46.96(4.36) v	40.64(4.35)	39.36(4.79)
		(9/7/4)	(5/12/3)	(5/12/3)	(1/18/1)	(3/17/0)

6 Contributions

The Greedy Prepend Algorithm is a decision list induction system that achieves state-of-the art classification accuracy on secondary structure prediction and part of speech tagging and matches the performance of C4.5 and support vector machines on a representative set of data sets from the UCI Machine Learning Repository.

The main contributions of the GPA are its simple objective function for rule selection and its heuristic search extensions which allow decision lists to be applied to large scale problems with thousands of features and millions of instances. The results show that decision lists can be competitive with the best known algorithms in large scale real world problems. Unlike Bayesian methods, neural networks, nearest neighbor algorithms, and support vector machines, rule based learners like GPA produce output that is human readable. Hence GPA is a better fit in decision support environments where a decision maker benefits from an explanation for the recommendations made by automated concept learning systems. GPA demonstrates by example that high classification accuracy is possible while still preserving a representation that can be easily interpreted by humans.

A GPA implementation that runs within WEKA [9], an open source environment for running machine learning experiments, is available from the first author.

References

1. Rivest, R.L.: Learning decision lists. Machine Learning **2** (1987) 229–246
2. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998) http://www.ics.uci.edu/∼mlearn/MLRepository.html.
3. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning **3** (1989) 261–283
4. Webb, G.I.: Recent progress in learning decision lists by prepending inferred rules. In: Proceedings of the Second Singapore International Conference on Intelligent Systems (SPICIS '94), Singapore (1994) B280–B285
5. Newlands, D., Webb, G.I.: Alternative strategies for decision list construction. In: Proceedings of the Fourth Data Mining Conference (DM IV 03). (2004) 265–273
6. Clark, P., Boswell, R.: Rule induction with CN2: Some recent improvements. In Kodratoff, Y., ed.: Machine Learning – Proceedings of the Fifth European Conference (EWSL-91), Berlin, Springer-Verlag (1991) 151–163
7. Webb, G.I.: Opus: An efficient admissible algorithm for unordered search. JAIR **3** (1995) 431–465
8. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the Workshop on Massive Datasets, Washington, DC, NRC, Committee on Applied and Theoretical Statistics (1993)
9. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2 edn. Morgan Kaufmann (2005)
10. Chou, P.Y., Fasman, G.D.: Conformational parameters for amino acids in helical, beta sheet and random coil regions calculated from proteins. Biochemistry **13**(2) (1974) 211–222

11. Levin, J.M., Pascarella, S., Argos, P., Garnier, J.: Quantification of secondary structure prediction improvement using multiple alignment. Prot. Engin. **6** (1993) 849–854
12. Rost, B., Sander, C.: Prediction of protein secondary structure at better than 70% accuracy. Journal of Molecular Biology **232** (1993) 584–599
13. Huang, J.T., Wang, M.T.: Secondary structural wobble: The limits of protein prediction accuracy. Biochemical and Biophysical Research Communications **294**(3) (2002) 621–625
14. Cuff, J.A., Barton, G.J.: Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. Proteins: Structure, Function, and Genetics **34** (1999) 508–519
15. King, R.D., Sternberg, M.J.E.: Identification and application of the concepts important for accurate and reliable protein secondary structure prediction. Protein Sci **5** (1996) 2298–2310
16. Frishman, D., Argos, P.: Seventy-five percent accuracy in protein secondary structure prediction. Proteins: Structure, Function, and Genetics **27** (1997) 329–335
17. Salamov, A.A., Solovyev, V.V.: Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments. Journal of Molecular Biology **247** (1995) 11–15
18. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics **19**(2) (1993) 313–330
19. Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L.: Coping with ambiguity and unknown words through probabilistic models. Computational Linguistics **19**(2) (1993) 359–382
20. Brill, E.: Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. Computational Linguistics **21**(4) (1995) 543–565
21. Ratnaparkhi, A.: A maximum entropy model for part-of-speech tagging. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. (1996)
22. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)