In *ECML-95: Proceedings of the Eighth European Conference on Machine Learning,* Heraklion, Crete, Greece, 1995.

Learning Classification Rules Using Lattices (Extended Abstract)

Mehran Sahami

Computer Science Department, Stanford University, Stanford, CA 94305, USA
Email: sahami@CS.Stanford.EDU

Abstract. This paper presents a novel induction algorithm, Rulearner, which induces classification rules using a Galois lattice as an explicit map through the search space of rules. The Rulearner system is shown to compare favorably with commonly used symbolic learning methods which use heuristics rather than an explicit map to guide their search through the rule space. Furthermore, our learning system is shown to be robust in the presence of noisy data. The Rulearner system is also capable of learning both decision lists and unordered rule sets allowing for comparisons of these different learning paradigms within the same algorithmic framework.

1 Introduction

Research in rule induction by means of search [MC, 1969; MI, 1982; CN, 1989] has been on-going for some time. While some systems, such as Version Spaces, make direct use of the data to be learned during rule induction, such methods are highly sensitive to noisy data. In other systems, the search through the space of rules is guided by heuristics as opposed to using the data to build an explicit map through the space of rules to induce. In this capacity, such algorithms are only *data-driven* to the extent that the heuristics employed in them make use of the data to be learned.

We present the Rulearner system which seeks to combine both the direct use of data with robustness in the presence of noise during the rule induction process. Since the algorithm uses a Galois lattice constructed from training data [OO, 1988] as an explicit guide through the rule space, the algorithm is directly *data-driven* as it does not simply heuristically fit the training data. Our system is also capable of inducing both an unordered set of classification rules as well as a decision list [RI, 1987]. This allows for the Rulearner system to be used in making direct comparisons between these two learning paradigms within a single algorithmic framework.

Several experiments with the Rulearner system are presented comparing it with the commonly used symbolic learning systems C4.5 [QU, 1993] and CN2 [CN, 1989].

2 Lattice Definitions

A lattice is defined to be a directed acyclic graph in which any two nodes, u and v, have a unique *join* (a node "higher" in the graph to which a u and v are connected by minimal length paths) and a unique *meet* (a node "lower" in the graph which is connected to u and v by minimal length paths) — referred to respectively as *least upper bounds* and *greatest lower bounds* in formal mathematics. We also define:

Definition 1. The *upward closure* of node u, denoted UC(u), is the set of all nodes, including u, that can be reached from u following upward arcs in the lattice.

Definition 2. The *downward closure* of a node u, denoted DC(u), is the set of all nodes which contain u in their upward closure.

Definition 3. The *cover* of a node u, denoted Cover(u), is the number of instance nodes in DC(u).

Our lattices are defined by representing each training instance by a single node (referred to as an *instance node*) in the "lowest" level of the lattice. In the "greatest" level of the lattice we create a node for every possible feature that an instance in the training set may have (referred to as *feature nodes*). These two sets of nodes uniquely define a set of internal arcs and nodes which comprise the complete lattice. We use the GRAND algorithm [OO, 1988] for lattice construction in our experiments.

3 The Rulearner Algorithm

The Rulearner algorithm takes as input (i) a lattice, L, (ii) a set of instance classification labelings, C, which correspond to the instance nodes in L, and (iii) a noise parameter, N, indicating a percentage by which each induced rule can misclassify some portion of the training instances to which it applies. The algorithm produces a set of symbolic classification rules as output. Furthermore, the user can configure the system to induce either a decision-list or an unordered set of rules, and can also decide whether the rules induced should only classify one (ie. positive) or all given labelings. While the algorithm is general enough to deal with more than two classification labelings, we present the algorithm here as a binary classifier for easier understanding.

The algorithm first labels all instance nodes (whose labelings are given in C) and then filters these labelings up the lattice. A node *u* is given a particular label, say POSITIVE, if the instance nodes in DC(*u*) are all of the class POSITIVE (allowing for some percentage of mislabelings given by the noise parameter N). If there are insufficient instance nodes of any given class in DC(*u*) to give *u* a particular label, then it is labeled MIXED. Each node is initially marked "active," and each time a rule is induced using a node *u*, all the nodes in DC(*u*) are marked "inactive". As long as "active" nodes remain in the lattice, there are still candidate nodes for rule induction and hence new rules are induced. The rule induction process simply finds a non-MIXED "active" node in the lattice which covers the most previously uncovered instance nodes and forms a rule. If several nodes have the same number of instances in their downward closures, we prefer the node which has the fewest features in its upward closure — an application of Occam's Razor. Intuitively, this corresponds to finding a minimal set of features that covers a large portion of the instance space with a given labeling. An important factor in this minimal set of features is that it is directly derived from commonalties in the underlying data and hence the antecedent of the rule induced is directly driven by the data in the training set.

```
PROCEDURE Rulearner(L, C, N)
   Initialize all nodes in the lattice to be "active"
   Label all nodes in the lattice (using the noise parameter N)
   WHILE (there are still "active" nodes in the lattice)
            u ← node with greatest cover and non-MIXED labeling
            Output rule: "feature nodes in UC(u) implies label of node u"
            FORALL (v ∈ DC(u) where v is an instance node)
                     FORALL (w ∈ UC(v))
                              Decrement the cover of w
                              IF (cover of w ≤ 0) mark w "inactive"
                     Mark v as "inactive"
            IF (Decision-List) re-label "active" nodes (using the noise parameter N)
END
```

Fig. 1. Pseudo-code for the Rulearner algorithm.

4 Experimental Results

The Rulearner system was first tested on the Monk's Problems [TH, 1991]. As a comparison, we tried several configuration of other induction systems to capture the *best* performance of those systems compared to Rulearner. For the first two Monk problems, we test three basic configurations of the Rulearner system: (i) decision-lists, (ii) unordered rule sets, and (iii) rule sets to classify only the positive instances, predicting negative as a default rule. The noise parameter, N, was set at 0%.

Algorithm	Monk 1	Monk 2
CN2 (decision-list)	100.0%	72.9%
CN2 (unordered rules)	98.6%	75.7%
C4.5 (unpruned tree)	76.6%	65.3%
C4.5 (pruned tree)	75.7%	65.0%
C4.5 -s (unpruned tree)	94.4%	69.0%
C4.5 -s (pruned tree)	100.0%	70.4%
Rulearner (decision-list)	100.0%	74.5%
Rulearner (unordered rules)	100.0%	74.8%
Rulearner (pos rules only)	100.0%	70.4%

Table 1. Accuracy of induction algorithms on MONK'S Problems 1 and 2.

In Monk 1, all three configurations of the Rulearner system not only performed with 100% accuracy but also learned the minimal concept description for the problem, whereas only the very best configurations of the other learning methods showed similar results. Monk 2 proves challenging to symbolic learning systems since the concept is not easily representable in DNF. Here we find that all the symbolic learning methods hover close to 70% in their accuracy and produce lengthy rule sets, reflective of the checkerboard distribution of classes in the instance space. These experiments show that symbolic learning methods must integrate a bias for more than just functions easily representable in DNF to fare well on a wide range of problems.

Algorithm	Monk 3
CN2 (DL, chi-sq=0.0)	93.3%
CN2 (unord., chi-sq=0.0)	90.7%
CN2 (DL, chi-sq=4.0)	94.4%
CN2 (unord., chi-sq=4.0)	87.5%
C4.5 (unpruned, CF=15%)	92.6%
C4.5 (pruned, CF=15%)	97.2%
C4.5 (unpruned, CF=25%)	92.6%
C4.5 (pruned, CF=25%)	97.2%
Rulearner (DL, N=5%)	94.4%
Rulearner (unord, N=5%)	94.0%
Rulearner (DL, N=10%)	94.4%
Rulearner (unord, N=10%)	95.1%

Table 2. Accuracy on Monk 3.

Algorithm	Breast Can.
Default Rule (majority)	71.7±7.2%
CN2 (DL, chi-sq=4.0)	73.3±6.0%
CN2 (unord., chi-sq=4.0)	71.3±4.5%
CN2 (DL, chi-sq=8.0)	72.1±4.0%
CN2 (unord., chi-sq=8.0)	72.9±5.3%
C4.5 (unpruned, CF=15%)	67.7±9.8%
C4.5 (pruned, CF=15%)	75.2±7.6%
C4.5 (unpruned, CF=25%)	67.7±9.8%
C4.5 (pruned, CF=25%)	73.3±4.5%
Rulearner (DL, N=30%)	74.9±5.8%
Rulearner (unord, N=30%)	74.1±6.4%
Rulearner (DL, N=40%)	74.8±6.6%
Rulearner (unord, N=40%)	73.6±7.8%

Table 3. Accuracy on Breast Cancer.

We also compared Rulearner on noisy domains: Monk 3 which contains 5% noise in the training set and the real world Yugoslavian Breast Cancer data. Again we tried a range of configurations and parameters to optimize the performance of all the

algorithms in our study. In Table 2 we see that all three algorithms are clearly able to learn in the presence of noise. More importantly, however, we see the importance of pruning as reflected in the results from C4.5. Since CN2 and Rulearner currently do no pruning of their induced rules, this could be a promising venue to further increase their accuracy on noisy data, especially since Rulearner outperforms unpruned C4.5.

In the Breast Cancer domain we performed a three-fold cross-validation and report both the accuracy and standard deviation of the results. This is a very difficult problem as none of the algorithms tested perform significantly better than the majority default rule. Here, we again see the importance of pruning as the results with C4.5 clearly indicate. In spite of performing no pruning, the Rulearner system still performs on par with the pruned trees produced by C4.5 and seems to outperform both CN2 and unpruned C4.5. Comparisons of the decision-lists and unordered rule sets produced by CN2 and Rulearner point to no clear winner at this point.

5 Conclusions

It appears that Rulearner is a viable lattice-based induction algorithm. Future work will lead us to examine how rule pruning may be employed to increase accuracy and conduct more comparative studies of the decision-list and unordered rule set paradigms. Methods for automatic noise parameter selection will also be pursued.

The interested reader should also be aware of the systems CHARADE [GA, 1987] and GRAND [OO, 1988] which also make use of lattices to guide the formation of classification rules. These systems, however, differ from ours in their induction mechanisms, biases, and methods for dealing with noise.

Acknowledgments The author thanks Nils Nilsson and Deon Oosthuizen for their thought provoking discussions. Additional thanks go to Oosthuizen for providing both the GRAND lattice construction program and the breast cancer data set, and to Peter Clark for providing CN2. George John and Pat Langley also provided useful insights. The author is supported by a Fred Gellert Foundation ARCS fellowship.

References

[CN, 1989] Clark, P. and Niblett, T. The CN2 Induction Algorithm. *Machine Learning, 3*:261-83, 1989.

[GA, 1987] Ganascia, J.G. CHARADE: A Rule System Learning System. In *Proceedings of the Tenth IJCAI, Volume 1*, pp. 345-347, Milan, Italy, 1987.

[MC, 1969] Michalski, R.S. On the Quasi-minimal Solution of the General Covering Problem. In *Proceedings of the Fifth International Symposium on Information Processing*, pp. 125-128, Bled, Yugoslavia, 1969.

[MI, 1982] Mitchell, T.M. Generalization as Search. *Artif. Intel.*, 18(2): 203-226, 1982.

[NI, 1992] Nilsson, N. J. N-Cube Lattices and Their Role in Machine Learning. *Working paper*, Department of Computer Science, Stanford University, Stanford, CA, 1992.

[OO, 1988] Oosthuizen, G.D. The Use of a Lattice in Knowledge Processing. PhD Thesis, University of Strathclyde, Glasgow, 1988.

[OO, 1994] Oosthuizen, G.D. *The Application of Concept Lattices to Machine Learning*. University of Pretoria Technical Report CSTR 94/01, 1994.

[OM, 1988] Oosthuizen, G.D. and McGregor, D.R. Induction Through Knowledge Base Normalization. *Proceedings of the European Conf. on Artificial Intelligence*, 1988.

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

[RI, 1987] Rivest, R.L. Learning Decision Lists. *Machine Learning, 2*:229-246, 1987.

[TH, 1991] Thrun, S.B. *et al. The MONK'S Problems.* Carnegie-Mellon University Technical Report CMU-CS-91-197, December, 1991.