

# Evaluating Rules Learned from Simulated Environments

William P. Headden\* and Marcus A. Maloof  
Department of Computer Science  
Georgetown University  
Washington, DC 20057-1232  
{headdenw, maloof}@cs.georgetown.edu

## ABSTRACT

We present preliminary results of study in which we used inductive learning to acquire search-control rules for planning. Using a simulator for a vehicle driving domain, we built example traces, executed them in simulation, and produced training examples for learning. Then, using a leave-one-trace-out methodology, we evaluated the J48 and CN2 learning algorithms on the tasks of determining whether the actions of staying in a lane and of changing lanes were valid or invalid. Results suggest that, in the simple scenarios considered, these learning algorithms acquired useful search-control rules, and more importantly, that these rules generalized well to other driving situations.

**KEYWORDS:** *planning, concept learning, search-control rules*

## 1. INTRODUCTION

In this paper, we describe work in progress on the use of inductive learning for acquiring search-control rules for planning. For the domain of vehicle driving, we constructed scenarios, which were configurations of the roadway, and example traces, which were successful and unsuccessful paths down the roadway for the scenarios. Then, using a simulator for the domain, we executed the traces to generate training data for inductive learning. The learning task was to acquire rules for determining if changing lanes and staying in the current lane would be invalid or valid actions.

We evaluated the J48 [1] and the CN2 [2] learning systems on both tasks. Since generalization to unseen traces is most important, we evaluated both methods using a “leave-one-trace-out” methodology. Empirical results suggest that both methods acquired rules that performed well on examples derived from unseen traces.

---

\*Current address: Department of Computer Science, Brown University, Providence, RI 02912, headdenw@cs.brown.edu

## 2. LEARNING SEARCH-CONTROL RULES

*Concept learning* is the task of generating concept descriptions from training data for the purpose of predicting observations not present in the original training set. Although this form of inductive learning is our focus, some researchers have used deductive forms of learning [3], such as explanation-based learning [4], for improving planning (e.g., [5]).

*Planning* is the task of generating a sequence of actions that achieve a goal [6]. To accomplish this task, classic planners start at an initial state and search the plan space for a goal state. The space of plans for non-trivial domains is likely to be large, so search of this space is equally likely to be intractable. Incorporating domain knowledge to control search is one method that not only reduces the complexity of search [5, 7], but also improves plan quality (e.g., the number of steps) [8–10].

Unfortunately, search-control knowledge is often difficult to obtain. It can be challenging to elicit, and intuitive knowledge can produce unintuitive results once incorporated into the planning system. This has prompted some researchers to investigate approaches that use inductive learning to automatically acquire search-control knowledge or search-control rules [5, 7–11]. Based on this research, we have begun investigating the use of inductive learning for acquiring search-control rules for a simple driving domain, as discussed in the next section.

## 3. TASK AND PROBLEM DESCRIPTION

Our driving domain consists of a one-way road with two lanes. The National Institute of Standards and Technology (NIST) built our simulator for this domain that lets cars accelerate, decelerate, and change lanes. By using parameters, users can define passing lanes and place on the roadway penetrable obstacles (e.g., a safety cone) and impenetrable obstacles (e.g., a parked car).

To date, we have considered two tasks:

1. changing lanes with the presence of a no-passing area, and
2. avoiding penetrable and impenetrable obstacles.

For each of these tasks, we designed a set of *scenarios*, which consisted of a particular configuration of the roadway. For example, for the second task, one such scenario was the presence of an impenetrable obstacle in the right lane midway between the start and goal positions.

For each scenario, we generated a set of *example traces* that moved the car from a start position to a goal position. For instance, for the aforementioned scenario, one trace started the car in the right lane, navigated it around the obstacle, and stopped it in the right lane. Another started it in the right lane, navigated it around the obstacle, and stopped it in the left lane. For inductive learning, we needed both positive and negative examples of state changes, so we also formed traces that forced the car to collide with obstacles, to change lanes in a no-passing zone, and to change lanes multiple times.

The execution of each example trace produced a set of training examples. Each training example consisted of a set of Boolean flags indicating various states of the car's current position, of its position if it were to stay in the same lane, and of its position if it were to change lanes. For the car's current position, flags indicated whether the car is traveling too fast, the presence of lane markings and barriers, and whether the road surface is suitable for driving. Similar flags were present for the new positions in the same and new lanes. There were also flags to indicate whether an obstacle is present, whether it is penetrable or impenetrable, and whether it is moving.

For each step of the example trace, the simulator generated a set of these flags along with the label *S* if the car stayed in the same lane and with the label *C* if it changed lanes. The simulator also indicated whether the action was valid (*V*) or invalid (*I*).

The learning task was to acquire two sets of rules: one for determining if staying in the same lane is a valid action, the other for determining if changing lanes is a valid action. Our intent is to use these sets of rules to control search in a planning system for this driving domain.

To simplify the construction of these traces, we took advantage of properties of the domain and the learning methods. We took advantage of the symmetry of the domain. As constructed, the simulator did not differentiate between the left lane and the right lane. There was simply the other lane into which the car could change. This greatly simplified the problem since training examples and control rules could be lane independent. (Unfortunately, this property will not exist in domains with, say, three or more lanes.)

There was also considerable redundancy in the space of examples, which reduced the number of distinct training examples that each trace produced. For instance, a lane free of obstacles in a no-passing zone yields the same training examples for its entire length. Symmetry, as discussed previously, also contributed to this redundancy.

We also took advantage of the ability of the learning methods to generalize the training examples and, in turn, the example traces. Rather than being exhaustive when constructing traces, we attempted to construct traces that would produce examples at the boundaries of concepts [12, 13], for these are critical for discrimination. For instance, we constructed traces just before and just after the onset of a passing zone. Our intent was to minimize the number of traces required for learning, but to maximize the coverage of the induced rules.

After developing the collection of example traces, we evaluated two inductive learners on the tasks of determining if staying in the same lane or changing lanes are valid actions. We selected CN2 [2] and J48, which is an implementation of C4.5 [14] present in Weka [1]. These learning methods have been well studied over the years, have performed well on a variety of tasks, and produce readable concept descriptions, which we deemed important for the initial stages of this project.

CN2 [2] uses as its concept description a set of if-then rules, each of which has a set of conditions as its antecedent and has a class label as its consequent. When classifying an unknown instance, CN2 uses the attribute values of the instance to find a rule with conditions that are all true, in which case, it predicts the class label in the rule's antecedent as that of the instance. To generate such rules, CN2 takes a set of positive and negative training examples, generates a general description for the positive class, and specializes it without "covering" or "intersecting" examples from the negative class. This has been described as *general-to-specific* search for hypotheses, and is in contrast to AQ19 [15], which learns rules through *specific-to-general* search.

J48 [1] uses an  $n$ -ary tree as its concept description, which has internal nodes that correspond to attributes and leaf nodes that correspond to class labels. Each internal node has links for the values or range of values for the attribute. To classify an unknown instance, J48 traverses the tree from the root to a leaf node, a traversal guided by the attribute values present in the instance. It returns the label of the leaf node as the prediction for the instance. To construct a decision tree, J48 finds the attribute that best splits the training examples into the desired classes, creates a node for the attribute and child nodes for each value of the attribute. It distributes the examples to the child nodes based on the values of the selected attribute, which it subsequently removes from further consideration. The algorithm proceeds recursively for each child node until producing

a node with examples of one class, whereby it stores the class label in the leaf node. J48 uses the *gain ratio* as the metric for selecting attributes, and it prunes trees to prevent overtraining [14].

To evaluate these methods, we took the approach of “leaving-one-trace-out.” Most traditional methods of evaluating learning algorithms focus on separating *all* available examples into the training and testing sets (e.g., [16–18]), but in this domain, the performance of a classifier on examples from unseen traces is more important than such performance on randomly selected examples from all traces. One could also investigate the degree to which control rules generalize across scenarios, but we have not yet pursued such an evaluation.

We have used a similar methodology in other work [19], in which we evaluated learning methods using data derived from overhead images. We trained methods on data obtained from a set of images and tested them on data derived from an unseen image. Such an evaluation matches more closely with how image-understanding systems would be used: One would rarely apply learning methods to the images on which they are trained. It is much more likely that they would be applied to newly collected images.

We found this approach useful for our driving domain. Specifically, we began with a collection of example traces and selected one for testing. We generated training examples by running the simulator on all of the traces. After building classifiers with CN2 and J48 using the training examples derived from the traces in the training set, we applied them to the training examples obtained from the single test trace. During performance, we measured accuracy on the testing examples. We repeated this process for each trace in the collection and averaged accuracy over all of the runs, which we present in the next section.

## 4. RESULTS

We present empirical results for two tasks: the lane-change task and the obstacle-avoidance task. The former posed a relatively easy learning problem because of the small number of training examples, so in the next sections, we mostly discuss the results for the latter task.

### 4.1. Lane-change Task

For the lane-change task, we developed 29 example traces, and applied the methodology described in the previous section. This was a simple task that yielded relatively few distinct training examples, because of the redundancy and symmetry of this task. As a result, both learning methods achieved 100% accuracy on the examples derived from the test traces.

### 4.2. Obstacle-avoidance Task

The obstacle-avoidance task, with its more degrees of variation, yielded a more interesting problem for machine learning. To date, we have developed eight different scenarios and have developed 40 traces for five of the eight. For each scenario, we applied the leave-one-trace-out methodology to the example traces. In the following paragraphs, we will present results for Scenario 1, which consisted of a single penetrable obstacle in the left lane.

Some of the rules that CN2 produced were quite intuitive. For example, for the first scenario, CN2 produced the following rules indicating whether changing lanes was a valid action:

```
IF    changeObstaclesClass00bs = F
THEN  class = V  [2 0]

IF    changeObstaclesClass00bs = T
THEN  class = I  [3 1]

(DEFAULT) class = V  [5 1]
```

The attribute `changeObstaclesClass00bs` indicates whether there is a class 0 (i.e., penetrable) obstacle in the opposite lane. The first rule states that it is valid to change lanes if there is no such obstacle in the opposite lane. The second states that it is invalid to change lanes if an obstacle is present. If neither of these rules is true, then the default is to return that an action is valid. The accuracy of this rule set on examples derived from a test trace was 100%. The average accuracy of rule sets across all examples derived from unseen traces was 83%.

The numbers within brackets indicate the number of examples each rule *covers*<sup>1</sup> from the training set for each class. For instance, the first rule covers two examples from the valid class and covers none from the invalid class.

To determine whether staying in the same lane was valid, CN2 produced the following rules:

```
IF    sameRoadLaneMarking = T
THEN  class = V  [42 0]

IF    sameObstaclesClass00bs = F
      AND numberLaneChanges < 0.50
THEN  class = V  [48 1]

IF    sameObstaclesClass00bs = T
      AND numberLaneChanges > 0.50
THEN  class = I  [0 1]
```

<sup>1</sup>A rule covers an example if the example’s attribute values satisfy the rule’s conditions.

```

IF    sameObstaclesClass00bs = T
THEN  class = I  [2 2]

IF    sameRoadLaneMarking = F
      AND numberLaneChanges > 0.50
THEN  class = I  [22 3]

(DEFAULT) class = V  [102 5]

```

These rules are similar to those for predicting whether changing lanes is a valid action. The accuracy of these rules across examples derived from unseen traces was 94%, with most of the accuracy being on the valid class. Notice that the last rule—not the default rule—covers twenty-two valid examples and three invalid examples even though it is a rule for predicting invalid actions.

The distribution of the default rule is the same as the distribution of training set, so notice that there were 102 training examples for the valid class, but only 5 training examples for the invalid class. A classifier that always predicts the majority class would attain an accuracy of roughly 95%, assuming the distribution of the training set is representative of that of the target environment. This skew in the data set presents a few challenges, which we discuss further in the next section.

## 5. DISCUSSION

During our study, we have encountered several challenges. One challenge was developing example traces. We created these manually, which was tedious, but possible only because of the small size of our domain. It is unclear how one might construct such training traces for a high-fidelity, realistic simulation. One possible solution is to have people use a high-fidelity simulator and take their valid and invalid traces as training data.

Another challenge was determining which traces to use for training and which to use for testing. One ideally wants to evaluate the degree to which learned control rules generalize to examples derived from unseen traces. In this study, we trained and tested using examples derived from all example traces using the method of leave-one-trace-out, which accomplished this goal, but it may be better to train on a representative set and test on an exhaustive set of example traces or training examples. This would be an ideal evaluation, especially since learned rules may eventually control a real vehicle, but since it requires an exhaustive set of testing examples, it may be impractical for domains with a large number of potential traces.

We considered constructing training examples directly, rather than deriving them from traces executed by the simulator. Given the current driving domain, since there were fourteen binary attributes describing states, this would have

required labeling  $2^{14} = 16,384$  training examples for each task. Some of these would not correspond to legal states or to states of interest. For instance, we did not consider moving obstacles in this study, so this flag would always be false and thus irrelevant for classification. In the end, we determined that identifying and labeling legal training examples was simply too cumbersome and would be an impractical scheme for real-world driving domains.

As mentioned in Section 4, there was a considerable *skew* in the training sets. That is, there were many more examples of valid actions than of invalid actions. This is intuitive since the roadway was relatively free of obstacles. Consequently, under most circumstances, staying in one lane or changing lanes will be a valid action. Naturally, it is those “other circumstances” with which we are concerned.

The problem with skewed data sets is that most learning algorithms will simply learn to predict the majority class, which was an issue in our work on detecting rooftops in overhead images [19]. Indeed, it is the examples of the minority class—the invalid class—that are most important, but they are underrepresented in the training set.

If we can perform a cost analysis of the driving domain, then methods exist to take such costs into account to form a classifier of minimal cost or of minimal risk [20, 21]. Such analyses are often difficult to obtain, but we can use receiver operating characteristic (ROC) analysis [22] to evaluate learning methods over all possible costs [17, 19, 23, 24], provided that a domain expert can identify an acceptable operating point.

Minimum-risk classifiers and ROC analysis assume that the cost of error on instances of a class are uniform, and unfortunately, for the driving domain, this is not the case. For instance, even though driving over a traffic cone and driving over—or attempting to drive over—a parked car are both invalid actions, the cost of driving over the cone is less than that of driving over the car. Boosting [25, 26] is an ensemble method that weights individual examples of a training set and produces predictions consistent with those weights. It may be applicable to this problem, but we are still left with the problem of developing a scheme for assigning weights to individual examples.

A final concern of ours is that of optimal local behavior versus optimal global behavior. As we have described, we are using learning algorithms to acquire search-control rules. These are a local representation of what to do in each state. However, it is not yet clear if these local rules will lead to better global performance of the planner. Thus our next task is to incorporate the learned rules into the planning system and continue our empirical inquiry.

## 6. CONCLUSION

In this paper, we have examined an application of inductive learning to the problem of learning search-control knowledge. We evaluated two learning methods, J48 and CN2, for the task of determining if staying in a lane is a valid action and if changing lanes is a valid action. Empirical results imply that such methods acquired useful control rules and that these rules generalized to unseen driving situations. We plan to expand our efforts by incorporating learned rules into the target planning system, obtaining a more sophisticated simulator, and evaluating the algorithms over a range of misclassification costs using ROC analysis.

## Acknowledgements

The authors thank Steve Balakirsky for providing advice and the simulator used in this study, and Clay Shields for supplying some of the equipment necessary for this project. This research was conducted in the Department of Computer Science at Georgetown University. The work was supported by the department and by the National Institute of Standards and Technology under grants 60NANB2D0013 and 60NANB3D1062.

## 7. REFERENCES

- [1] Witten, I., and Frank, E., *Data mining: Practical machine learning tools and techniques with Java implementations*. San Francisco, CA: Morgan Kaufmann, 2000.
- [2] Clark, P., and Niblett, T., "The CN2 induction algorithm," *Machine Learning*, Vol. 3, pp. 261–284, 1989.
- [3] Michalski, R.S., "Inferential Theory of Learning: Developing foundations for multistrategy learning," in *Machine Learning: A Multistrategy Approach*, Michalski, R.S., and Tecuci, G., Eds. San Francisco, CA: Morgan Kaufmann, 1994, Vol. 4, pp. 3–61.
- [4] Mitchell, T., Keller, R., and Kedar-Cabelli, S., "Explanation-based generalization: A unifying view," *Machine Learning*, Vol. 1, pp. 47–80, 1986.
- [5] Minton, S., "Explanation-based learning: A problem solving perspective," *Artificial Intelligence*, Vol. 40, pp. 63–118, 1989.
- [6] Russell S., and Norvig, P., *Artificial intelligence: A modern approach*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 2003.
- [7] Estlin, T., and Mooney, R., "Multi-strategy learning of search control for partial-order planning," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann, 1996, pp. 770–777.
- [8] Pérez, M., "Representing and learning quality-improving search control knowledge," in *Proceedings of the Thirteenth International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1996, pp. 382–390.
- [9] Estlin, T., and Mooney, R., "Learning to improve both efficiency and quality for planning," in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann, 1997, pp. 770–777.
- [10] Upal, M., and Elio, R., "Learning rewrite rules versus search control rules to improve plan quality," in *Proceedings of the Canadian Conference on Artificial Intelligence*, 2000, pp. 240–253.
- [11] Langley, P., "Learning to search: From weak methods to domain-specific heuristics," *Cognitive Science*, Vol. 9, pp. 217–260, 1985.
- [12] Maloof, M.A., and Michalski, R.S., "Selecting examples for partial memory learning," *Machine Learning*, Vol. 41, pp. 27–52, 2000.
- [13] Maloof, M.A., and Michalski, R.S., "Incremental learning with partial instance memory," *Artificial Intelligence*, to appear.
- [14] Quinlan, J.R., *C4.5: Programs for machine learning*. San Francisco, CA: Morgan Kaufmann, 1993.
- [15] Michalski, R.S., and Kaufman, K., "The AQ-19 system for machine learning and pattern discovery: A general description and user's guide," Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA, Reports of the Machine Learning and Inference Laboratory MLI 01-4, 2001.
- [16] Weiss, S., and Kulikowski, C., *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning and expert systems*. San Francisco, CA: Morgan Kaufmann, 1992.
- [17] Bradley, A., "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, Vol. 30, pp. 1145–1159, 1997.
- [18] Provost, F., and Fawcett, T., "Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions," in *Proceedings*

of the Third International Conference on Knowledge Discovery and Data Mining. Menlo Park, CA: AAAI Press, 1997, pp. 43–48.

- [19] Maloof, M.A., Langley, P., Binford, T.O., Nevatia, R., and Sage, S., “Improved rooftop detection in aerial images with machine learning,” *Machine Learning*, Vol. 53, pp. 157–191, 2003, to appear, <http://www.kluweronline.com/issn/0885-6125>.
- [20] Duda, R., Hart, P., and Stork, D., *Pattern classification*, 2nd ed. New York, NY: John Wiley & Sons, 2000.
- [21] Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., and Brunk, C., “Reducing misclassification costs,” in *Proceedings of the Eleventh International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1994, pp. 217–225.
- [22] Swets, J., and Pickett, R., *Evaluation of diagnostic systems: Methods from signal detection theory*. New York, NY: Academic Press, 1982.
- [23] Maloof, M.A., Langley, P., Sage, S., and Binford, T.O., “Learning to detect rooftops in aerial images,” in *Proceedings of the Image Understanding Workshop*. San Francisco, CA: Morgan Kaufmann, 1997, pp. 835–845.
- [24] Provost, F., Fawcett, T., and Kohavi, R., “The case against accuracy estimation for comparing induction algorithms,” in *Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1998, pp. 445–453.
- [25] Schapire, R., “A brief introduction to boosting,” in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann, 1999.
- [26] Joshi, M., Kumar, V., and Agarwal, R., “Evaluating boosting algorithms to classify rare classes: Comparison and improvements,” in *Proceedings of the First IEEE International Conference on Data Mining*. Los Alamitos, CA: IEEE Press, 2001, pp. 257–264.