University of Wisconsin Machine Learning Research Group Working Paper 99-1

Rule Extraction: Where Do We Go from Here?

Mark Craven
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania, 15213-3891
U.S.A.
mark.craven@cs.cmu.edu

Jude Shavlik
Department of Computer Sciences
University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin, 53706-1685
U.S.A.
shavlik@cs.wisc.edu

Abstract

We argue that despite being an actively researched area for nearly a decade, rule-extraction technology has not made as significant of an impact as it should have. A confluence of trends, however, has made the ability to extract comprehensible descriptions from complex learned models more important now than ever. We argue that rule-extraction methods can have a significant impact in the overlapping data-mining, machine-learning and neural-network communities if research is focused on several commonly overlooked issues. We then briefly describe how we have tried to address these issues in our own work.

Introduction

For nearly a decade, researchers have been investigating the task of converting learned neural-network models into more easily understood representations (Andrews, Diederich, & Tickle 1995). This type of work is commonly referred to as *rule extraction* since the representation language used to describe learned neural-net models by these methods is typically some form of propositional inference rules like those used in many "symbolic" AI systems. Although the problem of understanding trained neural networks is widely considered an important area of research, the rule-extraction community has had relatively little impact in the fields of data mining, machine learning, and neural networks. In this paper, we argue that there is great opportunity for the rule-extraction community to have a significant impact on machine-learning research and practice, but in order for the community to have this impact it will have to refocus its research effort on some widely neglected issues. We discuss these issues in detail, argue why they are important, and where applicable, describe how we have addressed them in our own rule-extraction work (Craven & Shavlik 1996; Craven 1996; Craven & Shavlik 1997)

We argue that the great opportunity that currently exists for the rule-extraction community derives from two notable trends. The first is the data-mining boom (Fayyad & Uthurusamy 1996). Increasingly, companies operating in every sector and in every part of the world are interested in exploiting the useful information embedded in their databases. Similarly, many scientific disciplines are becoming increasingly data-driven, and thus there is great interest in eliciting new knowledge from scientific databases. Two paramount criteria for most data-mining applications are that learned models have good predictive accuracy and that they be comprehensible.

The second trend of note is the recent interest in *ensemble* methods, such as *bagging* (Breiman 1996), *boosting* (Freund & Schapire 1997), and *error-correcting output codes* (Dietterich & Bakiri 1995). These methods learn models that are composed of multiple constituent models. They have generated a lot of attention and excitement because they often significantly improve the predictive accuracy of learned models, and there are solid theoretical underpinnings explaining why this is the case. A downside of these ensemble methods, however, is that they usually produce models that are very complex and difficult to understand.

The confluence of these two trends means that for many applications, (i) the most accurate models tend to be complex and incomprehensible, but (ii) the comprehensibility of learned models is of growing importance. This situation presents a unique opportunity for the rule-extraction community since the researchers in this community have a large amount of shared expertise in extracting comprehensible descriptions from complex models. Unfortunately, however, most current rule-extraction algorithms are not able to fulfill this challenge. In the next section, we discuss several shortfalls common to most rule-extraction algorithms, and recommend several lines of research to address these weaknesses.

Overlooked Issues

In our own research in rule extraction, we have evaluated our algorithms along several dimensions:

- *Comprehensibility:* The extent to which extracted representations are humanly comprehensible.
- *Fidelity:* The extent to which extracted representations accurately model the networks from which they were extracted.

- *Accuracy:* The ability of extracted representations to make accurate predictions on previously unseen cases.
- *Scalability:* The ability of the method to scale to networks with large input spaces and large numbers of units and weighted connections.
- *Generality:* The extent to which the method requires special training regimes or restrictions on network architecture.

Most other researchers in the field have also evaluated their methods using the first three criteria, but *scalability* and *generality* have been the focus of much less attention. In this section we argue that the impact of rule-extraction methods will hinge on how well these two issues are addressed in the near future. Additionally, we consider one other important issue, *software availability*, that we believe is key to the success of rule-extraction methods.

Scalability

Obviously, rule-extraction technology will have a greater impact if the available methods can be applied to a wide range of learned models in a wide range of application domains. In part, this means being able to apply rule-extraction methods to problems that are, in some sense "big." Thus, we would like to have rule-extraction algorithms that are scalable. In particular, we are interested in the following definition:

> *Scalability* refers to how the running time of a rule-extraction algorithm and the comprehensibility of its extracted models vary as a function of such factors as network, feature-set and training-set size.

Our definition departs from what most researchers in the community think of as scalability in that it includes the *comprehensibility of extracted models* as well as the running time of the method. Since comprehensibility is of fundamental importance in rule extraction, however, methods that scale well in terms of running time, but not in terms of comprehensibility will be of little value. In our experience, developing an algorithm that scales well in terms of comprehensibility is at least as hard as developing one that scales well in terms of running time. Whereas the latter issue has been the focus of much attention (e.g., Alexander & Mozer, 1995), the former has not.

Scaling in terms of comprehensibility is a hard problem for several reasons. *Decompositional* rule-extraction methods (Andrews, Diederich, & Tickle 1995), which extract rules by analyzing individual components of networks (e.g., hidden and output units), have a built-in tendency to produce rule sets whose size is proportional to the network size. In cases where the problem domain has a large number of features, however, even non-decompositional methods tend to produce large rule sets since "big" networks often represent complicated functions.

To address the issue of scalability in comprehensibility, we recommend that rule-extraction researchers pursue several lines of research that have not been adequately explored:

- *Methods for controlling the comprehensibility/fidelity trade-off.* One approach to extracting comprehensible rule sets from large, complex neural networks is to improve the comprehensibility of an extracted rule set by compromising on its fidelity to the network from which it was extracted. For example, our TREPAN algorithm (Craven & Shavlik 1996; Craven 1996) which extracts decision-tree descriptions of trained neural nets, incorporates a "knob" that allows the user to control the size of the tree returned. This knob allows the user to directly control the trade-off between comprehensibility and fidelity. We consider this particular mechanism to be only a first step in the direction of exploring such comprehensibility/fidelity knobs.
- *Methods for anytime rule extraction.* We conjecture that the comprehensibility of an extracted rule set can often be improved by investing more computation in exploring alternative representations of the rule set. For example, a given rule set may be re-expressed using a variety of truth-preserving transformations such as inventing new terms corresponding to frequently used conjunctions. This kind of concept transformation has been investigated in the *constructive induction* literature (Muggleton 1987). In general we argue that a productive line of research would be to develop *anytime* (Dean & Boddy 1988) rule-extraction algorithms. Such algorithms could be interrupted at any point in their computation and still provide a solution (i.e., an extracted rule set), but given more time they would generally find better (i.e., more comprehensible) solutions.

Generality

Most of the rule-extraction algorithms that have been developed to date can be applied only to networks within a narrow architectural class, or to networks that have been trained using a special training regime. We argue that, in order to have a large impact, rule-extraction methods will have to exhibit a high level of generality. That is, they should be applicable to neural networks that employ a wide array of topologies, transfer functions, input and output encodings, and training methods. In short, they should be applicable to networks developed by others without any initial intention of applying rule-extraction methods to them.

Taking this argument one step further, we argue that rule-extraction methods should be so general that they do not assume that the models they are trying to describe are even neural networks. As stated in the Introduction, with the rising popularity of ensembles, there are now many learned models which are not neural networks but which call for rule-extraction-type algorithms to elicit comprehensible descriptions of them. Indeed, several researchers have already recognized this need and have developed initial methods for translating ensemble models into more comprehensible decision trees (Breiman & Shang 1996) and rule sets (Domingos 1998).

In summary, we make the following recommendations to rule-extraction researchers:

- *Seek out collaborators who already have models they want to understand, and apply rule-extraction methods to these models.* Pursuing this task will entail first developing rule-extraction methods that are general enough that they can be applied to a wide variety of learned models.
- *Develop extraction methods that are applicable to ensembles and other hard-to-understand models.* The prevalence of learned ensemble models is rapidly increasing. The need for methods able to provide comprehensible descriptions of these models is growing accordingly.

Software Availability

Finally, another factor that we believe will be significant in determining the impact of rule-extraction methods is the extent to which researchers make their methods available to potential users. One way to make such a method available is to provide portable source code to users who can download, install and run it on their own machines. Quinlan's C4.5 (Quinlan 1993) and FOIL (Quinlan 1990) programs serve as good models here. Both algorithms have gained widespread prominence in the machine-learning community, in part because robust, portable C code is available to those who want to use the programs. Making source code available has the added advantage that users can easily investigate algorithm variants and develop new functionality.

Another model for making rule-extraction methods readily available is to set them up as on-line servers. Providing this type of service places less of a burden on users, since they do not have to download and install code, but it imposes a possibly significant computational cost on the service provider. Good models for this type of method availability can be found throughout the computational biology community where heavily used, and highly visible, servers perform such functions as DNA sequence analysis (Xu *et al.* 1996) and protein structure prediction (Rost 1996). These servers have had a large impact in the field of molecular biology because they provide valuable services, but are easy to use.

A Case Study in Rule Extraction

In this section we give a brief overview of an algorithm we have developed, TREPAN (Craven & Shavlik 1996; Craven 1996), and discuss how it addresses the issues presented in the previous section. We also describe an application of TREPAN that illustrates these points.

The TREPAN Algorithm

TREPAN takes a trained neural network and a set of training data as inputs. As output, it produces a decision tree that provides a close approximation to the function represented by the network. The task of TREPAN then, is to induce the function represented by the trained network. In many respects, TREPAN is similar to conventional decision-tree algorithms, such as CART (Breiman *et al.* 1984) and C4.5 (Quinlan 1993), which induce trees directly from training data. TREPAN's learning task differs in several key respects, however. First, the target concept to be learned by TREPAN is the function represented by the network. This means that TREPAN uses the network to label (i.e., get the predicted output value of) all instances. Second, because TREPAN can use the network to label instances, it learns not just from a fixed set of training data, but from arbitrarily large samples.

```
TREPAN
Input: trained network, training set used for network

    initialize the tree as a leaf node
    while stopping criteria not met
        pick the most promising leaf node to expand
            draw a sample of instances
            use the network to label the instances
            select a splitting test for the node
            for each possible outcome of the test
                make a new leaf node

Return: extracted decision tree
```

Figure 1: *The* TREPAN *algorithm.*

Figure 1 provides a sketch of the TREPAN algorithm. The basic idea of the method is to progressively refine an extracted description of a neural net by incrementally adding nodes to a decision tree that characterizes the network. Initially, TREPAN's description of the network is a single leaf node that predicts the class that the network itself predicts most often. This crude description of the network is refined by iteratively selecting a leaf node of the tree to expand into an internal node with leaves as children.

In order to decide which node to expand next, TREPAN uses an evaluation function to rank all of the leaves in the current tree and then picks the best one. The notion of the best node, in this case, is the one at which there is the greatest potential to increase the fidelity of the extracted tree to the network. The function used to evaluate node N is:

$$f(N) = reach(N) \times (1 - fidelity(N))$$

where $reach(N)$ is the estimated fraction of instances that reach N when passed through the tree, and $fidelity(N)$ is the estimated fidelity of the tree to the network for those instances. The motivation for expanding an extracted tree in this best-first manner is that it gives the user a fine degree of control over the size of the tree to be returned: a tree of arbitrary size (in terms of the number of internal nodes) can be selected to describe a given network.

Expanding a node in the tree involves two key tasks:

determining a logical test with which to partition the instances that reach the node, and determining the class labels for the leaves that are children of the newly expanded node. In order to make these decisions, TREPAN ensures that it has a reasonably large sample of instances. It gets these instances from two sources. First, it uses the network's training examples that reach the node. Second, TREPAN constructs a model (using the training examples) of the underlying distribution of data in the domain, and uses this model in a generative manner to draw instances. These instances are randomly drawn but are subject to the constraint that they would reach the node being expanded if they were classified by the tree. In both cases, TREPAN queries the neural network to get the class labels for the instances.

Selecting a test for an internal node in a decision tree involves deciding how to partition the part of the input space covered by the internal node. In order to select a splitting test from a set of candidates, TREPAN uses *information gain* as an evaluation measure.

An Application of TREPAN

TREPAN has been used to extract trees from networks trained in a wide variety of problem domains, including several in which the neural-networks were developed by others: elevator control (Crites & Barto 1996; Crites 1996), exchange-rate prediction (Weigend, Zimmermann, & Neuneier 1995) and climate modeling (Trimble, Santee, & Neidrauer 1997). In all of these cases, the networks were developed without any prior intention of applying rule-extraction methods to them. Here we briefly describe the application of TREPAN to Crites and Barto's elevator-control network. This network represented one of the first practical successes of reinforcement learning.

The elevator-control network operates in a simulated 10-story building with four elevator cars. The state of the system that is presented to this neural network includes information about such things as which hall buttons were pushed at what times, and the locations and speeds of the cars in the system. This information is encoded using 19 real-valued and 12 discrete-valued features. The network has 47 input units, 20 hidden units with logistic transfer functions, and two output units with linear transfer functions. The elevator car has a default control policy that specifies constraints on what actions it can take in various situations. The task for the reinforcement learner is to decide, given the current measurable state, whether a car should stop or continue on its current path. It does this by predicting two real-valued numbers representing the expected utility of taking the two possible actions.

Figure 2 shows the tree extracted by TREPAN from the elevator-control network. This tree consists of only four internal nodes and a total of 14 feature references. Since four of the five leaves of the tree predict the same action, that the tree can easily be simplified into a single rule. Note that two of the nodes in the tree have disjunctive tests; TREPAN can use m-of-n tests in trees as well.

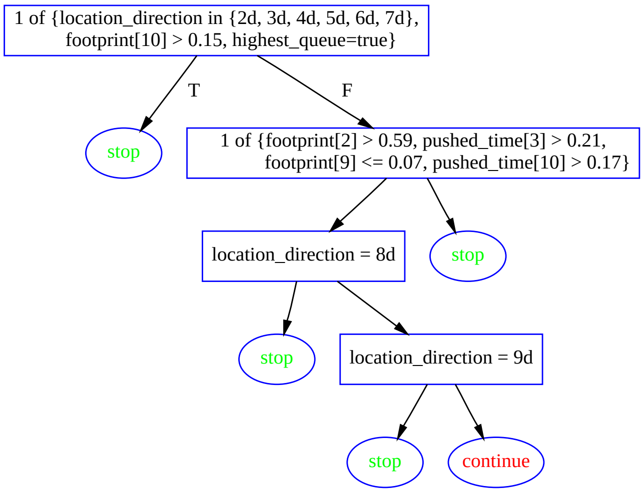

Figure 2: The tree extracted by TREPAN from the elevator-control network.

Table 1: TREPAN test-set fidelity for the elevator domain.

traffic profile	fidelity (%)
down traffic only	89.4
down and up traffic	90.8
down and 2× up traffic	91.9

Table 1 shows the test-set fidelity measurements for the tree extracted by TREPAN. Each line in the table represents the fidelity of the tree to the network when measured under a different traffic distribution in the building (i.e., different test sets). The results in this table indicate that the action selected by the tree agrees with the network in approximately 90% of encountered states, and furthermore this level of fidelity is fairly consistent across states that come from different types of traffic conditions. Nearly all of the cases where the TREPAN-extracted tree disagrees with the network are cases where the network predicts that the expected utility of taking the two actions is about the same.

To evaluate the predicted accuracy of our extracted tree, we used it as a policy to control the elevator simulator under the same conditions that Crites and Barto used to evaluate their networks. Table 2 shows the performance of the TREPAN-extracted tree, and the performance of the network and several conventional controllers. For all three performance measures, lower numbers are better. From this table, one can see that the performance of the extracted tree is nearly identical to the neural network For all three traffic profiles, the network and the extracted tree outperform the conventional control strategies considered.

Table 2: **Performance of Various Elevator-Control Policies.** Results are shown for three different performance measures on one of the testing traffic profiles. The first seven rows represent conventional elevator scheduling algorithms.

algorithm	Wait	Wait^2	System Time
SECTOR	30.3	1643	59.5
DLB	22.6	880	55.8
BASIC HUFF	23.2	875	54.7
LQF	23.5	877	53.5
HUFF	22.8	884	55.3
FIM	20.8	685	53.4
ESA	20.1	667	52.3
network	**18.7**	**582**	**45.8**
TREPAN	**18.6**	**577**	**45.8**

Discussion

We have argued that there is a great opportunity for rule-extraction research to have a large impact in the data-mining, machine-learning, and neural-network communities given the increasing prevalence of complex models and the increasing importance of comprehensible models. However, to meet this opportunity, rule-extraction research should greatly increase its emphasis on the issues of scalability and generality. Furthermore the community should make greater efforts to make its software products publicly available, either by distributing source code or by setting up on-line servers.

Our own research has led to the development of the TREPAN algorithm, which was designed, in part, to address the generality and scalability issues. TREPAN is quite general in that it it makes few assumptions about the architecture of a given network, and it does not require a special training method for the network. In fact, TREPAN does not even require that the model be a neural network. TREPAN can be applied to a wide variety of hard-to-understand models including ensembles.

TREPAN is also scalable in terms of running time and in terms of the comprehensibility of its learned models. Regarding the first scalability criterion, the computational complexity of expanding a decision tree node in TREPAN is a low-order polynomial in the sample size, the number of features and the maximum number of values for a discrete feature. It does not depend in any significant way on any other parameters of the given model. Regarding the second scalability criterion, TREPAN gives the user fine control over the complexity of the descriptions it returns. This capability derives from the fact that TREPAN represents its extracted models using decision trees, and it expands these trees in a best-first manner. TREPAN first extracts a very simple (i.e., one-node) description of a trained network, and then successively refines this description to improve its fidelity to the network. In this way, TREPAN explores increasingly more complex, but higher fidelity, descriptions of the given network.

The source code for TREPAN is freely available, and can be retrieved from:
`http://www.cs.cmu.edu/~craven/trepan.sh`.

We consider TREPAN to be only a first step towards the goal of general, scalable, and widely available rule-extraction methods. We encourage other researchers in the community to direct their efforts toward this same goal.

References

Alexander, J. A., and Mozer, M. C. 1995. Template-based algorithms for connectionist rule extraction. In Tesauro, G.; Touretzky, D.; and Leen, T., eds., *Advances in Neural Information Processing Systems*, volume 7. Cambridge, MA: MIT Press.

Andrews, R.; Diederich, J.; and Tickle, A. B. 1995. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems* 8(6).

Breiman, L., and Shang, N. 1996. Born again trees. Technical report, Department of Statistics, University of California, Berkeley. ftp://ftp.stat.berkeley.edu/pub/users/breiman/BAtrees.ps.

Breiman, L.; Friedman, J.; Olshen, R.; and Stone, C. 1984. *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Breiman, L. 1996. Bagging predictors. *Machine Learning* 24:123–140.

Craven, M. W., and Shavlik, J. W. 1996. Extracting tree-structured representations of trained networks. In Touretzky, D.; Mozer, M.; and Hasselmo, M., eds., *Advances in Neural Information Processing Systems*, volume 8. Cambridge, MA: MIT Press. 24–30.

Craven, M. W., and Shavlik, J. W. 1997. Understanding time-series networks: A case study in rule extraction. *International Journal of Neural Systems* 8(4):373–384.

Craven, M. W. 1996. *Extracting Comprehensible Models from Trained Neural Networks*. Ph.D. Dissertation, Computer Sciences Department, University of Wisconsin, Madison, WI.

Crites, R. H., and Barto, A. G. 1996. Improving elevator performance using reinforcement learning. In Touretzky, D.; Mozer, M.; and Hasselmo, M., eds., *Advances in Neural Information Processing Systems*, volume 8. Cambridge, MA: MIT Press. 1017–1023.

Crites, R. H. 1996. *Large-Scale Dynamic Optimization Using Teams of Reinforcement Learning Agents*. Ph.D. Dissertation, Computer Science Department, University of Massachusetts Amherst, Amherst, MA.

Dean, T., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 49–54. St. Paul, MN: Morgan Kaufmann.

Dietterich, T. G., and Bakiri, G. 1995. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2:263–286.

Domingos, P. 1998. Knowledge discovery via multiple models. *Intelligent Data Analysis* 2(3).

Fayyad, U., and Uthurusamy, R. 1996. Data mining and knowledge discovery in databases. *Communications of the ACM* 39(11):24–26.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and its application to boosting. *Journal of Computer and System Sciences* 55(1):119–139.

Muggleton, S. 1987. Duce, an oracle based approach to constructive induction. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 287–292.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5:239–2666.

Quinlan, J. 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Rost, B. 1996. PHD: Predicting one-dimensional protein structure by profile based neural networks. *Methods in Enzymology* 266:525–539.

Trimble, P. J.; Santee, E. R.; and Neidrauer, C. J. 1997. Including the effects of solar activity for more efficient water management: An application of neural networks. In *Proceedings of the AI Applications in Solar-Terrestrial Physics Workshop*. Lund, Sweden: European Space Agency.

Weigend, A. S.; Zimmermann, H. G.; and Neuneier, R. 1995. Clearning. Technical Report CU-CS-772-95, Computer Science Department, University of Colorado.

Xu, Y.; Mural, R. J.; Einstein, J. R.; Shah, M. B.; and Uberbacher, E. C. 1996. GRAIL: A multi-agent neural network system for gene identification. *Proceedings of the IEEE* 84(10):1544–1552.