Overcoming Process Delays with Decision Tree Induction

Bob Evans, R R Donnelley & Sons Company
Doug Fisher, Vanderbilt University

BY USING DECISION TREE INDUCTION TO DERIVE PROCESS CONTROL RULES, THIS SYSTEM LETS EXPERTS PARTICIPATE IN KNOWLEDGE ACQUISITION BY DOING WHAT THEY DO BEST: EXERCISING THEIR EXPERTISE.

PRINTERS ARE ALWAYS SEEKING higher productivity by increasing their production rates and minimizing process delays. Although higher productivity can be achieved by designing equipment that runs faster and requires fewer operators, acquiring such new equipment entails ever increasing capital outlays. As a result, printing managers are also interested in reducing delay time on existing equipment.

When process delays have known causes, they can be mitigated by acquiring causal rules from human experts and then applying sensors and automated real-time diagnostic devices to the process. However, for some delays the experts have only weak causal knowledge or none at all. In such cases, machine learning tools can collect training data and process it through an induction engine in search of diagnostic knowledge. We recently applied a machine learning strategy known as *decision tree induction*[1] to derive a set of rules about a long-standing problem in rotogravure printing. Our induction mechanism is embedded within a knowledge acquisition system that suggests plausible rules to an expert, who can override the rules or modify the data from which the rules were derived.

Rotogravure printing

Rotogravure printing involves rotating a chrome-plated, engraved copper cylinder in a bath of ink, scraping off the excess ink, and pressing a continuous supply of paper against the inked image with a rubber roller. Once the job is printed, the engraved image is removed from the cylinder, which is replated to be engraved for another job.

Sometimes a series of grooves — called a *band* — appears in the cylinder during printing, ruining the finished product. These grooves are not present at the start of the printing run, but once they appear the printing press must be shut down. A technician then removes the band by polishing it out of the cylinder, or by transporting the cylinder to a plating station where the chrome surface is removed, the band is polished out of the copper subsurface, and a chrome finish is replated. This process results in the press being shut down for an average of about 1.5 hours (at least 30 minutes, and as much as six hours). During this time, the wages of 3 to 10 crew members (depending on the piece being printed) must be charged to a delay time account. Because printing is time critical, these delays usually must be offset by scheduling crews on weekends at an unrecoverable overtime premium. In extreme cases, a new and very costly printing cylinder must be manufactured.

While the rotogravure process has been in commercial use since the turn of the century, cylinder banding has been seen regularly only since process speeds broke the 1,000 feet/minute barrier in the late 1950s. In 1989, banding became more than just a nuisance — it became a measurable component of the operating expenses for one R R Donnelley & Sons plant, which organized a task force to control the problem. Many process features were thought to contribute to banding, but there was no documentation conclusively associating them with the problem. In past efforts the banding just went away, the printers were left with a vague sense that "X may have had something to do with the bands," and some time later the banding returned. Thus, Bob Evans considered using a machine learning tool to extract diagnostic rules from observations of printing operations in which banding did and did not occur.

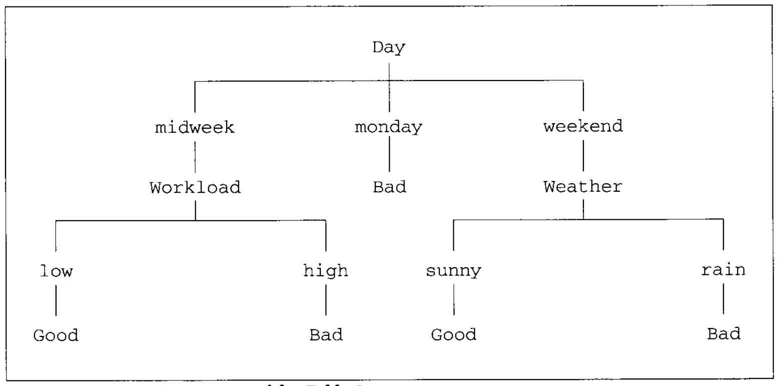


Figure 1. Decision tree generated for Table 1.

Table 1. A training set.

	Attributes			Class
	Day	Workload	Weather	
1	Midweek	Low	Rain	Good
2	Midweek	Low	Sunny	Good
3	Monday	Low	Rain	Bad
4	Weekend	Low	Rain	Bad
5	Weekend	High	Rain	Bad
6	Weekend	High	Sunny	Good
7	Monday	High	Sunny	Bad
8	Midweek	High	Rain	Bad
9	Midweek	High	Sunny	Bad
10	Monday	Low	Sunny	Bad
11	Monday	High	Rain	Bad
12	Weekend	Low	Sunny	Good

Machine learning for knowledge engineering

Traditional knowledge engineering involves extracting and refining rules directly from human experts. However, experts in some domains do not base their decisions on precise rules. Even when they do, it can be difficult to extract the rules directly: Experts are best at executing rules during decision making; they are not necessarily good at explaining those rules. Cognitive studies of knowledge compilation suggest that experts have compiled their knowledge into procedural form, thus rendering it efficient to execute but difficult to inspect and describe.[2] Directly extracting such compiled rules is cumbersome at best[3] and has motivated a search for other ways to ease knowledge engineering.

Machine learning is one alternative that can extract decision-making rules from data when human expertise is nonexistent, insufficiently refined, or in a compiled, procedural form. For example, by examining a medical database of patient case histories — including features such as gender, age, and a variety of medical tests, as well as expert diagnoses of each patient's malady — a learning program can construct a set of rules for diagnosing subsequent patients.[1]

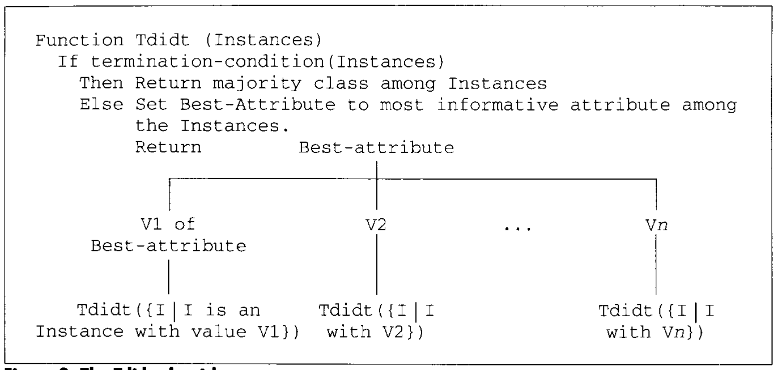
Machine learning systems perform well on some data sets in part because an appropriate data definition has evolved, as in the medical database. However, we cannot expect acceptable results if we naively apply machine learning to arbitrary data. In a new domain, machine learning must be incorporated into a process of *interactive induction*,[4,5] in which an expert collects and engineers the data, a machine generates rules consistent with the data, and an analyst acts as an intermediary, counseling the expert on the plausibility of the rules, and judging when an expert's knowledge confirms or overrides those rules. The expert or analyst may accept, reject, or modify the system-generated rules, but more often he or she suggests alternative features to be measured. The rule set is refined through subsequent data collection, rule induction, and expert consideration.

This process transforms knowledge engineering into a task that better fits human expert abilities: Experts can identify relevant features and supply examples of their decision-making behavior, but they need not explicate the precise interactions between features that define their rules; this task is left to a machine, which is well-suited to exploring plausible feature combinations in search of those that best fit expert behavior.[6] This premise — that data engineering is easier than direct rule engineering — lies behind research into semi-automatic knowledge acquisition.[7]

Top-down induction of decision trees

For interactive induction to be successful, the expert must be able to comprehend the output from the machine learning component. Analysts in many engineering disciplines use decision trees to format diagnostic and control knowledge, so we chose *decision tree induction* as our strategy for learning diagnostic rules from categorized data. (Decision trees also have advantages over other forms of rule-based knowledge, and still other advantages in terms of comprehensibility over various neural network and statistical strategies.[8])

For example, the ID3 induction algorithm[9] constructs the tree in Figure 1 for the data in Table 1. We categorize new



```
Function Tdidt (Instances)
  If termination-condition(Instances)
    Then Return majority class among Instances
    Else Set Best-Attribute to most informative attribute among
         the Instances.
         Return        Best-attribute

         V1 of                  V2          ...          Vn
    Best-attribute

  Tdidt({I | I is an       Tdidt({I | I              Tdidt({I | I
Instance with value V1})    with V2})                 with Vn})
```

Figure 2. The Tdidt algorithm.

observations by examining this tree from the top down. Based on its value for the attribute at the top (root) of the tree, an observation is classified into one of three subcategories at the next lower level. In this case, if the new observation has a value of `midweek` for `Day`, then categorization continues to the left. The attribute at the root of the appropriate subtree then guides further categorization. If value `high` is observed for `Workload`, then categorization moves to the right subtree. This process continues until a terminal node or leaf is reached; the category that labels the leaf is predicted as the category of the new observation. In this example, the predicted category is `Bad`.

Each path from root to leaf is a conjunctive rule that specifies a categorization. For example, a rule corresponding to the path we just followed through the tree states that

```
(Day=midweek)∧(Workload=high)
→ Bad
```

By structuring such rules in a tree, we explicitly represent the interactions between rules in a way that human analysts can understand. The tree also implies an ordering on conditions: In cases where tests must be made, their order is given as we follow a path from root to leaf.

Figure 2 shows our generic algorithm for the top-down induction of decision trees. The algorithm — called Tdidt — first tests a data set to see whether tree construction is worthwhile. If all the data are classified identically or some other statistical criterion is satisfied, then there is no need to proceed; Tdidt simply returns a leaf that is labeled by the most frequent class found among the data at this leaf. If the data are not sufficiently distinguished, Tdidt selects the "best" divisive attribute, uses its values to partition the data into subsets, and then recursively calls on these subsets to expand the tree. In the tree of Figure 1, `Day` was selected as the divisive attribute at the root, and `Workload` and `Weather` were selected to expand the leftmost and rightmost subpopulations. All observations with `(Day=monday)` were categorized as members of the `Bad` class, so no further decomposition was necessary. This decision tree includes all the attributes that describe the sample data, but this is usually not the case; typically, many attributes are irrelevant to category membership or otherwise not needed.

Tdidt's most important feature is the criterion it uses to choose divisive attributes. If it randomly selected them, the resulting tree would likely be too complex, and many attributes might be of little relevance in discriminating among the known classes. Instead, we want attributes that are the most relevant to or most informative of category membership. In an ideal attribute, each value would correspond to a subpopulation of identically classified observations. However, this rarely occurs, so decision tree induction systems select attributes based on an information-theoretic measure of uncertainty or entropy that generalizes this ideal situation (see the sidebar).

Data engineering for rotogravure printing

We spent a considerable amount of effort collecting appropriate training data. Some of the attributes thought to be associated with banding had been monitored (albeit for other reasons), but most were not. Instead, banding data was collected by manually filling out forms — not a high priority for industrial craftsmen. Moreover, although conventional problem-solving usually involves collecting only positive data (in this case, of print runs in which banding occurred), we also needed negative data so that Tdidt could better identify the conditions that can lead to banding. Bob Evans found that becoming something of an ambassador/evangelist for machine learning helped convince the craftsmen to gather and record enough data from print runs in which there was no banding.

We also encountered several other challenges in collecting appropriate training data: identifying class labels in training data, selecting descriptive attributes for training data, and dealing with numeric attributes.

Class identification. We selected two classes: Banded and NotBanded. Identifying the Banded class was straightforward: If a band appeared, the values of all attributes being monitored were recorded along with the class Banded, thus creating an instance for training. But when could a cylinder be regarded as NotBanded? Print quantities vary widely from one job to another, and bands frequently do not occur until well into the print run. The question became: "Even though this cylinder did not band, would it have banded if the print order had been greater?" To resolve this, a group of rotogravure printing experts reviewed the frequency distribution for the most recent bands (see Table 2) and decided that a cylinder that banded regardless of the number of impressions (cylinder revolutions) would be Banded, and that a cylinder that attained 1,000M impressions without banding would be NotBanded (unless it did subsequently band, in which case the class would be changed to Banded). Cylinders that did not band but that ran to less than 1,000M impressions were excluded from the training data, under the assumption that the classification of these data was indeterminate.

Attribute identification. Since banding delays had never been conclusively resolved, attribute selection was an iterative process of defining attributes, collecting data, executing Tdidt, reviewing the learned decision tree, and repeating until we achieved satisfactory results. Throughout the process, printing experts were consulted to suggest new attributes to be included. New data was collected, including the values of the new and existing attributes.

Table 2. Frequency of bands as a function of impressions.

Impressions at time of band	Bands
Less than 50M	50
50–100M	12
100–200M	6
200–500M	12
500–1,000M	3
More than 1,000M	11

Table 3. Banding attributes, domain values, and units. The algorithm treats both continuous and ordinal attributes as "numeric" attributes.

Attribute	Domain values/range	Domain type/units
Roughness	0–0.5	Continuous/microns
Anode distance	0–100	Continuous/millimeters
Chrome solution ratio	0–200	Continuous/ratio of chemical mix
Current density	0–200	Continuous/amperes per square decimeter
Plating tank	1910, 1911, Other	Nominal/plating tank
Viscosity	0–30	Continuous/seconds
Proof press cut	0–100	Continuous/percentage
Proof on coated ink	Yes, No	Boolean
Humidity	0–100	Continuous/percentage
Ink temperature	0–120	Continuous/degrees Fahrenheit
Blade oscillation	0–2	Continuous/inches
Blade pressure	0–100	Continuous/pounds
Type on cylinder	Yes, No	Boolean
Blade manufacturer	Benton, Daetwyler, Udeholm	Nominal/blade manufacturer
Varnish percentage	0–100	Continuous/percentage
Ink percentage	0–100	Continuous/percentage
Solvent percentage	0–100	Continuous/percentage
Wax	0–100	Continuous/gallons
Hardener	0–100	Continuous/gallons
Press speed	0–4,000	Continuous/feet per minute
Paper type	Uncoated, Super, Coated	Nominal/paper class
Ink type	Uncoated, Coated	Nominal/ink grade
Steam bar	On, Off	Boolean
Solvent type	Line, Lactol, Xylol, Naphtha, Other	Nominal/commercial solvent
Grain screened	Yes, No	Boolean
Press	TR802, TR813, TR815, TR816, TR821, TR824, TR827, TR828	Nominal/presses
Unit	1 ... 10	Ordinal/printing unit
ESA current	0–5	Continuous/milliamps
ESA voltage	0–5	Continuous/Kilovolts
Cylinder size	Catalog, Spiegel, Tabloid	Nominal/cylinder circumference
Basis weight	0–120	Continuous/pounds per ream

Any attributes that consistently showed little or no promise according to our information-theoretic measure were discarded as irrelevant.

The final set of attributes included many with numeric values (see Table 3). However, Tdidt requires nominal-valued attributes, so we needed a way to map the data to some discrete representation.

Numeric attributes. ID3 uses a simple method for mapping a continuous attribute to a nominal representation.[1] First, it orders an attribute's values over a population. The midpoint between any two consecutive values defines a possible cut point that divides the numeric range into two nominal values corresponding to numeric values less than or equal to the cut point, and values greater than the cut point. ID3 then evaluates the information score stemming from each such binary cut, and uses the cut with minimal uncertainty as a numeric attribute's score. This transformed numeric attribute then competes with other attributes as a possible divisive attribute.

This method splits a numeric range into two at each decision tree node, but the same attribute can be split at multiple nodes along a decision tree path. Thus, binary splits at individual nodes do not preclude finer-grained intervals implied by multiple binary splits throughout the tree.

Nonetheless, finer-grained splits at single nodes can be more comprehensible. The printing experts felt that the values of almost all of the numeric-valued attributes could be partitioned into three general ranges: favorable, neutral, and unfavorable. High humidity, for example, was regarded as a favorable condition, low humidity was unfavorable, and in between the extremes was a range of little or no impact. Within this "neutral" range the experts felt other attributes would be found that were controlling the class. The problem was that the experts could not agree on the boundary values. Initially, rough, arbitrary partitions were attempted with little success.

Bob Evans then designed a partitioning algorithm. The values of an attribute are ordered, and the median value of those corresponding to the Banded class and the NotBanded class define initial cut points, thus dividing the values into three subintervals: those less than or equal to the lower median, those greater than the upper median, and those in between. This basic strategy can be recursively applied on each subinterval, thus creating more than three subintervals at a node. It can also be used on the same attribute at multiple nodes (just as the ID3 approach is), thus creating even finer-grained intervals for a continuous attribute throughout the tree.

At the time, we knew of no work defining more than one split point at decision tree nodes. Recently, other partitioning methods for numeric data have been developed which we expect to investigate soon.[10]

Apos: a knowledge acquisition tool

Although Tdidt is our core mechanism for uncovering diagnostic rules, it is embedded within a more complete knowledge acquisition system called Apos, which facilitates direct interaction between a user and the Tdidt system (see Figure 3).

During execution, Apos uses an initial set of preclassified data points in an external file and (at the user's option) a set of weak heuristics like those in Table 4. The heuristics guide attribute selection in conjunction with the information-theoretic measure. Apos first partitions the numeric-valued attributes to provide ranges that Tdidt can process as discrete values. Apos then consults the heuristics to determine whether to proceed with the calculation. In this implementation, the heuristics amount to data distribution conditions; for example, low humidity values are more likely to

Selecting attributes

An information-theoretic measure of uncertainty or entropy assumes that a category label is encoded as a sequence of bits (0's and 1's) and that we want to minimize the expected number of bits necessary to convey category membership. This requires that more probable categories have smaller encodings. One such encoding procedure works roughly as follows (our induction algorithm does not use this procedure; we describe it here to show the motivation for the information measure that our system does use):[1]

Within a given population of observations, the probability of each category label is the proportion of observations with that label. Category labels are then sequenced by decreasing probability. The sequence is divided to define two subsequences of as nearly equal collective probability as possible. The first bit of each category in the first subsequence will be 1, and 0 otherwise. This process is repeatedly applied on each subsequence to define subsequent bits, until each subsequence represents only one category. This process traces out a binary tree, where each path corresponds to a binary encoding of a category represented at the path's terminal node. The number of bits required to encode a particular category is given by the depth of its corresponding leaf. Since the procedure repeatedly "halves" (an oversimplification) each (sub)sequence, a category C_k with $P(C_k)$ is encoded with approximately $-\log_2 P(C_k)$ bits; the *expected* number of bits required to specify an arbitrary category from among a set of possibilities is

$$-\Sigma_k P(C_k) \log_2 P(C_k)$$

The uncertainty measure is minimized when the probability of one class is 1.0, in which case there is no uncertainty about categorization, and $-P(C_k) \log_2 P(C_k) = -1.0 \times 0.0 = 0$ bits are required to encode the category. Uncertainty is maximized when categories are equally probable. For example, if four categories have probability 0.25 each, then $4 \times 0.25 \times 2.0 = 2$ bits are required to encode each category. In contrast, if classes have probabilities 0.5, 0.25, 0.125, and 0.125 then only $(0.5 * 1) + (0.25 * 2) + (2 * (0.125 * 3)) = 1.75$ bits are required on average. In general, greater uncertainty is reflected in the greater expected amount of information (bits) necessary to encode categories. A standard decision tree algorithm such as ID3 selects the attribute with values that *minimize* uncertainty in the appropriate categorization.

$$-\Sigma_k P(C_k \mid A_i = V_{ij}) \log_2 P(C_k \mid A_i = V_{ij})$$

measures the uncertainty in a subpopulation of objects sharing value V_{ij}. The expected number of bits required to encode categories across the subpopulations defined by the values of attribute A_i is

$$-\Sigma_j P(A_i = V_{ij}) \Sigma_k P(C_k \mid A_i = V_{ij}) \log_2 P(C_k \mid A_i = V_{ij})$$

This information measure is computed for each attribute. The attribute that minimizes this measure is selected to divide the tree. The strategy of selecting attributes that minimize uncertainty in categorization almost universally leads to smaller, more comprehensible decision trees.

An analyst can also use domain knowledge to select attributes, but in the absence of expertise we can get good results from principled, domain-independent heuristics such as uncertainty and many other related measures.[2]

References

1. C. Shannon, "A Mathematical Theory of Communication," *The Bell System Tech. J.*, Vol. 27, No. 3, 1948, pp. 379–423.
2. R. Lopez de Mantaras, "A Distance-Based Attribute Selection Measure for Decision Tree Induction, *Machine Learning*, Vol. 6, No. 1, Jan. 1991, pp. 81–92.

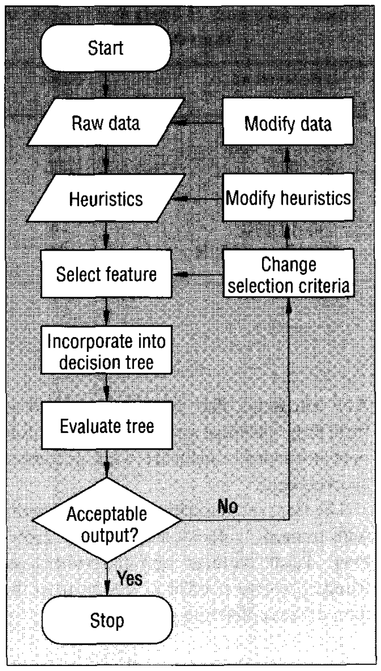


Figure 3. Design of Apos.

be associated with bands than high values. If the distribution of the current data set conflicts with the heuristics, a value of Nil is returned, disqualifying the attribute until some other attribute has been called to partition the data.

These heuristics need not be included, or they may be modified, allowing Apos to discover rules that are counter to the common wisdom. The heuristics make such contradictions explicit to the expert, thus helping to change expert perceptions. Apos presents the attributes and their respective information-theoretic scores to the user at each node as the decision tree is expanded. This allows the user to select attributes that might not have the lowest uncertainty but that do provide the promise of finding a terminal node. At the same time, the user is told of those attributes that tentatively appear to have little or no information value. This "inner loop" continues until either a leaf is found or Apos determines that the current path is blind and no leaf can be found, indicating that additional attributes are required.

The final output is a decision tree. If the user does not accept the tree after evaluating it, he or she can

- change the program's criteria for selecting attributes,
- add/delete attributes (which may require starting over with data collection),
- add/delete attribute domain values,
- add/delete/update heuristics, or
- modify data to reflect any changes in attribute domain values.

Apos then calls Tdidt again to build a new tree. This "outer loop" does not iterate to completion in one session. Thus, results from the latest iteration can be saved, and then used after data has been revised and collected.

Figure 4 shows a typical Apos subtree for the attribute Chrome Solution Ratio; we have replaced precise numeric ranges with labels such as high and low. The distribution of instances at the root (98 NotBanded, 79 Banded) is unacceptable. The problem is mitigated only slightly after branching on discretized values: very low values indicate that banding is very likely, while high values suggest that banding is less likely than at the root. Further expansion provides a conjunctive condition that predicts a low incidence of banding, and that we could use as a prescriptive rule to control against banding:

```
(Chrome Solution Ratio=high) ∧
(Ink Temperature=low) ∧
(Viscosity=high)
```

Results

While bands have long been a problem, specific data was available only since January 1, 1989 (see Table 5). The banding analysis team was formed in December 1989, and we began work on a machine learning approach.

In March 1990 a consultant in a related printing field recommended changing some process control variable values. These variables (attributes) were added to the attribute set we now use. In April the number of bands dropped dramatically, seemingly as a result of the consultant's changes. Meanwhile, Apos made no progress until June, when we implemented the partitioning algorithm that maps numeric attributes to a discrete set.

In September we presented management with a set of rules generated with the assistance of Apos. The rules generally agreed with the consultant's recommendations, and our report acknowledged that those suggestions were a partial solution. The report also indicated based on Apos output that the problem would return in late fall or early winter, when the plant's ambient humidity normally drops with the onset of cold weather. As predicted, in November banding returned to 1989 levels.

In December 1990 some craftsmen began using informal suggestions based on Apos output; bands returned to the level experienced before the drop in humidity. Finally, in April 1991 the experts felt confident enough about the system's performance to generate several trees and a table of conditions associated with banding and nonbanding outcomes; this table was formatted and formally distributed to each printing press. Since May 1991, the frequency of banding has steadily dropped as more and more supervisors and craftsmen "buy into" Apos's process control rules.

At the end of the almost year-long iterative process, the printing experts had discovered at least 10 important rules related to banding over an accumulated database of more than 500 job records. The fact that a consultant's suggestions were implemented before those of Apos somewhat muddies our test of Apos-generated rules alone. However, we can conclude that the Apos rules were primarily responsible for the drops in banding since December 1990.

The Apos rules generally agreed with the consultant's, but they more tightly con-

Table 4. Heuristics that Apos can consult during execution.

Attribute	Rule
Roughness	None
Anode distance	Lower values likely to band
Chrome solution ratio	Lower values likely to band
Current density	Lower values likely to band
Plating tank	None
Viscosity	Lower values likely to band
Proof press cut	Lower values likely to band
Proof on coated ink	None
Humidity	Lower values likely to band
Ink temperature	Higher temperatures likely to band
Blade oscillation	Lower values likely to band
Blade pressure	Higher values likely to band
Type on cylinder	None
Blade manufacturer	None
Varnish percentage	None
Ink percentage	None
Solvent percentage	None
Wax	None
Hardener	None
Press speed	None
Paper type	None
Ink type	None
Steam bar	"Yes" less likely to band
Solvent type	None
Grain screened	"Yes" less likely to band
Press	None
Unit	None
ESA current	None
ESA voltage	None
Cylinder size	None
Basis weight	None

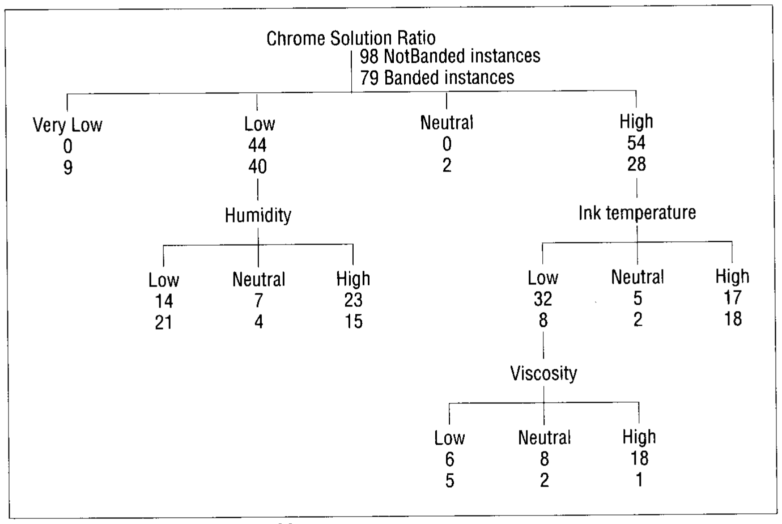


Figure 4. Example subtree created by Apos.

Table 5. Bands per month.

	Jan.	Feb.	Mar.	Apr.	May	June	July	Aug.	Sept.	Oct.	Nov.	Dec.
1989	35	40	40	70	53	34	40	31	56	39	62	38
1990	50	92	95	14	16	15	8	6	8	16	44	20
1991	13	10	11	20	11	17	11	13	6	7	16	3
1992	8	11	9	6	6	5	4	5	5	2	3	2
1993	5	7	4	4	1	1	0	1	11	3	5	0

strained the desirable boundary conditions involving numeric attributes. In one case, an Apos rule conflicted not only with one of the consultant's rules but also with common wisdom — the Apos rule is now used to good effect. Apos also recommended several more rules than did the consultant. Each of these additional rules are enforced in current plant practices.

Apos rules now guide the normative operation of printing jobs, and current banding incidents are often explainable by the system's rules. For example, Apos rules predicted seven of the eleven banding incidents in September 1993; unfavorable ink temperature was traced to a defective cooling system, which was replaced.

WHILE KNOWLEDGE ACQUISITION is a challenge, so is incorporating that knowledge into an existing operation. Process craftsmen are a highly skilled and cautious lot who demand much more than "the computer said so" to explain new procedures, particularly when those suggestions are contrary to popular opinion. To implement the knowledge gained from an intelligent system, a researcher must secure the confidence of these craftsmen through continuous interaction in an atmosphere of mutual respect.

An interactive induction system such as Apos facilitates this kind of interaction by letting the craftsmen communicate directly with the induction engine, override the system's recommendations, and suggest features for subsequent data collection. However, if nonintuitive or counterintuitive rules are ever to be discovered, the experts must sometimes postpone their biases. To get "optimal" results we must walk a fine line between data engineering in pursuit of novel findings and rule engineering based on direct expert intervention. An analyst familiar with inductive methods can aid this process.

A final facet of traditional problem-solving wisdom that we had to address was the desire by both management and craftsmen to find The Answer. Both groups resist probabilistic answers because of a strong bias that deterministic rules can and should be found. In theory this may be true, but in practice the data engineering necessary for such determinism is prohibitive; often the best we can hope for are cost-effective diagnostic rules that admit exceptional cases, derived through a moderate amount of data engineering. Our project was successful because craftsmen were willing to try a new but (in the context of their experience) inexact approach to an old problem.

Our experience and those of others[4,5] demonstrate that interactive induction is a valuable, practical, but underexploited tool for knowledge acquisition. We hope that practitioners continue to experiment with learning technologies, and that researchers recognize the role that machine learning can play in real-world applications.

Acknowledgments

We thank Ray Bareiss for early encouragement and direction on this work. Discussions with Jason Catlett at the 1991 Machine Learning Workshop on Engineering Automation were also influential. In addition, we thank R R Donnelley & Sons Company for support throughout the project. Several Donnelley personnel provided notable help in data engineering, including Don Borders, Frank Cook, Ed Diller, and Bert Jergenson. Doug Fisher was supported by NASA Ames grant NCC 2-645.

References

1. J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, Calif., 1991.

2. J.R. Anderson, *The Adaptive Character of Thought*, Lawrence Erlbaum Associates, Hillsdale, N.J., 1990.

3. M.A. Musen, "Automated Support for Building and Extending Expert Systems," *Machine Learning*, Vol. 4, No. 3, Dec. 1989, pp. 347–76.

4. W. Buntine and D. Stering, "Interactive Induction," in *Machine Intelligence*, Vol. 12, J.E. Hayes-Michie, D. Michie, and E. Tyugu, eds., Oxford University Press, Oxford, England, 1990, pp. 121–138.

5. P. Clark and S. Matwin, "Using Qualitative Models to Guide Inductive Learning," *Proc. 1993 Int'l Machine Learning Conf.*, Morgan Kaufmann, San Mateo, Calif., in press.

6. R. Michalski and C. Chilausky, "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis," *Int'l J. Policy Analysis and Information Systems*, Vol. 4, 1980, pp. 125–161.

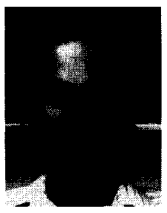
7. R. Bareiss, B. Porter, and K. Murray, "Supporting Start-to-Finish Development of Knowledge Bases," *Machine Learning*, Vol. 4, No. 3, Dec. 1989, pp. 259–284.

8. S. Weiss and C. Kulikowski, *Computer Systems that Learn*, Morgan Kaufmann, San Mateo, Calif., 1991.

9. J.R. Quinlan, "Induction of Decision Trees," *Machine Learning*, Vol. 1, No. 1, 1986, pp. 81–106.

10. U.M. Fayyad and K.B. Irani, "On the Handling of Continuous-Valued Attributes in Decision Tree Generation," *Machine Learning*, Vol. 8, No. 1, Jan. 1992, pp. 87–102.

Bob Evans is a software engineer with R R Donnelley & Sons Company and an adjunct assistant professor in computer science at Volunteer State Community College. His primary responsibility is process delay analysis. His recent work has focused on inductive learning methods to determine attribute interactions critical to printing productivity. Bob received his AB in mathematics from Indiana University in 1963 and his M.Eng. in computer science from Vanderbilt University in 1991. He is a member of IEEE and ACM. His address is R R Donnelley & Sons Company, 801 Steam Plant Road, Gallatin, TN 37066.



Doug Fisher is an assistant professor in computer science at Vanderbilt University. His research interests are in inductive and explanation-based learning, models of human learning, exploratory data analysis, cluster analysis, diagnosis, and aerospace applications. He is general chair of the 1995 AI and Statistics Workshop, a cochair of the 1994 SPIE Applications of Artificial Intelligence Conference, and he cochaired the machine learning area of the program committee for the 1991 National Conference on Artificial Intelligence. He received his PhD in 1987 from the Department of Information and Computer Science at the University of California at Irvine. He is a member of AAAI, ACM, and the editorial board of *Machine Learning*. His address is Computer Science Department, Box 1679, Station B, Vanderbilt University, Nashville, TN 37235; Internet: dfisher@vuse. vanderbilt.edu