Just Enough Learning (of Association Rules): The TAR2 "Treatment" Learner

Tim Menzies*, Ying Hu†

written: Nov 22 2002; accepted: Feb 2003; revised: Oct 2007

Abstract. An over-zealous machine learner can automatically generate large, intricate, theories which can be hard to understand. However, such intricate learning is not necessary in domains that lack complex relationships. A much simpler learner can suffice in domains with *narrow funnels*; i.e. where most domain variables are controlled by a very small subset.

Such a learner is TAR2: a weighted-class minimal contrast-set association rule learner that utilizes confidence-based pruning, but not support-based pruning. TAR2 learns *treatments*; i.e. constraints that can change an agent's environment. Treatments take two forms. *Controller treatments* hold the *smallest number* of conjunctions that *most improve* the current state of the system. *Monitor treatments* hold the *smallest number* of conjunctions that best detect future faulty system behavior. Such treatments tell an agent what to do (apply the controller) and what to watch for (the monitor conditions) within the current environment.

Because TAR2 generates very small theories, our experience has been that users prefer its tiny treatments. The success of such a simple learner suggests that many domains lack complex relationships.

Keywords: Association rules, treatment learning, contrast sets,

> "Don't tell me where I am, tell me where to go."
> - a (very busy) user

1. Introduction

Machine learners generate theories. People read theories. What kind of learners generate the kind of theories that people want to read?

If the reader is a busy person, then they might not need, or be able to use, complex theories. Rather, such a busy person might instead just want to know the *least* they need to do to achieve the *most* benefits. It therefore follows that machine learning for busy people should not strive for (e.g.) elaborate theories or (e.g.) increasing the expressive power of the language of the theory. Rather, a better goal might be to find the smallest theory with the most impact.

For example, Figure 1 and Figure 2 contrasts two theories learnt from the same data set using the TAR2 system discussed here and the C4.5 de-

* Computer Science Department, Portland State University, Portland, Oregon, USA, and Lane Department of Computer Science, West Virginia University, Morgantown, WV, USA, (tim@menzies.us)

† Dept. Electrical & Computer Engineering, University of British Columbia, Vancouver, B.C. Canada, (yingh@ece.ubc.ca)

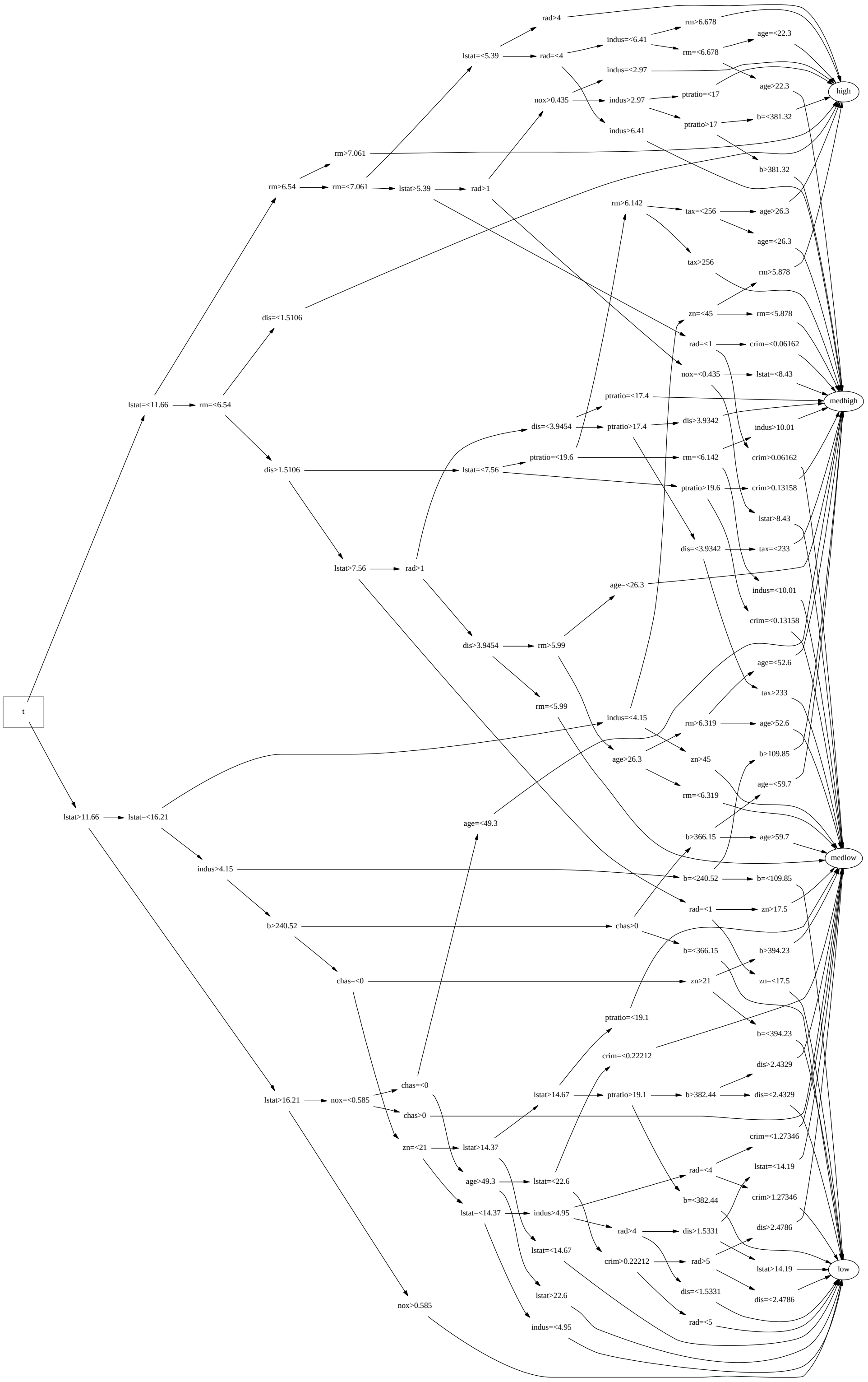


Figure 1. A learnt decision tree from 506 cases in HOUSING example set from the UC Irvine repository. Classes (right-hand-side), top-to-bottom, are "high", "medhigh", "medlow", and "low" This indicates median value of owner-occupied homes in $1000's.

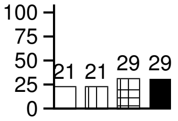
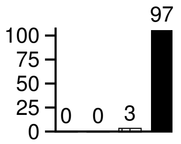
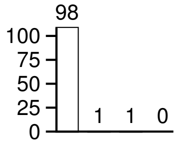
	`controllerH` (i.e. best action)	`monitorH` (i.e. disaster if..)
baseline	$6.7 \leq RM < 9.8$ $\wedge 12.6 \leq PTRATION < 15.9$	$0.6 \leq NOX < 1.9$ $\wedge 17.16 \leq LSTAT < 39.0$
		
sample= 506 examples	sample= 38 samples	sample= 81 examples

Figure 2. Treatments learnt in the same domain as Figure 1. This dataset has the class distribution shown in the bottom table, left-hand-side. KEY: LSTAT= lower status of the population; NOX= nitric oxides concentration (parts per 10 million); PTRATIO= pupil-teacher ratio by town; RM= average number of rooms per dwelling; KEY: □ low; ▥ medlow; ▦ medhigh; ■ high.

cision tree learner (Quinlan, 1992). In Figure 1, C4.5 has learnt a elaborate description of different kinds of houses in Boston. This description is quite large- its details are barely legible and we've had to compress the image in order to squeeze it onto one page. An automatic process could quickly parse this tree and use it to automatically make a classification. However, having worked with domain experts for many years, we assert that Figure 1 contains a daunting amount of detail for human readers.

Figure 2 shows' TAR2's minimal description of the *differences* between house types. This description is far shorter than Figure 1 and hence can quickly be explained to a domain expert. Figure 2 describes these differences between house types in terms of *treatments*; i.e. a constraint on controllable variable that changes the class distribution. The *controller* treatment (shown in the middle of Figure 2) is TAR2's comments on what might most improve the current situation. This current *baseline* situation is shown left-hand-side of Figure 2: the 506 housing examples contain 29% "high" quality houses. TAR2's *controller* treatment asserts that if we focus on houses with seven to nine rooms in suburbs with parent/teacher ratios of 12.6 to 15.9, then we will find 38 houses, 97% of which will be of high quality (and 97%$\gg$29%).

Similarly, the *monitor* treatment shown right-hand-side of Figure 2. describes what could most *degrade* the current baseline situation. This *monitor* treatment asserts that the worst thing we could do in the current situation would be to focus on houses where the air has nitrous oxides levels between 0.6 and 1.9 in suburbs where the living standard is between 17.16 and 39.0. The *monitor* treatment warns that if that worst-case policy is followed, then no "high" quality houses will be found. Indeed, it predicts that in that worst-case scenario, 98% of the houses found will be "low" quality.

Our experience with business users is that they prefer find TAR2's simpler theories. C4.5's theory contains more details but TAR2's theory gives succinct advice on how to change the current situation. As one user put it "decision trees just tell you where you are, treatments tell you what to do".

TAR2's succinct theories will miss the complex interrelations that C4.5 might find. We have reasons to believe that many domains lack complex relationship. Many domains exhibit a curious *narrow funnel effect*; i.e. a small number of critical variables control the remaining variables within a system (the metaphor being that all processing runs down the same narrow funnel (Menzies et al., 1999)). The concept of narrow funnels has been reported in many domains under a variety of names including:

- *Master-variables* in scheduling (Crawford and Baker, 1994);
- *Prime-implicants* in model-based diagnosis (Rymon, 1994) or machine learning (Rymon, 1993), or fault-tree analysis (Lutz and Woodhouse, 1999).
- *Backbones* in satisfiability (Parkes, 1999; Singer et al., 2000);
- *the dominance filtering* used in Pareto optimization of designs (Josephson et al., 1998);
- *Minimal environments* in the ATMS (DeKleer, 1986);
- The *base controversial assumptions* of HT4 (Menzies and Compton, 1997).
- The small *feature subset selection* effect (Kohavi and John, 1997) and the related *1R* effect (Holte, 1993).

Whatever the name, the core intuition in all these terms is the same: what happens in the total space of a system can be controlled by a small critical region.

We have argued previously that narrow funnels are very common (Menzies and Cukic, 2000b; Menzies et al., 2000; Menzies and Cukic, 2000a; Menzies and Singh, 2001). For systems with narrow funnels, the space of options within a large space reduces to just the range of a few variables within the narrow funnel. In such a reduced space, variables assignments outside the funnel are highly correlated to assignments within the funnel. Machine learning in such domains is very simple: an adequate theory need only comment on assignments to the variables that are highly correlated to funnel assignments.

This paper uses the TAR2 system to check the merits of assuming narrow funnels for the purposes of machine learning. TAR2's distinguishing feature is that it performs very well, yet it is seems overly simplistic. The algorithm outputs only two rules: the best smallest controller and the best smallest monitor. If domains contain complex relationships, then these two small associations will be useless. The algorithm's runtimes are exponential on the size of the treatments. Hence, the algorithm makes the following "small treatment assumption"; i.e. *adequate treatments can be built from small treatments*. If the small treatment assumption fails, then TAR2's exponential runtimes will make it impractically slow. Also, the algorithm relies on a *confidence1* measure which prunes the space of possible associations. The confidence1 measure we describe below has no special merit: it was merely the first one we could think of. Further, our initial implementation worked without algorithmic or memory management optimizations. Our only explanation for the surprising success of this simplistic implementation is that the small treatment assumption holds for the domains we studied.

The rest of this article discusses TAR2. After an introductory example and a discussion of related work, the TAR2 algorithm is presented. This is followed by examples and evaluations and an analysis of the general applicability of our approach. Our prior work has only offered high-level descriptions of treatment learning (e.g. (Menzies and Hu, 2003)). The contribution of this paper is a detailed discussion of the implementation and generality of treatment learning. In expanding on those details, we show numerous empirical results that have not previously been published. These studies are described in sufficient detail that if another author believed they had a better summarization method than treatment learning, then comparative studies could be conducted.

2. Related Work

If domains lack complex relationships, then it is possible that adequate theories can be learnt from small subsets of the available attributes. Various researchers have reported that this is indeed the case. For example Holte wrote a machine learner that was deliberately restricted to learning theories using a single attribute. Surprisingly, Holte found that learners that use many attributes such as C4.5 perform only moderately better this 1R algorithm (Holte, 1993). Nevertheless, TAR2 does not use the 1R technique since our results show that best treatments may require more than one attribute.

In other work, Kohavi and John *wrapped* their learners in a pre-processor that used a heuristic search to grow subsets from size 1. At each step in the growth, a learner was called to find the accuracy of the theory learned from the current subset. Subset growth was stopped when the addition of new attributes did not improve the accuracy. As shown in Figure 3, spectacular reductions in the number of the attributes can be achieved, with only minimal lose of

	number of attributes										
before:	10	13	15	180	22	8	25	36	6	6	6
after:	2	2	2	11	2	1	3	12	1	1	2
reduction:	80%	84%	87%	94%	90%	87%	88%	67%	83%	83%	67%
Δaccuracy:	0%	6%	5%	4%	2%	1%	0.5%	0%	-25%	6%	7%

Figure 3. Feature subset selection using wrappers, hill-climbing, and ID3 (i.e. C4.5 with pruning disabled). The ΔAccuracy figure is the difference in the accuracies of the theories found by ID3 the $before$ and $after$ attributes. From (Kohavi and John, 1997).

accuracy (Kohavi and John, 1997). Nevertheless, TAR2 does not use this technique since relevant feature selection with wrappers can be prohibitively slow since *each step* of the heuristic search requires a call to the learning algorithm.

If TAR2 isn't 1R or wrappers, what is it? This rest of this section expands on the following definition. TAR2 is a *weighted-class minimal contrast-set* association rule learner that uses *confidence measures* but not *support-based pruning*.

TAR2 learns treatments and the general form of a treatment is:

$$\begin{array}{lll} R_1 & if & Attr_1 = range_1 \wedge Attr_2 = range_2 \wedge ... \\ & then & good = more \wedge bad = less \\ R_2 & if & Attr_1 = range_1 \wedge Attr_2 = range_2 \wedge ... \\ & then & good = less \wedge bad = more \end{array}$$

where R_1 is the controller rule; R_2 is the monitor rule; *good* and *bad* are sets of classes that the agent likes and dislikes respectively; and *more* and *less* are the frequency of these classes, compared against the current situation, which we call the *baseline*. The nature of these output rules distinguishes TAR2 from many other learning strategies.

Association rule learning: Classifiers like C4.5 and CART learn rules with a single attribute pair on the right-hand side; e.g. *class= goodHouse*. Association rule learners like APRIORI (Agrawal and Srikant, 1994) and TAR2 generate rules containing multiple attribute pairs on both the left-hand-side and the right-hand-side of the rules. That is, classifiers have a small number of pre-defined targets (the classes) while, for association rule learners, the target is less constrained.

General association rule learners like APRIORI input a set of D transactions of items I and return associations between items of the form $LHS \Rightarrow RHS$ where $LHS \subset I$ and $RHS \subset I$ and $LHS \cap RHS = \emptyset$. A common restriction with classifiers is that they assume the entire example set can fit into RAM. Learners like APRIORI are designed for data sets that need not reside in main memory. For example, Agrawal and Srikant report experiments with association rule learning using very large data sets with 10,000,000 examples and size 843MB (Agrawal and Srikant, 1994). However, just like Webb (Webb, 2000), TAR2 makes the “memory-is-cheap assumption”; i.e. TAR2 loads all it’s examples into RAM.

Specialized association rule learners like CBA (Liu et al., 1998) and TAR2 impose restrictions on the right-hand-side. For example, TAR2’s right-hand-sides show a prediction of the *change* in the class distribution if the constraint in the left-hand-side were applied. The CBA learner finds *class association rules*; i.e. association rules where the conclusion is restricted to one classification class attribute. That is, CBA acts like a classifier, but can process larger datasets that (e.g.) C4.5. TAR2 restricts the right-hand-side attributes to just those containing criteria assessment.

Weighted-learning: Standard classifier algorithms such as C4.5 (Quinlan, 1992) or CART (Breiman et al., 1984) have no concept of class *weighting*. That is, these systems have no notion of a *good* or *bad* class. Such learners therefore can’t filter their learnt theories to emphasize the location of the *good* classes or *bad* classes. Association rule learners such as MINWAL (Cai et al., 1998), TARZAN (Menzies and Sinsel, 2000) and TAR2 explore *weighted learning* in which some items are given a higher priority weighting than others. Such weights can focus the learning onto issues that are of particular interest to some audience. For example TARZAN (Menzies and Sinsel, 2000) swung through the decision trees generated by C4.5 (Quinlan, 1992) and 10-way cross-validation. TARZAN returned the smallest treatments that occurred in most of the ensemble that *increased* the percentage of branches leading to some preferred highly weighted classes and *decreased* the percentage of branches leading to lower weighted class. TAR2 was an experiment with applying TARZAN’s tree pruning strategies directly to the C4.5 example sets. The resulting system is simpler, fast to execute, and does not require calling a learner such as C4.5 as a sub-routine.

Contrast sets: Instead of finding rules that describe the current situation, association rule learners like STUCCO (Bay and Pazzani, 1999) finds rules that differ meaningfully in their distribution across groups. For example, in STUCCO, an analyst could ask ”what are the differences between people with Ph.D. and bachelor degrees?”. TAR2’s variant on the STUCCO strategy is to combine contrast sets with weighted classes with minimality. That is, TAR2 treatments can be viewed as the smallest possible contrast sets that distinguish situations with numerous highly-weighted classes from situations that contain more lowly-weighted classes.

Support-based pruning: In the terminology of APRIORI, an association rule has *support* s if $s\%$ of the D contains $X \wedge Y$; i.e. $s = \frac{|X \wedge Y|}{|D|}$ (where $|X \wedge Y|$ denotes the number of examples containing both X and Y). The *confidence* c of an association rule is the percent of transactions containing X which also contain Y; i.e. $c = \frac{|X \wedge Y|}{|X|}$.

Many association rule learners use *support-based pruning* i.e. when searching for rules with high confidence, sets of items $I_i, ...I_k$ are only be examined if all its subsets are above some minimum support value. Note that support-based pruning is *not* the same as weighted-class learning since the former assesses a rule according to the amount of evidence in the input data set while the latter assesses a rule according the amount of emphasis a user has placed on the class associated with that rule.

Support-based pruning is impossible in weighted association rule learning since with weighted items, it is not always true that subsets of *interesting* items (i.e. where the weights are high) are also interesting (Cai et al., 1998). Another reason to reject support-based pruning is that it can force the learner to miss features that apply to a small, but interesting subset of the examples (Wang et al., 2001).

Confidence-based pruning: Without support-based pruning, association rule learners rely on confidence-based pruning to reject all rules that fall below a minimal threshold of adequate confidence. TAR2 uses *confidence1* pruning.

3. Confidence1 Pruning

TAR2 targets the attribute ranges that "nudge" a system away from undesired behavior and towards desired behavior. TAR2's score for each range is the *confidence1* measure. This value is high if a range occurs frequently in desired situations and infrequently in undesired situations. That is, if we were to impose this range as a constraint, then it would tend to "nudge" the system into better behavior.

To find confidence1, we assume that we can access $\$class$; i.e. some numeric value assigned to $class$. The class with the highest value is the $best$ class. The $lesser$ classes are the set of all classes, less the $best$ class.

In the following description, some attribute A has a specific setting $A.R$; i.e. some member of the range of A is assigned to A. Let $O[C]_{A.R}$ be the number of occurrences of some attribute range in some class C; i.e.

$$O[C]_{A.R} = |A.R \wedge class = C \wedge D|$$

Here, D is the set of training examples.

To generate confidence1, we compare the relative frequencies of an attribute range in different classes. This comparison is weighted by the difference in the scores of the classes, and normalized by the total frequency count of the attribute range; i.e.

Items				*Criteria*
outlook	*temp(°F)*	*humidity*	*windy?*	*class*
sunny	*85*	*86*	*false*	*none*
sunny	*80*	*90*	*true*	*none*
sunny	*72*	*95*	*false*	*none*
rain	*65*	*70*	*true*	*none*
rain	*71*	*96*	*true*	*none*
rain	*70*	*96*	*false*	*some*
rain	*68*	*80*	*false*	*some*
rain	*75*	*80*	*false*	*some*
sunny	*69*	*70*	*false*	*lots*
sunny	*75*	*70*	*true*	*lots*
overcast	*83*	*88*	*false*	*lots*
overcast	*64*	*65*	*true*	*lots*
overcast	*72*	*90*	*true*	*lots*
overcast	*81*	*75*	*false*	*lots*

Figure 4. A log of some golf-playing behavior.

$$\frac{\sum_{C\in lesser}((\$best - \$C) * (O[best]_{A.R} - O[C]_{A.R}))}{|A.R \wedge D|}$$

For example, from the golf playing example of Figure 4, let us assume that the classes have been scored as follows: ”lots”=8, ”some”=4, ”none”=2; i.e. ”lots” is the $best$ class. The range *outlook=overcast* appears four, zero, and zero times when playing ”lots”, ”some”, and ”none” golf (respectively). The confidence1 of *outlook=overcast* is therefore:

$$\frac{((8-2)*(4-0)) + ((8-4)*(4-0))}{4+0+0} = 10$$

Figure 5 shows the range of confidence1 seen in Figure 4. The confidence1 ranges shown in black are outstandingly high; i.e. these are the values which generate the best control treatments. TAR2 forms its treatments by exploring subsets of the ranges with outstandingly high confidence1 values.

4. Inside TAR2

TAR2 generates controller and monitor treatments. Monitors are generated using in same manner as generating controllers. However, before the monitor

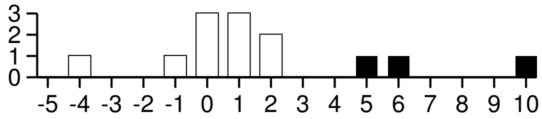


Figure 5. Frequency of confidence1 generated from Figure 4. Assumes that numeric ranges have been divided into 3 $bands$. Outstandingly high confidence1 values shown are in black. Y-axis is the number of ranges that have a particular confidence1 value.

input:	D	The examples.
	$items$	Attributes seen in the examples.
	$best$	The best combination of criteria.
	N	Desired size of LHS.
	$promising$	Threshold for a useful attribute range.
	$skew$	Threshold for acceptable number of $best$ entries in $treated.$
	$bands$	Number of divisions within continuous ranges.
output:	lhs	A conjunction of attribute ranges
	rhs	a change in the class distributions

```
$D_1 \leftarrow$ discretize($D$, $bands$)
$temp \leftarrow baseline \leftarrow$ frequency($D_1$)
for $attribute$ in $items$ $\{$
    for $R$ in $attribute.ranges$ $\{$
        if   confidence1($attribute.R$) $\geq promising$
        then $candidates \leftarrow candidates$ + $attribute.R\}\}$
for $C \subseteq candidates$ where $|C| = N$ $\{$
    $treated \leftarrow C \wedge D_1$
    $result \leftarrow$ frequency($treated$)
    if   $result > temp$ and $|best \wedge D_1| / |best \wedge treated| > skew$
    then   $\{lhs \leftarrow C$
              $rhs \leftarrow$ compare($baseline, result$)
              $temp \leftarrow result\}\}$
if ($lhs \neq \emptyset$ and $rhs \neq \emptyset$) then return ($lhs$, $rhs$)
else return "no treatment"
```

Figure 6. The TAR2 algorithm.

is generated, the scoring function for the criteria is reversed so TAR2 now seeks attribute ranges that *nudge* a system into worse behavior. The rest of this section discusses how to generate controllers.

The TAR2 algorithm is shown in Figure 6. The `frequency` function counts the frequency of examples falling into different criteria. Using this function, a $baseline$ class distribution is collected from D (this is used later to contrast different treatments) and copied to a $temp$ variable (this is used to store the best distribution seen so far). The `compare` function compares two frequencies to generate reports like (e.g.) 43% less ”lots” and 5% less ”some” and 167% more ”none”. When these percentages are *greater* than 100%, then

```
controllerG
if   outlook=overcast
then (230% more "lots" and no "some"
     and no "none").

monitorG
if    90 <= humidity < 97
then (43% less "lots" and 5% less "some"
     and 167% more "none").
```

Figure 7. Control and monitor rules found from Figure 4. To control *outlook*, unscrupulous owners of golf courses could (e.g.) bribe radio announces to lie about the weather report.

the treatment selects from a *greater* percentage of some class (compared to the baseline).

The `discretize` function divides the numeric ranges seen in the examples into $bands$ number of groups. TAR2 was originally designed using a very simple discretization policy; i.e. TAR2 sorts the known values and divides into $bands$with (roughly) the same cardinality. It was anticipated that this policy would be too simplistic and would have to be improved. However, our empirical results (see below) were so encouraging that we were never motivated to do so.

Once a treatment is found, it is applied to the example set to create a $treated$ example set; i.e. all the examples that don't contradict the proposed treatment (see line 8). A "good" treatment includes most of the examples that have the $best$ criteria (e.g. in the golf example of Figure 4, $best$= playing "lots" of golf). The $skew$parameter is used at line 10 to reject "bad" treatments; i.e. those that don't contain enough of the $best$ criteria. For example, at $skew$=5, at least 20% of the $best$ criteria must appear in the treatment.

TAR2 explores subsets of the $ranges$ found in a set of examples D (see line 7). Subset exploration is constrained to just the $ranges$ with an outstandingly large confidence1 score (see line 5). Even with this restriction, there are still an exponential number of such subsets. Hence, to be practical, TAR2 must seek the *minimal* possible number of control actions and monitors. Accordingly, the user of TAR2 constrains its learning to rule conditions of size N, where N is small (see line 7). Often, effective treatments can be found using $N \leq 4$ which suggests that narrow funnels existed in the datasets used for our case studies.

baseline:	controllerG:	monitorG:
(from Figure 4)	$outlook$= $overcast$	$humid = [90..97)$

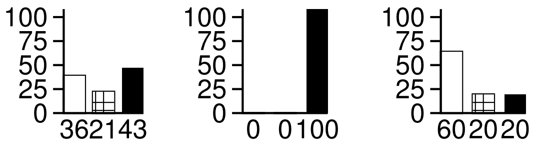


Figure 8. Percentage of classes seen in different situations. The left-hand-side histogram is a report of the class frequencies seen in Figure 4. The middle and right-hand-side histograms were generated by applying the treatments of Figure 7. KEY: □ none; ▦ some; ■ lots.

5. Examples and Experiments

5.1. Examples

The output of TAR2 describes constraints which, if applied to the dataset, may reject certain examples. For example, the `controllerG` treatment of Figure 7 contains the constraint $outlook = overcast$. If we reject all items in the golf dataset that contradicts this constraint, then our golfers now play ”lots”, ”some”, and ”none” golf in 100%, 0%, and 0% (respectively) of the constrained dataset (as shown in the middle histogram of Figure 8).

The monitor rule `monitorG` of Figure 7 was generated in a similar manner; but with the scoring system reversed; i.e. ”lots”=2, ”some”=4, ”none”=8. In this case, ”none” is the “$best$” class and TAR2 will find a treatment that selects for less golf behavior; i.e. $90 \leq humidity < 97$. After applying this constraint , the class distribution changes to the right-hand-side histogram of Figure 8.

5.2. Experiments

This section discusses experiments with TAR2 where the leaner was assessed via two methods:

Xvals: Standard N-way cross-validation studies.

Simulations: Simulations showing how well TAR2's treatments can control or monitor some model.

5.2.1. *Xval Studies*

Figure 9 to Figure 13 shows TAR2 executing over some samples from the UC Irvine repository (`http://www.ics.uci.edu/˜mlearn/`). These figures

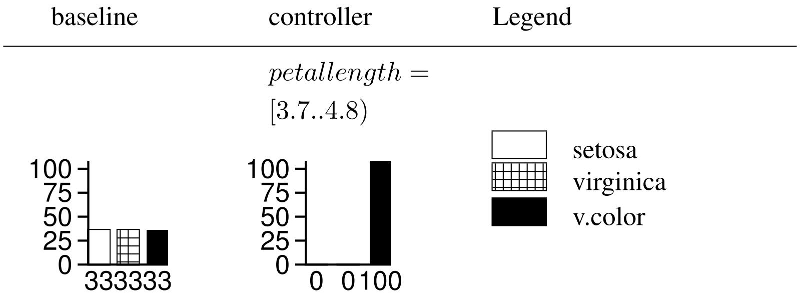


Figure 9. Iris

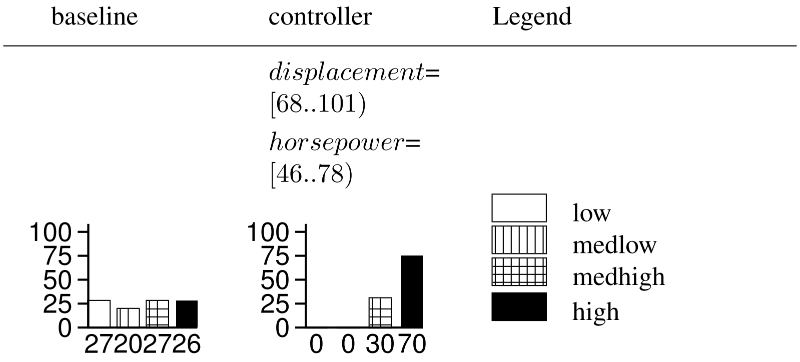


Figure 10. Autompg

display the effects of the treatment closest to the average improvement seen in a 10-way cross-validation study. Each figure show the class distributions as percentages and the domain classes are shown in a legend. In the legend, the heuristic worth assigned to each class is, top-to-bottom, worst-to-best.

In Figure 9, TAR2 was told that the worth of each type of flower was (in increasing order) *setosa*, *virginica*, then *v.color*. TAR2 then learnt that $3.7 \leq petallength < 4.8$ would select for the flower with highest worth (i.e. *v.color*).

Similarly, in Figure 10, TAR2 learnt a selector that favored high quality cars. By restricting engine size to $68 \leq displacement < 101$ and $46 \leq horsePower < 78$, the ratio of high quality cars increased from 26% to 70%. Further, the low and medium low cars have disappeared.

In Figure 11, TAR2 learnt a specialized feature extractor for finding pictures mixed in with text, horizontal lines, vertical lines, and graphics. According to TAR2, a height between 34 to 86, and a mean number of white-black transitions between 3.9 and 9.5 will locate text blocks, and nothing else.

In the car domain of Figure 12, most of the classes are non-best. The average best controller seen in the 10-way cross-validation for the car domain was

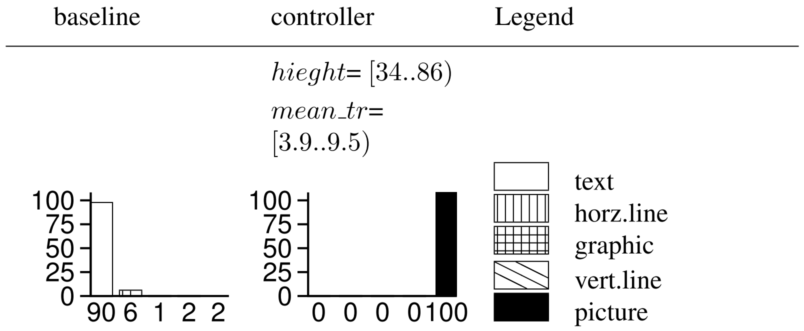


Figure 11. Page-block

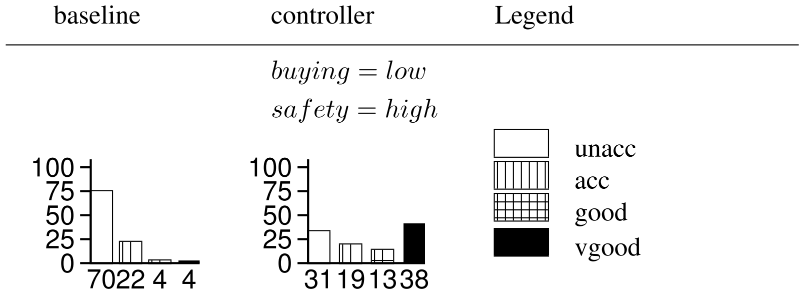


Figure 12. Car

buying=low and *safety=high*. While this controller increases the frequency of very good cars from 4% to 38%, this controller still leaves us with 31% unacceptable cars. While this controller is weak, the monitor obtained by reversing the class scoring is very strong. Figure 13 shows that monitor: if we select two

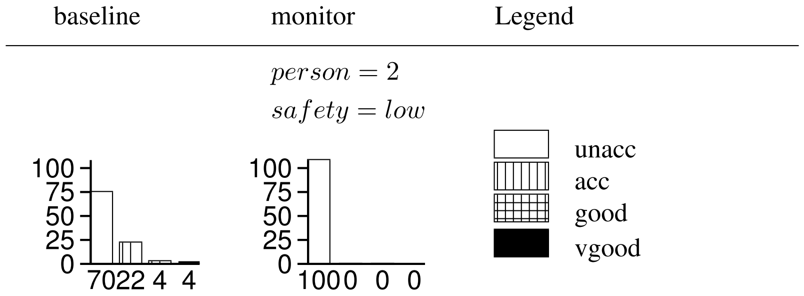


Figure 13. Car: reversing the class scoring.

person cars with low safety, then 100% of the cars are unacceptable. That is, when the best class occurs rarely in the dataset, TAR2 may be better at finding methods to *degrade* a system, rather than *improve* it.

5.2.2. *Simulation Studies*

Another way to assess TAR2 is to test how well it can control some model. To perform such an assessment, we (i) generated data sets from some model; (ii) applied TAR2 to find treatments from those data set; (iii) imposed those treatments as constraints on the model; (iv) ran the model a second time; (v) compared the outputs of the second run to the predictions made by TAR2.

In our first two simulations studies, a *baseline* class distribution was used by TAR2 to generate a best controller and a prediction of how this best controller would change the class distribution. We call the predicted distribution the *treated* distribution. The *actual* distribution was the class distribution seen after the best controller was imposed on the model and the model executed again. In Figure 14 and Figure 15, the treated distribution matches the result distribution almost exactly; i.e. TAR2 accurately predicted the effects of the controller treatment.

Figure 14 was generated from a model of software project risk. This risk model was implemented as part of the COCOMO project. The goal of the COCOMO project is to build an open-source software cost estimation model (Abts et al., 1998). Internally, the model contains a matrix of parameters that should be tuned to a particular software organization. Using COCOMO-II, the Madachy risk model can assess the risk of a software cost over-run (Madachy, 1997). For machine learning purposes, the goal of using the Madachy model is to find a change to a description of a software project that reduces the likelihood of a poor risk software project (Menzies and Sinsel, 2000; Menzies and Hu, 2001b). In the experiment shown in Figure 14, the model was executed 30,000 times using randomly selected inputs. When the treatments learnt from TAR2 treatments were imposed on model inputs, and the model was executed again, all the high risk projects were removed, the percentage of medium risk projects was significantly reduced, and the percentage of low risk projects was tripled.

Figure 15 shows TAR2 controlling a qualitative description of an electrical circuit. A qualitative description of a circuit of 47 wires connecting 9 light bulbs and 16 other components was coded in Prolog. The model was expressed as a set of constraints; e.g. the `sum` of the voltages of components in series is the `sum` of the voltage drop across each component. The goal of the circuit was to illuminate a space using the 9 light bulbs. The circuit is qualitative and qualitative mathematics is nondeterministic; e.g. sum of a negative and a positive value is unknown. The problem with the circuit was out-of-control nondeterminism. On backtracking, this circuit generated 35,228 different solutions to the constraints. In many of these solutions, the

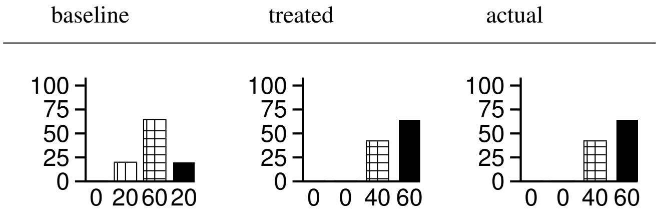


Figure 14. COCOMO key: □ very high risk; ▥ high risk; ▦ medium risk; ■ low risk.

circuit was unacceptably dark: only two bulbs glowing, on average (see the top histogram of Figure 15) . The goal of the machine learning was hence to find a minimal set of changes to the circuit to increase the illumination (Menzies and Hu, 2001a). Figure 15 shows the distribution of the frequency with which bulbs glowed in a qualitative circuit description. The behavior of qualitative circuits is notoriously hard to predict (Clancy and Kuipers, 1997) but TAR2 found two actions on the circuit that trebled the average number of bulbs that glowed (see the *treated* and *actual* plot of Figure 15).

Figure 16 shows a third simulation study with TAR2. Analysts at the NASA Jet Propulsion Laboratory debate satellite design by building a semantic network connecting design decisions to satellite requirements (Feather et al., 2000). Each edge is annotated with the numeric cost and benefits of taking some action. Some of these nodes represent base decisions within the project (e.g. selection of a particular type of power supply). Each set of decisions has an associated cost. The net can be executed by selecting actions and seeing what benefits results. One such network included 90 possible actions; i.e. $2^{99} \approx 10^{30}$ combinations of actions. Note the black line, top-left, of

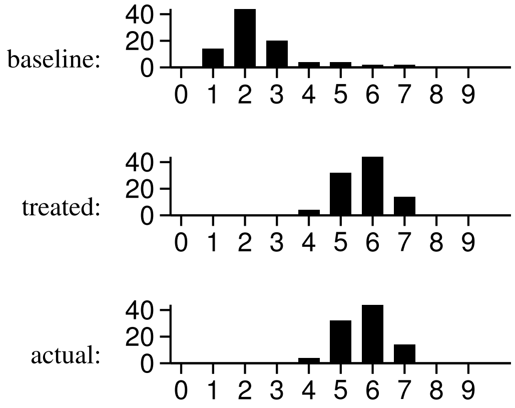


Figure 15. Circuit. X-axis denotes number of bulbs glowing in the circuit.

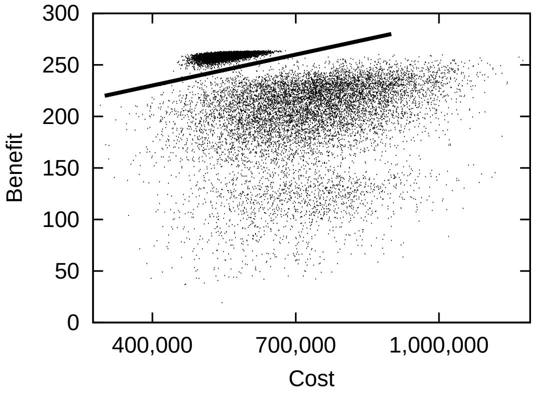


Figure 16. Results from the satellite domain. The dots below the line show the initial output of the model: note the very large spread in the costs and benefits. The dots above the line show the final outputs of the model after 5 iterations of TAR2 learning.

Figure 16. All the dots below this line were generated via 10,000 random selections of the decisions, and the collection of their associated costs and benefits. All the dots above this line represent high benefit, low cost projects found by TAR2 (Feather and Menzies, 2002). In this application, TAR2 was used as a knowledge acquisition tool. After each run of TAR2, the proposed best controller was debated with the analysts. Each run, and its associated debate, resulted in a new set of constraints for the semantic net. The new constraints were then imposed on the model before the next run. After five runs, TAR2 found 30 decisions (out of 99) that crucially effected the cost/benefit of the satellite. Note that this means TAR2 also found 99-30=67 decisions that could be safely ignored.

For comparison purposes, a genetic algorithm (GA) was also applied to the Figure 16 domain (Feather and Menzies, 2002). The GA also found decisions that generated high benefit, low cost projects. However, each such GA solution commented on every possible decisions and there was no apparent way to ascertain which of these are the most critical decisions. The TAR2 solution was deemed superior to the GA solution by the domain experts, since the TAR2 solution required just 30 actions rather than the 99 demanded by the GA.

Note that the Figure 16 case study is not a counter example to our thesis that many domains have narrow funnels. That study adopted the incremental approach for reasons of convenience. JPL's semantic net simulator was too slow to generate enough examples at one run. Hence, an incremental generate-and-constrain approach was taken.

	Attributes				treatment	runtime
source:domain	# examples	#continuous	#discrete	#classes	size	(secs)
UCI:iris	150	4	0	3	1	<1
UCI:wine	178	13	0	3	2	<1
UCI:car	1,728	0	6	4	2	<1
UCI:autompg	398	6	1	4	2	1
UCI:housing	506	13	0	4	2	1
UCI:page blocks	5,473	10	0	5	2	2
here:circuit	35,228	0	18	10	4	4
here:COCOMO	30,000	0	23	4	1	2
here:satellite	30,000	0	99	9	5	86
other:reachness	25,000	4	9	4	2	3
:	250,000	4	9	4	1	23

Figure 17. Runtimes for TAR2 on different domains (on a 333MHz Windows machine with 200MB of ram). “UCI” denotes data sets from the machine learning repository at UC Irvine. “Here” denotes data sets taken from this article. ‘ The text discusses experiments with 10,000 examples from the satellite domain. This table shows a larger case study of 30,000 examples. ‘Other” denotes a data set taken from (Menzies and Hu, 2002).

6. Generality

This section is an algorithmic assessment of TAR2. Such an algorithm assessment comments on TAR2’s ability to scale to larger domains.

Figure 17 reports TAR2 runtimes on data sets of different sizes. Figure 18 shows three studies where the size of the treatments (N, from line 7 in Figure 6) was held constant, and the size of the dataset was increased. Figure 19 shows one study were the size of the dataset was held constant and the size of the treatments was increased. Note that:

1. TAR2 can handle small to medium sized datasets. For example, the algorithm learnt effective treatments in 23 seconds from a dataset containing size 250,000 examples: see the *reachness* domain in Figure 17.

2. TAR2 has the potential to scale to large datasets. Assuming constant treatment size, TAR2’s runtimes are linear on dataset size: see Figure 18.

3. However, the algorithm is exponential on treatment size: see the marked increase in the runtimes between N=2 and N=3 in Figure 18 and the log-linear plot of Figure 19.

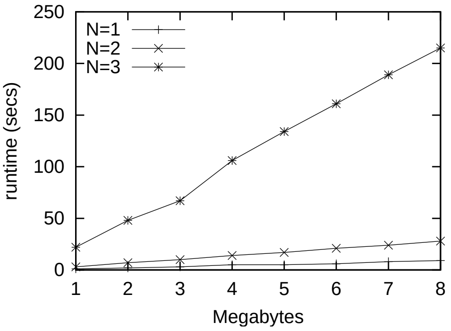


Figure 18. Increasing size of dataset and size of treatments. Datasets generated from the COCOMO model.

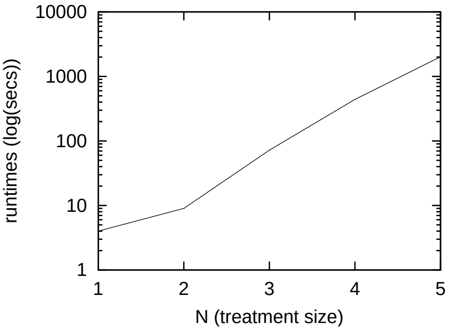


Figure 19. For different treatment sizes N, Increasing size of treatments, keeping data set size constant (3MB). Dataset generated from the COCOMO model.

The exponential impact of increasing treatment size is not necessarily a reason to reject TAR2. Firstly, if very large treatments are required, then a incremental treatment learning approach, such as used in the satellite case study of Figure 16, may suffice.

Secondly, recent experiments with a stochastic treatment learner, TAR3, suggest that linear-time treatment learning may be possible. TAR3 uses the confidence1 distribution as a probability distribution. Treatments are built at random by selecting from attribute ranges at random according to that distribution (so attribute ranges with high confidence1 values tend to get selected more often). Preliminary results with that method are most encouraging (Hu, 2003).

Thirdly, if most domains don't need large treatments, then this exponential impact will not be seen in practice. Elsewhere (Menzies and Singh, 2003), we have made an average case mathematical analysis of the ratio of the odds of a domain narrow funnels to the odds of larger funnels. Under a wide variety of

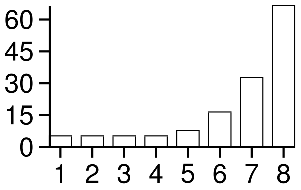


Figure 20. A hypothetical confidence1 frequency distribution with a large left tail that is inconsistent with narrow funnels. Note: yet to be observed in any example set.

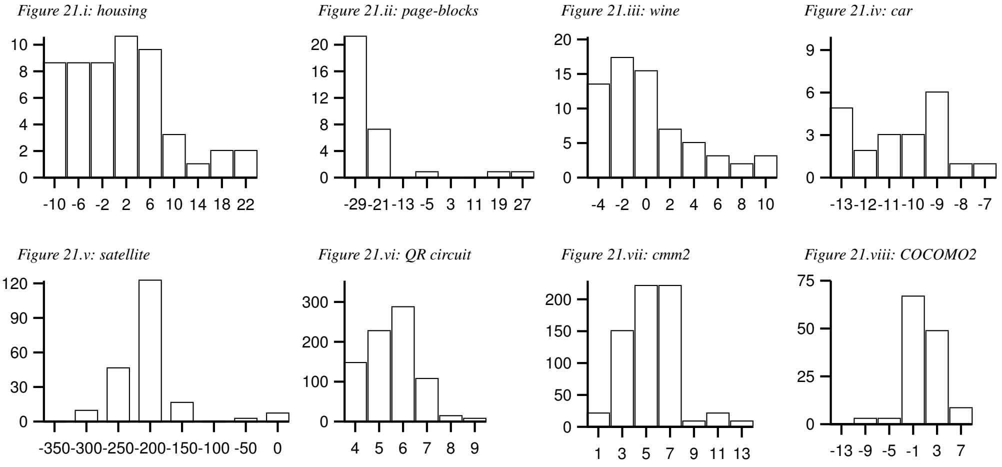


Figure 21.i: housing
Figure 21.ii: page-blocks
Figure 21.iii: wine
Figure 21.iv: car
Figure 21.v: satellite
Figure 21.vi: QR circuit
Figure 21.vii: cmm2
Figure 21.viii: COCOMO2

Figure 21. Confidence1 distributions seen in eight domains. Y-axis is the number of times a particular confidence1 was seen. Top row comes from datasets taken from the UC Irvine repository. Bottom row were generated from other domains discussed in this article.

assumptions, the same effect holds: the odds of narrower funnels are millions of times more likely that wider funnels (Menzies and Singh, 2001). Such a statistical analysis represents an average case result and may not apply to a particular domain. What would be useful would be some kind of assessment tool that checks if this average case statistical result applies to a particular domain.

The confidence1 distribution can be used to test for narrow funnels. Domains that contain such funnels would exhibit the following property: a small number of variables within the funnel exert a disproportionately large influence on the overall behavior of the system. A test for such variables is to check for *small right tails* in the confidence1 distributions. Figure 8 has such a *small right tail*; i.e. the bulk of the distribution lies away from the maximum value. Distributions with a *large right tail* such as Figure 20 are not consistent with narrow funnels. Figure 21 shows the confidence1 distributions seen in eight example sets: four from the UC Irvine repository and some of the other domains described above. Note that in all cases, the distribution has a small right tail; i.e. a small number of variables exert a disproportionately large influence on the overall behavior of the system. In all, we have applied TAR2 to 20 domains: the ones discussed in this paper and others not shown for space reasons. In none of those domains have we observed a large right tail.

7. Conclusion

The minimal theories of TAR2 will be inadequate if domains contain complex relationships. Domains with narrow funnels are not complex: the key controllers for the whole space are merely the few variables in the funnel.

The TAR2 association rule learner is both a test and an application of funnel theory. TAR2 offers two tests for narrow funnels. Firstly, a confidence1 distribution with a small right tail is consistent with a domain containing narrow funnels. Secondly, if a domain contains narrow funnels, then TAR2 should be able to generate adequate controllers and monitors for that domain. All the domains we have seen to date have these two features.

The open issue is how many other domains lack complex relationships. Based on around 20 case studies with TAR2 (some of which were reported above), Holte's prior work with 1R, and the wrapper studies of Kohavi and John, we have some empirical reasons to believe that many domains are not complex. Also, we have theoretical reasons for believing that narrow funnels are common enough (Menzies and Cukic, 2000b; Menzies et al., 2000; Menzies and Cukic, 2000a; Menzies and Singh, 2001) that TAR2 will often suffice.

The success of such a simple algorithm such as TAR2 suggests that it can be fruitful to *first* try lightweight methods *before* exploring heavyweight methods. We hence advocate using TAR2 as a preprocessor to other, more elaborate schemes.

To download and compile a treatment learner, see `http://unbox.org/wisp/tags/tar/`.

Acknowledgements

This research was conducted at West Virginia University, Portland State University, and the University of British Columbia partially under NASA contract NCC2-0979. The work was partially sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA IV&V Facility. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not constitute nor imply its endorsement by the United States Government.

References

Abts, C., B. Clark, S. Devnani-Chulani, E. Horowitz, R. Madachy, D. Reifer, R. Selby, and B. Steece: 1998, 'COCOMO II Model Definition Manual'. Technical report, Center for Software Engineering, USC,. http://sunset.usc.edu/COCOMOII/cocomox.html#downloads.

Agrawal, R. and R. Srikant: 1994, 'Fast Algorithms for Mining Association Rules'. In: *Proceedings of the 20th International Conference on Very Large Databases*. Available from http://www.almaden.ibm.com/cs/people/ragrawal/papers/vldb94_rj.ps.

Bay, S. and M. Pazzani: 1999, 'Detecting Change in Categorical Data: Mining Contrast Sets'. In: *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*. Available from http://www.ics.uci.edu/~pazzani/Publications/stucco.pdf.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone: 1984, 'Classification and Regression Trees'. Technical report, Wadsworth International, Monterey, CA.

Cai, C., A. Fu, C. Cheng, and W. Kwong: 1998, 'Mining Association Rules with Weighted Items'. In: *Proceedings of International Database Engineering and Applications Symposium (IDEAS 98)*. Available from http://www.cse.cuhk.edu.hk/~kdd/assoc_rule/paper.pdf.

Clancy, D. and B. Kuipers: 1997, 'Model Decomposition and Simulation: A component based qualitative simulation algorithm'. In: *AAAI-97*.

Crawford, J. and A. Baker: 1994, 'Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems'. In: *AAAI '94*.

DeKleer, J.: 1986, 'An Assumption-Based TMS'. *Artificial Intelligence* **28**, 163–196.

Feather, M., S. Cornford, and T. Larson: 2000, 'Combining the Best Attributes of Qualitative and Quantitative Risk Management Tool Support'. In: *15th IEEE International Conference on Automated Software Engineering, Grenoble, France*. pp. 309–312.

Feather, M. and T. Menzies: 2002, 'Converging on the Optimal Attainment of Requirements'. In: *IEEE Joint Conference On Requirements Engineering ICRE'02 and RE'02, 9-13th September, University of Essen, Germany*. Available from http://menzies.us/pdf/02re02.pdf.

Holte, R.: 1993, 'Very Simple Classification Rules Perform Well on Most Commonly Used Datasets'. *Machine Learning* **11**, 63.

Hu, Y.: 2003, 'Treatment Learning: Implementation and Application'. Masters Thesis, Department of Electrical Engineering, University of British Columbia.

Josephson, J., B. Chandrasekaran, M. Carroll, N. Iyer, B. Wasacz, and G. Rizzoni: 1998, 'Exploration Of Large Design Spaces: an Architecture and Preliminary Results'. In: *AAAI '98*. Available from http://www.cis.ohio-state.edu/~jj/Explore.ps.

Kohavi, R. and G. H. John: 1997, 'Wrappers for Feature Subset Selection'. *Artificial Intelligence* **97**(1-2), 273–324.

Liu, B., W. Hsu, and Y. Ma: 1998, 'Integrating classification and association rule mining'. In: *KDD*. pp. 80–86. Available from http://citeseer.nj.nec.com/liu98integrating.html.

Lutz, R. and R. Woodhouse: 1999, 'Bi-directional Analysis for Certification of Safety-Critical Software'. In: *1st International Software Assurance Certification Conference (ISACC'99)*. Available from http://www.cs.iastate.edu/~rlutz/publications/isacc99.ps.

Madachy, R.: 1997, 'Heuristic Risk Assessment Using Cost Factors'. *IEEE Software* **14**(3), 51–59.

Menzies, T. and P. Compton: 1997, 'Applications of Abduction: Hypothesis Testing of Neuroendocrinological Qualitative Compartmental Models'. *Artificial Intelligence in Medicine* **10**, 145–175. Available from `http://menzies.us/pdf/96aim.pdf`.

Menzies, T. and B. Cukic: 2000a, 'Adequacy of Limited Testing for Knowledge Based Systems'. *International Journal on Artificial Intelligence Tools (IJAIT)*. Available from `http://menzies.us/pdf/00ijait.pdf`.

Menzies, T. and B. Cukic: 2000b, 'When to Test Less'. *IEEE Software* **17**(5), 107–112. Available from `http://menzies.us/pdf/00iesoft.pdf`.

Menzies, T., B. Cukic, H. Singh, and J. Powell: 2000, 'Testing Nondeterminate Systems'. In: *ISSRE 2000*. Available from `http://menzies.us/pdf/00issre.pdf`.

Menzies, T., S. Easterbrook, B. Nuseibeh, and S. Waugh: 1999, 'An Empirical Investigation of Multiple Viewpoint Reasoning in Requirements Engineering'. In: *RE '99*. Available from `http://menzies.us/pdf/99re.pdf`.

Menzies, T. and Y. Hu: 2001a, 'Constraining discussions in requirements engineering'. In: *First International Workshop on Model-based Requirements Engineering*. Available from `http://menzies.us/pdf/01lesstalk.pdf`.

Menzies, T. and Y. Hu: 2001b, 'Reusing models for requirements engineering'. In: *First International Workshop on Model-based Requirements Engineering*. Available from `http://menzies.us/pdf/01reusere.pdf`.

Menzies, T. and Y. Hu: 2002, 'Agents in a Wild World'. In: C. Rouff (ed.): *Formal Approaches to Agent-Based Systems, book chapter*. Available from `http://menzies.us/pdf/01agents.pdf`.

Menzies, T. and Y. Hu: 2003, 'Data Mining for Very Busy People'. In: *IEEE Computer*. Available from `http://menzies.us/pdf/03tar2.pdf`.

Menzies, T. and H. Singh: 2001, 'Many Maybes Mean (Mostly) the Same Thing'. In: *2nd International Workshop on Soft Computing applied to Software Engineering (Netherlands), February*. Available from `http://menzies.us/pdf/00maybe.pdf`.

Menzies, T. and H. Singh: 2003, 'Many Maybes Mean (Mostly) the Same Thing'. In: M. Madravio (ed.): *Soft Computing in Software Engineering*. Springer-Verlag. Available from `http://menzies.us/pdf/03maybe.pdf`.

Menzies, T. and E. Sinsel: 2000, 'Practical Large Scale What-if Queries: Case Studies with Software Risk Assessment'. In: *Proceedings ASE 2000*. Available from `http://menzies.us/pdf/00ase.pdf`.

Parkes, A.: 1999, 'Lifted Search Engines for Satisfiability'.

Quinlan, R.: 1992, *C4.5: Programs for Machine Learning*. Morgan Kaufman. ISBN: 1558602380.

Rymon, R.: 1993, 'An SE-tree based Characterization of the Induction Problem'. In: *International Conference on Machine Learning*. pp. 268–275.

Rymon, R.: 1994, 'An SE-tree-based Prime Implicant Generation Algorithm'. In: *Annals of Math. and A.I., special issue on Model-Based Diagnosis*, Vol. 11. Available from `http://citeseer.nj.nec.com/193704.html`.

Singer, J., I. P. Gent, and A. Smaill: 2000, 'Backbone Fragility and the Local Search Cost Peak'. *Journal of Artificial Intelligence Research* **12**, 235–270.

Wang, K., Y. He, D. Cheung, and F. Chin: 2001, 'Mining confident rules without support requirement'. In: *10th ACM International Conference on Informationand Knowledge Management (CIKM 2001), Atlanta*. Available from `http://www.cs.sfu.ca/~wangk/publications.html`.

Webb, G.: 2000, 'Efficient search for association rules'. In: *Proceeding of KDD-2000 Boston, MA*. Available from `http://citeseer.nj.nec.com/webb00efficient.html`.