Tailoring rulesets to misclassification costs

Jason Catlett

Room 2T-412, AT&T Bell Laboratories
600 Mountain Ave, Murray Hill, NJ 07974, USA
Email: catlett@research.att.com

Abstract

This paper studies the capabilities obtained by modifying Quinlan's [9] C4.5 programs for inducing decision trees and rules to permit the specification of unequal misclassification costs for binary classification tasks. Setting this cost value allows important parameters such as the percentage classified as a given class to be moved over their complete range. In some applications such parameters require precise control, but a considerable degree of variation appears difficult to suppress, particularly with rules: it is present even in the unmodified versions that treat all errors as equal. Crossvalidation over a range of cost values seems the appropriate way to tune such parameters. Independent of misclassifications costs, the ability to explore a spectrum of classifiers can considerably assist exploratory data analysis, delivering clearer rules than the standard version may provide. These conclusions are illustrated on a simulated version of the game Blackjack.

1 Introduction and motivation

Most classifiers in machine learning (ML) are built with the aim of minimizing the number of errors on unseen examples from the same distribution as the training set [6], but statisticians have long taken for granted the situation where the misclassification costs differ [1], and where the number of a certain type of error must be limited [4]. In the two-class case the relative costs of the two types of error may be summarized as a single positive real which I will call the *loss ratio* (LR). A LR of 1 expresses equal costs. This paper examines modifications to programs for inducing rules that accept a specific loss ratio.

In many ML applications one class is comparatively rare and of special interest: in manufacturing and medicine, periods during which a system is behaving abnormally; in information retrieval (IR), pieces of text relevant to some particular category; in marketing, customers who take certain actions (desirable or undesirable). The misclassification costs may not be known and it may be difficult even to estimate them beyond identifying which is the more serious, but the minimum acceptable certainty factor of a rule for the rare class can be well below the 50% level, where standard ML algorithms will cease to generate rules for that class. The goal of induction during data analysis may be simply to explore the areas of the attribute space where the rare class is comparatively more common, even if it nowhere reaches a majority.

The next section describes a domain and the common basis for the experiments. Section 3 exhibits three rulesets built from the same training set with various values of the loss ratio parameter, to illustrate its benefit to exploratory data analysis [11]. Section 4 uses ROC analysis to show the high variance in parameters such as the false alarm rate. The implementation is sketched briefly in [7].

2 Description of domain and experiments

The domain this paper uses as an illustration is based on the popular casino game *Blackjack*, also called *twenty-one* [8]. This was chosen because the induced rulesets are small and comprehensible to a wide audience. For a case study of the method's application to large IR tasks, see [7]. Several departures from the real game are made for convenience. Cards are dealt from a notionally infinite deck; initially two to the player and one to the dealer, which the player can also see. Kings, queens and jacks count as 10 points;

Class	Rule #	% Coverage	% Error	ptc	pha	duc	phs
p	1	6.1	0.0	11—	y		
p		6.1	0.0				
d	2	85.9	33.4		n		
d		85.9	33.4				
d	def	8.0	40.2	ptc	pha	duc	phs
	all	100.0	32.0				

Table 1: A ruleset with low false-positive rate

aces as 1 or 11 at the holder's choice; all other cards as their numeric value. With a total of less than 21 the player may choose either to *sit* or *hit* (to decline or request an additional card). If the player ever exceeds 21, the dealer wins. If the player sits with less than 21, a dealer with a total below 17 is required to take extra cards to attain this minimum; if the dealer exceeds 21 or reaches 17 with less than the player's total, the player wins.

To keep the number of classes to two, the following simplifications are made: with blackjack (21 with only two cards) the player wins even if the dealer has the same, but in all other draws the dealer wins. We exclude complicated player options such as splitting, doubling, and insurance.

In this simulation the class we aim to predict is whether the player or dealer will win, given the following four attributes, each preceded by their abbreviated names: `ptc`: the total of the player's first two cards (counting aces as 1); `pha`: whether the player has at least one ace; `duc`: the dealer's "up" card; `phs`: which of the following two equiprobable strategies the player follows: to sit immediately, or to hit and continue doing so until the hand totals 17 or more. We do not attempt to predict whether the player will hit or sit, although some guidelines can be extracted from the rules as to which would be more advantageous in certain situations.

All the experiments in this paper use this domain, on training sets of size 1,000 except where otherwise specified. Most of the experiments use the original C4.5, but where a loss ratio other than 1 is specified, a version modified as was used, as sketched in [7]. A single independent test set of 10,000 examples was used across all experiments; training sets were generated independently. The class `p` (player wins) is considered the positive class. The test set contained 6,480 negative examples, so the default error from a classifier that always gives the majority class is 35.2%.

3 Varying the LR for exploratory data analysis

To exhibit the benefit to exploratory data analysis of varying the LR, I have chosen three rulesets built from the same training set using different values for the loss ratio. They have been automatically reformatted from C4.5's output into the form of *decision tables*, which many people find easier to understand as a whole than textual rules. Table 1 is the simplest; this ruleset is delivered by very small values for LR, such as 1/32. Attribute names are listed in the rightmost four columns. The columns indicate (left to right)

- the class of the rule's conclusion (`p` for player, `d` for dealer),
- the rule's identifying number (which in this case happens to be 1, but in general the ordering is arbitrary),
- the percentage of examples that the rule *covers* (i.e. those examples whose attribute values satisfy all the rule's conditions),
- the error rate on those examples matching the rule (Rule one is "sure-bet" at 0.0% errors), and

Class	Rule #	% Coverage	% Error	ptc	pha	duc	phs
p	9	6.5	7.6	9—	y	5—	
p	41	1.2	50.0	11-11		6- 8	
p	7	2.8	37.9	7—	y	—4	
p	5	1.3	76.9	5- 6	y	— 9	
p	62	2.8	37.9	—15		6- 6	s
p	43	1.2	75.0	13-14		6- 7	
p	1	0.2	50.0	— 2			h
p	31	0.9	44.4	10-10		5- 8	h
p	95	4.7	25.0	20—		2—	s
p	61	2.3	45.8	16-19		4- 5	s
		23.9	32.8				
d	75	2.6	29.6	12-19		8- 8	s
d	3	0.8	62.5	3- 3			h
d	68	0.7	42.9	— 5	n	7—	s
d	27	0.9	44.4	9- 9		— 8	h
d	42	31.9	22.4	12—			h
d	87	17.5	28.5	6-19	n	9—	
d	64	10.0	37.3		n	— 3	
d	8	0.4	25.0	7- 8	y	5- 9	
d	10	1.1	18.2	— 8	y	10 —	
		65.9	27.5				
d	def	10.4	37.7	ptc	pha	duc	phs
	all	100.0	29.8				

Table 2: A ruleset built by the unmodified C4.5

Class	Rule #	% Coverage	% Error	ptc	pha	duc	phs
d	28	2.7	14.3	17-19		9—	h
d	22	1.8	16.7	14-18		9- 9	
d	20	1.0	20.0	— 9	n	9- 9	
d	3	0.4	25.0	4- 8	y	10—	h
d	9	0.8	12.5	15-19		— 1	s
d	6	0.7	28.6	— 8	n	— 1	
		7.4	17.4				
p	2	13.7	32.9		y		
p	10	4.9	26.0	20—			s
p	29	7.8	62.5	17—		2—	s
p	12	37.1	61.8	—19		2- 8	
p	27	22.4	73.4	—16			
		85.9	58.2				
d	def	6.7	13.0	ptc	pha	duc	phs
	all	100.0	52.2				

Table 3: A ruleset with low false-negative rate

- the rule conditions: a blank slot indicates the attribute did not appear in any condition of the rule (this is sometimes called a *don't care*); a numeric range indicates the attribute value must be in that range (this is actually two conditions except in the case of ranges open at one end); and a discrete value indicates the condition is a test of equality for that value.

Thus the first rule could be read as "if the player's total is 11 or more and the player has an ace, then the player will win," reflecting the simplification assumed above. The class p (player will win) is given in the first column of the next line. The next line, Rule 2, simply states that if the player has no ace, then the dealer will win (the class is again given in the subsequent line). This rule is wrong in about a third of the cases. The second last line specifies *default class* as d. This *default rule* is used in the 8% of cases where neither of the above rules holds; it has a higher error rate. The last line specifies the total error rate of the classifier in the third column of the header line; in this case it is 32.0%.

The repetition of information in columns three and four is due to the fact that only one rule appears for each class in this ruleset. Table 2 (which was produced by the unmodified C4.5, notionally a LR of 1.0) has ten rules for class p; the figures to the right of the p give the total coverage of the rules for that class and the error rate on those examples covered. Where rules are not mutually exclusive, and example is counted under the first rule it satisfies. C4.5 groups all rules for a class together (with the possible exception of the default rule, as in Table 3 (LR=4)), so it effectively orders the classes: the rules for the second class are considered only if none of the rules for the first class is satisfied.

The error rates and coverage figures are based on an independent test set, which shows some rules to be performing very badly; Holte, Acker & Porter [5] call this the *problem of small disjuncts*. In this case all the rules with error rate above 40% have a coverage of less than 2.5%. Changing the value of LR seems to eradicate them (almost all the disjuncts in Table 3 covered less than this percentage, but are much more accurate), at the expense of a greatly increased error rate in the rules for the other class. Still, as argued by Segal, Etzioni, & Riddle [10], some applications are best served by a few accurate rules, even if their coverage is small.

The main point of this section is that varying the LR parameter can yield useful, enlightening rules that may otherwise be missed. The clearest example is the rule for blackjack (Rule 1, the only p-class rule in Table 1): on many training sets it does not appear in the ruleset built by the standard C4.5; in its stead is a less accurate rule of slightly wider coverage with a lower threshold for the player's total count and a condition on the dealer's count (Rule 9 in Table 2). (Its appearance becomes more frequent as training sets become larger.) Another example is the first rule in Table 3, which predicts that if a player hits with high total when the dealer is showing a 9 or 10 (making it unlikely his hand will exceed 21), then the dealer is very likely to win. This rule is extremely obvious and intuitive to anyone who has played the game, yet no comparable rule appears in the original C4.5 ruleset. Because of the "masking" effect of higher-priority rules, the conjecture that the original ruleset may nonetheless be getting these cases right could not be dismissed at a glance, but this only highlights the point: altering the loss ratio can make some rules clearer, in both senses: easier to understand, and more accurate and clear-cut. This capability can be very helpful in exploratory data analysis.

The simplicity of these rulesets is bought at the expense of less specific information on the class-probability value assigned to an unseen example: a more complex classifier can choose from a larger set of values. For example, Table 1 partitions the attribute space into three regions, each covered by one of the three rules (the blackjack rule, the rule that an aceless player will lose, and the default rule), whereas Table 2 has many more rules for each class. Each rule is associated with a class-probability value, just as a decision tree gives a separate class-probability at each leaf. A large set of possible class-probability values will be important in applications requiring an accurate estimate of each example's class-probability, but not in those where the only question of interest is whether the value is above or below a certain threshold. In the latter case the clarity offered by the simpler model may make them attractive. If the former case is attacked with decision trees, modifications along the lines of those in Section 6 are needed, as illustrated by the following case. Suppose the ideal decision tree has a probability value for the positive class of 0.4 at one leaf and 0.2 at its only sibling leaf. Pruning methods typically collapse these into the parent node, making it a leaf with a value of 0.3 (assuming the leaves were equally populated). The critical value here is

normally 0.5; this can be adjusted by changing LR, but the question of whether it is desirable to simplify the model by merging parts of it on either side of the critical value is a separate question whose answer depends on the application.

4 ROC Analysis

For applications where it is necessary to limit the number of a certain type of error, we use methods from *ROC analysis* [4]. The theory of *Receiver Operating Characteristics* originated in WWII research on radar detectors, aimed at finding the best tradeoff between false alarms and missed signals.

When classes are restricted to two values, called say positive and negative, a classifier's performance on a set of data can be summarized by counting each element as one of four possible cases: the true positives (TP) and true negatives (TN), where the classifier is correct, the false positives (FP), where a negative example is erroneously classified as positive, and false negatives (FN), where a positive example is erroneously classified as negative. The traditional ML measure of error rate is of course (FP+FN)/(TP+TN+FP+FN). The expression TP/(TP + FN), which is the fraction of positive examples that the classifier judges positive, is called the *True Positive Rate* (TPR), also called the *hit rate* in ROC theory, *recall* in IR, and *sensitivity* in medicine. The expression FP/(FP + TN) which is the fraction of negative examples that the classifier judges positive, is called the *False Positive Rate* (FPR), also called the *false alarm rate* in ROC, *fallout* in IR, and is equal to 1 – *specificity* in medicine.

ROC analysis plots a classifier's performance as a point on the unit square with the FPR on the x-axis and TPR on the y-axis, as shown in Figure 1. The data for this figure is for rulesets from 20 independently generated training sets of 1,000 blackjack examples each, with each of eleven powers of two[1] as the loss ratio parameter, for a total of 220 classifiers. The letters in Figure 1 correspond to the following points.

(A) A classifier that deems everything negative sits at (0,0); one that always says positive is at (1,1); a classifier that makes guesses unrelated to its input sits on the diagonal line between these extremes at a point determined by the frequency with with it says positive. Classifiers below this line are doing worse than guesswork.

(B) A family of near-perfect classifiers inhabits the curve that encompasses all but the top left corner of the unit square. Obviously such levels of accuracy are unattainable in noisy domains such as blackjack.

(T) The classifier in Table 1 is represented by the point above the T1; Table 2 is somewhere below T2, similarly with T3.

(C) The galaxy of points through C (each representing the performance of at least one of the 220 rulesets), has been fitted with a smooth curve (using the S function `lowess` with a parameter of f=0.1) [2]. Note that the curve spans the entire range and is continuously populated, except for a gap between (0,0) and (0, 0.139), the latter being the "sure-bet" rule for blackjack. This rule is dropped only for the smallest values of LR.

(D) The points with diamonds are those rulesets built with a value of 1 for LR: the original C4.5 algorithm. Note the considerable range along the curve this encompasses: (0.052, 0.284) to (0.225, 0.503).

(E4) Those points surrounded by triangles are from a value of 2 for LR. Note the even wider range, with outliers at E5 and E6.

On both trees and rules, the method affords the full range required over the curve. (An exception is the natural gap due to the blackjack rule, which is desirable given the domain.) Performance appears consistent with the original version. I compared this method with the method of duplicating examples of one class in proportion to the loss ratio, which gives poor range and control, and performance is starkly dominated by the LR method. This loss of accuracy may be due to duplication confounding the pruning mechanism.

The major concern revealed by this graph is the large variance observed in the key parameters of TPR and FPR. It is surprisingly large even with the original algorithm, where misclassification costs are assumed

[1] The values used were 32, 16, 8, 4, 2.82843, 2, 1.68179, 1.41421, 1.18921, 1, 0.707107, 0.5, 0.353553, 0.25, 0.125, 0.0625, and 0.03125.

Figure 2: ROC Curve for blackjack domain

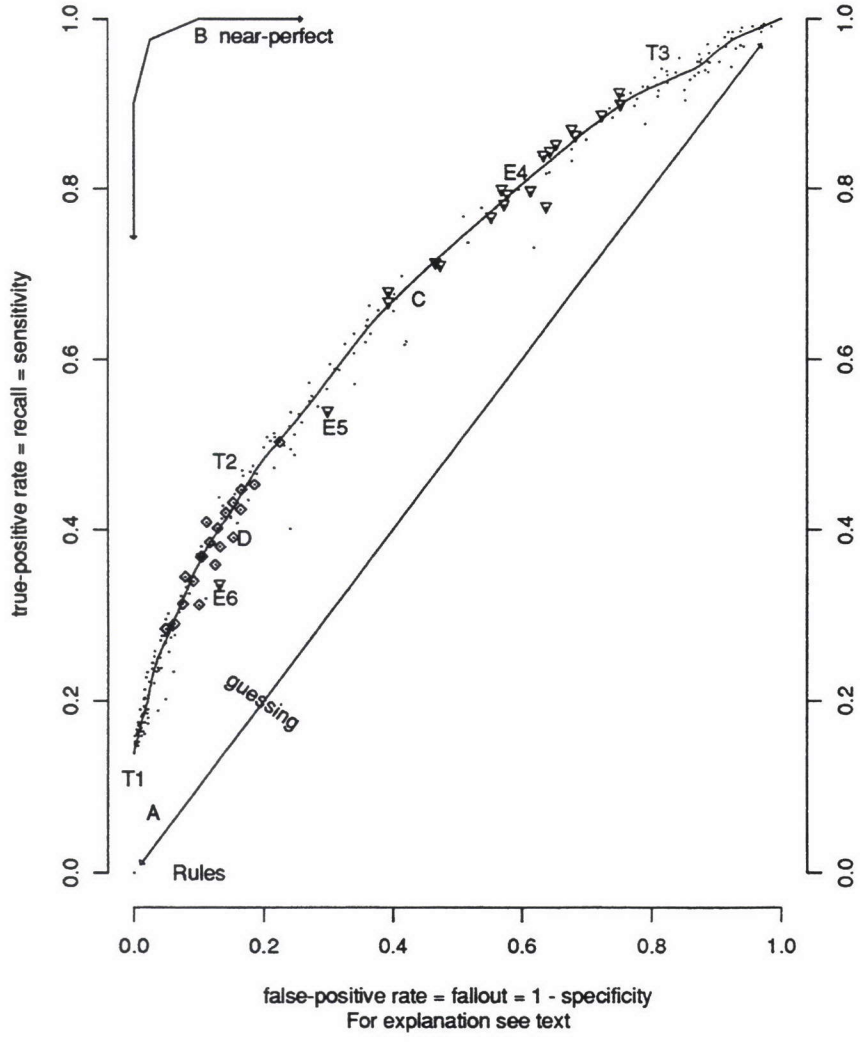

Figure 1: ROC Curve for blackjack

to be equal. The problem appears possibly greater for rules than trees and greater for values for LR other than 1, but is still too strong for some purposes in the original version. I believe that it would be very difficult to alter the algorithm to reduce this variance, and that at least for now we should content ourselves with using crossvalidation to achieve a desired value.

5 Conclusions

Exploring rulesets across a spectrum of values for loss ratio can considerably assist data analysis. An analysis of the performance of the original and modified versions of C4.5 shows that parameters such as false-positive rate exhibit considerable variance.

Acknowledgements

I thank David Lewis and Matthew Partridge for their comments on drafts, Ross Quinlan for the code for C4.5, and Alan Skea for assistance with Blackjack. Donald Michie converted me to the goal of machine learning as data analysis.

References

[1] T.W. Anderson. Multivariate analysis: classification and discrimination. In *International encyclopedia of statistics*, pages 628–635, NY, 1978. Free Press.

[2] Richard A. Becker, John M. Chambers, and Allan R. Wilks. *The new S language: a programming environment for data analysis and graphics.* Wadsworth and Brooks/Cole, Pacific Grove, CA, 1988.

[3] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.

[4] J.P. Egan. *Signal detection theory and ROC analysis.* Academic Press, NY, 1975.

[5] R. C. Holte, L. E. Acker, and B. W. Porter. Concept learning and the problem of small disjuncts. In *Proceedings IJCAI-89*, pages 813–818, 1989.

[6] Pat Langley and Dennis Kibler. Machine learning as empirical science. In *Proceedings of the third european working session on learning*, London, 1988. Pitman.

[7] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *ML-94*, pages 148–156, 1994.

[8] Albert H. Morehead and Geoffrey Mott-smith. *Hoyle's rules of games.* Signet, New York, 1959.

[9] J. Ross Quinlan. *C4.5: Programs for machine learning.* Morgan Kaufmann, 1993.

[10] Patricia Riddle, Richard Segal, and Oren Etzioni. Representation design and brute-force induction in a boeing manufacturing domain. *Applied Artificial Intelligence*, 8:125–147, 1994.

[11] John W. Tukey. *Exploratory data analysis.* Addison-Wesley, Reading, MA, 1977.

[12] Sholom M. Weiss and Casimir A. Kulikowski. *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems.* M. Kaufmann Publishers, San Mateo, Calif., 1991.