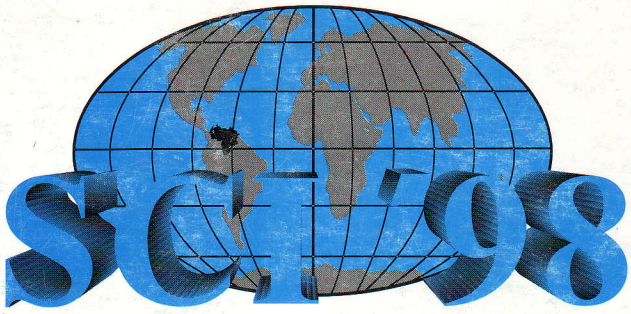
World Multiconference on

SYSTEMICS, CYBERNETICS AND INFORMATICS



July 12-16, 1998
Orlando, Florida

PROCEEDINGS

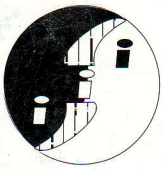
Volume 2

ISAS'98

(4th International Conference on Information Systems, Analysis and Synthesis)

Organized by

IIIS



International Institute of Informatics and Systemics

Member of the International Federation of Systems Research

IFSR

Edited by

Nagib Callaos
Tianruo Yang
José Aguilar

REAL: Real Attribute Learning Algorithm

Julio Michael Stern

Computer Science Department,
University of São Paulo
jstern@ime.usp.br

Celma de Oliveira Ribeiro

Industrial Engineering Department,
University of São Paulo
São Paulo, SP, 01452-990 , Brazil

Marcelo de Souza Lauretto

Supremum Assessoria e Consultoria Ltda.

Fábio Nakano

Supremum Assessoria e Consultoria Ltda.

ABSTRACT

This paper presents REAL, a Real-Valued Attribute Classification Tree Learning Algorithm. Several of the algorithm's unique features are explained by the users' demands for a decision support tool to be used for evaluating financial operations strategies. Compared to competing algorithms, in our applications, REAL presents major advantages:

1. The REAL classification trees usually have smaller error rates.
2. A single conviction (or trust) measure at each leaf is more convenient than the traditional (probability, confidence-level) pair.
3. No need for an external pruning criterion.

Keywords: Learning, Classification Tree, Decision Support, Finance, Operation Strategy.

1. INTRODUCTION

This paper presents the REAL Machine Learning Algorithm, for automatic construction of classification trees with real-valued attributes [2], [8], [10]. The REAL project started as an application to be used at the Brazilian stock market, trying to provide a good algorithm for predicting the adequacy of operation strategies. In this context, the success or failure of a given operation strategy corresponds to different classes, and the attributes are real-valued market indicators. The users' demands for a decision support tool also explain several of the algorithm's unique features.

The project began testing several Machine Learning tools presented at the ESPRIT-StatLog project [6]; The TDIDT - Top Down Induction Decision Tree - software CAL5 [7] proved to be specially useful for the application we had in mind. The CAL5 algorithm had a strong influence on our project, and we use it as our main performance benchmark. In our applications the REAL algorithm presented some major advantages:

1. Usually the classification trees have a smaller error rate.
2. The single conviction (or trust) measure at each leave is more convenient, than the traditional (probability, confidence) pair.
3. REAL interval partition procedure stops naturally, eliminating the need of an additional pruning procedure.

2. PROBLEM FORMULATION

The classification problems are stated as an $n \times (m+1)$ matrix A. Each row, $A(i,:)$, represents a different example, and each column, $A(:,j)$, a different attribute. The first m columns in each row are real-valued attributes, and the last column, $A(i,m+1)$ is the example's class. Part of these samples, the training set, is used by the algorithm to generate a classification tree, which is then tested with the remaining examples. The error rate in the classification of the examples in the test set is a simple way of evaluating the classification tree.

3. TREE CONSTRUCTION

Each main iteration of the REAL algorithm corresponds to the branching of a terminal node in the tree. The examples at that node are classified according to the value of a selected attribute, and new branches generated to each specific interval. The partition of a real-valued attribute's domain in adjacent non-overlapping (sub) intervals is the discretization process. Each main iteration of REAL includes:

1. The discretization of each attribute, and its evaluation by a loss function.
2. Selecting the best attribute, and branching the node accordingly.
3. Merging adjacent intervals that fail to reach a minimum conviction threshold.

4. CONVICTION AND LOSS FUNCTIONS

Given a node of a given class with n examples, k of which are misclassified and $(n-k)$ of which are correctly classified, we needed a single scalar parameter, cm, to measure the probability of misclassification and its confidence level. Such a simplified conviction (or trust) measure was a demand of REAL users operating at the stock market.

Let q be the misclassification probability for an example at a given node, let $p = (1-q)$ be the probability of correct classification, and assume we have a Bayesian distribution for q, namely

$$D(c) = \Pr(q \le c) = \Pr(p \ge 1-c)$$

We define the conviction measure: $100*(1-cm)\%$, where

$$cm = \min\ c \mid \Pr(q \le c) \ge (1-g(c))$$

and $g(\ )$ is a monotonically increasing bijection of $[0,1]$ onto itself. From our experience in the stock market application we learned to be extra cautious about making strong statements, so we make g( ) a convex function.

In this paper $D(c)$ is the posterior distribution for a sample taken from the Bernoulli distribution, with a uniform prior for q:

$$B(n,k,q) = comb(n,k) * q^k * p^{n-k}$$

$$D(c,n,k) = betainc(c,k+1,n-k+1)$$
$$= \int_{q=0}^{c} B(n,k,q) \ / \ \int_{q=0}^{1} B(n,k,q)$$

Also in this paper, we focus our attention on

$$g(c) = g(c,r) = c^r,\ r \ge 1.0$$

we call r the convexity parameter.

With these choices, the posterior is the easily computed incomplete beta function, and cm is the root of the monotonically decreasing function:

$$cm(n,k,r) = c \mid f(c) = 0,$$

$$f(c) = 1 - g(c) - D(c,n,k)$$
$$= 1 - c^r - betainc(c,k+1,n-k+1)$$

Finally, we want a loss function for the discretizations, based on the conviction measure. In this paper we use the overall sum of each example classification conviction, that is, the sum over all intervals of the interval's conviction measure times the number of examples in the interval:

$$loss = \sum_i n_i * cm_i$$

5. DISCRETIZATION PROCEDURE

Given an attribute, the first step of the discretization procedure is to order the examples in the node by the attribute's value, and then to join together the neighboring examples of the same class. So, at the end of this first step, we have the best ordered discretization for the selected attribute with uniform class clusters.

In the subsequent steps, we join intervals together, in order to decrease the overall loss function of the discretization. The gain of joining J adjacent intervals, $I_{h+1}, I_{h+2}, \ldots, I_{h+J}$, is the relative decrease in the loss function

$$gain(h,j) = \sum_j loss(n_j, k_j, r) \; - loss(n,k,r)$$

where $n = \sum_j n_j$ and k counts the minorities' examples in the new cluster (at the second step $k_j = 0$, because we begin with uniform class clusters).

At each step we perform the cluster joining operation with maximum gain. The discretization procedure stops when there are no more joining operations with positive gain.

The next examples show some clusters that would be joined together at the first step of the discretization procedure. The notation $(n,k,m,r,\pm)$ means that we have two uniform clusters of the same class, of size n and m, separated by a uniform cluster of size k of a different class; r is the convexity parameter, and + (−) means we would (not) join the clusters together.

(2,1,2,2,+)

(6,2,7,2,-) (6,2,8,2,+) (6,2,23,2,+) (6,2,24,-)

(7,2,6,2,-) (7,2,7,2,+) (7,2,42,2,+) (7,2,43,2,-)

(23,3,23,2,-) (23,3,43,2,-) (23,3,44,2,+)

(11,3,13,3,-) (11,3,14,3,+) (11,3,39,3,+) (11,3,40,3,-)

(12,3,12, 3,-) (12,3,13,3,+) (12,3,54, 3,+) (12,3,55,3,-)

In these examples we see that it takes extreme clusters of not only large but also balanced size, n and m, to "absorb" the noise or impurity in the middle cluster of size k. A larger convexity parameter, r, implies a larger loss at small clusters, and therefore makes it easier for sparse impurities to be absorbed.

6. BRANCHING AND MERGING

For each terminal node in the tree, we

1. perform the discretization procedure for each available attribute,
2. measure the loss function of the final discretization,
3. select the minimum loss attribute, and
4. branch the node according this attribute discretization.

If no attribute discretization decreases the loss function by a numerical precision threshold $\varepsilon > 0$, no branching takes place.

A premature discretization by a parameter selected at a given level may preclude further improvement of the classification tree by the branching process. For this reason we establish a conviction threshold, ct, and after each branching step we merge all adjacent intervals that do not achieve $cm < ct$. To prevent an infinite loop, the loss function value assigned to the merged interval is the sum of the losses of the merging intervals. At the final leaves, this merging is undone. The conviction threshold naturally stops the branching process, so there is no need for an external pruning procedure, like in most TDIDT algorithms.

7. COMPUTER IMPLEMENTATION

For the numerical tests, presented in section 9, we use a straightforward implementation of REAL. This implementation takes about 1 minute to train and test the classification tree for each of those problems, on a Pentium-200 IBM-PC.

REAL was implemented as a portable code in C++, and the final application uses a Microsoft VB-4.0 graphical user interface. In the straightforward implementation, REAL spends most of the execution time computing the function $cm(n,k,r)$. We can greatly accelerate the algorithm by using precomputed tables of $cm(n,k,r)$ values for small n, and precomputed tables of $cm(n,k,r)$ polynomial interpolation coefficients for larger n. To speed up the algorithm we can also restrict the search for join operations at the discretization step to small neighborhoods, i.e. to join only $3 \le J \le J_{max}$ clusters: Doing so will expedite the algorithm without any noticeable consistent degradation.

8. MARKET OPERATION STRATEGIES

A market operation strategy is a predefined set of rules determining an operator's actions in the market. The strategy shall have a predefined criterion for classifying a strategy application as failure or success [9].

As a simple example, let us define the strategy $buysell(t,d,l,u,c)$:

- At time t buy a given asset A, at its price $p(t)$.
- Sell A as soon as:
  1. $t' = t + d$, or
  2. $p(t') = p(t) * (1 + u/100)$, or
  3. $p(t') = p(t) * (1 - l/100)$.
- The strategy application is successful if

$$100 * (p(t')/p(t) - 1) \ge c$$

The parameters u, l, c and d can be interpreted as the desired and worst accepted returns (low and upper bound), the strategy application cost, and a time limit. Figure 1 displays $buysell(t,d,l,u,c)$ applications.

Technical Indicators

Market analysts use several tools to forecast asset prices, including tools based on accounting records, input-output, macro-economic analysis, etc; These are known as fundamentalist analysis tools. Other frequently used tools are "technical indicators". A technical indicator is function of one or more observed market variables. Several providers distribute day and "intraday" data for most of the stock, commodities and derivative markets. At BOVESPA, the São Paulo Stock Exchange, available daily data include, $H(t)$, $L(t)$, $O(t)$, $C(t)$, $M(t)$ and $V(t)$, respectively, the asset highest, lowest, opening, closing and mean price, and the asset transaction volume at day t. An example of technical indicator is an asset opening price relative to its highest price in the previous r days:

$$OH_r(t) = O(t) / \max\{H(t), H(t-1), \ldots, H(t-r)\}$$

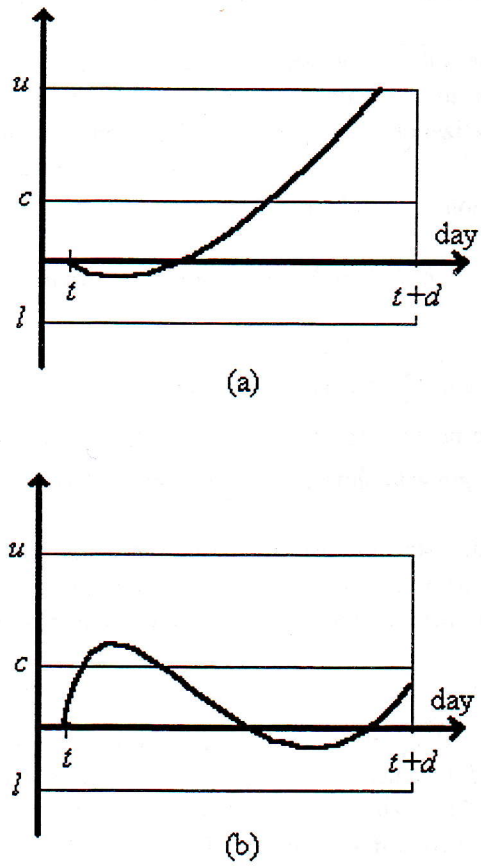


Figure 1: $buysell(t,d,l,u,c)$ strategy application examples

Technical indicators are traditional and widely used market analysis tools [4]. The familiarity of market operators with technical indicators explains why they feel confident with decision support systems providing logical decision rules based on them. The rules and attributes being used for a given classification can be

easily understood (and accepted or dismissed). This confidence translates in a more expedite and effective use of the system. In fact, basing the TDIDT system on technical indicators was a client request.

The core of the trading decision support system is a TDID strategy classification procedure based on the REAL algorithm. Let us consider $buysell(t,d,l,u,c)$ strategy classification examples. In our problems the example at time t has as attributes the technical indicators values at time $t-1$, and its class is the strategy application at time t success or failure based on the information available at time $t+d$. The examples are taken from non overlapping segments of historical time series.

Objective Functions

A classification procedure, applied to a given test set of examples, give us a classification matrix, also known as confusion matrix, like the matrix in table 1, where $n11$ and $n22$ are the number of correctly classified successful and failed applications of the strategy, $n12$ are the successful applications misclassified as failures, and $n21$ the opposite misclassifications. A market operator expects from the decision support system advice to help him detect almost all good opportunities to apply the strategy. When advised to apply the strategy the operator expects it to rarely fail. So our objectives are to, respectively, maximize and minimize the yield and application failure rates, ry and rf:

$$ry = n11/(n11+n12)$$

and

$$rf = n21/(n11+n21).$$

To conciliate these conflicting multiple objectives we define a merit function that can be interpreted as a conservative estimate of the cumulative gain for the $buysell(t,d,l,u,c)$ strategy:

$$merit = c*n11 - l*n21.$$

Actual/Attributed	Success	Failure
Success	$n11$	$n12$
Failure	$n21$	$n22$

Table 1: Classification (confusion) matrix

9. NUMERICAL TESTS

We tested the classification procedures of the previous sections on $buysell(t,d,l,u,c)$ with $d = 5\ days$, $l = 1\%$, $c = 1\%$ and $u = 3\%$, on non overlapping time series segments of two of the most liquid (traded) stocks at BOVESPA: Telebras-PN (TEL4), with aprox. 300 examples 45% successful, and Petrobras-PN (PET4), with aprox. 400 examples 40% successful. We divided our examples in 10 subsets. At each run the algorithm generated a tree using the data in 9 of the subsets, the learning-set, and tested it in the remaining subset, the test-set. The procedure was repeated for the algorithm's parameters in a discrete grid:

REAL: $(r,ct) \in \{1.0,1.5,\ldots,4.0\} \times \{0.1,0.15,0.2,\ldots,0.45\}$.

Cal5: $(S,\alpha) \in \{0.05,0.10,\ldots,0.90\} \times \{0.05,0.10,\ldots,0.90\}$.

NewID: $\phi \in \{0\%,2\%,4\%,\ldots,28\%,30\%\}$.

We also included in the benchmark NewID, a classic TDIDT algorithm, primarily developed for categorical classification, but also capable of handling real attributes [1]. At tables 2 and 3 we show the parameters optimizing the merit function mean, with the corresponding mean values for rf and ry.

For a sensitivity analysis we show similar results in a neighborhood of the optimal parameters in tables 4 and 5.

Algorithm	P^*	$merit$	rf	ry
REAL	$r = 3.5,\ ct = 0.4$	3.8	0.22	0.44
Cal5	$S = 0.3,\ \alpha = 0.1$	3.3	0.35	0.63
NewID	$\phi = 6\%$	2.9	0.25	0.45

Table 2: Optimal parameters for TEL4

Algorithm	P^*	$merit$	rf	ry
REAL	$r = 2,\ ct = 0.3$	4.3	0.24	0.39
Cal5	$S = 0.4,\ \alpha = 0.2$	3.8	0.23	0.35
NewID	$\phi = 8\%$	2.9	0.42	0.28

Table 3: Optimal parameters for PET4

r \ ct	0.35	ct* = 0.40	0.45
3.0	merit = 3.5 rf = 0.18 ry = 0.38	merit = 3.5 rf = 0.26 ry = 0.47	merit = 3.7 rf = 0.30 ry = 0.51
r* = 3.5	merit = 3.5 rf = 0.14 ry = 0.34	merit = 3.8 rf = 0.22 ry = 0.44	merit = 3.5 rf = 0.31 ry = 0.51
4.0	merit = 3.2 rf = 0.27 ry = 0.38	merit = 3.5 rf = 0.24 ry = 0.41	merit = 3.2 rf = 0.27 ry = 0.45

Table 4: REAL – Sensitivity analysis for TEL4

r \ ct	0.25	ct* = 0.30	0.35
1.5	merit = 2.7 rf = 0.30 ry = 0.24	merit = 2.2 rf = 0.36 ry = 0.35	merit = 1.8 rf = 0.40 ry = 0.43
r* = 2.0	merit = 3.0 rf = 0.35 ry = 0.23	merit = 4.3 rf = 0.24 ry = 0.39	merit = 3.1 rf = 0.36 ry = 0.45
2.5	merit = 0.1 rf = 0.94 ry = 0.02	merit = 3.2 rf = 0.36 ry = 0.26	merit = 3.4 rf = 0.26 ry = 0.32

Table 5: REAL – Sensitivity analysis for PET4

10. MORE NUMERICAL TESTS

We also tested REAL and Cal5 using van Cutsem's "emergency voltage conditions" dataset, where Cal5 had the previous best published performance [5]. We optimized the parameters over the same grid of the last section.

The generated trees are tested in two ways:

1. The standard counting of *hits* and *misses*, i.e. correct and incorrect classifications.
2. Eliminating the test examples that fall at leaves that do not achieve the targeted conviction (for REAL) or probability-confidence levels (for CAL5), and also eliminating the test examples with an attribute out-of-range at a node where it is used in the classification tree, and then counting the remaining *Hits* and *Misses*.

For a sensitivity analysis we show similar results in a neighborhood of the optimal parameters in tables 6, 7 and 8.

Algorithm	P*	hit	miss	Hit	Miss
REAL	(r,ct)=(1.5,0.2)	241.7	8.3	230.6	4.1
Cal5	(S,α)=(0.65,0.15)	240.6	9.4	236.2	6.9

Table 6: Optimal parameters for van Cutsem's dataset

r \ ct	0.15	ct* = 0.20	0.25
1.0	hits = 240.5	hits = 239.0	hits = 236.4
r* = 1.5	hits = 240.7	hits = 241.7	hits = 240.3
2.0	hits = 240.1	hits = 239.7	hits = 239.8

Table 7: REAL – Sensitivity analysis for van Cutsem's dataset

S \ α	0.10	α* = 0.15	0.20
0.6	hits = 237.8	hits = 238.3	hits = 239.2
S* = 0.65	hits = 238.5	hits = 240.6	hits = 239.8
0.7	hits = 239.9	hits = 238.8	hits = 239.6

Table 8: Cal5 – Sensitivity analysis for van Cutsem's dataset

11. CONCLUSIONS AND FURTHER RESEARCH

The users wanted a classification tool which gave them "understandable" classifications based on already familiar attributes. REAL provided such a tool. The simplified conviction (or trust) measure of each classification was greatly appreciated by the users when taking real time decisions. REAL performed better than any other decision tree algorithm we had access to, and proved to be a valuable decision support tool for evaluating stock market operation strategies, based on real-valued attributes (technical indicators).

From preliminary results in some other test problems we believe that the advantage of REAL over CAL5 will be greater for noisier data, but that statement requires more extensive numerical testing. We are currently studying the behavior of REAL discretization procedure with alternative loss functions, and also interested in comparing it to different approaches [3].

ACKNOWLEDGMENTS

We are grateful for the support received from DCC-IME-USP – the Computer Science Department of the Mathematics and Statistics Institute of the University of São Paulo, from CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico (grant 30138/91-7), from FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo (grant 96/2341-2), and from The São Paulo Commodities and Futures Exchange - *BM&F*.

We are also grateful to Junior Barrera, Alan M. Durham, Carlos Humes Jr., Flávio S. C. da Silva, Jacob Zimbarg Sobrinho and Carlos A. B. Pereira, from IME-USP, for many useful insights, to Wolfgang Mueller, from the Fraunhofer Institut, for a CAL5 SUN-Sparc executable code, and to Gerd Kock, from GMD-FIRST Berlin, for all the help at Germany.

REFERENCES

[1] R.Boswell. "Manual for NewID v 4.1". The Turing Institute, 1990.

[2] L.Breiman, J.H.Friedman, C.J.Stone. "Classification and Regression Trees." Chapman & Hall, London, 1984.

[3] J.Y.Ching, A.K.C.Wong, K.C.C.Chan. "Class-Dependent Discretization for Inductive Learning from Continuous and Mixed-Mode Data." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17 n.7, pp.641-651, 1995.

[4] R.W.Colby, A.Mayers. "The Encyclopedia of Technical Market Indicators." Dow Jones - Irwin, Homewood, Illinois, 1988.

[5] T. van Cutsem. "Decision Trees for Detecting Emergency Voltage Conditions." *Proc. Second International Workshop on Bulk Power System Voltage Phenomena*, pp.229-240, McHenry, USA, 1991.

[6] D.Michie, D.J.Spiegelhalter, C.C.Taylor. "Machine Learning, Neural and Statistical Classification." Ellis Horwood, New York, 1994.

[7] W.Mueller, F.Wysotzki. "Automatic Construction of Decision Trees for Classification." *Annals of Operations Research* 52, pp.231-247, 1994.

[8] J.R.Quinlan. "Induction of Decision Trees." *Machine Learning* 1, pp.221-234, 1986.

[9] J.M.Stern, F.Nakano, M.S.Lauretto. "Real Attribute Learning for Strategic Market Operation". Tech. Rep. RT-MAC-9606, IME-USP 1996.

[10] S.Unger, F.Wysotzki. "Lernfaehige Klassifizierungssysteme." Akademie Verlag, Berlin, 1981.