Optimal Data Analysis
2010, Vol. 1, Release 1 (September 17, 2010), 144-160

Automated CTA Software: Fundamental Concepts and Control Commands

Robert C. Soltysik, M.S. and Paul R. Yarnold, Ph.D.
Optimal Data Analysis, LLC

Fundamental methodological concepts are reviewed, and automated CTA software commands are annotated.

A decade in the making, commercially-available software which conducts automated hierarchically optimal *c*lassification *t*ree *a*nalysis[1] (CTA) is now being offered to organizations and individuals. This article reviews motivation underlying use of nonlinear models; shortcomings of suboptimal nonlinear methods; CTA methods, model interpretation and reporting; and use of automated software. Software commands and sample code used for solving (un)weighted classification problems are annotated.

"One Size Fits All" versus "Different Strokes for Different Folks"

Examples of linear models broadly used in applied research include models derived via logistic regression, log-linear, and discriminant analysis.[2,3] Regardless of derivation, all linear models share three important, usually unfulfilled assumptions.

First, linear models assume attributes in the model are important for every observation in the sample. In contrast, with nonlinear models different attribute sets can be used with different partitions of the sample: one set of attributes is used for classifying one partition of the sample; another set of attributes is used for classifying a different sample partition; and so forth.

Second, linear models assume the model attributes have identical direction of influence (positively or negatively predictive) for every observation. In contrast, with nonlinear models an attribute may predict class category 1 for one partition of the sample, versus category 0 for a different sample partition.

Third, linear models assume attributes in the model have the same coefficient value (or decision weight) for all sample observations. In contrast, in nonlinear models the coefficient for an attribute may assume two different values for two different sample partitions: for example, 0.2 and -1.8, respectively.

Traditional Nonlinear Methods

Nonlinear classification methods based on general linear model (GLM) or maximum-likelihood (ML) paradigms maximize variance ratios, or the value of the likelihood function for the sample, respectively. Examples of such suboptimal methods are chi-square automatic interaction detection, classification and regression tree analysis, genetic algorithms and neural networks. A problem for GLM-based methods involves satisfying the multivariate normally distributed (MND) assumption required for p to be valid, and a problem for ML-based methods is that model coefficients are biased except in the limit for enormous samples.[2,3] A common issue is that neither GLM nor ML methods explicitly maximize model *accuracy*.[1]

Example of a CTA Model

The first CTA model published involved exploratory research discriminating geriatric (at least 65 years of age) versus nongeriatric adult ambulatory medical patients on the basis of self-reported well-being.[4] Forty geriatric and 85 nongeriatric ambulatory medical patients completed a survey assessing five functional status dimensions (Basic and Intermediate Activities, Mental Health [absence of depression], Social Activity, Quality of Social Interaction), and including five single-item measures assessing health satisfaction, physical limitations, and quantity of social interaction. The CTA model (Figure 1) was constructed manually using ODA software.[1]

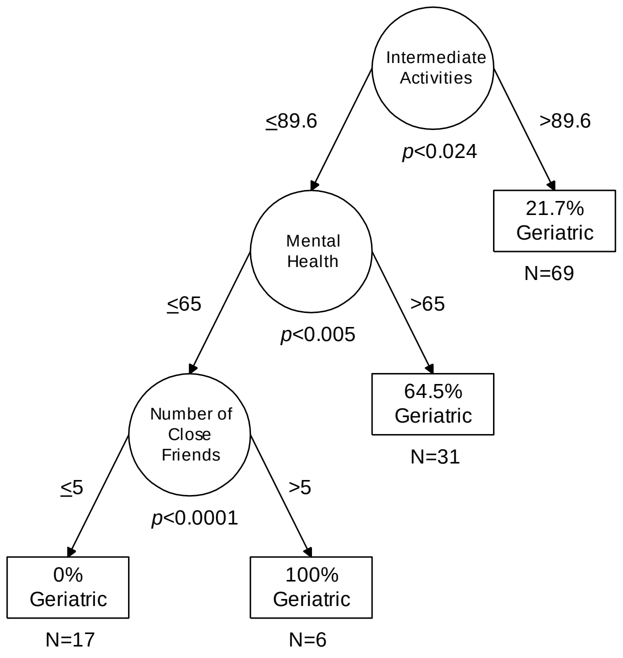

Figure 1: CTA Model Discriminating Geriatric *vs*. Nongeriatric Ambulatory Medical Patients

On first glance a depiction of any classification tree model may appear similar to results obtained by decision analysis (DA), because both methods depict findings using tree-like representations.[4] As seen, CTA models initiate with a *root node*, from which two or more *branches* emanate and lead to other *nodes*: branches indicate pathways through the tree, and all branches ultimately terminate in model *endpoints*. The CTA algorithm determines the attribute subset which predicts the outcome with maximum accuracy, beginning with the attribute which best discriminates the class variable (geriatric status) with maximum accuracy for the total sample. DA *estimates valence and likelihood associated with all possible decision-making strategies and outcomes*. In contrast, CTA *identifies a specific decision-making strategy which maximizes accuracy in predicting a specific outcome*.

Circles represent nodes in this schematic illustration of the CTA model, arrows indicate branches, and rectangles represent model endpoints. Numbers (or words, when attributes are categorical) adjacent to arrows indicate the value of the *cutpoint* (or *category*) for the node. Numbers underneath nodes give the generalized (per-comparison) Type I error rate for the node. The number of observations classified into each endpoint is indicated beneath the endpoint, and the percentage of geriatric observations is given inside the rectangle representing the endpoint.

Using CTA models to classify individual observations is straightforward. Consider a hypothetical person having an Intermediate Activities score=85, a Mental Health score=64, and 7 close friends. Starting with the first node, since the person's Intermediate Activities score is ≤89.6, the left branch is appropriate. At the second node the left branch is again appropriate because the person's Mental Health score is ≤65. Finally, at the third node the right branch is appropriate since the person has more than 5 close friends. The person is classified into the corresponding model endpoint: as seen, all six observations classified into this model endpoint were geriatric. Note that endpoints represent sample strata identified by the CTA model. The probability of being geriatric for this endpoint is $p_{geriatric}$=1 for the sample (in light of the small sample size at this endpoint, it may be more meaningful, depending on the application, to report $p_{geriatric} \geq 6/7$). In this example, had the patient instead reported 5 or fewer close friends, then the left-hand endpoint would be appropriate, with $p_{geriatric}$=0 (i.e., $p_{geriatric} \leq 1/18$).

Some intuitive aspects of CTA models are immediately obvious. For example, model "coefficients" are cutpoints or category descriptions expressed in their natural measurement units. In addition, sample stratification unfolds in a "flow" process which is easily visualized across model attributes. The manner in which CTA handles observations having missing data is also intuitive: linear models drop observations missing data on any attributes in the model, but CTA only drops observations which are missing data on attributes required in their classification. In the present example, imagine an observation having an Intermediate Activities score of 89.6 or greater, but missing data on number of close friends and/or on Mental Health. Using a linear model the observation would be dropped, but using CTA the observation would be classified.

Staging Tables

Staging tables (see Table 1) represent an alternative intuitive representation of CTA findings, and are useful for assigning "severity" or "propensity" scores (weights) to observations based on the findings of the CTA model. The rows of the staging table are simply model endpoints reorganized in increasing order of percent of class 1 (geriatric) membership. Stage is an *ordinal index* of geriatric propensity, and $p_{geriatric}$ is the corresponding *continuous index*: increasing values on either index indicates increasing propensity. Compared to Stage 1 (with $p_{geriatric}$ set at $\leq$1/18, or 0.056), $p_{geriatric}$ is approximately 4-times higher in Stage 2, 12-times higher in Stage 3, and 15-times higher in Stage 4 (with $p_{geriatric}$ set at $\geq$6/7, or 0.857).

To use the table to stage geriatric propensity for a given observation, simply evaluate the fit between the observation's data and each stage descriptor. Begin at Stage 1, and work sequentially through stages until identifying the descriptor which is *exactly true* for the data of the observation undergoing staging. Consider the hypothetical person discussed earlier. Stage 1 does not fit because the person has more than five close friends. Stage 2 does not fit because the person's Intermediate Activities score is $\leq$89.6. Stage 3 does not fit because the person's Mental Health score is $\leq$65. The staging table has only one degree of freedom, so through the process of elimination, it is clear that Stage 4 must be appropriate. Because the person has an Intermediate Activities score $\leq$89.6, a Mental Health score $\leq$65, and >5 close friends, Stage 4 clearly fits the data of this hypothetical person.

Table 1: Staging Table for Predicting Geriatric Status

Stage	Intermediate Activities	Mental Health	Close Friends	N	$p_{geriatric}$	Odds
1	$\leq$ 89.6	$\leq$ 65	$\leq$ 5	17	0	$\leq$1:17
2	> 89.6	------	-----	69	.217	1:4
3	$\leq$ 89.6	> 65	-----	31	.645	2:1
4	$\leq$ 89.6	$\leq$ 65	> 5	6	1	$\geq$6:1

Note: Increasing scores on Intermediate Activities indicate increasing adaptability, and increasing scores on Mental Health indicate decreasing depression.

Assessing Model Performance

Performance measures for CTA (and for all ODA methods) are also intuitively appealing, and are derived from a *confusion table*, as indicated for the present example in Table 2. Rows of the confusion table indicate the *actual* class category of any given observation in the *training sample* (used for model development), and columns indicate the class category *predicted* for an observation by the CTA model. For predictions involving the class category status of individual observations in the training sample, when the actual and predicted class categories are identical (e.g., a geriatric person is predicted to be geriatric) then the model is correct; otherwise it is incorrect. Row and column marginal totals (the sum of all table entries in the row or column, respectively) are presented in the borders of the confusion table. For example, for actual class=geriatric, the *row marginal* is 15+26=41. For predicted class=geriatric, the *column marginal* is 11+26=37. Finally, the total sample size which is classified by the model is given in the lower right-hand corner of the table: this total is equal to the sum of row marginals, and also to the sum of column marginals.

Table 2: Confusion Table for the Example CTA Model

	Predicted Class		
Actual Class	Nongeriatric	Geriatric	
Nongeriatric	71	11	82
Geriatric	15	26	41
	86	37	123

Assessing the performance of a CTA (or any classification) model begins by computing five standard epidemiological indices.[1] The first pair of indices assess the ability of the model to accurately classify observations in the different class categories. *Sensitivity* is the likelihood of correctly classifying an observation from Class 1, and is defined as the number of correctly classified Class 1 observations divided by the total number of Class 1 observations: here, 26/41=0.634. *Specificity* is the likelihood of correctly classifying an observation from Class 0, and is defined as the number of correctly classified Class 0 observations divided by the total number of Class 0 observations: 71/82=0.866.

The next set of indices address the accuracy of the model when it is used to make classifications. *Positive predictive value* (PPV) is the likelihood that an observation predicted to be a member of Class 1 is accurately classified (i.e., is in reality a member of Class 1): here, 26/37=0.703. *Negative predictive value* (NPV) is the likelihood that an observation predicted to be a member of Class 0 is accurately classified: here, 71/86=0.826.

Finally, overall accuracy, or *percentage accuracy in classification* (PAC), is 100% times the number of correctly classified observations divided by the total number of observations classified by the model: 100% x (71+26)/123=78.9%. In the literature, sensitivity, specificity, PPV and NPV are typically multiplied by 100% in order to report all five indices in a common, familiar metric, and because the focus of CTA (and all statistical models in the optimal data analysis paradigm) is predictive accuracy rather than probabilistic likelihood.[1,5]

Summarizing a confusion table is a methodic, straightforward process, as illustrated for the present example: Using the CTA model, a total of 30.1% [100% x (26+11)/123] of the sample is predicted to be geriatric. These predictions are correct 70.3% [100% x PPV] of the time, and correctly identify 63.4% [100% x sensitivity] of all geriatric observations. Also, 82.6% [100% x NPV] of the model-based predictions that an observation is nongeriatric are correct, and correctly classify 86.6% [100% x specificity] of all the nongeriatric observations. Overall, the model correctly classified 78.9% [PAC] of the observations in the sample.

Foregoing indices are bounded by 0 and 1 (or, equivalently, between 0% and 100%), and reference the *absolute predictive capacity* of a classification model. The ultimate objective is for all of these indices to reach their theoretical upper limit of 100% correct prediction. However, in the likely event that a statistical model fails to achieve perfect prediction, statistical criteria are used to assess the performance of CTA (and other) models, in terms of their *predictive capacity relative to chance*.

Copyright 2010 by Optimal Data Analysis, LLC
2155-0182/10/$3.00

Effect Size for Sensitivity (ESS)

None of the five absolute performance indices are normed relative to chance, or have an associated exact *p* value.[1] Accordingly, the performance of all models in the optimal data analysis paradigm, including CTA, is summarized using the *effect strength for sensitivity* (ESS) statistic, a normed index ranging between 0 (representing the level of classification accuracy expected by chance) and 100 (representing errorless classification).[1]

The formula for computing ESS for problems with class variables involving two categories (automated CTA software solves only two-category problems: CTA for more than two class categories has never been reported) is:

ESS=100% x (Mean PAC – 50)/50 (1),

where

Mean PAC=100% x (sensitivity + specificity)/2 (2).

For example, if a CTA model had sensitivity=0.85 and specificity=0.74, then mean PAC= 100% x [(0.85+0.74)/2]=79.5%, and ESS=100% x [(79.5-50)/50]=59.0%.

Using ESS one may directly compare the performance of different models, relative to chance, regardless of structural features of the analyses, such as sample size, number of class categories, number of attributes and attribute metric, sample skew, and so forth. The rule-of-thumb which is used for evaluating *ecological significance* of results achieved by classification models is: ESS<25% (one-quarter of the improvement in classification accuracy theoretically possible to attain beyond the performance achieved by chance) is a *relatively weak* effect; 25%≤ESS<50% is a *moderate* effect; 50%≤ESS <75% a *relatively strong* effect; and ESS≥75% is a *strong* effect.[1] Thus, in order to complete the summary of the confusion table which was presented earlier, append the following conclusion: "The CTA model yielded ESS=50.0%, a relatively strong effect."

It is noteworthy that linear models may classify all observations in the sample into the dominant class if the sample is highly skewed (e.g., more than 75% of the sample falls into one class category). In this case Mean PAC is 50%, and ESS=0. For expository purposes, Table 3 illustrates how Mean PAC and ESS are related if one class category is classified perfectly, and Table 4 emphasizes that mean PAC=50 is what is anticipated by chance.

Table 3: PAC in Each of Two Groups (PAC= 100% in One Group), Mean PAC, and ESS

Group A	Group B	Mean PAC	ESS
100	0	50	0
100	10	55	10
100	20	60	20
100	30	65	30
100	40	70	40
100	50	75	50
100	60	80	60
100	70	85	70
100	80	90	80
100	90	95	90
100	100	100	100

Table 4: Patterns of PAC in Each of Two Groups that Yield ESS=0

Group A	Group B	Mean PAC	ESS
100	0	50	0
90	10	50	0
80	20	50	0
70	30	50	0
60	40	50	0
50	50	50	0

Ostrander et al.[6] note that, in contrast to sensitivity and specificity, PPV and NPV are influenced by base rate of class category *c* (e.g., 0 or 1) in the population, and by the false positive rate—the likelihood that the model will classify an observation into class category *c* when the observation is *not* a member of *c*. A method is given for easily assessing the models *efficiency* over different base rates (an efficient model provides PAC for category *c* which is greater than the category *c* base rate).[6]

Model Interpretation

In addition to its greater accuracy versus logistic regression analysis or Fisher's discriminant analysis, CTA also produced substantively richer findings. In the present example the linear models identified two patient clusters: relatively active, depressed nongeriatric people; and relatively inactive, non-depressed geriatric people.

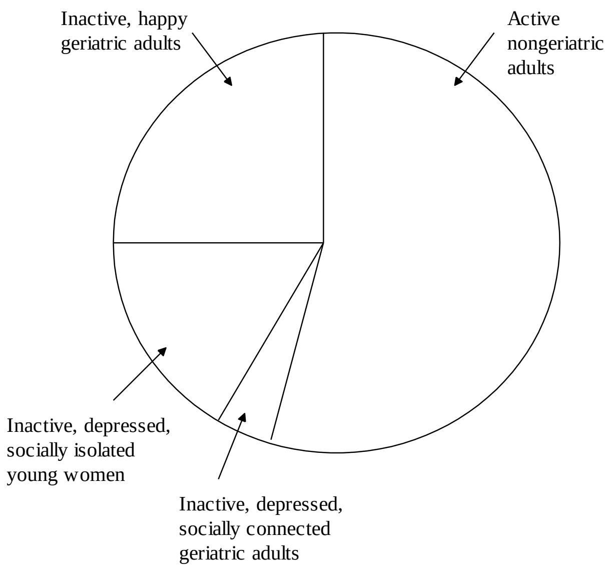

Figure 2: Pie-Chart Illustrating Distribution of Total Sample in Four CTA-Based Strata

In contrast, the CTA model identified four patient strata. Patients scoring >89.6 on Intermediate Activities were primarily (78.3%) relatively active nongeriatric adults (56% of total sample). Patients scoring at lower levels on Intermediate Activities, and at high levels (>65) on Mental Health, were largely (64.5%) relatively inactive, nondepressed geriatric adults (25% of sample). All the patients scoring at lower levels on both Intermediate Activities and Mental Health, and having fewer than six close friends, were inactive, depressed, socially isolated nongeriatric adults (14% of total sample, primarily young depressed women). Finally, all patients scoring at lower levels on both Intermediate Activities and Mental health, but having more than five close friends, were inactive, depressed, socially-connected geriatric adults (5% of sample).

Illustrating the portion of the total sample represented by CTA-identified strata, using a pie-chart, can facilitate understanding and development of policy implications of CTA-based findings: for example, by indicating the percentage of the sample that falls into each strata, the likelihood of attributing undue attention to comparitively rare strata is diminished (see Figure 2).

Table 5: AID Analysis for CTA Example

Attribute	Percent of Sample Evaluated in Part on the Basis of the Attribute	
Intermediate Activities	123/123	100.0%
Mental Health	54/123	43.9%
Number of Close Friends	23/123	18.7%

It is also informative to evaluate the attributes loading in the CTA model in terms of their importance in the prediction-making process. Conceptually related to the R^2 statistic from regression analysis, which indicates the percentage of the variance in the class (independent) variable which is explained by attributes (dependent measures) in the model[2], an *Attribute Importance in Discrimination* (AID) analysis indicates the percentage of the sample of classified observations which were influenced by the attribute (Table 5).

Only the root attribute is involved in the classification decisions for all observations in the sample. Easily seen in Figure 1, Mental Health was involved in classification decisions for all of the observations except for those classified on the right-hand side of the root attribute: 123–69=54 observations. Mental Health therefore influenced classification decisions for 100% x 54/123, or 43.9% of the total sample. Also easily seen in Figure 1, the Number of Close Friends influenced classification decisions for 100% x 23/123, or 18.7% of the total sample.

Validity Assessment in CTA

Limited by the daunting computational burden associated with manual construction of CTA models, *experimental* research addressing validity issues in CTA has been infeasible in the absence of automated software. Psychometric properties of scores created using optimal data analysis methods has been a major focus of the paradigm since its inception[1], and rigorous investigation in this area is underway.

Nevertheless, some preliminary research in this area has been reported. For example, a Bayesian method was developed for estimating the efficiency of a CTA model versus chance for any class variable base rate.[6] And, the first CTA model published in the field of medicine used a manual *hold-out* methodology to create a CTA model which was optimal for two random *split-halfs* of a single large sample.[7] This study used CTA to create a severity-of-illness score for predicting in-hospital mortality from *Pneumocystis carinii* pneumonia, which cross-generalized to independent random samples with strong ESS.[8]

For all models created in the optimal data analysis paradigm, the upper-bound of expected cross-generalizability of the findings to an independent random sample is estimated via jackknife (“leave-one-out”) analysis, whereby each observation in the sample is classified by a model created using a sample omitting the observation’s data.[1] In the absence of automated CTA software, only attributes with stable jackknife classification performance (i.e., with ESS that did not vary between training versus jackknife analyses) were used in manually-derived CTA models. However, an estimate of Type I error associated with the jackknife procedure may be determined by computing the ESSj from the confusion table generated by this procedure. The proportion of ESS values greater than ESSj obtained from randomly shuffled classes in the original Monte Carlo procedure estimates the jackknife Type I error, and setting this proportion to the desired value (e.g., 0.05) may be used in a decision rule to admit these attributes into the final model.

Obtaining CTA Models

The mechanics underlying construction of CTA models was described previously.[1,7,9] Recursively-derived CTA models chain together series of models, derived by univariate optimal discriminant analysis (UniODA), on monotonically diminishing sample strata.[1] Because they chain together UniODA models, CTA models may be derived manually[10] via ODA software[1] which conducts UniODA (advantages of using automated software are discussed ahead). Exact statistical distribution theory and Monte Carlo simulation methodology are available for testing one- (confirmatory, *a priori*) and two-tailed (exploratory, *post hoc*) hypotheses.[1]

Researchers are encouraged to construct at least one CTA model manually using ODA software, in order to gain a deeper understanding of the recursive mechanical nature of CTA. Furthermore, ODA and CTA software use identical command syntax, so skill and knowledge acquired by using ODA will generalize to operation of CTA.

Submitting a Program for Analysis

Automatic CTA software can be used to analyze problems with two class categories, 500 attributes, and 65,535 observations (methods to solve problems involving massive samples are undergoing alpha testing), and is available under either commercial or individual license: custom systems are also created for special-purpose applications. The software is available through the ODA webpage.[11] To run an analysis, registered users login to the ODA webpage and upload the associated command and data file. Analyses are executed in the order they were received, and all associated output is returned via eMail.

A quick word seems in order regarding why Optimal Data Analysis, LLC, adopted a "software as service" model for distributing access to the automated CTA software. From the perspective of users there are several advantages of this model: (1) users needn't tie-up their (probably slower) computers, our fast computers will do the work; (2) the most current version of the software is always immediately available; (3) one can work 24/7/365 from any computer, anywhere; and (4) if the system crashes then specialists will be scrambling to fix the problem immediately—and any problems may well be fixed before most users are even aware that an issue had occurred. Another advantage to both user and Optimal Data Analysis, LLC, is savings in money and time, because the software doesn't need to be adjusted to run in the context of many different types of constantly changing computers, operating systems and data-base programs. Users simply send text files to the CTA system, and the CTA system returns a text file output via eMail.

Interpreting Automated Software Output

The module which produces schematic illustrations of CTA models is currently under development, and investigation addressing optimal information display in this context is underway in our laboratory.[12] The present software reports CTA models using an intuitive shorthand notation describing the node constituents of the CTA model. To facilitate clarity, Figure 3 gives a schematic illustration of node structure underlying all CTA models.

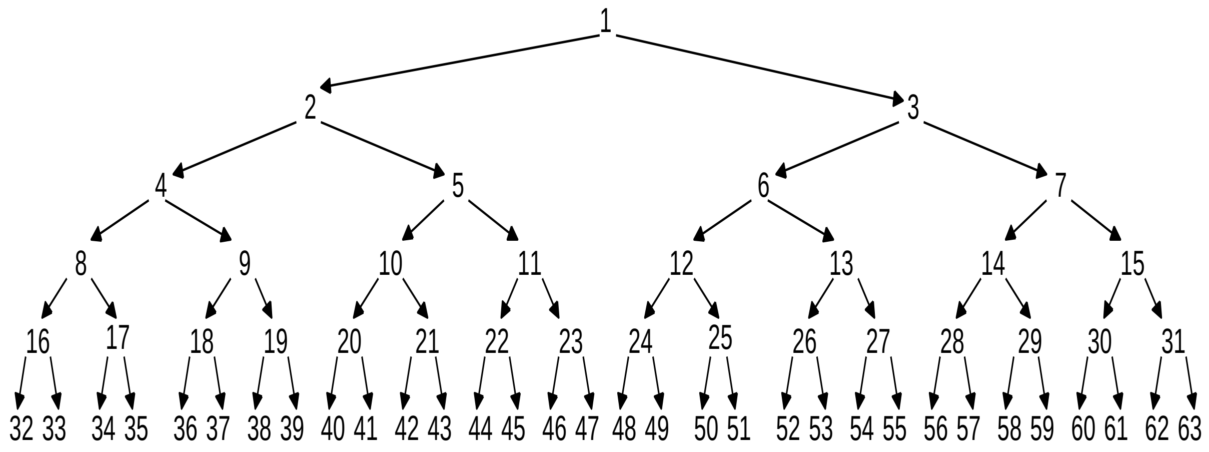

Figure 3: CTA Node Structure

It is a simple matter to determine the "identity number" of a node existing at a deeper depth than is illustrated in this five-level-deep tree (depth level 1 of the tree includes node 1; level 2 includes nodes 2 and 3; level 3 includes nodes 4-7; level 4 includes nodes 8-15; level 5 includes nodes 16-31; and level 6 includes nodes 32-63). From the perspective of node X (for X>1), the identify number of the node emanating from X's left-hand side is 2X, and from

Optimal Data Analysis
2010, Vol. 1, Release 1 (September 17, 2010), 144-160

X's right-hand side is 2X+1. For example, from node 47, node 94 (2x47) emanates to the left, and node 95 (94+1) emanates to the right. From node 94, node 188 emanates to the left, node 189 to the right, etcetera. Note that after the root attribute (depth 2 and deeper), all even-numbered nodes lie on the left-hand branch, and odd-numbered nodes on the right-hand branch, of the tree.

CTA software produces output employing node identity numbers to describe the CTA model: an example of CTA software output is presented in Figure 4 (hypothetical data). Respectively, the automated CTA software output lists: attribute name (D2, D3 and D4 loaded in the hypothetical CTA model); node identity number; tree depth level; sample size for the analysis indicated; ESS for the attribute; whether jackknife (leave-one-out, or LOO) validity analysis was stable (indicated) or unstable; jackknife ESS; *p* for the jackknife ESS; attribute metric (ORD=ordered, CAT=categorical); and CTA model shorthand.

The root attribute (here, D2) is listed first in the report. For each attribute the report *first indicates* the cutpoint and outcome for *the left-hand branch* emanating from the attribute, and *second for the right-hand branch*. *Branches ending in model endpoints are marked by an asterisk.* As seen, the left-hand branch emanating from D2 has a cutpoint of ≤6.2 units: observations having D2 scores ≤6.2 units are predicted to be a member of class 4, and this branch terminates in a model endpoint representing a total of 242 observations, of whom 165 (68.18%) are correctly classified. The remaining 242–165= 77 observations having D2 scores ≤6.2 units were members of class 5, and were misclassified by this branch of the CTA model.

```
ATTRIBUTE NODE LEV  OBS    p     ESS      LOO     ESSL   LOOp TYP           MODEL
--------- ---- ---  ---    -     ---      ---     ----   ---- ---  -------------------------
       D2    1   1  704  .000  48.44%  STABLE  48.44% .000 ORD  <=6.2-->4,165/242,68.18%*
                                                                  >6.2-->5,375/462,81.17%

       D3    3   2  292  .000  41.60%  STABLE  41.60% .000 ORD  <=4.5-->4,29/63,46.03%
                                                                  >4.5-->5,206/229,89.96%*

       D4    6   3   62  .039  28.99%  STABLE  28.99% .039 ORD  <=1.9-->4,18/30,60.00%*
                                                                  >1.9-->5,22/32,68.75%*
```

Figure 4: Sample CTA Software Output (Hypothetical Expository Data)

The right-hand branch emanating from D2 has a cutpoint of >6.2 units: observations having D2 scores >6.2 units are predicted to be members of class 5, but this branch does not terminate in a model endpoint. Rather, the model includes attribute D3 at node 3.

As seen, the left-hand branch emanating from D3 has a cutpoint of ≤4.5 units: observations having D3 scores ≤4.5 units are predicted to be members of class 4, but this branch does not terminate in a model endpoint.

The right-hand branch from D3 has a cutpoint of >4.5 units: observations with D3 scores >4.5 units are predicted to be members of class 5, and this branch terminates in a model endpoint representing a total of 229 observations, of whom 206 (89.96%) are correctly classified. The remaining 229–206=23 observations having D3 scores >4.5 units were members of class 4, and were misclassified by this branch of the CTA model.

Both branches emanating from D4 terminate in a model endpoint (this is always true for the last attribute listed in the output). The left-hand branch has a cutpoint of $\leq$1.9 units: observations with D4 scores $\leq$1.9 units are predicted to be members of class 4; this endpoint represents 30 observations of whom 18 (60.00%) are correctly classified and 30–18=12 (40.00%) are misclassified. And, the right-hand branch has a cutpoint of >1.9 units: observations having D4 scores >1.9 units are predicted to be members of class 5; this endpoint represents 32 observations of whom 22 (68.75%) are correctly classified and 32–22=10 (31.25%) are misclassified.

To construct an illustration of the final CTA model, referring to Figure 3 select nodes 1, 3 and 6 (see Table 3, column 2): these are depicted by circles (Figure 1). Branches are then depicted using arrows emanating from the left-hand side of the root attribute (D2), the right-hand side of D3, and both sides of D4, terminate in model endpoints depicted using rectangles (Figure 1). Add the Type I error rate beneath each attribute, cutpoint values adjacent to arrows, and text indicating the outcome for each endpoint—and the CTA model is complete.

Automated CTA Command Syntax

Table 6 gives an alphabetical roster and description of automated CTA software control commands and keywords (an example of an automated CTA program is provided ahead).

Table 6: Control Commands for Automated CTA Software

ATTRIBUTE

Syntax ATTRIBUTE *variable list* ;

Alias ATTR

Remarks The ATTRIBUTE command lists the attribute(s) to be used in the analysis. The TO keyword may be used to define multiple attributes in the list. For example, the command

ATTR A1 to A4;

indicates that A1, A2, A3 and A4 will be treated as attributes. Further exposition of the TO keyword is found in the discussion for VARS.

CATEGORICAL

Syntax CATEGORICAL {ON | OFF} ;
CATEGORICAL *variable list* ;

Alias CAT

Remarks The CATEGORICAL command specifies that categorical analysis will be used, and is required when the attribute to be analyzed is categorical. Using the ON keyword indicates that all variables in the variable list are categorical. CAT with no parameters is the same as CAT ON. The TO keyword may be used in the variable list (see the discussion under VARS).

CLASS

Syntax CLASS *variable list* ;

Remarks The mandatory CLASS command specifies the class variable to be used in the analysis. A separate analysis will be run for each class variable named. The TO keyword may be used in the variable list (see discussion under VARS).

DIRECTION

Syntax DIRECTION {< | LT | > | GT | OFF} *value list* ;

Aliases DIR, DIRECTIONAL

Remarks The DIRECTION command defines the presence and nature of a directional (i.e., *a priori*, one-tailed, or confirmatory) hypothesis. The parameter < or LT indicates that the class values in the value list are ordered in the "less than" direction. The parameter > or GT indicates the class values are ordered in the "greater than" direction. The value list must contain every value of the class variable currently defined. The default is OFF.

ENUMERATE

Syntax ENUMERATE {ROOT} {MINOBS *value*} ;

Remarks The ENUMERATE command with no options specifies that all combinations of attributes in the top three nodes will evaluated. ENUMERATE ROOT specifies that only the top node will have all attributes evaluated. ENUMERATE MINOBS *value* allows only solution trees with at least *value* observations in them.

EXCLUDE

Syntax EXCLUDE *variable* {= | <> | < | > | <= | >= | OFF} *value* (,*value2*,…) ;

Aliases EX, EXCL

Remarks This command excludes observations having the indicated *value* of *variable*. For example,

```
EXCLUDE D=4 ;
```

drops all observations with the value of 4 for attribute D. The command

```
EXCLUDE B=2 Z>=113 ;
```

drops all observations with the value of 2 for attribute B or values greater than or equal to 113 for attribute Z. Commas in the exclude string enable the user to exclude multiple values of a variable using a single command:

```
EXCLUDE C=2,4 ;
```

excludes all observations having a value of 2 or 4 for attribute C. Multiple EXCLUDE commands may be entered, up to a maximum of 100 clauses. Observations which satisfy any of the EXCLUDE clauses will be excluded.

FORCENODE

Syntax FORCENODE *node var* ;

Remarks The FORCENODE command forces CTA to insert the attribute *var* at node *node* in the solution tree. If the UniODA solution for this attribute is not significant, or this node is subsequently pruned, an error message will be printed.

GO

Syntax GO ;

Remarks The GO command begins execution of the currently defined analysis.

INCLUDE

Syntax INCLUDE *variable* {= | <> | < | > | <= | >= | OFF} *value* (,*value2*,…) ;

Aliases IN, INCL

Remarks The INCLUDE command functions in the same manner as the EXCLUDE command, except that only those observations with the indicated *value* for *variable* are included. If multiple INCLUDE statements exist, only those observations will be kept which satisfy all these INCLUDE statements.

LOO

Syntax LOO {*p*value | STABLE} ;

Remarks The LOO command indicates that leave-one-out analysis will be performed for every attribute in the tree. LOO STABLE allows only attributes with LOO ESS equal to the ESS for that attribute. LOO *pvalue* allows only those attributes in the solution tree which have an ESS that yields a $p \leq pvalue$.

MCARLO

Syntax MCARLO {ITERATIONS *value* | CUTOFF *pvalue* | STOP *confvalue* } ;

Alias MC

Remarks The MCARLO command controls Monte Carlo analysis for estimating Type I error, or *p*. The keywords specify stopping criteria; if any criterion is met, then the analysis stops. ITERATIONS (ITER) specifies the maximum number of Monte Carlo iterations. STOP xxx indicates the confidence level (in percent), which will stop processing for the current attribute, if the estimated Type I error rate (specified with the CUTOFF keyword) drops below this level. For example, the command

```
MCARLO ITER 70000
CUTOFF .05 STOP 99.9 ;
```

indicates a Monte Carlo analysis will be conducted, and will stop when one of the following occurs: (1) 70,000 iterations have been executed, (2) a confidence level of less than 99.9% that p<.05 has been obtained.

MAXLEVEL

Syntax MAXLEVEL *value* ;

Remarks The MAXLEVEL command specifies the deepest level or depth allowed in the solution tree.

MINDENOM

Syntax MINDENOM *value* ;

Remarks The MINDENOM command specifies that only attributes which yield a denominator of *value* or more will be allowed in the solution tree.

MISSING

Syntax MISSING {*variable list* | ALL} (*value*) ;

Alias MISS

Remarks The MISSING command tells ODA to treat observations with value (*value*) as missing for each variable on the list. For example, the command

```
MISSING X Y Z (-4) ;
```

indicates that observations with attrbutes X, Y, or Z equal to -4 will be dropped if they are present in a CLASS, ATTRIBUTE, WEIGHT, or GROUP variable. ALL specifies that the indicated missing value applies to all variables. The TO keyword may be used in the attribu-te list (see discussion under VARS).

Copyright 2010 by Optimal Data Analysis, LLC
2155-0182/10/$3.00

OPEN

Syntax OPEN {*path\file name* | DATA} ;

Remarks The OPEN command specifies the data file to be processed by ODA. This file must be in ASCII format. DATA indicates that a DATA statement, with inline data following, appears in the command stream.

OUTPUT

Syntax OUTPUT *path\file name* {APPEND} ;

Remarks The OUTPUT command specifies the output file containing the results of the ODA run. The default is ODA.OUT. APPEND indicates that the report is to be appended to the end of an already existing output file.

PRIORS

Syntax PRIORS {ON | OFF} ;

Remarks The PRIORS command indicates whether the ODA criterion will be weighted by the reciprocal of sample class membership. The default is ON. PRIORS with no parameters is the same as PRIORS ON.

PRUNE

Syntax PRUNE *pvalue* {NOPRIORS} ;

Remarks The PRUNE command indicates the p-value with which to optimally prune the classification tree. The NOPRIORS keyword should be used when PRIORS is turned OFF.

SKIPNODE

Syntax SKIPNODE *node* ;

Remarks The SKIPNODE command specifies that the node *node* will be empty of any attribute in the solution tree.

TITLE

Syntax TITLE *title* ;

Remarks The TITLE command specifies the title to be printed in the report. TITLE with no parameters erases the currently defined title.

USEFISHER

Syntax USEFISHER *value* ;

Remarks The USEFISHER command specifies that all probability calculations for categorical variable will be determined by Fisher's exact test, rather than by Monte Carlo.

VARS

Syntax VARS *variable list* ;

Remarks The VARS command specifies a list of attribute names corresponding to fields in the input data set. The TO keyword may be used to define multiple variables in the variable list. For example, the command

```
VARS X Y Z V1 TO V4 ;
```

specifies that the input file contains, in order, variables X, Y, Z, V1, V2, V3, and V4, and that there is at least one blank space separating all adjacent data. Alternatively, the data points may be separated by a single comma (with no spaces).

The TO keyword may be used to input a range of variables which have the same name except for the integer at the end of the name: the integers must be positive and ascending, increasing one unit per variable. Thus, VAR1 TO VAR10 is admissible (defining 10 variables). In contrast, VAR10 TO VAR1, VARA TO VARJ, or A TO X10, are not admissible.

The data for each observation may all exist on a single line of the data set, or may be placed on multiple adjacent lines. It is not recommended that a new observation is included on a line containing data from the previous observation.

WEIGHT

Syntax WEIGHT {*variable* | OFF} ;

Alias RETURN

Remarks The optional WEIGHT command specifies the weight variable for the analysis. The data values for the WEIGHT variable supply the weight the corresponding observation. The default is OFF.

Two Example Automated CTA Programs

Imagine an application in *finance*. In light of the recent calamitous failure of home mortgages, it is decided that a new credit-screening methodology is needed. Toward this objective a bank creates a dataset consisting of records describing all mortgages granted in the past three years (for exposition, imagine N=300 loans were made, of which, 10%, or 30 loans, were in default). The class variable is whether or not the loan went into default (label this class variable "Loan", and use dummy-codes 1=solvent, 0=default). The weight is the value of the loan in dollars (label this variable "Value"). Finally, imagine data are available for twenty attributes (Var1-Var20). Of these, Var1-Var10 are ordered, and the rest categorical.

Imagine that data and program files have been saved, and the output file will be saved, in the "c:cta" directory. As per the automated *CTA system job-naming convention*, *a common name is used for data, program and output files*: the name of the data file is "loan.dat"; the name of the program file is "loan.pgm"; and the name of the output file is "loan.out." The following code defines data and output files, assigns class, weight, and attribute variables, and defines the categorical attributes:

```
open c:\cta\loan.out;
output c:\cta\loan.out;
vars loan value var1 to var20;
class loan;
attr var1 to var20;
cat var11 to var20;
weight value;
```

It is decided *a priori* that, to increase the likelihood of the model cross-generalizing when applied to a validity sample, model endpoints should represent at least 5% of the total sample (5% of N=300 is N=15):

```
mindenom 15;
```

It is also decided *a priori* that to increase the likelihood of the model cross-generalizing, only variables stable in leave-one-out analysis would be allowed as model nodes:

```
loo stable;
```

It is decided *a priori* to use the system default (on) for weighting by prior odds intact, as another means of increasing the likelihood of the model cross-generalizing to an independent random sample, and also to explicitly maximize ESS (*setting priors off explicitly maximizes overall PAC*). The conventional experimentwise Type I error rate (p<0.05) is selected for pruning[13] to maximize ESS (experimentwise p<

0.05 is used automatically during model growth to control overfitting[1]):

```
prune .05;
```

Because there are relatively many categorical variables, it is decided to use Fisher's exact test to assess *p* for categorical variables[1] and reduce the number of Monte Carlo simulation experiments conducted:

```
usefisher;
```

Because the sample is modest in size, as is the number of attributes, and in light of the small number of failed loans in conjunction with the minimum denominator specification, it is decided that full enumeration of the first three nodes is feasible and appropriate, using 25,000 Monte Carlo experiments to compute *p* for all ordered attributes:

```
enumerate;
mcarlo iter 25000 cutoff .05 stop 99.9;
title loan default weighted CTA;
go;
```

Imagine an application in *space physics*. A phased array of 16 high-frequency antennas located in Goose Bay (Labrador), with a total transmitted power exceeding 6 kilowatts, was used to target free electrons in the ionosphere.[14] The class variable was labeled "return": "good" returns showed evidence of some type of structure in the ionosphere, and "bad" returns failed to provide evidence of structure (dummy-coded as "1" *vs*. "0", respectively). Received signals were processed using an autocorrelation function with two arguments per signal: time of pulse and pulse number. Because there were 17 pulse numbers for the Goose Bay system, there were thus 34 ordered attributes ("X1-X34"). There was no weight variable, and no categorical attribute. The objective is to maximize overall PAC—the total number of accurately classified good and bad returns.

Imagine that data and program files have been saved, and the output file will be saved, in the "c:cta" directory. As per the automated *CTA system job-naming convention*, *a common name is used for data, program and output files*: the name of the data file is "radar.dat"; the name of the program file is "radar.pgm"; and the name of the output file is "radar.out." The following code defines data and output files, and assigns class and attribute variables:

```
open c:\cta\radar.out;
output c:\cta\radar.out;
vars return x1 to x34;
class return;
attr x1 to x34;
```

It is decided *a priori* that, to maximize overall PAC achieved, the endpoint minimum denominator and model maximum depth would be unconstrained, but rather explicitly optimized by the program (no commands required).

Also, to maximize overall PAC it was decided to let attributes load as nodes even if unstable in LOO analysis, so long as their ESS in LOO analysis exceeded the ESS achieved by any other attribute:

```
loo 0.05;
```

It is decided *a priori* to set priors off in order to *explicitly* maximize overall PAC:

```
priors off;
```

The default setting for optimal pruning is priors on, so the prune command has to be adjusted to indicate that priors is set to off. Also, to maximize overall PAC, a statistical marginal loading will be allowed in the optimally-pruned model:

```
prune .10 nopriors;
```

Because there are no categorical attributes, the usefisher command is omitted. Because the sample is moderate in size, as is the number of attributes, and the attributes are ordered with few ties, analysis will be resource intensive. Also, 100,000 Monte Carlo experiments will be used in order to provide adequate statistical power for the small denominator endpoints that are anticipated:

```
mcarlo iter 100000 cutoff .05 stop 99.9;
```

Because UniODA analysis showed many attributes are loo-unstable, the analysis is judged to be too computationally intensive to attempt full enumeration on the first pass through the data via CTA (omitting the enumeration command results in an *algorithmic* analysis by default). Thus, after specifying enumeration of the root variable only, and providing a title, the program is ready to go:

```
enumerate root;
title RADAR maximum-PAC CTA;
go;
```

Advantages of Automated versus Manually-Derived CTA

Perhaps the most striking advantage of the automated software is that it is able to accomplish the example analyses just described, whereas *neither* of those analyses are *possible* to accomplish using manual derivation. Two specific advantages of the automated software are integrated automated pruning procedures: (a) sequentially-rejective Sidak "Bonferroni-type" multiple comparisons adjustment[1] to prevent model overfitting during the growth phase of the analysis; and (b) optimal pruning to maximize ESS at any specified experimentwise alpha level after growth has ceased.[13] And, those with experience conducting manual CTA using ODA[1] software would likely be amazed to hear that in recent speed trials (N=351, 34 continuous attributes) the automated software was able to solve *enumerated* CTA models averaging 0.7 CPU seconds per model, running 5,000 Monte Carlo experiments on a 3 GHz Intel Pentium D microcomputer. An *algorithmic* CTA derived manually for either type of CTA would typically require one or more man-days.

Initial comparisons of automated versus manual methods clearly reveal that the increased depth of search afforded by the enumeration capabilities of the automated software typically returns stronger, more efficient models.[8] The enumerated models may also be more consistent with original hypotheses than manually-derived counterparts.[15] Preliminary investigations in our laboratory suggest that the advantages of automated software become even more striking in applications which feature numerous, scattered, missing data. We are aware of several studies which compare previously-completed manually-derived CTA models *vs*. models derived using automated software, either planned or in progress. Monte Carlo simulation studies comparing the two methods are obviously warranted.

It is exciting to witness, whether as actor or spectator, the beginning of a new area of inquiry involving a powerful and evolving new methodology. Manually-derived CTA may be likened to an early telescope, focused by moving the body much like a trombone slide. Initial exploration using this early tool was fruitful and informative, and motivated the development of the automated system, which may be likened to a modern telescope. The modern instrument allows for pinpoint placement of the machine in any particular area (forcenode), aspect control including depth of field (maxlevel) and search (mcarlo iter; enumerate), luminosity (minobs; mindenom), fuzzy control (loo stable *vs*. .0x), and a standardized measure of acuity (ESS). It is likely that using these controls in a variety of applications will lead to refinements in the controls themselves, as well as in the methods of their operations, and these developments in turn may result in the creation of additional control features. For these reasons we anticipate surprising findings and major advances in the understanding of absolute and comparative capabilities of automated CTA—soon to come.

References

[1]Yarnold PR, Soltysik RC. *Optimal data analysis: a guidebook with software for Windows*. Washington, DC, APA Books, 2005.

[2]Grimm LG, Yarnold PR. *Reading and understanding multivariate statistics*. Washington, DC, APA Books, 1995.

[3]Grimm LG, Yarnold PR. *Reading and understanding more multivariate statistics*. Washington, DC, APA Books, 2000.

[4]Yarnold PR. Discriminating geriatric and non-geriatric patients using functional status information: an example of classification tree analysis via UniODA. *Educational and Psychological Measurement* 1996, 56:656-667.

[5]Yarnold PR, Soltysik RC. Optimal data analysis: a general statistical analysis paradigm. *Optimal Data Analysis* 2010, 1:10-22.

[6]Ostrander R, Weinfurt KP, Yarnold PR, August G (1998). Diagnosing attention deficit disorders using the BASC and the CBCL: test and construct validity analyses using optimal discriminant classification trees. *Journal of Consulting and Clinical Psychology* 1998, 66: 660-672.

[7]Yarnold PR, Soltysik RC, Bennett CL. Predicting in-hospital mortality of patients with AIDS-related *Pneumocystis carinii* pneumonia: an example of hierarchically optimal classification tree analysis. *Statistics in Medicine* 1997, 16: 1451-1463.

[8]Yarnold PR, Soltysik RC. Manual *vs*. automated CTA: optimal preadmission staging for inpatient mortality from *Pneumocystit cariini* pneumonia. *Optimal Data Analysis* 2010, 1:50-54.

[9]Yarnold PR, Soltysik RC. *Hierarchically optimal classification tree analysis: applications in medicine and allied health disciplines*. Submitted.

[10]Yarnold PR. *How to obtain a CTA model manually using ODA software*. In preparation.

[11]Automated CTA software webpage: www.OptimalDataAnalysis.com.

[12]Yarnold PR, Soltysik RC. *Automated CTA: initial standards for exploratory analysis*. In preparation.

[13]Yarnold PR, Soltysik RC. Maximizing the accuracy of classification trees by optimal pruning. *Optimal Data Analysis* 2010, 1:23-29.

[14]Sigillito VG, Wing SP, Hutton LV, Baker KB. Classification of RADAR returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest* 1989, 10:262-266.

[15]Coakley RM, Holmbeck GN, Bryant FB, Yarnold PR. Manual *vs*. automated CTA: predicting adolescent psychosocial adaptation. *Optimal Data Analysis* 2010, 1:55-58.

Author Notes

Correspondence: Optimal Data Analysis, 1220 Rosecrans St., Suite 330, San Diego, CA 92106. eMail: Journal@OptimalDataAnalysis.com.