About the IND Tree Package*

Wray Buntine, RIACS
NASA Ames Research Center
Mail Stop 269-2
Moffet Field, CA 94035

September 29, 1991

Abstract

This note introduces the IND Tree Package to prospective procurers and those users/installers looking at IND for the first time. IND does supervised learning using classification trees. IND integrates features from Breiman *et al.*'s CART and Quinlan's C4 with newer Bayesian and minimum encoding methods for growing classification trees, and provides an experimental control suite on top. The package comes with a manual, "man" entries, and a guide to tree methods and research. Information about obtaining IND, performance statistics, documentation, authorship, copyright, installation, etc., are given. IND is research software still under development, although it has been used considerably since late 1989. IND is implemented in C under UNIX.

What is a classification tree?

Supervised learning (rule induction, discrimination, learning of classification rules from data, etc.) is a task that is ubiquitous in artificial intelligence, applied statistics and pattern recognition. In machine learning it is used as an aid to knowledge acquisition.

We typically have an *expert* who is sufficiently knowledgeable to formulate the problem for us and is in possession of a *training set*, a set of examples each belonging to one of a small discrete set of mutually exclusive and exhaustive *classes*. Classes might be "positive" and "negative", or "diseased", "healthy" and "recovering", or similar. The task is to develop a *classification rule* to predict the *class* of further, unclassified examples. Examples are described in terms of attributes which may be binary ("true" or "false"), multi-valued or real-valued. The training set is usually historical data about the problem at hand.

The IND package represents classification rules as *decision trees* or *class probability trees*. These are a general form of classification rule that mix discrete and continuous data and are often suited when there is believed to be some form of non-linear structure in the data. A

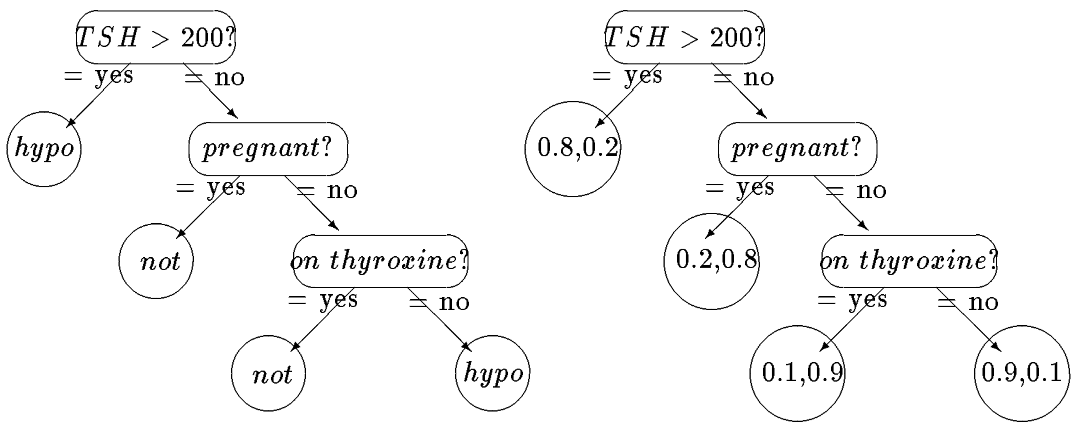

Figure 1: A decision tree and a class probability tree from the thyroid application

decision tree is shown in the left side of Figure 1. This has the classes *hypo* (hypothyroid) and *not* (not hypothyroid) at the leaves. This tree is referred to as a "decision tree" because decisions about class membership are represented at the leaf nodes. Notice that the real valued attributes TSH and FTI have been incorporated into the tree by making a binary test of the form $attribute < cut\text{-}point$. Also, the tree need not be binary; if a 4-valued attribute is tested at one of the nodes, then the tree might have 4 branches coming from the node, one for each value.

In typical problems involving noise, class probabilities are usually given at the leaf nodes instead of class decisions, forming a *class probability tree* (where each leaf node has a vector of class probabilities). A corresponding class probability tree is given in the right of Figure 1. The leaf nodes give predicted probabilities for the two classes. Notice that this tree is a representation for a conditional probability distribution of class given information higher in the tree.

Methods for learning decision trees and class probability trees have been under development in some form or another for some two decades. The standard technique for building classification trees from data is the so-called recursive partitioning algorithm that forms the basis of systems such as Quinlan's ID3 and C4, well know in the machine learning literature, and Breiman, Friedman, Olshen and Stone's CART, well known in the applied statistics literature.

What is IND?

IND is a suite of C programs and C shell scripts for building decision tree classifiers (i.e. supervised learning). Currently, several different decision tree methods are integrated (CART style[1], early C4 style, MML/MDL, and Bayesian averaging), as well as a simple Bayes classifier. Experiments have been performed to check that IND reimplements CART and

[1]The regression aspect of CART is not implemented.

the early C4 reasonably well. The new Bayesian/MML/MDL features can give performance improvement over these in many cases when used appropriately. IND is currently under development.

If you're interested in applying IND to applications, we give advice on which options to use and how to take into account features of your application and data when configuring your use of IND. If you're interested in running comparative trials or just experimenting with tree software, IND provides extensive experimental control (random partitioning, cross validation) and significance testing. The code for IND is provided (and sometimes even moderately documented) so you can develop your own extensions.

IND has a variety of features including interactive control of tree building, variable search such as multi-ply lookahead, missing values and subsetting, prediction of error rates, a range of priors for the Bayesian methods, printing options, etc., and comes with "man" entries, a user manual, and a start of the art guide to tree learning research.

Performance Statistics

Results are given in Table 1 below for IND running on databases available from University of California at Irvine[2]. The results show that the new features of IND can give more accurate class probability estimates for new examples (MSE column), and better predictions (error), but at the cost of increased computation (timing), depending on the problem.

The method "CART-like" corresponds to the CART-like version: 10-fold cross-validation with GINI index of diversity and the 0-SE splitting rule. The method "CART-test" is similar but uses a test (hold-out) set instead of cross-validation during cost-complexity pruning. The method "C4-early" corresponds to a version like early C4: the use of pessimistic pruning and the information gain splitting rule (the current C4.5 uses more sophisticated pruning), and "option trees" corresponds to the full Bayesian method of smoothing and averaging over multiple trees, choice of prior parameters such as "expected accuracy at leaves", and 2-ply lookahead. The method "Bayes trees" is with the Bayes splitting rule and Bayesian smoothing. Timing results are reported for a single trial on a SPARCstation 1 with 12Mb of memory. Error in percentage is reported for the average of 10 trials. MSE is the mean-square error or half-Brier score which measures the accuracy of the class probability estimates, with smaller being better.

The high standard deviations indicate one cannot safely compare these results with similar results appearing in the literature. However, in all but LED with 200, "option trees" is more accurate in prediction than the first two methods with confidence at the 5% level by the 1-tailed paired t-test. Notice "option trees" is up to 2 orders of magnitude slower. This tends to decrease as the set size increases because fewer trees become a good fit to the data.

[2]FTP to `ics.uci.edu` and look in `machine-learning-databases`

[3]The "voting-" data has had the attribute "physician-fee-freeze" deleted.

data	method	set size (train + test)	timing (train + test)	error ± std. dev.	MSE
LED	CART-like	100 + 2900	1.3sec + 1.9sec	35.7 ± 4.0	0.57
LED	C4-early	100 + 2900	0.8sec + 2.5sec	36.1 ± 4.6	0.57
LED	option trees	100 + 2900	17.8sec + 101.3sec	33.8 ± 2.5	0.51
LED	CART-like	200 + 2800	3.5sec + 2.6sec	32.1 ± 0.8	0.53
LED	C4-early	200 + 2800	2.0sec + 2.6sec	31.7 ± 0.9	0.53
LED	option trees	200 + 2800	16.6sec + 59.4sec	31.8 ± 1.3	0.48
glass	CART-like	100 + 114	4.8sec + 0.3sec	37.3 ± 4.7	0.60
glass	C4-early	100 + 114	1.5sec + 0.3sec	36.2 ± 4.3	0.59
glass	Bayes tree	100 + 114	3.0sec + 0.3sec	35.5 ± 4.6	0.52
glass	option trees	100 + 114	469.9sec + 102.1sec	31.9 ± 6.8	0.47
voting-[3]	CART-like	200 + 235	1.4sec + 0.1sec	12.3 ± 1.7	0.22
voting-	C4-early	200 + 235	0.9sec + 0.1sec	12.9 ± 1.5	0.22
voting-	option trees	200 + 235	187.2sec + 17.9sec	10.6 ± 1.5	0.16
letter-recg.	CART-test	16000 + 400	267.9sec + 14.1sec	17.7 ± 0.7	0.28
letter-recg.	C4-early	16000 + 400	327.5sec + 14.4sec	13.9 ± 0.6	0.22
letter-recg.	Bayes tree	16000 + 400	829.7sec + 29.9sec	12.8 ± 0.4	0.20

Table 1: Performance statistics

Obtaining IND

NASA has specific requirements on distribution of software that is still under development. Routine distribution is only available to industrial and academic organizations in the United States. Requests for sharing of software with potential foreign collaborators will be treated on a case-by-case basis. Please bear in mind that future developments and extensions by us are easier to justify to our management if you keep us informed of successful uses (industrial and academic) you make of the package.

A sample letter is attached at the back of this note. Potential foreign collaborators should discuss access first. To obtain the package, you should sign then mail or FAX this to:

FAX: (415) 604 6997 (before Nov. '91)
FAX: (415) 604 3594 (after)
post: IND Tree Package
C/O Wray Buntine
RIACS and Code FIA
Mail Stop 269-2
NASA Ames Research Center
Moffett Field, CA, 94035

The package will be sent to you via e-mail so you should send a brief e-mail request separately to:

e-mail: `ind@kronos.arc.nasa.gov`

Contents of the IND Directory

Scripts: various shell scripts for controlling the whole thing

Statlib: library of statistical functions

Treelib: library of tree processing functions

Eglib: library of example and contingency-table processing functions

Lib: library of general system utilities

Trees: tree generation, pruning and evaluation programs

Util: data encoder, statistical summary program, and random sampler

Doc/course: details of 4-week, 3rd-year undergrad course on trees

Doc/manual: LaTeX files for the IND user manual

Man/manl: man entries

Doc/Release.tex: this file

Include: include files for the many data structures (trees, sets, ...)

Data: sample data files

Doc/manual/copyright.troff: yes, the copyright

Compatability

The system has been developed exclusively in a SUN workstation environment under various releases of SunOS UNIX. It compiles under cc and gcc. Feedback on changes required to get the package running on other environments are welcome. For the most part, use of system calls has been minimal so porting to other UNIX environments should be quite easy.

Installation

Best thing is to flip through the manual to learn about what IND can do. These instructions are also reproduced there. The IND directory takes about 2.5Mb, and the object files take about 1.5Mb when stripped (which is done automatically by the "Makefile"s). Here's what you do to get the system installed.

1. Check for machine compatibility by looking at:

   - Lib/quickfit.c (top few #defines)
   - Include/Lib.h (last few lines)
   - Include/SET.h (at the top)

   for potential storage type and alignment problems. The system has only ever been compiled on SUNs so expect some problems on other UNIX machines. Some things like ftime() and "struct timeb" (in random.c), and a few others are used, which tend to be UNIX version dependent.

2. Modify the primary "Makefile" to add your "BIN" to the file. Then run "make Bin". This will modify all other IND Makefiles in the IND subdirectories so that they know where to place the bins.

3. Similarly, modify CC and CFLAGS in the primary "Makefile" and run "make cc" and/or "make cflags" respectively if you wish to change the factory-set options.

4. IND will also compile a slightly modified version of **xgraph** that requires certain X11 include files be on your system. Set variable Xinclude in "IND/Util/xgraph/Makefile". If you won't be using this, then modify "IND/Makefile" so that **xgraph** wont be made, and never use the "x" option in interactive mode. You probably wont notice the loss.

5. Compile IND using "make install". This will call make recursively in the various subdirectories to construct the ".o" files, the ".a" libs and then go on to make the programs and put them in your "BIN".

6. Add the "Scripts" directory to your own path with "set path = ($path .../IND/Scripts)", or alternatively copy the contents of the "Scripts" directory to the "BIN" directory. Run "rehash" if you changed your path.

7. Add the "Man" directory to your own MANPATH.

8. Try running some examples with "make test >& test.out". Compare or "diff" the output with "make.test.out". If you are not on a SUN then there may be some differences because the random number generator may operate differently due to integer arithmetic. There may also be slight differences in floating point.

Basic Help

The IND Manual: "An Introduction to IND and Recursive Partitioning" is the best place to start if you are unfamiliar with IND or recursive partitioning. (If you don't have a copy of the IND Manual, a LaTeX version of it is in the directory "IND/Doc/manual".) The manual contains an introduction to IND that walks through a few typical sessions, a tutorial on recursive partitioning, a description of IND options, and a fairly complete glossary and bibliography.

The man pages contained in "IND/Man/man1" are reasonably complete. (Many of the man pages should also included in the IND manual.) Using IND will require frequent reference to some of the man pages so be sure to add "IND/Man" to your MANPATH if you don't have a printed copy of the IND manual. A good place to start is to look at the man entry for "attributes" (to see how to set up data files) and the man entry for "mktree" (shows you how to run the CART-like, C4-like and Bayesian versions of the algorithms). Use "man -t -M IND/Man" if you want to print the man pages nicely using troff.

To get things going you need to format data files correctly. Look at man page "attributes(1) and at the ".attr" and ".dta" files in the "Data" directory for examples. Then flip though the major sections in the manual such as "A tour of IND" to see what options to run.

"IND/Doc/course" contains details of a 3 week graduate course on learning trees with reading material and questions, etc.

The "Scripts" directory contains all sorts of useful goodies that are nicely documented in their beginning; these control all the ghastly programs underneath. If you're doing comparative studies, then look at the "ttest", "truns" and "lstat" man entries. "truns" should be locally configured by you. At the moment its a tower-of-babel type system. IND really needs an X11 interface, but ...

If you have trouble printing any of the LaTeX files, then ask for Postscript versions to be sent to you. These weren't included in the distribution because they double the size of the release.

Authorship

IND was built on top of an early suite of software developed at Basser Department of Computer Science at Sydney University by a lineage of students of Jason Catlett: David Harper, Murray Dean, David Muller and Chris Carter, and possibly some others. Some early "man" entries and bits and pieces of the code where also done by Chris Carter while at the University of Technology, Sydney. The only programs or files that remain largely intact are IND/Util/sample. All others have been recoded and restructured to a large degree, except for the "symbol" structure in SYM.h and its associated routines. IND also greatly benefited from our interaction with Ross Quinlan.

IND was developed by Wray Buntine, while at S.O.C.S., University of Technology Sydney ('88-'89), Turing Institute ('89) and Strathclyde University in Glasgow ('89), and code FIA, NASA Ames Research Center and RIACS ('90). More recently ('91), Rich Caruana helped make the package more presentable while on a summer internship at NASA-Ames Research Center.

Copyright

Technical Questions and Feedback on Use on IND

We ask that you report to `ind@kronos.arc.nasa.gov` any use you make of IND, describing the application and your analysis of the results. Please feel free to make suggestions about desirable improvements and extensions, and perceived problem areas. We regard such feedback as an essential element of the development process. Please e-mail bug reports to `ind@kronos.arc.nasa.gov`. We give no guarantees about fixes, support, or question answering, but we will try to be helpful given our limited resources.

Planned Extensions

Here is a list of extensions that we would like to include in feature releases, support pending. Most are supported by research we have reported in journals and proceedings.

- Other learning algorithms such as rule learning, regression and/or back-propagation, learning Bayesian and/or Markov networks, and probabilistic variants of case-based or instance-based reasoning.
- X.11 interface to the package to reproduce the tasks of **ttest**, etc., with a nice point and click interface, and to allow interactive learning.
- Extensions to the methods such as missing values, incremental or large batch learning, etc.
- Clean up the search control and interface to the Bayesian option trees (-J option in **tgen**), to make this method more accessible. Smarter searching for option trees together with more compact summary tree to allow anytime search and to produce results more readily presentable.

Warnings

Software in the "IND/Bayes" directory (the simple Bayes classifier) has only been marginally tested. The full range of option combinations have not been tested and are not supported. In addition, options not listed in the man entries are not supported. For those looking at the code, bear in mind it is research code and various features exist at different stages of development. The option trees and -J option to **tgen** doesn't have anytime search control so can be difficult to use.

Date

Peter Friedland
Artificial Intelligence Research Branch
Mail Stop 269-2
Moffett Field, CA, 94035
USA

Dear Dr. Friedland

I would like to obtain the IND software package.

I understand the IND software package is still under development and is therefore not available through NASA's computer code dissemination center, COSMIC, so instead is available by NASA Management Instruction 22.10.2B. To receive the package I therefore agree to the following conditions:

1. Use the package only within the United States and not disclose it to third parties.
2. Assist NASA in the development of the package by reporting problems encountered with it and any modifications made; and
3. Place no limitations on NASA's use of reported modifications.

Also, when using the package, I will credit NASA Ames' and RIACS' contribution.

I anticipate using the code for the following (although other uses may be made):

.______________________________ .

.______________________________ .

Sincerely,

Name:
Title:
Company:
E-mail:
Address: