How Can We Deliver Advanced Statistical Tools to Physicists

Ilya Narsky, Caltech
Outline

- StatPatternRecognition: A C++ Package for Multivariate Classification
- What would be an ideal statistical framework for HEP?
- What software repository I would like.
Why write a new package when there is so much free software floating around?

- I wanted a C++ package
- I have not found a C++ implementation of all the methods I wanted to have in one package
  - decision trees, bump hunting, boosting and bagging
- I wanted to implement my own versions of the decision tree and the bump hunter that directly optimize criteria suited for physics analysis (such as $S/\sqrt{S+B}$)

The package has been under intensive development through the winter and spring. It is stable and in good shape now.
Methods

- Original implementations:
  - linear and quadratic discriminant analysis (aka Fisher)
  - simple binary splits
  - decision trees
  - bump hunting algorithm (PRIM, Friedman & Fisher)
  - boosting and random forest
  - combiner of classifiers
  - bootstrap
  - estimation of data moments

- Interfaces to the Stuttgart Neural Network Simulator:
  - feedforward backpropagation neural net and radial basis function net
  - Not for training!!! Simply for reading saved network configurations, computing the network output and storing it into ntuples.
Framework

- Tools for choosing input variables, imposing cuts, and storing input data with classifier output into ntuples.
- The package accepts weighted data.
- Input/Output (at present):
  - Input = ascii, in a format similar to that used by SNNS
  - Output
    - trained classifier configuration => ascii
    - input data with classifier output included => Hbook or Root ntuples using a BaBar-specific C++ wrapper
- OO design: abstract interfaces. Functionality can be extended by supplying new implementations.
A Special Flavor

- An arbitrary user-supplied optimization criterion
  - to be used with binary splits, decision trees and bump hunting algorithm
  - 7 criteria are currently implemented
    - 3 criteria used in commercial decision trees: correctly classified fraction of events, Gini index and cross-entropy
    - 4 criteria suited for physics analysis: signal purity, signal significance, 90% upper limit (Bayes with uniform prior), and tagging efficiency
    - Very easy to implement a new one, to your taste.

- The payoff is clear. For example, a decision tree optimizing the signal significance does give a better significance than the one optimizing the Gini index.
Documentation

- physics/0507157, Optimization of Signal Significance by Bagging Decision Trees
- README file included in the package
Application of Boosted and Bagged Decision Trees to Physics Analysis

- $B \rightarrow \gamma l\nu$ analysis at BaBar (see the 2 notes posted at physics archive)
- $B^0 / \bar{B}^0$ tagging at BaBar (in progress)
- $B \rightarrow K^* \nu\nu$ analysis at BaBar (in progress)

TABLE I: Signal significance, $S_{\text{train}}$, $S_{\text{valid}}$, and $S_{\text{test}}$, for the $B \rightarrow \gamma l\nu$ training, validation, and test samples obtained with various classification methods. The signal significance computed for the test sample should be used to judge the predictive power of the included classifiers. A branching fraction of $3 \times 10^{-6}$ was assumed for both $B \rightarrow \gamma \mu\nu$ and $B \rightarrow \gamma e\nu$ decays. $W_1$ and $W_0$ represent the signal and background, respectively, expected in the signal region after the classification criteria have been applied; these two numbers have been estimated using the test samples. All numbers have been normalized to the integrated luminosity of 210 fb$^{-1}$. The best value of the expected signal significance is shown in boldface.

<table>
<thead>
<tr>
<th>Method</th>
<th>$B \rightarrow \gamma e\nu$</th>
<th>$B \rightarrow \gamma \mu\nu$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S_{\text{train}}$</td>
<td>$S_{\text{valid}}$</td>
</tr>
<tr>
<td>Original method</td>
<td>2.66</td>
<td>-</td>
</tr>
<tr>
<td>Decision tree</td>
<td>3.28</td>
<td>2.72</td>
</tr>
<tr>
<td>Bump hunter with one bump</td>
<td>2.72</td>
<td>2.54</td>
</tr>
<tr>
<td>AdaBoost with binary splits</td>
<td>2.53</td>
<td>2.65</td>
</tr>
<tr>
<td>AdaBoost with decision trees</td>
<td>13.63</td>
<td>2.99</td>
</tr>
<tr>
<td>Combiner of background subclassifiers</td>
<td>3.03</td>
<td>2.88</td>
</tr>
<tr>
<td>Bagging decision trees</td>
<td>9.20</td>
<td>3.25</td>
</tr>
</tbody>
</table>
Availability

- In the public repository at BaBar. Runs out of the box.
- If you are not a BaBarian and want to install the package, you need to:
  - send me an email: narsky@hep.caltech.edu
  - receive a copy of the code
  - remove `#include “BaBar/BaBar.hh”` from all *.cc files – easy
  - resolve references to CLHEP headers and CERNLIB in your Makefile – easy
  - replace SprTupleWriter with your implementation – moderate
    - this class stores input data and classifier output into ntuples
    - depends on HepTuple, a BaBar-specific interface
    - only two ntuple methods are used: booking an ntuple and filling ntuple columns
- I am happy to provide help, both in installation and advice on how to use the package. Send me questions.
Future Work

- Development has been mostly finished. The package is stable and in good shape.
- I will occasionally add new non-critical features and fix bugs if I find any. If you want to be in the loop, I can put you on the mailing list.
- Improvements:
  - I could strip the package of CLHEP, CERNLIB and HepTuple dependencies to make it self-installing. Not sure if this is necessary. Need user feedback.
  - It would be a good idea to interface data input to ROOT. I won’t have time to work on this soon. If anyone is willing to work on this, I will be happy to collaborate.
  - If anyone wants to add new functionality to the package, I will be happy to collaborate.
A Statistical Framework for HEP

- **C++**
  - Easy to integrate in HEP environment. No wrappers needed!
  - Ultimate flexibility. You can include classes in your code, not just run executables.
  - Open source. Plenty of physicists can program in C++.

- **Software management scheme:**
  - At BaBar I support ~30 packages with ~100k lines of C++, IDL and Fortran code.
  - This experience has taught me one thing: It is much easier to handle many small packages with weak dependencies among them.

- **A set of loosely related C++ packages for specific tasks:**
  - multivariate classification
  - multivariate density estimation and goodness of fit
  - multivariate feature selection
  - ...

Ilya Narsky

Fermilab Workshop on Statistical Software Repositories, 2005
Why I don’t like R. Part 1.

- R is not C++
  - BaBar setup:
    - Production code: heavily dominated by C++, distributed in official releases, more or less clean and well maintained.
    - High-level user physics analysis code: mixture of C++, Fortran, and Root/Paw macros. Maintained privately by individual users. From the software management point of view, it’s a huge mess.
  - There is no way R can be part of the production code! (Examples: tagging, PID etc.)

- Performance (CPU and memory consumption)
  - Example: 100 boosted or bagged decision trees with ~1000 leaf nodes each using 500k training points in 14 dimensions.
  - Training with SPR takes several hours in a batch queue at SLAC.
  - How long would it take with R? Would it even be possible with R?

- I promise to benchmark my package against R. (Not sure when though.) From general experience and what R users told me, I would expect the C++ code to be more efficient.
Still not liking R… Part 2.

- Psychological barrier: how many people know C++ and how many know R?
  - We can’t simply use what statisticians did. We need to extend and develop code for physics applications. Example: decision tree that optimizes signal significance.

- I am in general prejudiced against monstrous packages. And even more – against integrating two monstrous packages like Root and R together.

- Not so many R modules are obviously useful for physics analysis. One could re-implement them in C++ in not too much time.
**Why I Might Like R**

- Has a number of advantages:
  - versatile – includes many methods
  - earned a good reputation among statisticians
  - extensive documentation and book manuals

- If someone wants a slow offline tool and willing to learn the new language, this should be a good thing.
A Software Repository – What I Would Like

- I am in favor of a C++ framework but…
- …restricting coding efforts to one particular language or one particular framework at this time might do more harm than good. (Feel free to convince me otherwise.)
- I’d like to see a public repository kept in one place (perhaps at Fermilab):
  - Open to all HEP researchers for uploading and downloading source code, libraries and executables. Each developer is responsible for maintaining an up-to-date copy, proper documentation etc.
  - Needs to be well advertised, so people know where to look for statistical software.
  - If the repository gets large, a searchable index would be nice.
- Plus a collection of links to external resources (Jim Linnemann’s page is quite impressive already).
Summary

- StatPatternRecognition is available to public. I will be happy to collaborate with anyone who wants to use and improve the package.
- R can be used as an offline analysis tool. But betting on R as the main statistical analysis tool for HEP would be shortsighted. I think we should go after developing or adopting C++ code for our specific HEP needs.
- I move for a public HEP statistical software repository in one place and open to free uploads.
- Whatever you do, advertise through publications and talks. All your efforts will mean nothing until you convince an average physicist through practical applications to physics analysis.