StatPatternRecognition: Status and Plans

Ilya Narsky, Caltech

Outline

- Package distribution and management
- Implemented classifiers and other tools
- User interface
- Near-future plans and solicitation

This is a technical talk on capabilities of the package. Not intended to explain how classifiers work or what classifiers you should be using in what situations. Please refer to

http://www.hep.caltech.edu/~narsky/spr.html
http://www-group.slac.stanford.edu/sluo/Lectures/Stat2006_Lectures.html

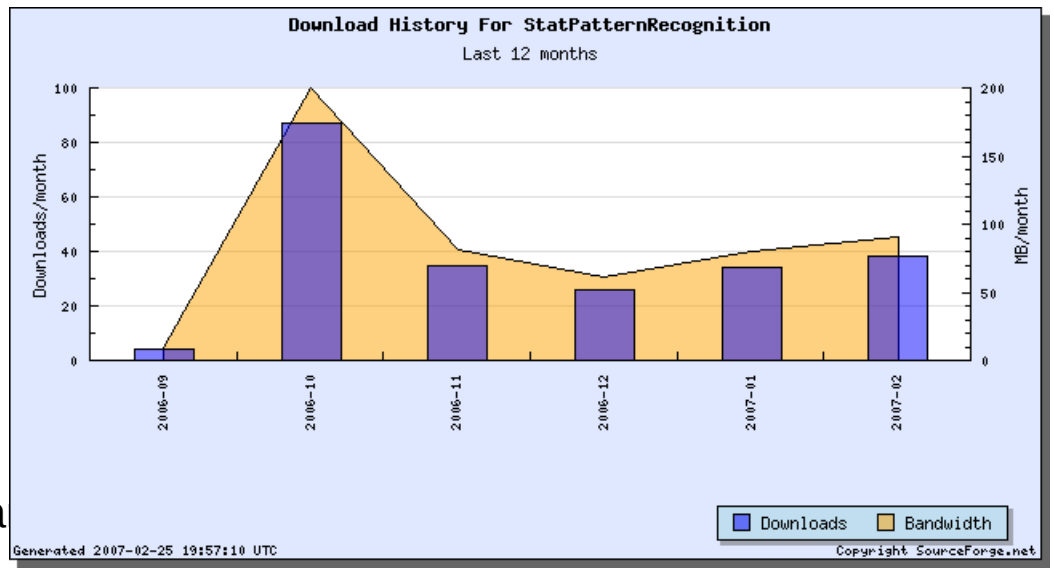
for more info on theory.

Distribution

- Between summer 2005 and summer 2006, SPR was distributed outside Babar as a standalone package with a mandatory dependency on CLHEP and an optional dependency on Root
- In summer 2006 SPR was stripped of CLHEP dependency and equipped with autotools (thanks to Andy Buckley and Xuan Luo)
- In September 2006 SPR was posted at Sourceforge: https://sourceforge.net/projects/statpatrec
- Since then, Sourceforge is the main source of SPR code. SPR versions elsewhere (including the one in Babar CVS) are not up-to-date.
- It is planned to write scripts for conversion of the Sourceforge version to Babar and CMS environments.

History at Sourceforge

- Since September 2006 several new methods and interface utilities have been implemented. 9 tags released. See further slides.
- Downloaded ~220 times.
- 26 subscribers on the Sourceforge mailing list
- My coworker submitted SPR into Fedora package repository. Will be part of OS soon… ☺



User builds

- SPR can be built in two versions:
  - standalone with Ascii I/O; no external dependencies
  - Root I/O
- Desired build options are chosen by the ./configure script. See INSTALL. Run `./configure --help` for a full list of options.
- The Ascii version of the package builds smoothly. The Root version build can bail out with errors related to Root classes. The fix is to change compilation flags. Again, see INSTALL for detail.
- Successful builds on 32-bit SL3, SL4, RH4 and 64-bit Fedora. Enthusiasts adapted SPR to WinXP and MacOS.

Implemented classifiers

- Decision split, or stump
- Decision trees (2 flavors)
- Bump hunter (PRIM, Friedman & Fisher)
- LDA (aka Fisher) and QDA
- Logistic regression
- Boosting: discrete AdaBoost, real AdaBoost, and epsilon-Boost
- Arc-x4 (a variant of boosting from Breiman)
- Bagging
- Random forest
- Backprop neural net with a logistic activation function (original implementation)
- Multi-class learner (Allwein, Schapire and Singer)
- Interfaces to SNNS neural nets (without training): Backprop neural net, and Radial Basis Functions

A tiny bit of theory

- Logistic regression
  - Like Fisher, computes an optimal linear boundary between two classes. Unlike Fisher, does not assume that the two classes have multivariate Gaussian distributions. Computes the boundary under more general assumptions.
- Boosting:
  - Discrete AdaBoost – computes misclassification rate for each weak classifier and increases weights of misclassified events by the amount derived from this misclassification rate
  - epsilon-Boost – increases weights of misclassified events by a fixed amount specified by the user
  - Real AdaBoost – multiplies weights of signal events by **(1-r)/r** and weights of background events by **r/(1-r),** where **r** is the weak classifier response for this event

Other tools (reminder)

- Cross-validation
- Bootstrap
- Tools for variable selection
- Decision trees and bump hunter can optimize any of 10 figures of merit implemented in the package
- Computation of data moments (mean, variance, covariance, kurtosis etc)
- Arbitrary grouping of input classes in two categories (signal and background)
- Multivariate GoF method proposed by Friedman at Phystat 2003

Boost and bag anything you like

- SPR was always capable of boosting and bagging any classifier in the package – if you were willing to do a bit of C++ coding
- Now you can boost and bag an arbitrary sequence of classifiers using SprBoosterApp and SprBaggerApp executables. All you need is specify classifier name and params in the input config file. See booster.config for examples.
- Example: Boost decision tree (odd cycles) and neural net (even cycles)

➢ **cat booster.config**

```
TopdownTree 5 0 8000
StdBackprop 30:15:7:1 100 0.1 0 0.1
```

➢ **SprBoosterApp –M 2 –n 100 –g 2 –t test.pat –d 1 train.pat booster.config**

(run 100 training cycles with Real AdaBoost and display exponential loss on test data)

What should you boost and bag?

- Boosted and bagged (plus random forest) decision trees are not news anymore. Enough examples in HEP analysis.
- Boosted random forests
  - Byron Roe et al., physics/0508045, PID at MiniBOONE. Performance same as that of optimal boosted decision trees.
- Boosted neural nets
  - Statistics literature: plenty, e.g., see next slide
  - Physics literature: Meiling, Mingmei and Lianshou, hep-ph/0606257, classification of quark and gluon jets from e+e- collisions at 91 GeV
- Bagged neural nets
  - none in physics; examples in stats literature
- Alternated boosting of decision tree and neural net
  - None to my knowledge. Seems like an obvious thing to try.

Boosted and bagged neural nets

- H. Drucker, "Boosting Using Neural Networks", in *Combining Artificial Neural Nets*, ed. A. Sharkey, Springer Series in Perspectives in Neural Computing
- Compares DT, boosted DT, NN, bagged NN and boosted NN. Boosted NN gives minimal classification error for all examples.
- 100-D data with 120k training events

Multi-class learner

- Method by Allwein, Schapire and Singer. Reduces any multi-class problem to a set of binary problems according to the user-specified class matrix. Popular strategies are one-vs-one and one-vs-all.
- Was implemented in SPR a long time ago but there used to be only one executable for multi-class learning with boosted binary splits.
- This executable has been replaced with SprMultiClassApp capable of using any classifier. Again, the user needs to choose the binary classifier in an input config file using syntax identical to booster.config.
- A beer pack for the first serious user of this algorithm is still waiting…

User tools (recent additions)

- SprInteractiveAnalysisApp
  - Interactive selection of classifiers and comparison of their performance on test data
  - Should be used only on small datasets in not too many dimensions
- SprOutputWriterApp
  - For reading stored classifier configurations and applying them to test data. Can handle any classifier and can read several classifiers at once. See example on next slide.
  - In tags prior to V05-00-00 each classifier had an individual configuration reader, e.g., SprAdaBoostDecisionTreeReader, SprBaggerDecisionTreeReader etc. In tag V05-00-00 all readers were replaced by a single class, SprClassifierReader. **This change is backwards-incompatible. You won't be able to read from classifier configuration files produced by earlier tags.**
- SprOutputAnalyzerApp
  - For people who don't like Root. Prints out efficiency curves for data and classifier responses stored in Ascii format.

Example: Analysis of Big Datasets

- **`SprBoosterApp –n 300 –f boost.spr train.pat booster.config`**

(train 300 cycles of Discrete AdaBoost and save classifier configuration into boost.spr)

- **`SprBaggerApp –n 100 –f bagger.spr train.pat bagger.config`**

(train 100 cycles of bagger and save configuration into bagger.spr)

- **`SprOutputWriterApp -C 'boost,bag' 'booster.spr,bagger.spr' test.pat test.root`**

(Read classifier configurations from booster.spr and bagger.spr and apply them to test data. Save input variables and classifier output into test.root. Classifier responses will be saved with names "boost" and "bag".)

This will work for any classifiers specified by user in booster.config and bagger.config due to the unified interface implemented in SprClassifierReader.

Plans

- Write a set of scripts to adapt the Sourceforge version to Babar and CMS environments. Someone at Caltech promised to do this for CMS. The Babar part will be my burden unless… Volunteers?
- Build a web service using Clarens framework. Clarens has been developed at Caltech and used for the most part at CMS. Use a web client to submit jobs to remote published resources.
- If you are interested in contributing, I have a laundry list of methods that would be good to implement.
- User feedback is appreciated. Not necessarily bug reports but mere statements “we applied SPR to our analysis and obtained such results” would be helpful.