Optimization of Signal Significance by Bagging Decision Trees

Ilya Narsky, Caltech

presented by Harrison Prosper

Tools for Multivariate Classification in HEP

- Physicists have some experience with
  - linear discriminant analysis (Fisher)
  - feedforward neural nets
  - radial basis function neural nets
  - Bayesian neural nets
  - decision trees
  - support vector machines
  - probability density estimation using kernels
  - nearest neighbor methods

Tools for Multivariate Classification in HEP

- New methods for combining classifiers:
  - Developed by statisticians in the last decade and largely unexplored in HEP.
  - Boosting (AdaBoost). Example: Particle identification at MiniBoone using boosted decision trees.
  - Bagging and random forests. Example: $B \rightarrow \gamma l \nu$ search at BaBar (presented in this talk).

Problems with some popular methods

- Kernel-based methods, and methods based on local averaging or nearest neighbors suffer from the “curse of dimensionality” (data are sparse in many dimensions):
  - Kernel methods are very sensitive to the choice of the distance metric.
  - “Local” methods can become non-local.
- Feedforward neural nets
  - Slow to train – CPU time scales as $O(D^2)$ if the number of hidden units is proportional to the number of input units D.
  - Strongly correlated inputs can be a problem.
  - Continuous and discrete inputs can be difficult to handle.

Boosted and bagged decision trees: Why are they better?

- Decision trees are robust in many dimensions (D). CPU time generally scales as O(D).

- A decision tree by itself is not powerful; need to boost or bag.

- With boosting or bagging one obtains what some regard as the best off-the-shelf method for classification in many dimensions.

StatPatternRecognition: A C++ package for multivariate classification

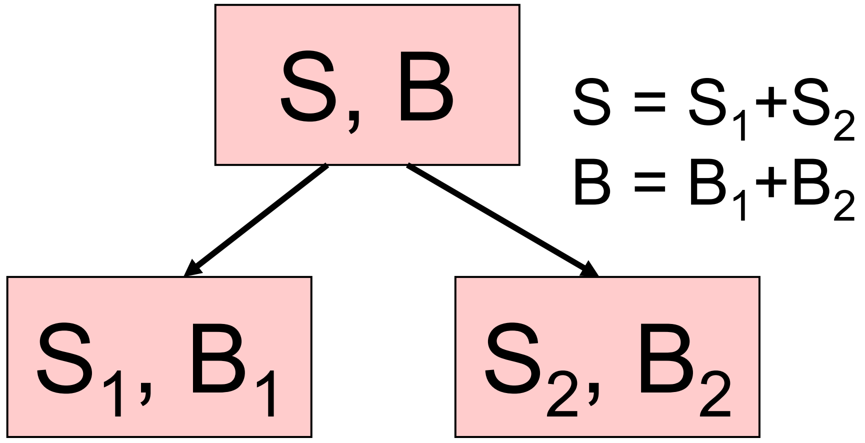
Described in: I. Narsky, physics/0507143 and physics/0507157

- Implemented classifiers and algorithms:
  - binary split
  - linear and quadratic discriminant analysis
  - decision trees
  - bump hunting algorithm (PRIM, Friedman & Fisher)
  - AdaBoost
  - bagging and random forest algorithms
  - AdaBoost and Bagger are capable of boosting/bagging any classifier implemented in the package

StatPatternRecognition

- C++ framework:
  - tools for imposing selection criteria and histogramming output
  - generic-purpose tools: bootstrap, estimation of data moments and correlation test
  - OO design – the package can be extended by supplying new implementations to existing interfaces
- Can be installed outside BaBar without too much effort.
- More details in the Poster session!

Decision trees in StatPatternRecognition



StatPatternRecognition allows the user to supply an arbitrary criterion for tree optimization by providing an implementation to the abstract C++ interface.

At the moment, following criteria are implemented:

- **Conventional: Gini index, cross-entropy, and correctly classified fraction of events.**
- **Physics: signal purity, S/√S+B, 90% Bayesian UL, and 2*(√S+B- √B).**

Conventional decision tree, e.g., CART:

- Each split minimizes Gini index:

$$Gini = \frac{S_1 B_1}{S_1 + B_1} + \frac{S_2 B_2}{S_2 + B_2}$$

- The tree spends 50% of time finding clean signal nodes and 50% of time finding clean background nodes.

Decision tree in StatPatternRecognition:

- Each split maximizes signal significance:

$$Signif = \max\left(\frac{S_1}{\sqrt{S_1 + B_1}}, \frac{S_2}{\sqrt{S_2 + B_2}} \right)$$

Boost or bag?

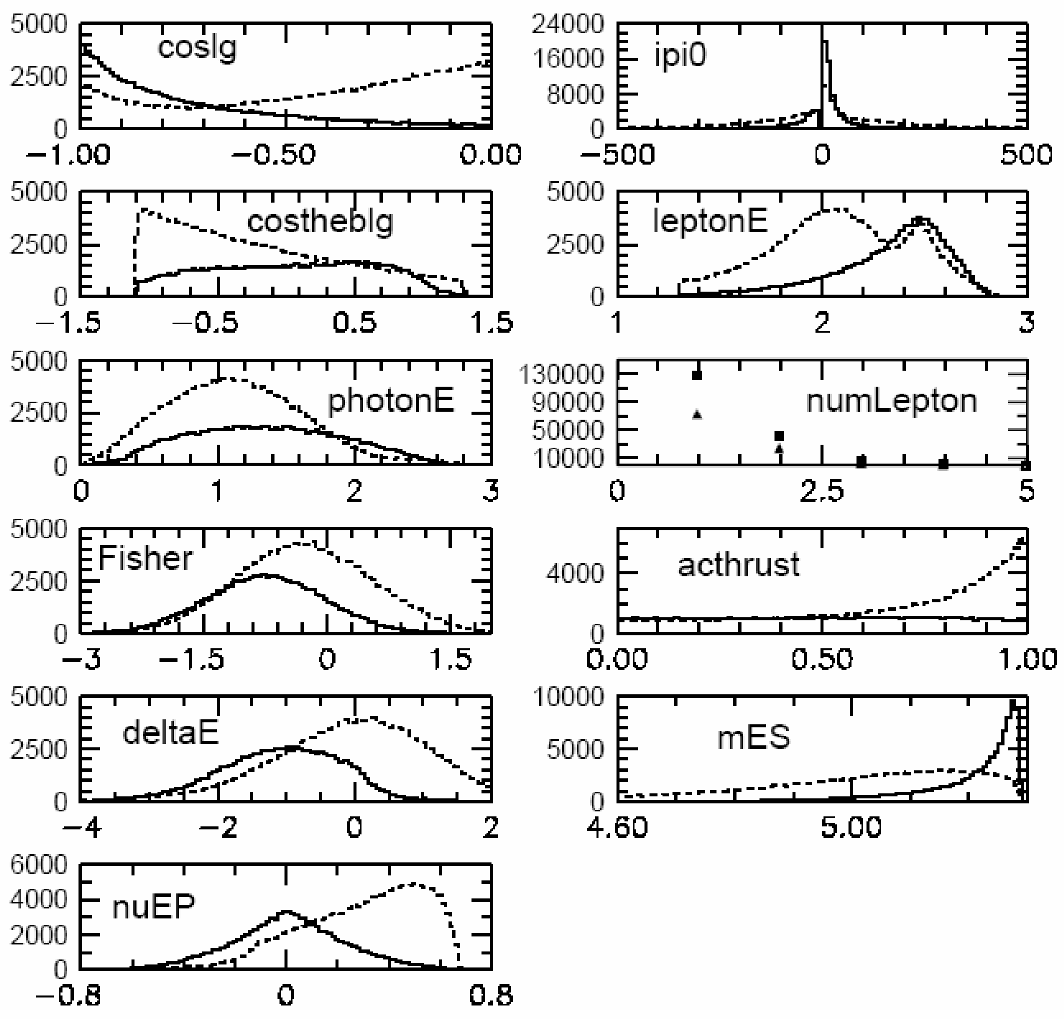
- **Boosting**
  - increase weights of misclassified events and reduce weights of correctly classified events
  - train a new classifier on the re-weighted sample
  - repeat => apply many classifiers sequentially
  - classify new data by a weighted vote of the classifiers

- **Bagging (Bootstrap AGGregatING)**
  - draw a bootstrap replica of the training sample
  - random forest: at each split, select a random sub-set of variables
  - train a new classifier on this replica
  - parallel algorithm => classifiers built on bootstrap replicas
  - classify new data by the majority vote of the classifiers

Boost or bag?

- Boosting
  - Boosting generally gives better predictive power but bagging tends to perform better in the presence of outliers and noise
    - Bauer and Kohavi, "An empirical comparison of voting classification algorithms", Machine Learning 36 (1999)
- **An interesting possibility for physics analysis:**
  - **bag decision trees optimizing a figure of merit suited for physics analysis**

Application to Search for $B \rightarrow \gamma l \nu$ at BaBar



11-D data: 10 continuous and 1 discrete variable.

Some variables are strongly correlated: ρ(Fisher,acthrust)=0.8 and ρ(costheblg,nuEP)=-0.87.

Signal significance S/√(S+B) is optimized under assumption $\mathcal{B}(B \rightarrow \gamma l \nu)=3 \cdot 10^{-6}$ in both electron and muon modes

Monte Carlo samples for each mode:

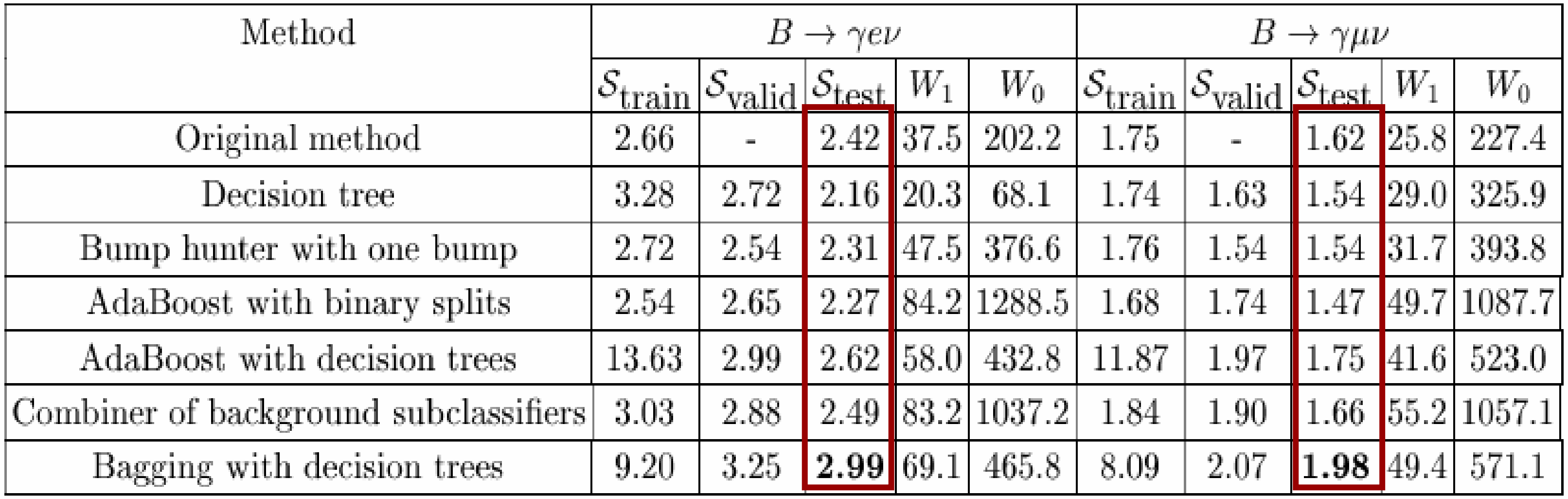
- training = 500K

- validation = 250K

- test = 250K

Results

Signal significances obtained with different classification methods



Method	$B \to \gamma e \nu$					$B \to \gamma \mu \nu$				
	$\mathcal{S}_{\text{train}}$	$\mathcal{S}_{\text{valid}}$	$\mathcal{S}_{\text{test}}$	W_1	W_0	$\mathcal{S}_{\text{train}}$	$\mathcal{S}_{\text{valid}}$	$\mathcal{S}_{\text{test}}$	W_1	W_0
Original method	2.66	-	2.42	37.5	202.2	1.75	-	1.62	25.8	227.4
Decision tree	3.28	2.72	2.16	20.3	68.1	1.74	1.63	1.54	29.0	325.9
Bump hunter with one bump	2.72	2.54	2.31	47.5	376.6	1.76	1.54	1.54	31.7	393.8
AdaBoost with binary splits	2.54	2.65	2.27	84.2	1288.5	1.68	1.74	1.47	49.7	1087.7
AdaBoost with decision trees	13.63	2.99	2.62	58.0	432.8	11.87	1.97	1.75	41.6	523.0
Combiner of background subclassifiers	3.03	2.88	2.49	83.2	1037.2	1.84	1.90	1.66	55.2	1057.1
Bagging with decision trees	9.20	3.25	**2.99**	69.1	465.8	8.09	2.07	**1.98**	49.4	571.1

Training 50 boosted decision trees or 100 bagged decision trees took several hours in a batch queue at SLAC.

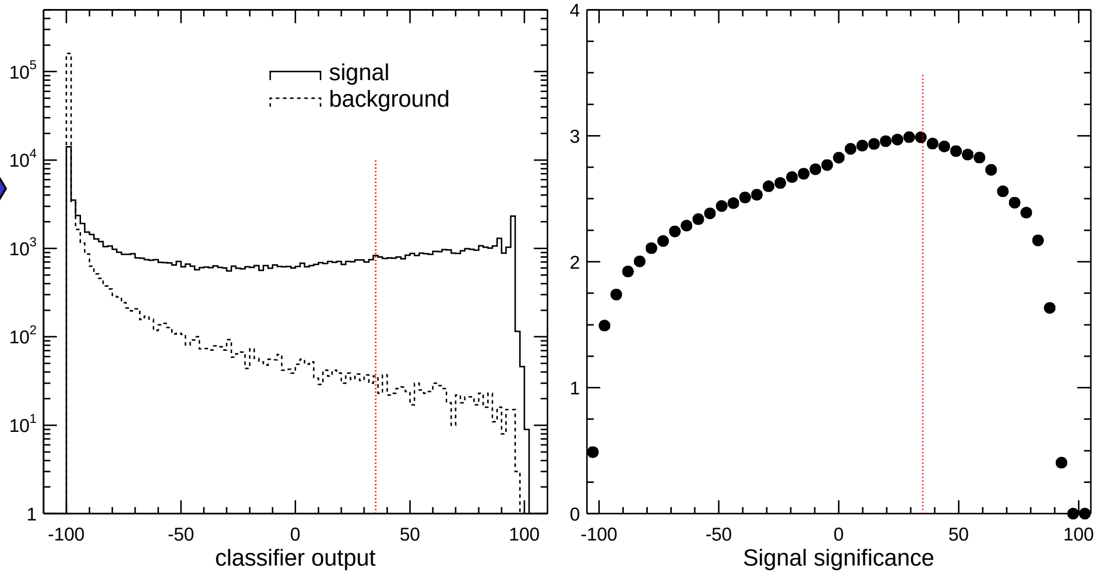
W_1 and W_0 are expected numbers of signal and background events in 210 fb^{-1} of data.

The best signal significance is obtained by bagging decision trees.

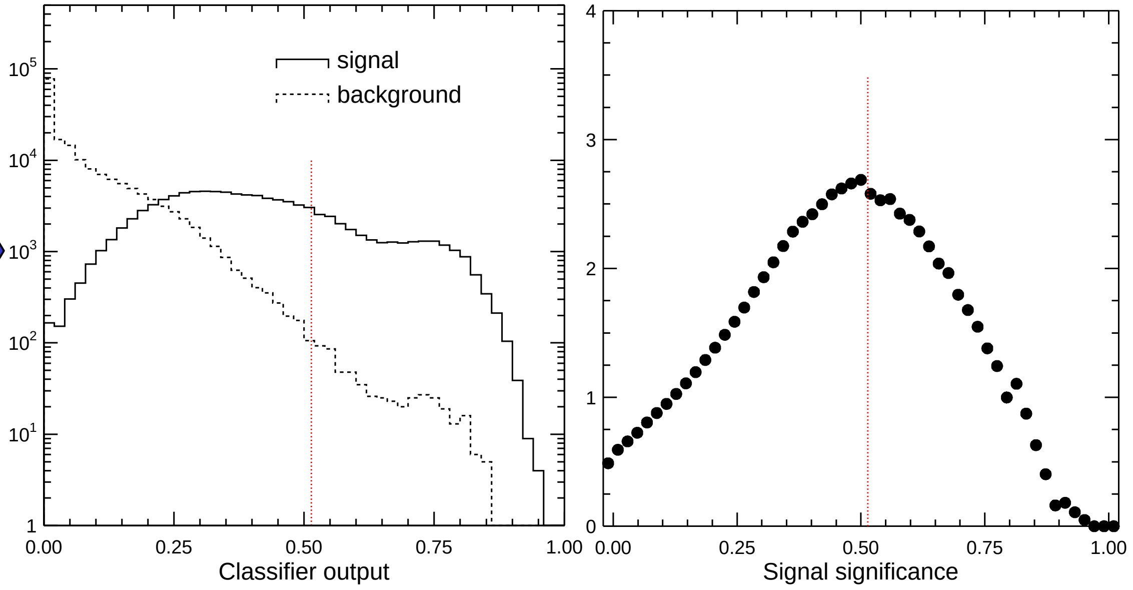
Bagged decision trees give a 14% improvement over boosted decision trees:

- **8% due to using bagging instead of boosting**
- **9% due to optimizing $S/\sqrt{(S+B)}$ instead of Gini index**

Bagging 100 decision trees optimizing the signal significance



Boosting 50 decision trees optimizing the Gini index



Summary

- Bagging decision trees that optimize a figure of merit suited for physics analysis is an interesting option. There is at least one example where this method is superior to all others.
- StatPatternRecognition is available for distribution to physicists. Email to narsky@hep.caltech.edu to request a copy.