Multivariate Methods in Particle Physics Today and Tomorrow

Harrison B. Prosper

Florida State University

5 November, 2008

ACAT 08, Erice, Sicily

Outline

- Introduction
- Multivariate Methods
  - In Theory
  - In Practice
- Outstanding Issues
- Summary

Introduction

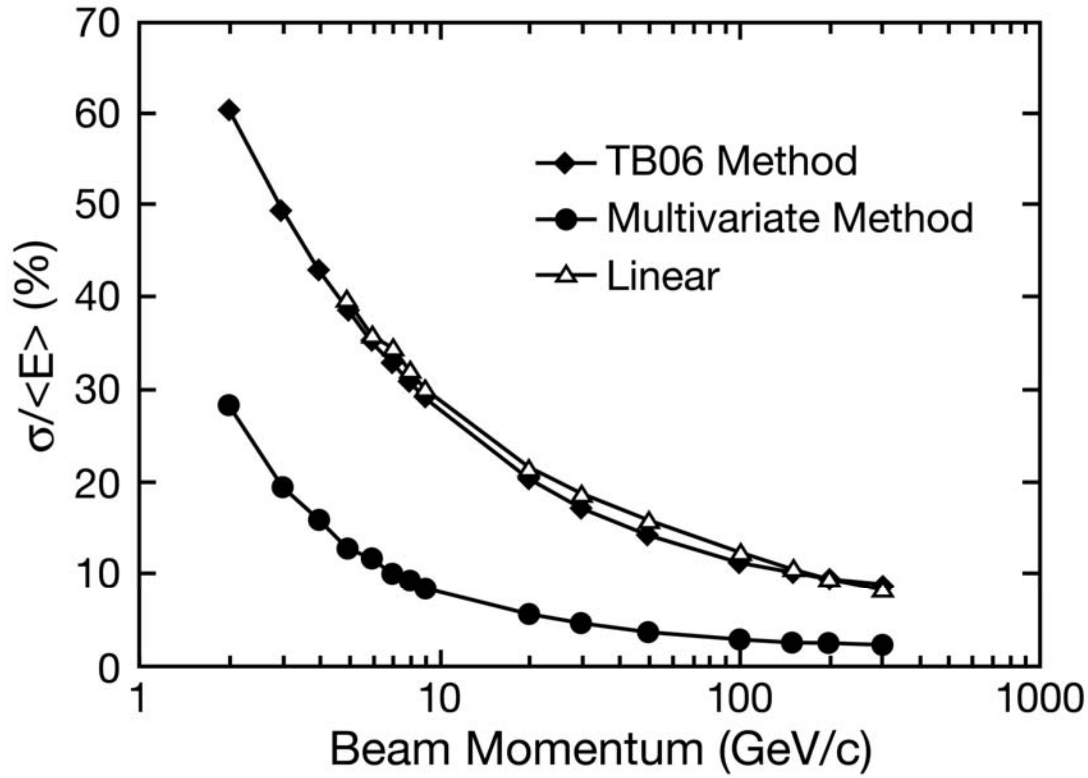
Multivariate methods can be useful in:

- Classification
- Function approximation
- Probability density estimation
- Data compression
- Variable selection
- Optimization
- Model comparison

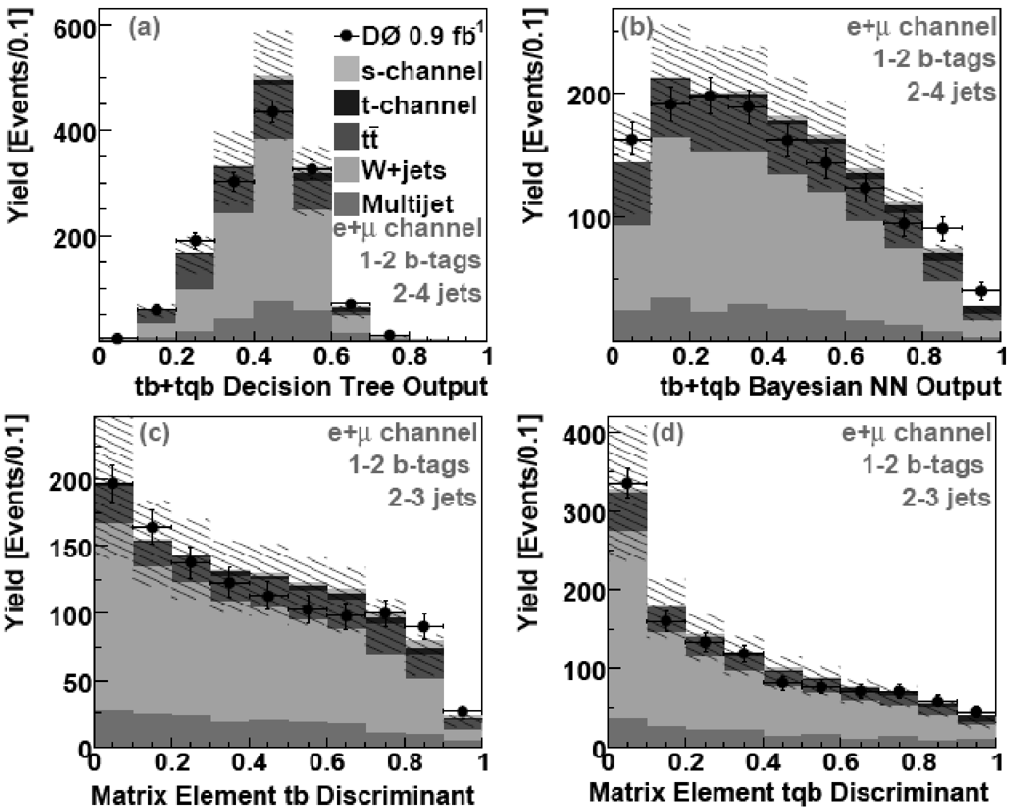
Example – Energy Measurements

Regression using neural networks to estimate single particle energies.

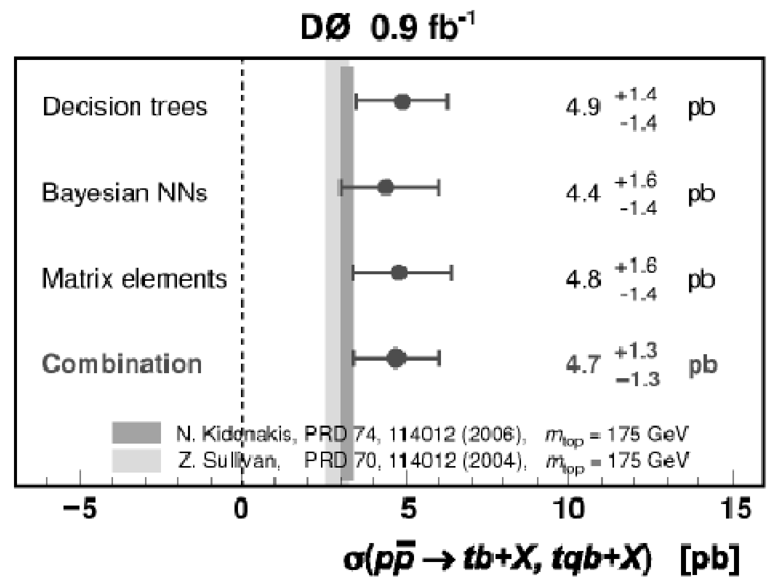
See poster by
Sergei Gleyzer
CMS Collaboration



Example – Single Top Search



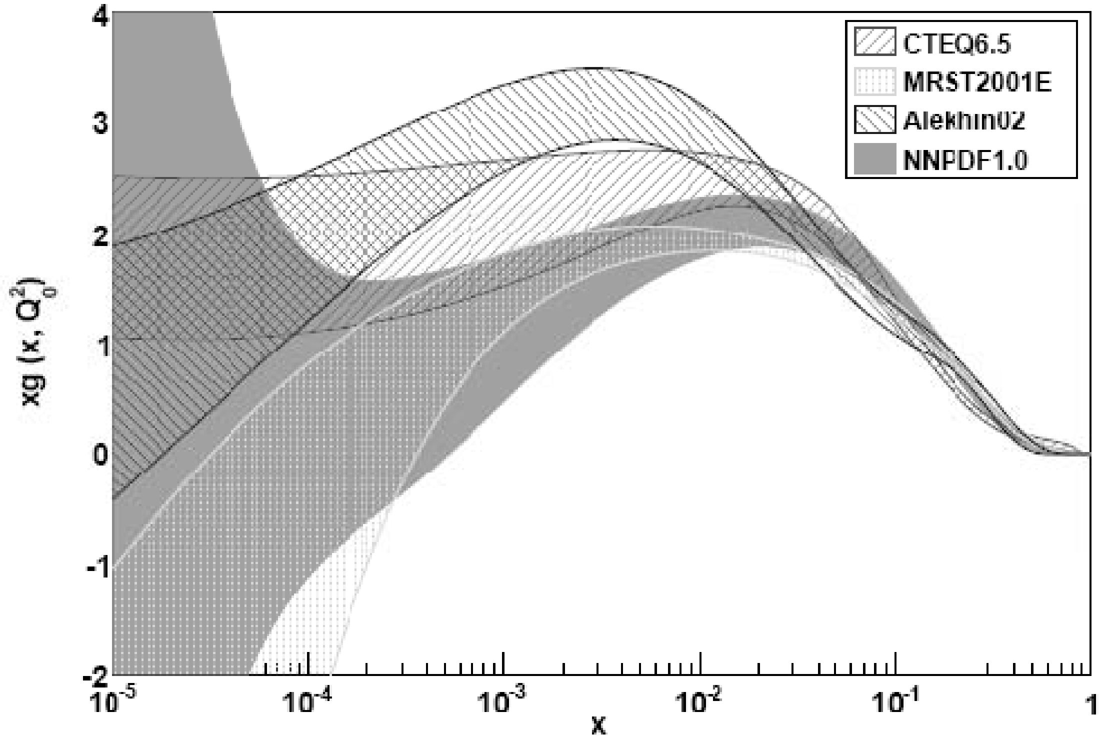
Single top quark search using
boosted decision trees
Bayesian neural networks
matrix element method



Dzero Collaboration,
PRD 78 012005, 2008

Example – Parton Distributions

Gluon distribution



PDFs modeled with **neural networks**, fitted using a **genetic algorithm**

The **NNPDF Collaboration**, R.D. Ball et al., arXiv: 0808.1231v2

Multivariate Methods: In Theory

Multivariate Methods

Two general approaches:

Machine Learning

Teach a machine to learn $y = f(x)$ by feeding it **training data** $\boldsymbol{T} = (\mathbf{x}, \mathbf{y}) = (x,y)_1, (x,y)_2,\ldots(x,y)_N$ and a **constraint** on the class of functions.

Bayesian Learning

Infer $y = f(x)$ given the **conditional likelihood** $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ for the training data and a **prior** on the space of functions $f(x)$.

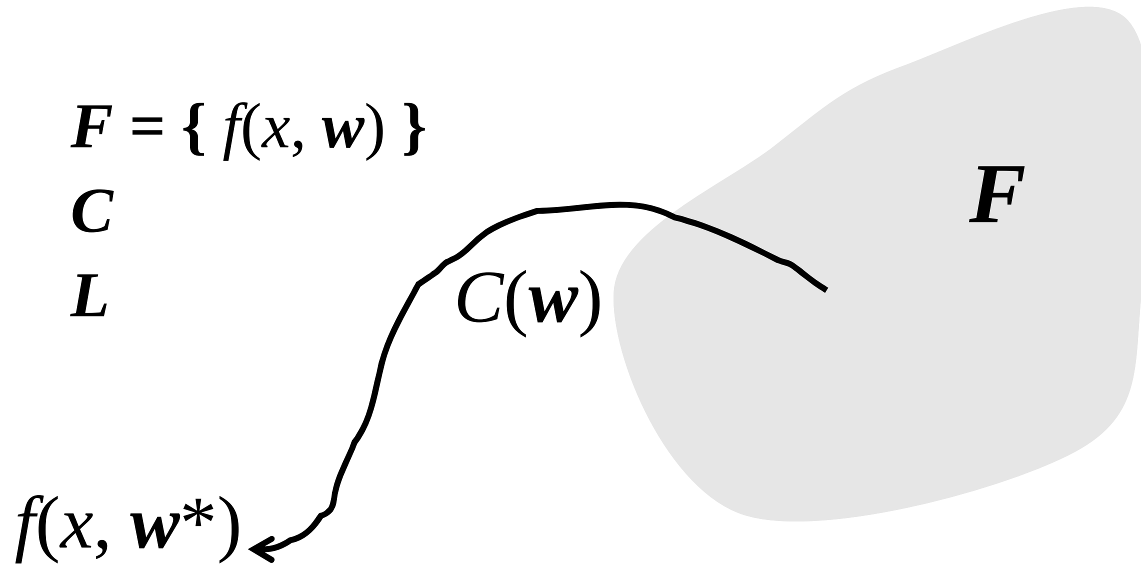
Machine Learning

Choose

Function class — $\boldsymbol{F} = \{ f(x, \boldsymbol{w}) \}$

Constraint — $\boldsymbol{C}$

Loss function — $\boldsymbol{L}$



Method

Find $f(x)$ by minimizing the **empirical risk *R***

$$R(w) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i, w))$$

subject to the constraint $C(\boldsymbol{w})$

Bayesian Learning

Choose

Function class	$\boldsymbol{F} = \{ f(x, \boldsymbol{w}) \}$
Prior	$\pi(\boldsymbol{w})$
Likelihood	$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$

Method

Use Bayes' theorem to infer the parameters:

$$\begin{aligned} p(\boldsymbol{w}|\boldsymbol{T}) &= p(\boldsymbol{T}|\boldsymbol{w})\ \pi(\boldsymbol{w})/p(\boldsymbol{T}) \\ &= p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})\ p(\boldsymbol{x}|\boldsymbol{w})\ \pi(\boldsymbol{w})/p(\boldsymbol{y}|\boldsymbol{x})\ p(\boldsymbol{x}) \\ &\sim p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})\ \pi(\boldsymbol{w}) \qquad \text{(assume } p(\boldsymbol{x}|\boldsymbol{w}) = p(\boldsymbol{x})\text{)} \end{aligned}$$

$p(\boldsymbol{w}|\boldsymbol{T})$ assigns a probability density to every function in the function class.

Regression

Many methods (e.g., neural networks, boosted decision trees, rule-based systems, random forests, etc.) are based on the **mean square empirical risk**

$$R(w) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i, w))^2$$

In the machine learning approach ***R*** is minimized with respect to the parameters, subject to the constraint.

In the Bayesian approach, one writes (typically) $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \exp(-N\, R / 2\sigma^2)/\sigma\sqrt{2\pi}$, computes the **posterior density** $p(\boldsymbol{w}|\boldsymbol{T})$, and then the **predictive distribution**:

$$\boxed{p(y \mid x, T) = \int p(y \mid x, w)\, p(w \mid T)\, dw}$$

Classification

If ***y*** has only two values **0** and **1**, then the ***mean*** of the predictive distribution

$$f(x) = \int y \, p(y \mid x, T) dy$$

reduces to

$$f(x) = p(S \mid x) = \frac{p(x \mid S)\, p(S)}{p(x \mid S)\, p(S) + p(x \mid B)\, p(B)}$$

where ***S*** is associated with *y* = **1** and ***B*** with *y* = **0**. This yields the **Bayes classifier** *if p*(***S***|*x*) **>** ***q*** *accept x as belonging to* ***S***.

A Bayes classifier is ***optimal*** in the sense that it achieves the ***lowest misclassification rate***.

Classification

In practice, it is sufficient to approximate the **discriminant**

$$D(x) = \frac{p(x \mid S)}{p(x \mid S) + p(x \mid B)}$$

because $D(x)$ and $p(\boldsymbol{S}|x)$ are related one-to-one:

$$p(S \mid x) = \frac{D(x)}{D(x) + [1 - D(x)] / A}$$

where $\boldsymbol{A} = p(\boldsymbol{S}) \,/\, p(\boldsymbol{B})$ is the prior signal to background ratio.

Classification – Points to Note

1. If your goal is to ***classify objects*** with the fewest errors, then the **Bayes classifier** is the ***optimal*** solution.

2. Consequently, if you have a classifier known to be ***close*** to the **Bayes limit**, then *any* other classifier, *however sophisticated it might be*, can ***at best*** be only marginally better than the one you have.

3. *All* classification methods, such as the ones in TMVA, are different numerical approximations of some function of the Bayes classifier.

Event Weighting

The probability $p(S|x)$ is optimal in another sense:

If one ***weights*** an admixture of **signal** and **background** events by the weight function

$$W(x) = p(S|x)$$

then the ***signal*** strength will be extracted with ***zero bias*** and the ***smallest possible variance***, provided that our models describe the signal and background densities accurately and the signal to background ratio $p(S)/p(B)$ is equal to the true value.

Roger Barlow, J. Comp. Phys. 72, 202 (1987)

Historical Aside – Hilbert's 13th Problem

Problem 13: Prove the conjecture

In general, it is ***impossible*** to do the following:

$$f(x_1,\ldots,x_n) = F(\, g_1(x_1),\ldots,\, g_n(x_n)\,)$$

But, in 1957, Kolmogorov ***disproved*** Hilbert's conjecture!

Today, we know that functions of the form

$$f(x_1,\cdots,x_n) = b + \sum_{i=1}^{H} v_i \tanh\left[a_i + \sum_{j=1}^{n} u_{ij} x_j \right]$$

can provide arbitrarily accurate approximations.

Multivariate Methods: In Practice

Introduction

A Short List of Multivariate Methods

- Random Grid Search
- Linear Discriminants
- Quadratic Discriminants
- Support Vector Machines
- Naïve Bayes (Likelihood Discriminant)
- Kernel Density Estimation
- Neural Networks
- Bayesian Neural Networks
- Decision Trees
- Random Forests
- Genetic Algorithms

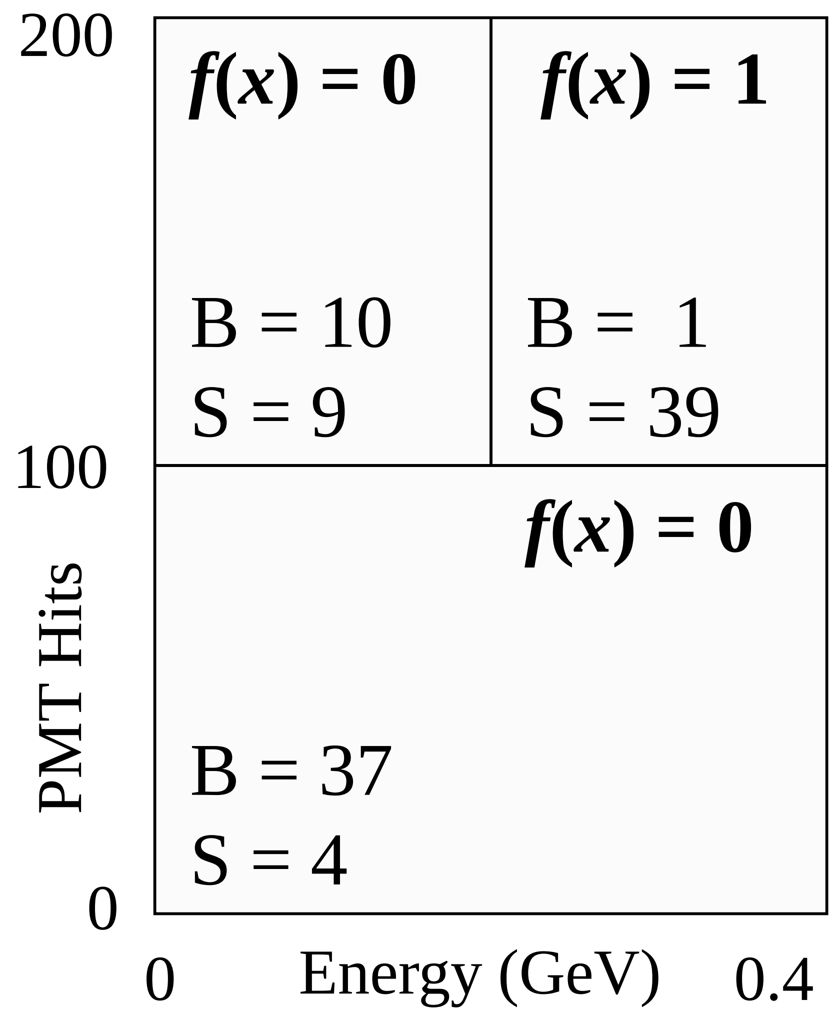
Decision Trees

A decision tree is an **n-dimensional histogram** whose bins are constructed recursively.

Each bin is associated with the value of the function ***f*(*x*)** to be approximated.

The partitioning of a bin is done using the ***best*** cut.

There are many ways to define best! (See, e.g., TMVA.)



MiniBoone, Byron Roe

Ensemble Learning

A few popular methods (used mostly with decision trees):

- **Bagging**: each tree trained on a **bootstrap sample** drawn from training set
- **Random Forest**: bagging with **randomized** trees
- **Boosting**: each tree trained on a **different weighting** of full training set

$$f(x) = a_0 + \sum_{k=1}^{K} a_k f(x, w_k)$$

Jeromme Friedman & Bogdan Popescu

Adaptive Boosting

Repeat K times:

1. Create a decision tree $f(x, \boldsymbol{w})$
2. Compute its error rate ε on the ***weighted*** training set
3. Compute $\alpha = \ln (1 - \varepsilon) / \varepsilon$
4. Modify training set: ***increase weight*** of ***incorrectly classified examples*** relative to those that are correctly classified

Then compute weighted average $f(x) = \sum \alpha_k \ f(x, w_k)$

Y. Freund and R.E. Schapire.
Journal of Computer and Sys. Sci. **55** (1), 119 (1997)

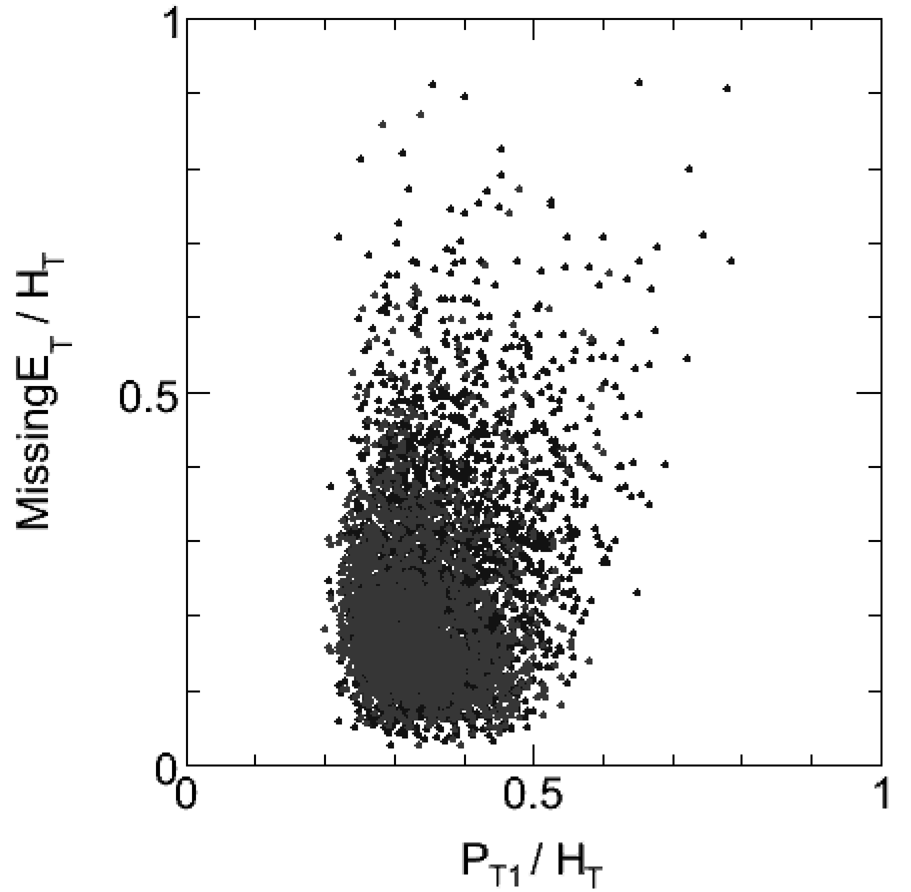
AdaBoost - Example

mSUGRA
@ focus point

vs

ttbar

growing trees



AdaBoost - Example

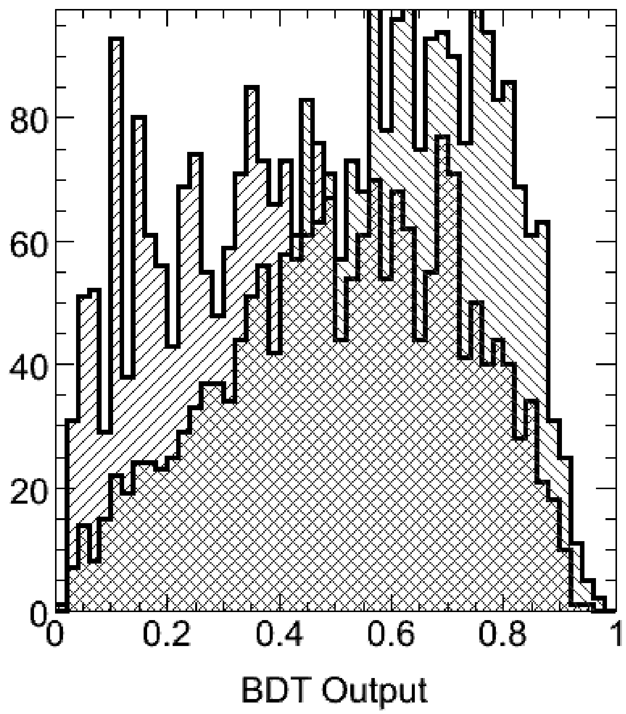
mSUGRA @ focus point

vs

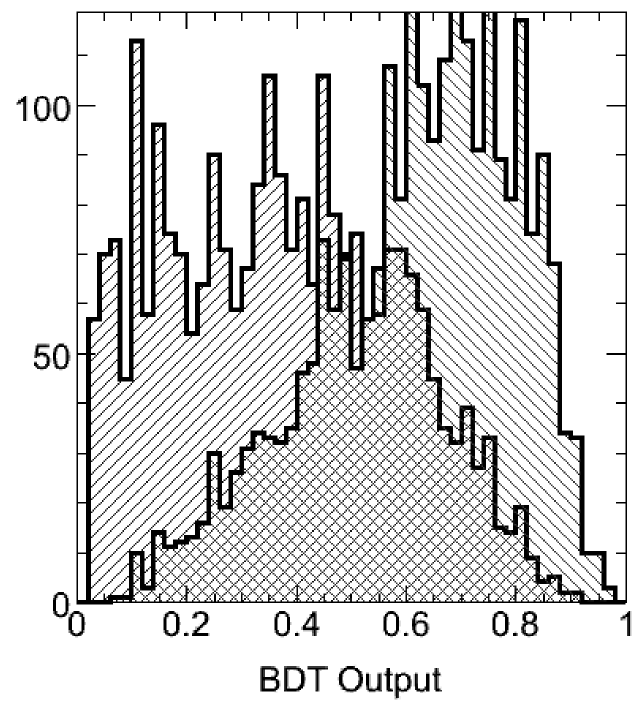
ttbar

Signal/background discrimination, averaging over an increasing number of trees, up to 1000

test sample



training sample



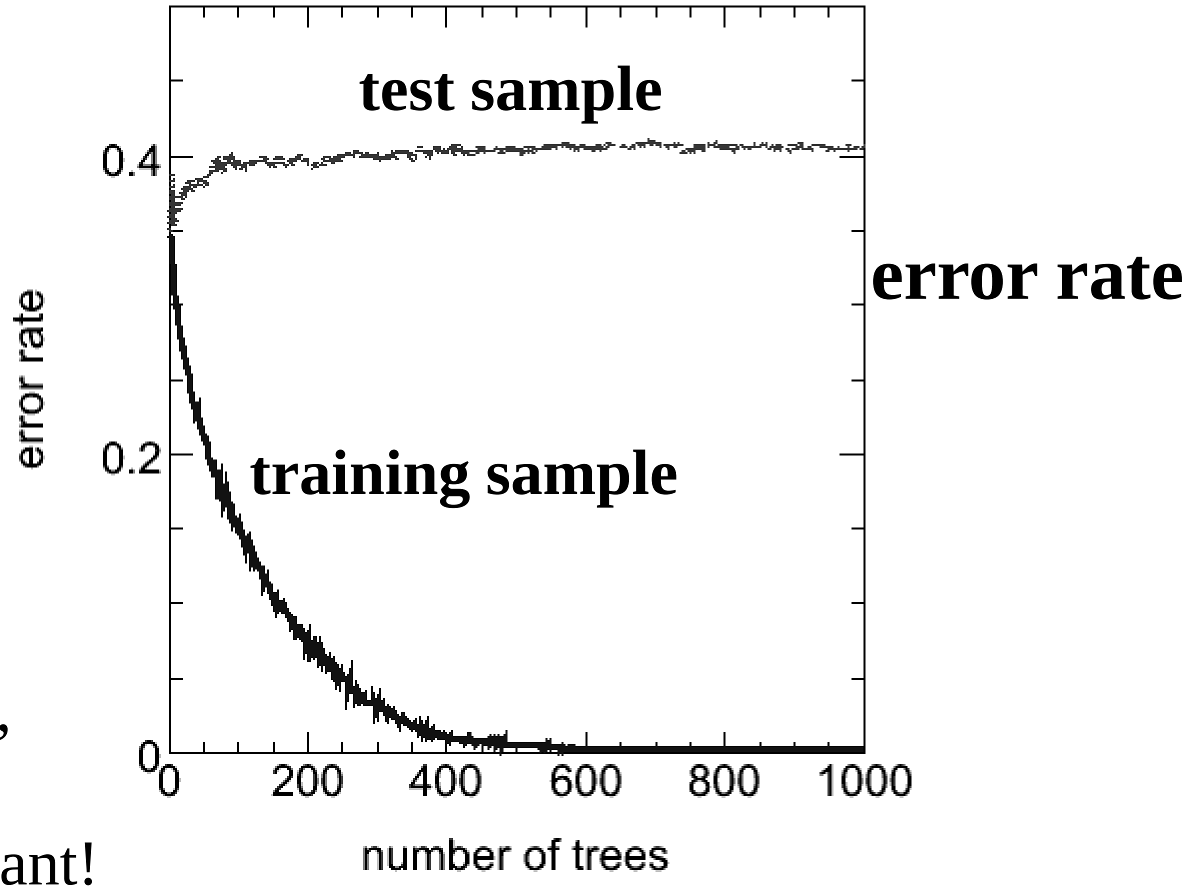
AdaBoost - Example

mSUGRA
@ focus
point

vs

ttbar

Training error goes
to ***zero*** exponentially,
while test error
remains almost constant!



Bayesian Neural Networks

Given

$$p(\boldsymbol{w}|\boldsymbol{T}) \sim p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})\, \pi(\boldsymbol{w})$$

where

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \prod \text{Gaussian}(y_k, f(x_k, \boldsymbol{w}), \sigma) \qquad \text{(for } \textbf{regression}\text{)}$$

or $$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) = \prod n(x_k, \boldsymbol{w})^{\mathrm{y}}\, [1 - n(x_k, \boldsymbol{w})]^{1-\mathrm{y}} \qquad \text{(for } \textbf{classification}\text{)}$$

and $$n(\mathbf{x}, \boldsymbol{w}) = 1/[1+\exp(-f(\mathbf{x}, \boldsymbol{w}))]$$

Compute

$$y(\mathbf{x}) = \int f(\boldsymbol{x}, \boldsymbol{w})\, p(\boldsymbol{w}|\boldsymbol{T})\, \mathrm{d}\boldsymbol{w} \text{ or } n(\mathbf{x}) = \int n(\boldsymbol{x}, \boldsymbol{w})\, p(\boldsymbol{w}|\boldsymbol{T})\, \mathrm{d}\boldsymbol{w}$$

$y(\boldsymbol{x})$ and $n(\boldsymbol{x})$ are called **Bayesian neural networks** (BNN).

The integrals are approximated using a **MCMC method** (Radford Neal, http://www.cs.toronto.edu/~radford/fbm.software.html).

BNN – Classification Example

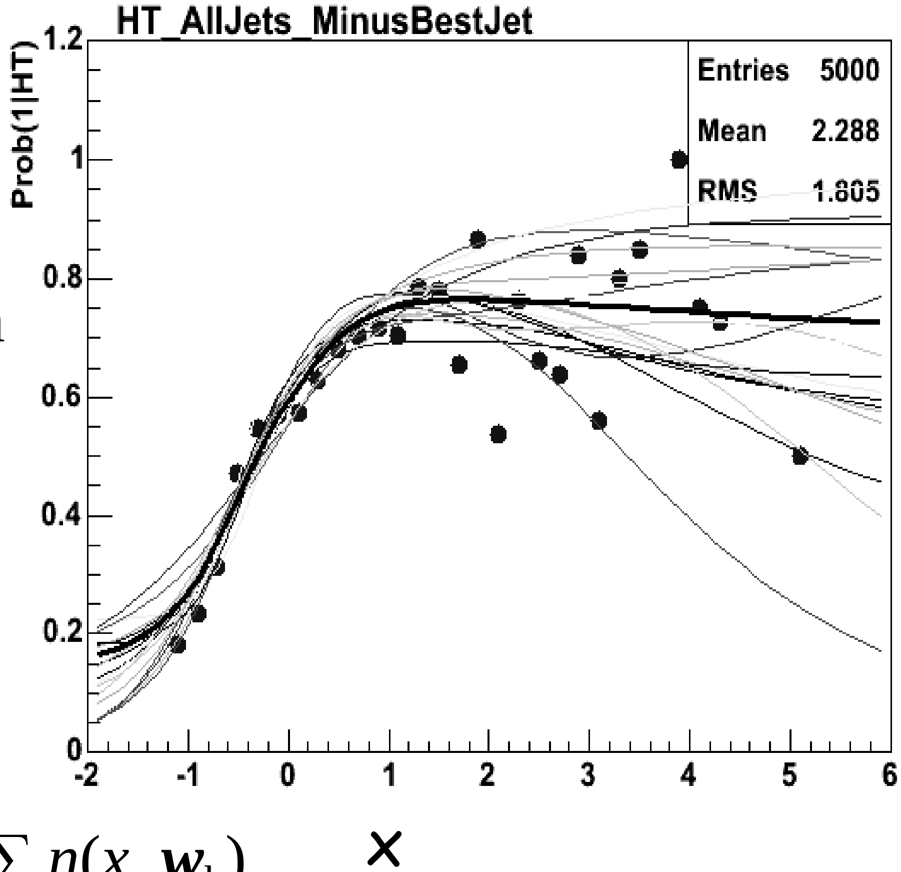
Dots

$D(x) = H_S/(H_S+H_B)$

H_S signal histogram
H_B background histogram

Curves

Individual neural networks
$n(x, \mathbf{w}_k)$



Black curve

$D(x) = E[\, n(x, \mathbf{w})\,] = (1/N) \sum n(x, \mathbf{w}_k)$

Outstanding Issues

Tuning Methods

- Is cross-validation sufficient to choose the function class (number of leaves, number of trees, number of hidden nodes etc.)?

Verification

- How can one confirm that an ***n-dimensional*** density is well-modeled?

- How can one find, characterize, and exclude, discrepant domains in n-dimensions ***automatically***?

Some Issues

Verification…

- Can one automate ***re-weighting*** of model data, event-by-event, to improve the match between real data and the model?

- How can one verify that $f(x)$ is close to the Bayes limit?

Looking Beyond the Lamppost

- Is there a sensible way to use multivariate methods when one does not know for certain where to look for signals?

Verification

Discriminant Verification

Any classifier ***f*(*x*)** close to the Bayes limit approximates

$$D(\mathrm{x}) = p(x|S) \,/\, [\; p(x|S) + p(x|B) \;]$$

Therefore, if we weight, ***event-by-event***, an admixture of *N* signal and *N* background events by the function ***f*(*x*)**

$$S_w(x) = N\, p(x|S)\, \boldsymbol{f(x)}$$

$$B_w(x) = N\, p(x|B)\, \boldsymbol{f(x)}$$

then the sum

$$S_w(x) + B_w(x) = N\,(p(x|S) + p(x|B))\, \boldsymbol{f(x)} = \boldsymbol{N\, p(x|S)},$$

i.e., we should recover the n-dimensional ***signal density***.

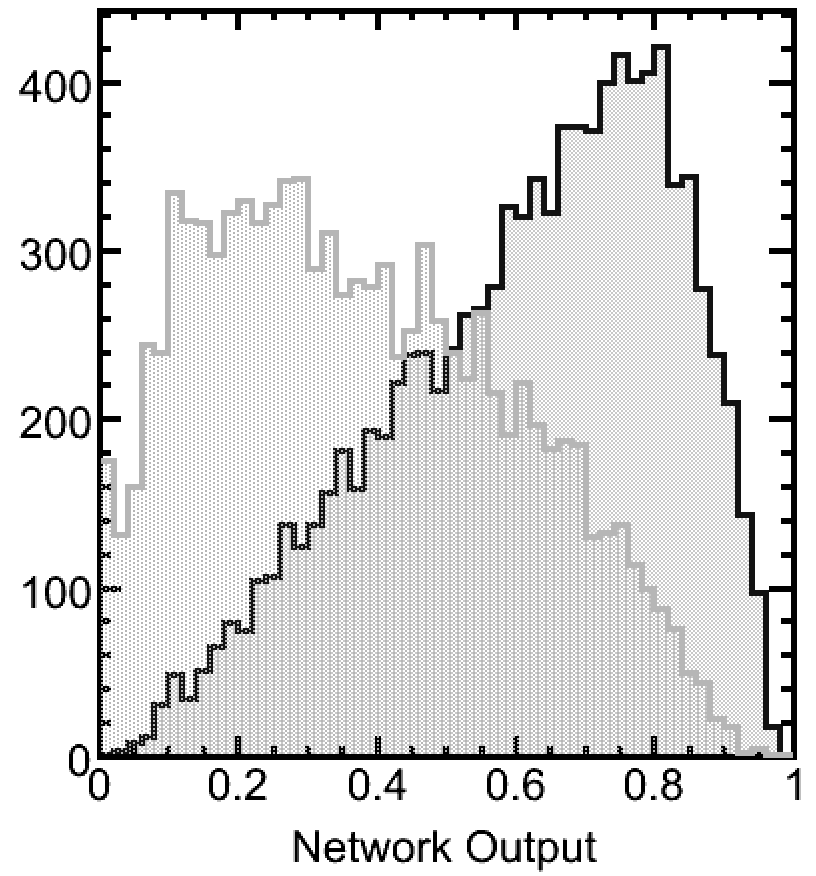
Verification – Example

Dzero single top quark search

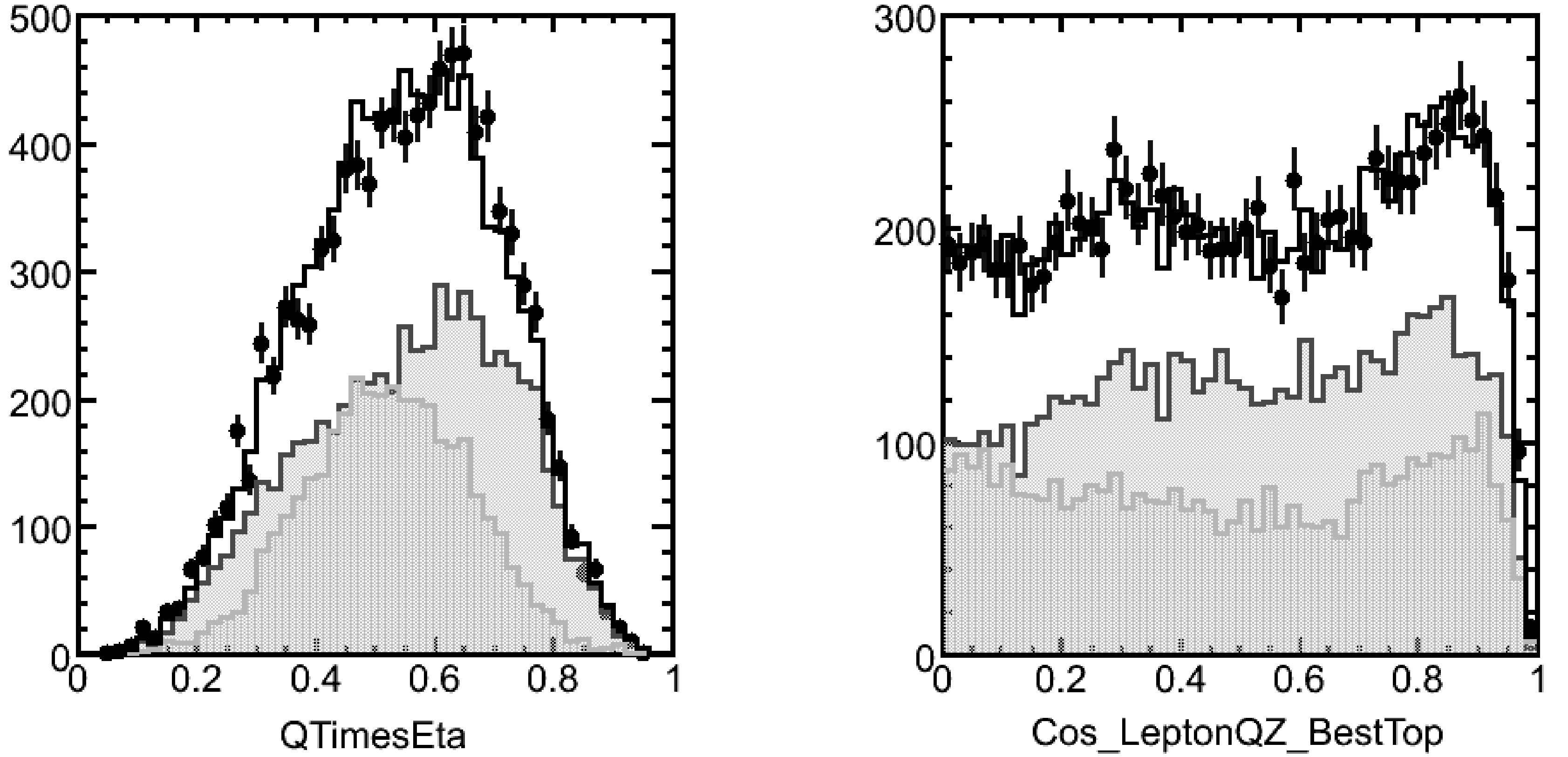
Verifying the Bayesian neural network discriminant.

Number of input variables ~ 24
Number of channels = 12

(e, μ) x (1, 2) b-tags x (2,3,4) jets



Verification – Example



Cyan plot: weighted signal **Green plot**: weighted background
Black curve: sum **Black dots**: signal

Summary

- Multivariate methods can be applied to many aspects of data analysis.
- Many practical methods, and convenient tools such as TMVA, are available for regression and classification.
- All methods approximate the same mathematical entities, but no one method is guaranteed to be the best in all circumstances. So, experiment with a few of them!
- Several issues remain. The most pressing is the need for sound methods, and convenient tools, to explore and quantify the quality of modeling of n-dimensional data.