PROCEEDINGS OF SCIENCE

Multivariate Methods in Particle Physics Today and Tomorrow

H. B. Prosper*
Department of Physics, Florida State University, Tallahassee, USA
E-mail: harry@hep.fsu.edu

Multivariate methods are used routinely in particle physics research to classify objects or to discriminate signal from background. They have also been used successfully to approximate multivariate functions. Moreover, as is evident from this conference, excellent easy-to-use implementations of these methods exist, making it easy to deploy these methods in data analysis. However, these methods are sophisticated. Therefore, from time to time, it is helpful to step back and reflect on what is being done. That is the aim of this paper. I begin with a review that places these (supervised learning) methods into a broader context and follow this with a survey of a few of the most promising recent developments. I end with an enumeration of what I consider to be the most pressing issues.

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7 2008
Erice, Italy

*Speaker.

PoS(ACAT08)010

1. Introduction

A multivariate method is any statistical technique that manipulates multiple variables simultaneously. Such methods find application in many tasks that arise in data analysis including

- object classification,
- function approximation,
- probability density estimation,
- data compression,
- variable selection,
- optimization,
- model comparison and hypothesis testing.

Most interesting data are multidimensional. A topical example in particle physics is the data that were searched recently for evidence of the creation of single top quarks in proton-antiproton collisions [1, 2, 3]. In these data, the final state of a typical single top quark event, comprising an electron or a muon, 2, 3 or 4 jets of hadrons, and missing transverse momentum, is characterized by about 20 measured quantities, which are simple functions of the underlying degrees of freedom of these events. For example, in an event with one electron, 2 jets, and missing transverse momentum, the number of degrees of freedom (taking each object as massless) is $3+2\times 3+2=11$. Another example arises in the correction of jet transverse momenta (see, for example, Ref. [4]). A typical correction function depends on the transverse momentum of the jet, its direction, and perhaps on variables that characterize how the jet energy has been distributed in the various particle calorimeters. Today, it is hard to find a topic at the frontier of particle physics research for which a single measured quantity is an adequate basis for analysis. There is no reason to expect this situation to change in the era of the Large Hadron Collider (LHC). On the contrary, it is quite clear that for the problems of tomorrow, the use of multivariate methods will be a necessity. A particularly interesting example of the benefits of a fully multivariate approach to a problem is the recent development, by the NNPDF Collaboration, of a promising technique to model parton distribution functions using neural networks with their parameters determined using a genetic algorithm [5].

Until relatively recently, data analysis using multivariate methods required familiarity with the details of a disparate collection of software tools and methods, rendering it difficult for the non-expert to deploy these methods. However, today with the advent of tools such as TMVA [6] anyone with modest software skills can make use of such methods in his or her data analysis. This is a good thing because practitioners are now freed from the need to get mired in technicalities and may, instead, focus on the problem being addressed. But there is a danger: the very simplicity of these tools encourages the view of multivariate methods as "black boxes", to treat them as such, and to use them with insufficient reflection on what is being done. A thing remains a black box, however, only if one refuses to peer inside. My goal here is to place these methods into a broader context and take a peek inside one or two of them, focusing on the problem of binary classification.

The paper is organized as follows. I begin with a discussion of classification in theory in the context of supervised learning, that is, the construction of classifiers using a representative sample of objects whose identity is known. (This is not a serious restriction because the overwhelming majority of applications in particle physics are of this kind.) Next, I discuss practical implementations of that theory. I end with an enumeration of outstanding issues and a summary.

2. Multivariate Methods: In Theory

In many applications, the fundamental multivariate task can be construed as follows: given *training* data $T = \{(y_1, x_1), (y_2, x_2), \cdots\}$, where $y_i \in \subset \mathbb{R}$—either $\mathbb{R}$ itself, [−1,1], [0,1], or the discrete sets $\{-1, 1\}$ or $\{0, 1\}$—and $x_i \in \subset \mathbb{R}^d$, approximate the function $y = f(x)$. If y is continuous the task is called regression; if y is discrete we are engaged in classification or discrimination. The jet energy correction problem is an example of regression, while the search for single top quark production [1, 2] is a recent example of the use of multivariate methods to discriminate signal events from background events. The values y_i are generally referred to as *targets*, while the values x_i are variously referred to as *inputs* or *feature vectors*.

Multivariate methods fall into two broad approaches, *machine learning* [7, 8] and *Bayesian learning* [9], each with its own community of researchers. The methods, jargon, and even theory, of each camp appear very different at first sight. However, upon closer inspection they are not as different as they seem. The key point is that both approaches are ultimately grounded in statistical theory.

2.1 Machine Learning

In this approach, the construction of an approximation to the function $y = f(x)$ is viewed as a problem of optimization; that is, one arrives at the approximation by minimizing an appropriate functional. The basic ingredients are:

- a class of parameterized functions $\mathbb{F} = \{f(x, w)\}$, where the parameters w are to be found by the optimization procedure;
- a constraint $C(w)$ on the class of functions, and
- a loss function, $L(y, f)$, that quantifies the loss incurred by a poor choice of the function $f(x, w)$ from the function class $\mathbb{F}$.

In practice, because one wants the selection of the function $f(x, w)$, from the function class, to be robust with respect to the choice of training data it is better to average the loss function over the training data. This defines the *empirical risk* function $R(w)$

$$R(w) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f_i), \tag{2.1}$$

where $f_i \equiv f(x_i, w)$ and N is the size of the training sample. The practical task is to minimize the empirical risk, subject to the constraint $C(w)$, that is, to minimize the objective function

$$E(w) = R(w) + \lambda C(w), \tag{2.2}$$

where λ is a tuning parameter that determines the severity of the constraint. By minimizing $E(w)$, a function $f(x,w^*)$ will be selected from the function class, which in the limit $N \to \infty$ converges to the one that would have been selected were it possible to minimize the true risk [7]. A well-known example of empirical risk minimization is a constrained χ^2 fit.

2.2 Bayesian Learning

In the Bayesian approach, an approximation to the function $y = f(x)$ is constructed by casting the problem as one of inference [9]. This requires the specification of

- a class of parameterized functions $\mathbb{F} = \{f(x,w)\}$;
- a prior density $\pi(f)$ over the space of functions, though in practice it is easier to use a prior $\pi(w)$ defined over the space of parameters, and
- a likelihood function, $p(y|x,w)$, proportional to the probability that the value of the function $f(x)$ is y given inputs x.

A significant advantage of the Bayesian approach is that *all* inference problems are solved the same way [10]. For the task at hand, the problem is to infer plausible values of the parameters w, and therefore plausible choices for the function $f(x,w)$, given the training sample T. This is done by computing the posterior probability $p(w|T)dw$, that is, the probability that the choice w is compatible with the training data T. The standard way to effect this calculation is to use Bayes' theorem

$$\begin{aligned} p(w|T) &= \frac{p(T|w)\pi(w)}{p(T)}, \\ &= \frac{p(y,x|w)\pi(w)}{p(y,x)}, \\ &= \frac{p(y|x,w)p(x|w)\pi(w)}{p(y|x)p(x)}, \\ &\sim p(y|x,w)\pi(w), \end{aligned} \tag{2.3}$$

where we have made the reasonable assumption that $p(x|w) = p(x)$, that is, that the probability density of the inputs is independent of the parameters of the function to be inferred. The function $p(w|T)dw$ assigns a probability to the functions $f(x,w)$ in the neighborhood, $[w, w+dw]$, of the point w. Larger values of $p(w|T)$ imply functions, $f(x,w)$, that are better matched to the training data.

Given the posterior density $p(w|T)$, the natural quantity to compute, in the Bayesian context [10], is the *predictive distribution*

$$p(y|x,T) = \int p(y|x,w)p(w|T)dw, \tag{2.4}$$

that is, the probability density that the value of the function is y, given input values x. Often, however, one wants a definite estimate of $y = f(x)$. This can be achieved by minimizing the following risk function

$$y = \arg_y \min \int L(z,y)p(z|x,T)dz, \tag{2.5}$$

with respect to y, where $L(z,y)$ is some loss function of which a common choice is the *quadratic loss* defined by

$$L(a,b) = (a-b)^2. \tag{2.6}$$

With this choice of loss function the optimal estimate of the function $y = f(x)$ is given by the mean of the predictive distribution,

$$y = \int z p(z|x,T) dz. \tag{2.7}$$

2.3 Machine versus Bayesian Learning

Although the two approaches sketched above appear rather different, in fact they are not as different as they seem. If we are prepared to make the following identifications

$$\begin{aligned} R(w) &\sim \ln p(y|x,w), \\ &= \sum_{i=1}^{N} \ln p(y_i|x_i,w), \end{aligned} \tag{2.8}$$

$$\lambda C(w) \sim \ln \pi(w), \tag{2.9}$$

then it becomes clear that the machine learning approach can be viewed as providing a *maximum a posteriori* (MAP) estimate of the function $y = f(x)$. In other words, minimizing the constrained empirical risk, Eq.(2.2), is equivalent to maximizing the posterior density, Eq.(2.3), to find a single best estimate $f(x,w^*)$ of the function $y = f(x)$, while in the Bayesian approach one assigns a probability to all possible choices for $f(x,w)$.

2.4 Regression and Classification

In both approaches, regression (for example, curve fitting) and classification differ by the choice of targets y and the loss functions, L. For regression, the targets are continuous functions of the inputs and the loss function most commonly used is the quadratic loss, given in Eq.(2.6). This choice leads to the following empirical risk

$$R(w) = \frac{1}{N} \sum_{i=1}^{N} [y_i - f(x_i,w)]^2, \tag{2.10}$$

or, equivalently, the likelihood function

$$p(y|x,w) = \exp(-NR(w)/2\sigma^2)/\sigma\sqrt{(2\pi)}. \tag{2.11}$$

For classification, a better choice for the likelihood function (and therefore the empirical risk) is [9]

$$p(y|x,w) = \prod_{i=1}^{N} f(x_i,w)^{\omega_i y_i} [1 - f(x_i,w)]^{\omega_i (1-y_i)}, \tag{2.12}$$

where $0 < f(x_i,w) < 1$ and the targets y_i take on the discrete values 0 and 1. For classification, $f(x,w)$ is interpreted as the probability that the feature vector x belongs to the class with target $y = 1$. In Eq.(2.12), we have introduced an optional weight ω_i for each feature vector x_i. In particle physics, Monte Carlo models of the data are typically weighted, event-by-event, to correct

for deficiencies in modeling. For building a classifier using such data, the likelihood function in Eq.(2.12) would be the most appropriate. The corresponding risk function for classification is

$$R(w) = \frac{1}{N}\sum_{i=1}^{N} \omega_i [y_i \ln f(x_i,w) + (1-y_i)\ln(1-f(x_i,w))]. \tag{2.13}$$

It is readily shown (see for example, Ref. [11]) that the optimal choice of the function $f(x,w)$ is the one that approximates the probability $p(1|x)$ that the feature vector x belongs to the class with label 1

$$p(1|x) = \frac{p(x|1)p(1)}{p(x|1)p(1)+p(x|0)p(0)}, \tag{2.14}$$

where $p(1)$ and $p(0)$ are the prior probabilities for the classes labeled by 1 and 0, respectively, and $p(x|1)$ and $p(x|0)$ are the corresponding probability densities of the feature vectors. In a signal/background discrimination problem, the signal class is typically labeled by 1 and the background class by 0 (or sometimes -1). In this case, $p(1)/p(0)$ would be the prior signal to background ratio.

Given the probability $p(1|x)$, we may now define the *Bayes classifier*:

Bayes classifier: if $p(1|x) > q$, accept x as belonging to the class labeled by 1,

where the quantity q is a threshold chosen by the analyst [1]. The Bayes classifier is optimal in the sense that it achieves the lowest misclassification rate. For example, if one's goal is to distinguish real electrons from fake ones and to do so with the fewest errors, then one should use the Bayes classifier. In practice, however, we usually do not know the prior probabilities $p(1)$ and $p(0)$; indeed, this is often what we are trying to measure. It would therefore appear that we cannot use the Bayes classifier. Happily, this is not so. For classification, it is sufficient to approximate the *discriminant*

$$D(x) = \frac{p(x|1)}{p(x|1)+p(x|0)}, \tag{2.15}$$

because $D(x)$ and $p(1|x)$ are related one-to-one as follows

$$p(1|x) = \frac{D(x)}{D(x)+[1-D(x)]/A}, \tag{2.16}$$

where $A = p(1)/p(0)$. Classification using $D(x)$, with a given threshold, is equivalent to classification using $p(1|x)$, albeit using a threshold that is unknown.

The function $p(1|x)$ is optimal in another sense [12]. If one weights an admixture of signal and background events by the weight function $W(x) = p(1|x)$, then the signal strength can be estimated with zero bias and the smallest possible variance, provided that the weight's dependence on x has been accurately modeled and the ratio A is equal to its true value. Since we do not know the true value, it may be possible to iterate: start with an estimate of A—perhaps based on a prediction—estimate the signal strength, and therefore A, and repeat the procedure until convergence is achieved. It would be interesting to see if this works.

I end this section, with a few points that should be borne in mind.

[1] Formally, this choice is arrived at by minimizing a suitable loss function.

- If your goal is to classify objects with the fewest errors, then the Bayes classifier is the optimal solution. However, if that is not your aim—perhaps, you wish to measure the Higgs mass with the smallest uncertainty—then the Bayes classifier is not necessarily optimal.

- But if classification is your goal and you have a classifier known to be close to the Bayes limit, that is, it is known to be close to $p(1|x)$—or equivalently, $D(x)$, then *any* other classifier, however sophisticated it may be, can at best be only marginally better than the one already in your possession.

- *All* classification methods, such as the ones in `TMVA` [6], are merely different numerical approximations of some function of the Bayes classifier.

3. Multivariate Methods: In Practice

In this section, I provide a brief review of two interesting recent developments in multivariate methods, with a focus on signal/background event discrimination: ensemble learning, in particular, boosting, and Bayesian neural networks. For the important task of signal/background event discrimination, many multivariate methods are available, many of which are implemented in the `TMVA` package. Here is an incomplete list.

- 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

I do not include genetic algorithms in this list because they are best viewed as one of several effective methods for minimizing complicated empirical risk functions [2]. In principle, a genetic algorithm can be used to minimize any objective function. An excellent recent example of their effectiveness is the work of the NNPDF Collaboration [5] mentioned in the introduction.

[2] In genetic algorithms they are called fitness functions.

3.1 Ensemble Learning

One of the most interesting recent developments in machine learning is the realization by Freund and Schapire [13] that powerful classifiers can be built by averaging over an ensemble of *weak* classifiers, that is, classifiers that perform only marginally better than random guessing. Their specific (extremely successful) algorithm, called `AdaBoost`—to which we shall return shortly, has been embedded in a general framework, called *ensemble learning*, developed by Friedman and Popescu [15]. In this framework, a classifier with better performance than any individual within the ensemble can be represented by the weighted average

$$f(x) = \sum_{i=1}^{K} a_k f(x, w_k). \tag{3.1}$$

Ensemble methods have been developed most extensively using decision trees. A decision tree (see, for example, Ref. [14, 2] for recent applications in particle physics) can be visualized as an n-dimensional histogram in the space of feature vectors x. In a binary classification problem, each bin would be associated with some value f, typically, -1 if the majority of points in a bin are of one class and 1 if they are mostly of the other class. The classification of an object is determined by the bin in which it falls. The clever aspect of a decision tree is the manner in which the n-dimensional histogram is constructed: it is done by recursive binary partitioning of the feature space. At each step, and for every bin, one determines along which axis and where on that axis a bin should be split. Typically, each bin is split in such a way as to yield sub-bins with purity greater than parent bin. The partitioning of bins ends when some well-defined criterion has been reached, such as a minimum number of entries per bin. A virtue of decision trees is that the path to each bin can be represented as a sequence of `if then else` statements. Consequently, one knows what sequence of decisions yielded a given classification. Another practical advantage is that decision trees can be built rapidly. Therefore, they are ideally suited to ensemble methods, which can require the construction of a large number of classifiers.

The three most popular ensemble methods, *bagging* [16], *random forest* [17], and *boosting* [13] differ by their choice of weak classifiers, $f(x, w_k)$, and their choice of weights a_k:

- **Bagging**: (**B**ootstrap **agg**regat**ing**) this is a simple average over trees, with each tree $f(x, w_k)$ trained on a different bootstrap sample [3] drawn from the training sample;
- **Random Forest**: this is bagging, in which each tree is randomized in some way, for example, by selecting a random subset of features at each binary split within the tree, and
- **Boosting**: this is a weighted average over trees, each trained on a different weighting of the full training sample.

These methods are not necessarily tied to decision trees. Bagging can be applied to any classifier, while random forests can be applied to classifiers for which some randomization in their construction can be introduced. Boosting can be applied to any classifier that can make use of event-by-event weights.

[3] A bootstrap sample is one drawn with replacement.

3.1.1 AdaBoost

AdaBoost [13] (**Ada**ptive **Boost**ing) is one of the most successful, and still not fully understood, ensemble methods. The algorithm, in its original form, is based on decision trees $f(x,w)$ with output values ± 1 and targets $y = \pm 1$. Consequently, the product $f(x,w)y$ is positive for a correct classification and negative for an incorrect one. With this in mind, the algorithm proceeds as follows.

repeat K **times:**

1. create a decision tree $f_k = f(x,w_k)$
2. compute its error rate ε_k on the training sample
3. compute $a_k = \ln[(1-\varepsilon_k)/\varepsilon_k]$
4. scale the weight of the n^{th} training event by the factor

$$\exp(-a_k f(x_n,w_k)y_n/2)/\sum_{i=1}^{N} \omega_i \exp(-a_k f(x_i,w_k)y_i/2),$$

that is, increase the weight of incorrectly classified events relative to those that are correctly classified.

This rather puzzling algorithm is remarkable. Figure 1 shows the results of this algorithm applied to the separation of mSUGRA events at the focus point [18] from $t\bar{t}$ events, at LHC energies. When the boosted classifier is applied to the training sample, the error rate is seen to go to zero exponentially as the number of trees increases; that is, with a sufficient number of trees every event would be classified *perfectly*! Such behavior is usually a sure sign that a classifier has been fit too tightly to the training data, that is, that the classifier has been over-trained. It is therefore remarkable that when applied to an *independent* testing sample, the error rate of the AdaBoost classifier remains essentially constant showing that the AdaBoost algorithm is highly resistant to over-training. This feature of AdaBoost classifiers has been observed repeatedly, but is yet to be fully understood (but see, for example, Ref. [19]).

It might be argued that the resistance to over-training is irrelevant because nothing is to be gained using more trees than are strictly necessary for classification. Not so if one desires a smooth approximation to the optimal classifier $f(x)$. As noted above, a decision tree is an n-dimensional histogram; consequently, it provides only a piece-wise constant approximation to the desired classifier $f(x)$. However, if one averages over many such histograms, each with a differing set of bins, then one will achieve a much smoother approximation to $f(x)$. Therefore, averaging over a larger number of trees than is strictly necessary may be still be useful, if only to achieve a sufficiently smooth approximation.

3.2 Bayesian Neural Networks

Bayesian neural networks (BNN) [9], which were introduced into particle physics recently [20], have been successfully deployed, for the first time, in the search for single top quark production [1, 2]. In general, a BNN is simply the predictive distribution, Eq.(2.4), in which the function

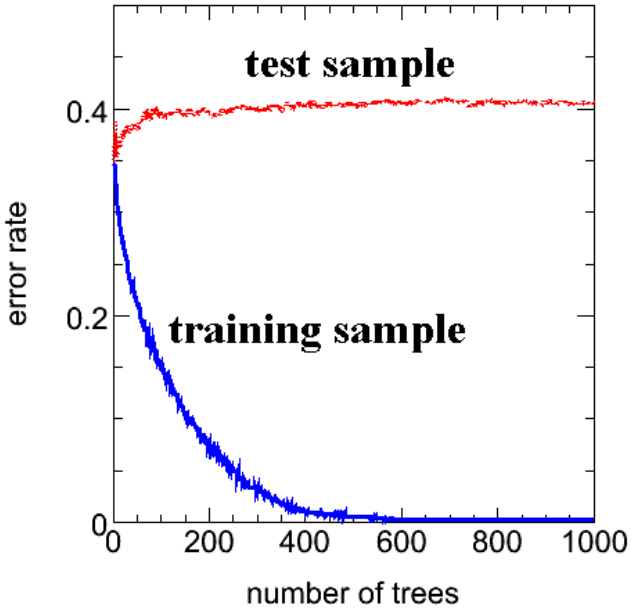

Figure 1: This shows the event classification error rate as a function of the number of decision trees over which the boosted classifier has been averaged. It is striking that the error rate on the training sample goes exponentially to zero, while the error rate on an independent testing sample remains essentially constant.

class is the class of feedforward neural networks with a fixed structure. However, as used by particle physicists [20], a BNN designed for classification is the *mean* of the predictive distribution, that is, it is the function

$$\begin{aligned} y(x) &= \int z\, p(z|x,T)dz, \\ &= \int f(x,w)p(w|T)dw. \end{aligned} \tag{3.2}$$

The second line follows from the form of the likelihood function $p(y|x,w) = f(x,w)^y\,[1-f(x,w)]^{1-y}$ and the use of a binary-valued target $y \in \{0,1\}$. The function class used in Refs. [1] and [2], available by default in the `Flexible Bayesian Modeling` (`FBM`) software by Neal [9], is

$$f(x,w) = \frac{1}{1+\exp[-g(x,w)]}, \tag{3.3}$$

where

$$g(x,w) = b + \sum_{j=1}^{H} v_j \tanh(a_j + \sum_{i=1}^{n} u_{ji}x_i). \tag{3.4}$$

n is the dimensionality of the feature vectors, that is, the inputs, and H is the number of hidden nodes. In neural network-based applications, the parameters $w = (b,v,a,u)$ are generally referred to as weights.

For realistic applications, the dimensionality of the parameter space of the functions $f(x,w)$ is typically in the hundreds. Therefore, about the only feasible way to approximate the integral

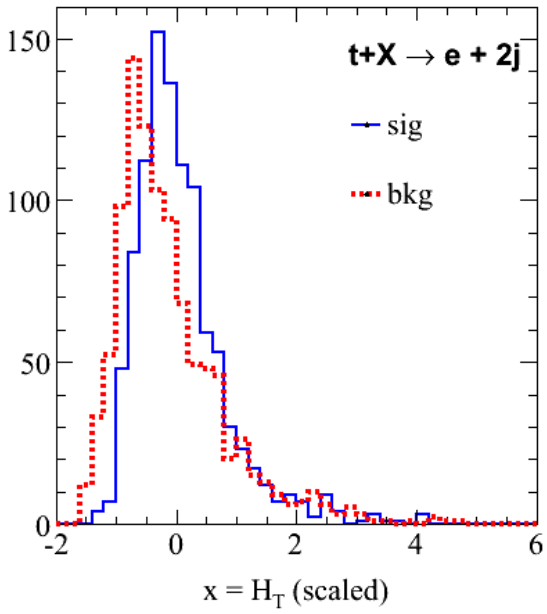

Figure 2: Distributions of the sum of transverse momenta (H_T) for single top quark (signal) and background events in the 2-jet final state (see Ref. [2]). The sum of the signal and background distributions has been scaled to have zero mean and unit variance and each distribution is normalized to the same area.

in Eq.(3.2) is by a Markov Chain Monte Carlo (MCMC) method, such as that implemented in the `FBM` software [9]. In the `FBM` software, a MCMC method is used to generate a sample of points, $w_1, w_2, \cdots, w_M$, from the posterior density $p(w|T)$. The integral, Eq.(3.2), is then approximated by the average

$$y(x) \approx \frac{1}{M} \sum_{m=1}^{M} f(x, w_m), \tag{3.5}$$

where M is the number of points sampled from $p(w|T)$. In practice, since the correlation between adjacent MCMC points is very high, one does not use all sampled points but rather a sparse subset thereof for which the point to point correlation is much lower than that of the full set of points. The average in Eq.(3.5) is over the sparse set of points.

We have skipped over a rather important detail. In order to arrive at the posterior density it is necessary to specify a prior density $\pi(w)$ over the parameter space. To do so, in general, is a very difficult problem. However, in practice one finds that a prior given by the product of Gaussians, one for each parameter and centered at zero, yields very good results provided that the widths of the Gaussians are appropriately chosen. The `FBM` package provides a mechanism for doing so.

The construction of BNNs is best illustrated by a simple 1-dimensional example. Figure 2 shows distributions of the sum of transverse momenta (H_T) for single top quark (signal) and background events with exactly 2 jets in the final state [2]. The sum of the distributions has been scaled to have zero mean and unit variance and each distribution is normalized to the same area. A BNN, based on neural networks with a single hidden layer of 20 nodes, was trained on a sample comprising 1000 signal plus 1000 background events. Five thousand points were sampled by MCMC

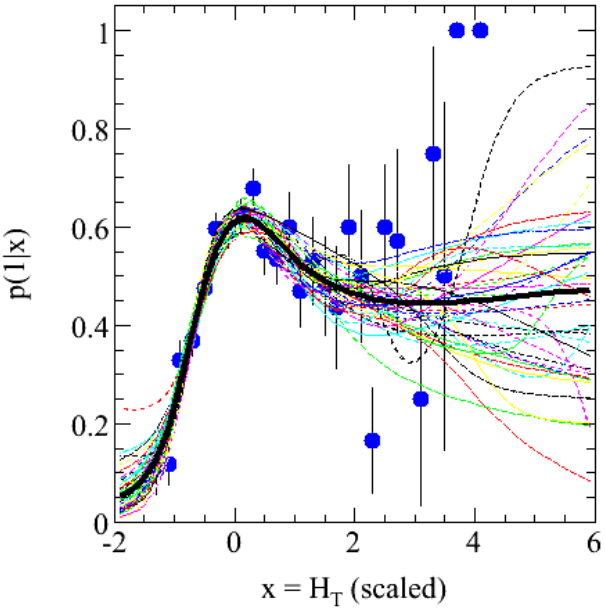

Figure 3: Each curve is a plot of $f(x,w_k)$, where k indexes one MCMC parameter point, while the thick curve is their average as a function of x. The points are calculated from the bin-by-bin ratio of $H(x|1)/[H(x|1)+H(x|0)]$, where $H(x|1)$ and $H(x|0)$ are the signal and background histograms, respectively. This ratio provides a direct approximation of the discriminant $D(x) = p(x|1)/[p(x|1)+p(x|0)]$. By construction, so does each function $f(x,w_k)$. It is evident, however, that their average provides a better approximation to the discriminant $D(x)$ than any of the individual functions.

from the posterior density, $p(w|T)$. (Each point was obtained as the last point in a sequence of 100 deterministic steps through the 61-dimensional parameter space followed by a stochastic step at the end.) To reduce the correlation between the points used to construct the BNN, a sparse subset of 250 points, from the 5000, was selected of which the last $M = 50$ points were used in the average, Eq.(3.5). In Fig. 3 is plotted the 50 functions $f(x,w_k)$, corresponding to the 50 points w_k. Since the training sample contains equal numbers of signal and background events, each function $f(x,w_k)$ approximates the discriminant $D(x) = p(x|1)/[p(x|1)+p(x|0)]$. Moreover, because this is a 1-dimensional problem, the discriminant can be approximated directly by computing the ratio $H(x|1)/[H(x|1)+H(x|0)]$, bin-by-bin, where $H(x|1)$ and $H(x|0)$ are the signal and background histograms, respectively, of the variable x. The large scatter at large values of x is due to the low counts in the tails of the distributions. The ensemble of functions, depicted in Fig. 2, constitute the predictive distribution, Eq.(2.4). The mean of the predictive distribution as a function of x, that is, the BNN as it is defined in particle physics, provides a good overall estimate of $D(x)$. Moreover, the predictive distribution can be used not only to estimate the discriminant $D(x)$, but also to estimate how well it has been estimated.

4. Outstanding Issues

Multivariate methods can be the basis of powerful analyses. They have been shown to be indispensable at the Tevatron and this is likely to be true also at the LHC. But, like all powerful tools, multivariate methods must be used with care. Below is a short list of what I consider to be the most pressing issues for which progress is sorely needed.

1. **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*?
   - How can one automate re-weighting of model data, event-by-event, in order to improve the match between real data and the model?
   - How can one verify that a classifier function is close to the Bayes limit?

2. **Looking Beyond the Lamppost**

   - Is there a robust and useful way to quantify the information content of a sample of n-dimensional signal and background data so that when it is compressed, say to 1-dimension, one is able to assess how much information has been lost?
   - Is there a sensible way to use multivariate methods when one does not know for certain where to look for signals? In particular, is there a useful way to compress data even if one does not know for certain what the signal should look like?

5. Summary

In this paper, we sought to shine a bit of light into methods that are often viewed as black boxes. However, with perhaps the exception of boosting, I contend that multivariate methods are only as mysterious as one wishes them to be. Moreover, all methods that deal with regression and classification are governed by the same underlying mathematics. The fact that one method may use a genetic algorithm to minimize an objective function while another uses back-propagation does not negate this point. One should distinguish between the formal mathematical underpinnings of these methods, which is essentially identical for all, from their necessarily approximate numerical implementations of which new methods are devised almost daily. Furthermore, the distinction between machine learning and Bayesian learning is more one of emphasis: machine learning emphasizes the best fit while Bayesian learning emphasizes averaging. However, in our survey of a couple of recent developments, we have seen that ensemble averaging is now a feature of both the machine learning and Bayesian learning approaches.

Multivariate methods can be applied to many aspects of data analysis. Today, with the advent of tools such as `TMVA`, they can be used by anyone with modest software skills. As emphasized in this paper, for a given loss function, all methods approximate the same mathematical entities. But no one method is guaranteed to be the best in all circumstances. Therefore, it is advisable

to experiment with a few of them. Several issues remain, the most pressing of which is the need for sound methods, and convenient tools, to explore and quantify the quality of modeling of n-dimensional data.

Acknowledgments

I wish to thank the organizers for inviting me to what proved to be a thoroughly enjoyable, and stimulating, conference.

References

[1] DØ Collaboration (V.M. Abazov et al.), *Evidence for production of single top quarks and first direct measurement of $|Vtb|$*, *Phys. Rev. Lett.***98** 181802 (2007) [hep-ex/0612052].

[2] DØ Collaboration (V.M. Abazov et al.), *Evidence for production of single top quarks*, *Phys. Rev. D***78** 012005 (2008) [arXiv:0803.0739, hep-ex].

[3] CDF Collaboration (T. Aaltonen et al.), *Measurement of the Single Top Quark Cross Section at CDF*, *Phys. Rev. Lett.* **101** 252001 (2008) [arXiv:0809.2581, hep-ex].

[4] ATLAS Collaboration (D. Lelas for the collaboration), *Jet reconstruction with first data in ATLAS*, *ATL-PHYS-PROC-2008-082, ATL-COM-PHYS-2008-246* (2008).

[5] NNPDF Collaboration (R.D. Ball et al.), *A determination of parton distributions with faithful uncertainty estimation*, *Nucl. Phys. B* **809** 1 (2009) [arXiv:0808.1231, hep-ph].

[6] J. Stelzer, *TMVA - Toolkit for Multivariate Analysis*, these proceedings.

[7] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 2nd Edition, 2000.

[8] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer-Verlag, New York, 2nd Edition, 2009.

[9] R.M. Neal, *Bayesian Learning of Neural Networks*, Springer-Verlag, New York, 1996.

[10] A. O'Hagan, *Kendall's Advanced Theory of Statistics: Volume 2B, Bayesian Inference*, Oxford University Press, New York, 2002.

[11] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2007.

[12] R.J. Barlow, *Event Classification Using Weighting Methods*, *J. Comput. Phys.* **72** 1 (1987).

[13] Y. Freund and R.E. Schapire, *A decision-theoretic generalization of on-line learning and and application to boosting*, *J. Comput. Sys. Sci.* **55**(1) 119 (1997).

[14] B. Roe et al., *Boosted decision trees, an alternative to artificial neural networks*, *Nucl. Instrum. Meth. A* **543** 577 (2005).

[15] J.H. Friedman and B.E. Popescu, *Predictive learning via rule ensembles*, *Ann. Appl. Stat.* **2**(3) 916 (2008); http://www-stat.stanford.edu/∼jhf/ftp/RuleFit.pdf.

[16] L. Breiman, *Bagging Predictors*, *Machine Learning*, **26** 123 (1996).

[17] L. Breiman, *Random Forests*, *Machine Learning*, **45** 5 (2001).

[18] H. Baer et al., *Model independent approach to focus point supersymmetry: From dark matter to collider searches*, *JHEP*, **510** 20 (2005).

[19] J.H. Friedman, T. Hastie and R. Tibshirani, *Additive logistic regression: a statistical view of boosting*, *Ann. Stat.* **28**(2) 377 (2000).

[20] P.C. Bhat and H.B. Prosper, *Bayesian neural networks*, in *Statistical Problems in Particles, Astrophysics and Cosmology*, Imperial College Press, Editors L. Lyons and M. Ünel, 2005.