IN2P3
Institut National de Physique Nucléaire
et de Physique des Particules

SOS '08

1^{ere} édition de l'Ecole de Statistique pour la physique

School of Statistics 2008

IPHC Strasbourg, 30 Juin - 04 Juillet 2008

Thématiques traitées :

Probabilité et incertitude	Imad LAKTINEH
Statistiques et tests d'hypothèses	Pierre LUTZ
Les analyses discriminantes multivariées	Harrison PROSPER
Les réseaux de neurones	Jerome SCHWINDLING
Les arbres de décision	Yann COADOU
L'utilisation de TMVA	Andreas HOECKER
L'utilisation de RooFit	Wouter VERKERKE
Combinaison des observables en physique des saveurs	Jerome CHARLES
Combinaison des observables en cosmologie	Jean-Marc VIREY

Comité organisateur et éditeur :

Jerome Baudot
Gautier Hamel de Monchenault
Olivier Leroy
Corinne Bérat
Imad Laktineh
Arnaud Lucotte

Table des matières

I Les analyses discriminantes Multivariées **5**

Harrison B. PROSPER : *Multivariate discriminants* **7**

- 1.1 Introduction 7
- 1.2 Part 1 7
  - 1.2.1 Optimal Discrimination 7
  - 1.2.2 Machine Learning 8
  - 1.2.3 Bayesian Learning 9
  - 1.2.4 Machine Learning is Bayesian Learning in Disguise 10
  - 1.2.5 Optimal Classification 10
  - 1.2.6 Optimal Signal Extraction 12
- 1.3 Part 2 13
  - 1.3.1 Random Grid Search 14
  - 1.3.2 Quadratic and Linear Discriminants 15
  - 1.3.3 Support Vector Machines 16
  - 1.3.4 Naïve Bayes 18
  - 1.3.5 Kernel Density Estimation 18
  - 1.3.6 Ensemble Learning 21
  - 1.3.7 Bayesian Neural Networks 23
- 1.4 Outstanding Issues 24
- 1.5 Conclusion 25

Première partie

Méthodes et techniques utilisées dans le cadre des analyses discriminantes multivariées

Multivariate discriminants

Harrison B. PROSPER
Florida State University, Floride, USA

1.1 Introduction

The classification of objects and the closely related task of signal enhancement relative to background are extremely important in data analysis. In this paper, we explain how both tasks — referred to generically as discrimination — can be achieved in principle and in practice. Since most interesting data are multidimensional, the most effective discrimination methods are those that make use of the dependencies between the variables. Each discrimination method results in an approximation to a function, $f(x)$, called a **multivariate discriminant** whose domain is the n-dimensional space of discriminating variables x and whose range is either the interval [0,1] or the interval [-1, 1]. In order to focus on the essential ideas of object discrimination, and avoid complications that serve no useful pedagogical purpose, we restrict the discussion to the important special case of object classification into two classes, that is, we consider **binary classification**. In a binary classification problem, an object is assigned to one class if $f(x) < q$ and to the other class if $f(x) \geq q$, where q is some threshold chosen by the analyst.

The paper is organized as follows. In Part 1, we describe the salient features of discrimination theory. In Part 2, we describe a few powerful, yet practical, methods for implementing the theory. We end with a discussion of outstanding issues and a few concluding remarks.

1.2 Part 1

1.2.1 Optimal Discrimination

A basic task in data analysis is to separate objects into one or more classes. Each object is characterized by an n-tuple of variables $x = (x_1, \cdots, x_n)$, referred to variously as **discriminating variables** or **feature variables**. Sometimes this separation is done with the goal of treating objects in a given class as if they are all of that class. For example, in high energy physics a common task is to identify electrons. In observational cosmology, the task may be to distinguish Type Ia supernovae from other supernovae. In both cases, the correct identity of the objects — electrons on the one hand and Type Ia supernovae on the other is the goal. Sometimes, however, the goal is simply to select, preferentially, objects of a certain type, for example, signal objects. In this section, we answer the following question : how should these objects be discriminated ? A straightforward answer is : apply a threshold, that is, a cut, to each discriminating variable ; should an object pass all the cuts, we declare it to be of the appropriate class. This answer however is incomplete. It is incomplete because it does not give rules for choosing the cuts, nor for determining whether the discrimination is optimal.

In statistical analyses, these rules are represented formally by **loss functions**, L. A loss function measures the loss incurred by making a poor choice, here the choice (or decision) about where to place cuts on the discriminating variables. Ideally, the loss function encodes as accurately as possible the goal of the analysis. For example, in a Higgs analysis, the goal may be to measure the Higgs mass as accurately as possible. Or perhaps the goal is to measure the Higgs production cross section with the smallest possible relative uncertainty. Each of these goals corresponds to a different loss function and therefore a different optimization problem. Consequently, cuts that are best for measuring the Higgs mass need not be the same as those that are best for the measuring the cross section. Moreover, even after having arrived at the *best* set of cuts on the discriminating variables, it is generally found that optimal discrimination has not been achieved because the absolute minimum of the loss function has not been reached. Therefore, when a colleague declares she has optimized the cuts, what she really means is that she has found the "best cuts." The "best cuts" may, or may not, be optimal. In general, in order to achieve optimal discrimination it is necessary to combine the discriminating variables into a single multivariate discriminant $f(x)$, where x denotes the n-tuple of discriminating variables and f is some appropriately constructed function.

There are two broad approaches to constructing such functions : **machine learning** [1, 2] and **Bayesian learning** [3], each with its own methods, jargon and theory. Superficially, the two approaches are very different. But, upon close inspection, they are seen to be merely different ways of expressing the same basic ideas. In both approaches, it is assumed that a function $y = f(x)$ exists that would achieve optimal object discrimination in the sense of minimizing the given loss function. In general, the function f is not known. The best that can be done is to find a good approximation to it. However, even though the detailed form of the multivariate discriminant is unknown, as we shall see shortly, for a specific choice of loss function it is possible to write an explicit formula for $f(x)$ in terms of the probability densities of the discriminating variables.

1.2.2 Machine Learning

In the machine learning approach [1, 2], the multivariate discriminant $y = f(x)$ is found by the direct or indirect minimization of a loss function. The basic elements of this approach are :

- a class of functions $\mathbb{F} = \{f(x, w)\}$, where w are parameters to be found by the minimization procedure ;
- a constraint $Q(w)$ on the class of functions, $\mathbb{F}$, and
- a loss function, $L(y, f)$, that quantifies the loss incurred if the function $f(x, w)$ is poorly chosen from the function class $\mathbb{F}$, that is, if $f(x, w)$ is far from the desired function $y = f(x)$.

The desired value, y, of the function $f(x)$, to be approximated by $f(x, w)$, is called the **target**.

In spite of what we have just stated, it turns out that minimizing the loss function $L(y, f)$ directly is not a good strategy ! The reason is because the function $f(x, w)$ would depend on the specific set of data used to find it. Minimizing the loss function on the observed data would be as ill-advised as tuning cuts using the same. It is generally agreed that the many decisions that must be made in any non-trivial analysis be as independent as possible of the data used to obtain the final results ; that is, there should be some degree of "blindness" with respect to the final results. Moreover, it is generally agreed that final results ought to be *robust* with respect to changes in the data used : a small change in the data should yield a correspondingly small change in the results.

In view of these comments, it is much more satisfactory to minimize not the loss function itself but rather its average with respect to all possible realizations of the data. These realizations are called **training data**, T, and are either the results of Monte Carlo simulations (and) or real data that are independent of those used to obtain the final results. The average of the loss function $L(y, f(x, w))$ defines a functional[1] called the **empirical risk** function $R(w)$

$$R(w) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i, w)), \tag{1.1}$$

where N is the sample size of the training data, T. Instead of finding $f(x, w)$ by minimizing the loss function we find it by minimizing the empirical risk, $R(w)$, subject to the constraint $Q(w)$, that is, by minimizing the **cost function**

$$C(w) = R(w) + \lambda Q(w), \tag{1.2}$$

where λ is a tunable parameter that determines how strongly the constraint is imposed. A constrained χ^2 fit of a function to data is a well-known cost function in data analysis. In high energy physics, it is used in everything from curve-fitting to the extraction of parton distribution functions using disparate data sets from multiple experiments. At the minimum of the cost function $C(w)$, a function $f(x, w_*)$ will be selected from the function class, which, in the limit $N \to \infty$, converges to the function $f(x, w_0)$ that would have been selected were it possible to minimize the *true* risk function

---

[1] A functional of a function $f(x)$ is a function that depends on *all* values of $f(x)$ simultaneously.

$$R(w) = \int L(y, f(x, w))\, p(y, x)\, dydx, \tag{1.3}$$

where $p(y, x)$ is the joint probability density of y and x. The empirical risk, given in Eq. (1.1), is a *consistent*[2] estimate of the risk function, Eq. (1.3), in that the empirical risk converges to the true risk as the training sample grows to infinity [1].

1.2.3 Bayesian Learning

In the Bayesian approach, every statistical problem is regarded as a problem of inference [4]. In the present context, the problem is to *infer* the function $y = f(x)$ [3]. The Bayesian approach requires specification of the following ingredients :

- a class of parameterized functions $\mathbb{F} = \{f(x, w)\}$;
- a prior density $\pi(f)$ over the space of functions and
- a likelihood function, $p(y|x, w)$, that is proportional to the probability that the value y is associated with the n-dimensional discriminating variable x.

The prior density $\pi(f)$ assigns a number to every function f that quantifies the relative probability of each function, *a priori*. It is technically challenging to construct priors over function spaces. Therefore, in practice, the prior density, $\pi(w)$, is defined over the space $\{w\}$ of parameters instead. Unfortunately, that too is a difficult task. But, given the prior density $\pi(w)$ and the likelihood function $p(y|x, w)$, Bayes' theorem

$$\begin{aligned} p(w|T) &= \frac{p(T|w)\pi(w)}{p(T)}, \\ &= \frac{p(y, x|w)\pi(w)}{p(y, x)}, \\ &= \frac{p(y|x, w)p(x|w)\pi(w)}{p(y|x)p(x)}, \\ &\sim p(y|x, w)\pi(w), \end{aligned} \tag{1.4}$$

can be used to assign a probability density $p(w|T)$ to every point w in the parameter space of the class of functions $\{f(x, w)\}$. Since each point w corresponds to a function $f(x, w)$, $p(w|T)$ is the probability density assigned to it. Suppose $p(w_1|T) > p(w_2|T)$ for two points w_1 and w_2, respectively, then the associated function $f(x, w_1)$ is more compatible with the training data T than is the function $f(x, w_2)$. In Eq. (1.4), we have made the reasonable assumption that the probability density of the n-dimensional discriminating variable x is independent of the parameters w, that is, that $p(x|w) = p(x)$.

The **posterior density** $p(w|T)$ is the final result of the Bayesian inference. But, it is the starting point for the calculation of many useful quantities. One such quantity is the **predictive distribution** [4],

$$p(y|x, T) = \int p(y|x, w)p(w|T)dw, \tag{1.5}$$

which can be used to predict plausible values for the unknown function $f(x)$, given the n-dimensional discriminating variable x and the training data T on which the predictions are based. In other words, for a given value of x, the predictive distribution gives a *distribution* of possible values of y and assigns a probability $p(y|x, T)dy$ to each. It is often useful, however, to have a single estimate of the true mapping $y = f(x)$, rather than a distribution of possible mappings. In principle, an estimate, $\hat{f}$, of f can be found in the usual way by minimizing a suitable risk function, for example,

$$f(x) \approx \hat{f}(x) = \arg_{\hat{f}} \min \int L(y, \hat{f})p(y|x, T)dy, \tag{1.6}$$

---

[2] A consistent estimate $\hat{X}$ of a quantity X is one that converges to the true value of the quantity as the data sample size goes to infinity.

with respect to $\hat{f}$, where $L(y, f)$ is the loss function. A common choice for the latter is the **quadratic loss**

$$L(a, b) = (a - b)^2, \tag{1.7}$$

which yields the following estimate

$$\hat{f}(x) = \int y p(y|x, T) dy, \tag{1.8}$$

of $y = f(x)$. Thus for a quadratic loss function, the optimal estimate of $y = f(x)$ is the mean of the predictive distribution, Eq. (1.8). Other loss functions, in general, will yield other estimates.

1.2.4 Machine Learning is Bayesian Learning in Disguise

We claimed above that the machine and Bayesian learning approaches are merely different ways to phrase the same mathematical problem. This is true provided one is willing to make the following identifications

$$\begin{aligned} R(w) &\sim \ln p(y|x, w), \\ &= \sum_{i=1}^{N} \ln p(y_i|x_i, w), \qquad (1.9) \\ \lambda Q(w) &\sim \ln \pi(w). \qquad (1.10) \end{aligned}$$

If this is accepted, then the machine learning approach provides a **maximum a posteriori** (MAP) estimate of the function $y = f(x)$. In other words, minimizing the cost function Eq.(1.2) is equivalent to maximizing the posterior density, Eq.(1.4). For a unimodal posterior density, this yields a single best estimate $f(x, w_*)$ of the function $y = f(x)$. But, in the fully Bayesian approach one averages, in effect, over all plausible choices for $f(x, w)$ weighted by the posterior density $p(w|T)$. Interestingly, one of the most promising recent advances in machine learning, ensemble learning — discussed in Part 2, embraces the idea of averaging over functions. In that sense, both approaches have become even more closely related [5, 6, 7].

1.2.5 Optimal Classification

Regression (for example, curve-fitting) and classification differ by the choice of targets y and loss function, L. In regression, the targets are continuous functions of x and the loss function most commonly used is the quadratic loss, given in Eq.(1.7). The corresponding empirical risk is

$$R(w) = \frac{1}{N} \sum_{i=1}^{N} [y_i - f(x_i, w)]^2, \tag{1.11}$$

which is equivalent to the likelihood function

$$p(y|x, w) = \exp(-NR(w)/2\sigma^2)/\sigma\sqrt{2\pi}. \tag{1.12}$$

The typical goal of classification is *to classify objects while making the fewest possible mistakes.* Suppose we wish to identify electrons or Type Ia supernovae. Surely, we would wish to make as few classification mistakes as possible. If so, it is possible to write a cost function that captures this goal explicitly. For concreteness, consider the classification of objects into one of two classes called *signal* S and *background* B. Although our discussion will be in terms of classification into a signal or a background class, the discussion would proceed unchanged for *any* binary classification problem ; for example, classification into electron and non-electron classes, or Type Ia and non-Type Ia classes.

For simplicity, let us suppose that each object to be classified is characterized by a 1-dimensional discriminating variable x. The generalization to n-dimensions is in principle straightforward and yields the same result as we are about to derive. The probability density of x for the signal and background classes are $p(x, S)$ and $p(x, B)$, respectively, as depicted in Fig. 1.1. The classification of objects into the disjoint

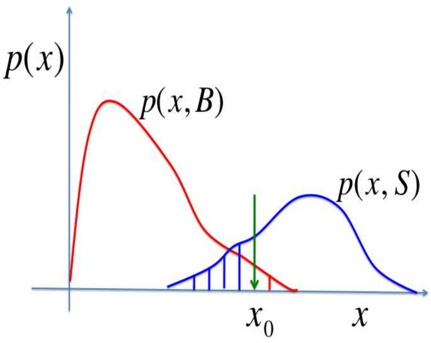

Figure 1.1 – The curve on the left depicts the density $p(x,B) = p(x|B)p(B)$, while the curve on the right depicts the density $p(x,S) = p(x|S)p(S)$. The ratio $k = p(S)/p(B)$ is the prior signal to background ratio and $p(x|S)$ and $p(x|B)$ are the conditional densities of x for the signal and background classes, respectively. By definition, the distribution of x is $p(x) = p(x,S) + p(x,B)$.

classes S and B is defined by the **decision boundary** $x = x_0$. Objects for which $x \geq x_0$ are classified as signal, while those with $x < x_0$ are presumed to be background. Figure 1.1 depicts the practical situation in which classification errors are unavoidable due to the overlap of the densities pertaining to objects of the classes S and B. However, it is possible to minimize the probability of such errors as follows. Let C_S be the cost of misclassifying signal as background and C_B the cost of misclassifying background as signal. We can write the cost function, C, for misclassification as

$$\begin{aligned} C &= C_S \int H(x_0 - x)\, p(x,S)\, dx, \\ &+ C_B \int H(x - x_0)\, p(x,B)\, dx, \end{aligned} \tag{1.13}$$

where $H(x)$ is the Heaviside (step) function : $H(x) = 1$ if $x > 0$ and 0 otherwise. The quantity C_S can be interpreted as the cost of signal loss, while C_B is the cost of background contamination. By definition, the cost function is minimal at the optimal choice of the decision boundary $x = x_0$. That boundary defines a discriminant function $f(x)$, even though one does not appear explicitly in Eq (1.13). Minimizing the cost function C with respect to the boundary $x = x_0$

$$\begin{aligned} 0 &= C_S \int \delta(x_0 - x)\, p(x,S)\, dx, \\ &- C_B \int \delta(x - x_0)\, p(x,B)\, dx, \end{aligned} \tag{1.14}$$

yields a function $BD(x)$ called the **Bayes discriminant**,

$$BD(x_0) = \frac{C_B}{C_S} = \frac{p(x_0,S)}{p(x_0,B)} = \frac{p(x_0|S)\, p(S)}{p(x_0|B)\, p(B)}. \tag{1.15}$$

As alluded to, the same result holds when x is multidimensional. In general, the Bayes discriminant can be written as

$$BD(x) = B(x)\frac{p(S)}{p(B)} = \frac{p(S|x)}{p(B|x)}, \tag{1.16}$$

where $B(x)$, the **Bayes factor**, is defined by

$$B(x) = \frac{p(x|S)}{p(x|B)}. \tag{1.17}$$

The last step in Eq. (1.16) follows from Bayes' theorem $p(K|x) = p(x|K)p(K)/p(x)$, where $K = S$ or B. Note that as we vary the cost ratio C_B/C_S, so does the boundary $x = x_0$, therefore, the Bayes factor is best viewed as a function of x. Note also, that $p(S|x) + p(B|x) = 1$. The Bayes factor reduces to the well-known likelihood ratio when the class densities, $p(x|S)$ and $p(x|B)$, are independent of unknown parameters.

The function

$$\begin{aligned} p(S|x) &= p(x|S)p(S)/p(x), \\ &= p(x|S)p(S)/[p(x|S)p(S) + p(x|B)p(B)] \end{aligned} \tag{1.18}$$

is the probability that an object characterized by the n-dimensiomnal variable x is of the signal class S. This probability, or *any* one-to-one function of it, is precisely that needed to achieve classification with the fewest mistakes. The equation $q = p(S|x)$ — where $q = C_B/(C_S + C_B)$ is a constant determined by the relative cost of background contamination versus signal loss — defines a decision boundary (which, in general, is a curved surface) in the space of the discriminating variable x. The decision boundary partitions this space into a signal-rich region $p(S|x) \geq q$ and a background-rich region $p(S|x) < q$. The upshot of this discussion is that binary classification, with the fewest possible mistakes, entails constructing a particular family of surfaces in the space of discriminating variables. Any function $f(x, w)$ that approximates the conditional class probability $p(S|x)$ with negligible error is said to have reached the **Bayes limit**. *All* multivariate discrimination methods whose goal is classification with the fewest errors are mathematically equivalent in that each attempts to approximate the *same* function, namely, the probability $p(S|x)$, or some function (or functional) thereof. Consequently, if we have found a method that yields discrimination close to the Bayes limit we know *a priori* that *no* method exists that can discriminate significantly better, however sophisticated it might be.

In most applications, the ratio $k = p(S)/p(B)$ is not known. Indeed, a measurement of the signal fraction $\epsilon = p(S)$ may well be the object of the analysis. If so, we cannot compute $p(S|x)$ because it depends on k. Fortunately, this is not a problem because, as noted above, we can specify an appropriate threshold on *any* one-to-one function of $p(S|x)$. In particular, we can apply a cut to the discriminant function

$$D(x) = s(x)/[s(x) + b(x)], \tag{1.19}$$

where, to simplify the notation we have set $s(x) \equiv p(x|S)$ and $b(x) \equiv p(x|B)$. A cut on $D(x)$ is equivalent to a cut on

$$p(S|x) = D(x)/[D(x) + (1 - D(x))/k], \tag{1.20}$$

albeit one that is unknown. As we shall see in Part 2, this fact is used routinely in the practical construction of multivariate discriminants.

At the risk of belaboring the point, we stress again that the discrimimant $D(x)$ is optimal if one's goal is to minimize the number of misclassifications or, as discussed in the next section, enhance signal relative to background. However, if, for example, the goal is to measure the mass of a particle with the smallest possible uncertainty, a discriminant function other than $D(x)$ may be needed. Unfortunately, in this case it is generally not possible to find the general form for the multivariate discriminant $f(x)$, nor is the interpretation of $f(x)$ as clear-cut as it is for $D(x)$ and the related function $p(S|x)$.

1.2.6 Optimal Signal Extraction

The function $p(S|x)$ is optimal in another sense [8]. Suppose we introduced an object-by-object weight $w(x)$, where to be definite we suppose the objects to be particle collision events, then the prior signal fraction $\epsilon = p(S)$ can be estimated using event-by-event weighting. To see this, we begin by writing the probability density of the data $d(x)$, which is presumed to be an admixture of signal and background described by the densities $s(x)$ and $b(x)$, respectively, in the form

$$d(x) = \epsilon\, s(x) + (1 - \epsilon)\, b(x). \tag{1.21}$$

Event-by-event weighting is simply multiplication throughout by the weight function $w(x)$

$$w(x)\, d(x) = \epsilon\, w(x)\, s(x) + (1 - \epsilon)\, w(x)\, b(x). \tag{1.22}$$

Now compute the expectations

$$\bar{w}_d = \int dx\, w(x)\, d(x), \tag{1.23}$$

$$\bar{w}_s = \int dx\, w(x)\, s(x), \tag{1.24}$$

$$\bar{w}_b = \int dx\, w(x)\, b(x), \tag{1.25}$$

for the data, signal and background, respectively. The signal fraction ϵ and the variance associated with its estimate are given by

$$\epsilon = (\bar{w}_d - \bar{w}_b)/(\bar{w}_s - \bar{w}_b), \tag{1.26}$$

and

$$\text{Var}(\epsilon) = \frac{1}{N}\int dx[(w_d(x) - \bar{w}_b)/(\bar{w}_s - \bar{w}_b)]^2 d(x), \tag{1.27}$$

respectively, where N is the number of observed events. Barlow [8] showed that the variance is minimized if one chooses the weighting function as follows

$$\begin{aligned} w(x) &= p(S|x), \\ &= s(x)/[s(x) + b(x)/k], \\ &= D(x)/[D(x) + (1 - D(x))/k], \end{aligned} \tag{1.28}$$

and if k has the correct value, namely, $k = p(S)/p(B) = \epsilon/(1-\epsilon)$. Again, this seems not to be terribly useful because we do not know k. However, the weighting function $w(x)$ can be computed using a reasonable guess for k, say a prediction. The weighting procedure will produce a new estimate of ϵ and therefore of k. Using the updated value of k, we could repeat the weighting procedure and continue to do so until we arrive at a self-consistent estimate of ϵ.

1.3 Part 2

We have seen that in order to solve the binary classification problem with the fewest number of mistakes it is sufficient to compute the multivariate discriminant

$$D(x) = s(x)/[s(x) + b(x)], \tag{1.29}$$

where $s(x)$ and $b(x)$ are the signal and background densities, respectively. Moreover, a cut on $D(x)$ is equivalent to an (unknown) cut on the probability $p(S|x)$ that an object characterized by the n-dimensional variable x is of the class S. The class S could be, for example, the signal class in a signal/background discrimination problem, or the electron class in an electron/fake-electron discrimination problem. Many methods have been devised to approximate $D(x)$, but none can be said to be superior to all other methods for every conceivable classification problem. So beware of claims to the contrary. A sensible strategy is to try a few methods [9] and use the one which, for the particular problem at hand, provides the best discrimination. Of course, this presupposes that we have a way to assess the quality of the approximation, and hence the discrimination. We shall return to this issue later.

Multivariate discrimination is a huge and rapidly evolving field for which an exhaustive survey would require an entire book [10]. However, given the comments above about the goal of discrimination, it suffices to limit discussion to a few classic methods, followed by a description of some interesting recent developments. We shall not discuss neural networks in detail here because they are covered in the lectures by Jerome Schwindling [11], nor shall we venture too far into a discussion of decision trees because they are covered in detail in the lectures of Yann Coadou [12]. Moreover, to render our discussion more concrete we shall continue to use the example of discrimination between signal and background events in high energy physics. In the sections below, we cover the following broad class of methods :

- Random Grid Search (RGS)
- Quadratic and Linear Discriminants
- Support Vector Machines
- Naïve Bayes
- Kernel Density Estimation
- Bayesian Neural Networks

1.3.1 Random Grid Search

The simplest way to classify events is to apply a cut to each component of an n-dimensional discriminating variable x. For decades this has been (and still is) the standard technique in high energy physics. The n cuts comprise a **cut-point**. Geometrically, a cut-point is the apex of a surface built from planes, parallel to each axis, that intersect at right-angles. For each cut-point, we compute the cost of misclassification and we choose the one with the minimum cost. The simplest way to vary the cut-points is to move systematically through a regular grid of them. However, this procedure becomes rapidly untenable as the dimensionality n increases because the number of cut-points grows like K^n, where K is the number of cuts along each axis. We encounter the infamous "curse of dimensionality."

Happily, there is a very simple and an enormously more efficient way to choose the cut-points as illustrated in Fig. 1.2. The figure shows a 2-dimensional space with $x = (x_1, x_2)$ in which background events tend to cluster near the origin, while signal events tend to cluster away from the origin. Since the goal is to extract signal as efficiently as possible, the key idea is to place each cut-point at each signal point, as provided, for example, by a Monte Carlo simulation of the signal. In this case, the number of cut-points is independent of the dimensionality of the space of discriminating variables, being simply the number of signal Monte Carlo events that are available for this purpose. The planes through each cut-point still intersect at right-angles, but, since the cut-points are randomly distributed, these planes form a random grid ; hence the name of this method : the random grid search [13].

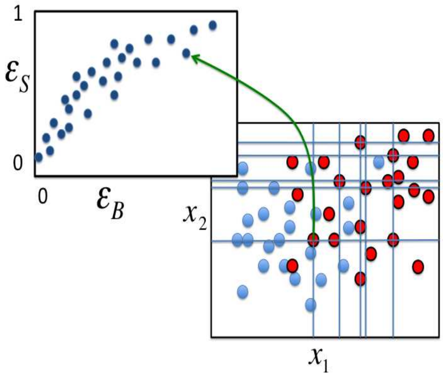

FIGURE 1.2 – A cut-point is placed at each signal point (black circles, filled-in red), as provided by a Monte Carlo simulation of signal events, and a cost of misclassification (or some other cost) is computed. The best cut-point is the one with the lowest cost. Two useful quantities to compute are the signal and background efficiencies ϵ_S = #(after)/#(before) and ϵ_B = #(after)/#(before), respectively, where "#(before)" and "#(after)" refer to the number of signal, or background, events before and after the cuts. These numbers can be plotted against each other, as indicated, and provide a visual way to compare the efficacy of different multivariate discriminants, independently of how each is constructed.

Figure 1.3 shows how this method fares compared with a Bayesian neural network (described below). In this example, the goal is to discriminate between 14 TeV proton-proton collision events generated using an mSUGRA model [14] (with parameters : $m_0 = 3280$ GeV, $m_{1/2} = 300$ GeV, $A_0 = 0$, $\tan\beta = 10$, $\text{sign}(\mu) = +1$, $m_{top} = 175$ GeV), in which the dominant reaction is gluino pair production, from the

Standard Model (SM) backgrounds (light-jet QCD, $t\bar{t}$, W+jets and Z+jets). The signal-to-background ratio $k = p(S)/p(B)$ after some basic cuts is predicted to be a daunting $k \sim 1/3000$. Application of the RGS algorithm to 5 kinematic variables (missing transverse energy, and the transverse momenta of the four jets with the highest transverse momenta) yields the signal and background counts shown in Fig. 1.3. Each point in the figure corresponds to a cut-point in the 5-dimensional space of discriminating variables. The dashed curve is the locus of the signal and backgrounds counts determined by varying cuts on the discriminant $D(x)$, computed using a Bayesian neural network. The figure illustrates a typical result : the random grid search can find cuts yielding good discrimination, but a properly constructed approximation to $D(x)$ always does at least as well as the cuts, and often much better.

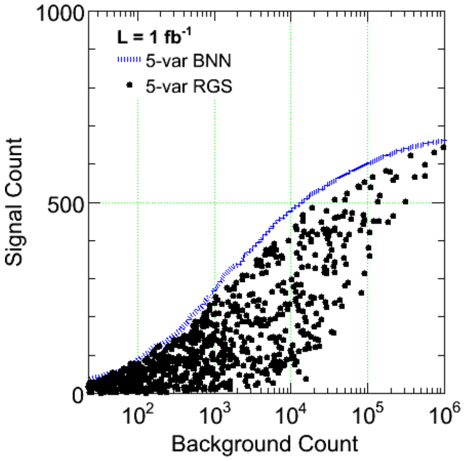

FIGURE 1.3 – A comparison of signal and background counts for varying cuts on a discriminant $D(x)$ computed using a Bayesian neural network with those obtained using the random grid search algorithm. The event counts are based on simulated proton-proton mSUGRA and SM collision events at 14 TeV for an integrated luminosity of 1 fb^{-1}. See text for more details.

1.3.2 Quadratic and Linear Discriminants

The random grid search is a considerable improvement over the regular grid. Moreover, in terms of the computational resources needed it is competitive with other cut-finding algorithms such as those based on genetic algorithms. However, as the example shown in Fig. 1.3 suggests, even the best cuts may fall short of optimal discrimination, while the vast majority will fall far short. As noted in Part 1, optimal discrimination requires the accurate calculation of the discriminant $D(x) = s(x)/[s(x) + b(x)]$, or a one-to-one function thereof, such as the Bayes factor $B(x) = s(x)/b(x)$.

One of the earliest examples of a successful multivariate discriminant is due to Fisher (see, for example, Ref. [10]). One way to motivate the **Fisher discriminant** is to embed it within the theory sketched in Part 1. Suppose we can approximate each density $s(x)$ and $b(x)$ as a multivariate Gaussian

$$\text{Gaussian}(x|\mu, \Sigma) = \exp[-(x - \mu)^T \Sigma^{-1} (x - \mu)/2]/(2\pi)^{n/2} \sqrt{|\Sigma|}, \tag{1.30}$$

where μ is a vector of means, Σ is the covariance matrix, and $|\Sigma|$ its determinant. Since, as noted, we may use any one-to-one function of $D(x)$ (or $B(x)$), we consider for convenience the logarithm of the Bayes factor $B(x)$. After dropping non-essential constants, it may be written as

$$\lambda(x) = \chi^2(\mu_B, \Sigma_B) - \chi^2(\mu_S, \Sigma_S), \tag{1.31}$$

where $\chi^2(\mu, \Sigma) = (x - \mu)^T \Sigma^{-1} (x - \mu)$ and where the subscripts S and B label the quantities pertaining to the signal and background, respectively. The quantity in Eq. (1.31) is called the **quadratic discriminant**. A fixed value of $\lambda(x)$ defines a hyper-paraboloidal decision boundary that divides the space into signal-rich and background-rich regions. A cut on $\lambda(x)$ provides optimal discrimination if the densities $s(x)$ and $b(x)$ are indeed multivariate Gaussians. If they are not, $\lambda(x)$ will not provide optimal discrimination, but it is easy to compute and may still provide a useful measure of discrimination.

Fisher considered the situation in which the means for the two classes differ, but their covariance matrices are identical ; for example, $\Sigma = \Sigma_S + \Sigma_B$. In this case, the quadratic discriminant reduces (modulo non-essential constants) to Fisher's discriminant

$$\lambda(x) = w \cdot x, \tag{1.32}$$

as illustrated in Fig. 1.4, where the vector w is given by

$$w = \Sigma^{-1}(\mu_S - \mu_B). \tag{1.33}$$

This is the simplest multivariate discriminant, beyond ones based on cuts such as the random grid search, and, like the quadratic discriminant, is easy to compute. We expect the Fisher discriminant to work well for signal and background densities whose 3rd or higher order correlations are small.

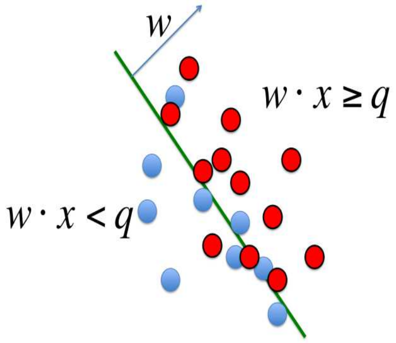

FIGURE 1.4 – The Fisher discriminant is a hyper-plane, with orientation given by the vector w, that partitions the space of variables x into two regions defined by the inequalities $w \cdot x \geq q$ and $w \cdot < q$, where q is an appropriate threshold.

1.3.3 Support Vector Machines

The Fisher discriminant is linear in the discriminating variables and therefore has the virtue of simplicity. However, it may fail completely for highly non-Gaussian densities. In the early 1990s, a significant advance occurred when it was recognized [1] that the linearity of the Fisher discriminant could be maintained, while achieving excellent discrimination even in highly non-Gaussian situations, if one can find a suitable map, $h : x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^H$, of the original n-dimensional space $\mathbb{R}^n$ of discriminating variables x into a space $\mathbb{R}^H$ of very high, perhaps infinite, dimensions.

Consider 3 parallel hyper-planes, A, B and C, written, without loss of generality, as

$$\begin{aligned} w \cdot h(x_1) + b &= +1 && (1.34)\\ w \cdot h(x) + b &= 0 && (1.35)\\ w \cdot h(x_2) + b &= -1, && (1.36) \end{aligned}$$

respectively. Plane B lies between the other two planes, as showen in Fig. 1.5. Moreover, plane A contains the point $h(x_1)$ and plane C the point $h(x_2)$. For now, we assume that the two classes of points are perfectly separable and hence by construction no points lie between planes A and C. Subtracting the equation for plane C from that of plane A yields $w \cdot [h(x_1) - h(x_2)] = 2$, which in terms of the unit vector $\hat{w} = w/|w|$, where $|w|$ is its norm, becomes $\hat{w} \cdot [h(x_1) - h(x_2)] = 2/|w|$. Plane B is called a **separating hyper-plane**. Intuitively, the best separating hyper-plane maximizes the separation, that is the **margin**, between planes A and C, $m = 2/|w|$. Therefore, the empirical risk function to be minimized may be taken to be $R(w) \propto |w|^2$. However, during the minimization, we need to make sure that we keep signal and background points separated. To that end, label points "above" plane A with the value $y = +1$ and those "below" plane C with $y = -1$. Points "above" plane A will have $w \cdot h(x) + b > 1$ and those "below" plane C will have $w \cdot h(x) + b < -1$. Consequently, all points will be correctly classified if the constraints $y_i[w \cdot h(x_i) + b] \geq 1, \forall i = 1, \cdots, N$ are satisfied, where N is the number of points in the training sample.

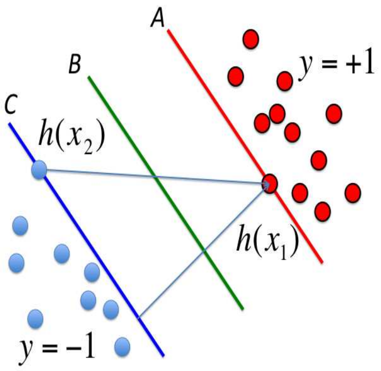

FIGURE 1.5 – Planes A, B and C are parallel with orientation given by the unit vector $\hat{w}$ normal to the planes. The distance between planes A and C, the margin, is just the projection of vector $h(x_1) - h(x_2)$ in the direction $\hat{w}$. Points "above" plane A are assigned the value $y = +1$, while points "below" plane C are assigned the value $y = -1$.

Therefore, the cost function to be minimized may be written as

$$C(w, b, \alpha) = \frac{1}{2}|w|^2 - \sum_{i=1}^{N} \alpha_i [y_i (w \cdot h(x_i) + b) - 1], \tag{1.37}$$

where the $\alpha_i > 0$ are Lagrange multipliers. When the cost function $C(w, b, \alpha)$ is minimized with respect to w and b, it may be re-written in the form

$$C(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j [h(x_i) \cdot h(x_j)]. \tag{1.38}$$

The optimization problem has now been reduced to that of minimizing an expression quadratic in the Lagrange multipliers, a problem that has been extensively studied [1]. At the minimum of $C(\alpha)$, most of the Lagrange multipliers α will turn out to be zero. The non-zero multipliers are those corresponding to points on the planes A and C ; these points are referred to as **support vectors** [1].

Unfortunately, in order to work with Eq. (1.38) two huge technical hurdles most be overcome. Firstly, a suitable mapping h of the vector x to some high, perhaps infinite dimensional, vector space must be found and secondly, an efficient way must be found to compute the scalar products of the vectors $h(x_i)$ and $h(x_j)$. The crucial insight that renders support vector machines, as this method is called, practical is that the scalar products $h(x_i) \cdot h(x_j)$ are equivalent to *some* kernel function $K(x_i, x_j)$, that is, a function which simultaneously maps a pair of vectors x_i and x_j to a space of perhaps infinite dimensions and performs the scalar product of the corresponding high-dimensional vectors $h(x_i)$ and $h(x_j)$. In terms of the kernel $K(x_i, x_j)$, the cost function to be minimized is

$$C(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j). \tag{1.39}$$

In practice, we usually do not know which mapping is best for a given problem and, consequently, we do not know *a priori* which kernel to use. We are therefore obliged to proceed by trial and error : we miminize the cost function using a few different standard kernels and use the one which gives the best results. Moreover, since signal and background events are generally non-separable, even in an infinite number of dimensions, we have to relax the constraints to (for example) $y_i(w \cdot x_i + b) \geq 1 - \zeta_i$, where the ζ_i are called, appropriately, **slack variables**. This will lead to a modified version of Eq. (1.38) in which, in addition to the coefficients α_i, the cost function will depend on the slack variables ζ_i, whose values will be determined during the miminization.

In expert hands, support vector machines can perform remarkably well, achieving discrimination close to the Bayes limit. However, it is generally difficult for a non-expert to obtain consistently excellent results.

1.3.4 Naïve Bayes

Every multivariate discrimination method that seeks to minimize the probability of classification errors must approximate the discriminant $D(x) = s(x)/[s(x) + b(x)]$, or some function thereof. Therefore, if we had accurate approximations for the densities $s(x)$ and $b(x)$ our problem would be solved. One of the most commonly used methods to approximate $D(x)$, which goes under many names — the most common being the misleading "likelihood discriminant," is called naïve Bayes. The method is very simple : we approximate each density as follows

$$\begin{aligned} s(x) &\approx \prod_{i=1}^{n} s_{1D}(x_i), \\ b(x) &\approx \prod_{i=1}^{n} b_{1D}(x_i), \end{aligned} \tag{1.40}$$

where $x = (x_1, x_2, \cdots, x_n)$ and s_{1D} and b_{1D} are the 1-dimensional marginal densities of $s(x)$ and $b(x)$, respectively ; that is, they are the projections of these densities onto the n axes x_1 to x_n. If the discriminating variables x_i are statistically independent, then the approximation signs in Eq. (1.40) can be replaced by equalities. Note that statistical independence is a more stringent condition than a lack of correlation. Variables that are uncorrelated can still be statistically dependent[3]. Nevertheless, this method can work well if the degree of statistical dependence is not too great. Moreover, the approximations to $s(x)$ and $b(x)$ are readily computed, using, for example, the next method to be discussed.

1.3.5 Kernel Density Estimation

As noted in the previous section, if it were possible to approximate the densities $s(x)$ and $b(x)$ with sufficient accuracy, our task would be over. In this section, we describe briefly a method first introduced by Parzens (see, for example, Refs. [2, 10]) in the 1960s, and developed subsequently by many people, called kernel density estimation (KDE). This method, which is illustrated in Fig. 1.6, is conceptually simple. A kernel function, $K(x, \mu)$, is placed at each training point μ. The density $p(x)$ at point x is approximated by the sum of the contributions from all the kernels,

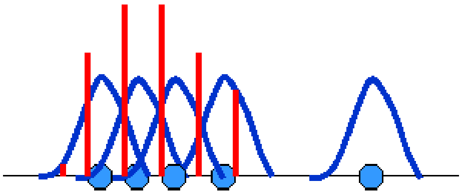

FIGURE 1.6 – A 1-dimensional example of the KDE method. A kernel with a suitably chosen width is placed at each training data point. The estimate of the density $p(x)$ at any point x (the vertical lines) is given by the sum of the contributions of all kernels at that point.

$$p(x) \approx \hat{p}(x) = \frac{1}{N}\sum_{j=1}^{N} K(x, \mu_j), \tag{1.41}$$

where N is the number of points in the training sample. In principle, any kernel can be used. In practice, most practitioners use diagonal multivariate Gaussians, $K(x, \mu) = \prod_{i=1}^{n} \text{Gaussian}(x_i|\mu, h_i)$, each dimension of which may have a different width (usually referred to as bandwidth) h_i. The method works in all dimensions, but is typically most practical if the dimensionality is not too high. In particular, it works very well in one dimension and can be used to implement the 1-dimensional densities of the naïve Bayes method.

---

[3] If variables are statistically independent, they are of necessity uncorrelated.

The key to this method is choosing the bandwidth h (that is, width) of the kernel. If the bandwidth is too narrow, the approximation will be very noisy. On the other hand, if the bandwidth were chosen too wide, the approximation would lose fine structure that may be present in the density being approximated. A considerable body of work exists on how these bandwidths should be chosen. For Gaussian densities, it can be shown that (see, for example, Ref. [2])

$$h = \sigma \left[\frac{4}{(n+2)N}\right]^{1/(n+4)} \tag{1.42}$$

is a good approximation to the optimal bandwidth in the sense that it minimizes the difference between the estimate and the true density. However, this bandwidth is far from optimal for densities that are highly non-Gaussian.

This point is illustrated in Fig. 1.7, which shows a kernel density estimate of the distribution of the simulated jet variables p_T, η, ϕ and p_T(true), where p_T and p_T(true) are the reconstructed, and true, jet transverse momenta, respectively. The bandwidths in each dimension are chosen according to Eq.(1.42). It is clear that this choice is not optimal. But, by a careful choice of bandwidths, the kernel density estimate

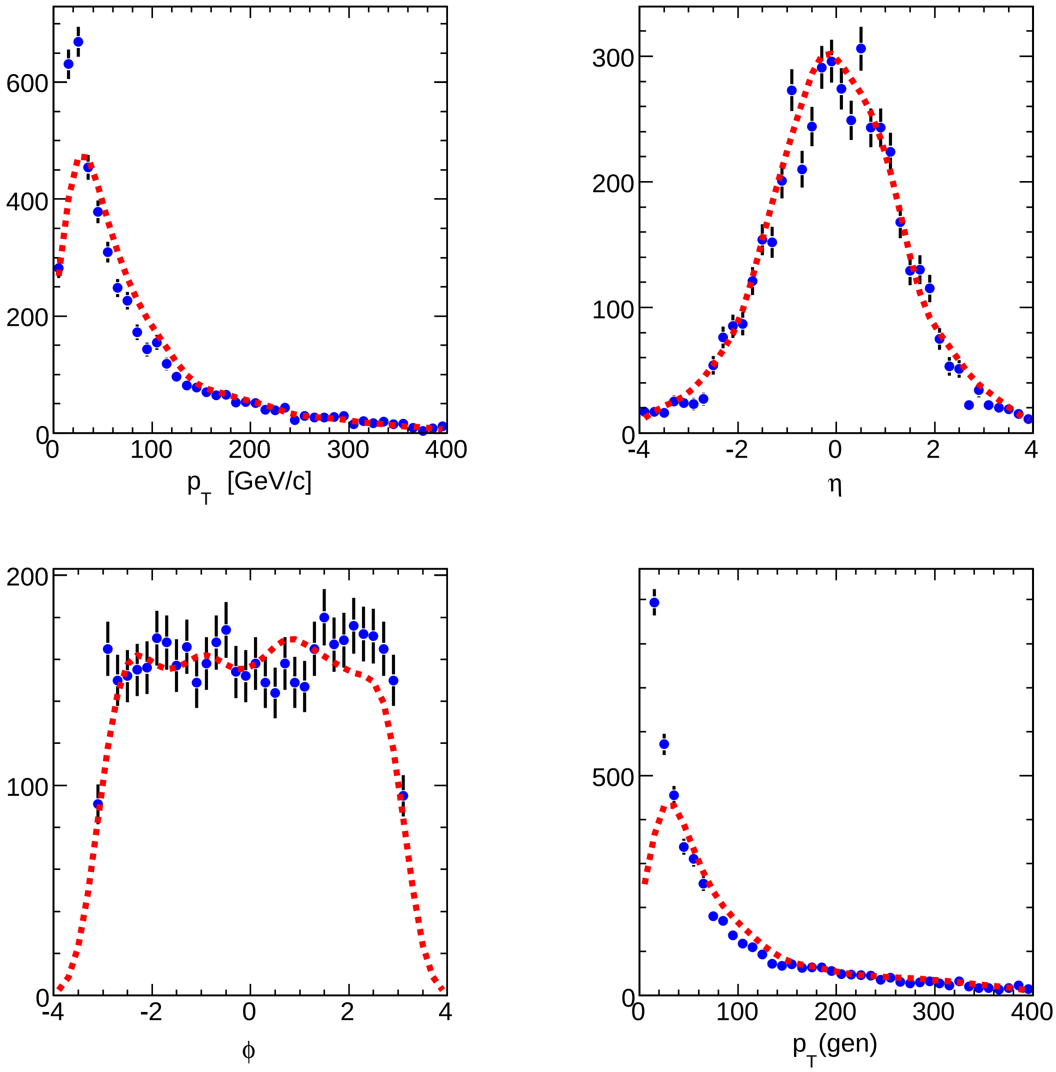

FIGURE 1.7 – Projections of a 4-dimensional density (points with error bars) compared with projections of a 4-dimensional KDE (dashed curve). The KDE uses bandwidths computed using Eq.(1.42). Clearly, these bandwidths do not yield a satisfactory approximation.

can be improved considerably, as is evident from Fig. 1.8. One application of the 4-dimensional density

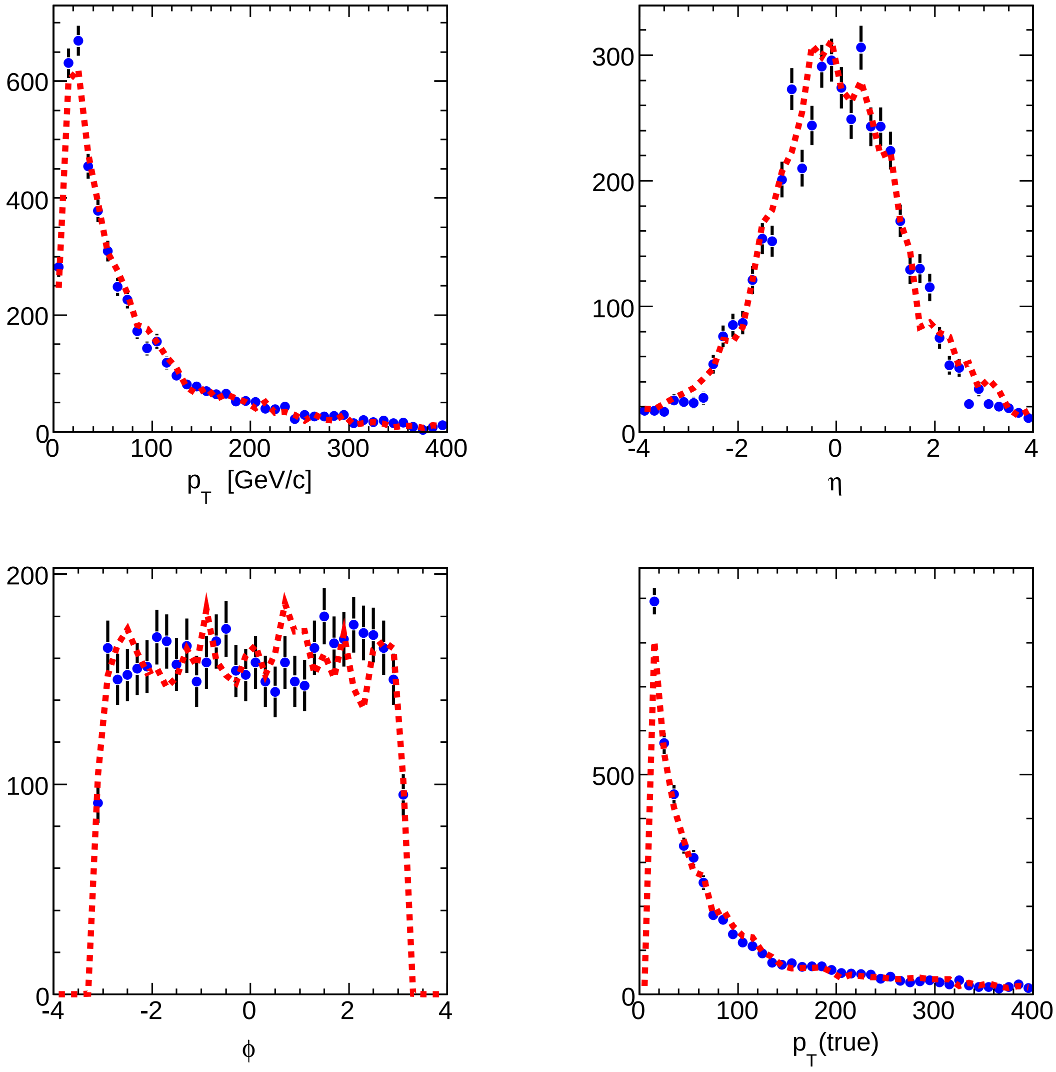

FIGURE 1.8 – Projections of a 4-dimensional density (points with error bars) compared with projections of a 4-dimensional KDE (dashed curve). The KDE uses somewhat better values for the bandwidths than those used in Fig. 1.7, yielding a much better kernel density estimate of the underlying distribution.

$p(p_T, \eta, \phi, p_T(\text{true}))$, along with $p(p_T, \eta, \phi)$, is to create a jet transverse momentum correction function. Traditionally, one assumes that the variables that determine the jet momentum correction function are statistically independent, which may or may not be true. However, given accurate estimates of the densities, the conditional density $p(p_T(\text{true})|p_T, \eta, \phi) = p(p_T, \eta, \phi, p_T(\text{true}))/p(p_T, \eta, \phi)$ can be calculated. The mean, or the mode, of this conditional density could be used as an estimate of the corrected jet transverse momentum.

Why does the KDE method work ? Here is a heuristic argument. Consider the limit of Eq. (1.41) as $N \to \infty$,

$$\hat{p}(x) = \int K(x, \mu)\, p(\mu)\, d\mu, \tag{1.43}$$

where $p(\mu)$ is the true density of the discriminating variable x. Presumably, just as would be done for the bins of a histogram, the kernel bandwidths would be made progressively smaller as more and more training data are used in the estimate $\hat{p}(x)$. Equation (1.42) shows that the optimal width (or rather an approximation to it) does indeed decrease with increasing N. In the limit $N \to \infty$, the kernels become δ-functions : $K(x, \mu) \to \delta^n(x - \mu)$ and, consequently, $\hat{p}(x) = p(x)$. The KDE method provides a consistent estimate of any reasonably behaved density.

In principle, any kernel will do provided that its width decreases with increasing size of the training sample. However, if the sample size is inadequate, the KDE method tends not to handle sharp structures accurately, such as boundaries ; the method tends to smooth things out. One way to improve accuracy, is to make the kernel width adapt to its local environment : where points are widely spread, a large bandwidth is used ; where data are tightly clustered, narrower kernels are used. Needless to say, there are many variations on this theme. However, these are merely refinements on what is fundamentally a simple idea.

1.3.6 Ensemble Learning

Until recently, the principal goal of developers of multivariate discrimination methods was finding effective methods to produce discriminants that performed near the Bayes limit. However, in the 1990s, the realization by Freund and Schapire [15] that high performance discriminants can be built by averaging over an ensemble of weakly performing ones — that is, averaging over **weak classifiers** that perform only marginally better than random guessing — has led to an explosion of methods that entail various kind of averaging.

Friedman and Popescu [5] have shown, however, that these methods can be embedded within a general theoretical framework they call **ensemble learning** in which the discriminant can be represented by a weighted average

$$f(x) = \sum_{i=1}^{K} a_k f(x, w_k). \tag{1.44}$$

Each method, is characterized by the choice of weak classifiers, $f(x, w_k)$, and the choice of coefficients a_k. The three most successful ensemble methods, **bagging** [6], **random forest** [7], and **boosting** [15] use decision trees [12, 16, 17, 18, 19] as the weak classifiers :

- **bagging** (**b**ootstrap **agg**regat**ing**) uses the coefficients $a_k = 1/N$, that is, a simple average is performed, where N is the number of trees — with each tree trained on a different bootstrap sample[4] drawn from the training sample ;
- **random forest** is bagging with the randomization of each tree, for example, by selecting a random subset of discriminating variables at each binary split within the tree, and
- **boosting** uses a particular form for the coefficients a_k, with each tree grown using a different weighting of the *full* training sample.

Although these methods use trees, in principle, they can be used with any weak classifier. 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.

AdaBoost

Decision trees are described in detail elsewhere in these proceedings [12], therefore, here we restrict our discussion of them to a few comments. Geometrically, a decision tree is simply an n-dimensional histogram. The cleverness is in how the bins of this histogram are constructed : they are constructed by recursively partitioning the space of discriminating variables. As is true of any histogram, a value is assigned to each bin (or leaf of the tree). For example, by counting how many signal and background events from the training sample contributed to a given bin, we could assign the value $y = -1$ if the background count was larger than the signal count or $y = +1$ if the reverse was true. Or we may wish to assign the value $y = s/(s + b)$, to approximate the discriminant $D(x)$ in the neighborhood of the point x, where s and b are the signal and background counts, respectively, in a given bin.

Decision trees are very fast to create compared with methods such as neural networks. Moreover, the reason why a particular event has been classified either as a signal or as a background event can be readily discerned from the sequence of `if - then - else` statements that comprise the decision tree. However, since a decision tree is a histogram, it provides only a *piece-wise constant* approximation to the function of interest, say $D(x)$. Therefore, if we want a smoother approximation we need to reduce the bin sizes, while simultaneously increasing the training sample size. Another way to produce a smoother, as well as better, approximation to the function of interest is to average over an ensemble of trees — a forest, if you will!

---

[4] A bootstrap sample is one drawn from another sample with replacement.

The `AdaBoost` (**Ada**ptive **Boost**ing [15]) algorithm is one of the most successful ensemble methods.

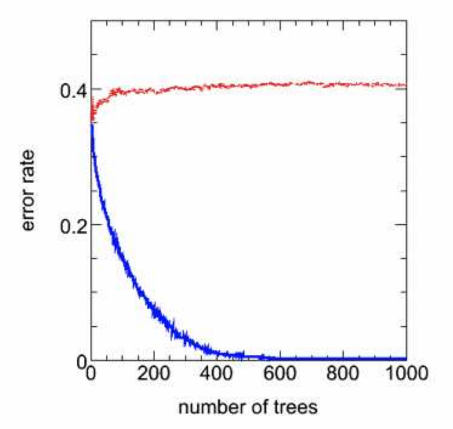

FIGURE 1.9 – This shows the event classification error rate as a function of the number of decision trees over which the boosted classifier has been averaged. It is striking that the error rate on the training sample goes exponentially to zero, while the error rate on an independent testing sample remains essentially constant.

The algorithm, in its original form, is based on decision trees $f(x, w)$ with binary values ± 1 and targets $y = \pm 1$. For a correct classification, the product $f(x, w)y$ is positive, while for an incorrect one it is negative. Given this observation, the algorithm proceed as follows

```
initialize weights $w_k$
repeat $K$ times :
    1. create a decision tree $f_k = f(x, w_k)$
    2. compute its error fraction $\epsilon_k$ on the current training sample
    3. set $a_k = \ln[(1 - \epsilon_k)/\epsilon_k]$
    4. scale the weight of the $n^{\mathrm{th}}$ 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.
compute $f(x)$ using Eq. (1.44).
```

Algorithms that progressively focus on incorrectly classified items are referred to generically as *boosting* algorithms. A classifier, for example sign($f(x)$), where $f(x)$ is given in Eq. (1.44), created through boosting is called a **boosted classifier**. The `AdaBoost` algorithm is as cryptic as it is remarkable. Figure 1.9 shows the results of applying `AdaBoost` to separate mSUGRA [14] events from $t\bar{t}$ events at the LHC. It is striking that when the boosted classifier is applied to the training sample, the error rate decreases to zero exponentially as the number of trees over which one averages increases. Indeed, it seems that given a sufficient number of trees, every *training* event would be classified *perfectly*. But what is truly remarkable, as shown in Fig. 1.9, is that the error rate of the boosted classifier, when applied to an *independent* testing sample, remains essentially constant even as error rate on the training sample goes to zero! A classifier that achieves an error rate close to zero on the training sample typically performs very badly on an independent set of events. This is true of all other classifiers, but it appears not to be true of those constructed using `AdaBoost`. One expects the `AdaBoost` algorithm to overtrain *eventually*. What is startling, is how far one can go before this happens. This extraordinary robustness to overtraining has been observed repeatedly. But, as yet, this behaviour has not been explained to everyone's satisfication (but see, for example, Ref. [20]).

1.3.7 Bayesian Neural Networks

The final method to be considered, Bayesian neural networks (BNN) [3, 21], is as the name implies one that embraces fully the Bayesian approach, which is to infer functions $f(x)$. Bayesian neural networks were deployed for the first time in the search for single top quark production at the Tevatron [17, 18]. In general, a BNN is simply the predictive distribution, Eq.(1.5), in which the function class is the class of **feedforward neural networks** with a fixed structure. However, as used by particle physicists [21], a BNN designed for classification is taken to be the *mean* of the predictive distribution, that is, it is the function

$$\begin{aligned} y(x) &= \int z\, p(z|x, T) dz, \\ &= \int f(x, w) p(w|T) dw. \end{aligned} \tag{1.45}$$

The second line follows from the form of the single-event likelihood function that the BNN method uses for classification, namely, $p(y|x, w) = f(x, w)^y \, [1 - f(x, w)]^{1-y}$, where the target y is binary-valued, $y \in \{0, 1\}$. By definition, $y = 1$ labels a signal event, while $y = 0$ labels background events. When $y = 1$, the likelihood function is $p(y|x, w) = f(x, w)$ and it is $1 - f(x, w)$ when $y = 0$; therefore, only $f(x, w)$ contributes to the mean of the predictive distribution, Eq. (1.45). The explicit function class used in Refs. [17] and [18], available by default in the `Flexible Bayesian Modeling` (`FBM`) software by Neal [3], is

$$f(x, w) = \frac{1}{1 + \exp[-g(x, w)]}, \tag{1.46}$$

where

$$g(x, w) = b + \sum_{j=1}^{H} v_j \tanh(a_j + \sum_{i=1}^{n} u_{ji} x_i), \tag{1.47}$$

where n is the dimensionality of the discriminating variables x, 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.

In realistic applications, the dimensionality of the parameter space of the functions $f(x, w)$ can number in the hundreds, precluding the use of standard (typically, adaptive) numerical integration methods, which are limited to dimensions no greater than about twenty. A feasible way to approximate the integral in Eq.(1.45) is by a Markov Chain Monte Carlo (MCMC) method, such as that implemented in the `FBM` software [3]. In the `FBM` software, an 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.(1.45), is then approximated by the average

$$y(x) \approx \frac{1}{M} \sum_{m=1}^{M} f(x, w_m), \tag{1.48}$$

where M is the number of points sampled from $p(w|T)$. If the Markov chain has converged — that is, the sampled points $\{w_j\}$ are a representative sample from $p(w|T)$, then Eq. 1.48 yields an unbiased estimate of the integral. However, because adjacent MCMC points are highly correlated, the average is often calculated using a sparse subset of the sampled points. Since the correlation of this sparse set is generally much lower than that of the original set of points, it somewhat easier to estimate how accurately the integral Eq. (1.45) has been approximated using Eq. (1.48).

We noted above the difficulty in specifying a prior over a space of functions. Instead, a prior density $\pi(w)$ is specified over the parameter space of the neural networks. Unfortunately, this is still a very difficult problem. What is done in practice is to use some plausible choice for the prior that captures some aspect of what is thought to be a sensible property, such as smoothness of approximation. In the `FBM` package, the prior is a product of zero-centered Gaussians, one for each parameter, with appropriately chosen widths. The `FBM` package provides mechanisms for setting the widths dynamically as the chain progresses.

We end with a simple 1-dimensional example of the construction of a BNN. Figure 1.10 shows distributions of the sum of (scaled) transverse momenta (H_T) for single top quark signal and background events

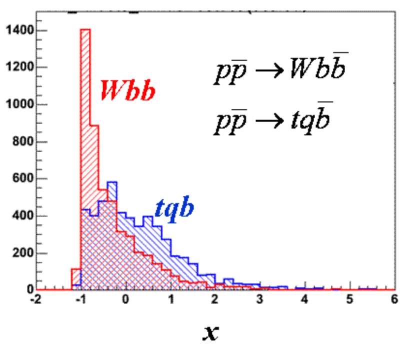

FIGURE 1.10 – Distributions of the sum of transverse momenta (H_T) for single top quark signal and background events in the 2-jet final state (see Ref. [18]). The sum of the signal and background distributions has been scaled to have zero mean and unit variance and each distribution is normalized to the same area.

with exactly 2 jets in the final state [18]. The function class used is the class of neural networks with a single hidden layer of $H = 20$ nodes. The training sample comprised 1000 signal plus 1000 background events. A sparse set of $M = 50$ sampled points were used in the average, Eq.(1.48). In Fig. 1.11 is plotted the 50 functions $f(x, w_k)$, corresponding to the 50 saved points w_k. Since the training sample contains equal numbers of signal and background events, each function $f(x, w_k)$ approximates the discriminant $D(x) = p(x|S)/[p(x|S) + p(x|B)]$. Since this is a 1-dimensional problem, the discriminant $D(x)$ can be approximated directly by computing the ratio $H(x|S)/[H(x|S)+H(x|B)]$, bin-by-bin, where $H(x|S)$ and $H(x|B)$ 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 shown in Fig. 1.10. The set of functions shown in Fig. 1.11 constitute the predictive distribution, Eq.(1.5). As is evident from Fig. 1.11, 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, an estimate of how well $D(x)$ has been approximated can be computed from the scatter, as a function of x, of the functions $f(x, w_k)$.

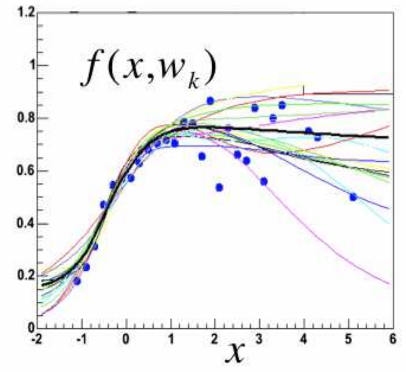

FIGURE 1.11 – Each curve is a plot of $f(x, w_k)$, where k indexes one MCMC parameter point, while the thick curve is their average as a function of x. The points are calculated from the bin-by-bin ratio of $H(x|S)/[H(x|S) + H(x|B)]$, where $H(x|S)$ and $H(x|B)$ are the signal and background histograms, respectively. This ratio provides a direct approximation of the discriminant $D(x) = p(x|S)/[p(x|S) + p(x|B)]$. 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.

1.4 Outstanding Issues

Multivariate discrimination methods have been used with considerable success in particle physics and are likely to continue to be indispensable for many analyses in the LHC era. Like all powerful tools, however, these methods must be used with due care. To that end, this requires the expenditure of some effort to understand what these methods can and cannot do. However, even though the methods are mathematically well-founded, and are the object of an enormous amount of active research, there are still important open questions and issues for which progress is sorely needed, a few of which are listed below.

- How can one verify that a discriminant function f is close to the Bayes limit ?
- A related question is : how can one confirm that an n-dimensional density $s(x)$ is well-modeled ? This is important for *any* analysis.
- If a sub-domain of a density $s(x)$ is not well-modeled, how can one find, characterize, and exclude, discrepant sub-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 ?
- Is there a practical way to quantify the information content of a sample of signal and background data so that when these data are compressed it is possible to assess how much information has been lost ?
- Is there a practical way to use multivariate discrimination when one does not know where to look for signals ?

At present, we are aware of only heuristic answers to some of these questions. One such heuristic answer is based on the known form of the discriminant, namely, $D(x) = s(x)/[s(x) + b(x)]$. Suppose we have constructed what we believe to be an accurate approximation, $\hat{f}(x)$, of $D(x)$. Then we would expect to recover the signal density $s(x)$ by weighting an admixture of signal and background events, with signal to background ratio k, with the weight function $w(x) = \hat{f}(x)/[\hat{f}(x) + (1 - \hat{f}(x))/k]$. Moreover, this should hold true for all values of k, in particular, for $k = 1$. This is a necessary condition for any discriminant that is close to the Bayes limit. It is not clear, however, that it is sufficient and for this reason remains a heuristic. Nevertheless, it has proven to be a particularly useful one in practice [18].

1.5 Conclusion

In this paper, we surveyed the theory and practice of multivariate discrimination. In so doing, the hope is that some light has been shed on methods that are often viewed as black boxes. We showed that if the goal is classification with the fewest errors, or the extraction of signal with the smallest uncertainty, the principal task is to approximate the discriminant $D(x) = s(x)/[s(x)+b(x)]$. All classification methods that share this goal are therefore equivalent and none is best in all circumstances. We have not dwelled at all on the algorithms for minimizing the various cost functions discussed. This was done deliberately : a distinction should be made between the results of a multivariate discrimination method, which is to furnish an approximation to some function of $D(x)$, and the algorithm used to minimize the cost function. A cost function can be minimized in many ways. The fact that in one case a genetic algorithm is used to minimize a cost function while in another case the back-propagation method is used does not alter the fact that what is being approximated is $D(x)$, in both cases. If minimizing classification errors is not the goal, however, then the discriminant $D(x)$ may not be the optimal function to use. The relevant cost function can still be minimized using any of a number of algorithms. But in general the resulting decision boundary will not have a simple probabilistic interpretation.

We have argued that machine learning and Bayesian learning are closely related. The difference is more one of emphasis : machine learning looks for the best fit while Bayesian learning focuses on averaging. However, with the advent of ensemble learning, ensemble averaging is now a feature of both the machine learning and Bayesian learning approaches.

Acknowledgments

I wish to thank the organizers for inviting me to lecture at this very enjoyable summer school.

Références

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

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

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

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

[5] 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`.

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

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

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

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

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

[11] J. Schwindling, *Les réseaux de neurones*, these proceedings.

[12] Y. Coadou, *Les arbres de décision*, these proceedings.

[13] H.B. Prosper, *The Random Grid Search : A Simple Way to Find Optimal Cuts*, Computing in High Energy Physics (CHEP 95) conference, Rio de Janeiro, Brazil, September 1995.

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

[15] 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).

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

[17] 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]`.

[18] 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]`.

[19] 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]`.

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

[21] 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.