Implementing likelihood-based statistical tests using a toy model in RooFit and RooStats

and

Exploring the basics of TMVA

by

Kayla McLean

kaylamc@uvic.ca

University of Victoria
3800 Finnerty Road
Victoria, British Columbia

Work Term Report

August 2014
Revised May 2015

Department of Physics and Astronomy
University of Victoria

Abstract

Likelihood-based statistical tests are widely used in ATLAS analyses and are extremely important in testing whether or not data is in agreement with a given hypothesis. After obtaining a dataset, there are two (of many) important tests to perform. Calculating the p-value for the null hypothesis, p_0, is one way to test of how compatible a dataset is with the background-only hypothesis. If the data is found to be compatible, as found in this experiment (using a generated dataset), then an upper limit can be set on the amount of signal that is present. This was done using the test statistic p_μ, and finding the 95% confidence level upper limit on the signal strength, μ. The first half of this report is dedicated to implementing these tests using `RooFit` and `RooStats`, and comparing their results in detail. Applying constraints on nuisance parameters was also included in this analysis. The second half of the report summarizes some of the basic discrimination techniques used in `TMVA` to improve the separation between signal and background events from a dataset, with a focus on Fisher discriminants and boosted decision trees.

Contents

I Likelihood-based statistical tests **1**

1 Introduction **1**

2 Theory **2**
2.1 Test statistic for discovery of a positive signal 3
2.2 Test statistic for upper limits 3
2.3 Asymptotics 4
2.4 Statistical significance 5

3 Analysis **5**
3.1 Toy model 5
3.2 `RooFit` 6
3.3 `RooStats` 8

4 Results **9**
4.1 `RooFit` VS `RooStats` 9
4.2 Nuisance parameter constraints 12

5 Conclusions **14**

II Exploring `TMVA` **14**

6 Introduction **14**

7 Theory **14**
7.1 Fisher discriminants 14
7.2 Boosted decision trees 15

8 Analysis and Results **16**
8.1 Input variables 16
8.2 `TMVA` training output 16

9 Conclusions **18**

10 Acknowledgements **18**

List of Figures

1 Visualization of the definition of the p-value and Z. 5
2 Experimental data compared to model curve 6
3 q_0 distribution as obtained from `RooFit`. 10
4 p_μ distribution as obtained from `RooFit` 10
5 $\hat{\mu}$, $\hat{\lambda}$, and $\hat{\mu_b}$ distributions obtained from `RooFit` 11
6 $\hat{\mu}$, $\hat{\lambda}$, and $\hat{\mu_b}$ distributions obtained from `RooFit` with a $G(1, 0.01)$ constraint on λ 13
7 Classification using linear discrimination. 14
8 Classification using cut-based discrimination. 15
9 An example of a boosted decision tree 15
10 Input variable distributions for signal and qq background. 16
11 Fisher discriminant response. 17
12 Boosted decision tree response. 17
13 ROC curve comparing Fisher and BDT efficiencies. 17
14 Optimal BDT output cut values. 18

List of Tables

1 Relation between Z, the p-value (one-sided), and the chance to obtain as (or more) incompatible data for a given hypothesis. 5
2 Parameter values used to generate experimental dataset. 6
3 `RooFit` and `RooStats` result comparisons. 10
4 `RooFit` and `RooStats` result comparisons after including a Gaussian constraint on λ. 12

Part I
Likelihood-based statistical tests

1 Introduction

In the ATLAS experiment, physicists are constantly searching for new physics beyond what is currently predicted by the Standard Model. Particle data from proton-proton collisions is collected from the ATLAS detector, and is analyzed to find if it is compatible with our current theories. Perhaps the most important test to perform on the data is to determine whether or not it is compatible with the *null* (background-only) *hypothesis*. If it can be determined that the data measured is incompatible with the null hypothesis, then a discovery can be claimed.

The method used to determine compatibility between experimental data and a hypothesis in ATLAS is a frequentist likelihood-based approach; we are interested in *the probability to observe as discrepant or more discrepant data than the experimental data, assuming that a given hypothesis is true* (for example, the null hypothesis). If this probability is large, then the data is compatible with the hypothesis. However, if this probability is small, then that means that the chance to observe the measured data is very unlikely, and the proposed hypothesis is therefore not likely. This probability is known as the *p-value* for the proposed model. The p-value often quoted alongside discoveries is the p-value for the null hypothesis, p_0, which is a measure of the compatibility between the measured data and the null hypothesis. The smaller the value of p_0, the more *significant* the discovery.

To test the "amount of signal" present in a dataset, we parametrize the expected number of signal events as being

$$\langle n_s \rangle = \mu s \tag{1}$$

Here, s is the predicted number of signal events, and μ is the *signal strength*. μ is the *parameter of interest* in the statistical tests to be discussed in this report. $\mu = 0$ corresponds to the null hypothesis, and this is the value that we are interested in testing our data against.

Once again, testing for compatibility with the null hypothesis is the most important test to perform on our data. This is the first of two tests discussed in this report. If it is found that the experimental data is *compatible* with the null hypothesis, then no discovery can be made; however, one can then put an *upper limit* on the signal strength, given the data. For this purpose, we test our data against different μ values (not $\mu = 0$) and calculate a p-value, p_μ. From this quantity, we can calculate an upper limit on μ at the *95% confidence level*. This is the second test to be discussed here.

In order to obtain the p-values for these two tests, p_0 and p_μ, we use what are known as *test statistics*. There are several different test statistics that exist, and their use depends on the hypotheses being tested, and on whether or not μ is taken to be one- or two-sided. For this analysis we chose to use the test statistics q_0 and $\tilde{q}_\mu$, which are used to test against the null hypothesis, and to find an upper limit on μ respectively, for a one-sided μ distribution [1]. μ is one-sided because we assume that a new signal would result in an *excess* of events compared to Standard Model predictions, i.e. $\mu \geq 0$. The first part of Section 2 discusses the statistical theory needed in order to introduce the test statistics q_0 in Section 2.1, and $\tilde{q}_\mu$ in Section 2.2. The main reason for using these test statistics is because they exhibit well-defined *asymptotic* behaviour for large enough datasets; a brief overview of this is given in Section 2.3.

Using these test statistics, an analysis was carried out in both `RooFit` and `RooStats`, two `ROOT` packages, which are typically used for modelling using probability densities, generating Monte Carlo experiments, and performing hypothesis tests. One "experimental" dataset was analyzed, which was generating using a Monte Carlo method and with no signal ($\mu = 0$). The model used assumed a Gaussian signal and an exponentially decaying background. This toy model is discussed in Section 3.1, and the analysis details for `RooFit` and `RooStats` are given in Sections 3.2 and 3.3 respectively.

The final results from both `RooFit` and `RooStats` are shown in Section 4, along with results after applying a constraint on one of the nuisance parame-

ters, a technique commonly done in ATLAS. Comparisons and conclusions are made in Section 5. In addition, the `RooFit` and `RooStats` code used in this analysis have been made available on SVN [2].

2 Theory

The test statistics used in this analysis are based on *frequentist* statistical theory. To begin, we first parametrize the expected (or average) total number of events obtained from an experiment. This is the sum of the expected number of signal events $\langle n_s \rangle$ and background events $\langle n_b \rangle$. The amount of observed signal is parametrized by μ, our parameter of interest. The expected number of total observed events is therefore defined to be

$$\langle n \rangle = \langle n_s \rangle + \langle n_b \rangle = \mu s + \mu_b b. \tag{2}$$

From this definition, it is easy to see that $\mu = 0$ corresponds to the background-only hypothesis, and the proposed signal + background hypothesis has $\mu = 1$. Once again it is assumed that $\mu \geq 0$. μ_b is the *background strength parameter*; this parameter is needed in order to ensure good modelling.[1]

n follows a Poisson distribution with mean $\langle n \rangle$. The number of signal events follows some probability density function $f_s(x|\boldsymbol{\theta}_s)$, where $\boldsymbol{\theta}_s$ are the signal nuisance parameters. Likewise, the number of background events follows $f_b(x|\boldsymbol{\theta}_b)$, with $\boldsymbol{\theta}_b$ as the background nuisance parameters. From this, we construct a *likelihood function* for the observed data $\boldsymbol{x}$. The likelihood of obtaining the dataset $\boldsymbol{x}$, given a model, μ, and the other nuisance parameters $\boldsymbol{\theta}$, is the product of the probabilities of obtaining n total events and obtaining the data $\boldsymbol{x}$. Therefore the likelihood function is given by the *marked Poisson process*

$$L(\boldsymbol{x}|\mu, \boldsymbol{\theta}) = \text{Pois}(n|\langle n \rangle) \prod_{j=1}^{N} f(x_j|\mu, \boldsymbol{\theta}). \tag{3}$$

[1] The model used in this analysis is an extended likelihood, i.e. n is allowed to vary according to a Poisson distribution (Equation 3). This Poisson mean is $\langle n \rangle = \mu s + \mu_b b$. Without this term ($n$ fixed), μ can be scaled so that μ_b would not be needed in the likelihood (equivalent to setting $\mu_b = 1$). In other words, if you let the total number of events vary, then you should let both signal *and* background yields vary.

Here, $\vec{\theta} = \left\{\vec{\theta}_s, \vec{\theta}_b, \mu_b\right\}$. $f(x|\mu, \boldsymbol{\theta})$ is the probability density function of the observable x, as assumed by the model. This is a combination of signal and background probability densities:

$$f(x|\mu, \boldsymbol{\theta}) = \frac{\mu s f_s(x|\boldsymbol{\theta}_s) + \mu_b b f_b(x|\boldsymbol{\theta}_b)}{\mu s + \mu_b b} \tag{4}$$

This quantity is normalized, and we demand that it is always ≥ 0.

It is important to note that the likelihood function in Equation 3 can be modified to include constraints on nuisance parameters previously obtained from other measurements. For example, if one of the nuisance parameters, say, θ_1, is found to have a Gaussian distribution with mean μ_1 and standard deviation σ_1, then one can include this probability density in the product of the likelihood function, providing a constraint in the likelihood fit. Then, the likelihood function becomes

$$L(\boldsymbol{x}|\mu, \boldsymbol{\theta}) = \text{Pois}(n|\langle n \rangle) \times \left(\prod_{j=1}^{n} f(x_j|\mu, \boldsymbol{\theta}) \right) \times \text{Gauss}\left(\mu_1|\theta_1, \sigma_1\right). \tag{5}$$

Here, μ_1 is obtained from the auxiliary measurement. This method is described in detail in many ATLAS papers; for example, see Section II in the combined Higgs search paper from July 2012 [3]. A simple case was assumed in this analysis for one nuisance parameter in both `RooFit` and `RooStats`.

Now, given the likelihood function in Equation 3, the frequentist *profile likelihood ratio* may be defined as

$$\lambda(\mu) = \frac{L(\mu, \hat{\hat{\boldsymbol{\theta}}})}{L(\hat{\mu}, \hat{\boldsymbol{\theta}})}. \tag{6}$$

The denominator is simply the maximum likelihood function, and $\hat{\mu}$ and $\hat{\boldsymbol{\theta}}$ are the *unconditional maximum likelihood estimators* of μ and $\boldsymbol{\theta}$. These are the "most likely" estimates for the parameters, given the data. The numerator is the *profiled likelihood* for μ. Here, μ is fixed and $\hat{\hat{\boldsymbol{\theta}}}$ is the *conditional maximum likelihood estimator* for $\boldsymbol{\theta}$. Since the maximum likelihood is in the denominator, $0 \leq \lambda(\mu) \leq 1$. This ratio is useful because it tests a hypothesized μ against the most probable value obtained from

the data, $\hat{\mu}$. If μ and $\hat{\mu}$ are close in value, $\lambda(\mu)$ will be close to 1. However, if μ is much different from $\hat{\mu}$, then $\lambda(\mu)$ will decrease because the profiled likelihood will be less for a μ far from $\hat{\mu}$; therefore smaller values of $\lambda(\mu)$ correspond to *greater incompatibility* between μ and $\hat{\mu}$. This ratio is included in the definitions of the test statistics that will be introduced in the following two sections.

2.1 Test statistic for discovery of a positive signal

The equations defined in these next three sections are taken from Cowan *et al.* [1], unless otherwise stated. Also, it is important to remember that these test statistics assume $\mu \geq 0$, but they allow $\hat{\mu} < 0$.[2]

For the discovery of a positive signal we are interested in rejecting the null hypothesis; if we can reject the $\mu = 0$ hypothesis, then we can claim a discovery. For this test, the statistic of interest is defined using the profile likelihood ratio from Equation 6 with $\mu = 0$:

$$q_0 = \begin{cases} -2\ln\lambda(0) & \hat{\mu} \geq 0 \\ 0 & \hat{\mu} < 0 \end{cases} \tag{7}$$

This statistic tests the $\mu = 0$ hypothesis against $\hat{\mu}$. q_0 will always be ≥ 0. If $\hat{\mu}$ is very different than 0, then $\lambda(0)$ will be small, corresponding to a large q_0. Therefore larger values of q_0 correspond to increasing incompatibility between $\hat{\mu}$ and $\mu = 0$. Note that negative $\hat{\mu}$ values are not considered to be discrepant; this is because we assume that $\mu \geq 0$, however we still want an estimator that can be negative.

Now the p-value for the null hypothesis can be defined from q_0:

$$p_0 = \int_{q_0,\text{obs}}^{\infty} f(q_0|0)\mathrm{d}q_0 \tag{8}$$

Here, $q_{0,\text{obs}}$ is the observed q_0 value as obtained from the experimental data, and $f(q_0|0)$ is the probability distribution of q_0. $f(q_0|0)$ can be obtained by generating Monte Carlo toys (with various generated values for n) and calculating q_0 for each toy; however, if the dataset (n) is large enough, then $f(q_0|0)$ can be determined exactly using asymptotics (see Section 2.3).

p_0 is *the probability of a q_0 value to be as or more discrepant with the null hypothesis than the observed q_0*. This makes sense; if one obtains a very small p_0 value (equivalent to a very large q_0 value), then the probability to have seen the data, assuming $\mu = 0$, is very small. Yet somehow the data has been observed, despite a very small probability of it happening. This means that $\hat{\mu}$ is not compatible with 0, and thus the null hypothesis is likely not feasible. Hence, a new model must be considered where $\mu \geq 0$ (i.e. a new discovery can be claimed). Quoting a *very* small value for p_0 is an important test when declaring a new discovery in ATLAS.

2.2 Test statistic for upper limits

Now we assume that our experimental data is *compatible* with the null hypothesis. In this case, we can still extract useful information from our data, despite the fact that a discovery cannot be made. We are now interested in putting an upper limit on μ at the 95% confidence level.

Before introducing the test statistic, we need to define a modified profile likelihood ratio:

$$\tilde{\lambda}(\mu) = \begin{cases} \frac{L(\mu,\hat{\hat{\boldsymbol{\theta}}}(\mu))}{L(\hat{\mu},\hat{\boldsymbol{\theta}})} & \hat{\mu} \geq 0 \\ \frac{L(\mu,\hat{\hat{\boldsymbol{\theta}}}(\mu))}{L(0,\hat{\hat{\boldsymbol{\theta}}}(0))} & \hat{\mu} < 0 \end{cases} \tag{9}$$

The case with $\hat{\mu} \geq 0$ is the ordinary ratio as we had before in Equation 6. For the case with $\hat{\mu} < 0$, μ is set to 0 in the denominator which is now also a profiled likelihood. The reason for this complicated definition once again has to do with the use of asymptotics (discussed in the next section). Now we can define the second test statistic for the purpose of finding an upper limit on μ:

[2] One could hypothetically define a test statistic that does not allow for $\hat{\mu} < 0$, but letting $\hat{\mu}$ be negative allows for the modelling of $\hat{\mu}$ as a Gaussian distributed variable. This means that asymptotics can be used to model the distributions of q_0, $\tilde{q}_\mu$, and the quantities that follow from them [1].

$$\widetilde{q}_\mu = \begin{cases} -2\ln\widetilde{\lambda}(\mu) & \hat{\mu} \leq \mu \\ 0 & \hat{\mu} > \mu \end{cases}$$
$$= \begin{cases} -2\ln\frac{L(\mu,\hat{\hat{\boldsymbol{\theta}}}(\mu))}{L(0,\hat{\hat{\boldsymbol{\theta}}}(0))} & \hat{\mu} < 0 \\ -2\ln\frac{L(\mu,\hat{\hat{\boldsymbol{\theta}}}(\mu))}{L(\hat{\mu},\hat{\boldsymbol{\theta}})} & 0 \leq \hat{\mu} \leq \mu \\ 0 & \hat{\mu} > \mu \end{cases} \quad (10)$$

The form of this statistic closely resembles that of q_0, except now the directions of the conditional statements have been reversed. Again, the details are related to the convenience of asymptotics. For this test we are most interested in testing various values of μ against $\hat{\mu}$ (except for a modified ratio when $\hat{\mu} < 0$). When $\hat{\mu} > \mu$, $\tilde{q}_\mu$ is set to 0 because here we do not consider this case to show increasing discrepancy. This is because we are interested in finding an *upper* limit on μ. This is the sole purpose of this test statistic.

Similar to before, we define a p-value for the μ hypothesis:

$$p_\mu = \int_{\tilde{q}_{\mu,\text{obs}}}^{\infty} f(\tilde{q}_\mu|\mu)\mathrm{d}\tilde{q}_\mu \quad (11)$$

This is the p-value for a given μ value. This definition is nearly identical to that for p_0, but now it is an integral over the probability distribution of $\tilde{q}_\mu$. Once again this distribution can be determined from Monte Carlo pseudo-experiments, or from asymptotics.

Once p_μ has been determined for several μ values, a 95% confidence level upper limit can be set. This is defined as the *the largest value of μ for which $p_\mu \geq 0.05$*. This value, μ^{upper}, is the upper limit on μ at the 95% confidence level. This quantity gives a measure of the maximum amount of signal that can be present, given the dataset.

2.3 Asymptotics

The two test statistics of interest in this analysis are defined in clever ways in order to allow the use of asymptotic approximations in order to determine their distributions. These approximations follow from the work of Wilks [4] and Wald [5]. Their work predicts the exact form of the profile likelihood ratio for a large dataset (n):

$$-2\ln\lambda(\mu) = \frac{(\mu-\hat{\mu})}{\sigma^2} + \mathcal{O}\left(1+\frac{1}{\sqrt{n}}\right) \quad (12)$$

This formula assumes that $\hat{\mu}$ has a Gaussian distribution with mean μ' and RMS σ. From this approximation, the exact distributions of q_0 and $\tilde{q}_\mu$ are obtained, and toys are not needed in order to find the corresponding p-values. This is extremely useful, and is the main reason why asymptotics are used in these tests.

For the case of q_0, the exact definition is given as follows:

$$q_0 = \begin{cases} \hat{\mu}^2/\sigma^2 & \hat{\mu} \geq 0 \\ 0 & \hat{\mu} < 0 \end{cases} \quad (13)$$

From this, the distribution of q_0 is

$$f(q_0|0) = \frac{1}{2}\delta(q_0) + \frac{1}{2}\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{q_0}}e^{-q_0/2}, \quad (14)$$

and the p-value is obtained from the cumulative distribution function of $f(q_0|0)$ as

$$p_0 = 1 - F(q_0|0), \quad (15)$$

where $F(q_0|0) = \Phi(\sqrt{q_0})$ (Φ is the cumulative distribution function of the standard Gaussian).

The equations that follow from $\tilde{q}_\mu$ are almost identical. The test statistic is found to be

$$\widetilde{q}_\mu = \begin{cases} \frac{\mu^2}{\sigma^2} - \frac{2\mu\hat{\mu}}{\sigma^2} & \hat{\mu} < 0 \\ \frac{(\mu-\hat{\mu})^2}{\sigma^2} & 0 \leq \hat{\mu} \leq \mu \\ 0 & \hat{\mu} > \mu \end{cases}, \quad (16)$$

with the distribution being

$$f(\tilde{q}_\mu|\mu) = \frac{1}{2}\delta(\tilde{q}_\mu) + \begin{cases} \frac{1}{2}\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{\tilde{q}_\mu}}e^{-\tilde{q}_\mu/2} & 0 < \tilde{q}_\mu \leq \frac{\mu^2}{\sigma^2} \\ \frac{1}{\sqrt{2\pi}(2\mu/\sigma)}e^{-\frac{1}{2}\frac{(\tilde{q}_\mu+\mu^2/\sigma^2)^2}{(2\mu/\sigma)^2}} & \tilde{q}_\mu > \frac{\mu^2}{\sigma^2} \end{cases} \quad (17)$$

and the p-value given by

$$p_\mu = 1 - F(\tilde{q}_\mu|\mu), \quad (18)$$

This collection of formulas are used to double-check the distributions obtained from toys. One last important formula to note is the theoretical value for the upper limit on μ at confidence level $1-\alpha$:

$$\mu^{\text{upper}} = \hat{\mu} + \sigma\Phi^{-1}(1-\alpha) \tag{19}$$

The paper by Cowan *et al.* [1] has more details on these asymptotic equations.

2.4 Statistical significance

When the Higgs boson discovery was first announced in July 2012, a "5σ significance" was also quoted. This quantity has a corresponding p-value, but converting the p-value into units of a Gaussian standard deviation is more convenient.

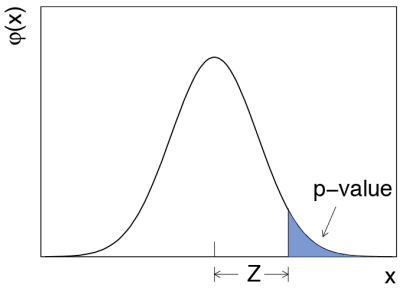

Figure 1: Visualization of the definition of the p-value and Z.

The relationship between the p-value and the significance Z is shown in Figure 1. The function $\phi(x)$ is the standard normal distribution, given by

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}. \tag{20}$$

It should be noted that the p-value shown in Figure 1 is for a one-sided test statistic. In this case the relationship between Z and the p-value is defined such that "a Gaussian distributed variable found Z standard deviations above its mean has an upper-tail probability equal to p" [1]. Z is therefore

$$Z = \Phi^{-1}(1-p), \tag{21}$$

where Φ^{-1} is the quantile (inverse of the cumulative distribution $\Phi(x)$) of the standard Gaussian.

Table 1 summarizes the corresponding p-values for a given Z, as well as the "chance" to obtain the data. This table makes it clear why quoting a value for Z is more convenient than quoting a p-value. In ATLAS, a 3σ significance corresponds to "evidence" for a theory, and a 5σ significance corresponds to a discovery. For the Higgs discovery, the significance quoted was calculated from p_0.

Z (σ)	p-value	Chance
1	1.587×10^{-1}	1/6
2	2.275×10^{-2}	1/44
3	1.349×10^{-3}	1/741
4	3.167×10^{-5}	1/31,574
5	2.867×10^{-7}	1/3,488,556

Table 1: Relation between Z, the p-value (one-sided), and the chance to obtain as (or more) incompatible data for a given hypothesis.

3 Analysis

3.1 Toy model

In order to study some test statistics, a "proposed" model must be defined. For this study we used a Gaussian signal where all parameters were assumed to be known (hence no nuisance parameters). The signal probability density function was defined to be

$$f_s(x|\boldsymbol{\theta}_s) = f_s(x) = \frac{1}{\sigma_x\sqrt{2\pi}} e^{\frac{-(x-\mu_x)^2}{2\sigma_x^2}}, \tag{22}$$

where μ_x and σ_x are the mean and standard deviation of the Gaussian signal respectively. Note that this function is assumed known with no free parameters. Similarly, we defined the background as being an exponentially decaying probability density given by

$$f_b(x|\boldsymbol{\theta}_b) = f_b(x|\lambda) = \frac{1}{\lambda} e^{-\frac{x}{\lambda}}, \tag{23}$$

where λ is the background shape nuisance parameter.[3] The addition of these two probability densities gives the total model PDF, $f(x|\mu, \boldsymbol{\theta})$, as defined in the likelihood in Equation 3. The other two parameters, μ and μ_b, appear in this PDF in order to

[3] This is not to be confused with the profile likelihood ratio $\lambda(\mu)$.

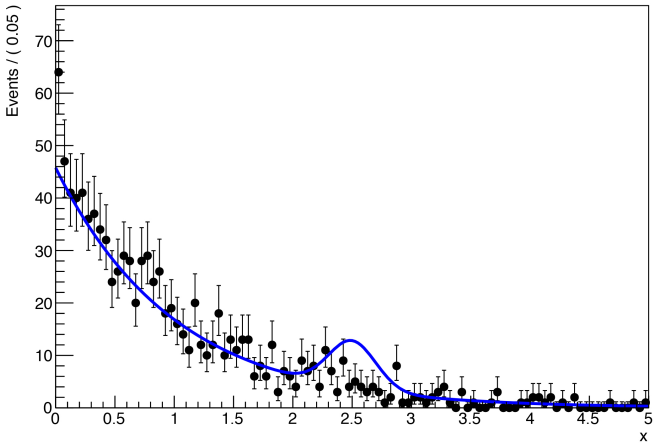

Figure 2: Experimental data compared to model curve. The black points are generated with $\mu = 0$ and the blue line is the model assuming $\mu = 1$.

ensure proper normalization (see Equation 4). μ_b can be thought of as another nuisance parameter. μ is our parameter of interest. The nuisance parameters are denoted $\vec{\theta} = \{\lambda, \mu_b\}$, with $\vec{\theta}_b = \{\lambda\}$ and $\vec{\theta}_s = \{\}$. x is the random variable we are measuring; it could technically be anything (mass, energy, etc). This model PDF in particular was formulated to mimic the PDF used in the $H \to \gamma\gamma$ analysis, where the x variable was the invariant mass of the two photons, $m_{\gamma\gamma}$.

Figure 2 shows the "experimental" dataset used in this analysis, as well as the model PDF. The blue line is not a fit to the data. The data was generated using Monte Carlos with $\mu = 0$. This was done in order to ensure that the data would be compatible with the null hypothesis, and an upper limit test could be carried out as well. The data therefore follows the background-only PDF (an exponential decay). The blue line shows the total PDF with the Gaussian bump and $\mu = 1$. Table 2 summarizes the parameter values used in order to generate the experimental data.

Once the model is defined and the experimental dataset is generated, the analysis is carried out in both `RooFit` and `RooStats`.

3.2 `RooFit`

`RooFit` is a `ROOT` package meant to "provide a toolkit for modeling the expected distribution of

Parameter	Value
b	1000
s	100
λ	1.0
μ_b	1.0
μ_x	2.5
σ_x	0.2

Table 2: Parameter values used to generate experimental dataset.

events in a physics analysis. Models can be used to perform likelihood fits, produce plots, and generate 'toy Monte Carlo' samples for various studies. The `RooFit` tools are integrated with the object-oriented and interactive `ROOT` graphical environment" [6]. This is the first tool used to calculate test statistics for the dataset, and the results are then compared to the same analysis done in `RooStats`. The programs written for this analysis is available on SVN.

There are four main parts to the `RooFit` program. The first is dedicated to defining the model PDF and parameters. A Gaussian PDF object can be defined using a `RooGaussian` object. The following code shows an example of how to do this. The signal yield μs is also calculated by defining μ and s separately as `RooRealVars`, and then multiplying them together using `RooFormulaVar`. Then the model PDF is created from the signal and background (not shown) PDFs using `RooAddPdf`. This automatically creates an extended likelihood because the yields μs and $\mu_b b$ are both given as arguments. If only one argument is given, then `RooFit` assumes that the likelihood is not extended.

```
RooGaussian signal("signal", "gaussian
   signal PDF", x, meanx, sigmax);

RooRealVar s("s", "signal yield", 100);
RooRealVar mu("mu", "signal strength",
   1., -10., 10.);
RooFormulaVar mus("mus", "mu*s",
   RooArgList(mu, s));

RooAddPdf model("model", "gaussian plus
   exponential PDF", RooArgList(signal,
   background), RooArgList(mus, mubb));
```

The second part of the program is dedicated to *either* generating or reading-in experimental data. The data used in this analysis was generated using this method and is also provided in a `.root` file on SVN. To generate a dataset with only background, μ is set to 0 and the event yield is determined from a Poisson distribution with mean $\langle n \rangle = \mu s + \mu_b b$. An example of generating a `RooDataSet` according to the model, but with $\mu = 0$, is shown in the code below using a random number generator, where the generated dataset contains M events.

```
mu.setVal(0.);
int M = rdm->Poisson(mus.getVal() +
   mubb.getVal());
RooDataSet dataset =
   model.generate(x,M);
```

It is important to note that the "experimental" dataset in this study was generated with $M = 1000$.

The third section of the program is to calculate p_0. This requires finding q_0^{exp} for the experimental data, and then running toy pseudo-experiments and calculating a q_0 value for each, according to Equation 7. For each toy, the values for λ and μ_b used to generate are set to $\hat{\hat{\lambda}}$ and $\hat{\hat{\mu}}_b$ respectively. Then the integral is carried out to calculate p_0 using Equation 8. The following section of code shows the calculation of q_0^{exp} (this is $q_{0,\text{obs}}$ in the formula for p_0).

```
double q0_exp = 0.;
double lambdaHatHat0;
double mubHatHat0;

mu.setVal(0.);
// calculate q0_exp
{
  // obtain ordinary NLL
  RooFitResult* r =
    model.fitTo(*dataset,
    Minimizer("Minuit2", "minimize"));
  // extract NLL and mu estimator
  double nllExp = r->minNll();
  double muHatExp0 = mu.getVal();
...
  // obtain profiled NLL for mu = 0
  mu.setVal(0.);
  mu.setConstant();
  r = model.fitTo(*dataset,
    Minimizer("Minuit2", "minimize"));
  // extract profiled lambda and mub
  // estimators for quality control
  lambdaHatHat0 = lambda.getVal();
  mubHatHat0 = mub.getVal();

  if (muHatExp0 >= 0.)  {
    // extract profiled NLL
    double pnll = r->minNll();
    // calculate q0_exp
    q0_exp = 2.*(pnll - nllExp);
  }
  // otherwise muHat < 0 and q0_exp = 0
}

// let mu vary again for toys
mu.setConstant(false);
```

Please note that some of the code has been modified (or omitted) here for clarity. See the full code for all details.

After this is done for the experimental data, we run toys by varying n and repeat the exact same methodology, calculating a q_0 for each. The code below illustrates this, as well as the p-value calculation.

```
// loop over toys
for (int i = 1; i <= nToys; ++i) {
  double q0 = 0.;
  mu.setVal(0.);
  mub.setVal(mubHatHat0);
  lambda.setVal(lambdaHatHat0);

  // create N events from Poisson
  // distribution
  int N = rdm->Poisson(mus.getVal() +
    mubb.getVal());
  // generate data from model
  RooDataSet* toyData =
    model.generate(x, N);
...
```

```
  // obtain ordinary NLL
  RooFitResult* r =
    model.fitTo(*toyData,
    Minimizer("Minuit2", "minimize"));
  double nll = r->minNll();

  // fill histograms with estimators
  hmuHat0->Fill(mu.getVal());
  hmuHat0err->Fill(mu.getError());
  hlambdaHat0->Fill(lambda.getVal());
  hlambdaHat0err->
    Fill(lambda.getError());
  hmubHat0->Fill(mub.getVal());
  hmubHat0err->Fill(mub.getError());

  if (mu.getVal() >= 0.)  {
    // obtain profiled NLL for mu = 0
    mu.setVal(0.);
    mu.setConstant();
    r = model.fitTo(*toyData,
      Minimizer("Minuit2", "minimize"));
    // extract profiled NLL
    double pnll = r->minNll();
    // calculate q0 for this toy
    q0 = 2.*(pnll - nll);
  }

  // q0 remains 0 if mu < 0
  // let mu vary again for the next toy
  mu.setConstant(false);

  // track how many times q0 > q0_exp
  // to calculate p0
  if (q0 > q0_exp) ++n;
  // fill histogram with q0 values
  f0->Fill(q0);
}

// calculate p-value
double p0 = double(n)/double(nToys); }
```

For each toy, we track whether or not q_0 is larger than $q_0^{\rm exp}$. Counting the number of times this occurs, the integral (p-value) is approximately that number divided by the total number of toys (this is just the fractional area above $q_0^{\rm exp}$).

Finally, the fourth and last part of the code calculates p_μ for a range of μ values, and then finds the 95% confidence level upper limit on μ. This essentially requires doing the exact same thing as for the p_0 calculation, except the test statistic $\tilde{q}_\mu$ is a bit more complicated, and we are interested in testing a range of μ values with $\mu \neq 0$, and calculating p_μ. Now we are required to run pseudo-experiments for each different μ value. This is more computationally intensive, especially when a scan over a fine range of μ is required, and thousands of toys are required for each. Therefore using asymptotics for finding p_μ is highly beneficial since no toys are needed. No example code is shown for this section because the idea is similar as for the calculation of p_0. Then at the end the program finds the maximum μ value above which $p_\mu < 0.05$.

Using `RooFit` is beneficial in that every single step in the program is controlled directly. However, at the same time programming every step is time consuming, and repeating this for every statistical test is not desirable. So a second tool, `RooStats`, is used to repeat this analysis and compare results, as well as ease of use.

3.3 `RooStats`

`RooStats` is a similar `ROOT` toolkit, but it contains much more advanced tools for running toys and computing test statistics. For example, all of the model information is contained in a class called `RooWorkspace`, and a `ModelConfig` is used to define different quantities inside the model. Here is some sample code from the program:

```
ModelConfig c("config");
c.SetWorkspace(w);
c.SetPdf("model");
c.SetObservable("x");
c.SetParameterOfInterest("mu");
c.SetNuisanceParameters("mub,lambda");
```

After this initial setup, the calculation of p_0 is very easy. The user sets up the test statistic using a `ProfileLikelihoodTestStat`, and then sets it to one-sided. Then $q_0^{\rm exp}$ is calculated from the dataset in one line. Then toys are set up using `ToyMCSampler` and the distribution of q_0 is found from the toys using

FrequentistCalculator. The result is then simply extracted using HypoTestResult (see below).

```
ProfileLikelihoodTestStat* plts = new
  ProfileLikelihoodTestStat(*c.GetPdf());
plts->SetOneSidedDiscovery(true);
double q0exp = 2.*
  plts->Evaluate(dataset, poi);

ToyMCSampler toymcs(*plts, ntoys);
FrequentistCalculator freqCalc(dataExp,
  c, cNull, toymcs);
HypoTestResult freqCalcResult =
  *freqCalc.GetHypoTest();
```

Then p_0, along with its error, are obtained from the HypoTestResult:

```
double p0 = freqCalcResult.NullPValue();
double p0Error =
  freqCalcResult.NullPValueError();
```

These samples of code show essentially *all* of the important lines for the calculation of p_0 in the RooStats program. What took 100 lines to do in RooFit was done in a handful of lines in RooStats.

For the case of calculating an upper limit, the results are similar; RooStats already has the functions programmed in order to do this easily. The method used to calculate the upper limit is known as the *Feldman Cousins* technique. The code below shows an example of how to set up the FeldmanCousins tool, setting the confidence level, and setting up the ProfileLikelihoodTestStat $\tilde{q}_\mu$. nMus is the number of μ values to study for the current range that μ has in its RooRealVar definition. nToysPerMu is the number of toys done for a given μ value.

```
FeldmanCousins fc(dataset, cNull);
fc.SetConfidenceLevel(0.95);
fc.SetNBins(nMus);

ToyMCSampler* toymcs =
  (ToyMCSampler*)fc.GetTestStatSampler();
toymcs->SetNToys(nToysPerMu);
ProfileLikelihoodTestStat* qmu = dynamic
  _cast<ProfileLikelihoodTestStat*>(toym
  cs->GetTestStatistic());
qmu->SetOneSided(true);
```

Then the upper limit is extracted from the interval calculator:

```
PointSetInterval* interval =
  fc.GetInterval();
double observedUL =
  interval->UpperLimit(*mu);
```

Once again, this example code does all of the calculations needed to find the upper limit, compared to pages of code needed to do the same in RooFit. This makes RooStats very useful in comparison; however, one of its downfalls is that it does not allow for the monitoring of toys. For example, in RooFit it is very easy to extract the estimator information for each toy, and plot their distributions (this is discussed in Section 4). This is not possible with RooStats. Everything is hidden "under the hood" and away from the user. Therefore the user needs to be confident that they understand RooStats and the complicated classes that it consists of.

4 Results

4.1 RooFit VS RooStats

This section summarizes the results from both RooFit and RooStats. Table 3 provides a brief summary of all estimator and test statistic values obtained from both. Overall we see extremely good agreement for estimator values, q_0^{exp}, and p_0. The upper limits are very similar to the asymptotic prediction, but they do differ by a few percent; this is expected to be due to the randomness of the toy Monte Carlos used to find the distribution of p_μ. We don't expect them to be exactly the same; the

Quantity	RooFit	RooStats
$\hat{\mu}$	0.135 ± 0.095	0.135 ± 0.095
$\hat{\mu_b}$	0.986 ± 0.033	0.986 ± 0.033
$\hat{\lambda}$	0.965 ± 0.037	0.965 ± 0.037
$\hat{\hat{\mu_b}}$ $(\mu = 0)$	1.000 ± 0.032	1.000 ± 0.032
$\hat{\hat{\lambda}}$ $(\mu = 0)$	0.990 ± 0.034	0.990 ± 0.034
q_0^{exp}	2.228	2.228
p_0	0.067	0.067 ± 0.002
p_0^{th}	0.068	0.068
Z	1.5σ	1.5σ
μ^{upper}	0.308	0.322
$\mu_{\text{th}}^{\text{upper}}$	0.309	0.309

Table 3: RooFit and RooStats result comparisons.

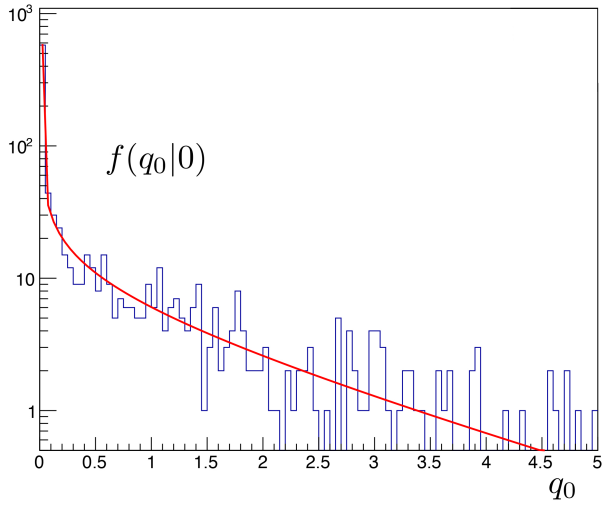

Figure 3: q_0 distribution as obtained from RooFit.

experimental datasets were identical, but the toys were not of course. It is also interesting to note that $\hat{\mu}$ for the dataset was 0.135; there was a statistical fluctuation in the data so that a small excess was seen. This is often what happens in an experiment, but this analysis showed that it was nothing to get excited about ($Z = 1.5\sigma$).

Figure 3 shows $f(q_0|0)$ using RooFit (a nearly identical plot can be easily extracted from HypoTestPlot in RooStats with one command). The red line on the plot shows the asymptotic prediction for $f(q_0|0)$, as given by Equation 14. The plot shown was generated for 10,000 toys. Similarly, Figure 4 shows the distribution of p_μ for

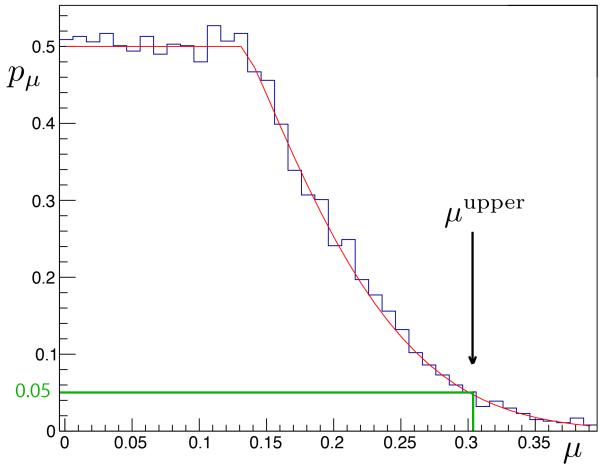

Figure 4: p_μ distribution as obtained from RooFit (the green line provides a visual aid and is not exactly accurate).

$0 < \mu \leq 0.4$. Once again the red line is the asymptotic prediction given by Equation 18. The green line shows where $p_\mu = 0.05$, and the corresponding μ^{upper}. This plot was *not* obtainable in RooStats; it simply calculates the upper limit, and does not allow the user to extract/plot any information from the toys used to find p_μ. This made it difficult to verify that the toys were being generated in the correct manner.

Monitoring toys was entirely possible using RooFit. Estimator values can be extracted easily by calling – for the case of $\hat{\mu}$ – `mu.getVal()` and `mu.getError()` after minimizing the negative log-likelihood. Then the estimator distributions may be plotted and monitored to ensure that they have the correct mean, that they follow a Gaussian distribution, etc. Unexpected behaviours in these plots led to the discovery of a bug in the program, or an error in the modelling. This was how it was determined that the μ_b nuisance parameter was needed in the model, because biases were seen in these plots without it. The correct distributions obtained from RooFit are shown in Figure 5.

The final step is to apply a nuisance parameter constraint to the likelihood, seeing as this is a very important procedure in ATLAS analyses. This is discussed in the next section.

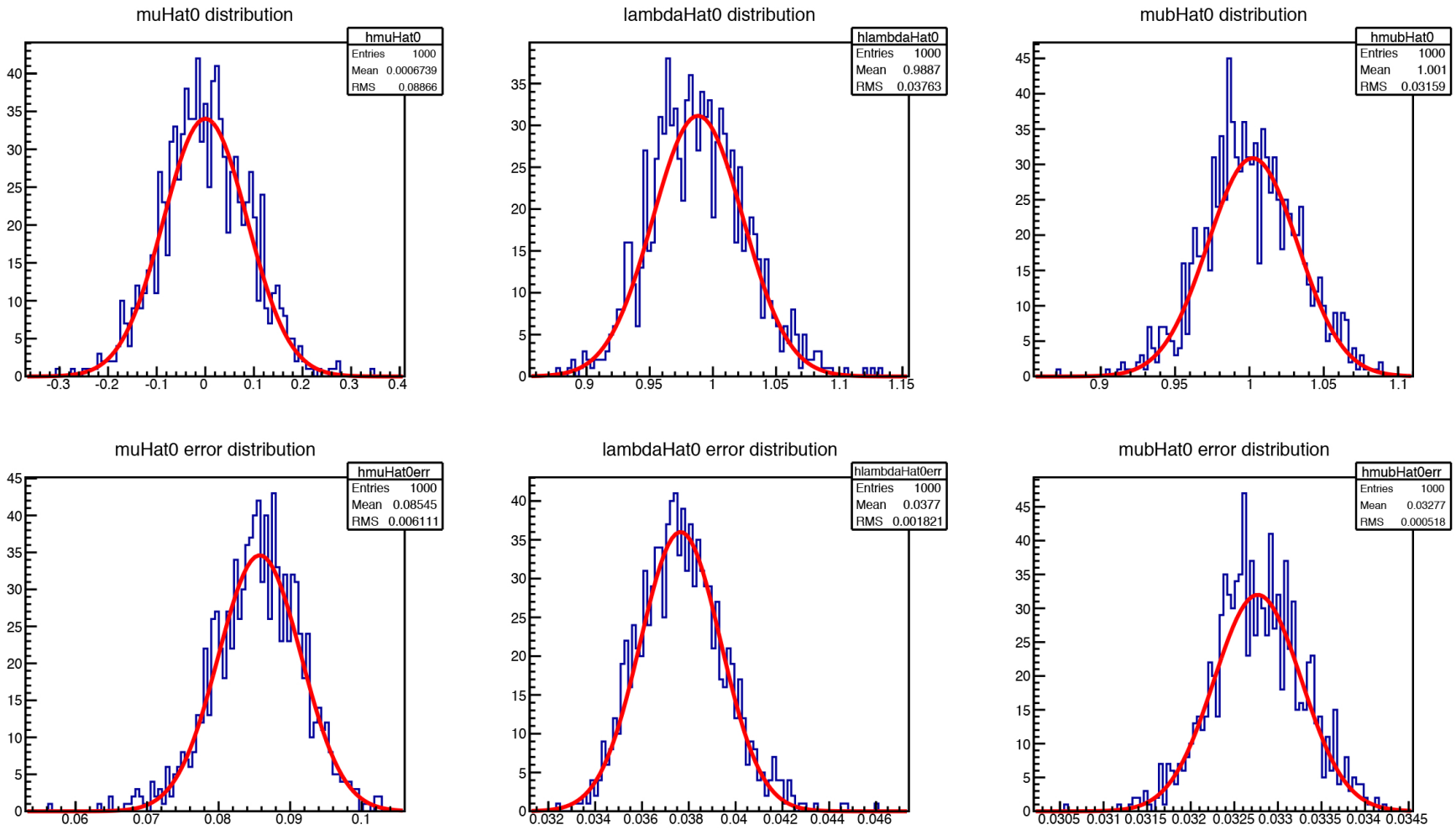

Figure 5: $\hat{\mu}$, $\hat{\lambda}$, and $\hat{\mu_b}$ distributions obtained from `RooFit` ($\mu = 0$ for p_0 calculation), and their errors.

4.2 Nuisance parameter constraints

Auxiliary measurements of nuisance parameters can be included in the likelihood function. This constrains the minimization of the negative log-likelihood in hopes of improving p_0 and the upper limit. In this analysis a Gaussian constraint is applied to λ, according to Equation 5. Now the likelihood function looks like

$$L(\vec{x}|\mu,\lambda,\mu_b) = \text{Pois}(n|\langle n\rangle) \times \left(\prod_{j=1}^{n} f(x_j|\mu,\lambda,\mu_b)\right) \times \text{Gauss}\left(\tilde{\mu}_\lambda|\hat{\hat{\lambda}},\sigma_\lambda\right). \quad (24)$$

Here, $\tilde{\mu}_\lambda$ is the mean value of λ drawn its Gaussian distribution from the auxiliary measurement. This is discussed in more detail at the end of this section.

In `RooFit` and `RooStats` applying this constraint is very simple to do. During the model definition the original model, now called `premodel`, is multiplied by the Gaussian PDF constraint `gauss` using `RooProdPdf`:

```
RooGaussian gauss("gauss", "gauss",
  lambda, meanlambda, sigmalambda);
RooProdPdf model("model",
  "premodel*gauss", RooArgList(premodel,
  gauss));
```

We wanted to apply a tight constraint on λ, so we looked at the $\hat{\lambda}$ distribution from Figure 5 and chose $\sigma_\lambda = 0.01$. The original $\hat{\lambda}$ distribution had an RMS of 0.037, so providing a tighter constraint with an RMS of 0.01 should show improved results compared with no constraint.

The estimator distributions with the constraint applied are shown in Figure 6, and the values obtained for the parameters are given in Table 4. The distribution of $\hat{\lambda}$ now has an RMS near 0.01, and the error has a much smaller RMS compared with no constraint. This is reflected in Table 4; with the constraint, the value for $\hat{\mu}$ has decreased to 0.102 from 0.135. p_0 has also increased to 0.108 from 0.067, signifying that the data is more compatible with the null hypothesis with the constraint included. This is as we expected, since the experimental dataset was generated with $\mu = 0$ and $\lambda = 1$,

Quantity	`RooFit`	`RooStats`
$\hat{\mu}$	0.102 ± 0.087	0.102 ± 0.087
$\hat{\mu_b}$	0.990 ± 0.032	0.990 ± 0.032
$\hat{\lambda}$	0.998 ± 0.010	0.998 ± 0.010
$\hat{\hat{\mu_b}}$ $(\mu = 0)$	1.000 ± 0.032	1.000 ± 0.032
$\hat{\hat{\lambda}}$ $(\mu = 0)$	0.999 ± 0.010	0.999 ± 0.010
q_0^{exp}	1.527	1.527
p_0	0.108	0.1000 ± 0.001
p_0^{th}	0.108	0.108
Z	1.24σ	1.28σ
μ^{upper}	0.267	0.271
$\mu_{\text{th}}^{\text{upper}}$	0.260	0.260

Table 4: `RooFit` and `RooStats` result comparisons after including a Gaussian constraint on λ.

and now we are constraining λ closer to 1 by reducing the RMS. Now the model is in better agreement with the data, so we expect a larger p-value. We also see that μ^{upper} has improved, decreasing from ~ 0.31 to closer to 0.27.

`RooFit` and `RooStats` handle including constraints differently. Once the modified model has been defined, `RooStats` knows exactly how to handle everything automatically. In `RooFit` the toys must be generated manually, and the correct values of λ must be used in order to obtain the correct $\hat{\lambda}$ distribution. In `RooFit`, for each toy, the value of λ is drawn from the Gaussian distribution (with mean $\hat{\hat{\lambda}}$ and standard deviation σ_λ), and is set to $\tilde{\mu}_\lambda$. This can be thought of as a separate experiment that is done simultaneously. The mean of the Gaussian that it is drawn from is set to $\hat{\hat{\lambda}}$.

```
// generate lambda according to the
// Gaussian constraint
double toyLambda = rdm->Gaus(1.0, 0.01);
// set mean of constraint to toyLambda
meanlambda.setVal(toyLambda);
```

Please note that in the `RooFit` code provided, the mean is set to 1, but it should actually be set to the value of $\hat{\hat{\lambda}}$, as in Table 4. The toys are then generated using this value for $\hat{\lambda}$. These steps are necessary in `RooFit` in order to generate toys that follow an unbiased Gaussian $\hat{\lambda}$ distribution [7], [8].

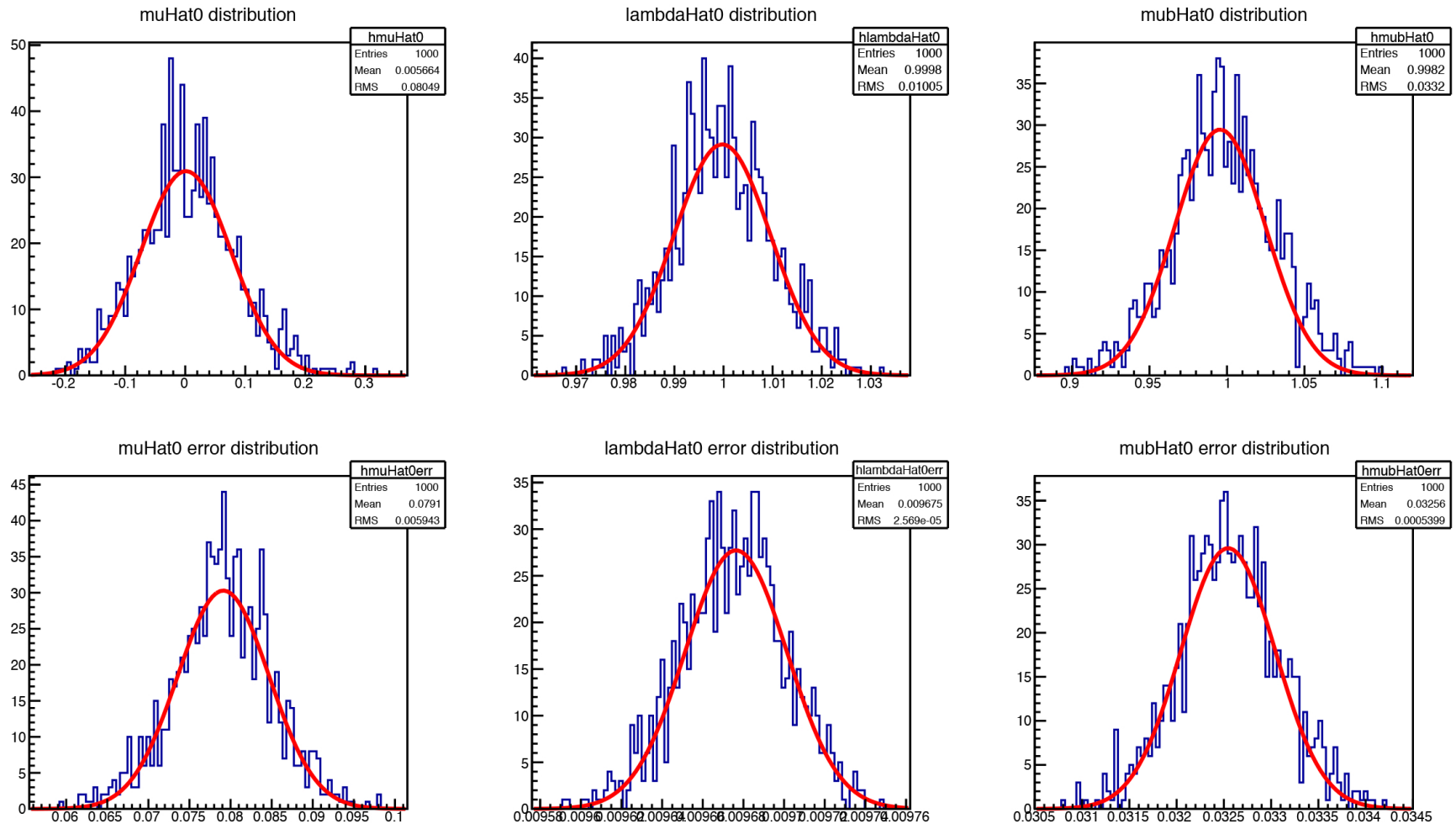

Figure 6: $\hat{\mu}$, $\hat{\lambda}$, and $\hat{\mu_b}$ distributions obtained from `RooFit` with a $G(1, 0.01)$ constraint on λ ($\mu = 0$ for p_0 calculation), and their errors.

5 Conclusions

This analysis was successful in using both `RooFit` and `RooStats` to calculate two likelihood-based test statistics for an experimental dataset using a toy model. Using `RooFit`, it was easy to tweak every step in the program and do checks to ensure that the analysis was being done correctly. At the same time, programming these two tests was time consuming and there were many complicated details to consider. Using `RooStats` was much easier in terms of the amount of programming required, but the details used in its complex classes are mostly hidden from the user. For this reason, learning the methods for these analyses would have been very difficult if `RooStats` was used before `RooFit`. However, once the user was comfortable enough using `RooStats`, then one could really appreciate and trust the functions that it already has in place. Overall, using both toolkits was very useful in learning about the methods of likelihood-based tests in ATLAS.

Part II
Exploring `TMVA`

6 Introduction

In ATLAS we are often interested in the number of signal events present in an experimental dataset. Each event has variables $\{x_1, x_2, ...\}$ associated with it, for example energy of a given particle, etc. From these variables it is possible to distinguish between signal and background events, and the difficulty in doing so depends on the differences in signal and background distributions. If the signal particles have some energy, and background particles have very different energies, then it will be easier to distinguish between them. But what happens when the signal events have very similar variable distributions compared to background? Then it becomes difficult to detect the differences. This is where multivariate analysis (MVA) techniques become useful.

MVA techniques are used to help distinguish between signal and background events by using input variables of each event. First, a training sample dataset is used to "train" the classifier. There are several different types of MVA methods, each with its own pros and cons; they all use different techniques to determine the classification of an event as signal or background. Once the training is done, tests can be performed to test for *overtraining* and to verify how efficient the classifier is at identifying signal and background events. `TMVA` is a `ROOT` package that does these tests automatically, and makes it easy to compare the training results of several different classifiers at once. After the classifier is trained, then it can be used to actually *classify* "real" events.

In this report the focus will be on the training of the classifiers and evaluating how well each worked at distinguishing between signal and background; I will discuss implementing two different kinds of methods in `TMVA`: *Fisher discriminants* and *boosted decision trees*. The dataset analyzed in this study are Monte Carlo events from a $H \to WW \to e\nu\mu\nu$ study with two sources of background and four input variables. The details will be discussed in the Analysis section.

7 Theory

7.1 Fisher discriminants

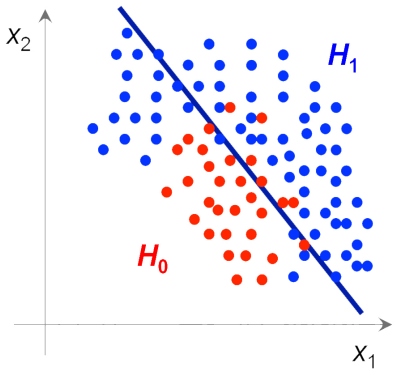

Figure 7: Classification using linear discrimination.

Figure 7 shows an example of *linear discriminant analysis*. If H_0 are signal events and H_1 are background events, the discriminator determines a linear combination of the variables x_1 and x_2 in order to provide the best discrimination between signal and background.

The TMVA User's Guide [10] describes the details of Fisher discriminants (as well as boosted decision trees). Here I give the general concept of Fisher discriminants as given in the guide:

> "In the method of Fisher discriminants event selection is performed in a transformed variable space with zero linear correlations, by distinguishing the mean values of the signal and background distributions. The linear discriminant analysis determines an axis in the (correlated) hyperspace of the input variables such that, when projecting the output classes (signal and background) upon this axis, they are pushed as far as possible away from each other, while events of a same class are confined in a close vicinity."

The discriminant itself is calculated using what are known as *Fisher coefficients*:

$$F_k = \frac{\sqrt{N_S N_B}}{N_S + N_B} \sum_{\ell=1}^{n_{\text{var}}} W_{k\ell}^{-1} \left(\bar{x}_{S,\ell} - \bar{x}_{B,\ell}\right) \tag{25}$$

Here, $N_{S(B)}$ are the number of signal (background) events, $\bar{x}_{S(B),\ell}$ is the class-specific sample means for signal (background) of the ℓth input variable, and $W_{k\ell}$ is the *within-class matrix*:

$$\begin{aligned} W_{k\ell} &= \sum_{U=S,B} \langle x_{U,k} - \bar{x}_{U,k} \rangle \langle x_{U,\ell} - \bar{x}_{U,\ell} \rangle \\ &= C_{S,k\ell} + C_{B,k\ell} \end{aligned} \tag{26}$$

The Fisher coefficients are then used to calculate the *Fisher discriminant*

$$y_{Fi}(i) = F_0 + \sum_{k=1}^{n_{\text{var}}} F_k x_k(i). \tag{27}$$

F_0 is an offset which centres the sample mean $\bar{y}_{Fi}$ of all $N_s + N_b$ events to zero.

Fisher discrimination is the first method studied in this analysis. Overall Fisher discriminants have very good qualities; they are quick to train, are not susceptible to overtraining, and are robust even when weak input variables are included in the training. Their major downfall is that they have difficulties discriminating variables that have non-linear relationships.

7.2 Boosted decision trees

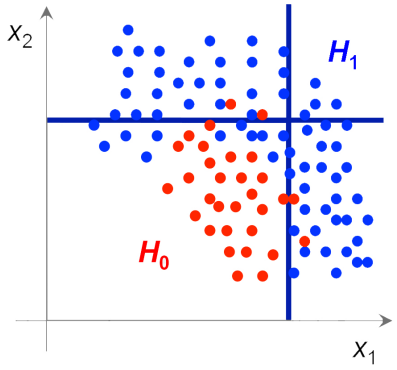

Figure 8: Classification using cut-based discrimination.

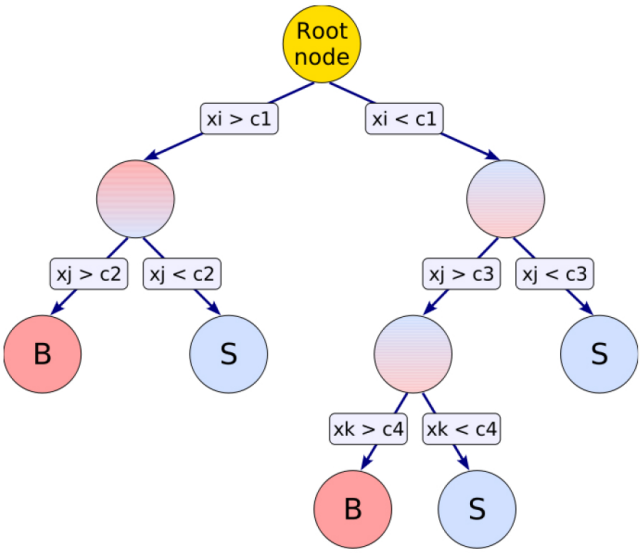

Figure 9: An example of a boosted decision tree [10]

Figure 8 shows an example of how a boosted decision tree (BDT) decides to discriminate between signal and background events using cuts on input variables. This is typically the simplest way to provide discrimination between signal and background events. For example, if an event has a particle with energy below a certain threshold, classify the event as signal.

The actual application is more complicated, however, but it is much more effective; each variable cut forms a branch on a tree of decisions, and they depend on the preceding cuts (see Figure 9). At each

node the BDT makes a cut to provide the best discrimination between signal and background. The BDT runs through a series of cuts before classifying an event as signal or background; these are called leaf nodes. During the training process, once the tree has classified the events in the training sample, then those events are reweighted according to which events were misclassified. Then the reweighted events are run through another tree and it makes optimal cuts again. This is then repeated for a forest of trees, until the trees are combined into a single classifier.

BDTs are popular in ATLAS. Unlike Fisher discriminants, they are able to perform well with nonlinear relationships between input variables, which is a huge advantage. However they do take longer to train, and they are susceptible to overtraining. But there are tests in place to monitor this. BDTs typically only train on half of the training sample. After training, the other half of the dataset, called the test sample, is used to test for overtraining.

8 Analysis and Results

The signal events for this analysis are Higgs produced via gluon-gluon fusion, followed by the decay $H \to WW \to e\nu\mu\nu$. There are several different backgrounds for this process; the two used to train classifiers in this study are $gg \to WW \to e\nu\mu\nu$ and $qq \to WW \to e\nu\mu\nu$.

8.1 Input variables

Figure 10 shows the four input variables used in this analysis for the signal and the $qq \to WW$ background. The first plot which shows the distribution of `MT_TrackHWW_Clj` is the *transverse mass* of the system. The invariant mass of the Higgs particle cannot be reconstructed due to the two undetected neutrinos in the final state. The transverse mass is defined as follows:

$$\begin{aligned} m_T &= \sqrt{E_{T,\ell\ell+\nu\nu}^2 - |\vec{p}_{T,\ell\ell+\nu\nu}|^2} \\ &= \sqrt{\left(E_{T,\ell\ell} + E_T^{\text{miss}}\right)^2 - |\vec{p}_{T,\ell\ell} + \vec{p}_T^{\text{miss}}|^2} \end{aligned} \tag{28}$$

Here E_T and p_T are the transverse energy and transverse momentum of the quantity ($\ell\ell$, $\ell\ell + \nu\nu$, or missing). In addition, $E_T^{\text{miss}} = |\vec{p}_T^{\text{miss}}|$, where $\vec{p}_T^{\text{miss}} = -\sum \vec{p}_T$ for all measured particles.

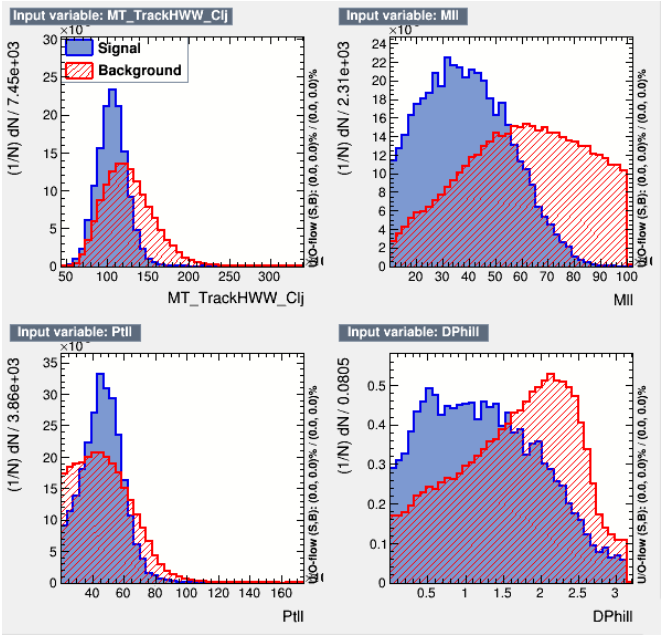

Figure 10: Input variable distributions for signal and qq background.

The other three variables have simpler definitions. $m_{\ell\ell}$ is the invariant mass of the two leptons, $p_{T,\ell\ell}$ is the transverse momentum of the two leptons, and $\Delta\phi_{\ell\ell}$ is the azimuthal angle between the two leptons.

These four variables provide good discrimination between signal and background events. The discrimination is best when the input variables have different distributions between signal and background events. In analyses such as Higgs parity studies, the signal and background variable distributions are not distinguishable by eye. It was shown for this analysis that m_T provides the most discrimination out of these four variables. This was done by analyzing the ROC curve after removing each variable from the training (more on ROC curves in the next section).

8.2 `TMVA` training output

After training, `TMVA` automatically produces the results needed to evaluate the training, including the correlation matrices for variables and the classifier response. Figure 11 shows the response of the

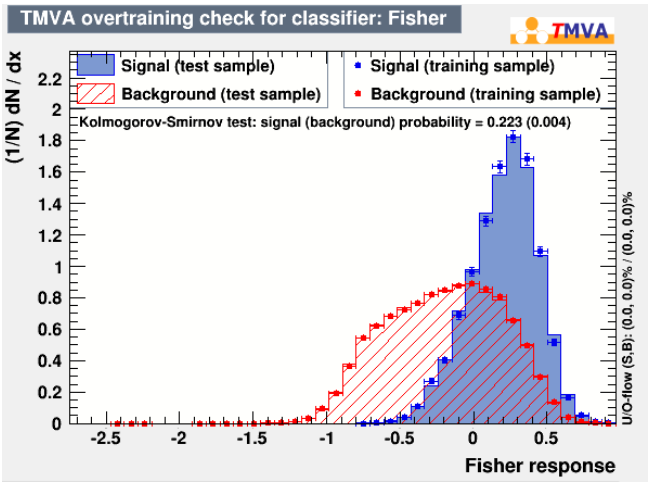

Figure 11: Fisher discriminant response.

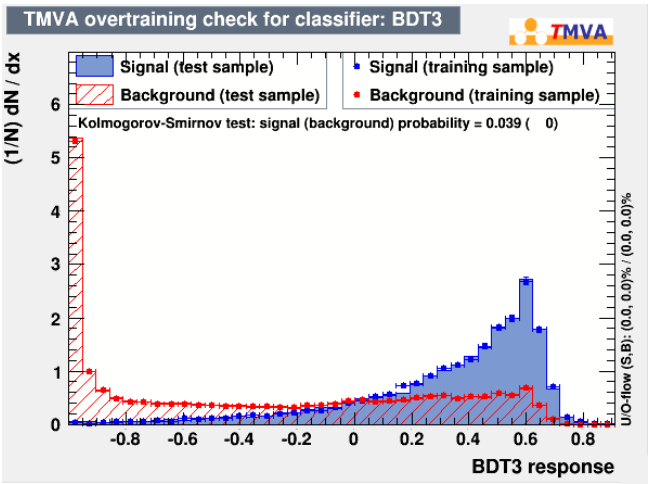

Figure 12: Boosted decision tree response.

Fisher classifier. The response is shown for signal and background events for both the training and test samples. The response itself is a variable that the Fisher classifier creates to identify how "signal-like" or "background-like" an event is. Events closer to -3 are more like background, and events closer to $+3$ are more like signal. The blue curve is generally higher, which is good because this is the signal distribution. The background distribution is more "background-like," as it should be. Unfortunately there is a lot of overlap between the two distributions, and there are not a lot of background events near -3.

Figure 12 shows the BDT response which ranges from -1 (background) to 1 (signal). There is a very

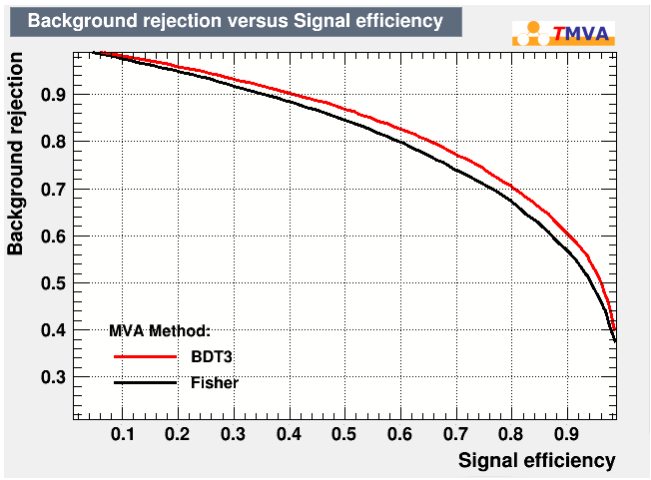

Figure 13: ROC curve comparing Fisher and BDT efficiencies.

clear difference between the signal and background distributions. Most of the background events are down near -1, which is ideal, and many signal events are higher between 0 and 1. Compared to the Fisher discriminator, the BDT performed much better when discriminating between signal and background.

Figure 13 clearly illustrates this difference. This is known as a ROC (receiver operating characteristic) curve, and it is a plot of (1− background efficiency) against signal efficiency. Ideally this curve would have a background rejection of 1 and a signal efficiency of 1 (this would give the maximum area of 1 under the ROC curve), but the two curves shown are more realistic cases of what they look like. The red curve corresponds to the BDT classifier, which has better background rejection at a given signal efficiency compared to the Fisher discriminant method. As expected, the BDT has better performance because it is able to discriminate well between variables that have non-linear relationships. This is the major downfall of using Fisher discriminants. It should also be noted that `TMVA` automatically overlays the ROC curves for several different MVA methods, giving a very useful way to monitor the performance for different methods.

`TMVA` also automatically displays the optimal BDT response cut in order to obtain the best possible signal significance (Figure 14). This same plot is also generated for the Fisher method. Using the `TMVA` GUI allows the user to input the number of

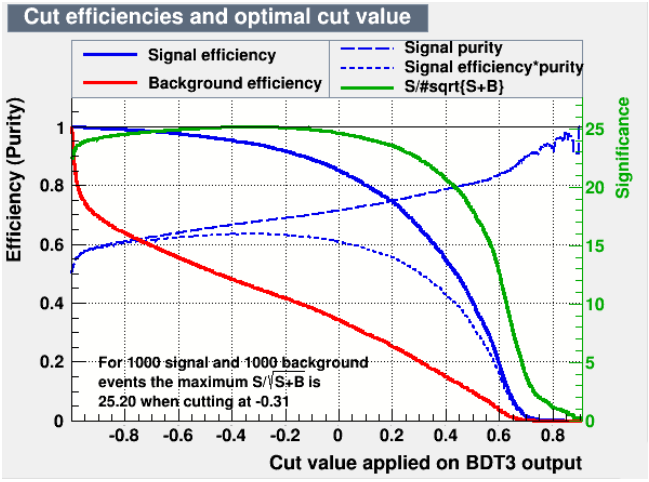

Figure 14: Optimal BDT output cut values.

signal and background events, and the plot will automatically update to accommodate the change.

9 Conclusions

This study provided a brief overview of some of the tools available in `TMVA`, and is certainly not meant as a complete guide. More detailed analyses have been done for $H \to WW \to e\nu\mu\nu$ using BDTs, including using them to *classify* events. Please see Manuela Venturi's PhD thesis [11] for a much more detailed analysis.

Overall, `TMVA` was a very useful tool to train and study the responses of several MVA techniques at once. There is an entire suite of discriminator methods available in `TMVA`, each with its own pros and cons. It was extremely easy to manipulate variables, method booking options, etc., and observe the change in response. For any analysis that requires the use of MVA techniques, this kind of tool is simply invaluable.

10 Acknowledgements

Thank you to Michel Lefebvre for being an incredible, patient supervisor. His guidance and enthusiasm were incredibly motivating, and I am very grateful for having the opportunity to work with him this summer. My passion for particle physics is growing stronger every day, and I am very excited to continue on in ATLAS for my MSc.

References

[1] G. Cowan et al. (2013), arXiv:1007.1727v3 [physics.data-an].

[2] `RooFit` code (including dataset): `svn.cern.ch/reps/atlas-kmclean/ROOTExamples/RooFitToy/tags/RooFitToy-00-00-01`. `RooStats` code: `svn.cern.ch/reps/atlas-kmclean/ROOTExamples/RooStatsToy/tags/RooStatsToy-00-00-00`.

[3] The ATLAS Collaboration. Combined search for the Standard Model Higgs boson in pp collisions at s = 7 TeV with the ATLAS detector. arXiv:1207.0319 [hep-ex].

[4] S. S. Wilks, The large-sample distribution of the likelihood ratio for testing composite hypotheses, Ann. Math. Statist. 9 (1938) 60-2.

[5] A. Wald. Tests of Statistical Hypotheses Concerning Several Parameters When the Number of Observations is Large, Transactions of the American Mathematical Society, Vol. 54, No. 3 (Nov., 1943), pp. 426-482.

[6] `RooFit` documentation. Website, accessed May 2014, `http://roofit.sourceforge.net/`.

[7] L. Demortier and L. Lyons. Everything you always wanted to know about pulls, CDF note (2002) 43.

[8] T. Moritz Karbach, M. Schlupp. (2012), arXiv:1210.7141 [physics.data-an].

[9] `RooStats` TWiki page. Website, accessed June 2014, `https://twiki.cern.ch/twiki/bin/view/RooStats/WebHome`.

[10] `TMVA` User's Guide. PDF file, accessed July 2014, `http://tmva.sourceforge.net/docu/TMVAUsersGuide.pdf`.

[11] Manuela Venturi. Search for the Standard Model Higgs boson in the $H \to W^+W^- \to \ell^+\nu\ell^-\bar{\nu}$ final state with the ATLAS experiment and study of its spin and parity quantum numbers. PhD thesis, Albert Ludwigs University Freiburg, 2014.