RADBOUD UNIVERSITY NIJMEGEN

BACHELOR THESIS

Using multivariate analysis to improve boosted Higgs to $b\bar{b}$ mass resolution.

Author:
Thijs MIEDEMA

Supervisor:
Dr. Frank FILTHAUT
Veronica FABIANI

Institute for Mathematics, Astrophysics and Particle Physics

July 10, 2017

Radboud University Nijmegen

Abstract

Faculty of Science
Institute for Mathematics, Astrophysics and Particle Physics

Using multivariate analysis to improve boosted Higgs to $b\bar{b}$ mass resolution.

by Thijs MIEDEMA

We try to apply multivariate analysis to extract extra information from jet substructure. By training machine learning algorithms on the background process gluon to $b\bar{b}$ we construct a general purpose algorithm without imprinting the Higgs boson into it. By then evaluating on Higgs boson data we show that the conventional way of scoring a machine learning algorithm is unsuited for this purpose, resulting in shifting mass spectra.

Acknowledgements

I would like to thank my project supervisors Frank Filthaut and Veronica Fabiani for their everlasting patience with all my questions and tinkering with algorithms. I would also like to thank my housemates who provided me with my fuel (coffee), and especially Rutger who joined me in the Thesis writing crunch sessions.

Contents

Chapter 1

Introduction

1.1 Research question

The Higgs boson was discovered in 2012 by Atlas and CMS at CERN [1, 2], with a mass of 125 GeV. However, it was only observed in the $\gamma\gamma$, ZZ and WW decay channel (see Chapter 2). The most probable decay channel of the Higgs, the production of a bottom-antibottom pair, has not been measured due to the large background. However, with the increased center-of-mass energy the Large Hadron Collider, or LHC, now runs at it is more probable to produce an energetic Higgs boson, what we call a boosted Higgs (see Chapter 2 for more detail on this). The Higgs boson will show up as a single jet in the detector. Unlike the resolved channel, where to separate b-jets appear in your detector, this high energy jet is much less likely to be produced in background processes. The final problem, the jet mass reconstruction algorithm is not accurate enough to get a good resolution on the Higgs boson mass in this way. Also, there might be undiscovered particles that also should pop up in this channel. By improving the mass resolution on the Higgs boson we might get a better understanding of this channel and maybe see more in the higher mass regions as well.

But is it possible to use multivariate techniques to improve the mass resolution of this energetic Higgs?

1.2 Motivation

An enormous amount of data is produced at the LHC and we should try and use this data to the fullest extent possible. Using modern machine learning algorithms to extract more statistics out of existing data is cheap compared to building bigger collider experiments and can inspire advancements in both physics and data science.

1.3 Prior work

The boosted Higgs channel is an area of interest for many[3, 4]. Most focus on identification techniques, but multivariate techniques have only been applied on the resolved case. Also, these have almost exclusively been carried

out using TMVA, a machine learning software package developed for particle physics[5]. It is a great tool but lacks some of the recent improvements in the machine learning field. Exploring the possibilities of TensorFlow and Scikit-Learn and the tools developed to do so shows there might be merit in using open source software instead of sticking with the proprietary software developed inside of the physics community.

Chapter 2

The Standard Model

2.1 Particle processes

The standard model is a set of elementary particles and interactions that dictate how matter and forces work, see figure 2.1. At collider experiments all of them can be produced, but due to their short lifetime they usually decay before reaching the detector. We can see their decay products however. Because of the conservation of both momentum and energy, these decay products can give us information about the particle that decayed.

The specific particle we are interested in is the Higgs boson. The most recently discovered elementary particle can provide us ways to look beyond the standard model.

Decays and creations of particles in the standard model are best displayed in Feynman diagrams. These diagrams display the how incoming particles interact to form the final state. They are two-dimensional, where one axis represents space and one represents time. I will use the convention of displaying my time axis horizontally. A Feynman diagram can be turned into an equation. Solving this equation, often not a trivial exercise, yields a cross section (σ) or decay width(Γ)The difference lies in the process occurring: a cross section represents the probability of an interaction to occur while the decay with represents the probability of a single particle decaying into something else. The decay width is the sum of all widths of different decay modes, called partial widths. The ratio between a partial width and the decay width gives the probability of that particle decaying into that specific outcome. Cross sections can be compared in the same way. This predicts how often each process will happen relative to each other. For example, take these two decays of the W^- boson in figure 2.2.

The input of this diagram is obviously always the same, but the outcome isn't. Which one of the two happens is random, but the ratio of which can be estimated with theory and can be determined experimentally. For example, the branching ratio for the first diagram is $\Gamma(W^- \to l^- \nu_l)/\Gamma(Total) = 10.86 \pm 0.09\%$ [6].

But it doesn't stop there. The resultant particles of a decay might also be unstable and decay again. This iterative process of several chained decays might go on for a while. In a collision experiment this means you can detect sometimes see hundreds of particles coming from a single point. Especially processes that involve quarks and gluons, like the one we are interested in,

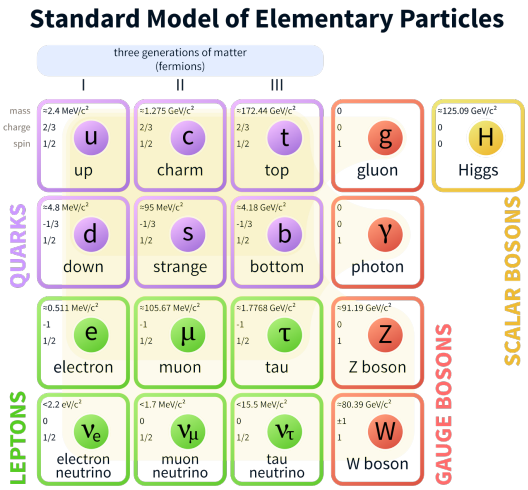


FIGURE 2.1: The particles in the Standard Model.

FIGURE 2.2: Two decay modes for the W^- boson. Here, l is an electron, muon or tau lepton and ν_l the associated neutrino. The $c\bar{s}$ could be any of the other quarks, excluding the top quark, while maintaining charge conservation (the top quark is too heavy to be produced).

give rise to showers of particles that result in a jet. This is due to the fact that they carry a colour charge. Colour charge is the charge involved in the strong force. This force is so strong relative to the other forces that colour charge cannot exist on its own. This is called confinement [7]. This means that quarks and gluons produced in collisions quickly collect other quarks and gluons around them so the total colour charge is zero or white again. This process results in mesons (quark-antiquark) or baryons(quark-quark-quark), together called hadrons, which are colour singlets (meaning they have zero total colour charge). This process is called hadronization [8].

Because of momentum conservation, all particles produced in a decay chain move in similar directions, especially when the original particle was very energetic. This results in a cone-shaped shower of particles in your detector. This is what we call a jet. A jet can have different shapes and sizes. A highly energetic particle (also referred to as a "hard particle") might make a relatively big jet. When reconstructing bigger jets we aptly name them Fat Jets.

Often a fat jet contains smaller cones inside because the original particle

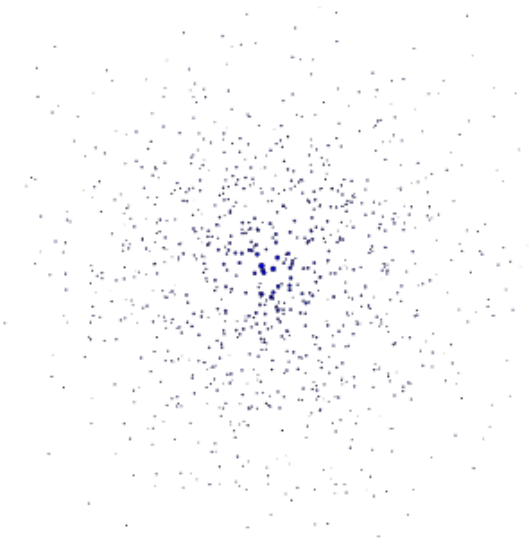


FIGURE 2.3: An impression of what a Jet could look like in a detector. Imagine the image as a square cut out of the cylindrical wrapper outside the collider. Each dot represents a particle impact, the size of the dots their energy.

decayed into two or more particles that still had a lot of energy and long decay chains. These are referred to as subjets. This is the origin of jet substructure and an important jet property to try to reconstruct.

But before we can identify jet substructures we need to identify the jets themselves. When you would view a single, clean event this would be an easy task for a human. Take for example 2.3. It is easy to spot the circular shape here, as a slice of a cone, but in reality this might be a lot more difficult. There could be several jets in one event and to make it worse several events are happening at once (pile-up), requiring to be separated afterward in software. Jets can be close together or even go through each other. Several algorithms have been developed to perform jet identification and solve these problems. To mention a few: CDFJetClu, MidPoint, SIScone, kt, Cambridge/Aachen and anti-kt. Several are still in use, usually for very specific problems except for the anti-kt algorithm, which is the default one for basically all jet reconstruction. kt, Cambridge/Aachen and anti-kt can all be derived from the following equation.

$$d_{ij} = min(p_{Ti}^{2\beta}, p_{Tj}^{2\beta}) \frac{\Delta_{ij}^2}{R^2} \tag{2.1}$$

$$d_{iB} = p_{Ti}^{2\beta} \tag{2.2}$$

Here, β determines the algorithm: -1 for anti-kt, 0 for Cambridge/Aachen and 1 for kt. d_{ij} is a distance measure between particles, while d_{iB} is a distance measure between a particle and the beam. The rest are explained in the ATLAS design paper in section 1.1, quoted here:

"The nominal interaction point is defined as the origin of the coordinate system, while the beam direction defines the z-axis and the x-y plane is transverse to the beam direction. The positive x-axis is defined as pointing from the interaction point to the centre of the LHC ring and the positive y-axis is defined as pointing upwards. The side-A of the detector is defined as that with positive z and side-C is that with negative z. The azimuthal angle ϕ is measured as usual around the beam axis, and the polar angle θ is the angle from the beam axis. The pseudorapidity is defined as $\eta = \ln\tan(\theta/2)$ (in the case of massive objects such as jets, the rapidity $y = 1/2\ln[(E+p_z)/(E-p_z)]$ is used). The transverse momentum p_T , the transverse energy E_T , and the missing transverse energy E^T_{miss} are defined in the x-y plane unless stated otherwise. The distance ΔR in the pseudorapidity-azimuthal angle space is defined as $R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$."[9]

Based on these distances d you can cluster particles together. You take the smallest distance, if that is a d_{iB} you call i a jet and remove it from the list of entities. If it is a d_{ij} you combine i and j into one entity i and remove j. You continue combining until everything is removed as a jet candidate.

2.2 Higgs

With the existence in the Higgs boson confirmed by Atlas and CMS in 2012 [1, 2] with a mass of 125GeV we now know that one of the main decay modes for the Higgs boson should be to a bottom-antibottom pair, or $h^0 \rightarrow b\bar{b}$ for short (see figure 2.4, $\Gamma(H- > bb)$ 57%). Evidence for this process exists[10] but not enough to accurately determine that branching ratio experimentally [11]. This is because this decay channel is not as clean as for example $\gamma\gamma$. The b and $\bar{b}$ both hadronize to form colour singlets, as explained earlier this chapter. Those hadrons decay and result in a jet. The energy and momentum of a jet are a much more complex problem than that of a photon. The hadrons bottom quark decays into an up or charm quark via the weak interaction. This decay is quite slow when compared to all the strong interactions, resulting in a displaced vertex in the jet, a new origin of particles outside the main interaction point. The displacement is in the order of millimeters, but with the precision of the Atlas detector this is detectable. The main way of identifying these b-jets is looking for B-mesons in the decay products or displaced charm-mesons. This whole process of identification is referred to as b-tagging. The current Atlas b-tagging efficiency is about 70% and going up with higher energies, determined by Monte-Carlo simulation of the whole detector.

With the b-tagging and the knowledge that the Higgs boson decays to $b\bar{b}$-pairs often we could look for events that have two jets with a btag relatively close together, with maybe a jet going the opposite way to cancel out the momentum, illustrated in figure 2.5. However, these events occur plenty out of normal Quantum Chromodynamic(QCD) processes. This background will completely drown out the signal we are looking for. Therefore it is more

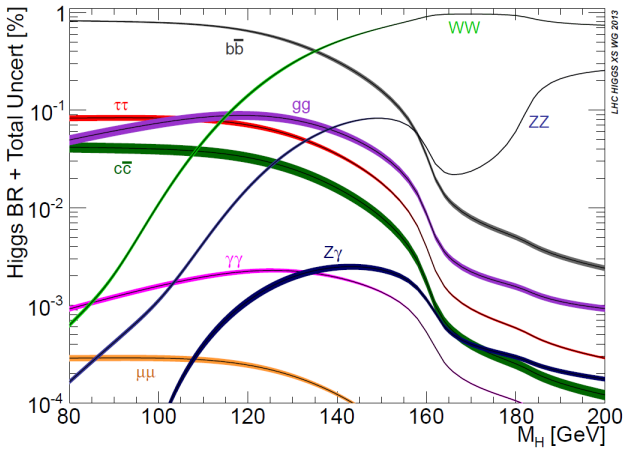


FIGURE 2.4: The Higgs boson decay modes as a function of the Higgs boson mass, which we now know to be $\approx 125GeV$ [12].

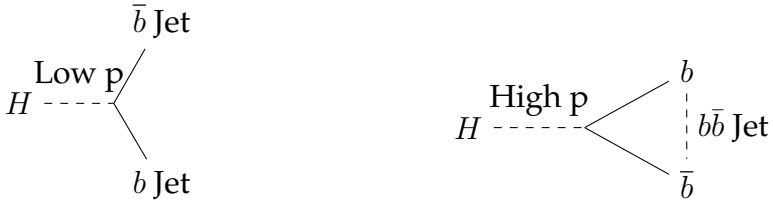


FIGURE 2.5: $h^0 \rightarrow b\bar{b}$ with two separate two jets, the resolved case and $h^0 \rightarrow b\bar{b}$ in a single jet, the boosted case.

interesting though is the situation where the Higgs boson has so much momentum the bottom quarks are close enough together that we see only a single jet with both bottom quarks inside, because the distance between those quarks is related to the Higgs momentum $d_{b\bar{b}} \approx \frac{1}{\sqrt{z(1-z)}} \frac{2m}{p_T}$ (z and z-1 are the momentum fractions of the b and $\bar{b}$ respectively) [13]. These high momentum single jet channels are called boosted. The boosted variant of the $h^0 \rightarrow b\bar{b}$ channel in particular is much less contaminated with background processes since there is less chance to produce such high momentum bottom quarks so close together from normal background.

Chapter 3

Jet structure

3.1 Simple properties

Once a group of particles has been classified as a jet there are several properties that you can calculate. A few ones are obvious, such as the average direction of the jet, the total transverse momentum and energy, which can be calculated by summing over the total set of particles in the jet. A naive estimate for the invariant mass of the jet would be estimating the total momentum by taking the jet direction and transverse momentum and simply applying Pythagoras. Then we apply Einsteins formula

$$m^2 = E^2 - p^2 \tag{3.1}$$

3.2 Missing neutrinos

Of course, this is not the full picture. The jet can be 'contaminated' by other particles that are not part of the jet but end up inside the jet area. Also, energy and transverse momentum are lost due to the usual problem in particle physics: all neutrinos will be missing from your data. Various techniques exist to make an attempt to fix these. In general you try to identify the muons, electrons and taus involved in the jet and argue that if they are not produced in a lepton-antilepton pair there should be an associated neutrino. You then look at the muon momentum and direction and the missing transverse momentum and energy in the event and make some estimate on the momentum of the missing neutrino.

3.3 Jet Substructure

However, there are yet more challenges. When dealing with a fat jet it is not just a blob with an energy, momentum and mass, it has an inner structure. It is, in some cases, quite easy for a human to put the jets into groups. When you look at 3.1 it is easy to imagine that the left one is probably caused by a single particle, while the left one were two particles close together. But we have millions upon millions of jet events from ATLAS, so we need to write some algorithm to do this for us. The first step is to somehow "clean" the Fat Jet. By trying to remove particles that originate from another interaction not

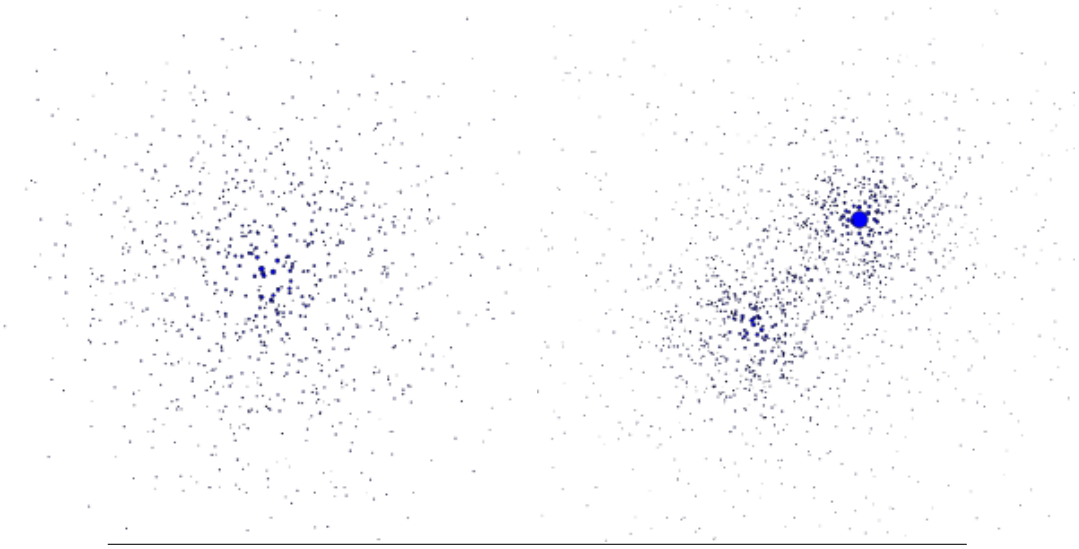


FIGURE 3.1: An impression of an event that has a jet with one and a jet with two substructures.

connected to the jet and maybe leaving out insignificant particles, especially those with lower energy you get something that is much easier to handle in processing in a later stage. Several of these algorithms exist. To name a few: jet grooming, jet trimming[14], jet pruning[15], softdrop and more. After these procedures you can run the normal jet finding algorithms again on the contents of the jet.

3.3.1 Jet trimming example

The following is a simplified example of the steps involved when using jet trimming. First, the anti-kt algorithm is used, but with R=1 instead is R=0.4. This is because we are working in the regime of boosted jets instead. We want to use this on fat jets with substructure, not resolved cases. We then redo the jet finding with anti-kt but now with R=0.2. We discard any subjets we have now find with a to low p_T (5% of the p_T of the original fat jet). See figure 3.2 for an example of how that would look if used on our fatjets in figure 3.1.

3.3.2 Jet substructure variables

We now have some data on the interior structure of the jet, but nowhere near perfect. Subjets might overlap and thus be classified as a single subjet. The whole process of jet trimming and rerunning jet finding is also very computationally expensive. We also have no measure of confidence in the data. We either have one, two, etc subjets, and if it is 'ambiguous' that doesn't show in our final data. Take the following image for example:

As a human you might say 'Unsure' or '1.5?' but we haven't given the jet trimming algorithm the possibility to make that choice. A new algorithm was published in 2011, the so-called 'N-subjettiness inclusive jet shape', or τ_N [16]. It tries to define some sort of probability that a certain jet has the

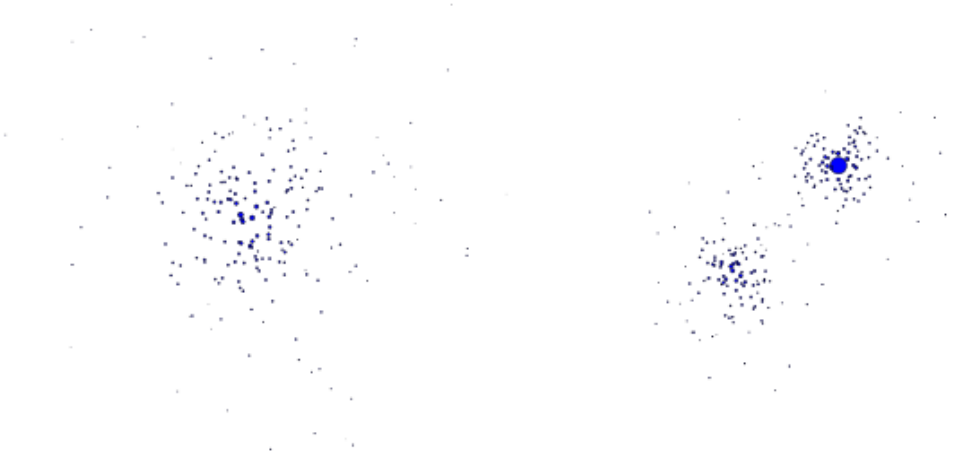


FIGURE 3.2: Jet trimming applied to the jets from 3.1.

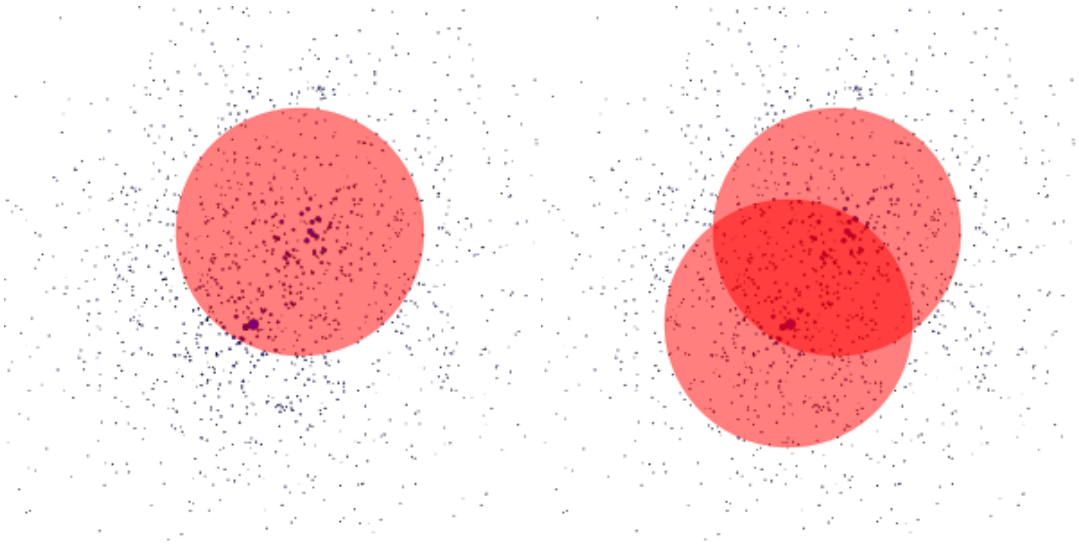


FIGURE 3.3: Sometimes jet classification can be ambiguous. Here, the same structure could be said to be one jet, or two.

specified number N of subjets.

$$\tau_N = \frac{1}{d_0} \sum_i^K p_{T,i} min \left\{ \Delta R_{1,i}, \Delta R_{2,i}, \cdots, \Delta R_{i-1,i}, \Delta R_{i+1,i}, \cdots \Delta R_{N,i} \right\} \tag{3.2}$$

Here i sums over the total K particles, $\Delta R_{J,p}$ denotes the 'rapidity-azimuth' (see chapter 2) distance between candidate subjet J and particle p, d_0 is a normalization factor. When for example there are two obvious subjets τ_1 will always be large, since wherever we place the candidate subjet, the contribution of one or both of the subjets will be substantial. However, with τ_2, you can put the candidate subjet centers in the obvious places and all contributions will be fairly small. So the N for which $\tau_N < \tau_{N-1}$ is probably the real amount of subjets.

Therefore, we define the N-subjettiness ratios, $\frac{\tau_2}{\tau_1}$, shortened to τ_{21}, and also τ_{32} and so on. These variables are now normalized, so it is easy to compare them. A τ_{21} that is below 0.5 with a τ_{32} that is 0.7 is a pretty strong indicator that the jet you are looking at has two substructures.

Since the time N-subjettiness was introduced there has been more work on improving this. In 2013 and 2014 respectively the C2 and D2 variables were published [17, 18]. They are both based on the Energy Correlator Double Ratio functions:

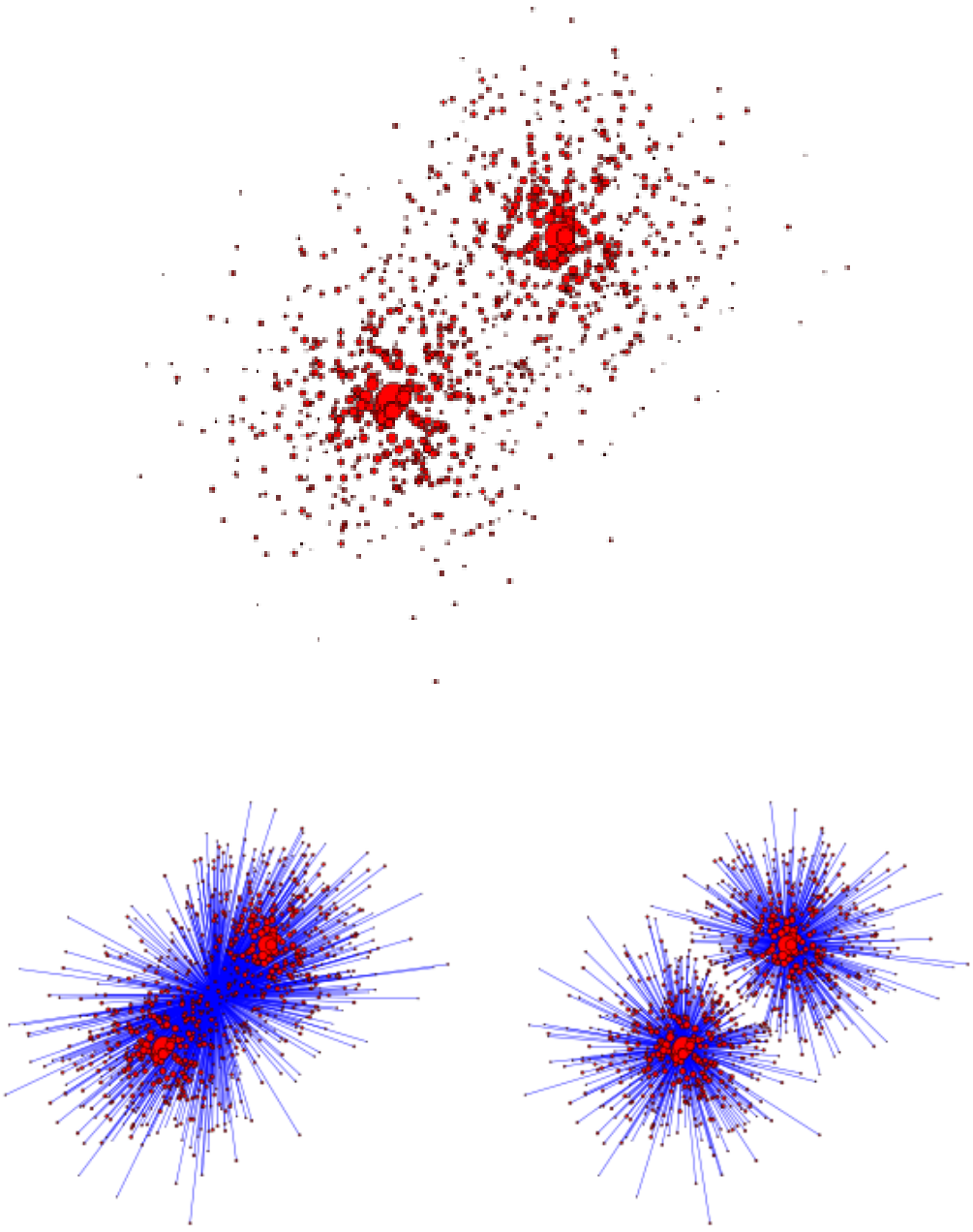
$$ECF(N,\beta) = \sum_{i_1<i_2<i_3\cdots<i_N \in J} \left(\prod_{a=1}^{N} E_{i_a} \right) \left(\prod_{b=1}^{N-1} \prod_{c=b+1}^{N} \theta_{i_b i_c} \right)^{\beta} \tag{3.3}$$

These functions are so general and versatile that combining them in different ways you can get a lot of different information out of them. The computational complexity is exponential in N, so most applications stick to N=3 as the maximum N used. C2 is constructed as follows:

$$C_2^{(\beta)} = \frac{ECF(3,\beta)ECF(1,\beta)}{ECF(2,\beta)^2} \tag{3.4}$$

C_2 should be sensitive to jets with two subjets, or two-pronged jets. The parameter β can be used to focus on a specific type of jet. It controls, roughly, how important the outer regions of the jet cone are compared to the center. These are referenced to as collinear and wide-angle respectively.

The D2 variable is again trying to improve on the work done with the C2 variable. The research done by Andrew J. Larkoski, Ian Moult, and Duff Neill identified some problems with the C2 variable, mainly that it is very sensitive to pile-up. Almost every time protons collide in ATLAS, a beam crossing, in the order of 25 interactions occur [19]. The data from each interaction must be separated afterwards to be able to use that data for analysis. Errors in this separation process can mean that particles from one interaction event end up in another. These pile-up data contamination's are a serious factor to consider in your analysis.



(A) τ_1 (B) τ_2

FIGURE 3.4: Mock calculation of the N-subjettiness variables. They should be calculated by multiplying the size of the dot times the length of the line. This is obviously smaller for the second one, while a third one will yield a much smaller improvement.

For the definition of D@ we first define the normalized Energy Corelation Double Ration function:

$$e_n^{(\beta)} = \frac{ECF(n,\beta)}{(ECF(1,\beta))^n} \tag{3.5}$$

The authors argue that using this leaves the result without unit, allowing fairer comparison between the functions. Now we can define D2 as:

$$D_2^{(\beta)} = \frac{e_3^{(\beta)}}{\left(e_3^{(\beta)}\right)^3} \tag{3.6}$$

Chapter 4

Event simulation

4.1 Motivation

The mass reconstruction of a jet is a sophisticated algorithm that tries to take into account all the particles in the jet and the uncertainties in their energy and momentum measurements and missing neutrinos. However, it is still developed and tweaked by physicists, who cannot cover the total amount of data that jets can contain. The substructure variables might contain hints as to improve our mass estimates, we just need to find out how to get those "hidden statistics" out of there. That is where machine learning comes in. However, in order to train these algorithms, we need to feed them loads of events where we already know what the actual mass is. This is where we turn to simulating events instead of using real data. Here we can control exactly which processes are happening and can determine what the mass of jets is in truth. Luckily several tools have been developed to achieve exactly this.

4.2 Software

The first step is a program called MadGraph5_aMCNLO[20], from here on just MadGraph. This piece of software simulates the low-level Feynman diagrams, from protons to Higgs boson to $b\bar{b}$ for example. This allows us to generate only those events we are interested in. But we have another problem. If we just generate Higgs boson to $b\bar{b}$ the truth mass will always be the Higgs boson mass. This means that training a machine learning algorithms on it is difficult, since those are always based on finding some kind of error minimum, and always giving the answer 125GeV is a sound way to do just that. Therefore we decided training on gluon to $b\bar{b}$ instead. This spectrum is smooth so training on it will disallow“cheating” by the machine learning algorithm. So we ask MadGraph to generate two kinds of samples, gluon to $b\bar{b}$and Higgs boson to $b\bar{b}$. We will use one for training and the other one for testing. Listed in figure 4.1 are some of the Feynman diagrams MadGraph generates. The thing to note is that there is always a $b\bar{b}$-pair in the output.

But we aren't there yet. We know bottom quarks must decay quickly (in order of millimeters [21]) and fall apart into jets. This part of the process is covered by PYTHIA[22]. It takes the particles generated by MadGraph and selects the unstable particles. It then simulates a decay chain and makes

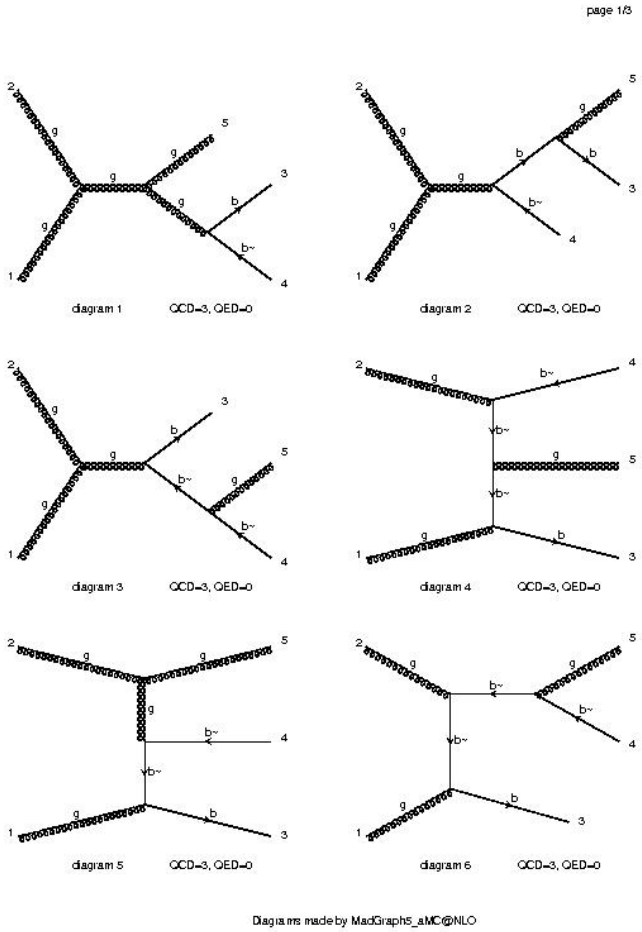


FIGURE 4.1: Feynman diagrams generated by MadGraph.

sure the final state contains no coloured particles as explained in chapter 2[1]. The output from PYTHIA has the stable particles from MadGraph and the mesons, hadrons and leptons it produced.

The next and final step is simulating the ATLAS detector itself. After PYTHIA we know the exact momenta of every single particle. However, in the real world we don't see neutrinos, we have uncertainties on how fast muons are traveling. Our energy estimations of photons, mesons, hadrons might be off. Maybe we think something is a muon but is an electron. Etcetera, etcetera, etcetera. Luckily there is software available that does all of this for us. Delphes is a detector simulation framework, taking the output from PYTHIA and outputting the event as if an actual collider experiment had recorded it [25]. It can simulate various detectors, CMS, ATLAS, or even collider experiments that still have to be built[26]. You can run Delphes fairly fast on a normal computer, especially considering it has to simulate millions of particles interacting with a huge detector. The trick is that it doesn't actually do that by doing some serious corner cutting. Instead of simulating the full detector it takes the accuracy results from the actual full and slow detector simulation ATLAS does internally and applies these to your simulated data.

[1] Achieved by using a Lund string technique, see [23, 24]

Chapter 5

Simulation results

5.1 Data spectra

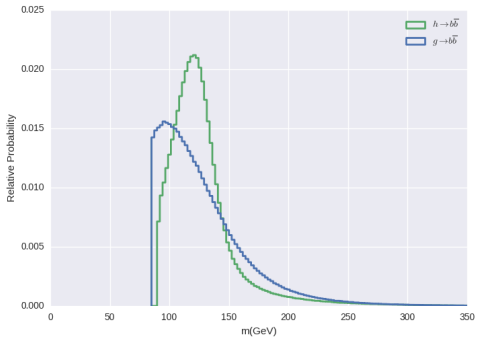
Once we have obtained several million event records that contain $b\bar{b}$-jets we need to check if they are suitable and calculate the necessary training parameters. That means finding the $b\bar{b}$-pair in our event record, checking if they are contained into the same jet and not reconstructed into two separate jets. Then we also need to add their four-momenta to calculate their invariant mass. This is the truth variable we want to train on after all. A check for NaN values is needed because machine learning libraries don't play well with those, a single NaN in your data might propagate to all your weights in the algorithm. This means any training you did prior to that that is unsaved will be lost. We also need to calculate the substructure variables, τ_{21}, τ_{32}, C_2, D_2. All of this is done with a custom piece of software, for more details see A.

We can now also plot spectra of our input variables. This is an essential step in the whole process, since this is the point where you find out if the event generation actually produced correct events. We executed about 25 runs of data generation before these spectra were satisfactory. Some key points can also be made as a conclusion from these plots:

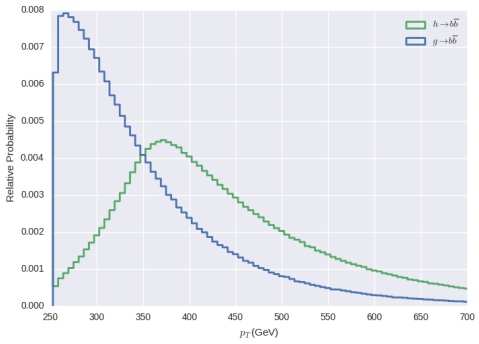
Gluon to $b\bar{b}$ jets fundamentally differ from Higgs to $b\bar{b}$ jets. The difference is visible best in the substructure variables. In general, the variables associated with two substructures peak lower, which can be interpreted as a clearer separation in the two substructures in a Higgs jet then in a gluon jet. This probably has as a root cause that the Higgs boson and gluon to $b\bar{b}$ decays are governed by two totally different processes. All of this is guesswork at this point and involve quantum field theory calculations far beyond my current understanding, but by looking at the changes in the τ_{21} spectrum as a function of invariant mass and transverse momentum this can be viewed directly. Low energy and almost on-shell gluons tend to have a higher τ_{21}, indicating that the two substructures are less clear.

5.2 Input and Output format

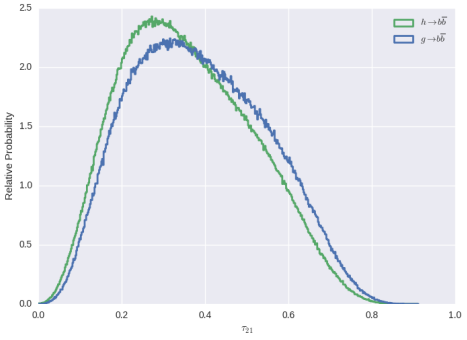
With a custom tool, see appendix A, we can transform event records from Delphes to a vector of floats for each event, containing all the information we need. The input is shaped like this:



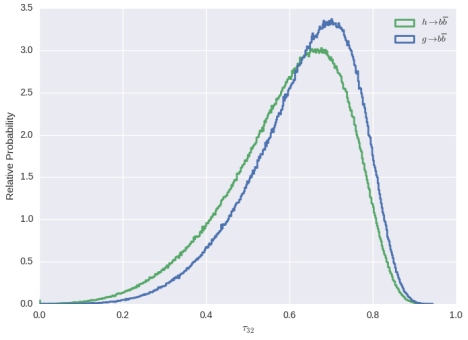
(A) Jet reconstructed mass



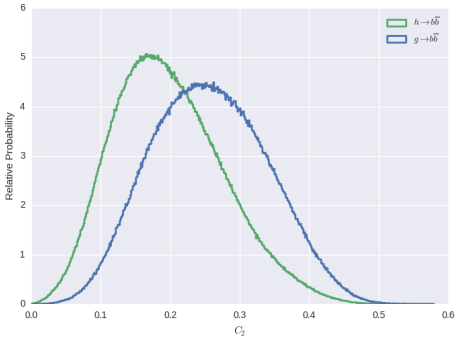
(B) Jet transverse momentum p_T



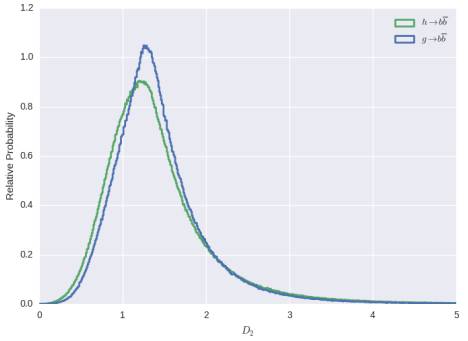
(C) Jet substructure variable τ_{21}



(D) Jet substructure variable τ_{32}



(E) Jet substructure variable C_2



(F) Jet substructure variable D_2

FIGURE 5.1: Data spectra of the final simulation.

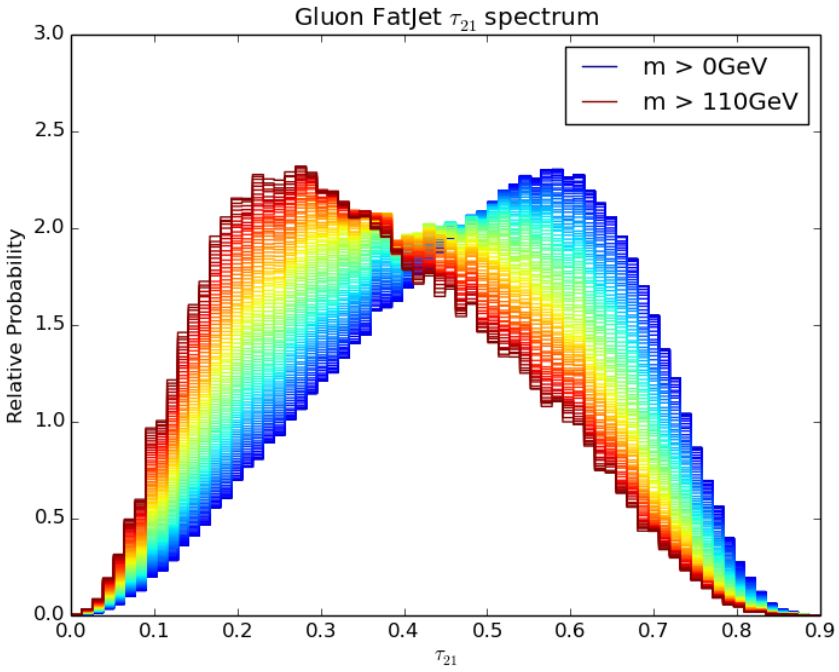


FIGURE 5.2: The τ_{21} distribution as a function of the cut on invariant mass. This shows that jets originating from more off-shell gluons behave more like two-pronged jets then (almost) on-shell gluons.

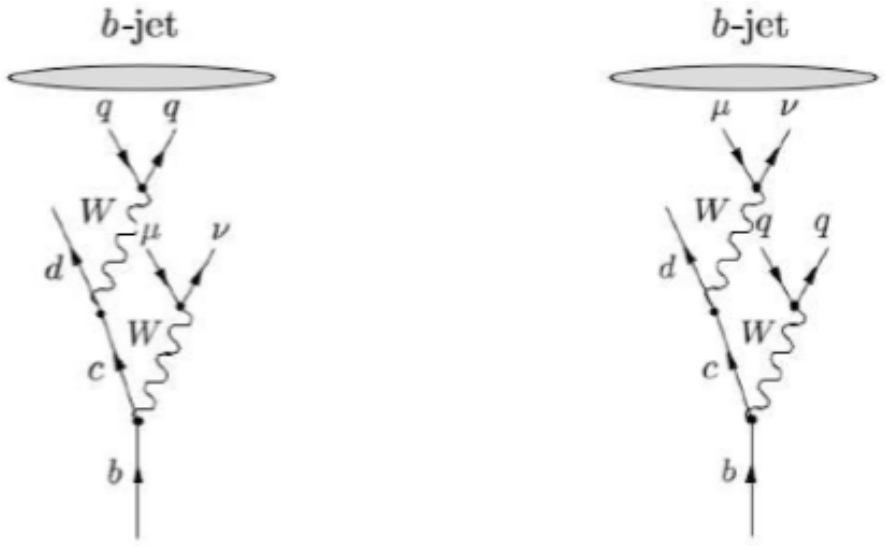


FIGURE 5.3: The semileptonic decay of the b giving rise to muons associated with the jets.[27]

$$\begin{aligned}\vec{x} \equiv \{&\eta, \phi, m_{reconstructed}, p_T, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_{21}, \tau_{32}, C_2, D_2, \\ &E_T^{miss}, \eta_{E_T^{miss}}, \phi_{E_T^{miss}}, \Delta R(muon, jet), p_T^{muon}, E^{muon}, q^{muon}\}\end{aligned} \tag{5.1}$$

You see we include one muon in the input data. This is the closest muon to the jet axis. Muons could be of importance since they are direct evidence that we are missing some momentum and energy due to the missing neutrino. They get produced when the b quark or resulting c quark decays semileptonic, as displayed in figure 5.3. We hope that the trained algorithms pick up on this. We defined three variables for our algorithms to train on:

$$\begin{aligned} y_1 &\equiv m_{truth} \\ y_2 &\equiv m_{truth} - m_{reconstructed} \\ y_3 &\equiv \frac{m_{truth}}{m_{reconstructed}} - 1 \end{aligned} \tag{5.2}$$

We refer to these three as Direct, Adding and Scaled respectively. The reason for training on the difference or ratio is motivated by the fact that we make cuts on generator and reconstruction level, which results in hard boundaries in the data. These hard boundaries get picked up my the algorithms very easily and results in artifacts in your output spectrum. We don't want the algorithms to make hard cuts when these are used on real-world data, so we hide the hard cuts with this technique. The added benefit is that the output is semi-normalized, which makes the algorithms converge more quickly when training.

Chapter 6

Machine Learning

6.1 Algorithms

There are roughly two types of machine learning problems: classification and regression. Classification deals with problems that require datapoints to be assigned to different groups, or classes as they're called, hence the name. It can be classifying wines to be red or white based on chemical properties[28], which is called one-to-one or binary classification. There is also multiclass classification or open set recognition, where the total number of classes might be unknown or input might not be part of any class at all. This happens with, for example, written character or face recognition. Regression on the other hand does not work with classes but predicts some continuous variable. To extend our previous examples: it could be used to predict the wine alcohol content from the rest of the chemical properties or predict the age of someone based off a picture.

There are more algorithms then I can list here. They are all designed to either tackle a very specific type of problem, or work on a very broad spectrum of problems. I will highlight a few that are the starting point for almost any problem and can help directing you to the right algorithm, or might be already good enough to satisfy your requirements.

6.1.1 Principal Component Analysis (PCA)

We start off by cheating and mentioning an algorithm that is not machine learning at all. It is, however, the starting point of almost any machine learning exercise. PCA is a way of dimensionality reduction of your problem. Even though your problem might have a lot of input variables, they can be strongly correlated and thus you could express your data in fewer variables. PCA does this by performing a sort of eigenvalue decomposition of the input. The input data is of course not a square matrix so that isn't possible, but it can be approached by so-called singular value decomposition. This results in a set of vectors that have the property that they are ordered in terms of variance. The first vector gets the largest variance, while the last one has the least. This might be hard to visualize, but figure 6.1 should make it immediately clear.

So even though your problem might have 10-dimensional input, the first three PCA dimensions will most of the time already capture a lot of what is

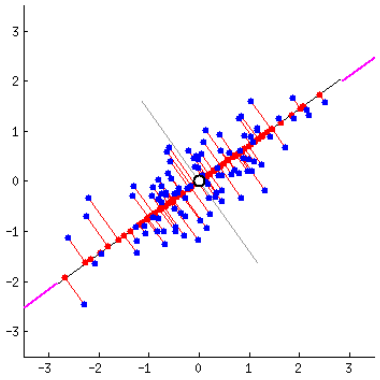


FIGURE 6.1: A 2D to 1D PCA projection. By maximizing the variance in the first dimension and minimizing in the second you get an optimal way to represent your data in less dimensions. This generalizes to an arbitrary amount of dimensions. [29, 30]

happening in your data. The plot in figure 6.2 was made very early into the internship, trying to figure out what was wrong with the $g \rightarrow b\bar{b}$ matching algorithm. It immediately shows that there is a certain type of datapoints that the algorithm does not handle well. This provided a clue to my as to what direction a solution should be sought.

6.1.2 K-means

A very basic classifier for exploratory work or simple problems [31]. There are a good number of extensions on this algorithm, but the basic idea stays the same: each class has a mean datapoint, called a centroid. This centroid should be the best representation of a class. For every new datapoint added you find the closest centroid and assign the datapoint that class. Then you update the centroid accordingly to reflect the addition of the new datapoint. "Closest" is of course an empty term without context: in normal k-means you use euclidean distance, normalized to the variance in each input variable. Some derived algorithms change this, for example the K-medians[32] or Minkowski-weighted k-means[33] algorithms. There are several problems with K-means, the first is that it is very computationally expensive to run on large datasets, since the algorithms complexity scales exponentially with the number of datapoints. The second is that it is highly sensitive to initial conditions and local minima. It can, therefore, be beneficial to run the algorithm several times, randomizing initial conditions for the centroids each time. But then we are back at problem one. Again, several derived algorithms attempt to fix this problem, but discussing those is outside the scope of this document.

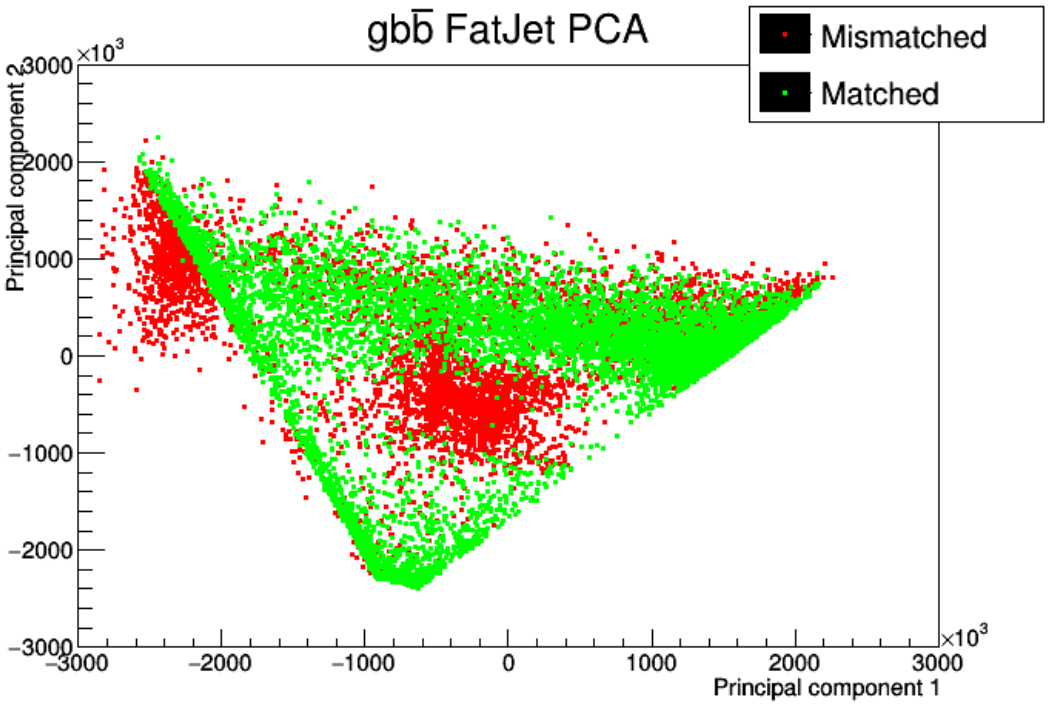


FIGURE 6.2: A plot from the early days of the internship. This is a plot of the first two principal components of my FatJet variables. The axis have no direct units, but the in the analysis later on it turned out that the red blot in the middle, which is a class of jets my algorithm was unable to match to $g \to b\bar{b}$-jets are resolved $b\bar{b}$ cases (see chapter 2.

6.1.3 Naive Bayes

This classifier is based on the idea is that every class has a mean and deviation for each continuous input variable, and you can calculate the chance that a new datapoint belongs to a certain class. This algorithm can be trained in linear times by means of Bayesian interference, and is still in use everywhere, even though it was invented in the 1950s. A nice side effect of using Bayesian statistics is that you don't calculate the class of a datapoint, but the chances to belong to each of the classes. This can give a measure of how sure your algorithm is of the classification, a very useful feature.

6.1.4 Logistic regression

In it's simplest form logistic regression is used in cases where the 'answer' of a datapoint should be yes or no, with a certain level of confidence. It is a type of linear model, where a result is expressed as a number between 0 and 1 that represents the chance of the answer being 'yes'. One of the most used examples is predicting whether a student passed his exam based on his hours of study. The training data can be a survey among students inquiring on the time they spent on studying and whether they passed. Imagining a second input variable is not hard, for example the mark they got for the homework exercises could provide serious improvements on the performance of the algorithm.

6.1.5 Linear regression

This is the most basic for of a regression model. It generalizes logistical regression, but the answers are not bound to the [0,1] interval. Take your datapoint, multiply with the weights, and take the sum, and you have your answer. This algorithm and many others are trained by using gradient descent [34]. Gradient descent is a technique where you take all of your weights and slightly change them, thereby sampling the "weight-space", resulting in some sort of numerical derivative. You then take a small step in direction of this derivative, by which you should slowly descent to the point in the weight-space that minimizes the error.

The logistical regression algorithm can again be generalized to multiple outputs by making the weights also multidimensional. Now make a step to minimizing the error function by numerically approximating the gradient with the current dataset. Several methods exist to speed this up, often using momentum techniques. The algorithm keeps track of a speed, so if the gradient stays steep in one direction for several steps the algorithm 'speeds up', taking bigger steps towards the minimum.

6.1.6 Multilayer perceptron

The essence of a neural network. 'Neurons' are organized in layers, taking some inputs from the previous layer, adding some bias, applying a nonlinear

so-called 'activation function' and then outputting to the next layer. The layers are organized as one input layer, then one or more hidden layers and an output layer. These are also trained by gradient descent, albeit more complex ones are often employed, see for example the AdamOptimizer [35].

6.1.7 Recurrent neural network (RNN)

RNN's are used in situations where there is a sequence of datapoints that all have the same structure but the amount is unknown. You feed the network your sequence and after you fed it all the data of that event you 'ask' the network for an answer. The network has 'memory', modifying its own internal state after each datapoint. A more advanced version of this is the Long Short-Term Memory network model. Explaining it is beyond the scope of this document, but it excels at analyzing time series and can even be used to compose music.

6.1.8 Wide-and-deep neural network

Wide and deep networks are a relatively recent development in the field. It really isn't a groundbreaking idea, it just combines a linear or logistic regressor with a deep (≥ 2 hidden layers) neural net, operating them side-by-side, combining their results. Although this does not sound revolutionary, it turns out this works very well.

6.1.9 Decision trees

A very aptly named algorithm. A decision tree is a set of nodes in a tree shape. You start at a specified starting node. Each node contains a "question", for example "if variable x is bigger than 10, go to node 3, else, go to node 4". You go through the nodes until you reach an end node that gives the output for your target variable. They can also be boosted. This boosting has no relation to jets and refers to the practice of emphasizing training data the algorithm previously got wrong, by assigning a bigger weight to that particular datapoint.

6.1.10 Random Forests

Used for both regression and classification, this algorithm is based on the decision tree. By training a lot of trees with some randomness in each tree, as the name might suggest, this algorithm averages over all those trees. This is to prevent overtraining and local minima.

6.2 Tools

In the world of machine learning there are a ton of tools that can be used to build, train and evaluate algorithms. We opted for using TensorFlow and

Scikit-Learn. TensorFlow is being developed by Google it is gaining traction, amounting for about 3000 papers since its introduction to the world. It is supposedly heavily used by Google itself and we see great improvements by them in the machine learning area. One of the great advantages of TensorFlow above almost any other library is the fact that you can write your network in a very abstract form in Python, giving you great freedom in what you want to do. But due to the fact that the network you write is compiled with C++, it is also very fast to train. There are also a few tools included, TensorBoard and the TensorFlow Debugger, that can give you insight into the training itself, visualizing what might be wrong.

The other framework used Scikit-Learn. Older, but actively developed, it is the most used machine learning Python library. It has tons of ready-made algorithms. You have less freedom than with TensorFlow, but for that you get an easy way to try a lot of different thing in a relatively short amount of time.

6.3 Algorithm performance

All machine learning algorithm have one thing in common. If given two versions of an algorithm, you should be able to calculate which one is better. If this were not the case a human would be needed every step of training, telling the computer which version did best. In classification problems we have a few options, but they all result from the question: how many datapoints did the algorith classify correctly, how many did it get wrong. In regression the choice is also seemingly obvious, you just sum all the errors of the algorithm, and maybe normalize to the amount of datapoints for easy comparison. This results in the definition of the root mean squared error:

$$\texttt{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(y_i - y_{target}\right)^2}{n}} \tag{6.1}$$

Chapter 7

Results

In tables 7.1, 7.2 the results for random forests are displayed and in 7.3 and 7.4 the results for a linear regressor. They each include six different ways of training: on mass directly, the difference with the reconstructed jet mass and the ratio with the reconstructed jet mass, as mentioned in Chapter 6. The results are then split in a weighed and unweighted training set. The weighed set has factors attached to each event with regards to their truth mass, so that the resulting weighed mass spectrum is constant over the 70-270 GeV region. The algorithms are trained with two million events and evaluated with one million events. During both stages the algorithm is scored with the root mean squared error.

Immediately something becomes clear. Even though the RMSE of our regressions have always improved over the jet reconstruction, our spectra, the place were we can "extract" the actual physics, has not. All Higgs boson peaks get offset. So, even though the quality of individual points has improved, our spectrum has not. This pattern persists with all tested algorithms (Bayesian, AdaBoost, Decision tree, Extratrees, Wide-and-deep). When looking back at the original question, is it possible multivariate techniques to improve on this resolution by including jet substructure variables into our analysis? We can conclude that there is a critical flaw in the way machine learning algorithms are scored. We have a different goal that most machine learning problems, that try to optimise their performance on individual cases and give no value to the correctness of the spectrum. A machine learning algorithm trained with the RMSE has a tendency to err towards the mean. This is not a weird thing. If asked "I have a random number between 1 and 100, guess as close as possible." anyone with sense would answer "50". The same thing happens here as with machine learning algorithms. Sure, you have optimized your performance on the individual events, but your spectrum is utterly destroyed. Of course here you are guessing a random number, while in our case the input data gives some clue to the output. But you can easily visualize the behaviour of the RMSE way of training as function of the correlation between the input and output, as shown in figures 7.5 and 7.6.

But apart from the RMSE we have other problems. The gluon to $b\bar{b}$ is fundamentally different than Higgs boson to $b\bar{b}$, as explained in Chapter 5. In the end our training data probably still is unsuitable for the purpose of learning the Higgs boson to $b\bar{b}$ process. Of course getting the spectrum right would be trivial if you would just use the Higgs to $b\bar{b}$ data directly, as tried before, even in 2011 by another student [36]. The fundamental problem with

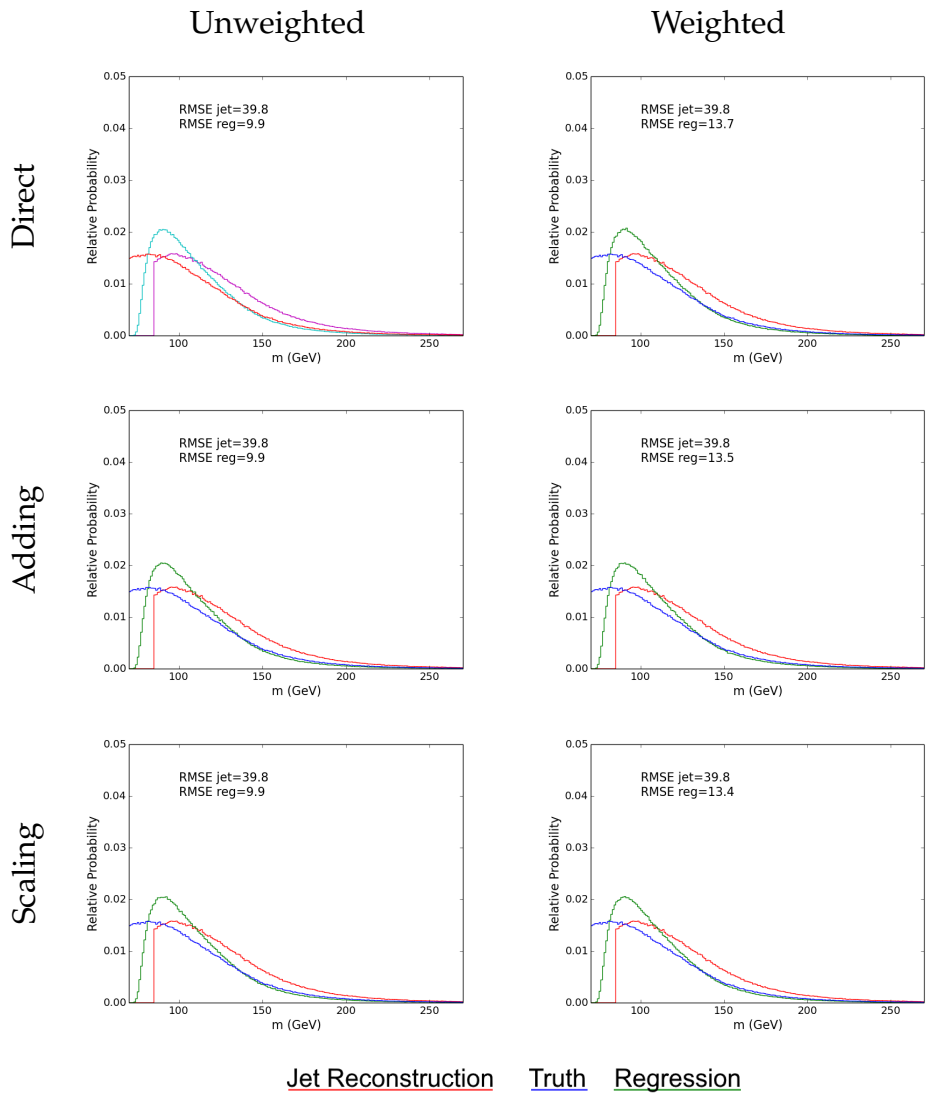


FIGURE 7.1: Results for random forests trained in different ways on gluon to $b\bar{b}$ data.

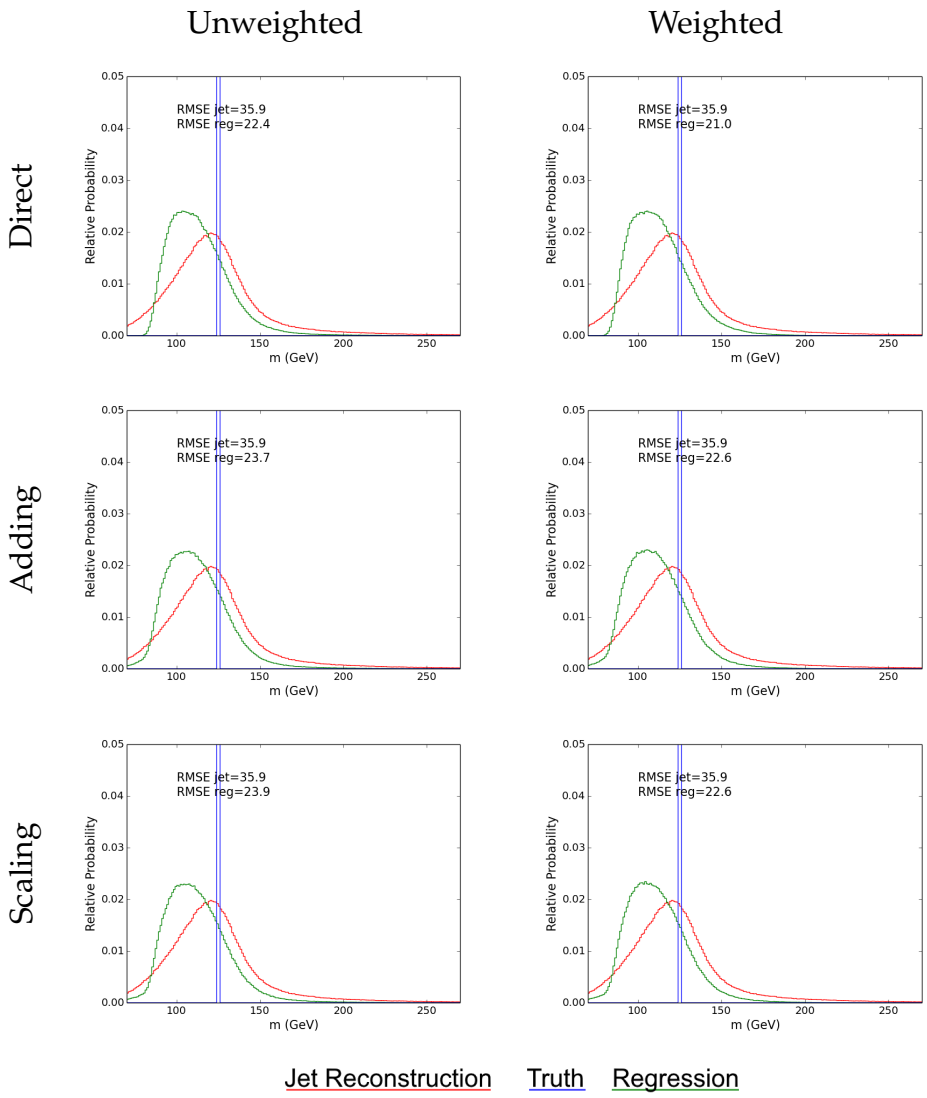


FIGURE 7.2: Results for random forests trained in different ways on Higgs boson to $b\bar{b}$ data.

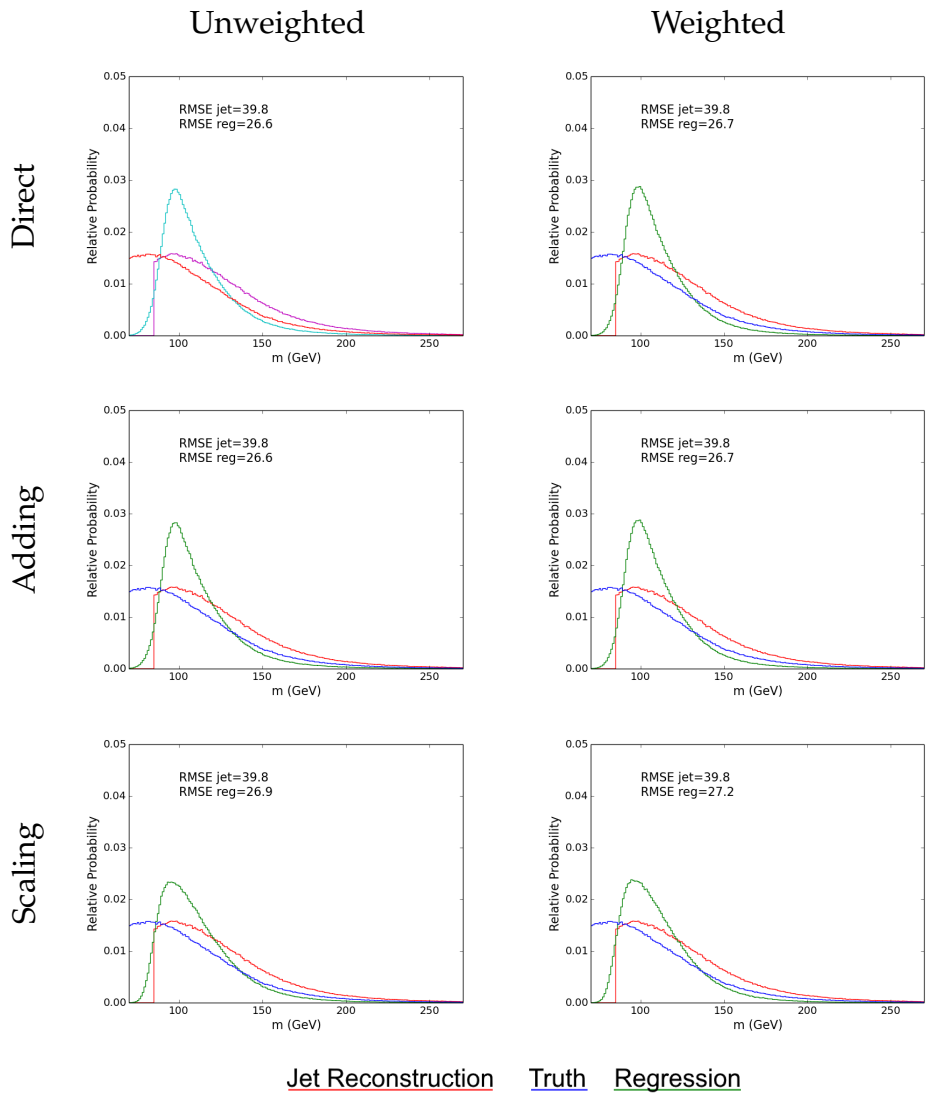


FIGURE 7.3: Results for linear regressors trained in different ways on gluon to $b\bar{b}$ data.

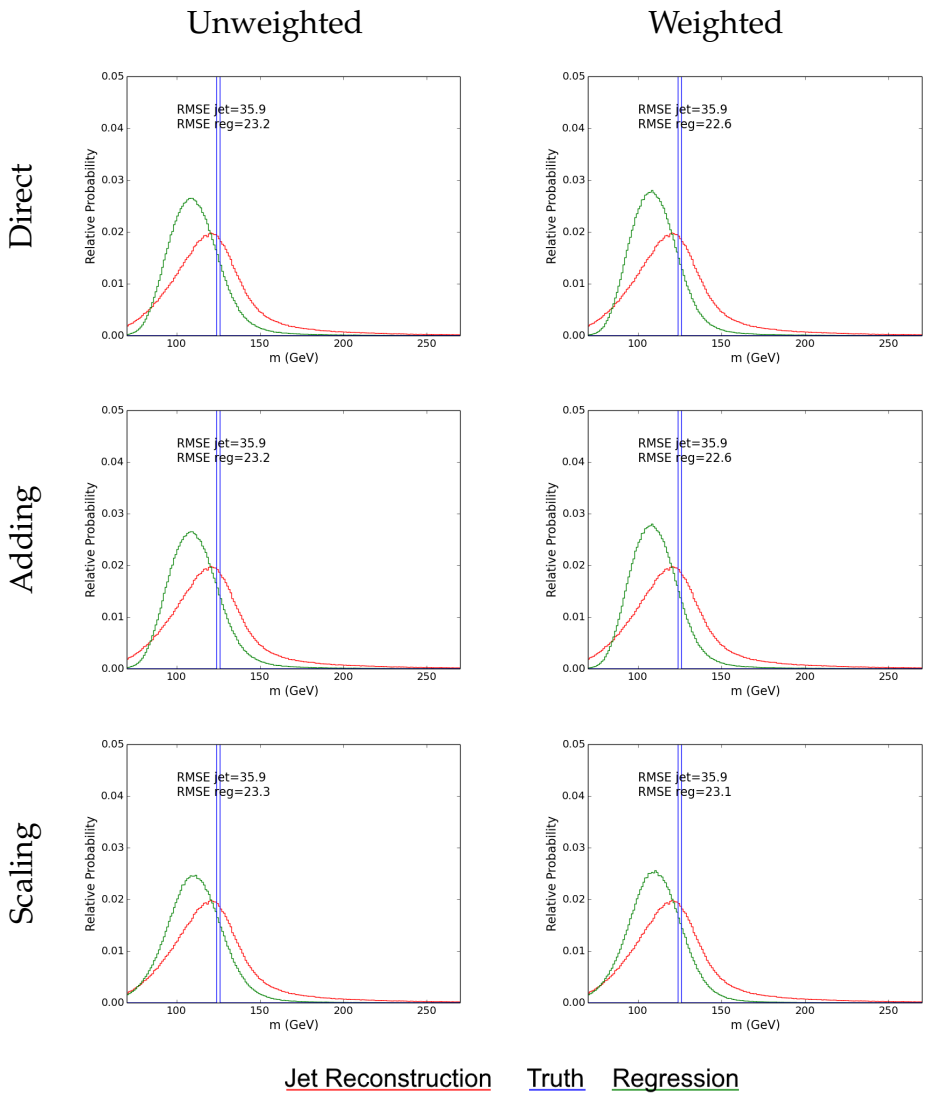
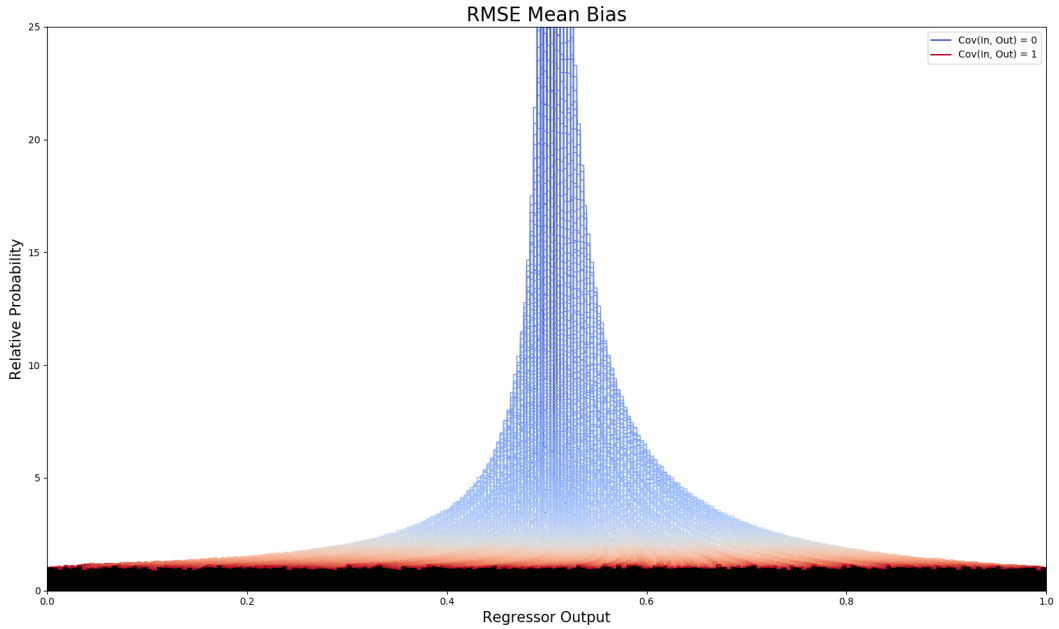
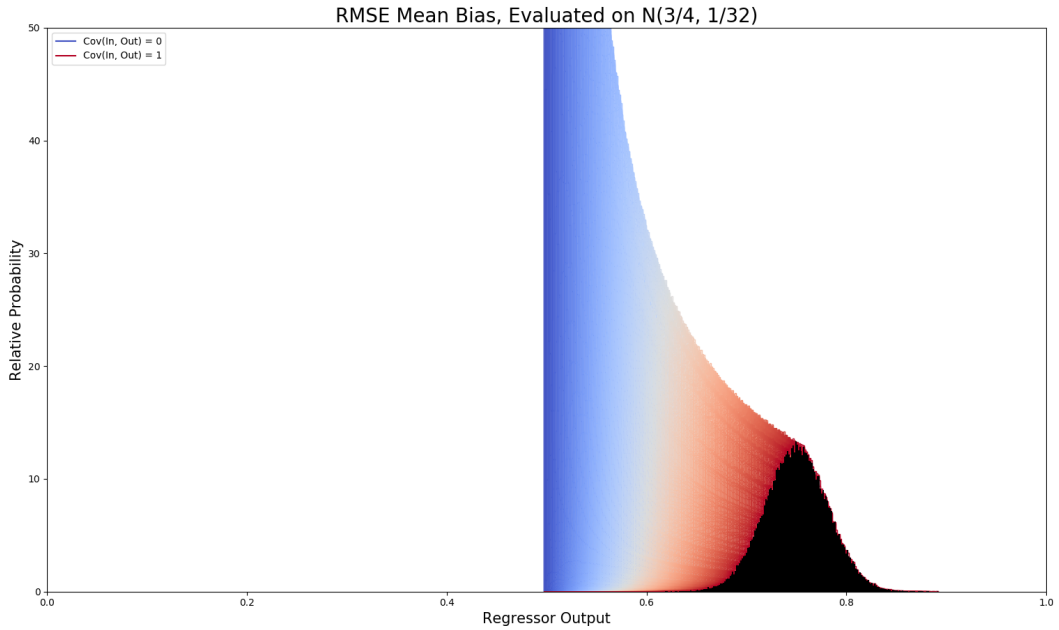


FIGURE 7.4: Results for linear regressors trained in different ways on Higgs boson to $b\bar{b}$ data.



(A) Output spectra of a trained linear regressor on a flat input spectrum, as function of the correlation between the input and output spectra.



(B) The already trained algorithms evaluated on a different spectrum with a peak, again as function of the correlation.

FIGURE 7.5: Visualisation of the RMSE mean bias.

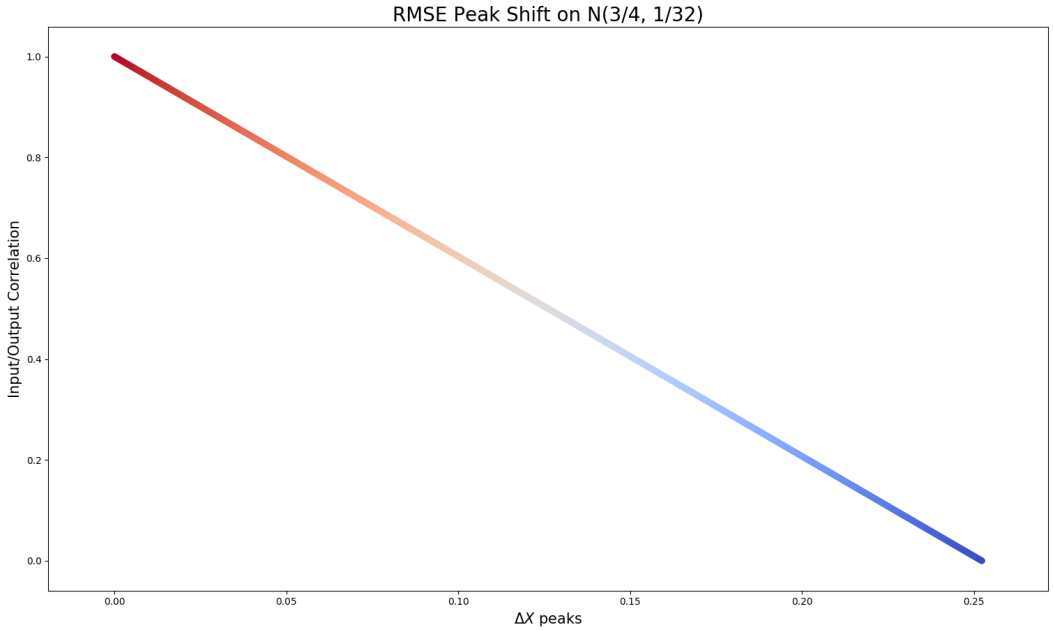


FIGURE 7.6: The recoreded peak shift as function of the correlation.

that is that that would mean that you would most likely see the Higgs boson even if it wasn't there and your algorithm most likely is not suited to find new particles in future data. So even though that would work, it doesn't tick the wish list box of being a general method that results in an algorithm that can be used on any data.

Another way to approach this problem would be to use the Z boson to $b\bar{b}$ process to train your algorithm and evaluate on Higgs boson to $b\bar{b}$. These are still different but at least there is no colour charge involved. Here you encounter the problem that Z is is a mass resonance again, altough not quite as narrow as the Higgs peak. Another, more involved way to go about this is modifying the way MadGraph and PYTHIA work themselves. If you generate events where the Higgs truth mass is a flat spectrum you get jets that behave as Higgs jets, obviously, but without encoding any preference for one mass or another into your algorithm. Another road you could travel is to try and exploit the fundamental differences found between Higgs and gluon to $b\bar{b}$ an train a classification algorithm to separate them.

Appendix A

Custom software

During the project I wrote several pieces of software that I deem usable outside of this projects scope.

A.1 TreeReBuild

A C++ ROOT Macro that transforms the Delphes output into a tree that is readable by root_mldata. This macro scans all the events for $b\bar{b}$-jets and stores those. There are three different ways to match $b\bar{b}$-pairs with a jet, depending on if jet constituent information is available or not. I performed an analysis on the matching performance by running them on all data and comparing their results.

A.1.1 Old style matching

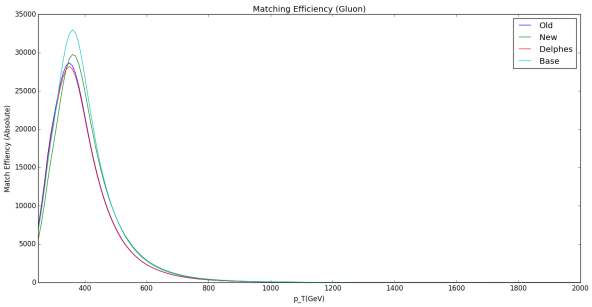
Developed in the early stages of the internship. Takes all the $b\bar{b}$ -pairs from the event and tries to match them to a jet by looking at angular separation. There are two versions of this, the older version does not require Delphes libraries to be loaded but does not output a nice rebuilt ROOT tree to use with root_mldata. This is the style used for all algorithm training and validation.

A.1.2 Delphes style matching

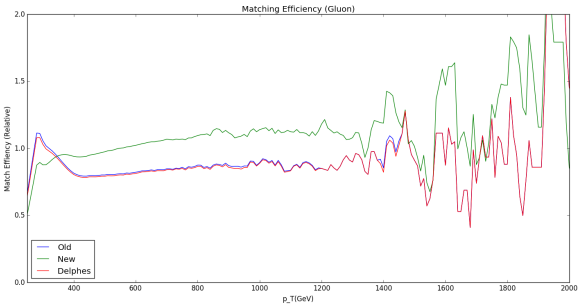
Inspired by the Delphes flavour association module, which is again based on internal Monte-Carlo tools, see `https://cp3.irmp.ucl.ac.be/projects/delphes/browser/git/modules/JetFlavorAssociation.cc`. Delphes style matching is almost the same in efficiency and output, but generally has a bit tighter criteria. This is because Delphes may drop the flavour association when the jet is too contaminated by other things happening in the event. However, events that are contaminated might be the area where the most to gain might be. It might be useful in other cases but not for our project.

A.1.3 New style matching

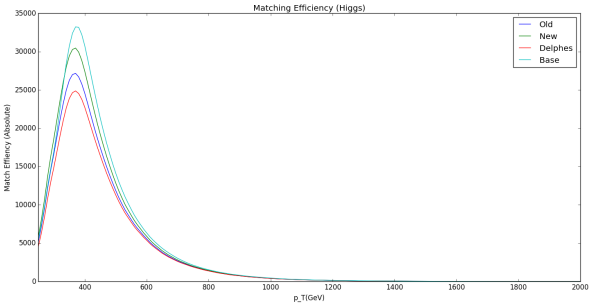
A small sidestep was made by trying to enumerate all the particles in a jet and performing a search through their parent particles on truth level. The



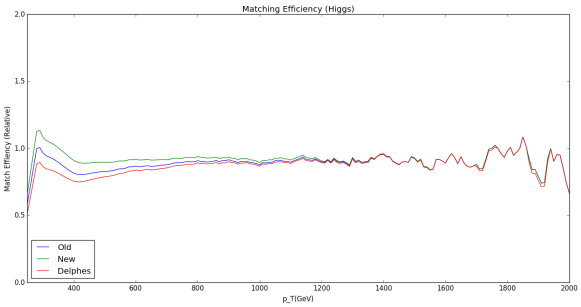
(A) Absolute $g \to b\bar{b}$ matching efficiency



(B) Relative $g \to b\bar{b}$ matching efficiency



(C) Absolute $h^0 \to b\bar{b}$ matching efficiency



(D) Relative $h^0 \to b\bar{b}$ matching efficiency

FIGURE A.1: An analysis of matching algorithm performance.

resulting efficiency was suspiciously high. There is probably some mistake in there, but the bug was not worth finding and fixing due to time constraints.

A.2 root_mldata

Python package to read ROOT trees into numpy arrays (using root_numpy) in a format that is suitable for any numpy-supporting machine learning library, suck as TensorFlow, Scikit-Learn and Keras. Can be configured using standard JSON files for a range of neat options.

A.3 tf-nntrainer

A set of neural networks that can be initialized, trained and retrained with the TensorFlow library. Includes a command line interface for configurable sessions. Also able to extensively evaluate the performance of the networks. Plotting library to visualize spectra included. Includes multilayer perceptrons, recurrent neural networks, normal linear regression and wide-and-deep networks. Bindings for the excellent utility tensorboard, a training visualization interface, and tfdebug, a debugging interface, are included as well.

A.4 scikit-learn trainer

Train and save any scikit-learn regressor, with tools to evaluate performance and plot spectra. Works with all regressors in Scikit-learn and can be fully configured with JSON files.

Bibliography

[1] The ATLAS Collaboration. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: (2012). DOI: 10.1016/j.physletb.2012.08.020. eprint: arXiv:1207.7214.

[2] The CMS Collaboration. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: (2012). DOI: 10.1016/j.physletb.2012.08.021. eprint: arXiv:1207.7235.

[3] Matthias Schlaffer. *Boosted Higgs Channels*. 2015. eprint: arXiv:1509.04958.

[4] Jonathan M. Butterworth, Inês Ochoa, and Tim Scanlon. "Boosted Higgs $\to b\bar{b}$ in vector-boson associated production at 14 TeV". In: (2015). DOI: 10.1140/epjc/s10052-015-3592-5. eprint: arXiv:1506.04973.

[5] A. Hoecker et al. "TMVA - Toolkit for Multivariate Data Analysis". In: (2007). eprint: arXiv:physics/0703039.

[6] *Review of Particle Physics*. http://pdglive.lbl.gov. Accessed: 2017-03-05. 2016.

[7] Adriano Di Giacomo. "Understanding Color Confinement". In: (2013). DOI: 10.1051/epjconf/20147000019. eprint: arXiv:1310.2401.

[8] B. R. Webber. "Fragmentation and Hadronization". In: (1999). eprint: arXiv:hep-ph/9912292.

[9] The ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (2008), S08003. URL: http://stacks.iop.org/1748-0221/3/i=08/a=S08003.

[10] *Cern Courier: CMS identifies Higgs bosons decaying to bottom quarks*. http://cerncourier.com/cws/article/cern/61251.

[11] Bruce Mellado Garcia et al. "CERN Report 4: Part I Standard Model Predictions". In: (May 2016). URL: https://cds.cern.ch/record/2150771.

[12] *Higgs boson branching ratios*. http://hep.wits.ac.za/research/higgsphys.php.

[13] Jonathan M. Butterworth et al. "Jet substructure as a new Higgs search channel at the LHC". In: (2008). DOI: 10.1103/PhysRevLett.100.242001. eprint: arXiv:0802.2470.

[14] David Krohn, Jesse Thaler, and Lian-Tao Wang. "Jet Trimming". In: (2009). DOI: 10.1007/JHEP02(2010)084. eprint: arXiv:0912.1342.

[15] Stephen D. Ellis, Christopher K. Vermilion, and Jonathan R. Walsh. "Recombination Algorithms and Jet Substructure: Pruning as a Tool for Heavy Particle Searches". In: (2009). DOI: 10.1103/PhysRevD.81.094023. eprint: arXiv:0912.0033.

[16] Jesse Thaler and Ken Van Tilburg. "Identifying Boosted Objects with N-subjettiness". In: (2010). DOI: 10.1007/JHEP03(2011)015. eprint: arXiv:1011.2268.

[17] Andrew J. Larkoski, Gavin P. Salam, and Jesse Thaler. "Energy Correlation Functions for Jet Substructure". In: (2013). DOI: 10.1007/JHEP06(2013)108. eprint: arXiv:1305.0007.

[18] Andrew J. Larkoski, Ian Moult, and Duff Neill. "Power Counting to Better Jet Observables". In: (2014). DOI: 10.1007/JHEP12(2014)009. eprint: arXiv:1409.6298.

[19] The ATLAS Collaboration. *Luminosity Public Results Run 2*. https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResults

[20] J. Alwall et al. "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations". In: (2014). DOI: 10.1007/JHEP07(2014)079. eprint: arXiv:1405.0301.

[21] Snezana Nektarijevic. *Jet Vertex Charge Reconstruction*. Tech. rep. ATL-PHYS-PROC-2015-200. Geneva: CERN, Dec. 2015. URL: https://cds.cern.ch/record/2117630.

[22] Torbjörn Sjöstrand et al. "An Introduction to PYTHIA 8.2". In: (2014). DOI: 10.1016/j.cpc.2015.01.024. eprint: arXiv:1410.3012.

[23] Bo Andersson et al. "Parton Fragmentation and String Dynamics". In: *Phys. Rept.* 97 (1983), pp. 31–145. DOI: 10.1016/0370-1573(83)90080-7.

[24] *PYTHIA: Fragmentation*. http://home.thep.lu.se/~torbjorn/pythia82html/Fragmentation.html.

[25] J. de Favereau et al. "DELPHES 3, A modular framework for fast simulation of a generic collider experiment". In: (2013). DOI: 10.1007/JHEP02(2014)057. arXiv: 1307.6346.

[26] W Barletta et al. "Future Hadron Colliders: from physics perspectives to technology R & D". In: *Nucl. Instrum. Methods Phys. Res., A* 764.CERN-ACC-2014-0230 (Jan. 2014), 352–368. 17 p. URL: https://cds.cern.ch/record/1956771.

[27] *Semileptonic b decay*. http://www.gla.ac.uk/media/media_443034_en.png.

[28] P. Cortez et al. "Modeling wine preferences by data mining from physicochemical properties". In: *Decision Support Systems* 47.4 (1998), pp. 547–553.

[29] Karl Pearson F.R.S. "LIII. On lines and planes of closest fit to systems of points in space". In: *Philosophical Magazine Series 6* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720. eprint: http://dx.doi.org/10.1080/14786440109462720.

[30] *Stack Overflow: Making sense of PCA*. https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues.

[31] J. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297. URL: http://projecteuclid.org/euclid.bsmsp/1200512992.

[32] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. ISBN: 0-13-022278-X.

[33] Renato Cordeiro de Amorim and Boris Mirkin. "Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering". In: *Pattern Recognition* 45.3 (2012), pp. 1061–1075. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2011.08.012. URL: http://www.sciencedirect.com/science/article/pii/S0031320311003517.

[34] Marcin Andrychowicz et al. *Learning to learn by gradient descent by gradient descent*. 2016. eprint: arXiv:1606.04474.

[35] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. eprint: arXiv:1412.6980.

[36] W Clark Smith. *Extracting $b\bar{b}$ Higgs decay signals using multivariate techniques*. http://www2.slac.stanford.edu/suli/2011/PAPERS/PAPERS/Smith_suli_paper.pdf. 2011.