Classification of events for the ANTARES neutrino detector

Master Thesis in Computational Engineering

submitted
by

Stefan Geißelsöder

born 14.3.1986 in Ansbach

Written at

Lehrstuhl für Mustererkennung (Informatik 5)
Department Informatik
Friedrich-Alexander-Universität Erlangen-Nürnberg

in Cooperation with

Erlangen Centre for Astroparticle Physics (ECAP)
Physikalisches Institut
Friedrich-Alexander-Universität Erlangen-Nürnberg

Advisor: Dipl.-Inf. Patrick Kugler, Dipl.-Inf. Wilhelm Haas, Dr. Thomas Eberl,
Prof. Dr. Gisela Anton, Prof. Dr. Joachim Hornegger, Prof. Dr. Björn Eskofier

Started: 01.03.2011

Finished: 18.7.2011

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien des Lehrstuhls für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Erlangen, den 18.Juli 2011

Übersicht

Diese Masterarbeit beschäftigt sich mit der Klassifizierung der verschiedenen Ereignistypen im ANTARES Neutrinoteleskop. Da der mit Abstand überwiegende Teil der vom Teleskop aufgenommenen Daten aus unerwünschten Untergrundereignissen besteht, ist eine gute Trennung der verschiedenen Ereignisse für die Analyse sehr hilfreich. Die im Teleskop möglichen physikalischen Ereignisse wurden analysiert und zu vier Hauptgruppen zusammengefasst.
Basierend auf Simulationsdaten wird ein Random Decision Forest Klassifikator auf 137 speziell dafür berechnete Merkmale trainiert. Die Leistungsfähigkeit dieser Methode wird anhand verschiedener Aufgabenstellungen auf Simulationsdaten evaluiert und der Einfluss sowohl der Programmparameter, als auch von Störfaktoren auf die Gesamtleistung untersucht. Desweiteren wird das Klassifikationsverhalten auch für vom ANTARES Teleskop gemessene Daten untersucht. Die gezeigten Klassifikationsraten von 92% bei 100% Akzeptanzrate bzw. Effizienz für die entscheidende Aufgabenstellung lassen sich durch das selektive Abweisen von schwer klassifizierbaren Signalen weiter steigern. Gegenüber den getesten Störeinflüssen zeigt sich die Methode generell robust. Obwohl die Genauigkeit noch nicht ausreicht, um sämtliche Klassifikationsprobleme, die im Rahmen der Datenanalyse auftreten, vollständig zu lösen, so lassen die bisherigen Ergebnisse darauf schließen, dass mit dieser Arbeit ein vielseitiges und sehr nützliches Werkzeug für die Auswertung der ANTARES-Daten gelungen ist.

Abstract

This thesis presents an event classification for the ANTARES neutrino telescope. Since most of the recorded signals are undesired background events, a reliable distinction between the different event types is very useful for the data analysis. The possible event types are analyzed and four main types are identified. Simulation data is used to train a random decision forest classifier for 137 specifically designed heuristic features. The performance of this flexible classification is evaluated in several tasks on simulated data, the relevant parameters are studied, and its robustness against background noise and other problems is analyzed. The behavior for data recorded by the ANTARES telescope and a selection of neutrino candidates is also shown and interpreted. The results in general show a high classification performance with 92% classification rate at an acceptance rate or efficiency of 100% for the most relevant classification task, which can be further improved by the selective rejection of events. The algorithm proves to be rather robust against the influence of all tested disturbances. Although the precision is not yet sufficient to perfectly solve all classification problems posed by ANTARES, the obtained data suggest that this work constitutes a versatile and very useful tool for the analysis of ANTARES data.

Contents

Chapter 1

Introduction

Until the 20^{th} century astronomy was limited to the visible part of the electromagnetic wave spectrum. This slowly began to change once parts of the background noise observed in transatlantic communication were identified as the first radio signals from the Milky Way by Karl Guthe Jansky in 1931 [Jan33]. With the discovery of more radio sources and the insights gained by early radio astronomy, the next logical step was to examine other wavelengths as well. In the following decades new ground and satellite based telescopes expanded our knowledge about the universe. Many sources were found, for instance by X-ray or γ-ray telescopes. Every time a new energy range was used, also new discoveries were made. A good example for an object revealing different properties at different wavelengths is the galaxy Centaurus A as seen in figure 1.1.
But all electromagnetic waves share the major drawback that matter absorbs the radiation. This prevents a glimpse into very dense regions like the Galactic Center or at extremely distant sources. The exploration of exactly these regions promises exciting and probably also completely new phenomena. To overcome the typical problems for photon based telescopes one solution would be to observe particles instead of electromagnetic waves. Such particles for instance are "cosmic rays", mainly consisting of protons and alpha particles. But due to their electric charge these are deflected for instance by the galactic magnetic fields. This effect strongly limits the usage of cosmic ray particles for a telescope to special cases like very high energies, which is explored for instance by the Pierre Auger Observatory [Col11].

The comparatively new field of neutrino astronomy is a promising candidate to overcome these drawbacks. Several models predict, that neutrinos are often generated by the same sources that also produce cosmic rays. But since neutrinos do not carry any electric charge, they are not deflected by the galactic magnetic field and point back to their source. Furthermore, they only interact via gravity and the short ranged weak force, allowing them to travel large distances

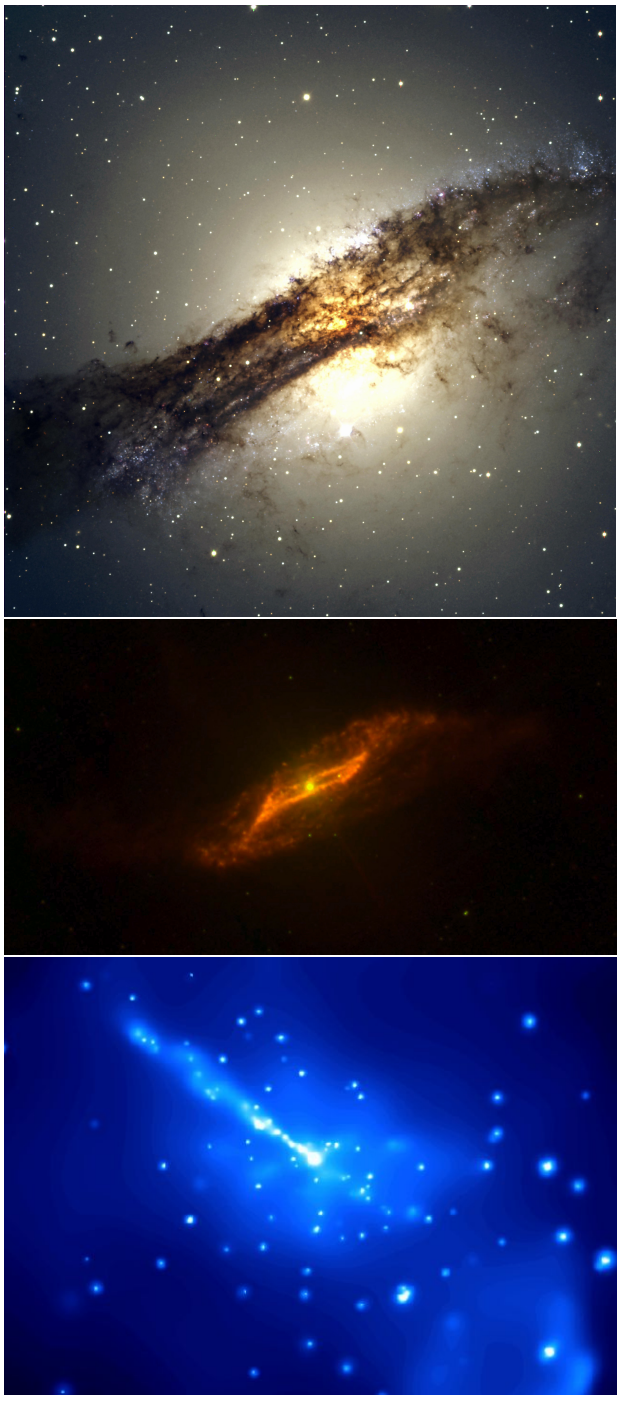

Figure 1.1: The galaxy Centaurus A (NGC 5128), as seen in optical wavelength (top), infrared (middle) and gamma rays (bottom). Each part of the spectrum reveals different characteristics.

unaltered. Because neutrinos cannot be observed directly, their detection must rely on secondary particles, which are generated once a neutrino interacts with matter. These secondary particles

can emit Cherenkov radiation, which occurs if charged particles travel faster than the speed of light in the medium they are currently traversing. This radiation can then be measured for instance by highly sensitive photomultipliers.
The “Astronomy with a Neutrino Telescope and Abyss environmental RESearch” project (ANTARES) [A+11] is a Cherenkov radation based neutrino telescope, located in the Mediterranean sea 40 km off the French coast. ANTARES is built at a depth of 2500 m to shield it against the background of muons generated by interactions of cosmic rays in the Earth’s atmosphere.

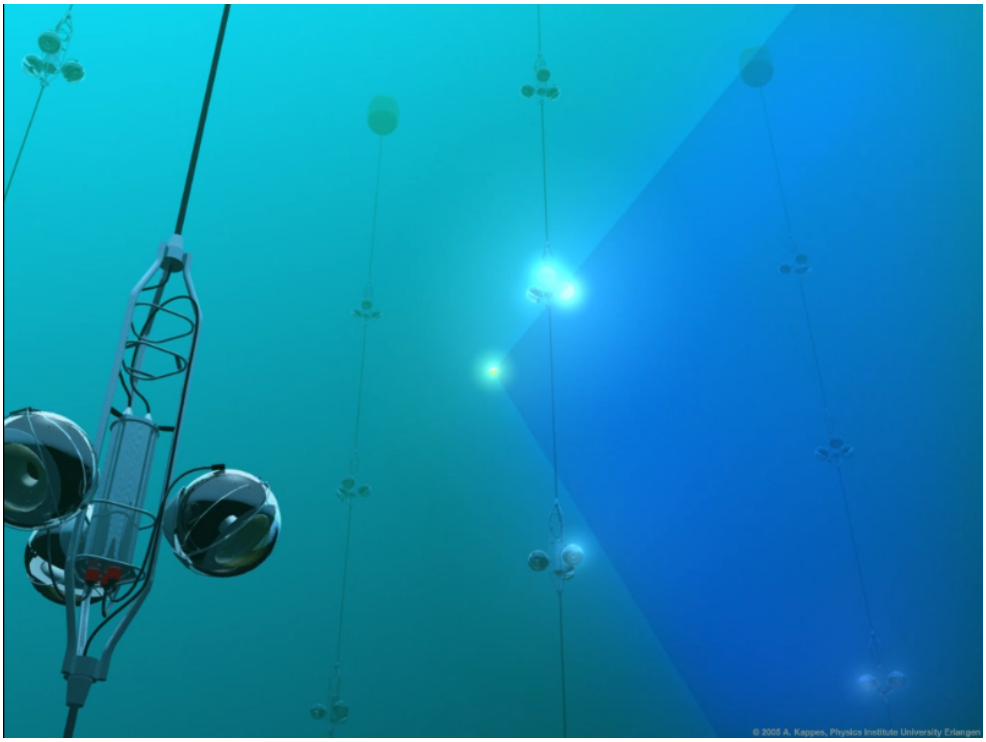

Figure 1.2: Artistic view of a neutrino detection within Antares

Every complex experiment today involves sophisticated simulations and excessive computations during the experiment and for the analysis of the obtained data. The C++ software framework “Seatray“ [1] used in this experiment has its roots in a similar experiment, the ”ICECUBE” [HK10] detector at the South Pole. All scripts and programs designed for this thesis follow the common design structure required by this framework.

Several algorithms have been designed to reconstruct for instance the direction of a specific particle or its energy. But because these algorithms rely on the assumption to reconstruct an event of one specific type, the results of these algorithms can of course only be meaningful if the

[1] http://wiki.km3net.physik.uni-erlangen.de/index.php

input signal really has been of the assumed type. A proper distinction becomes even more useful, if one takes into account the high rate of undesired atmospheric particles. There are around six orders of magnitude more background muon events from the atmosphere than actual atmospheric neutrino events, which themselves still constitute an irreducible background in the search for high-energy cosmic neutrinos. An analysis based on almost only background of course could not give useful results. While some algorithms already exist to distinguish the different event types, a significant improvement would be beneficial for all later algorithms and the analyses based on them. Automatically labeling the ANTARES data constitutes a problem very well known to the pattern recognition community: classification.

1.1 Thesis overview

The focus of this thesis lies on the classification of events using Monte Carlo simulations as a basis. For this purpose several features were designed specifically to use the information available in the ANTARES neutrino telescope. A detailed description of the features used is contained in chapter 5.2. To find the best suited subset of available features, one needs to solve an optimization problem. One way to solve this was using genetic algorithms to search for the optimal subset. The other approach was to use an open source software package, namely "Weka 3". A description of the JAVA tool developed at the University of Waikato can be found in [HDW94]. The classification method was chosen after investigation of many available methods, which were already implemented in Weka 3. For the given task and the available features the "random decision forest" (RDF) outperformed all others, or in some cases offered the same performance in considerably shorter runtime. Therefore this algorithm is used in the implementation "SG-Classify", which is relying on a data analysis library called "alglib"[2] by Vladimir Bystritsky and Sergey Bochkanov. The concepts behind Random Decision Forests are introduced in chapter 4.3.

After the experiment and the implementation are explained in chapters 3 to 5, chapter 6 contains the results. Several different scenarios (two classes, four classes, ...) are tested and the robustness for several background noise levels and hardware defects is evaluated. The performance is compared to already existing tools, where the RDF classification yields better classification results and increased flexibility. The performance of the algorithm used on real data samples instead of simulated data is also evaluated, because very often it is not feasible to have Monte Carlo simulation match reality in every tiny detail. Nevertheless the simulation data are essential, because the RDF classifier must be trained on already labeled data, which only simulations can

[2] www.alglib.net

provide reasonably.

1.2 Literature review

In the following a short overview on already existing literature related to this topic will be given and the differences to this thesis are highlighted. Some patents related to the topic can be found in appendix D.

Developed at CERN in Geneva, Switzerland, ROOT is a flexible framework for data analysis commonly used throughout the particle physics community. The "Toolkit for Multivariate Data Analysis" (TMVA) is a widely applicable tool for data treatment and classification integrated into ROOT. For further information about ROOT and TMVA see [BR97] and [HPJ+09]. Already implemented into TMVA are algorithms such as AdaBoost, Support Vector Machines or Boosted Decision Trees. But since this is a multi-purpose framework with no direct consideration of the ANTARES telescope, of course there is no specifically designed feature extraction (as presented in chapter 5.2). Also the classification algorithm Random Decision Forests, desired for this thesis due to the high performance on the features, is not contained. ICECUBE, which is a similar neutrino telescope, also has to deal with signal classification. A very promising approach to tackle the classification, which also uses Random Decision Forests, is pursued by Tim Ruhe at the University of Dortmund [RMS11]. The feature sets of are different, although the structure of the experiments is rather similar. The algorithm discards events using other relatively complex direction reconstructions before they are actually classified, which is not done in this thesis. Additionally, the implementation with which the results are produced is Rapidminer [MWK+06], a tool which (in this case) internally uses the Weka implementation of RDFs. To be able to run within the Seatray framework, the classification needs to be directly integrated within the source code. Therefore these tools are also no option.

During the time ANTARES has been planned and operated several approaches have been already evaluated. The neutrino telescope does not only observe optical signals, but also acoustic ones. The different events also lead to different acoustic signals, which also need to be identified. Max Neff has evaluated several classifiers, one of them the RDF algorithm, on these acoustic data [N+10]. It turns out that some event types can be separated very well and that clustered boosting trees showed the best performance. As an example for the optical signals a Hough transformation approach by Stefanie Wagner [Wag10] should be mentioned. This approach intuitively seems to be suited to reconstruct straight tracks. But the developed algorithm returns a hit selection, instead of a classification as result. A hit selection attempts to find those hits which are likely not

background noise, which corresponds to a signal segmentation in pattern recognition. Another previous work by Klaus Geyer [Gey10], dealing with exactly the same issue as this thesis, uses artificial neural networks to identify the signals. The focus was on separating up and down going signals. The main problems to overcome were the sensitivity of neural networks especially when considering weak features and partly the background noise. Loosely translated the author concludes “Considering the high number of atmospheric muons a background reduction of around 60% is not sufficient” (page 55).

Chapter 2

Neutrinos

2.1 Properties

Neutrinos are everywhere, around us, within walls and even within ourselves. In every second of our life billions of solar neutrinos travel trough our body usually without any interaction. These weakly interacting particles were first postulated by Wolfgang Pauli in 1930 in a letter [Pau30]. He found that the electrons resulting from β-decay according to theory should all have the same energy instead of the continuous spectrum which is observed. To solve this he postulated a neutral particle with those properties that were actually found in neutrinos decades later. This particle must carry away some of the energy resulting from β-decay undetected. The experimental proof for the existence of neutrinos succeeded in 1956. In the proximity of a nuclear reactor, Cowan and Reines [CRH$^+$56] were able to detect chemical elements within their experiment only explainable to be products of neutrino interactions.

The widely accepted standard model of particle physics categorizes the neutrino as a lepton. All leptons have the spin number of $\frac{1}{2}$ in common. The most well-known lepton is the electron e. Two others are the muon μ and the tau τ. For each of these three particles a corresponding neutrino is known, the three so called "Neutrino flavors": muon neutrino ν_μ, electron neutrino ν_e and tau neutrino ν_τ. During the last decades research has confirmed the possibility for a neutrino of one flavor to change to another one. This so called "neutrino oscillation" is described in detail in [F$^+$98]. For each of these six particles an antiparticle exists, denoted by a bar on top as for $\bar{\nu}_\mu$. Leptons are, unlike quarks and particles consisting of quarks, not subject to the strong force. This fundamental force for instance binds protons and neutrons to each other in a nucleus. In contrast to other leptons all neutrino flavors do not carry any (electromagnetic) charge either. That is

why they are also not subjected to any electromagnetic forces. The remaining two fundamental forces, gravity and the weak force, do influence neutrinos, but since the masses of neutrinos are very small ($m_\nu < 2\ \mathrm{eV/c^2}$; from [Y+06] compared to $m_e \approx 0.511\ \mathrm{MeV/c^2}$ for an electron) gravity only has an almost negligible influence on neutrinos. The weak force on the other hand has a very short effective range. Taking all this into consideration, it becomes obvious that the detection of neutrinos and especially of their direction is a nontrivial task.

2.2 Detection

Due to the minimal interaction of neutrinos with matter a direct detection is far beyond the possibilities of today's technology. However, the detection can be done indirectly by searching for interaction products of neutrinos with nuclei. Two different mechanisms for such an interaction between a high-energy neutrino ν_l of flavor $l = \mu, e, \tau$ and a target nucleus N are known. The charged-current (CC) interaction reads as:

$$\nu_l + N \rightarrow l + X \tag{2.1}$$

The neutral-current (NC) interaction as:

$$\nu_l + N \rightarrow \nu_l + X \tag{2.2}$$

If they overcome the energy threshold the resulting interaction products emit Cherenkov radiation, which can be detected. A far more detailed explanation of the processes relevant for high-energy neutrino interactions can be found in [G+96] or [CS10].

2.3 Cherenkov radiation

Cherenkov radiation occurs whenever a charged particle, for instance a μ travels with a velocity v exceeding the speed of light in the insulator medium it is traveling through. If c is the speed of light in vacuum, the speed of light for sea water would be around $0.75 \cdot c$ whereas the produced secondary particles move with relativistic velocities $v \approx c$. Due to the charge of the particle it polarizes the molecules of the medium along its track. At the relativistic velocities an overall dipole moment is generated. The light is emitted when the electrons of the insulator return to the equilibrium state. The emission from one particle forms a cone with a characteristic angle θ_c which depends on the index of refraction of the medium n and the velocity of the particle v.

With $\beta = \frac{v}{c}$, the characteristic Cherenkov angle is given by

$$\cos\theta_c = \frac{\frac{c}{n}}{\beta c} = \frac{1}{\beta n}. \tag{2.3}$$

The schematic of such a cone is shown in figure 2.2. The muon is located at the very right where the light fronts meet.

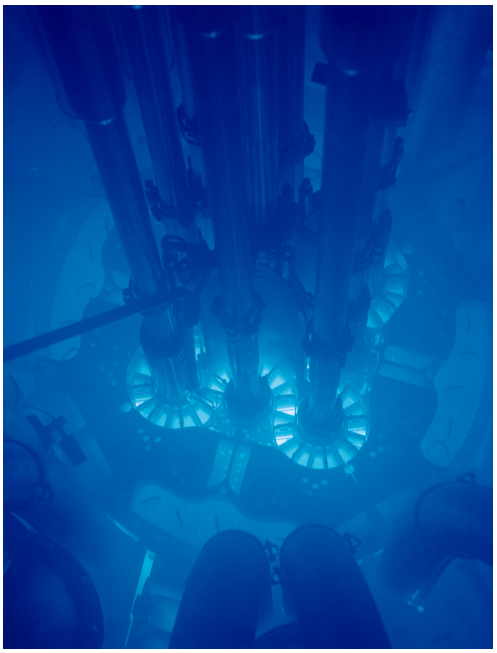

Figure 2.1: Cherenkov radiation from a nuclear reactor

For sea water and $\beta \approx 1$, the angle is approximately $42°$. Complex algorithms can exploit this characteristic for an accurate track reconstruction. An example where Cherenkov radiation is directly visible is in nuclear reactors as seen in figure 2.1. In figure 2.3 a rendered scene of a muon in the ANTARES telescope emitting Cherenkov radiation at the characteristic angle is depicted.

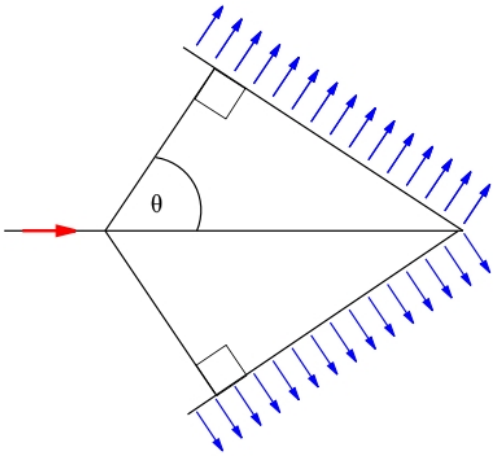

Figure 2.2: Schematic of the characteristic angle. The blue arrows indicate the light front at a given point in time

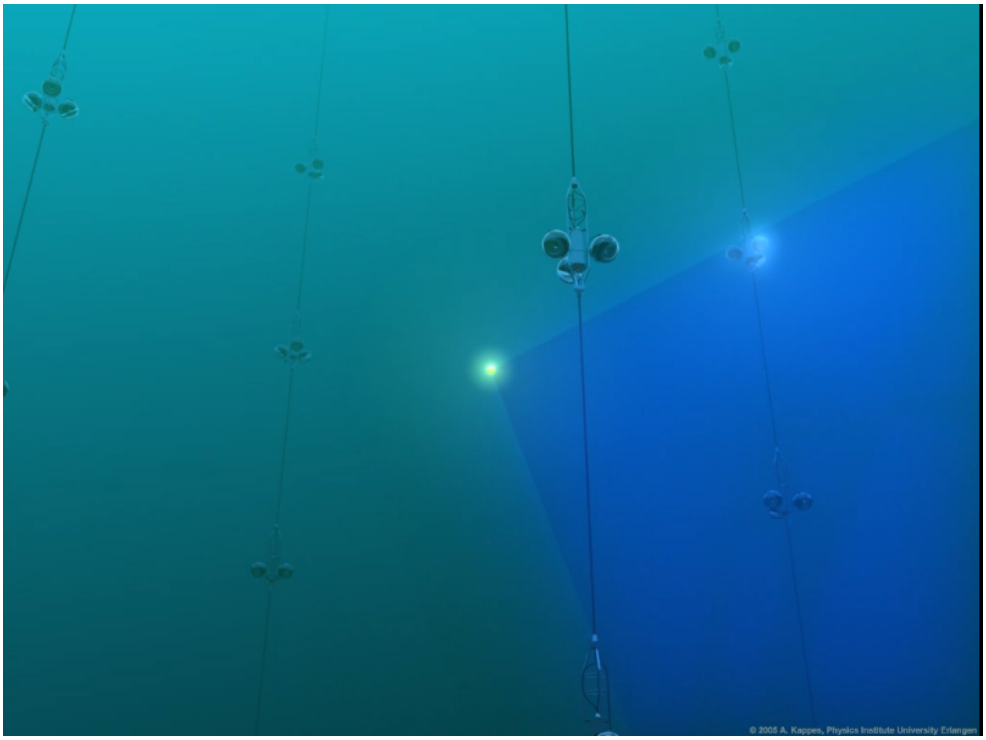

Figure 2.3: Rendered scene: A muon is traveling through the detector, emitting blue Cherenkov radiation at its characteristic angle

2.4 Signal types

The main goal of this thesis is to distinguish the signals which the different events induce in a Cherenkov radiation based telescope which can be seen in figure 2.4. While atmospheric muons

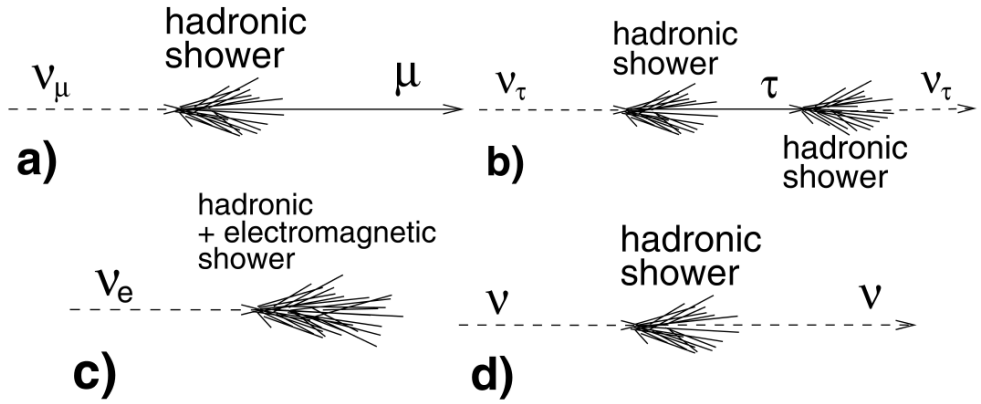

Figure 2.4: The possible signals in the detector: a) CC ν_μ b) CC ν_τ c) CC ν_e d) NC ν; from [CS10]

and neutrinos reach the detector from above, only neutrinos up to energies $E_\nu \approx 1$PeV may pass through the Earth practically unhindered and can be detected as upgoing signals.

For NC interactions, described in equation 2.2, the signal for all neutrino flavors is a hadronic shower with the neutrino continuing its path with lower energy. A hadronic shower consists of secondary interaction products such as pions, kaons, protons and neutrons in variable numbers. Due to the small extension of a shower event compared to the geometric layout of the used neutrino telescope, showers can approximately be considered point sources.

The CC interactions of all neutrino flavors produce the corresponding lepton, as stated in equation 2.1. For instance the CC interaction of the ν_e results in an electron. But this highly energetic electron has a high probability to radiate a photon via bremsstrahlung, resulting in $\mathrm{e^+e^-}$ pair production and finally an electromagnetic shower event. This event signature is very similar to those of the NC interactions, since both shower types cannot be resolved reasonably.

The CC ν_μ is the most interesting interaction for neutrino astronomy because the muon as secondary particle generates a reconstructible track. At the interaction location called ”vertex” a shower event is visible, with one Cherenkov radiation emitting muon track shooting away through the water. The length of this track depends on the energy of the muon and therefore on the energy of the primary muon-neutrino. For cosmic neutrinos the track length ranges from several tens of meters up to tens of kilometers. Therefore a muon generated far outside the detector may still be reconstructed. One very important aspect for any astronomy based on these tracks is

the difference between the initial neutrino direction and the resulting muon direction. In [CS10] an energy dependent limit for the angular difference can be found to be

$$\theta_{\nu\mu} \leq \frac{0.6^\circ}{\sqrt{E_\nu(\mathrm{TeV})}} \tag{2.4}$$

with $\theta_{\nu\mu}$ denoting the angular difference and E_ν $[\mathrm{TeV}]$ the energy of the primary neutrino in tera electronvolt (10^9 eV). For interesting energies above one TeV the angular difference is therefore well below one degree. To be able to use muon-neutrinos in astronomy, the small angular deviation of the detected muon from the original neutrino direction is essential.
The last possibility, the CC ν_τ event, consists of one vertex shower, a τ particle traveling some distance emitting Cherenkov radiation and a second shower once the τ decays. But the traveling distance of the τ is highly energy dependent. For lower energies the distance is just a few meters, so the two showers cannot be distinguished. For very high energies the distance can reach up to several kilometers. Therefore many of the interesting high-energy events (showers with tracks and the second shower often outside the detector) look exactly like a CC ν_μ interaction. That is the reason why no specific data are available for this event type and it is not separately considered here.
Each lepton has an antiparticle. But since the detector signals generated by the antiparticles do not differ from their respective counterparts they cannot be distinguished in the experiments. The computations and results consist of randomly mixed neutrinos and the corresponding antineutrinos, since only the signal character is relevant for this thesis and not its real frequency, which would be influenced by the composition. Muons which are generated in the atmosphere can be observed as the same tracks as for CC ν_μ, but downgoing only and sometimes as bundles of more than one muon. One must note that very energetic muons produce electromagnetic showers along their track. Nevertheless these are considered to be tracks, too. In a nutshell only the signal classes

- tracks (μ)
- pure showers (NC ν, CC ν_e)
- combinations of both (CC ν_μ)

can be reasonably separated.

2.5 Possible astronomical neutrino sources

Until today only two sources of extraterrestrial neutrinos have been observed. For once neutrinos from our suns fusion processes have been analyzed. The other observation [Pea99] were 24 neutrinos originating from the supernova SN 1987A detected by three experiments, Kamiokande II, the Irvine-Michigan-Brookhaven detector and Baksan. But all those neutrinos reach an energy of at most some MeV (10^6 eV). The ANTARES neutrino telescope, explained in chapter 3, is designed primarily for the search of cosmic neutrinos with energies from 10 GeV ($10 \cdot 10^9$ eV) up to EeV (10^{18} eV).

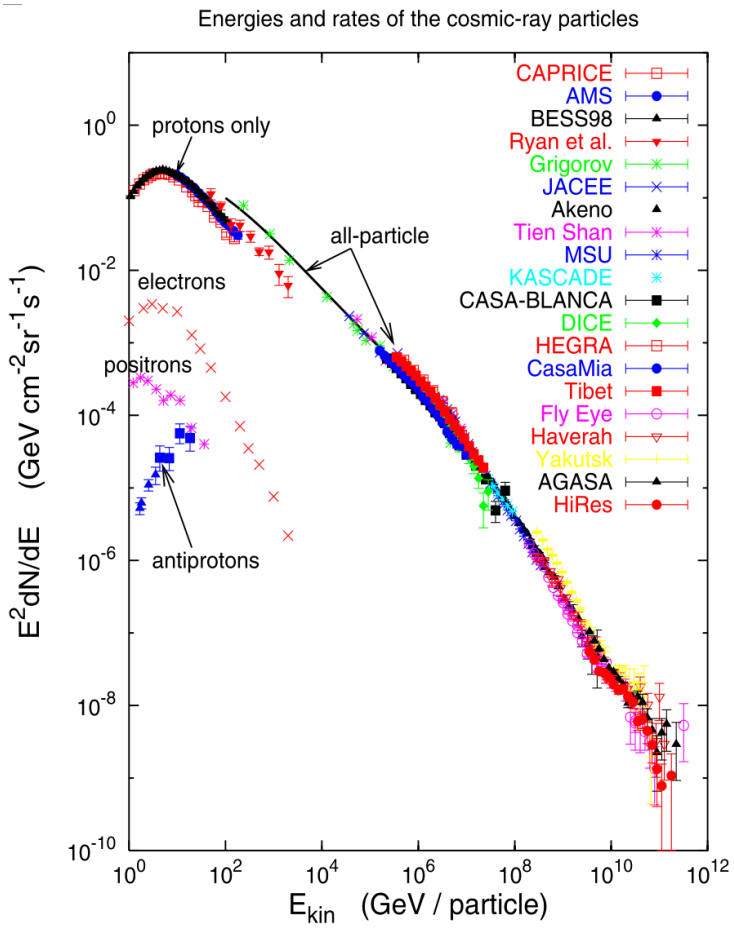

Figure 2.5: Energy spectrum of cosmic rays; from [Hil06]

Many models predict a connection between the origin of high-energy neutrinos and high-energy cosmic rays, as explained for instance in [Bec08]. Cosmic rays (CRs) are in fact charged particles rather than rays. They mainly consist of protons, but also some heavier nuclei. These particles strike the Earth at all times but due to our planet's atmosphere they interact before

reaching the ground. CRs cannot simply be traced back to their origin because the galactic and intergalactic magnetic fields bend their trajectory during their journey. The energy of these particles shows a characteristic energy spectrum reaching up to several 10^{20} eV, as shown in figure 2.5.

One must add that for charged particles this energy already constitutes an upper limit. Charged particles accelerated to energies above the so called Greisen-Zatsepin-Kuzmin (GZK) cutoff loose energy by interactions with the cosmic microwave background radiation on their way through the universe. The details of this mechanism were published for instance in [Gre66]. Since many source candidates for high-energy CRs likely also are neutrino emitters [CS10], they are potential research targets for neutrino astronomy.
Some candidates from within our Galaxy are:

- Shell-type supernova remnants
- Pulsar Wind Nebulae
  A known example is the crab nebula.
- The Galactic Center
- Microquasars
  A stellar sized black hole consuming matter from a companion star, distributed throughout the Galaxy.
- The diffuse flux from CRs in the galactic plane
  High energy CRs may travel some time through space, deflected by magnetic fields. Their interactions may produce neutrinos from all directions.

Neutrinos with extremely high energies are expected to be produced at extragalactic sources, for instance:

- Active galactic nuclei
  Galaxies with a massive black hole actively consuming large amounts of matter.
- Gamma ray bursts
  Very short outbreaks of extreme energies. Most likely associated with supernovae of type Ib/c.
- Starburst galaxies
  Galaxies with unusually high star formation rates.

- Diffuse flux
  Due to the GZK cutoff extragalactic sources of ultra-high-energy cosmic rays are practically "guaranteed" to contribute to a diffuse flux of neutrinos of very high energies.

And of course there might exist sources which are not yet considered to produce neutrinos or which are completely unknown.

Chapter 3

ANTARES

The first underwater neutrino detection experiments were conducted by the U.S. experiment DUMAND and at the Russian Lake Baikal. Much of the knowledge gained there was incorporated into the European ANTARES detector. Today it is the largest neutrino telescope in the northern hemisphere, with only the similar ICECUBE detector at the South Pole having a larger instrumented volume. Further information about ICECUBE can be found in [HK10]. The location in the northern hemisphere, in our case the Mediterranean sea 40 km off the French coast, allows a neutrino telescope to watch the southern sky and therefore the center of our galaxy. The ANTARES project was set up in 1996. Today it involves research institutes from France, Italy, the Netherlands, Morocco, Russia, Spain, Romania and Germany. Research and development was conducted until 1999 and several possible locations were inspected. Further preparation led to the deployment of a first testing line in 2005. Until May 2008 the detector hardware was deployed and connected step by step.

3.1 The detector

The fully featured ANTARES detector [A$^+$11] consists of twelve instrumented lines, each made of electro-optical cables. They're anchored at the bottom of the sea and held upright with a buoy at the end of each line. These lines are spaced approximately 60 to 70 meters away from each other. The line geometry as top view can be seen in figure 3.2. Attached to each line are 25 so called storeys, each storey housing three optical modules (OMs). Simply speaking, an optical module is a pressure resistant glass sphere with a photomultiplier inside. These photomultipliers actually measure the Cherenkov radiation and integrate in short integration windows to digitally store the intensity [A$^+$10]. The three OMs on one storey are spaced 120° apart such that 360°

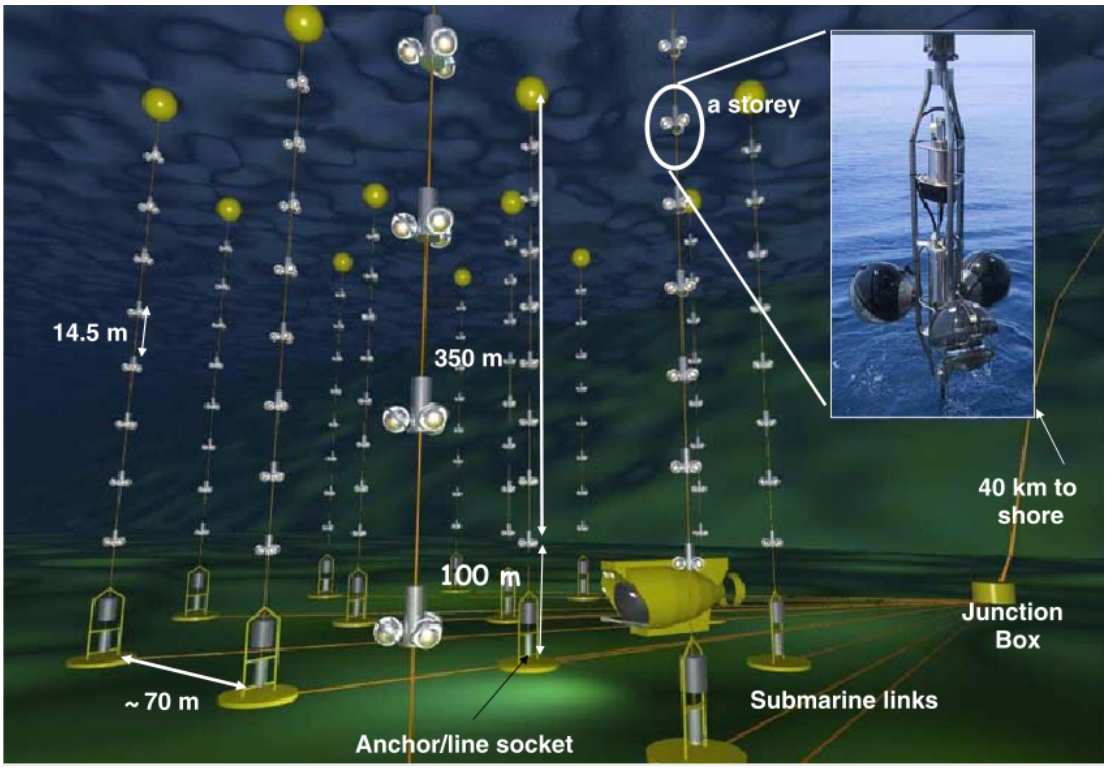

Figure 3.1: The Antares detector: Artistic depiction

are covered optimally. All OMs are oriented 45° downwards from the horizontal, since the main focus lies on upward going muon-neutrinos. The lowest storey on each line is located about 100 meters above the sea bottom with each following storey 14.5 meters above the other. For calibration purposes lasers, LEDs, compasses and tiltmeters are attached to some storeys and to one additional line, the so called instrumentation line. This line also houses some other useful equipment.

The digitalized data from one OM is called a L0 hit if it exceeds a certain minimum threshold charge, by default 0.3 photoelectrons. All L0 hits are collected and then, together with the exact timing information, sent onshore to a computer center. The time resolution of the detector is $\approx 0.5\text{ns}$. The data is processed there “online“ (without delay) to find either coincident hits on one storey or a L0 hit with a high amplitude, typically above 3 photoelectrons. In the set of these so called ”L1 hits” other algorithms called triggers search for coincidences within a time window of typically 2200ns. There are five standard triggers, “3D”, “3N”, “3S”, “3T” and “1D”. The default active triggers for data analysis at the moment are 3N and 3T. If a trigger is found the corresponding hits are flagged as “triggered“ and all L0 hits of the full detector during the time (including a time window of $\pm 2200\text{ns}$ around the cluster of triggered hits) are written to disk.

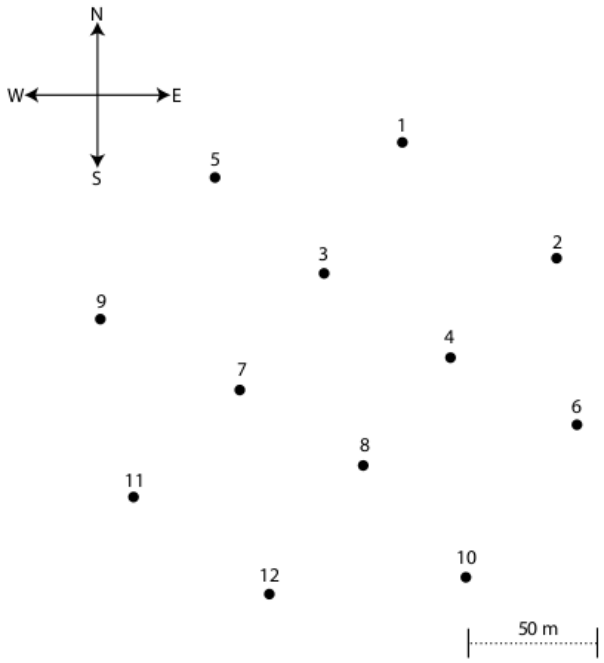

Figure 3.2: The Antares detector: Schematic top view; from [Bru08]

This is then called an event. To be able to reconstruct the geometry of the lines at a later time by means of acoustic triangulation, every two minutes acoustic pingers are fired and the hydrophone data is saved to disk. The recording of all measured data may also be triggered otherwise, for instance externally when a gamma ray burst, a very short outbreak of extreme energy, is detected by a satellite.

In short, the data obtained from the ANTARES detector for one event is a list of measurements, each consisting of

1. the line number
2. the OM number on this line
3. the measured light intensities during one integration window
4. the time when the light was measured

One hit refers to this four dimensional data vector and one event consists of many hits. Line and OM number encode the three space dimensions.

Figure 3.3: Detailed view on a storey with its three optical modules

3.2 Background noise

In the ANTARES telescope there are various background noise sources. For once atmospheric muons and atmospheric neutrinos produce many undesired signals, as mentioned in chapter 2.4. The water is meant to shield the telescope against as many of the muon events as possible, but this depends on the depth of the site in the first place. Figure 3.4 shows, that the huge number of muons reaching the neutrino telescope from straight above (cosine of 1.0) drops quickly near to the horizon (cosine of 0.0). The shielding effect can also be seen at the considerably smaller numbers of muons from all directions at $3880\ \mathrm{m}$ depth compared to the numbers at $1680\ \mathrm{m}$. The neutrinos produced in the atmosphere may also reach the detector but due to their nature there is almost no shielding effect against them. This constitutes an irreducible background for neutrino astronomy.

Besides undesired particle-induced signals there exist also other sources for optical noise in

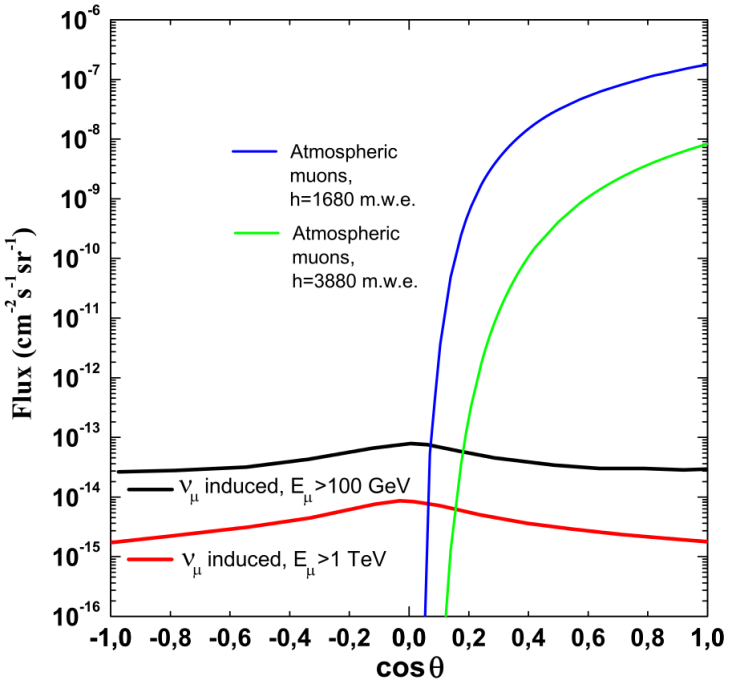

Figure 3.4: Fluxes of atmospheric muons for different depth and of neutrinos of different energy thresholds (as functions of the cosine of the zenith angle); from [CS10]

sea water. The radioactive decay of ^{40}K is always present. The two main decay channels are:

$$^{40}\mathrm{K} \rightarrow {}^{40}\mathrm{Ca} + e^- + \bar{\nu}_e \tag{3.1}$$

$$^{40}\mathrm{K} + e^- \rightarrow {}^{40}\mathrm{Ar} + \nu_e + \gamma \tag{3.2}$$

For instance the electrons from the first channel emit Cherenkov light which randomly adds to the detected signals. Since the occurrence of ^{40}K is mainly determined by the salinity of the water this background varies only little within the Mediterranean sea. According to [CS10], page 676, the contributions to the signal are at an almost constant counting rate of 30 - 40 kHz per photomultiplier.

A very variable source for background noise is bioluminescence. Microorganisms produce light partially depending on factors like the water flow velocity, the time of the year and many others. The minimal rate for this background effect is about 20 to 30 kHz. Unfortunately, this luminescence can reach several hundreds of kHz and sometimes completely prevents any useful data collection. Figure 3.5 shows the amount of these microorganisms with depth. The second source for bioluminescence are bigger organisms which sometimes happen to traverse

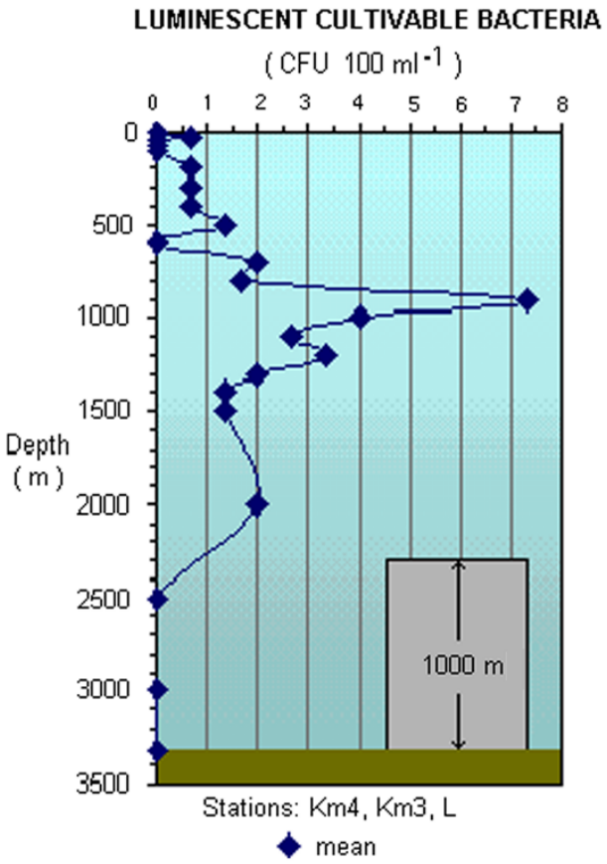

Figure 3.5: Amount of cultivable luminescent bacteria per volume for different depths from [CS10]

the detector, resulting in a dead time for the data acquisition system during which no useful data can be measured. The software for a working neutrino telescope must be able to deal with the omnipresent background signals as well as possible.

3.3 Software framework

Every experiment producing complex data needs a well designed, flexible and efficient software framework to make the knowledge contained in the data accessible. For the ANTARES detector this software is called “Seatray”. Its roots lie at the similar ICECUBE experiment and the software Icetray designed there. The software structure is highly modularized, as each application or tool is coded as a module in C++ with python bindings to the framework. These bindings allow the code to be executed by a script written in python. The coarse structure of such a script

is to create a tray object, which allows adding the desired modules to it. In this way a chain of modules to be executed for each frame (all data for one event) is set up. Each module is always connected to the following one. Once completed the tray is executed. Usually one of the first active modules is a data reader. It passes the plain data to the next module, which processes the data and passes the possibly altered data to the next module until the final module is reached, which has no further output. For the thesis one C++ module was implemented to compute certain properties, called “features“ for the events and another C++ tool to evaluate these features and classify the event based on trained knowledge. An event within the software framework will also be called ”frame“.

The second piece of software relevant for the ANTARES collaboration is a variety of different simulation tools forming different simulation chains. They start by simulating all relevant primary particles like cosmic ray particles, neutrinos or atmospheric muons. These particles are propagated until they interact. The resulting secondary particles (and the primary particle if it still exists) are then traced and many possible interactions are taken into account using Monte Carlo Simulation. Resulting radiation from various processes, especially the relevant Cherenkov radiation, is generated and propagated until it reaches an OM of the detector or gets absorbed. The optical and technical properties of the OM and the following electronics also need to be simulated and also the background from the various sources. The different simulation chain combinations differ slightly. Some codes do trace certain particles exactly while others use heuristics, for instance for parts of the shower events. As a whole the simulation chain results in a virtual detector response, but with known input particles and known energies and paths.

In the experiment nobody can assure that an observed response of the detector belongs to one specific event type, and even if it was possible for certain events, it would not be feasible to reach a sufficient number of events by manual selection. This is why the classification must rely entirely on simulated data for the training of the classifier and the performance can also be evaluated only on simulation data. The resulting final algorithm must be able to process real data, too. But the classifier of course cannot deliver a better performance than to what extent the simulation data match the real data, since the classifier can only be trained for the labeled data created by simulations.

Chapter 4

Pattern recognition

For the last decades computer science has not only been a science on its own, but also provided the tools allowing huge progress in many other fields. From the various computer science disciplines especially simulation, pattern recognition and visualization stand out as universal tools. For the task of identifying desired events, this thesis extensively uses already existing Monte Carlo simulation results which are processed by pattern recognition algorithms.

4.1 Pattern recognition pipeline

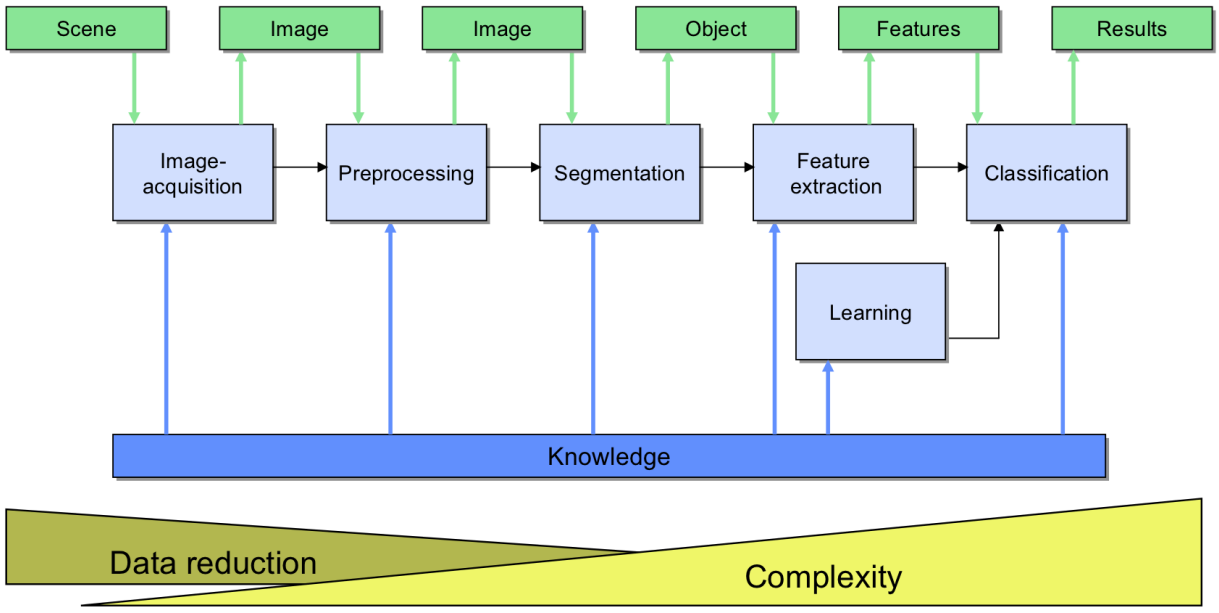

Figure 4.1: The pattern recognition pipeline (here for images); from [Wit08]

Pattern recognition follows an almost fixed workflow to achieve the identification of patterns contained in arbitrary data.

1. Data acquisition
   The first step is obtaining the data by sensors. The measurements are sampled and quantized to be stored digitally. Commonly cameras or microphones are used or, in this case, the ANTARES neutrino telescope.

2. Preprocessing
   Preprocessing means the application of filters or other techniques to the raw data. Gaussian filters, image reconstructions, normalization or edge detection are examples for often applied algorithms. A detailed insight can be found for example in [HP97].

3. Segmentation
   The part of the data containing the desired signal must be identified. There are many diverse methods for this surprisingly difficult task, for instance histograms and thresholds [DHS01] or Color-Structure-Codes [PS03] for images.

4. Feature extraction
   Feature extraction is the computation of "well suited" features on the data to contain and expose the relevant information. A feature can be any descriptive number computed from the data. Often standard transformations like wavelet transformation [Pol94] or Fourier transformation is used. Heuristic features would be for instance the total energy of a signal or the fraction of red in an image.

5. Feature selection
   To improve the classification performance often an optimal feature subset is computed. These features should contain the maximum amount of information about the classes, but the minimum amount of useless information. This is especially relevant for time critical applications. Finding these features is an optimization problem and therefore can be solved by many common optimization techniques.

6. Training
   The extracted information is used to train a suitable classification algorithm. This can be done once before usage (offline training) or continuously (online training). A very wide range of possible algorithms is available, popular examples are the Bayesian classifier,

Support Vector Machines (SVM), Artificial Neural Networks, the Nearest Neighbor classifier or Decision Trees. Detailed information about many classification algorithms can be found in [DHS01].

7. Classification
   The application of the learned knowledge by the classifier to new data tries to assign each event to the correct class. For labeled data this assignment is used to evaluated the performance of the classifier.

4.2 Genetic Algorithms

Genetic algorithms (GAs) are one of the many examples where technology is directly inspired by nature. Just like the evolutionary process causes adaptations of species to their environment, genetic algorithms start with a population of possible solutions also called individuals, improve these solutions and output the fittest solutions to the problem.

The steps can briefly be explained here:

- Initial population:
  An initial population is set up randomly. Solutions for various problems may look very diverse, some composed of real values, integers or just boolean values. In analogy to nature each of these numbers is called a ”gene“.

- Selection by fitness function:
  To determine the fitness of these solutions a so called fitness function (in optimization often cost function) needs to be specified. In nature this function corresponds to a combination of survival and reproduction. For a computer algorithm this function usually assigns a real value to each possible solution. The next step is to advance from one generation to the next. Individuals from each generation are dropped with a probability inverse proportional to their fitness. This leads to a smaller number of surviving solutions with a higher average fitness.

- New individuals:
  As an analogy to biological sexual reproduction mutation and crossover are performed to fill the generation again. New solutions are formed by crossovers of the surviving solutions from the old generation. These new individuals randomly contain parts of the parameters from the two ”fit enough“ old ones. Completely random mutations usually strike with

”low” probabilities. They help to explore possibilities otherwise not reachable by the initial setup.

- Repetition:
  The new fitnesses are determined, the population is shrunk, new individuals are created and the process continues until a certain criterion is fulfilled.

With time this population develops towards an optimally fit solution for the given fitness function. This process stops either after “enough“ iterations are performed or once the best-performing individual did not change for a certain number of iterations meaning a maximum has been reached. Unfortunately there is no guarantee that this is the globally best solution. The algorithm might run into a local optimum. Once stuck there only random mutation might lead to a better solution, but often this takes a long time. A ”big enough“ population size can increase the search space to include the global optimum. But how to define ”big enough“, ”enough” iterations, ”low“ probabilities and other parameters of the genetic algorithm is task specific and cannot be answered beforehand. The major drawback of genetic algorithms, the in practice often very long runtime, disfavors them especially for time-critical applications. Nevertheless the immense flexibility and the in practice robust behavior have caused the success of genetic algorithms in many applications. A deeper insight into the topic can be found in [Hol75].

4.3 Random Decision Forests

Assuming ”well suited“ features could be extracted from the data, the success of the overall classification stands and falls with the classification algorithm. A good classifier must fit the data used for its training well, while still generalizing the information well enough to handle unseen data with the highest possible accuracy. Different classifiers are known to have different tendencies when it comes to adaptation and generalization. The random decision forest which is used in this thesis is an advancement based on the ordinary decision tree [Ho98]. The tree structure is very well known in computer science and decision trees follow exactly this scheme. The basic elements are called nodes, with a special node called ”root” as the first node within a tree. Starting from the root an arbitrary number of other nodes are linked to the root as its children. For the special case of binary trees the number of children is two. Each node by that relation is connected to its parent. The nodes can have children that are connected in the same way again. End nodes without any children are called “leaves”. A simple example for such a decision tree with two features F1 and F2 and two classes C1 and C2 can be seen in figure

4.2. The nodes are drawn as ellipsoids, the root node is at the top and the leaves are drawn as rectangles containing the classes. In this example each leaf contains exactly one class. There is no need for a normalization of the features, since the splits are made completely independent from each other. Figure 4.3 shows an example for a partitioning of a two dimensional feature space. This is not the same partitioning as generated by the tree of figure 4.2. A simple split on one feature results in axis-parallel decision boundaries, the border where events will be classified to be class one on the one side and class two on the other side.

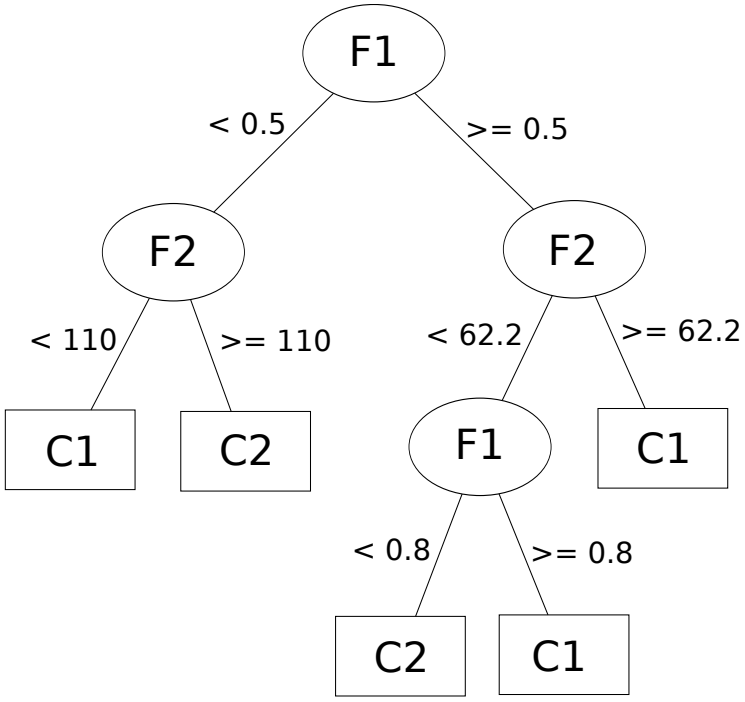

Figure 4.2: A decision tree example with two features F1 and F2 and two classes C1 and C2. Ellipses are nodes, rectangles are leaf nodes. The top node is the root.

The efficient training of such a decision tree constitutes a nontrivial task for which various algorithms have been proposed over time, for instance CART, the ID3 algorithm or the C4.5 algorithm. The details of these algorithms can be found in the literature, for example in [Qui85] or [DHS01].

The questions for one node of a binary, real valued tree with a simple threshold based split are which feature to use and what value to use as a threshold. This problem has to be solved recursively for each resulting child node until all nodes are leaves.

The optimal solution for one node results in the children nodes to contain as pure subsets as possible. Maximizing the purity is the same as minimizing the impurity. For this impurity i of

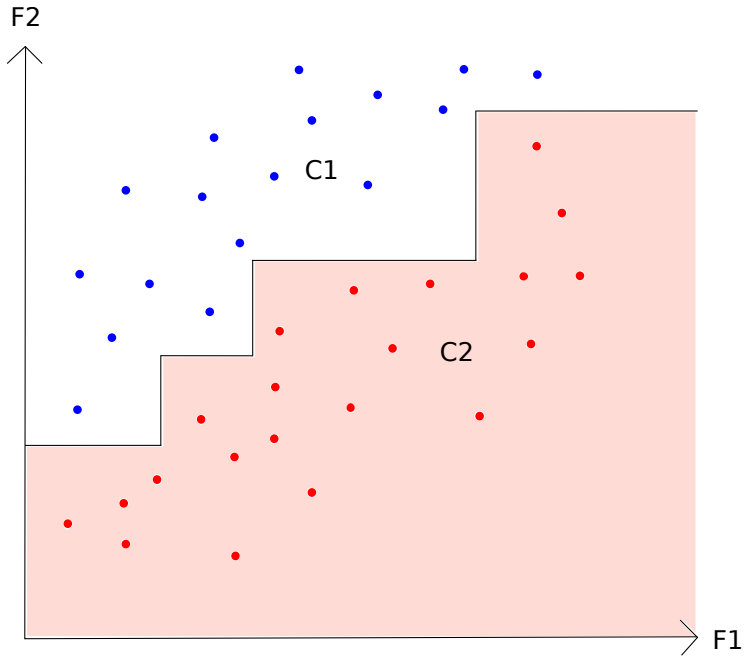

Figure 4.3: A two dimensional feature space divided by a RDF. Simple splits on one feature result in decision boundaries parallel to the axes.

a node N several measures have been proposed. Particularly useful for binary decisions is the definition

$$i(N) = P(\omega_1) \cdot P(\omega_2) \tag{4.1}$$

where $P(\omega_i)$ is the fraction of patterns at node N that are in category ω_i. This is a special case of the popular "Gini impurity" for more choices, which is defined as

$$i(N) = \sum_{i \neq j} P(\omega_i) \cdot P(\omega_j) \tag{4.2}$$

Both are taken from [DHS01]. Regardless of what measurement is used, some optimization technique must be used to determine the feature and the split for each node. For real valued decisions for instance a gradient based method might be chosen [DHS01]. This is done recursively until all child-nodes contain only events of one class, which means they have an impurity of zero. At this point trees tend to overfit the data, meaning they are extremely accurate on the training data but don't generalize the knowledge any more. This is a major drawback for the performance on unseen data, and often some backpruning mechanism is used to cut back unsuitably specific branches. But again, there are many different techniques with specific advantages and disadvantages. For more detailed information on this topic and decision tree generation in general, again

[DHS01] is recommended.

To classify an event the algorithms starting point is always at the trees root. All non-leaf nodes are designed to evaluate a certain function on one or more features. The simplest (and very common) case is just a threshold for one feature. For a binary decision tree the first child node is evaluated next if the value of the feature is below the threshold or the second child if the value is above. Non-binary nodes work in an analogous manner with potentially more thresholds or function outcomes. This procedure is repeated until a leaf is reached. The leaf either directly contains the class number instead of another evaluation function or a probability distribution from which the class number can be sampled.

A major drawback of this method is the relatively low ability to generalize in knowledge to improve the performance for unseen data. To optimize this feature again several approaches like boosting or bagging have been proposed, but there is no "optimal" tradeoff between accuracy and generalization for all problem settings [DHS01].
An advancement of decision trees, the "random decision forests" (RDF), "random forests" or "random subspace decision trees", according to the original author, have exactly this ability to generalize well. The idea behind RDFs is to generate several decision trees instead of just one, training each on a random subsample of all features. The subspace of the original search space allows different features to become relevant for a class assignment. Each decision tree is additionally trained on just a subsample of the training data. As a result of both reductions each decision tree is relatively simple and comparatively fast to evaluate. For each event every tree of a RDF is evaluated and the majority of the trees decisions determines the final class assignment. Due to its random exploration of the search space, this method is relatively robust and generalizes significantly better than an ordinary decision tree. A small performance comparison of selected classifiers can be found in the results chapter.

4.4 Evaluation methods

To evaluate the performance of the final RDF classifiers, a test data set is classified, which is completely disjunct from the data set used for classifier training. But there are many ways to display the same results. At first some terms need to be defined. In the following the term "rate" will often be used in the sense of "fraction" with values between 0.0 (meaning none) and 1.0 (meaning all).

Definition 4.1

Quality criterion: *The quality criterion is the minimum rate with which the individual trees*

of the RDF classifier must agree on the same class. It may also be called minimal or desired tree or forest agreement. Its value is specified by a parameter named "QualityCriterion".

Definition 4.2

Acceptance rate: *The acceptance rate is the rate with which the test samples were classified with a tree agreement of at least the QualityCriterion value or higher, regardless of their original or assigned class.*

Definition 4.3

Classification rate: *The classification rate is the rate with which all accepted test samples from all classes were labeled correctly. It is also called overall classification rate.*

Definition 4.4

Rejection rate: *The rejection rate is the rate with which events from the test sample were put into the rejection class. The rejection rate therefore is always* $1.0 - \mathrm{AcceptanceRate}$*, since an event can only be either accepted or rejected.*

Definition 4.5

Efficiency: *The efficiency is the rate with which a certain (desired) class is accepted. This is also called class specific acceptance rate.*

Definition 4.6

Purity: *The purity is the rate with which test samples labeled to be a specific (desired) class actually are from that class. Impurity can then be defined as* $1.0 - Purity$*.*

Relevant to note is that the efficiency and the overall acceptance rate are coupled. An increase in one value only allows the other one to either increase too, or to stay the same. The same holds for a decrease in one value. Again the other one must either stay the same or shrink. The purity of the classes is related to the classification rate.

4.4.1 Cross validation

To lower the variance of results generated by a partly random algorithm cross validation can be used. The ordinary k-fold cross validation uses all available data and randomly splits this data into k equal parts. The classification is then performed k times, each time with another part used as test data and all other parts used as training data. By this procedure every instance has been evaluated at least once. In this thesis cross validation is used more loosely, as for each of the

classification repetitions the test and training data are randomly selected again. For the huge datasets available for this task there is no difference in the classification rate between the two approaches.

4.4.2 Confusion Matrix

The first method to display the results is by plain numbers. For a task with n classes the confusion matrix is a matrix of size n x n. It holds the exact numbers how many instances from which class have been classified to be which class. To increase the confusion about the confusion matrix, there are two possibilities to write down the matrix. In this thesis the original classes are listed row-wise and the column indicates the result of the classification.
An illustrative example is table 4.1. The first column just contains the actual class numbers of

	$Result \rightarrow$	0	1	2	3	R	
$Actual\ class \downarrow$	0	1448	4	1	40	1832	0.970
	1	57	960	58	20	2171	0.877
	2	3	113	1275	25	1835	0.900
	3	185	20	62	878	2075	0.767
		0.855	0.875	0.913	0.912		

Table 4.1: Example confusion matrix for a four class problem with rejection class

the events. The first row contains the resulting class numbers and an "R" for rejection class. The second row contains all events of actual class zero. The numbers now tell that 1448 of the class zero events were (correctly) classified as class zero. 4 class zero events were incorrectly classified as class one. Only one class zero event was classified as class two and 40 class zero events were incorrectly classified as class three. 1832 class zero events were rejected because they did not fulfill the desired quality criterion. The last column of this row is the rate with which accepted class zero events were classified correctly, so 97%. If A is the sum of all accepted class zero events ($1448 + 4 + 1 + 40 = 1493$) and R the number of rejected class zero events (1832), the efficiency for class zero can be calculated as $\frac{A}{A+R} = 0.449$. In the last row the second column states the purity for class zero. In this case it is 0.855. For the other classes the numbers are analogous. The most relevant statement for a classification, the overall classification rate, can simply be computed by the average of the classification rate for each class weighted with the number of classified instances for this class. In this case the overall classification rate is 0.886 with an acceptance rate of 0.394. To obtain statements about the classification performance for all classes and not just for the dominant muon events, the presented results are computed on data

with a homogeneous class distribution if not stated otherwise.

A confusion matrix is a compact way to represent almost all relevant information about a classification result. But especially when many results or specific criteria are to be compared one often prefers a figure instead of plain numbers.

4.4.3 ROC Curve

A receiver operating characteristics (ROC) curve is a handy tool to visualize the performance of a classifier. This method is even independent of the class distribution within the test sample. A drawback is that the default ROC curve is defined for two class problems only. It relies on two values, the true positives and the false positives. For a two class problem without rejection the confusion matrix looks something like table 4.2.

This confusion matrix already implies that class zero is "positive" in terms of a hypothesis,

	0	1	
0	tp	tn	x
1	fp	fn	y
	u	v	

Table 4.2: Example confusion matrix for ROC curves

class one is negative. True positive (tp) is then defined as the number of events from class zero which are correctly classified as class zero. The tp rate is this number normalized to the total number of class zero events. False positive (fp) are all events from class one which are classified as class zero. The fp rate is again normalized, this time to the number of class one events. Both values therefore may range from zero to one. A classification algorithm that guesses randomly results in both values to be the same. Low values indicate a conservative classification, likely rejecting positive values. High values indicate a very liberal behavior, accepting many events as possible class zero events. A good classifier produces high true positive values and low false positives. A classifier without any quality parameter produces only one point in the ROC space, but by variation of the tree agreement requirement QualityCriterion this algorithm can produce a ROC curve instead of just one point. Due to an implicit quality criterion of $\frac{1}{n}$ for n class tasks the ROC curves cannot extend to high false positive rates in this implementation. More detailed information about ROC curves can be found at [Faw06]. ROC curves will be presented for two class tasks only.

Chapter 5

Methods and Implementation

The classification software package developed during this thesis, humbly called SGClassify, consists of two C++ modules and several Python scripts. As already denoted the C++ code is designed to integrate into the Seatray framework. Following the pattern recognition pipeline described in 4.1 the first step is to obtain data. The C++ codes and the python scripts invoking them can be found on the accompanying disk or partly in appendix E.

5.1 Overview

The pattern recognition pipeline scheme explained in section 4.1 is adapted to the needs and challenges encountered in this thesis. A short overview is given here, the detailed explanation of the essential steps can be found in the following chapters.

1. Data acquisition
   The data acquisition is done in two ways. For once the full ANTARES telescope has taken data since 2008, with first data acquisition dating back to 2005 for a small test set-up. The other source for data are results of various simulations. For all relevant events there are plenty of data available on the storage system of the Erlangen Center for Astroparticle Physics (ECAP). For training and performance evaluation only the simulation data can be used, since real data isn't labeled.

2. Preprocessing
   The only preprocessing steps performed for this experiment are the calibration of the raw data and to to some extend also the L0 hit selection, because similar to an image preprocessing it selectively reduces noise in the data.

3. Segmentation
   To identify the signal, L0 and L1 hit detection as well as trigger algorithms are applied by either the data acquisition system for recorded data or the seatray framework for simulation data. The tools developed for this thesis perform segmentation by individual thresholds for the different feature computations.

4. Feature extraction
   The module SGFeatureExtract computes a set of specifically designed heuristic features to extract the information.

5. Feature selection
   Genetic algorithms are used to solve the optimization problem of feature selection. The individuals within the population consist of ”use or don’t use” for each feature and the fitness for each individual corresponds to the classification rate achieved with this feature set.

6. Training
   By evaluating many different algorithms using the tool Weka the Random Decision Forests algorithm was selected due to its good performance. Therefore RDF classifiers are trained on the simulation data.

7. Classification
   The RDFs can be applied to every unseen event and will classify it according to the data they were trained on.

5.2 Feature extraction

The data needs to be transformed into feature sets which allow further processing and contain the relevant information. As a short reminder about chapter 3.1, the data for one event consists of a list of hit entries and each hit entry of:

- Charge
- Time
- Line number
- OM number

OM stands for optical module, a hit is an OM detecting radiation. Line number and OM number describe the location in the detector geometry, charge and time are the observables.

5.2.1 Time binning

The data in its original form will later be referred to as “calibrated data”, since the calibration procedure has already been applied to the “raw data“ before the feature extraction starts. Because the data this form is quite complicated to treat, the first thing done was to discretize all data once more in time. This so called ”binning“ simply computes a time range from the first to the last hit and then splits this range into n_{bins} parts, so called ”bins”. If two or more hits occur during the same time bin their charges are added up and the sum is stored in that bin. The line number and OM number are untouched during that process. By this processing the data space (except for the charges) becomes finite because space and time now can only have a limited number of values.

The number of lines is n_{lines}, the number of OMs on one line is n_{OMs}. The data then can be represented by a vector of size $n_{lines} \cdot n_{OMs} \cdot n_{bins}$, with each entry storing the overall charge observed at that location during that time.

5.2.2 Storey data

The next step was to simplify the three OMs on one storey to be one detection unit. “Storey“ refers to the storey as seen in figure 3.3 with its three optical modules. The data simplification is done by summing over the three neighboring OMs and storing the sum as charge for this storey location. The data is then a vector of size $n_{lines} \cdot n_{storeys} \cdot n_{bins}$, where $n_{storeys}$ is the number of storeys per line. This form of the data will be called “basic form” when referred to later on.

With these two simplifications the data becomes comparatively easy to handle. Furthermore, instead of evaluating the positions and geometries of the detector using the geometry calibration called “alignment“, the idealized geometry with straight lines is assumed, all lines positioned as depicted in figure 3.2. All rotations are already eliminated since the OMs are aggregated to virtual storeys without direction. A “level“ of the detector refers to all storeys on a certain height.

A numbered list of all features can be found in appendix A. F_i is used to refer to the i^{th} feature of this list.

At this point the first features can already be computed:

- Maximum charge within the whole detector during all time-bins of this event (in the following called F_1)

- Level containing F_1 (this maximal charge)

With the information about F_1 and its location, the second highest charge in the neighborhood of F_1 can be searched. Neighborhood in this case is defined as (if available): One level above the level of F_1, one level below, all lines within approximately 100m radius and the levels above and below on these lines. This radius corresponds to the lines left, right, up, down and all four diagonals in the top view of the detector, see figure 3.2. The search window for the time is first set to $0.04 \cdot n_{bins}$, a number which was found by heuristics. Since the whole event time, corresponding to n_{bins} bins is typically around several $1000ns$, this number of bins should be around a few hundred ns.

For each of these locations the maximum charge within the time search window is computed. Time in this case of course are time-bins, since everything is computed on the further discretized data only. This evaluation results in several features:

- Whether the second highest charge is below or above F_1 (Possible values are $-1, 0, 1$)
- Absolute value of the time difference (in number of bins) from F_1 to the second highest charge in the neighborhood
- Average time difference from F_1 to all neighboring maxima
- Highest time difference from F_1 to a neighboring maximum
- Sum of the squared deviations of all time differences from the average time difference
- Variance of the time differences
- Average time difference to the maxima of the same level
- Absolute value of the average time difference to the maxima of the same level
- Average time difference to the maxima of the level above
- Absolute value of the average time difference to the maxima of the level above
- Average time difference to the maxima of the level below
- Absolute value of the average time difference to the maxima of the level below

The motivation for these specific features is that the sign of the bin difference to the upper and lower levels should be different for upgoing and downgoing events. The absolute values on the other hand might distinguish a particle traveling with almost the speed of light in vacuum c from a shower, because the light from a shower of course only travels with the speed of light in seawater, which is about $0.75 \cdot c$.

These features are then evaluated again with different search windows for the time. The search windows of the second search is twice as long, the third four times longer, and the fourth eight times longer. The number of bins in the search windows therefore are $0.04 \cdot n_{bins}$, $0.08 \cdot n_{bins}$, $0.16 \cdot n_{bins}$ and $0.32 \cdot n_{bins}$. The different search windows allow to capture different possible timings corresponding to different physics events. For instance should muon maxima be detected earlier than those cause by shower events.

5.2.3 Level data

In the next step the data becomes even more condensed. For all storeys of one level, the square of each storeys charge is summed up and this sum is then stored as one new value, representing the whole level. By summing over all storeys on one level the line information is eliminated. The resulting data contains only the space dimension height discretized as levels and the time information discretized into time bins. On that dataset of size $n_{storeys} \cdot n_{bins}$ the maximum in time is searched for each level. The charge and the time of the maxima are saved. The levels are separated into those with a maximum greater than the average of the maxima and those with a lower maximum.
From the set of the levels with greater maximum charges than the average, the largest connected cluster is computed. The term "connected" is weakened a bit such that one missing level is still accepted. As an example, the levels 1,2,4,5,8,15 are above the threshold. The algorithm will find the set of levels 1,2,4 and 5 with the size of the set "correctly" reported to be 5, since level 3 is assumed to be missing due to malfunctioning OMs or other reasons. Although there is a certain not to be neglected probability that level 8 also has seen parts of the event it might also just be noise near the event. But the algorithm by design tends to cut off minor parts from the signal which could already be background. The result of this segmentation procedure is a cluster containing the relevant levels the signal has passed.

The first features from these computations are

- Level containing the maximal charge

- Size of the connected level cluster

To further analyze the level cluster the upper half and the lower half of the levels in the set are compared. The times of the maxima of each level of the upper half are summed, and the same is done for the levels of the lower half. A very robust timing information can be extracted by subtracting the times of the lower half from that of the upper half. An upgoing signal necessarily produces its signals first in the lower levels of the set and later in the higher ones. The times (measured in time-bins) of the maxima of the lower levels therefore will be smaller than the times of the upper half. The result of the of the subtraction will be positive, indicating a very high probability for an upgoing signal.

Written into the feature vector are

- Time difference of the upper half minus the lower half
- Normalized time difference, computed as the time difference divided by the number of levels in the cluster

To compensate the tendency of the cluster segmentation algorithm to harshly cut off levels one level is added to the cluster on both sides if possible. If the cluster already contains the lowest or highest level only one level is added to the other side. In the previous example the cluster contained level 1 to level 5. In this case only level 6 can be added, since the level 1 already is the lowest level. Then the same time differences are computed again for the extended level cluster.

Additionally computed on these data:

- Sum of the upper and the lower times
- Sum normalized by the number of levels in the cluster
- Variance of all times in the cluster

On the extended cluster the features are computed once more, but with weighted summation over the times. The weight for every time value is the square root of the corresponding charge. The idea behind this is to emphasize the better observed parts of the event.

For this computation the additional features are:

- Upper sum normalized to the number of levels in the upper cluster half
- Lower sum normalized

5.2.4 Line data

For the next features the data in the basic form (from 5.2.2) is used again as a starting point. Instead of eliminating the line information and storing levelwise, this time the level information is dropped. This is achieved by summation over all levels of each line, storing only the sum for each line. The resulting data is of size $n_{lines} \cdot n_{bins}$.
Just as for the levelwise data, the charge maxima over time for each line are now computed. From these a threshold is generated as the average of all line maxima excluding the highest one. The highest line charge maximum is excluded since directly up or downgoing events otherwise lead to an instable threshold computation, very often resulting in just the most energetic line peaking above the threshold. With this threshold the set of lines is again split into lines with higher or lower maximal values. Analogous to the levelwise computation, the search for a connected cluster is the next step. To achieve this for the more complicated two dimensional line geometry shown in figure 3.2, "connected" is defined as the lines left, right, up, down and diagonal next to each other in the top view of the detector.

The features extracted by these computations are

- Number of lines above the threshold
- Size of the connected line cluster

5.2.5 Time independent

For a third time the computation starts at the basic form of the data (see 5.2.2). With all space dimensions already been treated, this data processing removes all timing information. The detector geometry is kept and a summation for each line and level is done over the time bins. The resulting data is of size $n_{lines} \cdot n_{levels}$ and exposes the overall shape of the event. For this data the term neighborhood is the same as in 5.2.2, except that there is no search window in time anymore.

The features computed on this dataset are:

- Maximal charge seen by one storey during the whole event
- Level which has detected the maximal charge
- Average charge of all neighboring storeys around the maximal charge
- Ratio of the maximal charge to the average charge around it

- Fraction of storeys above the line threshold (computed analogously to the threshold for the line data)
- Fraction of storeys above the average storey charge

5.2.6 L1 filtered data

All features described until now are derived from the full calibrated data set, described at the beginning of chapter 5.2. The SEATRAY framework itself already contains several data processing techniques. The so called L1 hit selection described in chapter 3.1 already provides a decent way to select hits with a high probability to contain physically relevant information. Although the whole feature extraction was not in any way designed to operate on these sparsely filled data, the same features computed only for the preselected L1 hits contain a similar amount of information than those computed from the full data. Therefore all features are computed on both to be able to provide the highest accuracy and robustness possible. One feature is added exclusively for the L1 computation.

- The total number of L1 Hits for this event

5.2.7 Feature summary

In total all computations result in 137 features.
These features could be transformed to an advantageous space, for instance by linear combination. Very often a principle component analysis (PCA) is used for that purpose. To reach a space where the classes might be better separable also an extended feature set was computed composed of the old features and every feature added with, subtracted from, multiplied and divided by every feature.

5.3 Feature selection

The goal of a good feature selection is to keep those features containing significant or exclusive information while eliminating features carrying no statistical relevant information. Features with no useful information still have to be processed and therefore might not only not aid the classification but in fact decrease the classification performance.
To find the best possible subset of features a genetic algorithm (GA, see chapter 4.2) was used in this thesis. An already existing suitable implementation called ”galib” by Matthew Wall [Wal96]

was incorporated into the source code. A steady state GA with a one-dimensional string genome was used. The individuals are strings of plain one bit "use or don't use" choices, and the fitness function is the classification rate multiplied by the acceptance rate.
The relevant parameters to modify the behavior of this implementation and the used values are

- Population size: 100
- Probability of individual replacement: 50%
- Number of Generations: 200
- Probability for random mutations: 1%
- Probability for a genetic crossover: 80%

The algorithms implemented in Weka were also tried for this task.

5.4 Classifier training

Once the features are computed they are stored in a file together with the correct label for every event. Using these files the training of the RDF classifier can start. As input the tool SGClassify uses the generated .dat files and a configuration file with the ending .sgc. Its output is a random decision forest (RDF) saved as .rdf file. The simple .rdf file format is not compatible with other software and designed for later use with SGClassify only. The values specified in the config file determine the properties of the RDF. In the config file a '#' character marks the rest of a line as a comment. Each value is followed by at least one empty space and exactly one value. A config file usually specifies the following values. (some example values are already set):

- `folder` data/100kHz/
- `verbose` 1
- `saveForestToDisk` 1
- `filenameToSave` Forest.rdf
- `filenameToLoad` Forest.rdf
- `QualityCriterion` 0.0

- `NumberOfTrees` 100
- `RateOfVariablesPerTree` 0.2
- `RateOfEventsPerTree` 0.3

The `folder` variable determines the path to the data folder. The folder must end on a '/' character. Within this folder any files with the ending .dat are processed. It is the responsibility of the user to supply only compatible data with this folder. One possibility to provide compatible files is to separate the data for training and for classification. To achieve this the files must contain either the phrase "Train" or "Test" within the filename. The test data can be provided labeled or unlabeled. If test data contains one less value per line than the training data, the test data is automatically considered to be unlabeled. This automatism requires some caution by the user. As an alternative to the train/test separation all data contained in .dat files can also be used as training sample. A very important rule is that the class number (if present) must be the first number of each line. If the data sets were created by SGFeatureExtract the data will automatically be formated and separated correctly.
The parameter `verbose` enables most of the programs output. In most cases the user will want to enable this. `saveForestToDisk` triggers saving a generated RDF to a file. The filename is defined by the parameter `filenameToSave`. If the tool is started by a binary for loading the RDF instead of creating a new one the filename of the RDF is specified by `filenameToLoad`. The parameter `QualityCriterion` causes the evaluation to only consider those events where the individual trees of one RDF showed at least an agreement of the QualityCriterion rate. All events which do not meet this criterion are discarded, creating a rejection class for the classification. If as many event candidates as possible are to be found this parameter should be set to 0.0. If the purity of the found sample is to be increased the parameter QualityCriterion should be set as close to 1.0 as desired. It is relevant to note that with only 20 individual trees a QualityCriterion of 0.95 already equals 1 since no agreement values between 0.95 and 1 are possible.

The `NumberOfTrees` simply determines the actual number of individual trees within the RDF. `RateOfVariablesPerTree` determines what fraction of the total set of features is used for each decision tree. `RateOfEventsPerTree` sets the fraction how many of the training events are used for each tree. `NumberOfTrees`, `RateOfVariablesPerTree` and `RateOfEventsPerTree` are the key parameters for the accuracy and the runtime of the program. The runtime of the program scales linearly with `NumberOfTrees` until, of course, the limit of the main memory is reached. A very high value will not decrease the classifica-

tion performance but impairs the runtime and often does not significantly increase the performance. A very low value on the other hand considerably limits the achievable performance. High values for `RateOfVariablesPerTree` tend to improve the classification for the worst performing classes, while lower `RateOfVariablesPerTree` values of course are beneficial for the runtime. Especially if many trees are trained and a high minimal agreement of the trees is demanded lower `RateOfVariablesPerTree` may lead to an unstable performance. `RateOfEventsPerTree` is recommended by the library to be set between 0.15 and 0.66. It is again a tradoff between performance (with a saturation) against runtime.

Additionally to these parameters the config file may also specify:

- `homogenTrain` 1
- `minNumber` 500
- `homogenTest` 0
- `useClassMapping` 0
- `classMap`
- `useFeatureSelection` 0
- `selectFeatures`
- `NumberOfEventsToSkip` 0

The parameter `homogenTrain` usually should be set to 1. It enables a random resampling of the training data to obtain the same number of events for every class. How many events per class is determined by the minimal number of events of one class as default. This step is necessary to obtain the same prior probability for each class for the training. The RDF classifier is very sensitive to this property. To ensure fair conditions all results presented are computed with homogenized training sets. If different event probabilities are to be exploited, the parameter may be set to 0. But in extreme cases of, for example 10000 instances for class A and 10 instances for class B very likely a RDF will be created, which classifies every possible signal to be of class A. The `minNumber` parameter is an addition to the homogenTrain parameter. It prevents the classes to be reduced to very few instances in the case of one single class being represented by only a small number of events. By default all classes are sampled down to the minimal number of instances of one class. If `minNumber` is set to a specific number all classes are at most sampled down to that number ignoring classes with fewer instances. `homogenTest` analogously

subsamples the test data just like `homogenTrain` works for the training data. There is no `minNumber` available for the test data. The parameter `useClassMapping` enables a mapping for the class numbers. This is useful if the data contains more classes than actually desired for the classification. In the case of this thesis, the data are at least separated into muons, upgoing and downgoing neutrinos and shower events. For a classification considering atmospheric muons as one class and everything else as a second class this mapping can be used. The actual mapping is specified by the vector `classMap`. `classMap` is a list of integer numbers separated by commas. The position in the list specifies which original class is mapped to the integer at this position as a new class number. `UseFeatureSelection` enables or disables a selection of features specified in the variable called `selectFeatures`. The format of `selectFeatures` is the same as for `classMap`, but it consists of zeros and ones. The digit '1' at a certain position n means, that the n th feature will be used. A '0' means that this feature is not considered at all. The parameter `NumberOfEventsToSkip` allows data to be skipped already at reading time. These data are not processed further in any way. The skipping is implemented as reading one feature vector and then ignore the next `NumberOfEventsToSkip` ones. For very large data files this turns out to be quite handy.

If everything is specified accordingly, the tool will create a .rdf file which contains the fully build random decision forest. A demonstration config file can be found in the appendix E.

5.5 Classifier usage

The actual application of the classifier is rather simple. One needs to choose a binary, usually "SGTrainAndUse" or "SGLoadAndUse", provide the feature set(s) to be classified and the QualityCriterion parameter may be specified according to the individual needs. As output the resulting class number is provided as well as the rate of trees which agreed on this class. One must consider that this rate is not an absolutely accurate measurement for the probability that the class decision is correct, but they are strongly correlated. It is however possible that a tree built on weak features disagrees with those built on stronger features. One has to keep in mind that not every feature can be considered strong for every possible classification. That means a decision agreement of all trees is only possible if those built exclusively on weak feature by chance hit the correct class too. The more classes there are to choose from, the less likely the correct one is hit by such a tree. This could be improved but not relieved if features which can be considered weak for one specific classification are ignored. Detailed results for different quality criterion parameters can be found in the results section.

For the evaluation of the classification performance several approaches exist. They differ in how to obtain the results and even more in how to interpret them. We first evaluate the different implemented ways offered to classify data. All binaries mentioned can be found in the folder "bin". The syntax is "./path/to/binaryfile path/to/configfile.sgc".

5.5.1 Test and Training sets

If the data sets are computed using SGFeatureExtract two .dat files are created. One filename contains the term "Train". This file contains 80% of the computed data and the other file with "Test" in its name contains the remaining 20%. The split is done by simple counting. If a labeled set is constructed by hand or by a different algorithm, one must note that the class number must be the first in each line. All following numbers are the features for the event. If the test set is labeled and the verbose property is set by the config file, the algorithm will output the relevant evaluation data in the end. A detailed explanation of the evaluation output can be found in chapter 6.2.

SGClassify can read the training and test data and will try to automatically recognize whether or not the test data set is labeled. If the numbers per line match, the data is supposed to be labeled. If the test set contains one less variable than the training set, the data is supposed to be unlabeled and any other case results in an error message. The RDF is trained on the training set and then evaluated on the test set. The binary for this purpose is called "SGTrainAndUse".

If the algorithm is told to save the constructed RDF to a file, this can be loaded and reused. The algorithm then still reads in data, but loads the RDF from file and evaluates on the test set. The binary for this purpose is "SGLoadAndUse".

5.5.2 Cross validation

Especially if one wants to have reliable performance measurements or if there are only few data instances available, Cross Validation is a good choice. As already stated in the methods section, k-fold cross validation in the narrow sense partitions the data only once into k parts and covers all events exactly once in k runs. This cross validation implementation takes all data (in this case training and test sample) and several times randomly resorts the data into a new training and test set of fixed size. These two sets are always disjoint. The classifier is trained upon the new training set and its performance is evaluated on the new test set. The next random sorting is performed, the next training, the next evaluation. This is done for a desired number of times

(default is five). At the end the performance for all runs is taken into account. This can be invoked by the binary "SGCrossValid".

The ratio of training to test set size is variable, default here is again $4:1$. One may also use this method in an extreme fashion called "Leave-One-Out". The Test set in this case consists of exactly one instance, the training set out of all others. This is repeated for every single instance and the performance for all instances is accounted. Since the datasets evaluated tend to be huge, Leave-One-Out is definitely not recommended! The binary is called "SGOneOutCross".

5.5.3 SEATRAY module

To be able to use SGClassify within the Seatray framework, the functions are also incorporated into a Seatray module called SGClassify. As input for this module an already trained RDF and a quality criterion must be specified. The features for the event must already be present in the frame as a vector called SGFeatures, usually written by the module SGFeatureExtract used prior to SGClassify on this frame. The result of the classification will afterwards be written into the frame too. The classification result is called SGClass and the fractions of trees voting for each class are stored as vector called SGVotes. This information can be evaluated by follow up modules to invoke the correct reconstruction depending on the SGClass result.

5.5.4 Others

The binaries "SGOptimizeGA" invokes the genetic algorithm searching for the optimal feature selection for the current problem and "SGTrainAndQuality" computes the given problem for many different parameters and for every run outputs performance data displayable with gnuplot [TW10].

Chapter 6

Results

6.1 Classifier and Feature selection

A comparison of various classification algorithms was performed using Weka on an early 23 feature set. The most relevant features were already contained in this set, but no background noise has been used for this evaluation. A small excerpt for selected classifiers with their names in Weka is shown in table 6.1. Except for the Multilayer Perceptron, all other algorithms above

Method	Classification rate
RDF	0.88
Random Tree	0.82
Multilayer Perceptron	0.84
AdaboostM1	0.69
Naive Bayes	0.75
SVM	0.77
Stacking	0.51
ZeroR	0.51
SimpleCART	0.84

Table 6.1: Classification rates for some classifiers obtained with Weka

80% are tree-based, with the best tree based algorithm being RDF. The Multilayer Perceptron, a neural network algorithm, was not considered due to the by far longer runtime for comparable performance. Therefore the RDF algorithm was chosen.

The two feature subsets S1 and S2, which are the results of genetic algorithm optimization with the parameters stated in chapter 5.3 are contained in appendix B. S1 was the result for a two class muon identification task on simulation data with 100 kHz and a RDF with 50 trees,

0.1 features per tree rate and 0.15 events per tree rate. The meaning of these parameters is explained in chapter 5.4. The numbers are lower than in the performance chapter, because one single GAOptimization with these parameters already takes several days on an Intel Core 2 Quad Q6600. S2 is the outcome for a two class Shower identification on the same data and the same RDF parameters. Table 6.2 contains three runs for the two tasks and for the three feature sets S1, S2 and the full feature set. The numbers show that the "optimized" feature sets have no significant influence on the outcome, but allow roughly the same performance with considerably less features. Some previous feature selections have shown greater improvement, but never above 4% in classification rate. On the other hand a feature set which was optimized for a different task in many cases decreases the performance.

Task	S1	S2	Full feature set
Muon id. 1	0.876	0.857	0.880
Muon id. 2	0.881	0.864	0.879
Muon id. 3	0.878	0.866	0.874
Shower id. 1	0.735	0.762	0.763
Shower id. 2	0.749	0.755	0.751
Shower id. 3	0.747	0.759	0.758

Table 6.2: Classification rates for different feature sets

The feature selection using genetic algorithms unfortunately did not result in acceptable feature subsets. Another way to find optimal subsets was to use algorithms built into Weka. But the results were either unsatisfying because of a decreasing classification accuracy for the "optimal" features or the memory consumption of the built in algorithms exceeded the maximal memory capacity of the machines running Weka although the available memory for the java virtual machine had already been manually increased to the limit.

At the end of the day no optimal feature subset with respect to performance and robustness could be determined which surpasses the full feature set for other classification experiments than exactly the one it was selected for. Therefore all results have been computed on the full feature set to exclude the possibility of an unfit feature set for a specific task.

6.2 Performance

All presented performance measurements are done with 100 kHz white noise background if not stated otherwise. The 100 kHz training data sample contains:

- 583270 Atmospheric muons

- 26144 Neutral Current (NC) Showers
- 671480 Upgoing neutrinos
- 543848 Downgoing neutrinos

The independent test sample contains one fourth of these numbers for each class. Due to the requirement of a homogenized class distribution for the RDF, only a randomized homogeneous subsample was used for each RDF training.

6.2.1 Two class muon suppression

The two class muon suppression is clearly motivated since most events by far are undesired atmospheric muon events. Current ANTARES neutrino point source analyses according to the last estimate still contain a contamination of about 40% downgoing muons. To suppress these events the task is to distinguish

- Class number 0: Atmospheric muons
- Class number 1: Upgoing neutrinos, downgoing neutrinos and shower events

For such a two class problem a quality criterion of 0.5 (at least 50% agreement of the trees) is implicitly present, since the majority of the trees decides between the two possible outcomes. With no additional quality criterion set the confusion matrix seen in table 6.3 is the result for a RDF trained with the parameters: Number of trees ($ntrees$) = 250, Rate of Features used per Tree ($rvar$) = 0.3 and a Rate of Events per Tree ($revt$) = 0.4. The overall classification rate is 0.882.

	0	1	R	
0	3283	269	0	0.924
1	555	2871	0	0.838
	0.855	0.914		

Table 6.3: Confusion matrix for muon identification; Class 0 = Muons, Class 1 = Neutrinos & Showers

The confusion matrix in table 6.3 shows the behavior of the classifier if all events are evaluated, regardless of their classification agreement. In many real world applications one will want to have a certain classification rate or maybe purity in the accepted sample. Figure 6.1 shows the dependency of the classification rate from the quality criterion (the tree agreement) for several

forest sizes. The starting point at 0.5 is due to the two class nature of this classification task. The first thing that meets the eye is the stepwise increase for RDFs with only 5 or 10 trees. The simple explanation for this is that five trees can only agree to 20%, 40%, 60%, 80% or 100%. A quality criterion in between will not alter the outcome. The starting point of each plateau corresponds to the true performance, the ramps between these plateaus are in fact steps rather than ramps. The performance development for 100, 1000 and 5000 trees is almost the same, except when it comes to very high accuracies. A RDF with 5000 trees will more likely contain several "weak" trees almost choosing randomly, than a RDF with 100 trees. To demand an agreement of 99% therefore is a more severe restriction for a huge RDF than for a mid-sized one. If no event is able to fulfill this demand any more, the performance drops to 0.0, not to 0.6 as suggested by the figure. The y-axis is fixed to 0.6 to allow a better judgement of the individual performances.

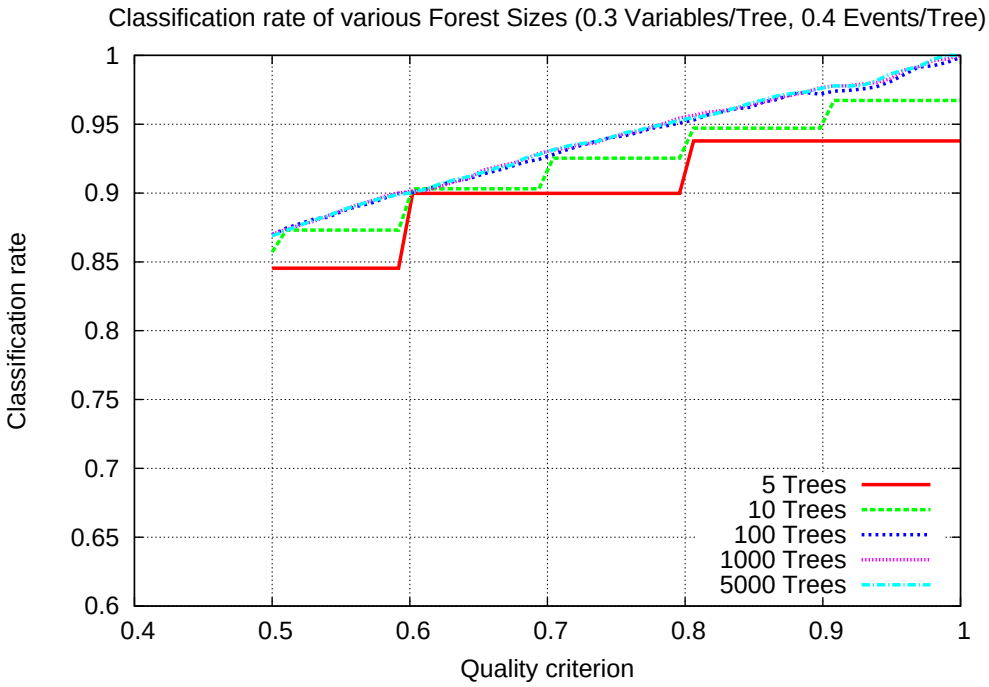

Figure 6.1: Muon identification: Classification rate versus Quality criterion for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

Figure 6.2 displays the effect of an increasing QualityCriterion parameter on the acceptance rate or efficiency of the classifier. Expectedly one can observe that the higher the demanded

quality is, the fewer events can satisfy this demand. The trend for all RDF sizes is a steepening of the curve with a higher QualityCriterion. Again the number of trees does not significantly contribute to the performance any more after a certain point of between 100 and at most 1000 trees. Unfortunately, one can see the acceptance drop to just a few percent if an agreement of 100% is demanded for the RDF.

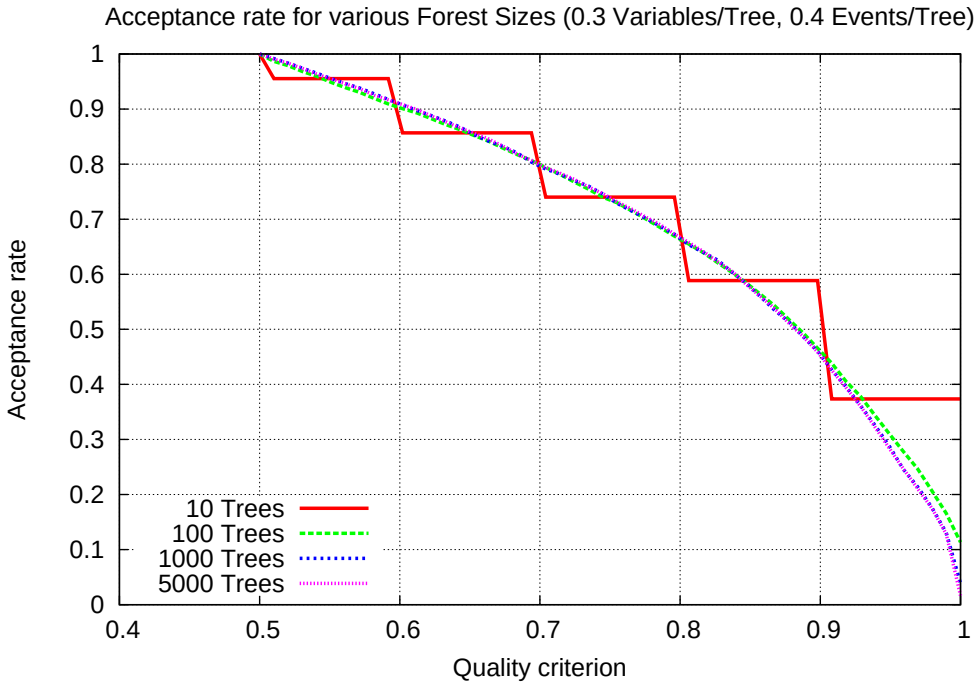

Figure 6.2: Muon identification: Acceptance rate versus Quality criterion for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

The combined version of the Classification rate and the Acceptance rate can be seen in figure 6.3. This is the best way of visualizing the actual performance at one glance. The QualityCriterion in this case is the parameter which determines the location on the graphs. A QualityCriterion of 0.0 to 0.5 corresponds to the starting points at the right at an Acceptance rate of 1. With an increasing QualityCriterion parameter the points move to the top left. A classification rate of around 90% to 95% is reached very quickly, but the curve flattens drastically when nearing a 100% classification rate.

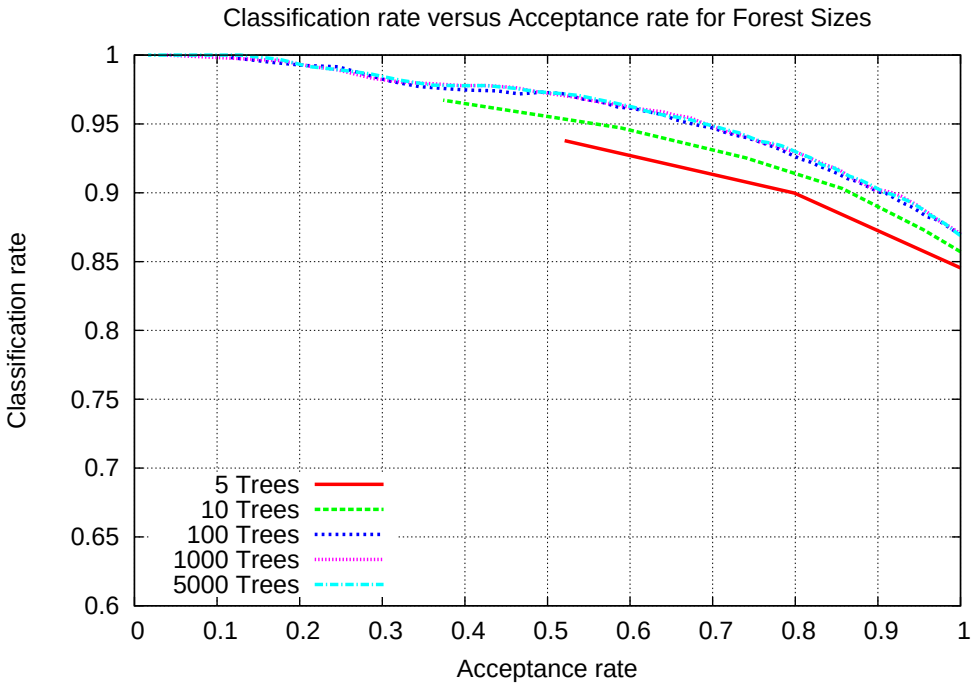

Figure 6.3: Muon identification: Classification rate versus Acceptance rate for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

Figure 6.4 compares the Classification rate for many forest sizes, each for 100% Acceptance rate. The observable performance increase with the number of trees lasts until 50-100 trees, any more trees have a negligible influence on the performance for this task.

Figure 6.5 shows a similar setup as Figure 6.4, but this time the impact of different feature rates is compared. Again very small values lead to a worse performance. From 0.1 on the performance stays roughly the same with a slight peak at 0.4. In contrast to the number of trees, the fraction of the features per tree therefore has an optimal value. This is especially relevant since low values lead to considerably shorter computation times.

The remaining performance dependence on the number of events per tree is visualized in figure 6.6. As always the performance is worse for too small values but stabilizes, here for values between 0.2 and 0.4. The absolute maximum is reached at higher values, but the difference between values greater than 0.4 is almost neglectable compared to the significantly increased runtime for higher values.

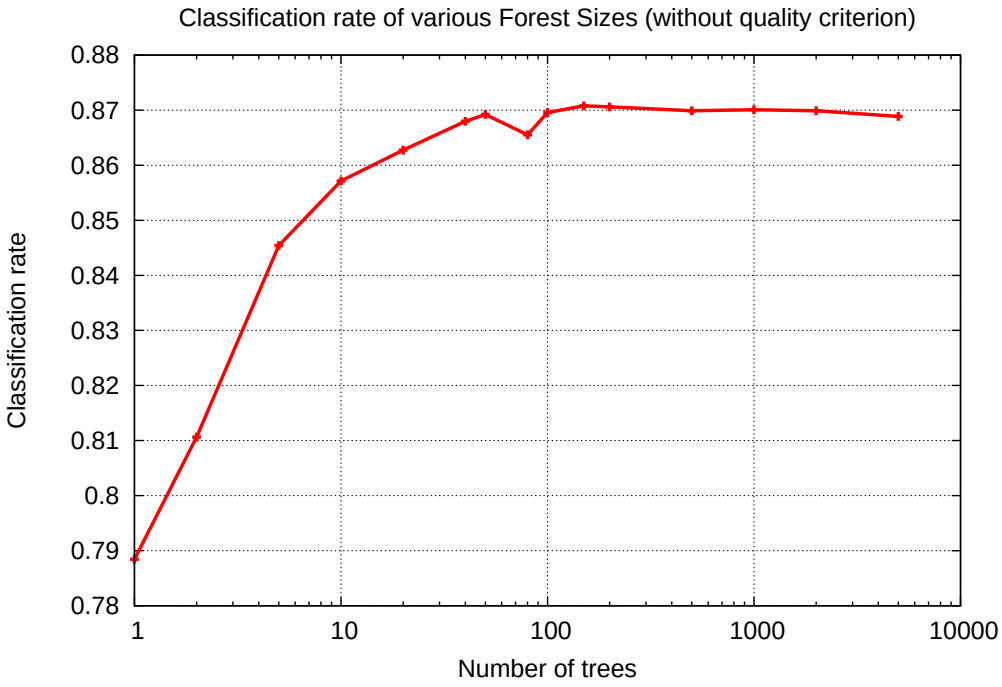

Figure 6.4: Muon identification: Comparison of different forest sizes, 0.3 Features/Tree rate, 0.4 Events/Tree rate

The ROC space usually allows a decent comparison of different classifiers for diverse tasks. Here some restrictions result in an uncommon ROC curve. The implicit quality cut for each task with n classes of $\frac{1}{n}$ prevents the classifier to reach many false positive classifications. That is why the x-axis only reaches until to 0.12 instead of the default 1. Rejected instances do not influence the ROC curve at all, since these result in false or true negatives, which don't affect the corresponding ROC curve. With that being said the ROC curve for each RDF starts at the top right for a QualityCriterion of 0.0 and continues to the lower left. A comparison of ROC curves makes sense against other tasks or different classifiers.

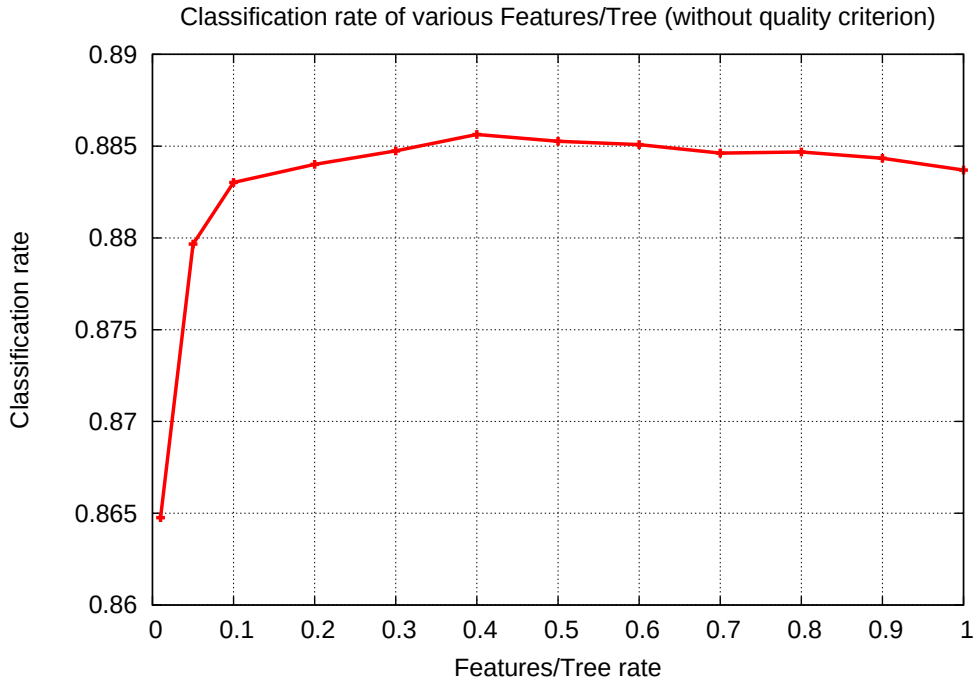

Figure 6.5: Muon identification: Comparison of different Features/Tree rates, 250 Trees, 0.4 Rate Events/Tree

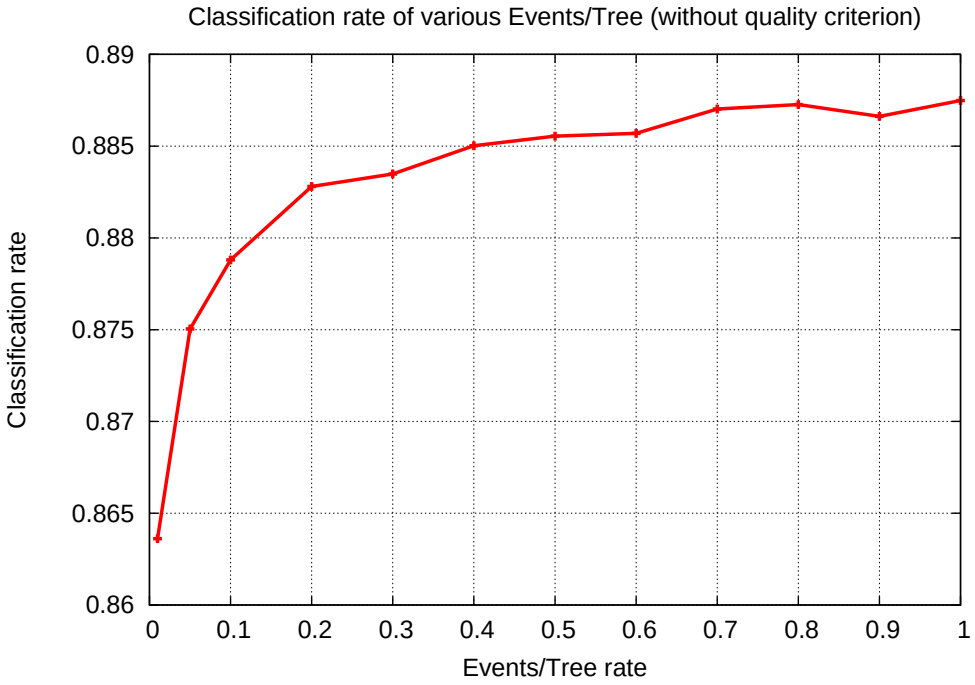

Figure 6.6: Muon identification: Comparison of different Events/Tree rates, 250 Trees, 0.3 Rate Features/Tree

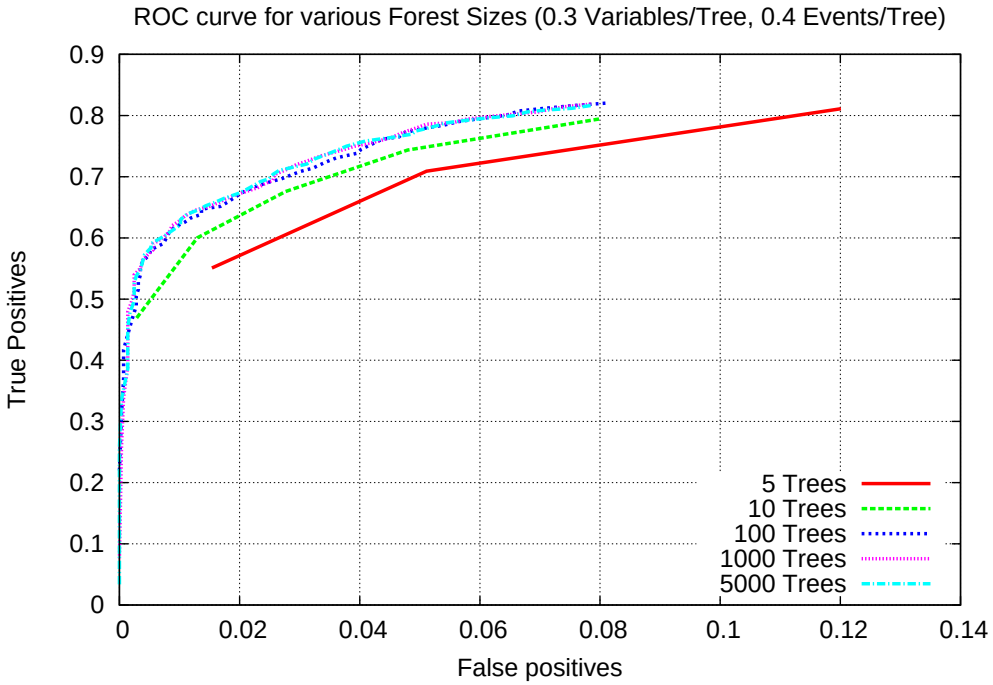

Figure 6.7: Muon identification: ROC curves for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

6.2.2 Two class shower identification

Besides others one of the goals set for this thesis is to detect shower events. If shower events for instance are not filtered from the input for track reconstruction algorithms they often simply return the best fitting track for this shower. With results that are in parts based on such events the significance of the analysis of course decreases. On the other hand some analyses particularly need shower events.
To identify the shower events the classes are

- Class number 0: Atmospheric muons, Upgoing neutrinos, Downgoing neutrinos
- Class number 1: Shower events

With no additional quality criterion set the confusion matrix in table 6.4 is the result for a RDF trained with the parameters: Number of trees ($ntrees$) = 250, Rate of Features used per Tree ($rvar$) = 0.3 and a Rate of Events per Tree ($revt$) = 0.4. The overall classification rate for this setup is 0.831.

	0	1	R	
0	2713	536	0	0.835
1	568	2698	0	0.826
	0.827	0.834		

Table 6.4: Confusion matrix for Shower identification; Class 0 = Muons & Neutrinos, Class 1 = Showers

The classification rates versus the quality criterion shown in figure 6.8 shows that the achieved performance is well below the one shown in figure 6.2 for a muon identification, but still clearly above 80%. The behavior for various forest sizes is similar to the previous task for larger forest sizes, but forest with only few trees offer a considerably lower classification rate. Small values decrease the performance, very high values only increase the runtime.

Figure 6.9 depicts the Acceptance rate versus the QualityCriterion parameter. The overall tendency is similar as for the previous task, but an extremely significant difference is that there are absolutely no events left for very high tree agreement demands. Even for the 100 tree case almost no event meets the 100% demand.

The combined information can be seen in figure 6.10. This plot reveals the worse performance compared to the muon identification task in chapter 6.2.1, especially for high acceptance rates. Nevertheless the performance seems suited to identify showers with a high precision.

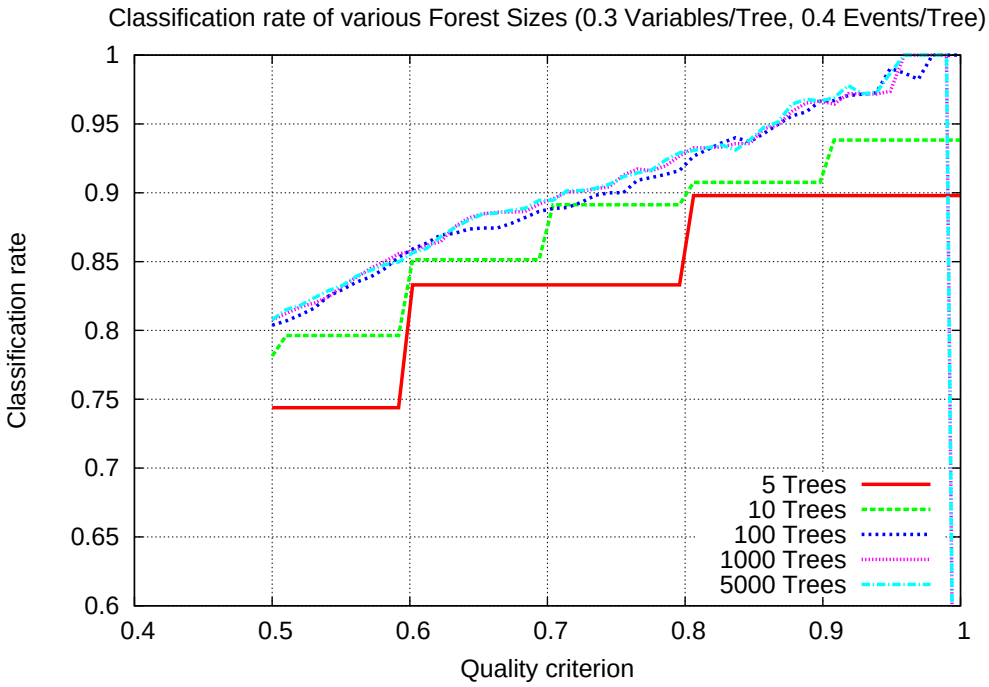

Figure 6.8: Shower identification: Classification rate versus Quality criterion for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

Since the dependencies of the performance on the number of trees, on the rate of features per tree and on the rate of events per tree looks very similar to the previous task, these plots are not described here and can be found in appendix C.

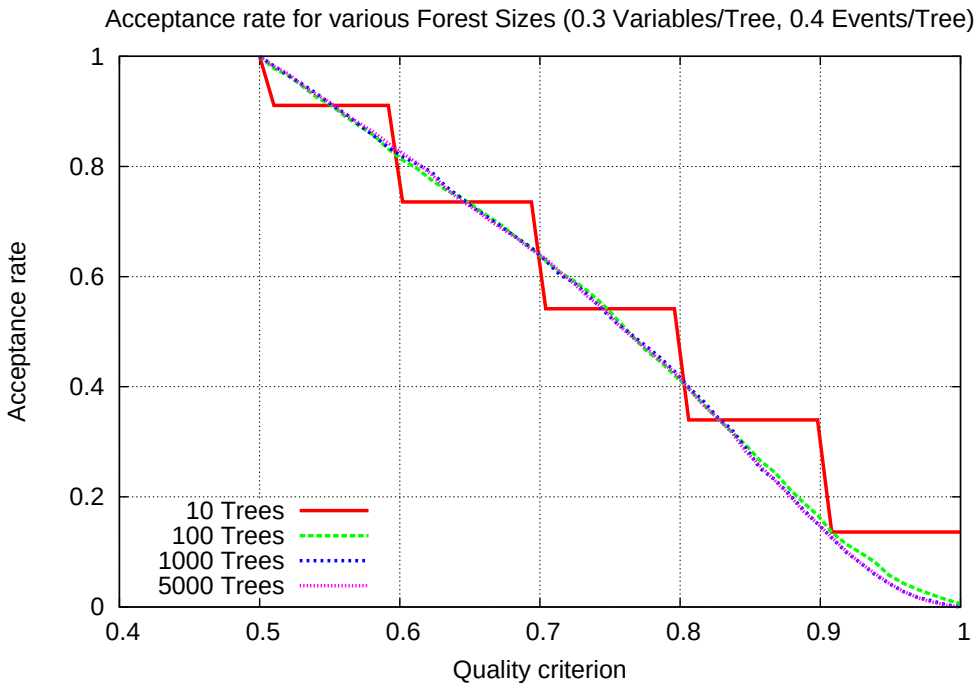

Figure 6.9: Shower identification: Acceptance rate versus Quality criterion for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

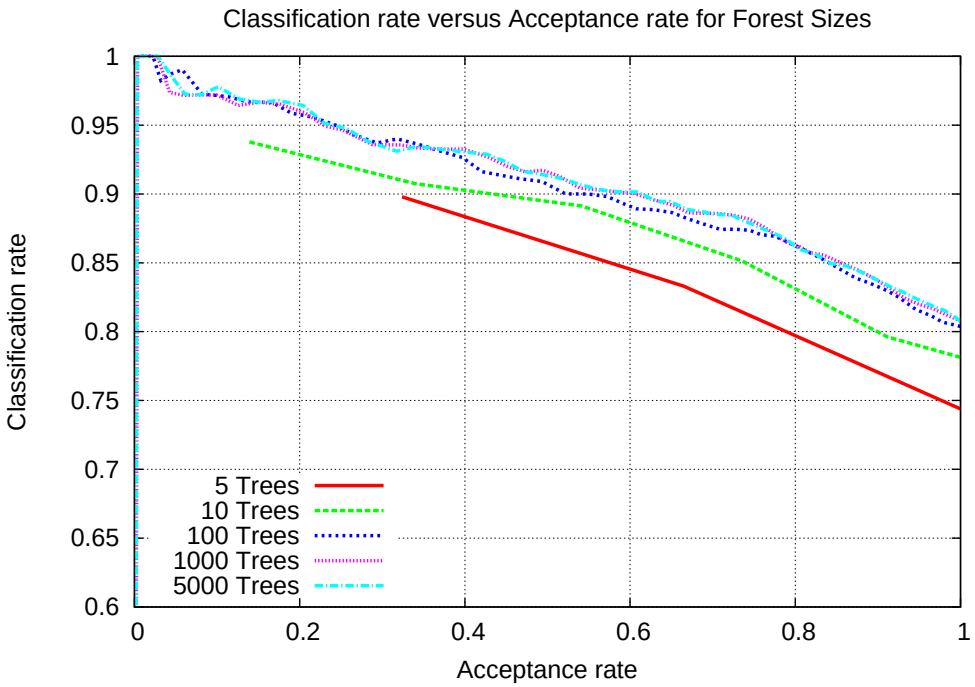

Figure 6.10: Shower identification: Classification rate versus Acceptance rate for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

6.2.3 Two class up/down identification

A decently working separation into up- and downgoing signals would already allow a filtering of many undesired signals. Although downgoing neutrino signals would be lost by a simple downgoing rejection, for many purposes a very pure upgoing neutrino sample could more than justify this. Additionally many algorithms are designed to work on upgoing neutrino signals.

For this setup the classes are:

- Class number 0: Atmospheric muons, Downgoing neutrinos
- Class number 1: Upgoing neutrinos

With no additional quality criterion set the confusion matrix in table 6.5 is the result for a RDF trained with the parameters: Number of trees ($ntrees$) = 250, Rate of Features used per Tree ($rvar$) = 0.3 and a Rate of Events per Tree ($revt$) = 0.4. The overall classification rate is 0.919.

	0	1	R	
0	10077	1020	0	0.908
1	782	10350	0	0.930
	0.928	0.910		

Table 6.5: Confusion matrix for Up/Down identification; Class 0 = Muons & Downgoing Neutrinos, Class 1 = Upgoing neutrinos

The figures are analogously to the previous tasks. Figure 6.11 shows the dependence of the Classification rate on the quality criterion, again for multiple forest sizes. The up/down classification start off with very good 92% without a quality criterion for the relevant forest sizes and increases steadily to 100%. This is significantly better than for Shower or even muon detection.

The Acceptance rates versus the quality criterion can be seen in figure 6.12. The strong drop in acceptance is at high QualityCriterion values compared to the other tasks, indicating a secure classification for many events. As always the forest size has only minor influence, but it's worth to note that for 100 trees 15% of all events are classified correctly with 100% agreement, a drastically better performance than for the other tasks.

The crucial Classification rate versus Acceptance rate plot is seen in figure 6.13 confirming the classification performance for this task.

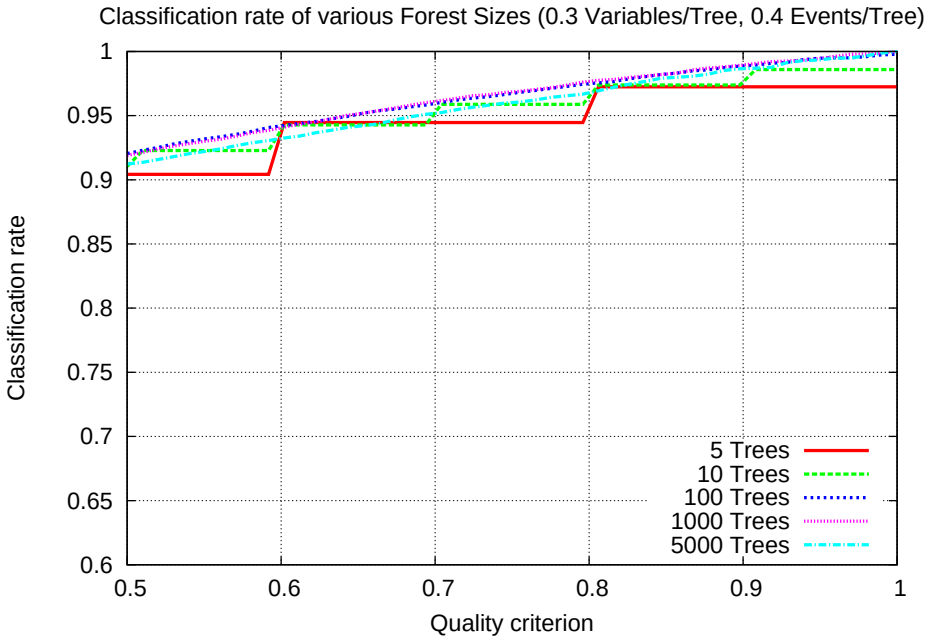

Figure 6.11: Up/Down identification: Classification rate versus Quality criterion for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

Since the dependencies of the performance on the number of trees, on the rate of features per tree and on the rate of events per tree looks almost exactly like for the other tasks but shifted to a higher classification rate, these plots can be found in appendix C.

Up to this point from the tested two class tasks this classification by far works best. A different setup for up/down classification without downgoing muon neutrinos is evaluated in chapter 6.6 for a comparison.

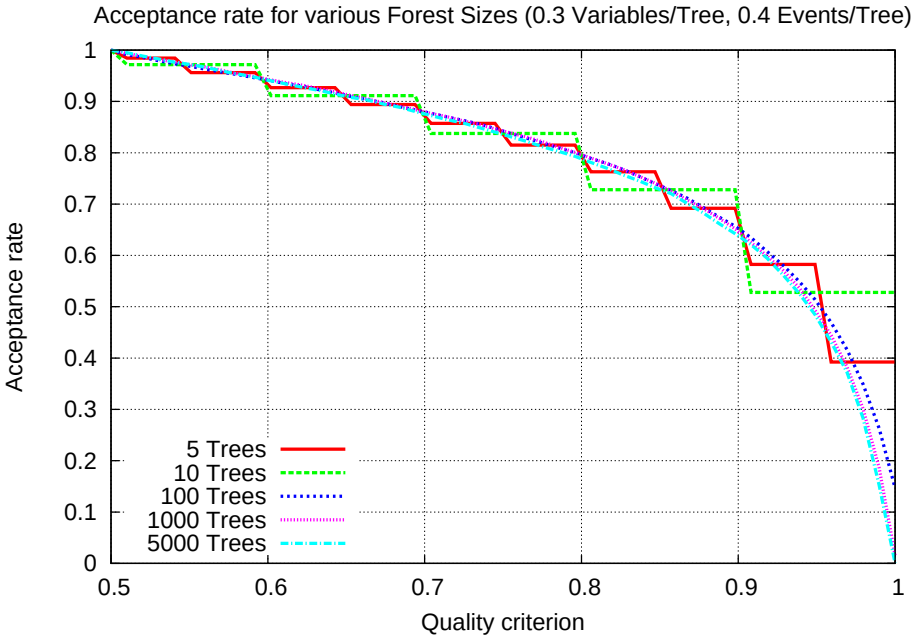

Figure 6.12: Up/Down identification: Acceptance rate versus Quality criterion for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

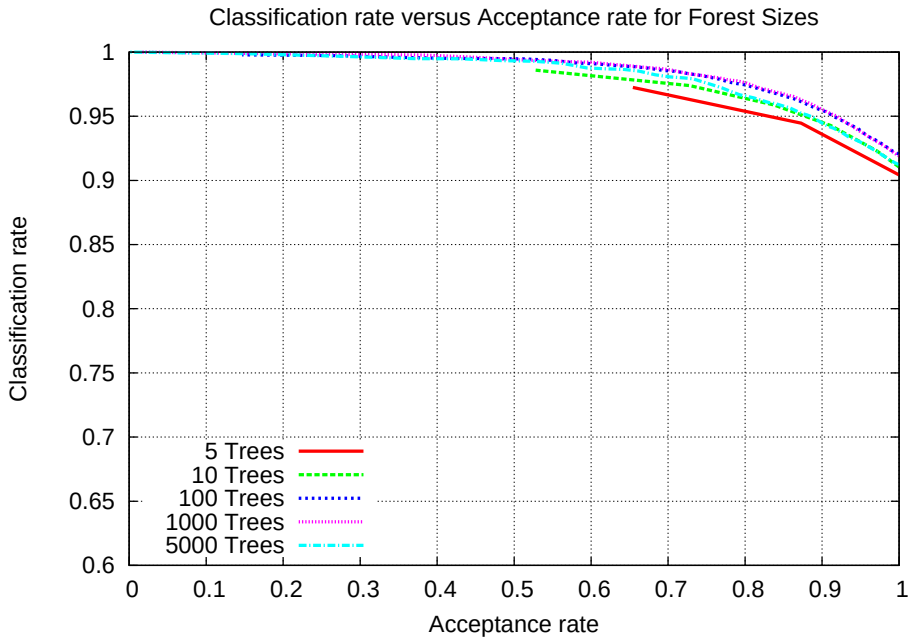

Figure 6.13: Up/Down identification: Classification rate versus Acceptance rate for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

6.2.4 Four class particle identification

The four class particle selection is a relatively tough task. Not only does a classification with four classes by definition contain more options to deliver a wrong result. Furthermore the two classes 0: (downgoing) atmospheric muons and 3: downgoing muon neutrinos produce a very similar signal in many cases and therefore have an increased intermixture rate. If this classification performed as well as the two class tasks, one would not need a task specific classification but would simply use one classifier for all tasks.

To identify all events separately the classes simply are:

- Class number 0: Atmospheric muons
- Class number 1: Shower events
- Class number 2: Upgoing neutrinos
- Class number 3: Downgoing neutrinos

For such a four class problem a quality criterion of 0.25 (at least 25% agreement of the trees) is implicitly present, since the majority of the trees decides between the four possible outcomes. With no additional quality criterion set the confusion matrix in table 6.6 is the result for a RDF trained with the parameters: Number of trees ($ntrees$) = 250, Rate of Features used per Tree ($rvar$) = 0.3 and a Rate of Events per Tree ($revt$) = 0.4. The overall classification rate was 0.724.

	0	1	2	3	R	
0	2659	169	27	401	0	0.817
1	308	2307	433	218	0	0.706
2	58	575	2495	245	0	0.740
3	591	276	324	2061	0	0.633
	0.735	0.693	0.761	0.705		

Table 6.6: Confusion matrix for Shower identification; Class 0 = Muons, Class 1 = Showers, Class 2 = Upgoing neutrinos, Class 3 = Downgoing neutrinos

As always the dependence of the classification rate on the quality criterion is analyzed first. It can be seen in figure 6.14. Obviously the classification rate starts at a worse performance than for all two class problems. The influence of the number of trees on the performance did not change dramatically, but similar to the Shower identification in section 6.2.2 5 and 10 trees offer

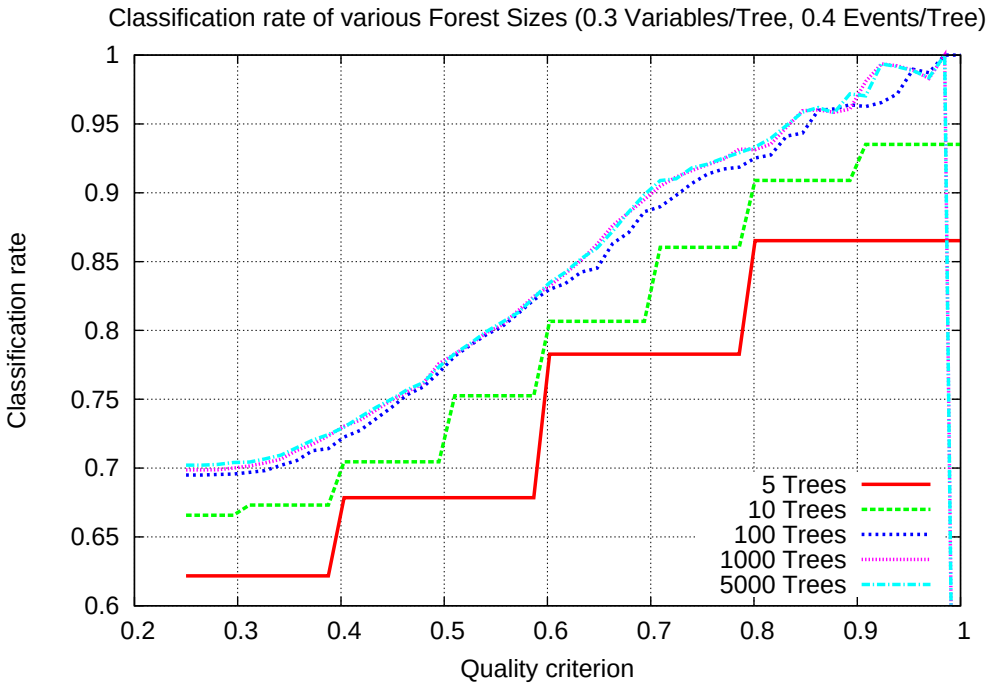

Figure 6.14: Particle identification: Classification rate versus Quality criterion for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

a poorer performance. Against expectations, the classification rate hits 100% almost at the same QualityCriterion as for some two class tasks.

The dependence of the acceptance rate on the quality criterion shown in figure 6.9 at least does not immediately drop heavily. This means that only few events are classified almost undecided by a slight majority. But where two class problems start with an Acceptance rate of 1 at a QualityCriterion of 0.5, the four class particle identification already has to reject around 25% of all events. This is one of the points where the different difficulties of two and four class tasks become obvious.

To help estimating the true performance for the four class case figure 6.16 shows the classification rate versus the acceptance rate. A quality criterion parameter of up to 0.25 corresponds to the Acceptance rate of 1. Higher values move towards the upper left on the graphs. The performance starts off substantially worse than for two class problems, but surprisingly the gap is shrinking for higher quality criteria.

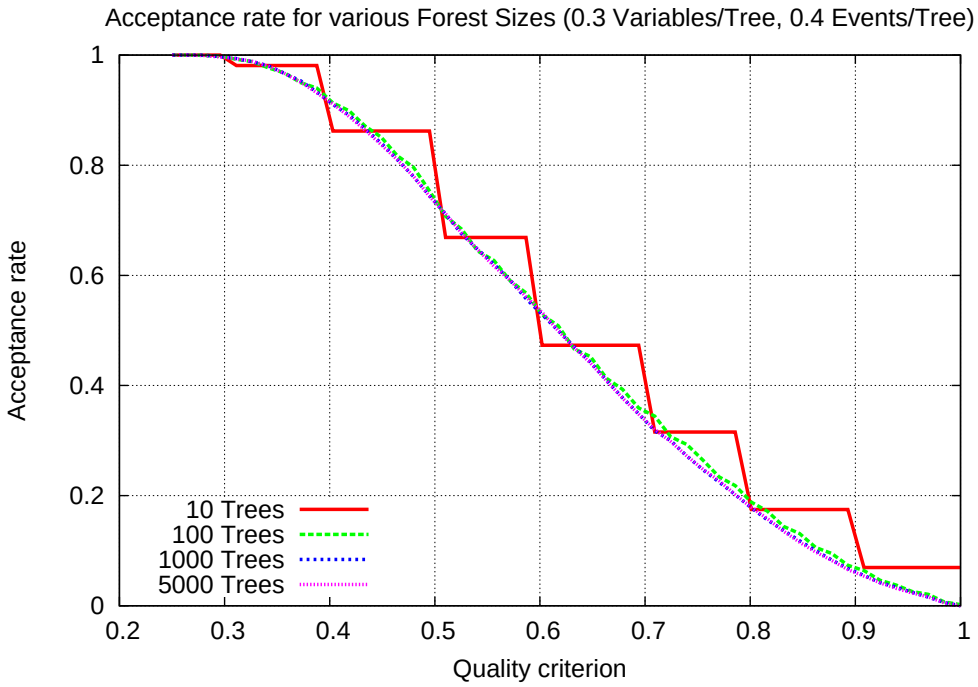

Figure 6.15: Particle identification: Acceptance rate versus Quality criterion for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

Since comparable ROC curves can only be computed for two class problems there is no ROC curve presented for this task.

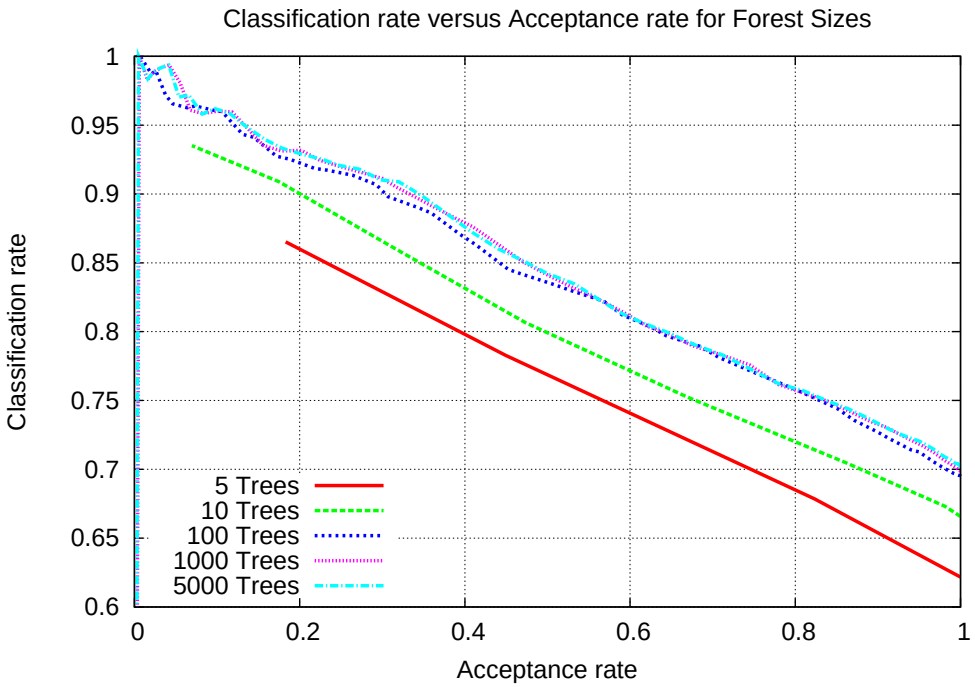

Figure 6.16: Particle identification: Classification rate versus Acceptance rate for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

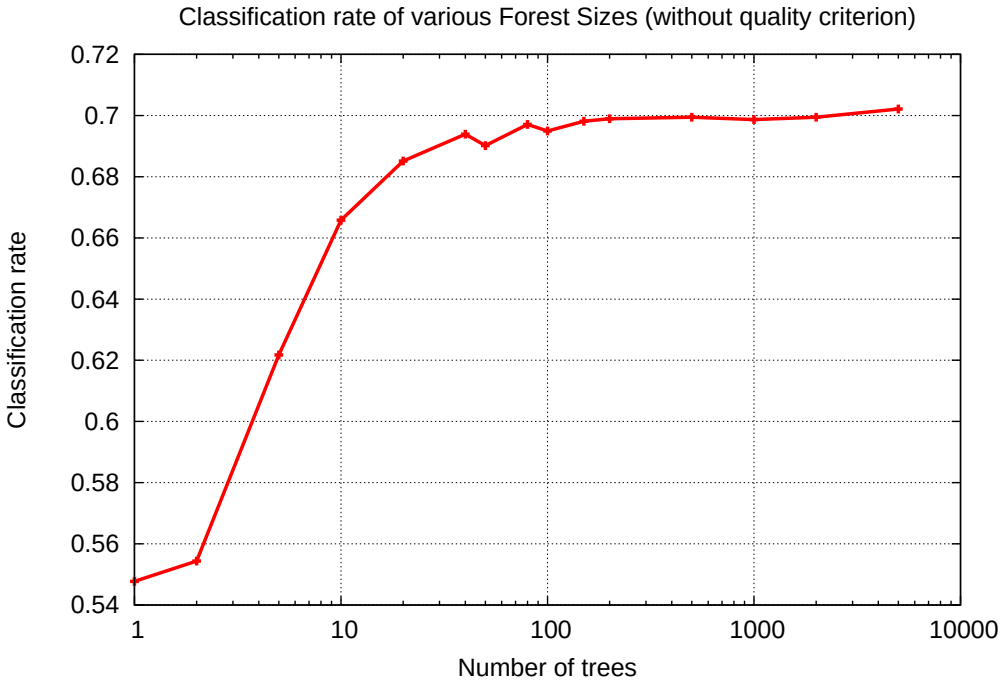

Figure 6.17: Particle identification: Comparison of different forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

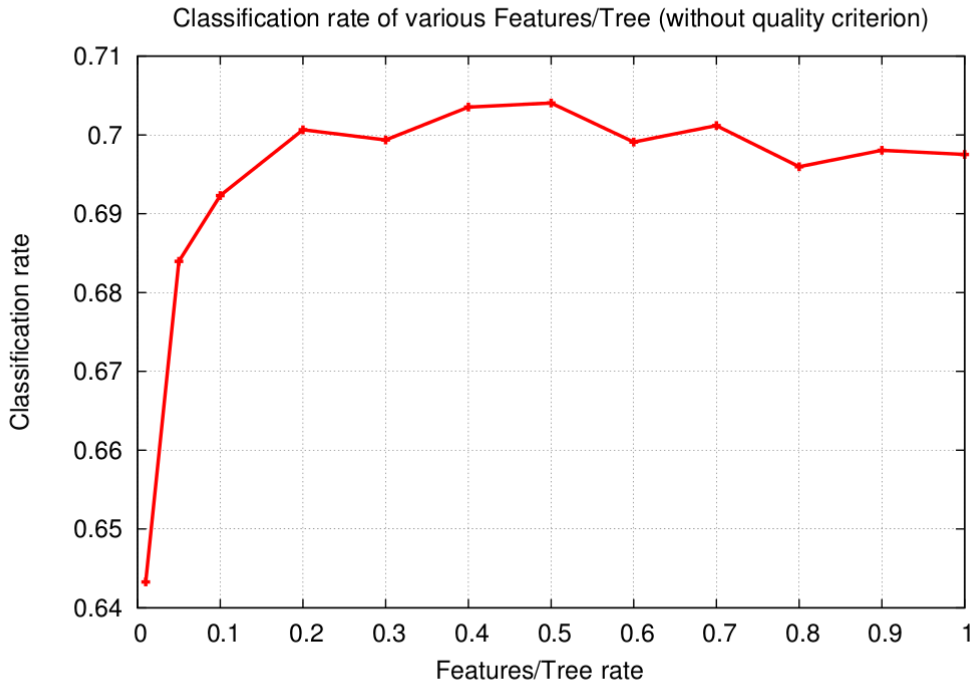

Figure 6.18: Particle identification: Comparison of different Features/Tree rates, 250 Trees, 0.4 Rate Events/Tree

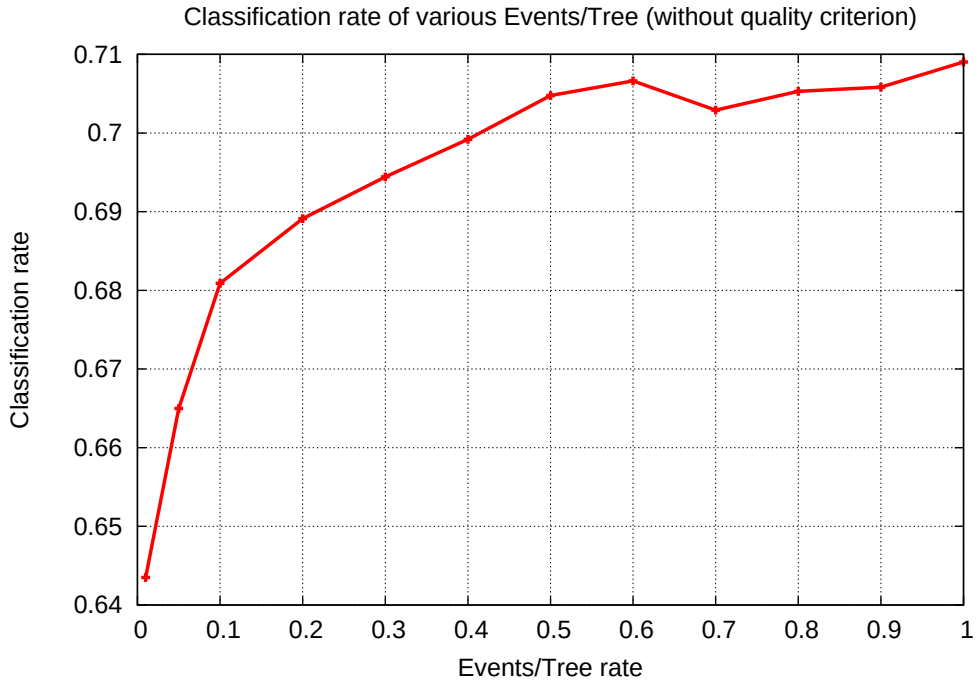

Figure 6.19: Particle identification: Comparison of different Events/Tree rates, 250 Trees, 0.3 Rate Features/Tree

Figure 6.20 shows the relative fraction of each reconstructed class for the different quality criteria. One has to keep in mind that the numbers in this figure are relative fractions of the overall number of events that could be reconstructed with the particular quality criterion. For low quality criteria, the values are close together, showing that for all classes the number of events recognized to be that class are approximately the same. This matches reality, because the test set contained almost exactly the same number of events for each class. This is relatively stable until a quality criterion of about 0.7, when neutrino events can be reconstructed with a sufficiently high precision, but muons and shower events increasingly cannot be classified certain enough. Therefore the relative composition of the remaining events shows an increasing fraction of neutrino events, especially upgoing ones.

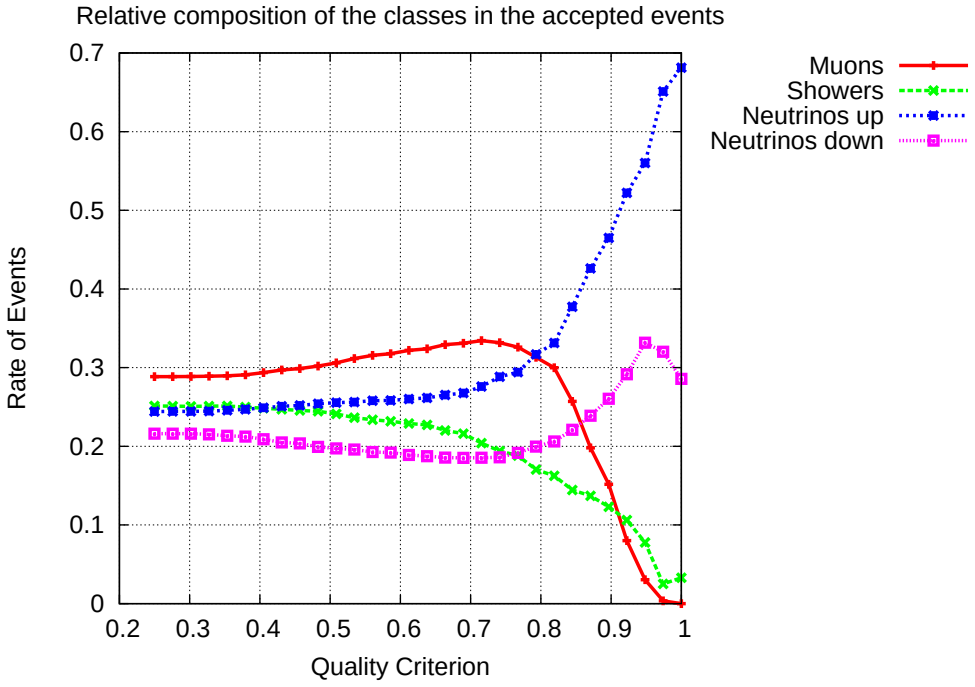

Figure 6.20: Four class particle identification: Relative distribution of the different assigned classes over the Quality Criterion

6.3 Parameter dependence

6.3.1 Angular dependence

Figure 6.21 shows the performance of a RDF (100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree) that was trained on simulation data from all angles available. The test set in this case also contained shower events for the up down classification. The events were separated into four ranges depending on the angle of the primary particle. Range 4 contains all events close to the horizon with an angle of $\pm 0.125\pi$ from the horizontal direction. Range 3 contains all events with angles between $\pm 0.125\pi$ and $\pm 0.25\pi$ from the horizon. Range 2 contains all events with angles between $\pm 0.25\pi$ and $\pm 0.375\pi$ from the horizon. Range 1 contains the almost perfectly up or downgoing events with angular differences greater than $\pm 0.375\pi$ to the horizon. These four ranges have been evaluated individually and compared to the overall performance. The results show that straight upgoing or downgoing events can be identified with a very high precision of 99%, while only 80% of the almost horizontal events can be identified correctly without a Quality Criterion and even with very high criteria only slightly above 98%.

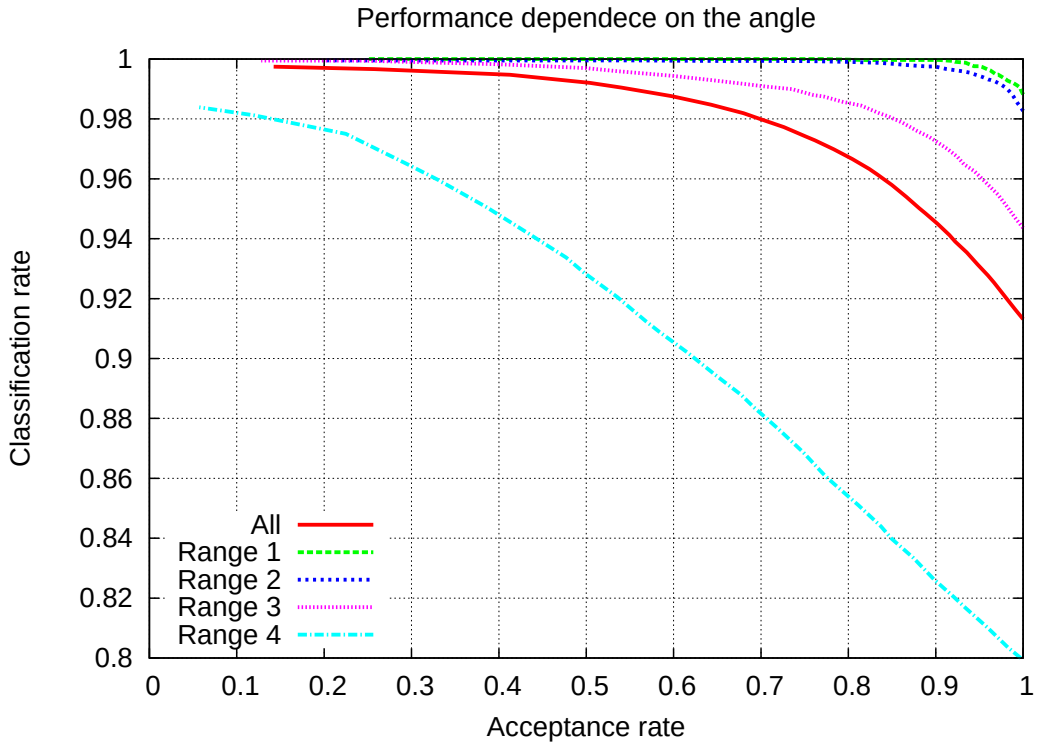

Figure 6.21: Angular dependence of the classification performance for the up/down task with 100 kHz noise

6.3.2 Energy dependence

Another important dependency is on the energy of the primary particle. Since these vary over several orders of magnitude the simulation data were separated logarithmically. The range number is computed as the rounded down result of $\log_{10}(\text{Energy of primary particle[GeV]})$. The boundaries of the ranges are listed in table 6.7. There were no events with energies below range 1 or above range 7 contained in the simulation data. The RDF with 100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree was trained on the simulation data with all energies. The results for the test sets of the different energy ranges can be seen in figure 6.22. The performance for the whole sample is labeled “All”.

One can see that the performance is better for lower energies than for higher energy ranges with one exception. The classification for the lowest energies in range 1 doesn’t perform better but worse than for energy range 2.

Range	Min. Energy	Max. Energy
1	10 GeV	100 GeV
2	100 GeV	1 TeV
3	1 TeV	10 TeV
4	10 TeV	100 TeV
5	100 TeV	1 PeV
6	1 PeV	10 PeV
7	10 PeV	100 PeV

Table 6.7: Minimal and maximal energy in the energy ranges

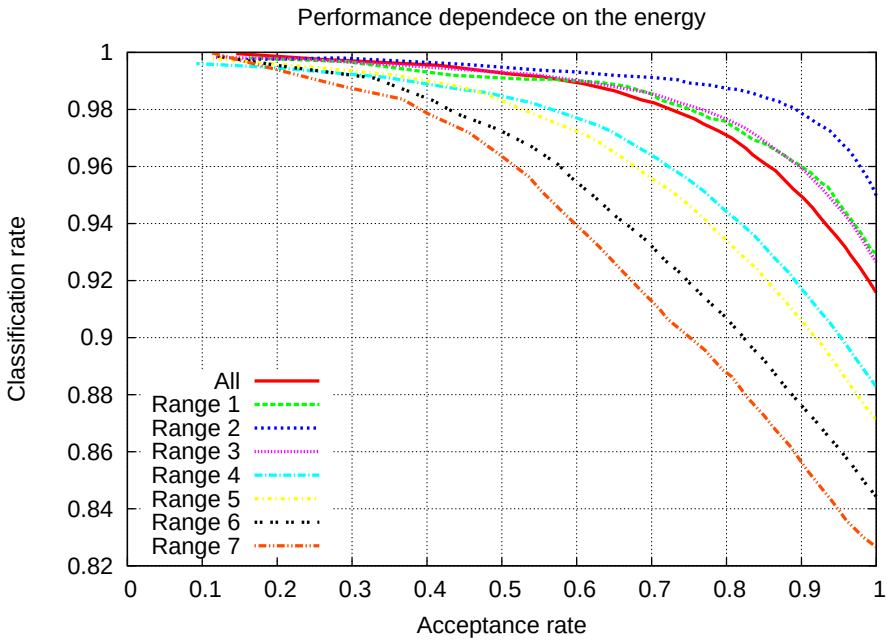

Figure 6.22: Energy dependence of the classification performance for the up/down task with 100 kHz noise

6.3.3 Trigger dependence

The trigger algorithms applied during data acquisition are a useful tool to take care of the vast masses of data produced by the ANTARES detector. All algorithms that are executed after the triggers, like in this thesis SGFeatureExtract and SGClassify, don't have to deal with many events that almost certainly don't contain any reconstructible physics. But due to the different selection mechanisms used within the different triggers the use of another trigger might influence

relevant properties of the data passed to the algorithms. All results presented until this point were computed with activated triggers “3N” and “3T”. The combination of these two triggers can be considered the current standard setting. To investigate their influence, the triggers are changed, once to only 3N and once to 3T only, and the four class experiment is repeated. The results for the different triggers can be seen in figure 6.23. The performance for only one active trigger is about 4% better than for the combined trigger of 3N and 3T. The advantage for only one trigger continues until very low acceptance rates.

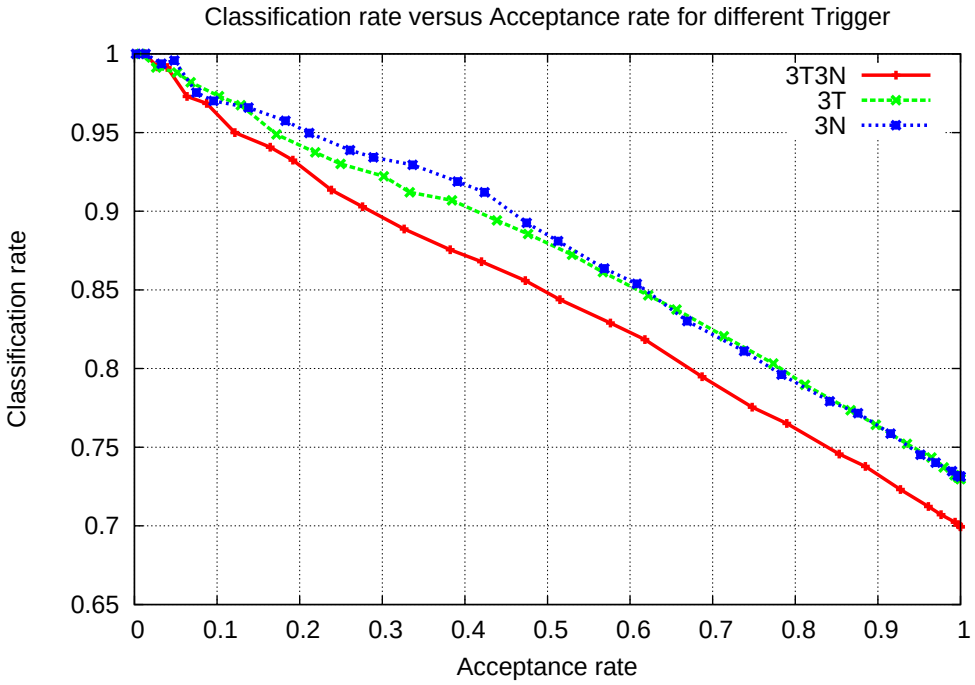

Figure 6.23: Comparison of different trigger for the four class classification, 100kHz, 100 Trees, 0.2 Features/Tree, 0.3 Events/Tree

6.4 Robustness

6.4.1 Background noise

The background noise for the detector is very variable and often exceeds the targeted operation range from 60 to 100 kHz. The algorithm must be able to deal with this. Therefore the performance for different noise levels from 0 up to 300 kHz is evaluated. To emulate the background the module "I3NoiseHitsAdder" is utilized. The RDFs were trained with the same background noise rates as they were tested later on.

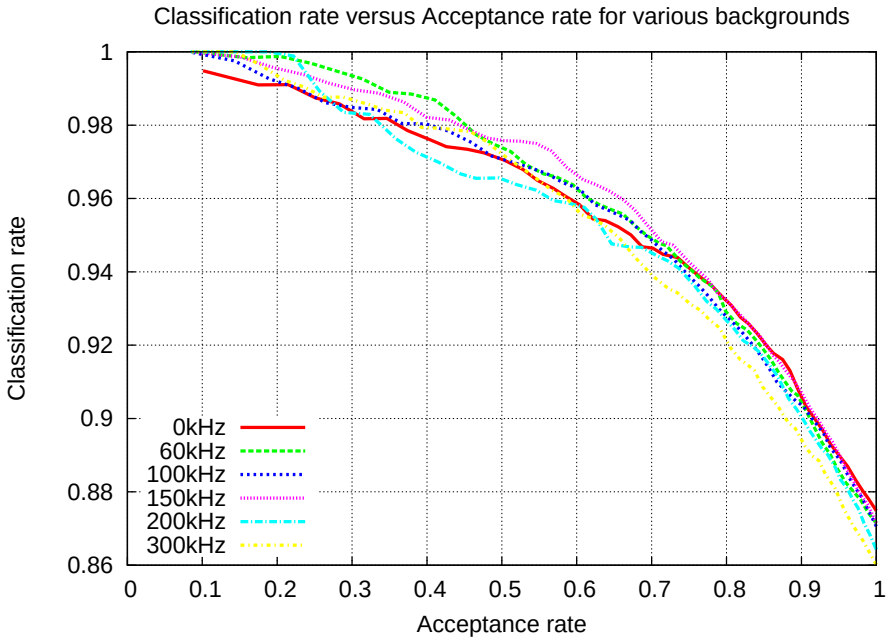

Figure 6.24: Comparison of different background rates for the muon classification task, 200 Trees, 0.3 Features/Tree Rate, 0.4 Events/Tree Rate

Figure 6.24 shows the robustness of the muon classification task against various background rates. There is no significant difference between the different background rates recognizable. Without any quality criterion the classification behaves as expected, with less background leading to a better performance (visible for an Acceptance rate of 1). But as soon as any quality criterion is demanded and therefore the acceptance rate drops, the classification rates intersect each other. The variations due to random selections are bigger than those between the background levels.

The only background rate with a noticeably poorer performance is 300 kHz, but even this effect vanishes for higher classification rates. The "ideal" case of 0 kHz does not reach a classification rate of 100% where most others do.

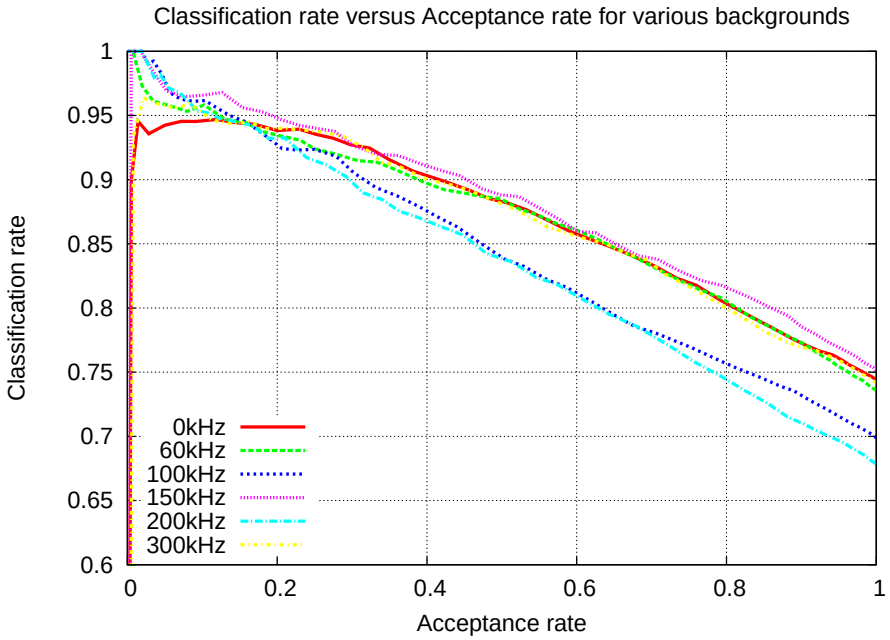

Figure 6.25: Comparison of different background rates for the four class task, 200 Trees, 0.3 Features/Tree Rate, 0.4 Events/Tree Rate

The second task evaluated for the background comparison is the four class case since it constitutes the hardest presented classification task. The performance for the various background rates is shown in figure 6.25. The drop of the classification rate on the very left for high Quality-Criterion parameter again is not just to 0.6 but to 0.0, since no event fulfills the demanded 100% tree agreement. The various background rates are easier to distinguish. RDFs trained with 100 kHz and 200 kHz perform poorer than all others. The performances for the other background rates are relatively very close together. For high quality criteria the classification rates meet again for the different background rates.

Additionally the performance of the RDFs trained for a certain background must be robust even if the background noise rate changes. The algorithm will likely be used for data with 100 to 150 kHz noise. Figure 6.26 shows the performance of a RDF trained with 100 kHz used on 100 and 200 kHz test data, as well as a RDF trained with 150 kHz evaluated on 150 and only 60 kHz

noise. The performance for the RDF trained with the lower background rate of 100 kHz used for 100 kHz is best, followed by the 150 kHz RDF used on 150 kHz. But the performance of the 150 kHz RDF applied to 60 kHz is very close to the use on 150 kHz. In contrast to that there is a considerable gap for the RDF trained for less noise than it is used for. Trained with 100 kHz and used on 200 the classification rate of the RDF drops by about 5%. But this effect disperses for lower acceptance rates.

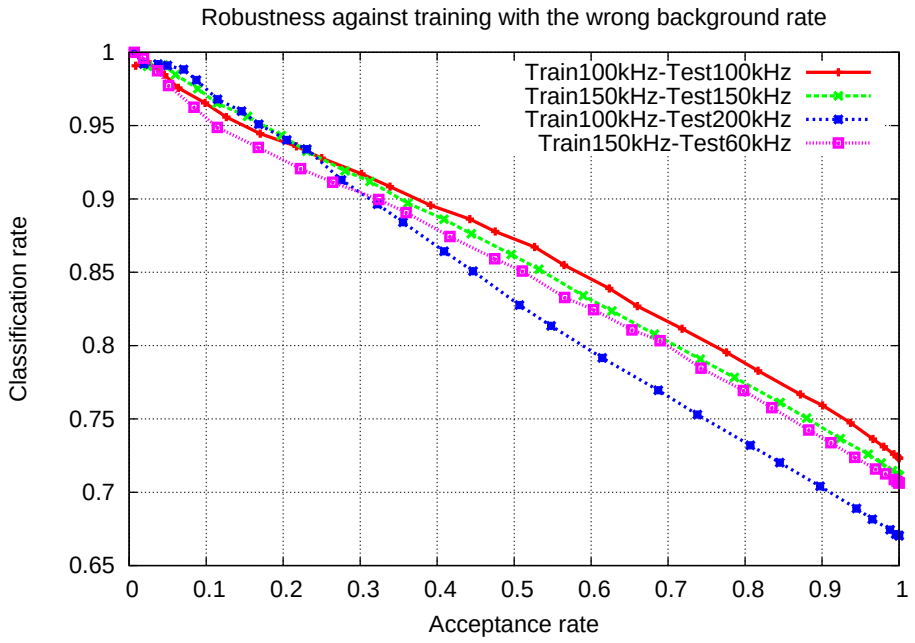

Figure 6.26: Robustness of RDFs against changing background rates for the four class classification task. One RDF trained for 100 kHz used for 100 and 200 kHz, one RDF trained for 150 kHz used for 150 and 60 kHz. All RDFs with 100 Trees, 0.3 Features/Tree Rate, 0.4 Events/Tree Rate

6.4.2 Detector failures

In reality the ANTARES detector has never been fully intact. Due to construction weaknesses, wrong handling of the equipment or other reasons, several OMs do not function properly. To produce useful results the classification algorithm must be able to handle these defective or broken OMs. The emulation of these dead OMs is done using the modules “AntEmulateOMCondition”

and “AntMaskOMCondition”. 0, 150 and 300 defect OMs were tested. The RDFs were trained for the number of dead OMs and then also evaluated for this number. The results are illustrated in figure 6.27. To put these results into the right perspective, the trigger algorithms need to be taken into account. If many OMs are defect, the trigger algorithms will find less coincidences and therefore will pass fewer events to the classifier. Table 6.8 clearly shows this effect for the file “km3v3r6.mupagev3r4.run_0214.evt.bz2“, one of the simulated muon event files.

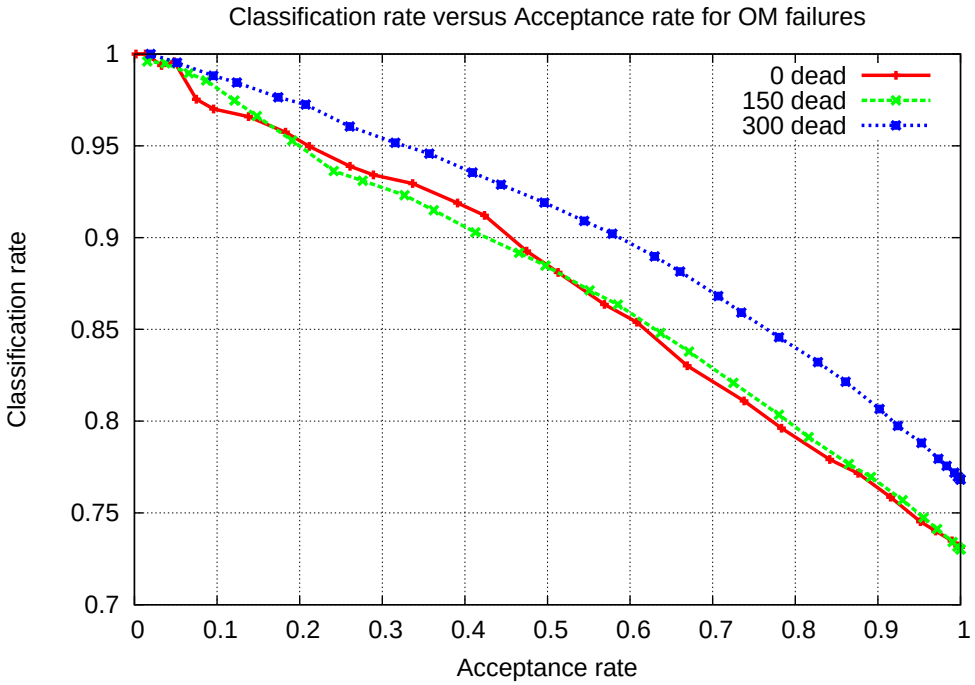

Figure 6.27: Robustness of RDFs against randomly selected defective optical modules. All RDFs with 100 Trees, 0.3 Features/Tree Rate, 0.4 Events/Tree Rate

Number of dead OMs	0	150	300	700
Number of processed events	100689	68841	42672	3522

Table 6.8: Effect of dead OMs on the trigger rate using 3N and 3T Triggers

6.5 Evaluation on real data

When it comes to real data there suddenly are more uncertainties. One major point is how well the Monte Carlo simulation describes the reality. If the always necessary idealizations contained in simulated data cause a significant difference in the feature distributions, the event classification has no chance of correcting this, since it can only be trained on labeled simulation data. This is an excellent point where one can see that sophisticated simulation is crucial for virtually everything developed later on. Presenting this in detail for all 137 is neither possible nor necessary here. Therefore three feature will be analyzed as examples for well and not so well matching features. The plots are generated by a histogram (a binning method) which evaluates the complete range for each feature for all events and splits this range into 10000 bins. For each of these bins the number of events with a value of the feature exactly within this bin is counted. In the end a normalization is performed to take care of unequal numbers of events for the different classes. All empty bins (where no events had such a value for this feature) are not displayed.

The real data sample which is used is certain to consist of virtually only atmospheric muons, therefore these two classes should match. The first feature is feature F_{50} (see Appendix A). It is one of the significant features for up/down classification. Figure 6.28 shows a zoom into the relevant parts of the distribution. One can clearly see that upgoing neutrino events mostly have positive values for F_{50}, while downgoing muons tend to have negative values. The relevant comparison here is that the distribution of th real data events (in red) looks very similar to the muon distribution (cyan). They rise at the same values, but the peak is higher for muons than for real data. The events not contained in the peak for the real data must be hidden in the long tails (which would reach out to $\pm$ several thousand), since the values are normalized to the number of events.

Figure 6.29 shows the relevant part of the distributions of F_{11}. The classwise distributions differ in the length and the thickness of the tails and the shape and exact location of the main bulks. The only significant difference is at the right where the real data have far more events at the maximal value of 40. The limit of 40 is due to the computation of F_{11} using a search window.

A zoom into the bulks of this distribution is shown in figure 6.30. It reveals a nice matching of the real data events and the muon events at this point.

Unfortunately there also are some features where the distributions do not agree. Feature F_{38} is such a case, where the simulated classes have considerably different properties than the real events. The classwise distributions are shown in figure 6.31. The size and location of the real data distribution does not agree with the muon distribution.

To state this clearly, for most features the real data agree very well with their expectation.

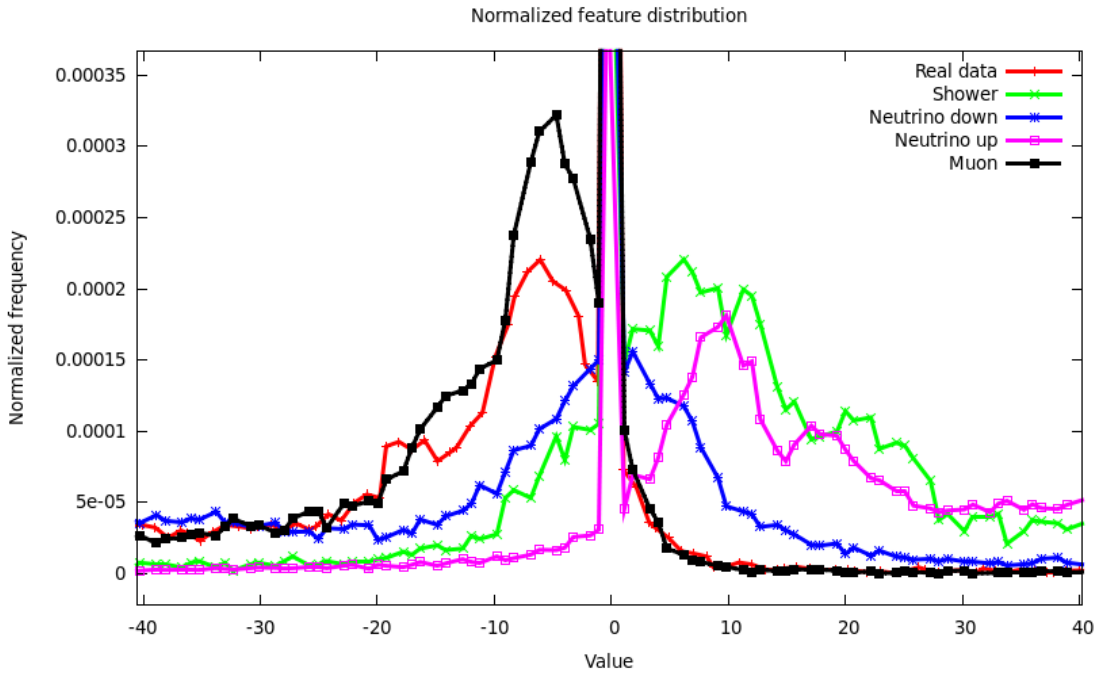

Figure 6.28: A zoom into the classwise distribution of F_{50}

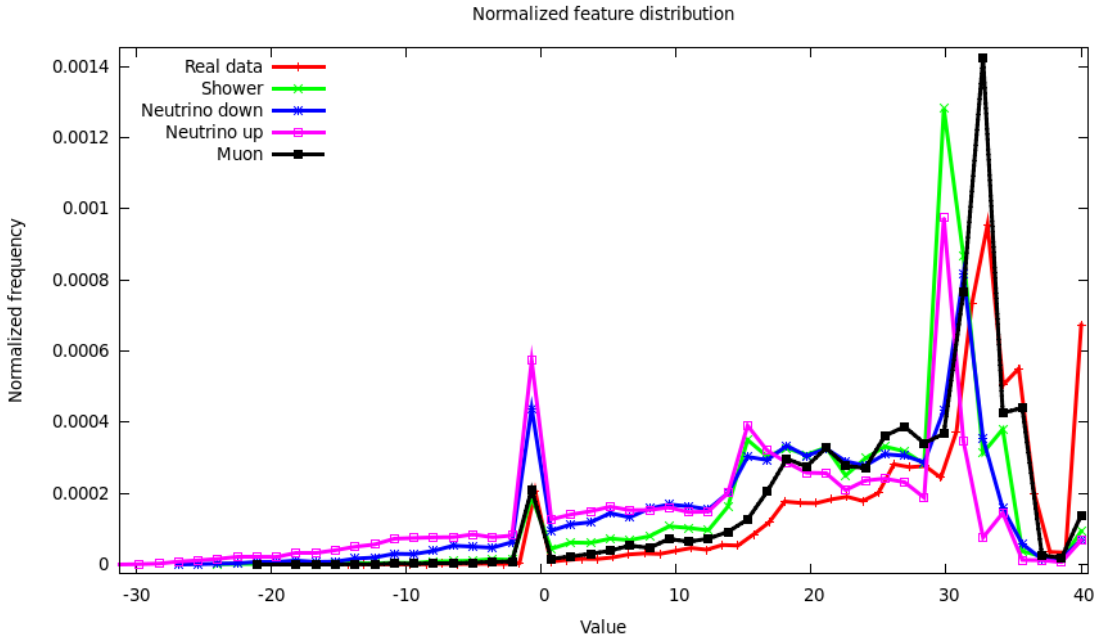

Figure 6.29: Classwise distribution of F_{11}

F_{38} has been selected to show that discrepancies are possible.

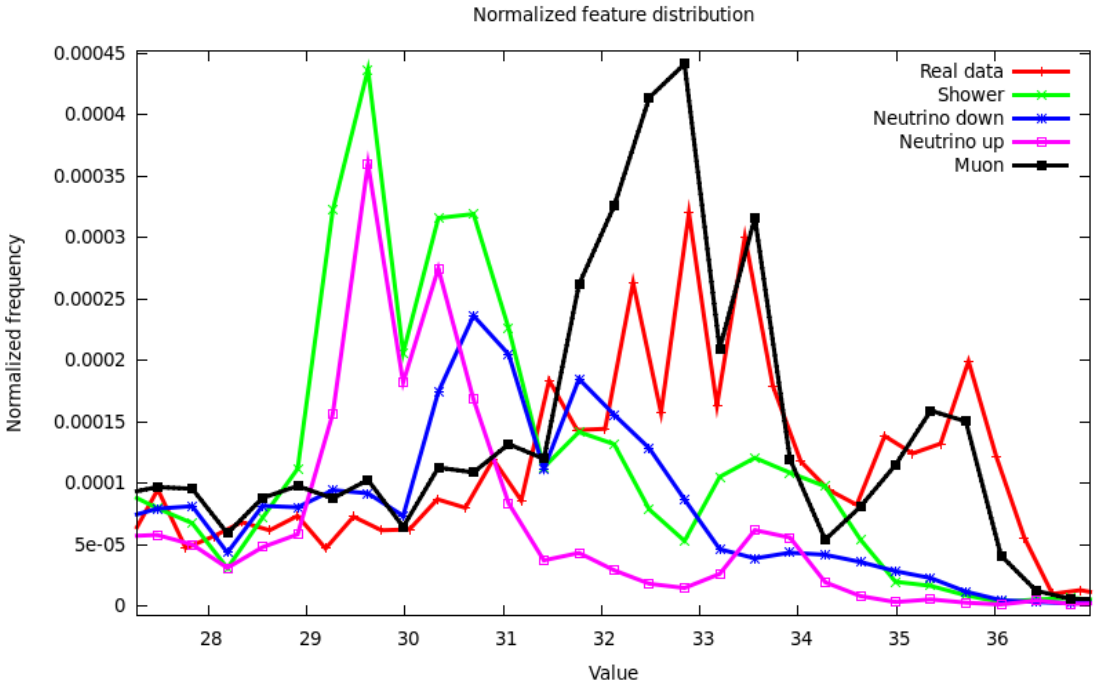

Figure 6.30: Zoom into the classwise distribution of F_{11}

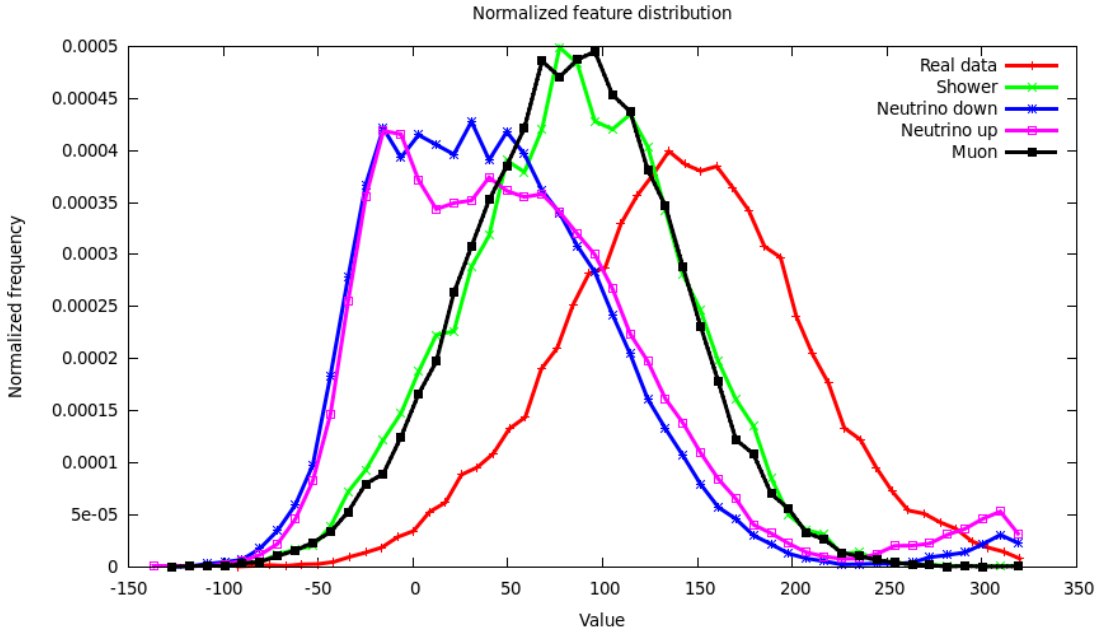

Figure 6.31: Classwise distribution of F_{38}

The second major issue is the lack of meta information for real data. For a recorded event there is no label which kind of event has been detected. But without this information also no per-

formance evaluation can be done. Luckily, for this particular problem the vast majority of signals are known to be background events caused by atmospheric muons. This justifies the assumption that triggered real data consist almost exclusively of downgoing muons. The expectation for the classification therefore must be to find mostly muons possibly with small fractions of other candidates.
In principle all of the already mentioned tasks lead to a RDF usable for this task. Since the expectation are almost only muons, the obvious approach is to evaluate with the RDF trained in chapter 6.2.1 for the atmospheric muon detection. The optimal case would be a working four class particle identification from chapter 6.2.4. But since the main goal of the ANTARES experiment is to find upgoing cosmic neutrinos, a setup specifically detecting these is already very useful.

The data files used for this analysis are

- Antares_035997.11-02-01.CAL.i3.gz
- Antares_035999.11-02-01.CAL.i3.gz
- Antares_036000.11-02-01.CAL.i3.gz
- Antares_036002.11-02-01.CAL.i3.gz
- Antares_036029.11-02-01.CAL.i3.gz

from October 2008, when the full twelve line configuration had been installed. For the presented results these five files are mixed and the resulting test sample is evaluated. The RDFs were trained using the simulated data with 150kHz background noise, each with 100 trees, 0.2 rate of Variables per tree and 0.3 rate of events per tree. Although the data files from 2008 have been recorded with approximately 60kHz, 150kHz was chosen for the training because the true rates, especially for later presented point-source analysis data varies and furthermore the comparison of wrong background rates in chapter 6.4.1 has revealed, that a somewhat higher background during training does not drastically affect the performance.

6.5.1 Real data sample - Four class identification

The first evaluation is done with the RDF trained for four classes. It is supposed to give an idea about the overall structure instead of exact numbers, since it likely is the most inaccurate of the trained RDFs. Its classification rate without quality criterion on the Monte Carlo data was at 70%, which is significantly less than for the two class cases. The results can be seen in figure

6.32 and the corresponding acceptance rate is contained in figure 6.34. Table 6.9 contains the exact numbers for the case without a quality criterion and a demanded forest agreement of 70% of all trees. The classification expectedly identifies predominantly muons, but with a rate of only 86% instead of the predicted 99%. This can be compared with the results of a four class classification on 100% muons from simulation in figure 6.33. The two distributions look very much alike.

QC	0	1	2	3	R
0.0	17865	643	401	2054	0
0.7	8263	19	11	161	12509

Table 6.9: Class distribution for the real data sample for the four class classification at Quality Criterion = 0.0 and Quality Criterion = 0.7

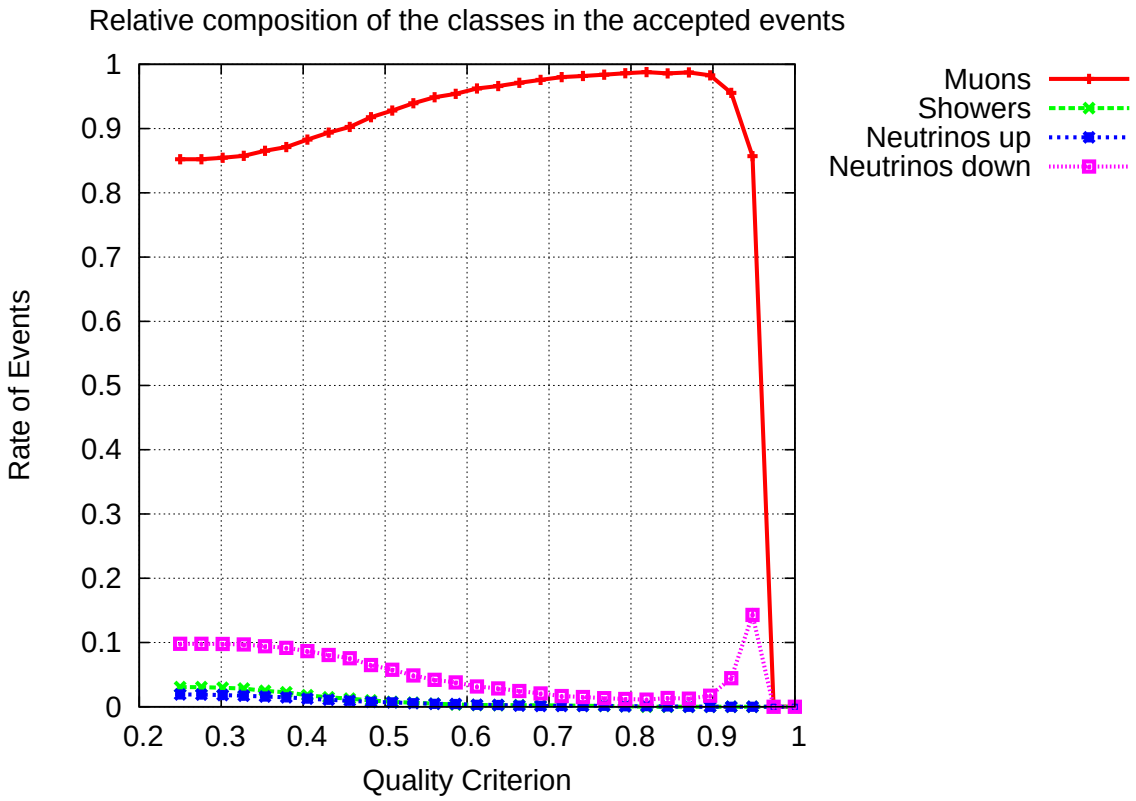

Figure 6.32: Relative event distributions in fractions of all accepted events at a certain quality criterion for real data, 100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree rate, Four class identification

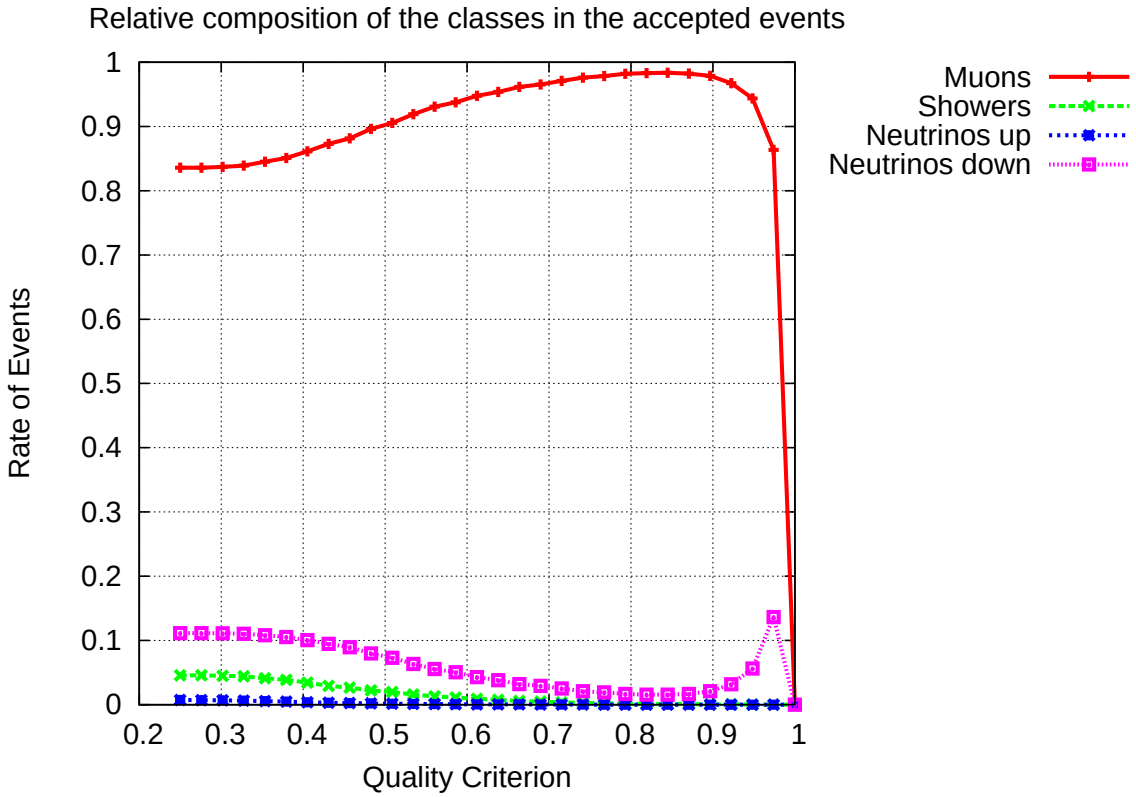

Figure 6.33: Relative event distributions in fractions of all accepted events at a certain quality criterion for simulation data of 100% muons, 100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree rate, Four class identification

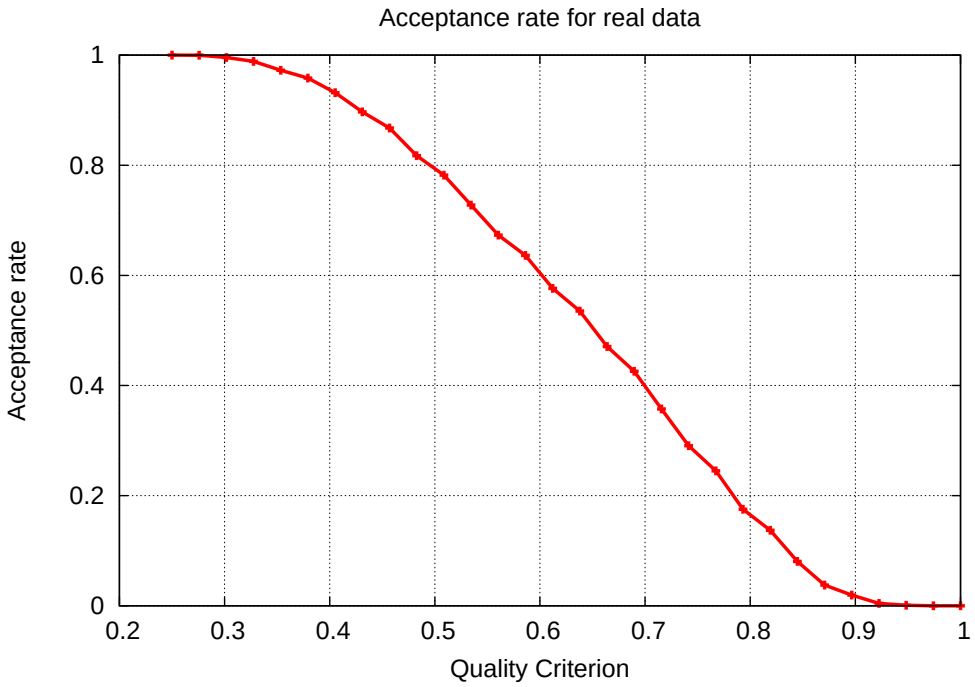

Figure 6.34: The Acceptance rate versus the Quality Criterion for the four class identification, corresponding to 6.32

6.5.2 Real data sample - Muon identification

To obtain a higher precision than with the four classes case, a RDF specifically trained to identify one particular event type was used for the real data sample. The RDF used here showed 88% classification rate without a quality criterion for the Monte Carlo data. The results in plain numbers for QualityCriterion values of 0.0 and 0.7 are contained in table 6.10. The whole performance, again in relative numbers is shown in figure 6.35 and the corresponding acceptance rate in figure 6.36. These results match the predictions considerably better than for the four class classification, but again the behavior becomes unstable for very high quality criteria, since only very few events are able to fulfill them (in this case only 20 events for a quality criterion of 1.0). It is very interesting to note that from these 20 events eight are classified as clearly not muons.

QC	0	1	R
0.0	19398	1565	0
0.7	16270	578	4115

Table 6.10: Class distribution for the real data sample for the two class muon classification at Quality Criterion = 0.0 and Quality Criterion = 0.7

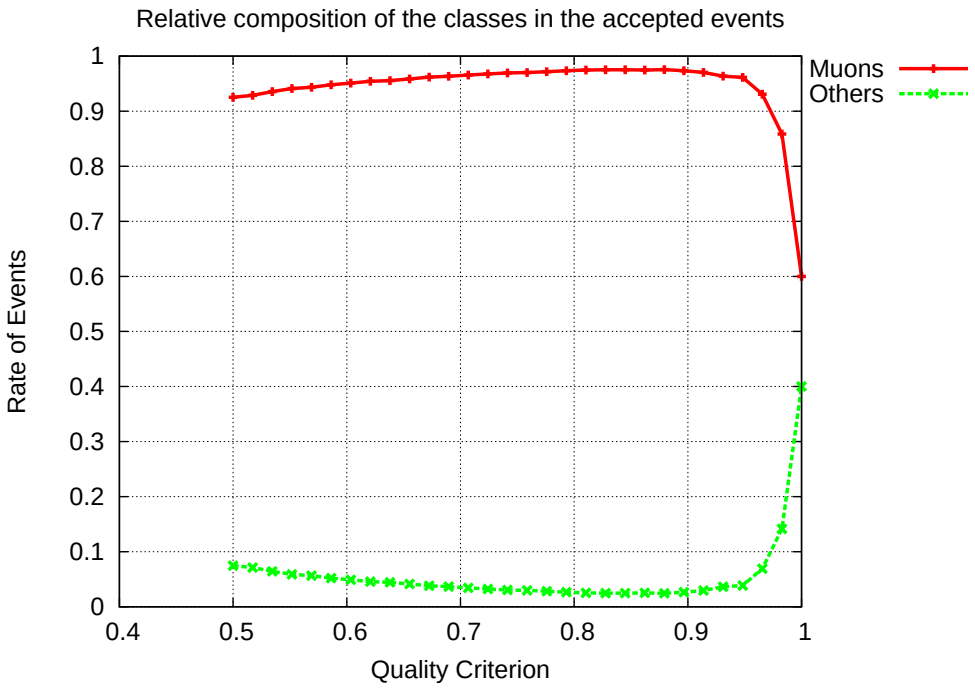

Figure 6.35: Relative event distributions in fractions of all accepted events at a certain quality criterion for real data, 100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree rate, Two class muon identification

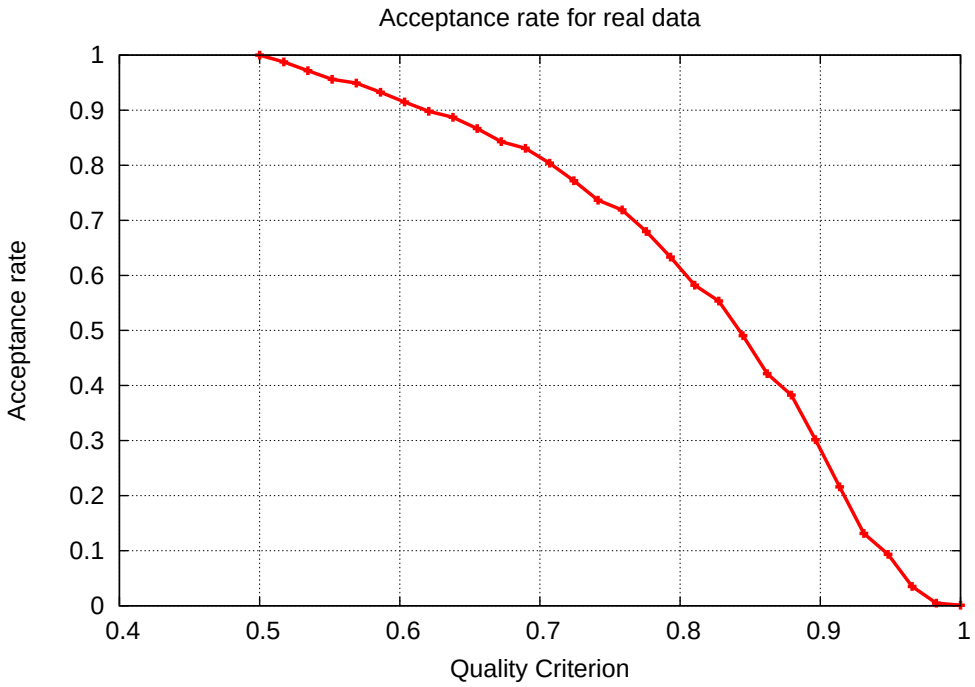

Figure 6.36: The Acceptance rate versus the Quality Criterion for the two class muon identification in real data, corresponding to 6.35

6.5.3 Real data sample - Neutrino identification

The classification identifying specifically upgoing neutrinos shows an almost perfect match with the composition predicted by theory, see figure 6.37. Primarily events not matching the upgoing neutrino pattern are found but no statement about the composition of ”Others“ can be made here. With increasing quality criterion the distribution more and more reaches the prediction of virtually only background events being contained in the data file. Unfortunately there is no prediction possible when true upgoing neutrino signals would be rejected by this RDF.

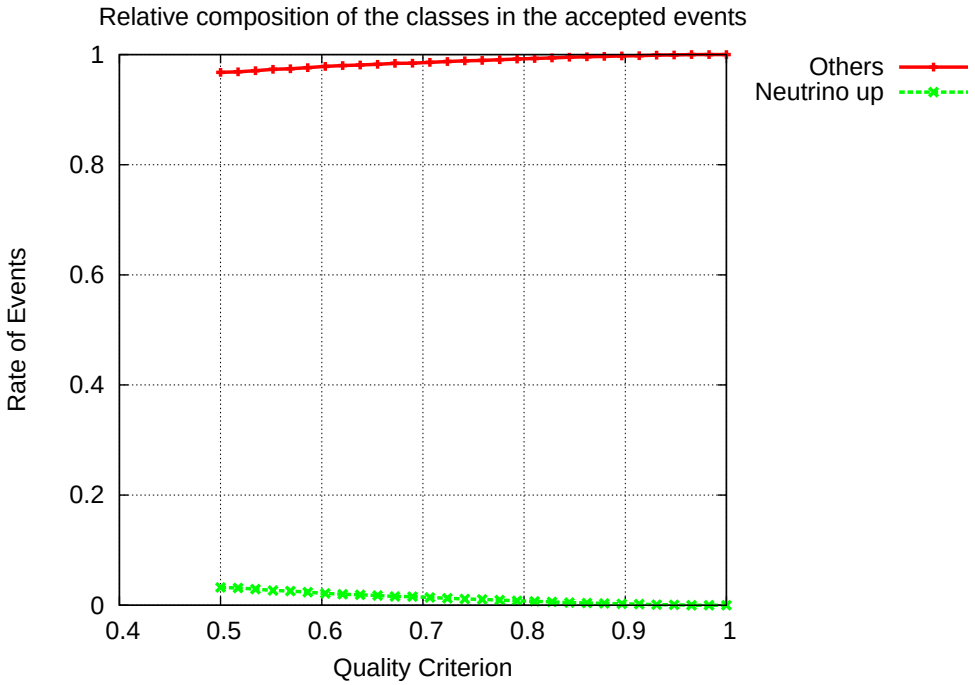

Figure 6.37: Relative class distributions for real data, 100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree rate

The figure illustrating the Acceptance rate versus Quality Criterion corresponding to 6.37 can be found in appendix C.

6.5.4 Point source events - Four class identification

In the following a selection of 2087 upgoing cosmic muon-neutrino candidates is analyzed. These events have been selected for a point source analysis and are estimated to contain about 60% upgoing neutrinos and 40% downgoing muons. The first experiment on these neutrino candidates was to classify using the same four class RDF as in section 6.5.1. The results in table 6.11 show that the results deviate considerably from the expected 60% upgoing neutrinos and 40% downgoing muons. Especially the high number of shower events is not at all included in the predictions for this sample. But the number of atmospheric muons is relatively low. The overall performance and the corresponding acceptance rate can be seen in figures 6.38 and 6.40. As the quality parameter rises, the relative number of neutrinos found in the sample increases comparatively stable up to 70% at a quality criterion of 0.9. After that there are only neutrino events left to fulfill the desired forest agreement, which explains the jump to 1 in the classification rate and the drop to zero once no events are accepted any more. As a comparison to this behavior the class distribution for a simulated dataset of 50% athmospheric muons and 50% upgoing neutrinos has been evaluated. The results shown in figure 6.39 show a similar trend for the upgoing neutrinos and about the same low number of misclassified downgoing neutrinos. But on the one hand, there are considerably less shower events found in the Monte Carlo sample, on the other hand the roughly correct, high fraction of muons is identified.

QC	0	1	2	3	R
0.0	266	681	861	279	0
0.69	19	99	183	14	1772

Table 6.11: Class distribution for the point source sample for the four class classification at Quality Criterion = 0.0 and Quality Criterion = 0.69

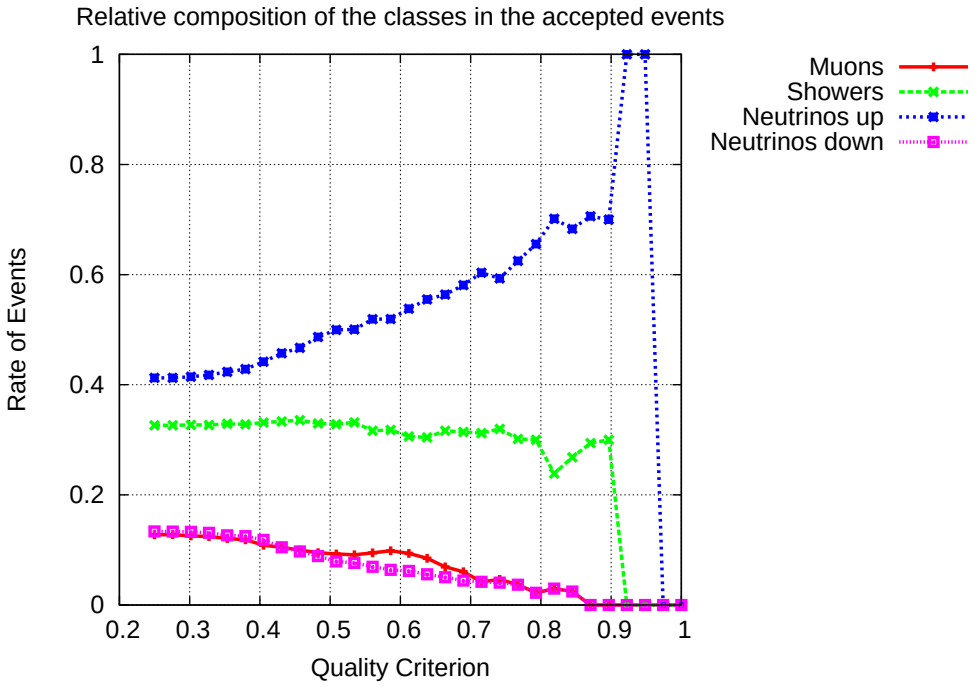

Figure 6.38: Relative class distributions for point source data, 100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree rate

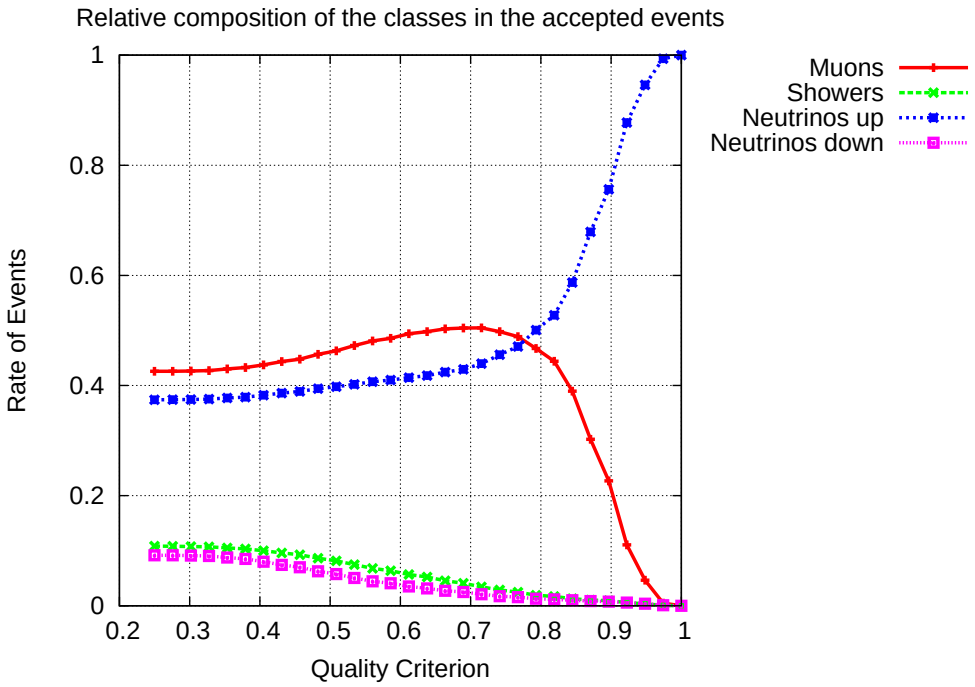

Figure 6.39: Relative class distributions for simulated data of 50% downgoing muons and 50% upgoing neutrinos, 100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree rate

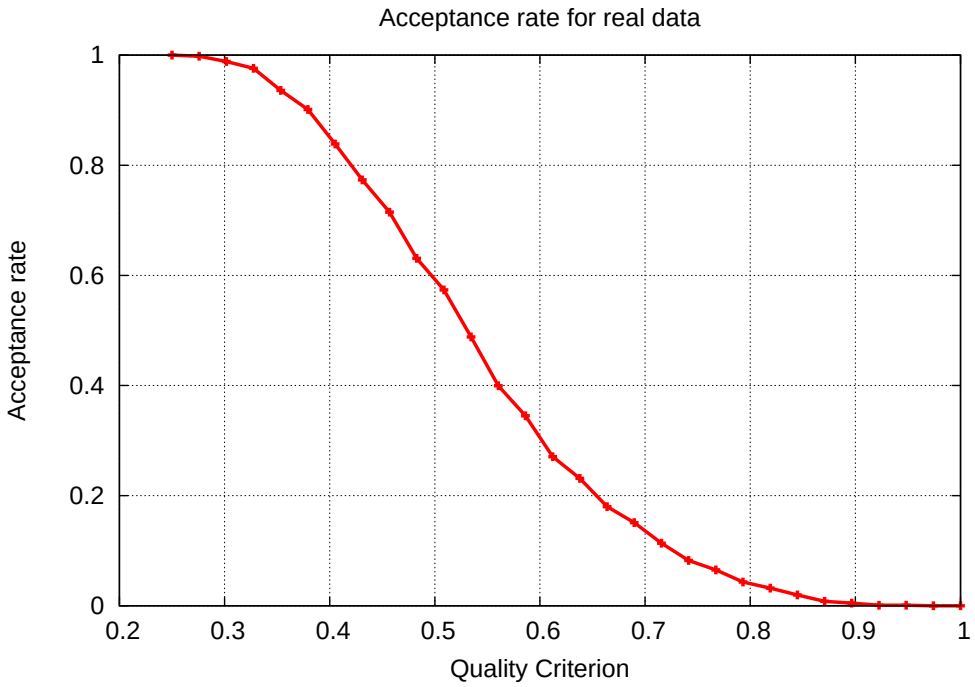

Figure 6.40: The Acceptance rate versus the Quality Criterion for the four class identification, corresponding to 6.38

6.5.5 Point source events - Muon identification

A dedicated search for patterns induced by atmospheric muons is supposed to deliver more reliable numbers. It reveals a fraction of twenty percent identified as muons. But the pattern of high energetic neutrino candidates seems to produce clearer results, since this fraction increases with the quality criterion. The results are contained in table 6.12 and figure 6.41.

QC	0	1	R
0.0	426	1661	0

Table 6.12: Class distribution for the point source events for muon classification at Quality Criterion = 0.0

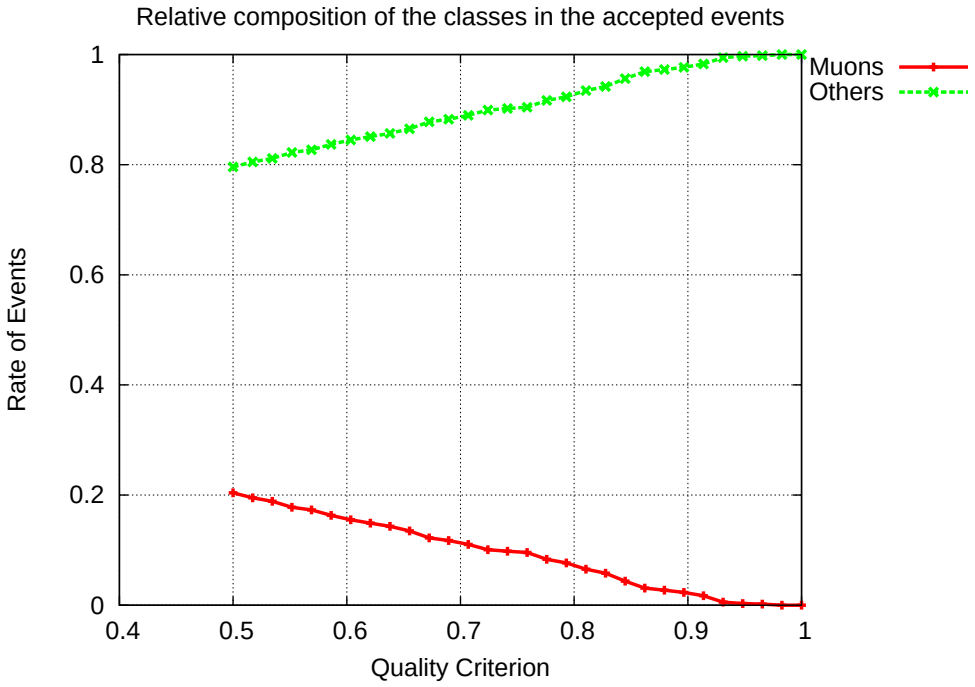

Figure 6.41: Relative class distributions for point source data, 100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree rate

The corresponding Acceptance rate versus Quality criterion plot can be found in appendix C.

6.5.6 Point source events - Neutrino identification

In theory a search for upgoing neutrino events should result in the inverse numbers as in chapter 6.5.5. But since the previous experiments have already shown that not all signals do belong clearly to either upgoing neutrinos or downgoing muons, this should not exactly be the case. The results seen in figure 6.42 for low quality criterion values show the expected trend. This might even be the best classification result for the point source sample. The exact numbers are shown in table 6.13. But for higher rejection rates, the relative amount of neutrinos drops, whereas the fraction of others, meaning downgoing muons, downgoing neutrinos and shower events, rises. But since the results of chapter 6.5.5 already showed that these events likely aren't muons, the absolutely certain events might be shower events or, less likely, downgoing neutrinos. Figure 6.43 shows the Acceptance rate corresponding to figure 6.42. As always the acceptance rates for two class problems are higher than for the four class classifications.

QC	0	1	R
0.0	714	1373	0

Table 6.13: Class distribution for the point source events for neutrino classification at Quality Criterion = 0.0

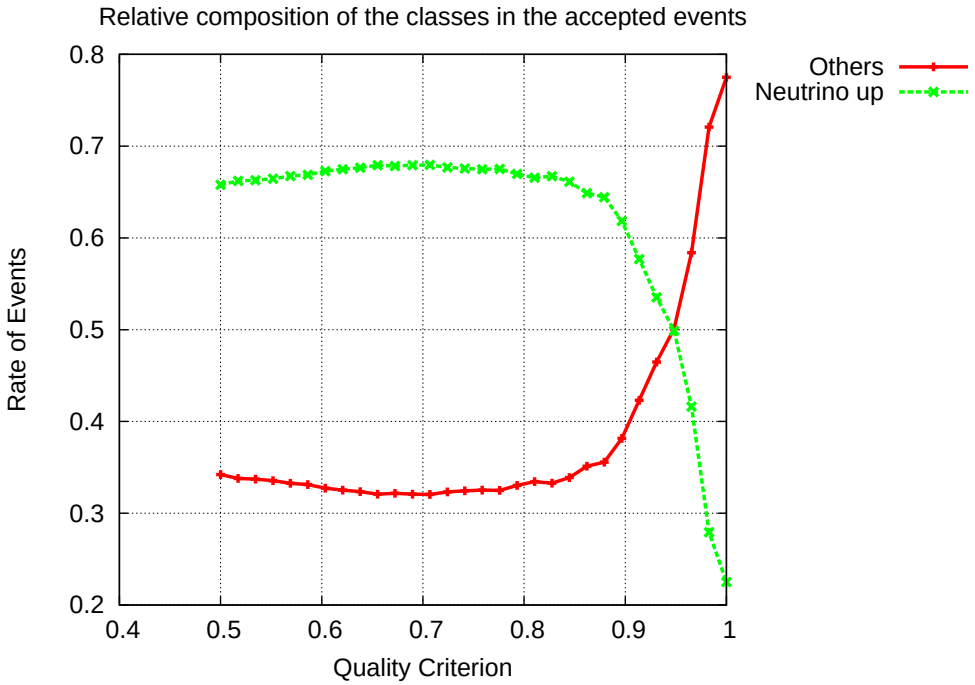

Figure 6.42: Relative class distributions for point source data, 100 Trees, 0.2 Features/Tree rate, 0.3 Events/Tree rate

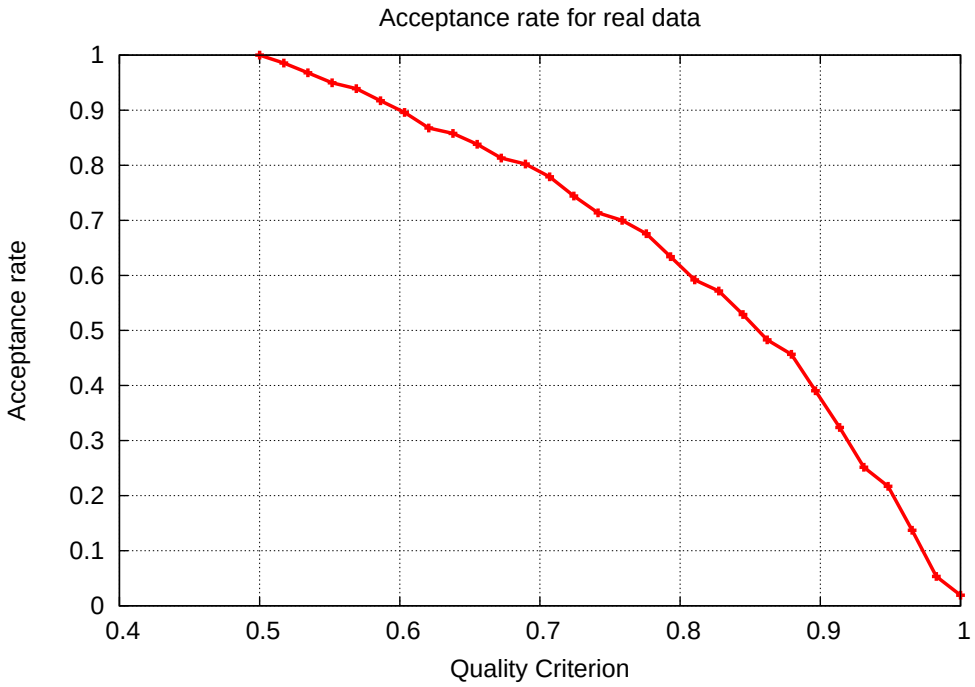

Figure 6.43: The Acceptance rate versus the Quality Criterion for the two class Neutrino identification, corresponding to 6.42

6.5.7 Point source events - Up/down classification

Since the for point source data the other classifier did not all agree completely, the reliable up/down classification is applied to these data, too. The results in figure 6.44 show a good agreement with the previous two class classifications of a little less than 30% downgoing muons within the sample.

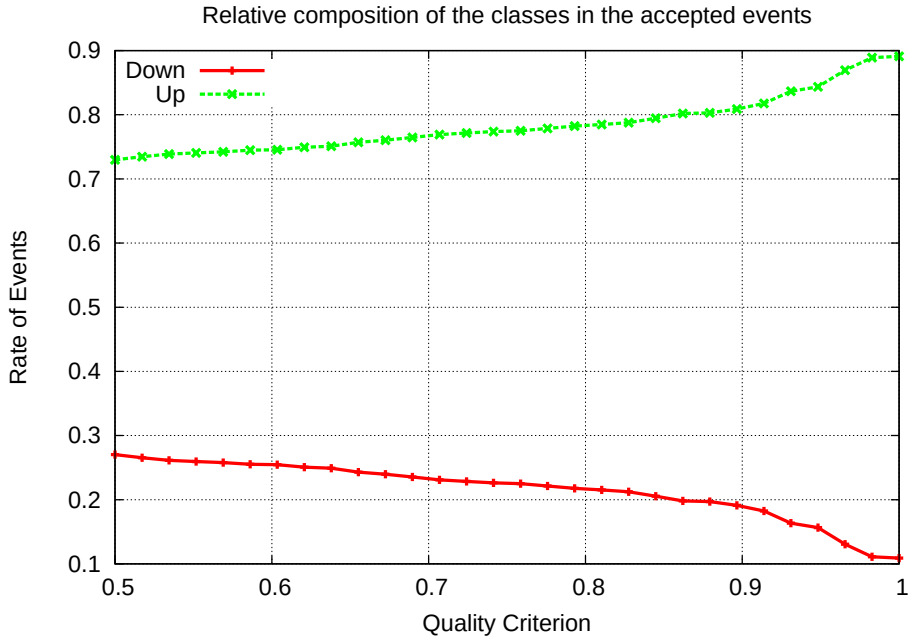

Figure 6.44: Relative class distributions for the point source data

6.6 Results for comparison to previous work

The final classification performance from [Gey10] is shown in figure 6.45. The results are achieved using artificial neural networks on a simulation data set containing downgoing muons and upgoing neutrinos with 60 kHz background noise. Green results (E XVI) are from a network trained for 60 kHz background, red results (E XV) from a net trained without background noise.

Figure 6.46 shows a setup for a comparison. The two class problem computed on simulation data with a background of 60kHz. The classes are:

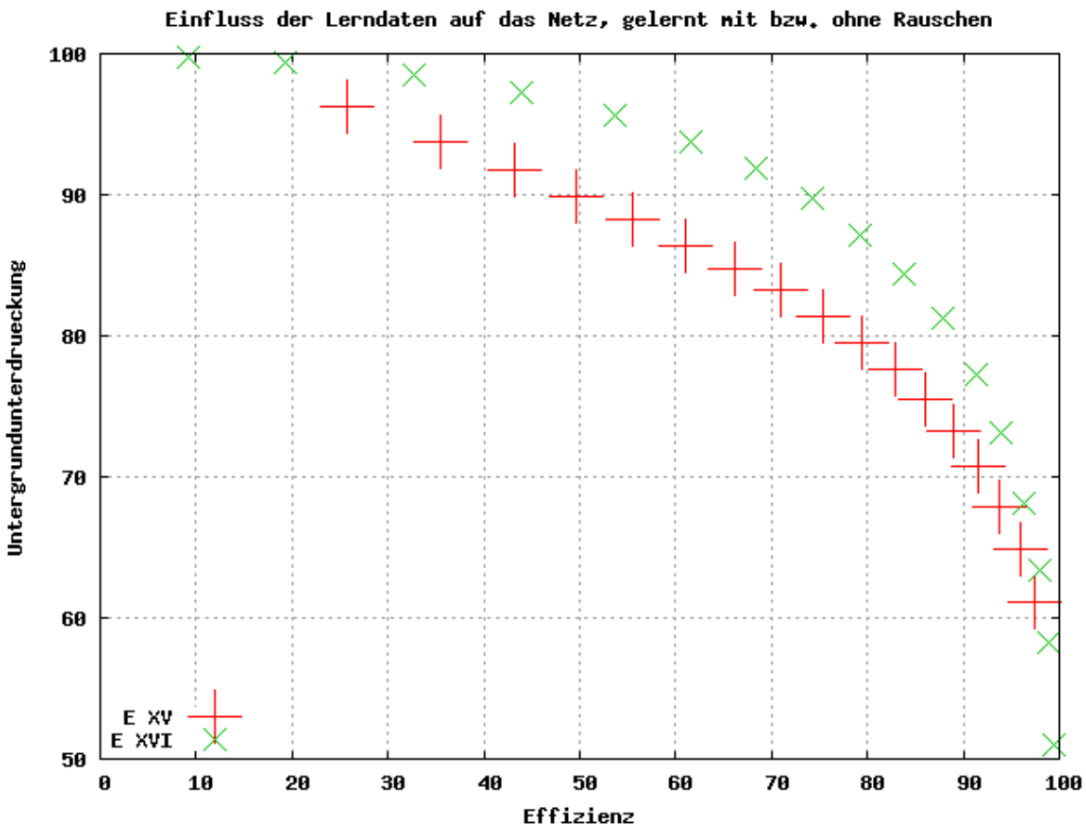

Figure 6.45: The final results using artificial neural networks for a two class downgoing muon and upgoing neutrino separation task with 60kHz noise; from [Gey10]

- Class number 0: Atmospheric muons
- Class number 1: Upgoing neutrinos

Data for other events like showers are not contained in the training or test set. This corresponds to a up/down classification as shown in chapter 6.2.3, but without downgoing neutrinos and less noise, the performance is further improved. The RDF was trained with 100 trees, 0.2 Feature/Tree rate and 0.3 Events/Tree rate.

Since the up/down classification will likely be the most useful tool, the signal to noise ratio is evaluated for this task. Signal to noise factor SNF here is computed as the number of upgoing neutrinos n_ν divided by the number of downgoing muons n_μ.

$$SNF = \frac{n_\nu}{n_\mu} \tag{6.1}$$

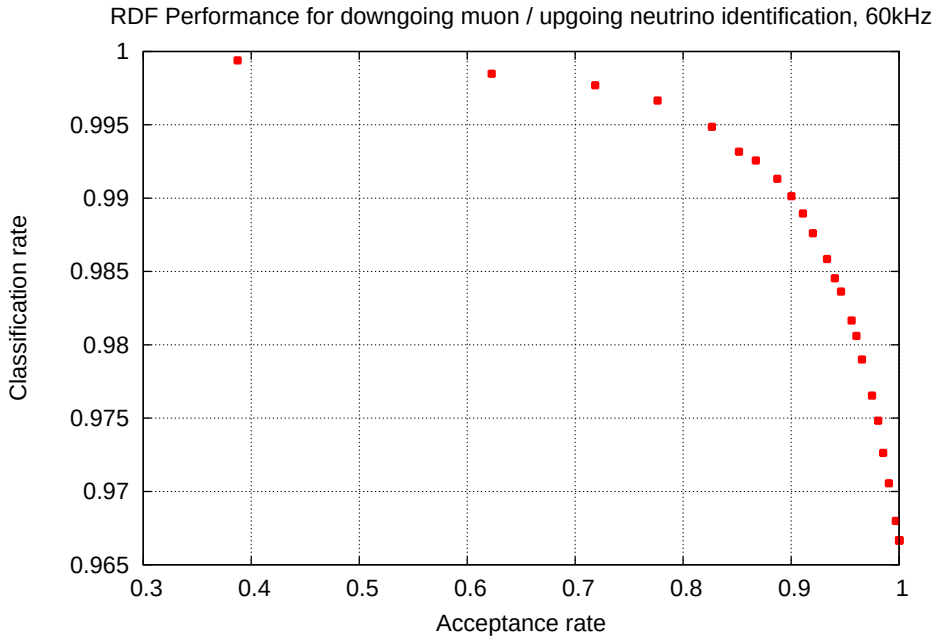

Figure 6.46: Two class downgoing muon and upgoing neutrino separation task with 60kHz

The signal to noise ratio is just the $\log_{10}$ of the SNF.

$$SNR = \log_{10}(\frac{n_\nu}{n_\mu}) \tag{6.2}$$

The setup again contains only downgoing muons and upgoing muon-neutrinos, but with 100 kHz background. Table 6.14 shows the starting conditions before any classification and the results after classifications with different quality criteria.

Qc	Before classification	0.0	0.8	0.95	0.99
n_μ	729086	16357	4300	1092	290
n_ν	2843	2741	2578	2199	1720
Efficiency ν	1.0	0.9641	0.9068	0.7735	0.605
SNF	0.0039	0.1676	0.5995	2.0137	5.931
SNR	-2.4089	-0.7757	-0.2222	0.304	0.7731

Table 6.14: Signal to noise ratios for up/down classification with different quality criteria

6.7 Runtime

Concerning the runtime of the algorithm, one must take into account that some part is already caused by the initialization of seatray and various algorithms like triggers. Therefore the runtimes were measure with one event only to investigate the initialization time, and once for the whole file "km3v3r6.mupagev3r4.run_0214.evt.bz2" with 100689 events. The computed events per second rates are based on the runtimes for the full, 100689 event data file. The names in table 6.15 are shortened: SGFeatureExtract is SGFE and SGClassify is SGC. The presented times are the average of three runs. Until now the implementation has not been optimized with respect to runtime.

	Without	SGFE	SGFE & SGC
1 event	$4.47\ s$	$4.45\ s$	$15.18\ s$
100689 events	$91\ m$	$340\ m$	$398\ m$
Events / s	18.4	4.94	4.22

Table 6.15: Runtime comparison for the algorithms

Chapter 7

Discussion

Chapter 6 has presented many experiments and their results as well as tests to find the limitations of the algorithm. A detailed analysis of the results offers deeper insights. The first results in table 6.1 show the motivation for using the RDF as classifier. The neural networks do not yield better results than the tree based algorithms, but need considerably longer for training and classification. All other algorithms above 80% classification rate are tree based and therefore should share many advantages and drawbacks. Therefore the RDF as best performing tree based algorithm with 88% classification rate has been selected. Techniques to enhance the feature space like PCA or the extended feature set described in chapter 5.2.7 showed absolutely no improvement. Also the results of the feature selection are not satisfying. The genetic algorithm takes a long time to converge to one solution and due to the randomness contained in the RDF classifier this effect is even more severe. As the numbers show, the performance loss with one optimal subset for different tasks is greater than the gain for the task it was optimized. For the two cases presented, there even isn't a significant gain detectable. The fact that the RDF performance doesn't increase very much with an optimal subset is plausible, since during the training for every tree, each feature subset available for one tree is already evaluated according to the relevance of the features. Weak features therefore can only influence each tree and by that the overall outcome in a minor fashion. This is not only a plausible explanation why the attempted techniques didn't result in significant enhancement, but it even implies that some feature selection is already granted as an intrinsic property of the RDFs. Nevertheless some optimization would still be possible at this stage.

The performance section begins with the two class muon suppression. The first observation is, that RDFs do not benefit from arbitrarily high numbers of trees. Every experiment clearly shows the same trend of an increasing performance until roughly 100 trees and an almost con-

stant classification rate from there on. This fact is especially relevant since the runtime of the program in the best case scales linearly with the number of trees. The dependence on the other parameters which determine the structure of a RDF in principle follows a similar trend. Too low parameter values decrease the performance but higher values than some optimum often only increase the runtime instead of the precision. Comparing the presented two class experiments, the up/down classification works best. Very likely this is the case due to the specific design of several features to capture exactly this property. Almost all features computed from the levelwise data contribute this information to a certain extent. The decent performance allows this classification to deliver a relatively reliable decision, especially with an adjusted QualityCriterion parameter. The up/down classification should not be just considered one out of many tasks, because with this information alone most undesired signals can already be rejected. The very promising performance of 97% correctly classified events at an acceptance rate of 75% will likely allow this classification to enhance the ANTARES data analysis in the near future. As a contrast the NC shower recognition presents the worst performance of the two class tasks. That is not exactly surprising, since high-energetic muons also produce showers along their track. This can result in similar distributions for some features and ultimately in an increased misclassification probability. Similar effects can be observed nicely in the four class particle identification experiment. The classes containing downgoing atmospheric muons and downgoing muon neutrinos show a considerably higher number of confusions than any other two classes. The second most confusions are not between upgoing and downgoing neutrinos, but between upgoing neutrinos and shower events. In general one trained RDF covering most classification needs, as attempted in the four class particle identification task of chapter 6.2.4, would be an extremely handy tool. Although this classification works reasonably well, nobody will want to ignore the substantially better classification rate and therefore purity at a desired acceptance rate, or the higher acceptance rate needed to achieve a desired classification rate, if a specific signal type is to be recognized.

To compare the results with already existing methods the final result from [Gey10] is shown in figure 6.45. The same setting (Two class downgoing muons and upgoing neutrino identification at 60kHz) has been evaluated with the RDF method and the results can be seen in figure 6.46 in chapter 6.6. It is relevant to keep the scaling of the y axis in mind for this comparison. The random decision forest method using the features explained in chapter 5.2 performs better for the specific task of downgoing muon rejection. This is likely caused by the nice property of RDFs to be insensitive towards weak features, whereas neural networks tend to be liable for performance impairing features. Also the features themselves were designed to emphasize the signal direction.

The results of the signal to noise ratio evaluation additionally presented for this classification task clearly show the potential of the algorithm.

Considering the robustness for many parameters the combination of SGFeatureExtract and SGClassify has demonstrated a surprisingly stable performance. For the two class muon detection the variation due to the many randomizations performed by the RDF algorithm are even bigger than the deviation between the background rates which cover the range where most other algorithms are able to work from 0 kHz up to 300 kHz. The fact that the performance for the setup without any background noise is below average can at least be made plausible by the suspicion that the feature space is partitioned more fine grained but less robust without any perturbations due to noise. This means that the RDF is not able to generalize as well without noise as for noisy training data. The picture would be the same for the four class case, if it weren't for the 100 and 200 kHz performances. Unfortunately there is no obvious explanation for this behavior and the reason for the drop of about 5% in classification rate could not be investigated.

Also a positive property is the robustness of the classification against changes of the background noise, at least for the case of decreased noise. While the RDF trained with 100 kHz looses up to 10% in classification rate, the RDF trained on 150 kHz data merely worsens, especially for low quality criteria. The results suggest that if the training cannot be exactly adapted to the background rates, a higher background rate is to be preferred. This property has already been used for the evaluation on real data in chapter 6.5.

Because SGClassify or at least SGFeatureExtract is running as a module within Seatray, it doesn't operate completely independent. The different trigger algorithms show different characteristics concerning the events they pass on. A test for the common triggers shows an increased performance for only one active trigger. This is likely due to an increased similarity for all events of one class passed to the classification. Once a second trigger is active, events from one class may be selected for two different criteria, increasing the intraclass variance.

The tests with randomly malfunctioning OMs showed that these may even increase the classification rate instead of decreasing it. What might sound impossible in the first place seems to be just a result of the triggering algorithms. The less optical modules are working properly the less events will fulfill the triggering condition. All events that are still triggered have approximately the same high probability to contain a decent signal. The effect is clearly visible once the number of classified events for one file is observed for different dead OM numbers as seen in table 6.8. This certainly is only possible since almost all feature computations operate on local data only. Therefore dead OMs further away from the brightest part of the event do not at all influence the features and by that also the classification. On the other hand events which occur directly at dead

OMs are less likely to be triggered at all. The observation that the classification performance for 300 dead OMs even increases above the ordinary level offers lots of room for speculation, but no conclusive explanation could be found. Some features could be slightly shifted towards a better separation possibility.

The performance of the classification depends on the angle of the event, since the detector isn't completely symmetric. For the very relevant up/down classification this dependence clearly shows that signals from directly above or below the detector are the easiest to classify concerning up/down separation. Events which traverse the detector almost horizontally are harder to classify, but even for these events a classification rate of 80% was achieved. It is a simple fact that, due to the different speed of light in sea water and the velocity of the muon, it is possible for an almost horizontal but still downgoing muon to be measured in a lower part of the detector first and then in the part above. To take even these cases into account a complex reconstruction would be necessary. Therefore almost horizontal events are to be considered a small weakness of this classification.

Not only the angle of the primary particle but also its energy can have a relevant influence on the classification. Instead of allowing a better identification of the event for higher energies, the classification rate drops as the energy increases, except for low energies of $10^{10}\mathrm{eV}$ to $10^{11}\mathrm{eV}$, where the performance indeed does increase. The tendency to deliver worse results for higher energies is surprising at first, but on the one hand the generation of showers along a muon track for high energies considerably changes the detected signal. This usually should not be a problem, but it adds one more uncertainty to the interpretation. To exclude this effect the up/down experiment for very high energies was repeated with a RDF trained on this energy range. Since the classification rate only increased by 4% this can be excluded as main cause. The higher probability of ultra-high-energy neutrinos to interact during their path through the Earth prevents neutrinos with these high energies to reach the detector from below. They can therefore almost exclusively be observed near the horizon, where the classification rate of this method performs worst. Since this effect of course is also contained in the Monte Carlo data, it very likely explains the unexpected drop of the classification rate. The improved performance for the lowest two energy ranges, where all neutrinos can still pass the Earth unhindered, further supports this idea.

Based on all these observations the properties of the RDF classification algorithm can be abstracted. It seems to be a well performing and very robust tool. But for the real world application the main criterion is the performance on data recorded by the ANTARES detector.

Although there seem to be some differences between the Monte Carlo data and the recorded

data in some features, for the vast majority of the features both data types agree very well. This is supported by the results for the datasets recorded in 2008, since they match the certainly correct theoretical predictions relatively well. Without a quality criterion the exact number of muons is definitely underestimated, but a higher quality criterion also cannot guarantee a correct classification. All statements for high quality criteria are certainly influenced by how well the particular event class in reality matches the Monte Carlo events. A good match for a class increases the chances of an event being classified with a very high tree agreement. Therefore relatively more events of this class will be detected for higher quality criteria. But the acceptance rate drops considerable faster for high quality criteria, see figure 6.34, than it did for the four class Monte Carlo classification, see figure 6.15. Nevertheless the tendency is correct, and the strong similarity between figure 6.32 for the real data sample and figure 6.33 for simulated muon events shows that the real muon events are reconstructed with almost exactly the same behavior as the simulated ones.

The evaluation for the point source neutrino candidates with four classes very likely reveals some minor differences between real and simulation data. Many neutrino events seem not to match the upgoing neutrino class exactly, but have some similarities with shower events (likely due to their high energies and the resulting showers along the track). One might also want to add that if shower events are considered to light homogeneously in every direction, they must be closer to upgoing events in the feature space, because the OMs are oriented approximately 45° downwards, making them more sensitive for the upgoing light of a point source than for the downgoing part. A comparison of the relative class distributions in figures 6.38 for the point source candidates and figure 6.39 for a sample consisting of 50% upgoing neutrino and 50% muons suggests that not only neutrinos but also muons could have been falsely reconstructed to be showers. The alternate explanation, that already the selection process for finding neutrino events had a high probability to confuse them with shower events seems unlikely, since already the experiments on simulation data revealed a tendency to confuse upgoing neutrinos and shower events, see for instance table 6.6. Apart from the shower events a confusion between muons and downgoing neutrinos would explain the low number of muons. If both classes are considered muons, they reach about 26.1%.

The same dataset analyzed with a RDF trained for the two class muon rejection task delivers a far more robust result, which also matches the predictions better without further interpretation. The trend of ”Others“ events being classified more certain on the other hand might indicate a RDF behavior slightly favoring an event to belong to the others class. The results of the two class upgoing neutrino identifying RDF presented in figure 6.42, show even better agreement

with the prediction and the trends are relatively stable for quality criteria up to around 0.85. For higher quality criteria some ”Others“ events are still classified with great certainty. Likely a slightly incorrect shower event class causes the high number of misclassifications for events of other classes. Since the up/down classification can't be impaired by a possibly shifted shower class, its results can be considered relatively certain. They confirm the other two class results of only around 30% downgoing atmospheric muons.

With all the results observed the conclusion must be drawn that the developed algorithms perform well on simulation data and give very reasonable results for real data as far as this can be judged. The analyses performed in ANTARES can benefit from this classification.

Chapter 8

Summary and Outlook

The goal of this thesis was to identify the different event signals observable in the ANTARES neutrino detector. To achieve this, the theoretical possibilities for particle interactions were presented and condensed to the relevant aspects for this task. Then several pattern recognition techniques were discussed, preliminarily evaluated and adapted to the specific problem setting. This includes the computation of 137 features, a feature selection with genetic algorithms, the adaptation of a random decision forest and its fine-tuning. The resulting classification tools were trained on a large number of Monte Carlo simulation data, which are indispensable since they are the only labeled data and therefore offer the only possibility to incorporate knowledge about the problem into the classifier during its training. The feature selection unfortunately didn't produce sufficiently robust optimal feature subsets and therefore the results are computed with the full feature set. The performance of the trained classifiers has been evaluated in detail for various tasks on the simulation data. The influence of all relevant parameters and external effects on the performance has been studied. For all tasks with two classes a classification rate of 80% and above was achieved rejecting no instances at all and 70% for the four class task identifying all different signal types at once. The up/down classification even showed a classification rate of 92% rejecting no event and reached 97% correctly classified events at an acceptance rate of 75%. All results can significantly be enhanced by a selective rejection of uncertain decisions. A comparison with previous work has revealed a considerable improvement. Although an evaluation on real data recorded by ANTARES cannot be exact, the application of the classification to ANTARES data from 2008 showed qualitative agreement with the predictions. A second application to a sample of point source neutrino candidates clearly showed the limitations of the four class RDF, but a two class identification of one specific signal was again possible with a relatively high accuracy, always assuming the predictions for this case are correct, which cannot

be assured with certainty. The results indicate a composition of 70% upgoing neutrinos and 30% downgoing muons. To further improve the performance for this classification several steps are possible. First of all the quality criterion based on the number of trees that agree on one class should be improved. This could be done by weighting each tree by its ”classification strength“. By this weighting the quality selection for classifications would become more stable and meaningful at the same time. The certainty of each tree on the decision it has made might also be helpful, if the individual trees were designed and evaluated to allow such a statement.
Some improvement should be found to achieve a better performance for almost horizontal tracks. Also some additional descriptive features which contribute more information would certainly be helpful. They could for instance be computed on data still containing the individual orientation for each optical module instead of one charge for the whole storey. But clever features exposing different aspects from the already used data are also possible. This should help to reduce the number of misclassifications.
One could also use optimized feature sets for each classification. This can already be easily achieved by the genetic algorithm optimization, but clearly requires a huge amount of additional time for each single classification for only a slight improvement. It would also decrease the very handy robustness of the current full feature set. A different method for searching optimal feature sets might have more success.
Another approach which could also enhance the classification performance are improved simulation data, for instance the so called ”run-by-run“ Monte Carlo. These adapt the parameter used for the simulation to what was measured during a specific data recording. A classifier trained on such adapted data might yield a slightly better performance for the corresponding recorded data.
A completely different strategy would be to incorporate the results of other algorithms as features for the classification. Strong features, for instance from a sophisticated track or shower reconstruction, would automatically be recognized as more meaningful than other features by the tree training algorithm. This is almost certain to work and could drastically reduce the number of misclassified events. However one would need to prevent the classification to rely almost exclusively on this feature to be able to correct wrong decisions of the reconstruction algorithms. The drawback can be primarily seen in the runtime, because a more complex computation usually also takes a longer time to compute. It was part of the initial specification for this thesis not to rely on such algorithms, but it should be evaluated in the future.

The current algorithm can significantly improve the identification of desired events in the ANTARES neutrino telescope. Probably the best way to achieve this is by using the up/down classification. The analyses performed in ANTARES can surely benefit from this work.

Danksagungen

An dieser Stelle möchte ich mir die Zeit nehmen, mich bei allen zu bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Zuallererst bei meiner Familie, die mich `immer` unterstüzt hat und bei Mone, weil sie auch die stressigsten Zeiten mit mir durchsteht.

Bei Professor Hornegger und Frau Professor Anton für die Vergabe dieses Themas.
Bei Professor Eskofier, Florian Folger, Wilhelm Haas und Klaus Geyer für wertvollen Rat aus den jeweiligen Fachbereichen.
Bei der Sportinformatikgruppe und der ECAP Gruppe für die angenehme Arbeitsatmosphäre.

Bei Professor Rüde, Dr. Iglberger und Peter Bürkel für sehr viel Wissen und Freude an der Wissenschaft.

Und ganz besonders möchte ich mich bei Dr. Thomas Eberl und Patrick Kugler bedanken, die mit nicht klein zu kriegendem Enthusiasmus und zu den unmöglichsten Zeiten immer ein offenes Ohr, einen klugen Rat und ein paar korrekturgelesene Seiten für mich hatten.

Vielen Dank euch allen.

Appendix A

List of features

This is the full list of all features computed and stored.
Features 1 up to 68 are computed on the full data set, while features 69 up to 137 are computed on the L1 data only.

1. Maximal storey charge of all times (max)
2. Level with the maximal charge
3. Whether the second highest charge is below or above the highest

   Search window $0.04 \cdot n_{bins}$:
4. Absolute value of the time difference (in number of bins) from the maximal charge to the second highest charge in the neighborhood
5. Average time difference from the max to all neighboring maxima
6. Highest time difference from the max to a neighboring maximum
7. Sum of the squared deviations of all time differences from the average time difference
8. Variance of the time differences
9. Average time difference to the maxima of the same level
10. Absolute value of the average time difference to the maxima of the same level
11. Average time difference to the maxima of the level above

12. Absolute value of the average time difference to the maxima of the level above
13. Average time difference to the maxima of the level below
14. Absolute value of the average time difference to the maxima of the level below

Search window $0.08 \cdot n_{bins}$:

15. Absolute value of the time difference (in number of bins) from the maximal charge to the second highest charge in the neighborhood
16. Average time difference from the max to all neighboring maxima
17. Highest time difference from the max to a neighboring maximum
18. Sum of the squared deviations of all time differences from the average time difference
19. Variance of the time differences
20. Average time difference to the maxima of the same level
21. Absolute value of the average time difference to the maxima of the same level
22. Average time difference to the maxima of the level above
23. Absolute value of the average time difference to the maxima of the level above
24. Average time difference to the maxima of the level below
25. Absolute value of the average time difference to the maxima of the level below

Search window $0.16 \cdot n_{bins}$:

26. Absolute value of the time difference (in number of bins) from the maximal charge to the second highest charge in the neighborhood
27. Average time difference from the max to all neighboring maxima
28. Highest time difference from the max to a neighboring maximum
29. Sum of the squared deviations of all time differences from the average time difference

30. Variance of the time differences
31. Average time difference to the maxima of the same level
32. Absolute value of the average time difference to the maxima of the same level
33. Average time difference to the maxima of the level above
34. Absolute value of the average time difference to the maxima of the level above
35. Average time difference to the maxima of the level below
36. Absolute value of the average time difference to the maxima of the level below

Search window $0.32 \cdot n_{bins}$:

37. Absolute value of the time difference (in number of bins) from the maximal charge to the second highest charge in the neighborhood
38. Average time difference from the max to all neighboring maxima
39. Highest time difference from the max to a neighboring maximum
40. Sum of the squared deviations of all time differences from the average time difference
41. Variance of the time differences
42. Average time difference to the maxima of the same level
43. Absolute value of the average time difference to the maxima of the same level
44. Average time difference to the maxima of the level above
45. Absolute value of the average time difference to the maxima of the level above
46. Average time difference to the maxima of the level below
47. Absolute value of the average time difference to the maxima of the level below

Level data:
Small level cluster:

48. Level containing the maximal charge
49. Size of the connected level cluster
50. Time difference of the upper half minus the lower half
51. Normalized time difference, computed as the time difference divided by the number of levels in the cluster

    Extended level cluster:

52. Time difference of the upper half minus the lower half
53. Normalized time difference, computed as the time difference divided by the number of levels in the cluster
54. Sum of the upper and the lower times
55. Sum normalized by the number of levels in the cluster
56. Variance of all times in the cluster
57. Time difference of the upper half minus the lower half, each time value weighted by $\sqrt{\mathrm{Levelcharge}}$
58. Normalized time difference, computed as the time difference divided by the number of levels in the cluster, each time value weighted by $\sqrt{\mathrm{Levelcharge}}$
59. Upper weighted sum normalized to the number of levels in the upper cluster half
60. Lower weighted sum normalized to the number of levels in the lower cluster half

    Line data:

61. Number of lines above the threshold
62. Size of the connected line cluster

    Data without time information:

63. Maximal charge seen by one storey during the whole event

64. Level which has detected the maximal charge
65. Average charge of all neighboring storeys around the maximal charge
66. Ratio of the maximal charge to the average charge around it
67. Fraction of storeys above the line threshold (computed analogously to the threshold for the line data)
68. Fraction of storeys above the average storey charge

L1 data:

69. Number of L1 hits
70. Maximal storey charge of all times (max)
71. Level with the maximal charge
72. Whether the second highest charge is below or above the highest

Search window $0.04 \cdot n_{bins}$:

73. Absolute value of the time difference (in number of bins) from the maximal charge to the second highest charge in the neighborhood
74. Average time difference from the max to all neighboring maxima
75. Highest time difference from the max to a neighboring maximum
76. Sum of the squared deviations of all time differences from the average time difference
77. Variance of the time differences
78. Average time difference to the maxima of the same level
79. Absolute value of the average time difference to the maxima of the same level
80. Average time difference to the maxima of the level above
81. Absolute value of the average time difference to the maxima of the level above

82. Average time difference to the maxima of the level below
83. Absolute value of the average time difference to the maxima of the level below

Search window $0.08 \cdot n_{bins}$:

84. Absolute value of the time difference (in number of bins) from the maximal charge to the second highest charge in the neighborhood
85. Average time difference from the max to all neighboring maxima
86. Highest time difference from the max to a neighboring maximum
87. Sum of the squared deviations of all time differences from the average time difference
88. Variance of the time differences
89. Average time difference to the maxima of the same level
90. Absolute value of the average time difference to the maxima of the same level
91. Average time difference to the maxima of the level above
92. Absolute value of the average time difference to the maxima of the level above
93. Average time difference to the maxima of the level below
94. Absolute value of the average time difference to the maxima of the level below

Search window $0.16 \cdot n_{bins}$:

95. Absolute value of the time difference (in number of bins) from the maximal charge to the second highest charge in the neighborhood
96. Average time difference from the max to all neighboring maxima
97. Highest time difference from the max to a neighboring maximum
98. Sum of the squared deviations of all time differences from the average time difference
99. Variance of the time differences

100. Average time difference to the maxima of the same level
101. Absolute value of the average time difference to the maxima of the same level
102. Average time difference to the maxima of the level above
103. Absolute value of the average time difference to the maxima of the level above
104. Average time difference to the maxima of the level below
105. Absolute value of the average time difference to the maxima of the level below

Search window $0.32 \cdot n_{bins}$:

106. Absolute value of the time difference (in number of bins) from the maximal charge to the second highest charge in the neighborhood
107. Average time difference from the max to all neighboring maxima
108. Highest time difference from the max to a neighboring maximum
109. Sum of the squared deviations of all time differences from the average time difference
110. Variance of the time differences
111. Average time difference to the maxima of the same level
112. Absolute value of the average time difference to the maxima of the same level
113. Average time difference to the maxima of the level above
114. Absolute value of the average time difference to the maxima of the level above
115. Average time difference to the maxima of the level below
116. Absolute value of the average time difference to the maxima of the level below

Level data:
Small level cluster:

117. Level containing the maximal charge

118. Size of the connected level cluster

119. Time difference of the upper half minus the lower half

120. Normalized time difference, computed as the time difference divided by the number of levels in the cluster

Extended level cluster:

121. Time difference of the upper half minus the lower half

122. Normalized time difference, computed as the time difference divided by the number of levels in the cluster

123. Sum of the upper and the lower times

124. Sum normalized by the number of levels in the cluster

125. Variance of all times in the cluster

126. Time difference of the upper half minus the lower half, each time value weighted by $\sqrt{\mathrm{Levelcharge}}$

127. Normalized time difference, computed as the time difference divided by the number of levels in the cluster, each time value weighted by $\sqrt{\mathrm{Levelcharge}}$

128. Upper weighted sum normalized to the number of levels in the upper cluster half

129. Lower weighted sum normalized to the number of levels in the lower cluster half

Line data:

130. Number of lines above the threshold

131. Size of the connected line cluster

Data without time information:

132. Maximal charge seen by one storey during the whole event

133. Level which has detected the maximal charge

134. Average charge of all neighboring storeys around the maximal charge

135. Ratio of the maximal charge to the average charge around it

136. Fraction of storeys above the line threshold (computed analogously to the threshold for the line data)

137. Fraction of storeys above the average storey charge

Appendix B

Feature selection

The feature subsets optimized for muon identification S1 and shower identification S2 in the format output by the GA.
S1: 1110101100101110100011001110010100101110000010011000110001011111110100111111101110001001111011110010011110011110111101001011010100101010000
S2: 1010010000000101001111100001101000001111011001111010011001010111110001011100100100000010111000010000110100011101010101111111101110101010

Appendix C

Further figures

Additional figures for 100 kHz background.

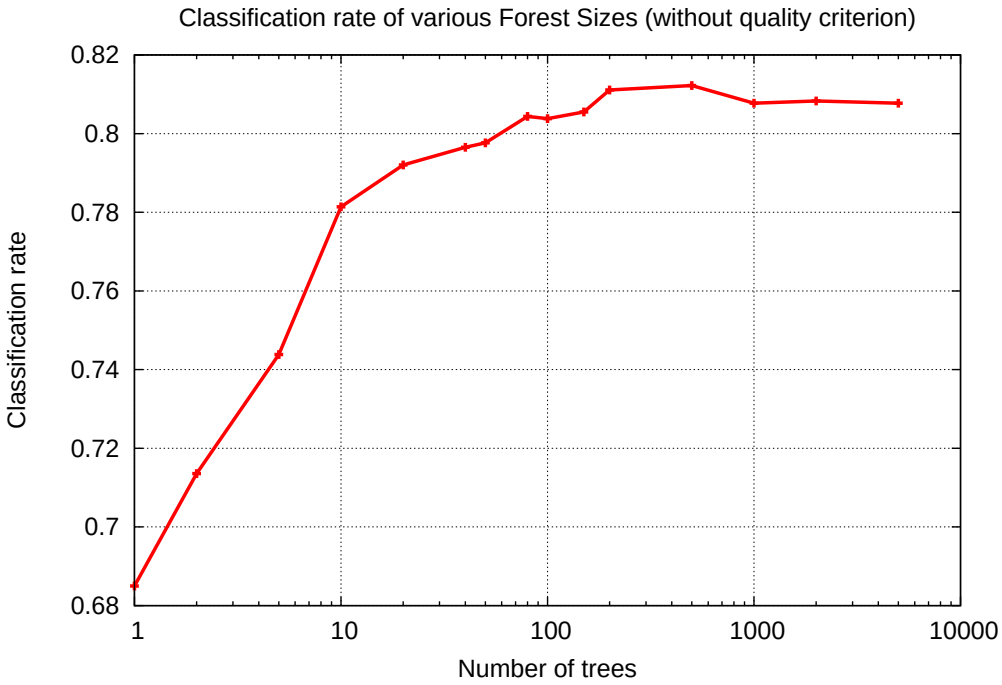

Figure C.1: Shower identification: Comparison of different forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

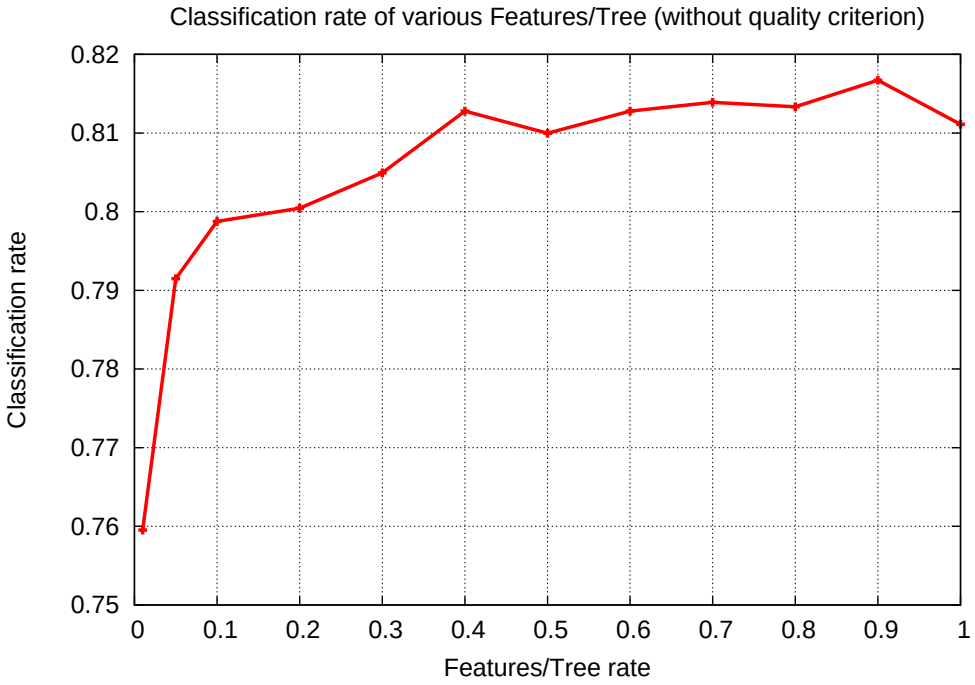

Figure C.2: Shower identification: Comparison of different Features/Tree rates, 250 Trees, 0.4 Rate Events/Tree

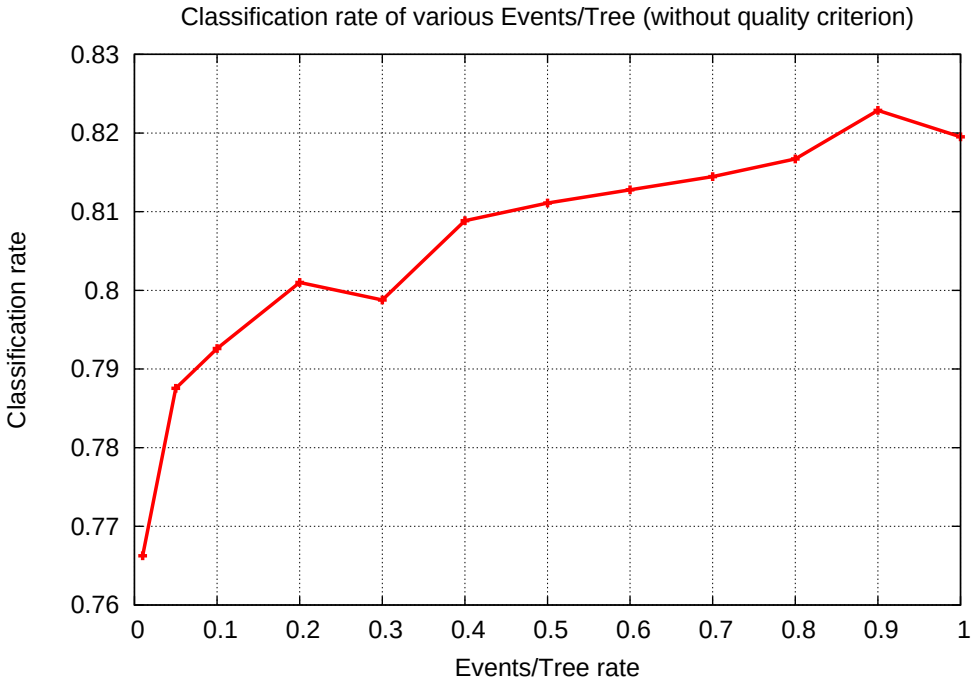

Figure C.3: Shower identification: Comparison of different Events/Tree rates, 250 Trees, 0.3 Rate Features/Tree

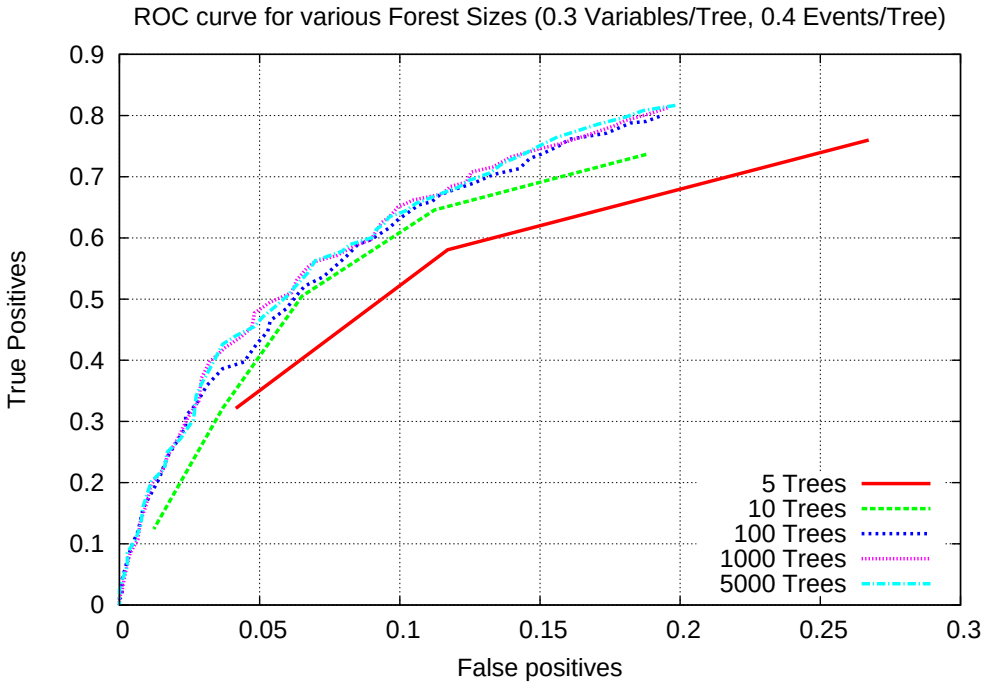

Figure C.4: Shower identification: ROC curves for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

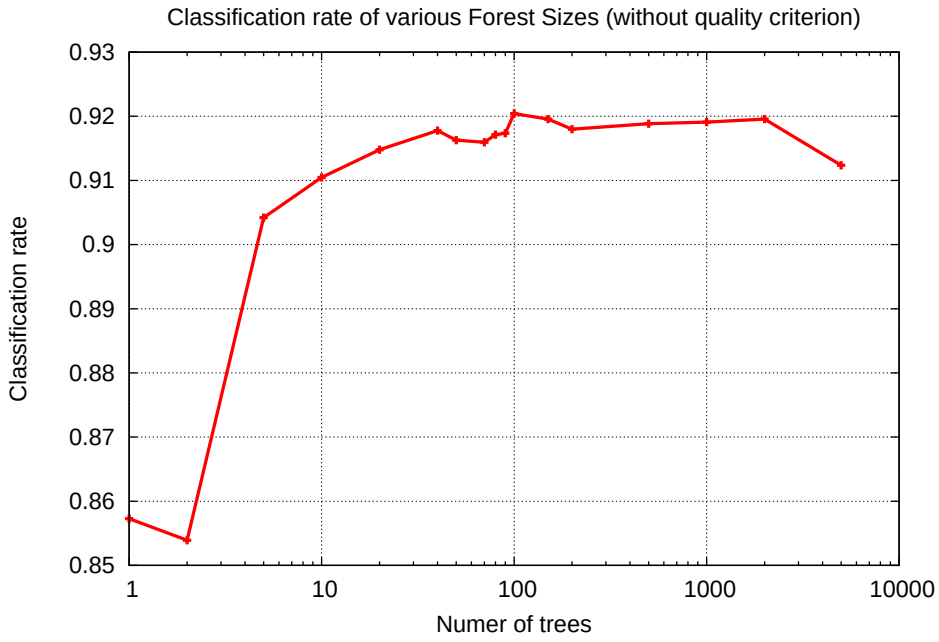

Figure C.5: Up/Down identification: Comparison of different forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

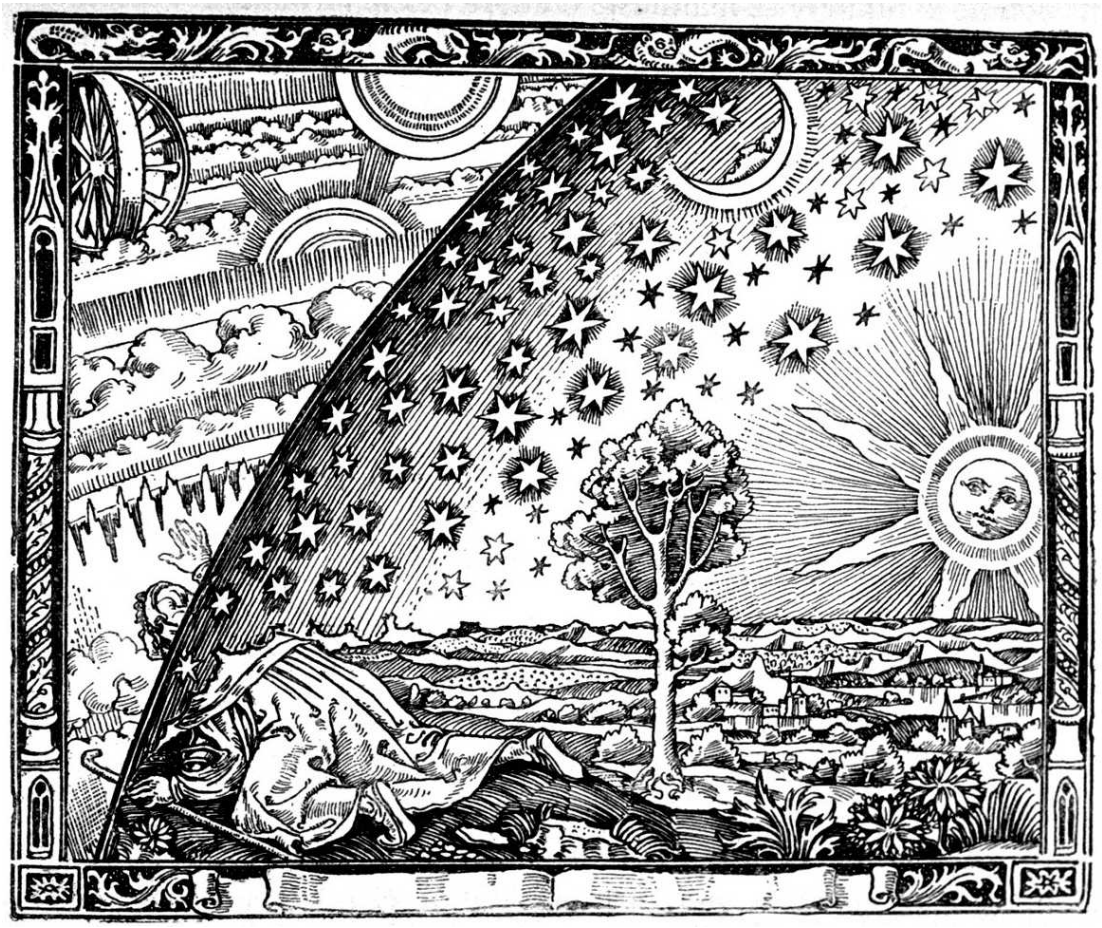

Figure C.6: This is the bonus figure. It symbolizes the search for knowledge.

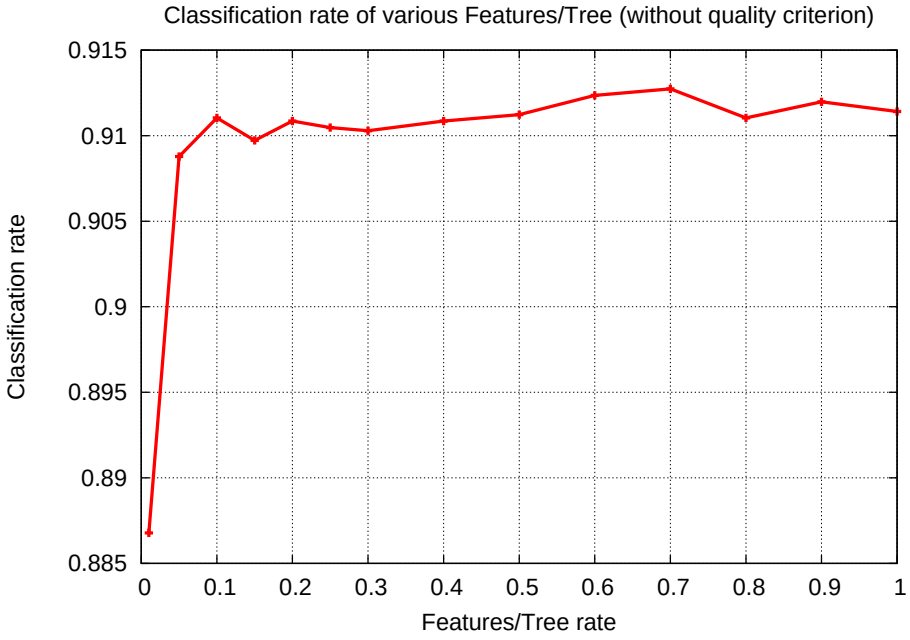

Figure C.7: Up/Down identification: Comparison of different Features/Tree rates, 250 Trees, 0.4 Rate Events/Tree

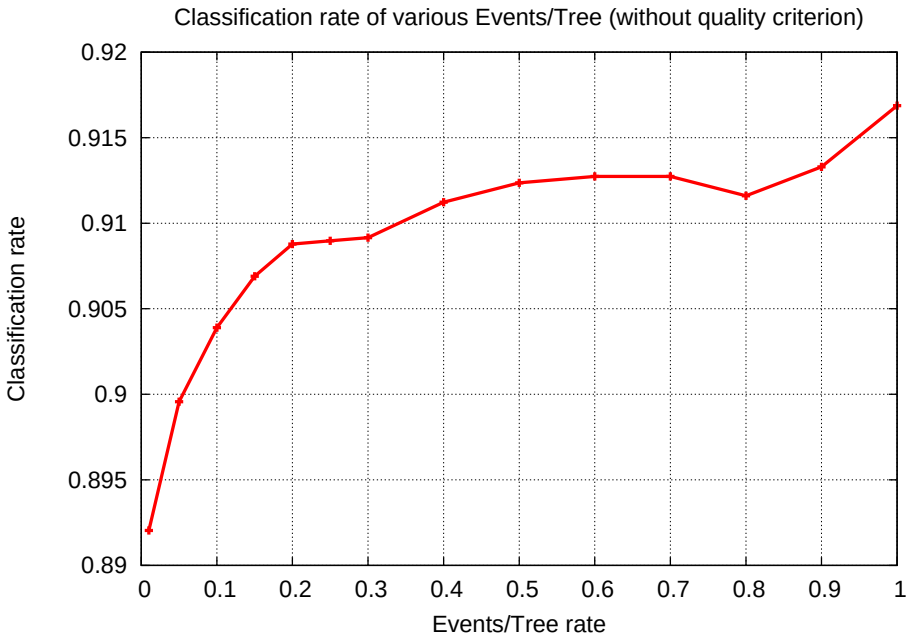

Figure C.8: Up/Down identification: Comparison of different Events/Tree rates, 250 Trees, 0.3 Rate Features/Tree

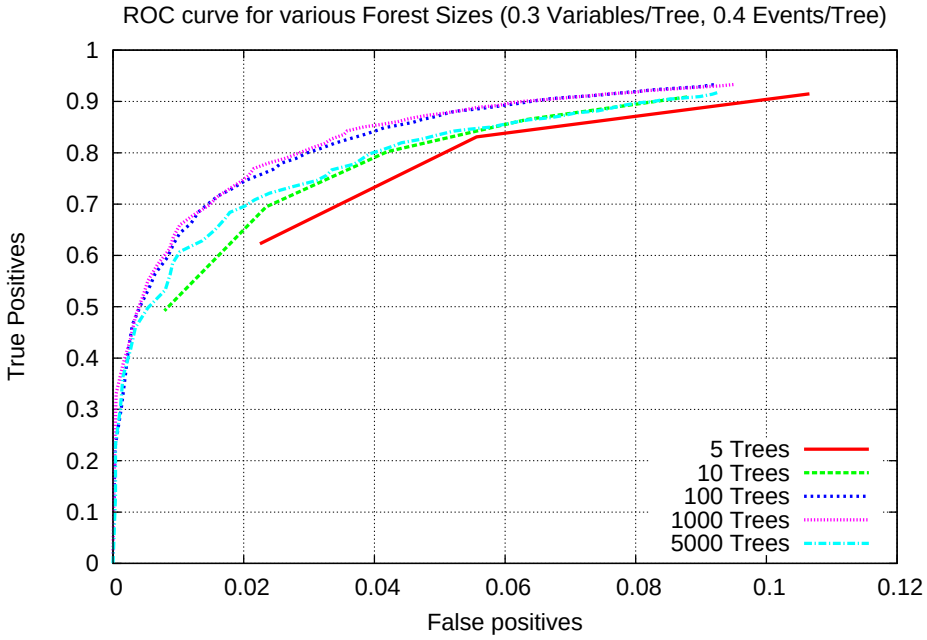

Figure C.9: Up/Down identification: ROC curves for various forest sizes, 0.3 Rate Features/Tree, 0.4 Rate Events/Tree

Appendix D

Patent investigation

This is the obligatory patent research demanded for this thesis. Since neutrino astronomy is still only relevant in research and not yet to the industry, there are no exactly matching patents. But for neutrinos in general some patents do exist, although some are more optimistic than the average patent. One example, which describes fascinating possibilities should be presented here.
The author of this thesis wishes the inventor of the following patent the best luck to achieve his ambitious goals.

Navigation system based on neutrino detection

Abstract:
A method and system for navigating is disclosed. The method and system comprises measuring the angle of arrival of neutrinos emitted by a source and tagging the neutrino measurements utilizing an accurate clock. The method and system further includes processing the tagged neutrino measurements through a computational model of a neutrino generator and combining the processed measurements with navigational aids to provide location information. A system and method in accordance with an embodiment measures angle of arrival of neutrinos generated by the sun, and therefore derives navigation information which is obtainable deep underground or underwater. Additionally, the system provides robust navigation, without drift, in the absence of other common navigation systems such as global positioning systems.

Agent: Sawyer Law Group, LLP - Palo Alto, CA, US
Inventor: Gregory M. GUTT
USPTO Applicaton #: #20090228210 - Class: 701220 (USPTO)

Since the aim of this patent seems to be a detector more likely of the size of about a backpack or a car instead of a gigantic laboratory or even a free floating detector, for this patent there has not been any danger to infringe its claims with this thesis. There also is neither an artificial neutrino beam nor a detailed evaluation of the solar neutrinos used within this thesis.

There are some more neutrino patents, but none of them seems to be applicable to this thesis.

Appendix E

Source Code

This is an example for the content of a configuration file which is needed for the training of an RDF:

```
folder /pi1/data/sgeisselsoeder/seatray/build/udclassify/featuresAlle_100kHz_Energy/
# remember the '/' at the end!
verbose 0

saveForestToDisk 0

filenameToSave Forest.dff
filenameToLoad Forest.dff
QualityCriterion 0.7

NumberOfTrees 100

RateOfVariablesPerTree 0.2
RateOfEventsPerTree 0.3
homogenTrain 1

minNumber 500
homogenTest 1
useClassMapping 1

# For the examples class 0 = athm. muons, 1 = NC Showers,
# 2 = upgoing muon neutrinos, 3 = downgoing muon neutrinos
```

```
    # without further separation
# classMap 0,-1,1,0,99 # up/down classification, shower events are not used
classMap 0,1,0,0,99 # Shower identification
#the 99 at the end helps to notice if a number is missing, since the class is then mapped to class 99

    # with energy separation
# Shower identification for ten bin energy resolved data
#classMap 0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,99
# All particle id
#classMap 0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,99
# Muon identification
#classMap 0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,99

    # with angular separation
#classMap 0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,99 # four bin angular separation, All particle id
useFeatureSelection 0

selectFeatures 1,0,1,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,1,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0,0,0,
1,1,1,1,0,1,1,0,0,1,1,1,1,0,1,0,0,1,1,0,0,1,0,1,0,1,1,1,1,1,0,0,0,1,0,1,1,1,0,1,1,0,0,1
,0,0,1,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0,1,0,0,0,0,0,1,1,0,1,0,0,0,1,1,1,0,1,0,1,0,1,0,1,1,1,1
,1,1,1,1,0,1,1,1,0,1,0,1,0
# the whole vector must be in one line, but to be printable it is separated here
# a 1 at the i_th position means use this feature, a 0 means don't use it
NumberOfEventsToSkip 0
```

The Python scripts to invoke the classification is presented here:
First the content of configfile.py:

```
#!/usr/bin/env python

# The noise rate in kHz
NoiseRateSG = 100.

# The name of the data
SimulationDataSetSG = ”RecoPulseSeriesAfterARS“

# Define the number of defect OMs
number_dead_OMs = 0

#trigger=[’3T’]
trigger=[’3T’,’3N’]
```

Parts of the script to invoke the classification (the full scripts are contained on disk):

```
#!/usr/bin/env python

... import and load whatever needed ...

import configfile

folder = ”/home/sgeisselsoeder/data/muonFileEinzeln/“;

filelist = os.listdir(folder);
random.shuffle(filelist)

for filename in filelist:
```

For simulation: read the data, do PMT simulation, L0 hits, L1 hits, Trigger, some more ...
For real data: read the data, L0 hits, L1 hits, Trigger, some more ...

```
    tray.AddModule("I3SGFeatureExtract","sgfeatureextract")(
("Class",0),
("InputFile",filename),
("DataName", configfile.SimulationDataSetSG),
("NewNoise", configfile.NoiseRateSG)
)

    tray.AddModule("I3SGClassify","sgclassify")(
("QualityCriterion",0.0),
("Classifier","/home/sgeisselsoeder/SGClassify/Forest.dff"),
)

    # The obligatory last module that terminates the chain
tray.AddModule("TrashCan", "the can")

    # ——————————————————————————————————————-

    # Now start the event loop ...
tray.Execute()

    # do some clean up
tray.Finish()
```

List of Figures

List of Tables

Bibliography

[A+10] J.A. Aguilar et al. Performance of the front-end electronics of the ANTARES neutrino telescope. *Nuclear Instruments and Methods in Physics Research A*, 622:59–73, 2010.

[A+11] J.A. Aguilar et al. ANTARES: the first undersea neutrino telescope. *Nuclear Instruments and Methods in Physics Research*, page arXiv:1104.1607, 2011.

[Bec08] J. K. Becker. High-Energy Neutrinos in the Context of Multimessenger Astrophysics. *Physics Reports*, 4-5:173–246, 2008.

[BR97] Rene Brun and Fons Rademakers. ROOT - An object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A*, 389:81–86, 1997.

[Bru08] Ronald Bruijn. *The Antares Neutrino Telescope: Performance Studies and Analysis of First Data*. PhD thesis, UNIVERSITEIT VAN AMSTERDAM, 2008.

[Col11] The Pierre Auger Collaboration. Search for first harmonic modulation in the right ascension distribution of cosmic rays detected at the Pierre Auger Observatory. *Astroparticle Physics*, 34:627–639, 2011.

[CRH+56] C. L. Jr. Cowan, F. Reines, F. B. Harrison, H. W. Kruse, and A. D. McGuire. Detection of the Free Neutrino: A Confirmation. *Science*, 124(3212):103–104, 1956.

[CS10] T. Chiarusi and M. Spurio. High-energy astrophysics with neutrino telescopes. *The European Pysical Journal C*, 65:649–701, 2010.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley Interscience, 2001.

[F+98] Fukuda et al. Evidence for Oscillation of Atmospheric Neutrinos. *Physical Review Letters*, 81(8):1562–1567, Aug 1998.

[Faw06] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[G+96] Raj Gandhi et al. Ultrahigh-Energy Neutrino Interactions. *Astroparticle Physics*, 5:81–110, 1996.

[Gey10] Klaus Geyer. Ereignisklassifikation mit Hilfe von Neuronalen Netzen für das ANTARES Neutrino-Teleskop. Diploma thesis, Erlangen Center for Astroparticle Physics, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2010.

[Gre66] Kenneth Greisen. End to the Cosmic-Ray Spectrum? *Physical Review Letters*, 16:748–750, 1966.

[HDW94] G. Holmes, A. Donkin, and I.H. Witten. WEKA: A Machine Learning Workbench. In *Second Australia and New Zealand Conference on Intelligent Information Systems*, Brisbane, Australia, 1994.

[Hil06] A. M. Hillas. Cosmic Rays: Recent Progress and some Current Questions. 2006. arXiv:astro-ph/0607109v2.

[HK10] Francis Halzen and Spencer R. Klein. IceCube: An Instrument for Neutrino Astronomy. *Review of Scientific Instruments*, 81:1–24, 2010.

[Ho98] Tin Kam Ho. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:832–844, 1998.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Atrificial Intelligence*. The University of Michigan Press, 1975.

[HP97] Joachim Hornegger and Dietrich W. R. Paulus. *Pattern Recognition of Images and Speech in C++*. Verlag Vieweg, 1997.

[HPJ+09] A. Hoecker, P.Speckmayer, J.Stelzer, J.Therhaag, E. von Thoerne, and H.Voss. *TMVA 4 Toolkit for Multivariate Data Analysis with ROOT Users Guide*, 2009. http://tmva.sourceforge.net.

[Jan33] K. G. Jansky. Radio Waves from Outside the Solar System. *Nature*, 132:66, July 1933.

[MWK+06] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. YALE: Rapid Prototyping for Complex Data Mining Tasks. In T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, editors, *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 935–940. ACM Press., 2006.

[N+10] M. Neff et al. Signal classification for acoustic neutrino detection. *Nuclear Instruments and Methods in Physics Research*, A:arXiv:1104.3248v1, 2010.

[Pau30] Wolfgang Pauli. Offener Brief an die Gruppe der Radioaktiven bei der Gauvereins-Tagung zu Tübingen. Open letter, 12 1930.

[Pea99] John A. Peacock. *Cosmological Physics*. Cambridge University Press, 1999.

[Pol94] Robi Polikar. The Wavelet Tutorial. http://users.rowan.edu/∼polikar/wavelets/wttutorial.html, 1994.

[PS03] Lutz Priese and Patrick Sturm. Introduction to the Color Structure Code and its Implemenatation. www.uni-koblenz.de/∼lb/lb_downloads/download/csc.pdf, March 2003.

[Qui85] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1985.

[RMS11] Tim Ruhe, K. Morik, and Benjamin Schowe. Data mining on Ice. In *Proceedings of the GREAT workshop on Astrostatistics and data mining 2011*, 2011.

[TW10] Colin Kelley Thomas Williams. *gnuplot 4.4 An Interactive Plotting Program*, 2010. http://www.gnuplot.info/.

[Wag10] Stefanie Wagner. Entwicklung und Analyse eines Hitselektionsalgorithmus für Niederenergieereignisse im ANTARES Neutrinoteleskop basierend auf einer Hough-Transformation. Master's thesis, Erlangen Center for Astroparticle Physics, Physikalisches Institut Friedrich-Alexander-Universität Erlangen-Nürnberg, 2010.

[Wal96] Matthew Wall. *GAlib: A C++ Library of Genetic Algorithm Components Version 2.4*. Mechanical Engineering Department Massachusetts Institute of Technology, 1996. http://lancet.mit.edu/ga/.

[Wit08] Thomas Wittenberg. Algorithms for Pattern Recognition and their Realisation II - Selective Undergraduate Course, 04 2008. Friedrich-Alexander-Universität Erlangen-Nürnberg, Lecture Slides.

[Y+06] Yao et al. Review of Particle Physics. *Journal of Physics G: Nuclear and Particle Physics*, 33:1, 2006.