Statistical Learning in High Energy and Astrophysics

Jens Zimmermann

München 2005

Statistical Learning in High Energy and Astrophysics

Jens Zimmermann

Dissertation
an der Fakultät für Physik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Jens Zimmermann
aus Berlin

München, den 16. Juni 2005

Erstgutachter: Prof. Dr. Christian Kiesling
Zweitgutachter: Prof. Dr. Paul Tavan
Tag der mündlichen Prüfung: 24. Oktober 2005

Contents

1 Introduction and Motivation **1**

2 Experiments, Detectors and Physics Motivation **5**
2.1 The H1 Experiment at HERA . . . 5
2.1.1 HERA . . . 6
2.1.2 H1 Experiment . . . 7
2.1.3 Trigger System . . . 9
2.1.4 The Level 2 Neural Network Trigger . . . 12
2.1.5 Important Physics Channels for L2NN . . . 14
2.1.6 Offline Analysis: Instantons . . . 19
2.2 Higgs Boson Parity Measurement at a Future Linear Collider . . . 21
2.3 Small-Angle Neutron Scattering Detector . . . 24
2.4 The MAGIC Telescope . . . 26
2.4.1 Extensive Air Showers and Imaging of Cherenkov Light . . . 28
2.4.2 Shower Image Analysis and Background Rejection . . . 31
2.4.3 Energy Estimation . . . 35
2.5 The XEUS Satellite . . . 36
2.5.1 The Mesh-Experiment . . . 40

3 Statistical Learning for Physics Experiments **41**
3.1 Statistical Learning in the World of Artificial Intelligence . . . 41
3.2 Inputs . . . 42
3.3 Supervised and Unsupervised Learning . . . 43
3.4 Classification vs. Regression . . . 45
3.5 Online vs. Offline . . . 45
3.6 Preprocessing . . . 46
3.7 Standard Approach to Classification: “Cuts” . . . 47
3.8 Standard Approach to Regression: “Fit” . . . 49
3.9 Knowledge and Time . . . 50
3.10 Prerequisite: Training Data . . . 52
3.11 Overtraining and Regularisation . . . 53
3.12 Performance Evaluation . . . 55
3.12.1 Performance Evaluation for Classification . . . 56
3.12.2 Performance Evaluation for Regression . . . 59
3.13 Calculation of Uncertainties for Statistical Learning Methods . . . 59
3.13.1 Statistical Uncertainties . . . 60
3.13.2 Systematic Uncertainties . . . 63

3.14 Data Mining . . . 65
3.15 Comparison of Statistical Learning Methods . . . 67
3.15.1 Comparing Hypotheses . . . 68
3.15.2 Comparing Learning Methods . . . 69

4 Statistical Learning Theory 71
4.1 Error Measurement . . . 71
4.2 Bayesian Learning . . . 73
4.3 PAC Learning (Probably Approximately Correct) . . . 74
4.4 The VC-Framework (Vapnik-Chervonenkis) . . . 75
4.5 Criticism: No-Free-Lunch Theorems . . . 77
4.6 Regularisation Schemes . . . 79
4.6.1 Occam's Razor . . . 79
4.6.2 MDL Principle (Minimum Description Length) . . . 79
4.6.3 Structural Risk Minimisation . . . 80

5 Statistical Learning Methods 81
5.1 Model-Based vs. Instance-Based Methods . . . 82
5.2 Decision Trees . . . 82
5.3 Local Density Estimators . . . 84
5.3.1 k-Nearest-Neighbours . . . 84
5.3.2 Kernel Methods . . . 85
5.3.3 Range Search . . . 86
5.3.4 Naive Bayes . . . 87
5.4 Methods Based on Linear Separation . . . 89
5.4.1 Linear Discriminant Analysis . . . 89
5.4.2 Neural Networks . . . 90
5.4.3 Support Vector Machines . . . 93
5.5 Meta Learning Strategies . . . 95
5.5.1 Stacking . . . 96
5.5.2 Bagging . . . 96
5.5.3 Random Subspace Method . . . 97
5.5.4 Boosting . . . 97
5.6 Typical Properties of Different Classification Methods . . . 98
5.6.1 Toy Example 'Hole' . . . 98
5.6.2 Toy Example 'Rings' . . . 101
5.6.3 Toy Example 'Gaussians' . . . 103
5.6.4 Summary of Typical Properties of Different Classification Methods 106

6 Software Development 109
6.1 Data Access and Preprocessing . . . 109
6.2 Data Visualisation . . . 110
6.3 Statistical Learning Methods and their Automation . . . 111
6.4 Performance Evaluation and Control . . . 112

7 Analysis and Results **115**
7.1 Overview . . . 115
7.2 Applications for H1: The Level 2 Neural Network Trigger . . . 117
7.2.1 Training Data . . . 117
7.2.2 Performance Check . . . 118
7.2.3 Deeply Virtual Compton Scattering . . . 119
7.2.4 Charged Current Interactions . . . 131
7.2.5 Exclusive J/ψ Photoproduction – $J/\psi \to e^+e^-$ . . . 139
7.2.6 Inelastic J/ψ Photoproduction – $J/\psi \to \mu^+\mu^-$ . . . 142
7.2.7 D^* and Dijet production . . . 145
7.2.8 Summary of Newly Developed Neural Networks . . . 149
7.3 Applications for H1: Instanton Purification . . . 150
7.3.1 Datasets . . . 150
7.3.2 Evaluation Strategy . . . 151
7.3.3 Training and Search Strategy . . . 154
7.3.4 Verification of Previous Results . . . 154
7.3.5 New Results . . . 158
7.4 Higgs Boson Parity Measurement at a Future Linear Collider . . . 162
7.4.1 Data Source and Preprocessing . . . 162
7.4.2 Training and Evaluation Strategy . . . 162
7.4.3 Results . . . 165
7.4.4 Future Research . . . 169
7.5 Position Measurement for a Small-Angle Neutron Scattering Detector . . . 170
7.5.1 Data Sources . . . 170
7.5.2 Preprocessing . . . 171
7.5.3 Results . . . 171
7.6 Applications for the MAGIC Telescope . . . 177
7.6.1 Gamma-Hadron Separation . . . 177
7.6.2 Energy Estimation . . . 187
7.6.3 Future Research . . . 193
7.7 Applications for the XEUS satellite . . . 195
7.7.1 Pileup Rejection . . . 195
7.7.2 Sub-pixel Resolution . . . 203

8 Discussion **213**
8.1 Physics Results with Statistical Learning Methods . . . 213
8.2 Performance of Statistical Learning Methods . . . 214
8.3 Control of Statistical Learning Methods . . . 215
8.4 Handling of Statistical Learning Methods . . . 217
8.5 The "Best Learning Method" . . . 218
8.6 Artificial Intelligence in Statistical Learning Methods . . . 218
8.7 The Future of Statistical Learning in Physics Analysis . . . 219

9 Conclusion **221**

A Statistical Learning Methods in Hardware **223**

B Implementation Details **227**
B.1 Preprocessing for the Pixel-detector 227
B.1.1 Correction of detector and readout effects 227
B.1.2 Signal Extraction 228
B.2 Analysis of the Mesh Experiment 229
B.3 Automatic Parameter Optimisation 231
B.3.1 Parameters for Neural Networks 231
B.3.2 Parameters for Support Vector Machines 232
B.3.3 Parameters for Random Forests 232

Index **242**

Danksagung **250**

Abstract

This thesis studies the performance of statistical learning methods in high energy and astrophysics where they have become a standard tool in physics analysis. They are used to perform complex classification or regression by intelligent pattern recognition. This kind of artificial intelligence is achieved by the principle "learning from examples": The examples describe the relationship between detector events and their classification.

The application of statistical learning methods is either motivated by the lack of knowledge about this relationship or by tight time restrictions. In the first case learning from examples is the only possibility since no theory is available which would allow to build an algorithm in the classical way. In the second case a classical algorithm exists but is too slow to cope with the time restrictions. It is therefore replaced by a pattern recognition machine which implements a fast statistical learning method. But even in applications where some kind of classical algorithm had done a good job, statistical learning methods convinced by their remarkable performance.

This thesis gives an introduction to statistical learning methods and how they are applied correctly in physics analysis. Their flexibility and high performance will be discussed by showing intriguing results from high energy and astrophysics. These include the development of highly efficient triggers, powerful purification of event samples and exact reconstruction of hidden event parameters.

The presented studies also show typical problems in the application of statistical learning methods. They should be only second choice in all cases where an algorithm based on prior knowledge exists. Some examples in physics analyses are found where these methods are not used in the right way leading either to wrong predictions or bad performance. Physicists also often hesitate to profit from these methods because they fear that statistical learning methods cannot be controlled in a physically correct way. Besides there are many different statistical learning methods to choose from and all the different methods have their advantages and disadvantages – compared to each other and to classical algorithms.

By discussing several examples from high energy and astrophysics experiments the principles, advantages and weaknesses of all popular statistical learning methods will be analysed. A focus will be put on neural networks as they form some kind of standard among different learning methods in physics analysis.

Zusammenfassung

Die vorliegende Arbeit untersucht die Leistungsfähigkeit von statistischen Lernmethoden in den Bereichen der Hochenergie- und Astrophysik, wo sie heute zu einem Standardhilfsmittel der physikalischen Analyse geworden sind. Sie werden für komplexe Klassifikations- oder Regressionsaufgaben eingesetzt, die sie durch intelligente Mustererkennung bewältigen. Diese Form der künstlichen Intelligenz wird durch das Prinzip "Lernen an Beispielen" erreicht, wobei die Beispiele den Zusammenhang zwischen den Detektordaten und deren Klassifikation darstellen.

Gründe für die Anwendung von statistischen Lernmethoden sind entweder ein Mangel an Wissen über diesen Zusammenhang oder zeitliche Beschränkungen. Im ersten Fall ist das Lernen an Beispielen der einzig gangbare Weg, da keine Theorie vorhanden ist, die das Erstellen eines Algorithmus' auf klassischem Wege erlauben würde. Im zweiten Fall existiert ein klassischer Algorithmus, der aber zu langsam und damit angesichts der vorgegebenen Zeitanforderungen ungeeignet ist. Er wird deshalb von einer Mustererkennungsmaschine ersetzt, die eine schnelle statistische Lernmethode implementiert. Aber sogar in Anwendungen, in denen ein klassischer Algorithmus gute Dienste tat, überzeugten die Ergebnisse statistischer Lernmethoden.

Diese Arbeit gibt eine Einführung in statistische Lernmethoden und ihre korrekte Anwendung in der physikalischen Analyse. Ihre Flexibilität und hohe Leistungsfähigkeit werden diskutiert, indem eindrucksvolle Resultate aus der Hochenergie- und Astrophysik gezeigt werden. Darunter fallen die Entwicklung hoch effizienter Trigger, die effiziente Bereinigung von Datensätzen und die exakte Rekonstruktion versteckter Parameter eines Ereignisses.

Die vorliegende Untersuchung zeigt darüber hinaus auch typische Probleme der Anwendung statistischer Lernmethoden auf. Sie sollten in jenen Fällen nur die zweite Wahl sein, in denen ein auf Vorwissen basierender Algorithmus vorhanden ist. Bei nicht korrekter Anwendung führen sie zu falschen Vorhersagen oder schlechten Leistungen, wie leider Beispiele in physikalischen Analysen belegen. Physiker zögern auch oft von diesen Methoden zu profitieren, weil sie befürchten, dass statistische Lernmethoden nicht in physikalisch adäquater Weise kontrolliert werden können. Außerdem gibt es viele verschiedene statistische Lernmethoden, unter denen ausgewählt werden muss. Jede einzelne Methode hat – verglichen mit den anderen Lernmethoden und klassischen Algorithmen – ihre eigenen Vor- und Nachteile.

Die Diskussion zahlreicher Beispiele aus Experimenten der Hochenergie- und Astrophysik analysiert die Prinzipien, Vorteile und Schwächen aller gängigen statistischen Lernmethoden. Ein Schwerpunkt liegt dabei auf neuronalen Netzen, da diese eine Art Standard unter den verschiedenen Lernmethoden in der physikalischen Analyse darstellen.

Résumé

Cette thèse de doctorat étudie la performance d'approches de l'apprentissage statistiques dans les domaines de la physique des hautes energies et de l'astrophysique où elles sont devenues un outil standard pour l'analyse physique. On les utilise pour des problèmes de classification ou de régression complexes qu'elles surmontent à l'aide de Reconnaissance de Formes intelligente. Cette forme d'intelligence artificielle est atteinte par le principe de "l'apprentissage à partir d'exemples", c'est-à-dire les exemples représentent la relation entre les donnés du détecteur et leur classification.

Les raisons pour l'application d'approches de l'apprentissage statistiques sont soit un défaut de savoir sur cette relation soit des limitations dans le temps. Dans le premier cas, l'apprentissage à partir d'exemples est la seule possibilité car il n'existe aucune théorie qui permettrait d'élaborer un algorithme sur la voie classique. Dans le deuxième cas, il y a un algorithme classique qui travaille trop lentement et qui est incapable de remplir les conditions temporelles. C'est pour cela qu'il est remplacé par une machine à Reconnaissance de Formes qui implante une approche de l'apprentissage statistique rapide. Même pour les applications où un algorithme classique convenait bien, les résultats des approches de l'apprentissage statistiques ont convaincu.

Cette thèse de doctorat donne une introduction aux approches de l'apprentissage statistiques et à leur application correcte dans l'analyse physique. Leur flexibilité et puissance seront discutées en montrant de résultats impressionnants de la physique des hautes énergies et de l'astrophysique. Parmi ces résultats on trouvera le développement de déclenchements très efficaces, l'épuration effective d'enregistrements et la reconstruction exacte de paramètres cachés d'un événement.

En plus, cette enquête montre des problèmes typiques de l'application des approches de l'apprentissage statistiques. Elles devraient être seulement le second choix dans tous les cas où un algorithme basé sur la connaissance des dépendances fonctionnelles existe. Utilisées de fa con incorrecte, les approches de l'apprentissage statistiques conduisent à de fausses prédictions ou à des performances médiocres comme le montre, hélas, des exemples dans des analyses physiques. Des physiciens hésitent aussi souvent de profiter des ces approches car ils craignent qu'on ne peut pas contrôler les approches de l'apprentissage statistiques de fa con physicalement adéquate. De plus, il y a beaucoup d'approches de l'apprentissage statistiques entre lesquelles il faut choisir. Chaque approche a – comparée avec les autres et avec les algorithmes classiques – ses propres avantages et inconvénients.

La discussion de nombreux exemples tirés d'expériences de la physique des hautes énergies et de l'astrophysique analyse les principes, avantages et défaillances de toutes les approches de l'apprentissage statistiques habituelles. Un accent sera mis sur les réseaux neuronaux, car ils représentent une sorte de standard parmi les différentes approches de l'apprentissage dans l'analyse physique.

Chapter 1

Introduction and Motivation

Statistical learning methods arise from the idea to leave pattern recognition tasks to the computer that could only be done by humans so far. They are used in physics analysis to perform a classification (usually to suppress some kind of background) or a regression (to reconstruct quantities which have not been measured directly).

All kinds of statistical learning methods have found their way into physics analysis because experimental data – especially the large amount of data coming from modern high energy and astrophysics experiments – demands sophisticated human-like analysis done by computers. While most of this analysis is done offline, i.e. stored data is processed, online applications can already be found where a statistical learning method decides within fractions of milliseconds whether an event coming from the detector should be stored or whether it should be considered as background which should be rejected.

The application of statistical learning methods in physics analysis has three roots: The basis of mathematical statistics, the fascination of artificial intelligence and of course the need of physics analysis.

Mathematical statistics provides reliable methods to analyse data from stochastic processes. Based on the underlying theory of probability and sampling, mathematical statistics is the foundation for a correct description of statistical learning. Apart from the basis of any theoretical treatment of statistical processes it also provides basic methods to describe models and make predictions.

The field of **artificial intelligence** is driven by the goal to make machines as intelligent as humans. Naturally, this target has its own fascination and motivation. But as soon as machines were able to cope with at least basic tasks they were immediately employed in all kinds of problems due to their high performance, speed, predictability and finally cheapness compared to human workers. The field of statistical learning is one of the major components of current research in artificial intelligence. The different fields of application range from industrial processes and banking over medicine to science.

Analysis of **physics experiments**, especially in high energy and astrophysics, is a part of science that has always required the most recent developments in computational power and analysis methods. The branch of computer aided physics emerged from this constantly rising need for fast, reliable, understandable and clever algorithms in modern physics analysis.

The complexity of pattern recognition problems leads to the principle of "learning from examples": Every statistical learning method needs to be trained with examples of the correct behaviour. Mathematically examples $(\vec{x}_i, y_i)$ of inputs $\vec{x}$ and corresponding target

y are given and a rule $\vec{x} \rightarrow out(\vec{x})$ should be inferred where $out(\vec{x}_i)$ should come as close to y_i as possible. In a classification problem the target value distinguishes typically two classes, the signal and the background. Any event given to the trained learning method is classified by using the complex detector information inside the input vector: An event decision is performed by pattern recognition based on the event signature.

Choosing a statistical learning method instead of some classical algorithm may have different reasons. One big advantage of statistical learning methods is their ability to construct a functional dependence only by using examples. If the detector response is not completely understood, if measured data does not match theoretical predictions or no theory is available at all: In all these cases the lack of knowledge forbids the development of any algorithm by hand, whereas statistical learning does not need this knowledge. Furthermore, classically developed algorithms often turn out to perform much worse than a statistical learning method in the same place because the available knowledge was either not sufficient to build a better algorithm or the prediction did not agree with the measured data. If a classically developed algorithm performs at least as good as a statistical learning method finally the need for a very fast processing may result in the decision to use statistical learning for which very fast hardware implementations exist.

Although there may be convincing arguments to use a statistical learning method, physicists sometimes hesitate to do so because they fear that this kind of algorithm is not well understood, introduces some kind of additional uncertainty or cannot be handled or controlled in an appropriate way. In addition the large variety of statistical learning methods makes the choice difficult.

In this thesis an introduction to statistical learning with all its finesses will be given, common problems and questions will be discussed and a clear guide to the successful and correct application of these methods to physics experiments will be provided. Furthermore a comparative study among statistical learning methods and classical algorithms will reveal the huge potential of statistical learning which will then lead to physics results that could not have been obtained without the help of these methods.

A completely new kind of intimate connection between artificial intelligence and its application in physics analysis will be presented. The existing literature does not cover the large gap between algorithms and their theory on the one side and the needs of physics analysis on the other side. Typical papers or PhD theses describe the application of one specific algorithm to one specific problem in physics analysis. Almost no background of general concepts of such kind of algorithms or a theoretical analysis of their properties is presented. Also no overview over different fields of application of these methods and their general behaviour in physics applications is given. This thesis provides the missing overview and generalisation, the complete discussion of typical problems and their solutions. Many new physics results will be presented as a direct product from the successful application of different statistical learning methods in different physics experiments.

Chapter 2 (Experiments, Detectors and Physics Motivation) will start with a description of all the experiments from high energy and astrophysics on which the following analysis will be based. Chapter 3 (Statistical Learning for Physics Experiments) will then provide a complete practice oriented introduction to the application of statistical learning methods while chapter 4 (Statistical Learning Theory) covers the basics of the theoretical description of these methods based on mathematical statistics. Chapter 5 (Statistical Learning Methods) presents the most popular learning methods in detail by emphasising the underlying ideas and geometrical interpretations. The software framework which was

created to perform the studies presented in this thesis is introduced in chapter 6 (Software Development). Chapter 7 (Analysis and Results) provides a description of all analysis steps and their results for each of the experiments. Since this chapter will present all results and all insights into the successful application of statistical learning methods experiment-wise, chapter 8 (Discussion) summarises the most important results by allowing an overview over the similarities among the different applications. Finally, chapter 9 (Conclusion) gives a brief summary of the main results of this thesis.

Chapter 2

Experiments, Detectors and Physics Motivation

The experiments which will be discussed in this chapter will fulfil different purposes. With their broad range from high energy physics to astrophysics they provide valuable examples for different aspects of statistical learning as discussed in chapter 3. Furthermore, they form the basis for the analysis of different statistical learning methods and their comparison among each other and to classical methods. Finally, the application of statistical learning methods in these experiments results directly or indirectly in intriguing physics results that could not have been obtained without statistical learning.

The following section 2.1 will start with the H1 experiment at the electron-proton collider HERA. Two different applications for statistical learning methods at H1 will be introduced: background suppression is needed on the one hand already inside the trigger system and on the other hand in any analysis which needs a pure event sample. Section 2.2 continues with high energy physics: the determination of the Higgs boson parity at a future linear collider is discussed. Section 2.3 will then introduce a neutron detector: here the incident position of the neutron needs to be reconstructed. In section 2.4 the MAGIC telescope will be introduced. This Cherenkov telescope can make use of statistical learning methods in the suppression of events which have been induced by a cosmic hadron and in the estimation of the energy of a cosmic photon. Finally section 2.5 moves on to another astrophysics experiment: the XEUS satellite. The X-ray pixel-detector which will be used on this satellite poses two problems to statistical learning methods: Pileup events in which the signals of two or more photons overlay have to be rejected and the incident position of each photon should be reconstructed with sub-pixel resolution.

2.1 The H1 Experiment at HERA

At HERA, which will be described in the following section, electrons/positrons and protons are brought to collision. In lowest order (Born approximation) their interaction (*ep*-scattering) is described by the exchange of a gauge boson (γ^*, Z^0, W^+, W^-) between the electron/positron and a parton of the proton. This exchange is called neutral current if the exchanged boson is neutral (γ^* or Z^0) or charged current if the exchanged boson is charged (W^+, W^-). The interaction of the point-like electron/positron with the proton makes it possible to probe the partonic structure of the proton and to investigate electroweak effects

at the same time.

The classification of events which are observed in the detector is one of the most important analysis steps. The trigger system has to make sure that all interesting *ep*-interactions are logged but that the background, coming mostly from interactions of the proton beam with residual gas or detector components, is suppressed enough to have an acceptable logging rate. Fast statistical learning methods can cope with this task since they can recognise the patterns of different event types. This "online" trigger application will be introduced in section 2.1.4.

Classifying events is also very important in any analysis which relies on very clean datasets in which a specific class of events should be enriched and all competing background should be suppressed as much as possible. Section 2.1.6 will introduce the search for instantons as an "offline" analysis where this kind of background suppression is important.

2.1.1 HERA

The HERA collider (Hadron-Elektron-Ring-Anlage) [1] at DESY (Deutsches Elektronen Synchrotron), Hamburg, accelerates electrons/positrons and protons up to $27.5GeV$ and $920GeV$ respectively in $6.3km$ long storage rings. Figure 2.1 shows the ring, the pre-accelerators and the experiments. There are two colliding experiments H1 (north) and ZEUS (south) and two fixed target experiments HERMES (east) and HERA-B (west).

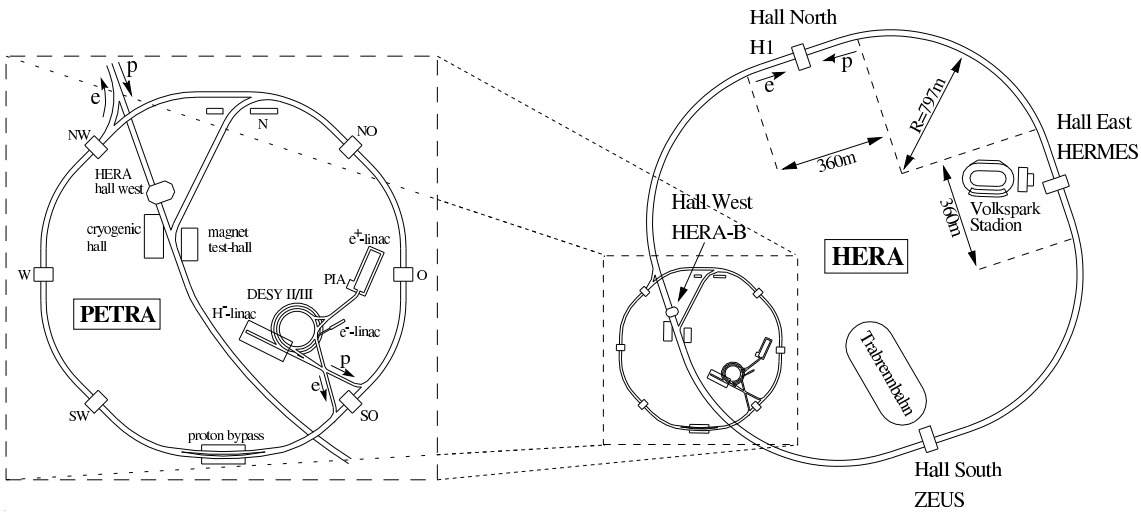

Figure 2.1: The HERA collider with its four experiments (right) and the structure of the pre-accelerators magnified (left).

The electrons/positrons and protons travel in small packets (*bunches*) through the rings. Every 96 ns a pair of them is brought to collision (*bunch-crossing*) at the two interaction points inside the H1 and ZEUS experiment. Figure 2.2 shows a typical fill of positrons and protons in the HERA collider in 2004. The positron current decreases strongly over time because of synchrotron radiation. The time of data-taking is given by the luminosity which starts shortly after both beams have been injected and stops when the beams are dumped after the positron current has decreased below a certain value. This time of data-taking during one fill is separated into many runs. Each run defines a period of (more or less) stable detector operation.

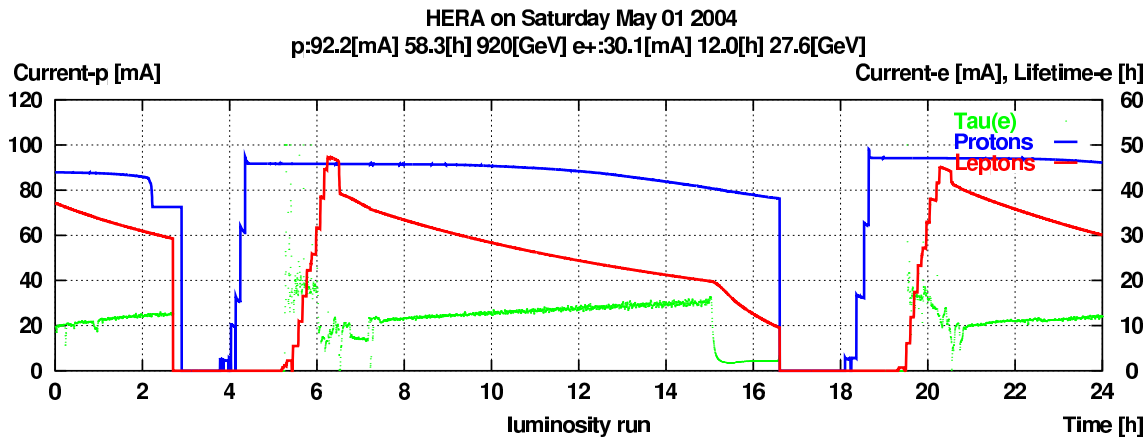

Figure 2.2: HERA currents on 01.05.04: Proton currents around 90 mA and initial positron currents around 40 mA were reached. The fill in the middle allowed data-taking from 08.00 am to 04.30 pm.

With the end of 2003 the data taking of the HERA II period started with the completion of a major upgrade of HERA. Two important changes were done: The instantaneous luminosity was increased by installing additional focusing magnets around the colliding experiments and it was made possible to use polarised electrons/positrons by installing spin rotators[1] and instruments which measure the polarisation. The idea of the luminosity upgrade is of course to collect more physics events (the rate is expected to increase about a factor 3-5) but, much more dramatic, it may lead to much higher background rates because of increased synchrotron radiation and beam-beam effects. This trigger-problem will be discussed below.

2.1.2 H1 Experiment

The H1-Experiment [2] is shown schematically in figure 2.3. The layered structure is cylindrically symmetric to the beam axis. Because of the high proton energy the centre-of-mass system is boosted in the positive z-direction (forward) with respect to the laboratory frame. Therefore the instrumentation in the forward and backward region differs.

The H1 detector was designed to study high-energy interactions between electrons and protons provided by the HERA collider. Many different types of ep-interactions will be discussed in section 2.1.4 where they are introduced as physics classes which should pass the trigger system. The data transfer rate from the detector readout system to mass storage manages around 10 events per second ($\approx$ 1.1 MByte/s). This bandwidth limit means that by far not all interactions can be written to permanent storage. With an initial rate of about 10 MHz (96ns bunch crossing time) the rate must be reduced by a factor one million.

The largest contribution to the initial event rate are background events which result from non ep-interactions. They consist of beam-gas or beam-wall interactions (protons collide with nuclei of the residual gas, with the beam pipe or with detector components), synchrotron radiation and cosmic radiation. These background events should be filtered out to be able to record real physics events. A much smaller contribution to the initial

[1] The electrons/positrons are naturally polarised in longitudinal direction due to synchrotron radiation. The spin rotators rotate the spin of the electron/positron from its natural orientation to obtain transversely polarised leptons.

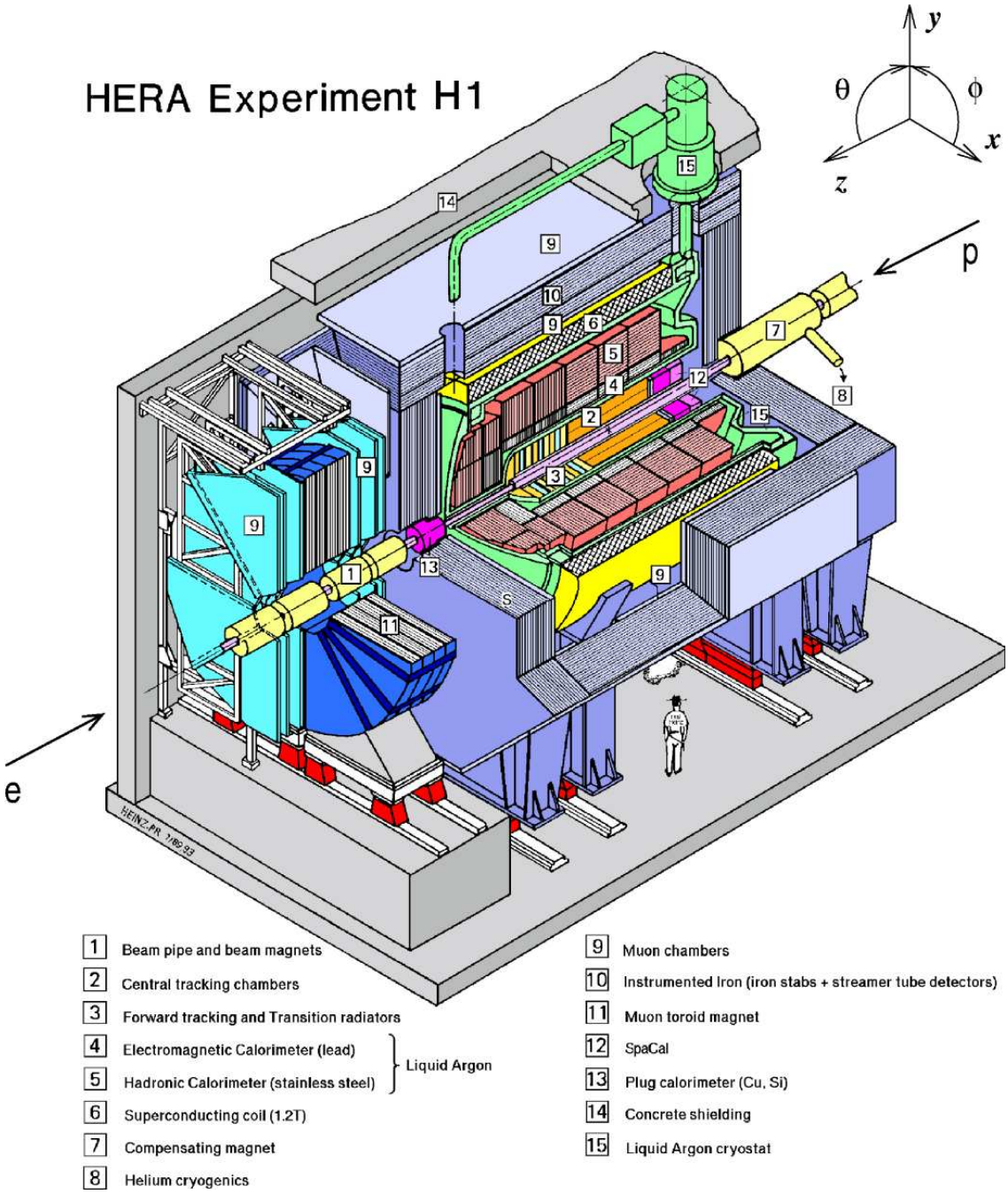

Figure 2.3: The H1 experiment with a list of detector components.

event rate are genuine *ep*-interactions. The larger part of them is photoproduction (cross section of several μb), and events which one is really interested in (high Q^2) are again a minor contribution. The multi level trigger system which copes with the rate reduction problem will be described in the following section.

Some of the detector components will now be discussed in more detail as they are part of the trigger system:

- The main components of the **central tracking chambers** are two central jet chambers (drift-chambers) [3] which provide a good resolution in the $r\phi(xy)$ plane and the multi wire proportional chambers [4] which provide a good resolution in z-direction. The tracking chambers are used to determine the interaction vertex, the momenta and types of produced charged particles.

- The **liquid argon calorimeter** (LAr) [5] surrounds the central tracking system and is divided into an inner electromagnetic and an outer hadronic part. In both parts liquid argon is the active medium, the absorber material is lead in the electromagnetic part and steel in the hadronic part. The calorimeter has a very fine granularity, the coarsest segmentation has three barrels: inner forward (IF, positive z), forward (FB) and central (CB). The calorimeters are used to measure electromagnetic and hadronic energies and to identify the particles which generated the measured energies.

- The **spaghetti calorimeter** (SpaCal) [6] is located in the backward (negative z) region. It is also separated into an electromagnetic and a hadronic part, both consist of grooved lead plates as absorbers with grooves containing scintillating (spaghetti like) fibres which lead the scintillation light to photomultipliers.

- The **central muon detector** forms together with the forward muon detector the muon system. It is part of the instrumented iron yoke which returns the magnetic flux of the main solenoidal coil. The central muon detector is used to detect muons with an energy of at least $1.2GeV$.

2.1.3 Trigger System

A multi level trigger system copes with the rate reduction task by selecting interesting physics events on the one hand and rejecting obvious background on the other hand. Figure 2.4 shows the four trigger levels implemented in the H1 experiment. The trigger system performs a stepwise reduction of the event rate by increasing the level of analysis from one trigger level to the next. It starts with the very fast but also simple first trigger level which selects events if they match one of the predefined conditions to certain detector components (simple AND/OR of threshold conditions for energies or counters) and ends with the very slow but also very sophisticated fourth trigger level on which a full event reconstruction is done. Each trigger level consists of many *sub-triggers* which are dedicated to one or a few special kinds of *ep*-interactions. An event is passed on to the next trigger level if one of the sub-triggers of the previous level "fires".

L1: First Trigger Level

Most detector components provide fast trigger information for each bunch-crossing (every $96ns$). Pipelines store this information to provide a dead-time free level 1 trigger. The

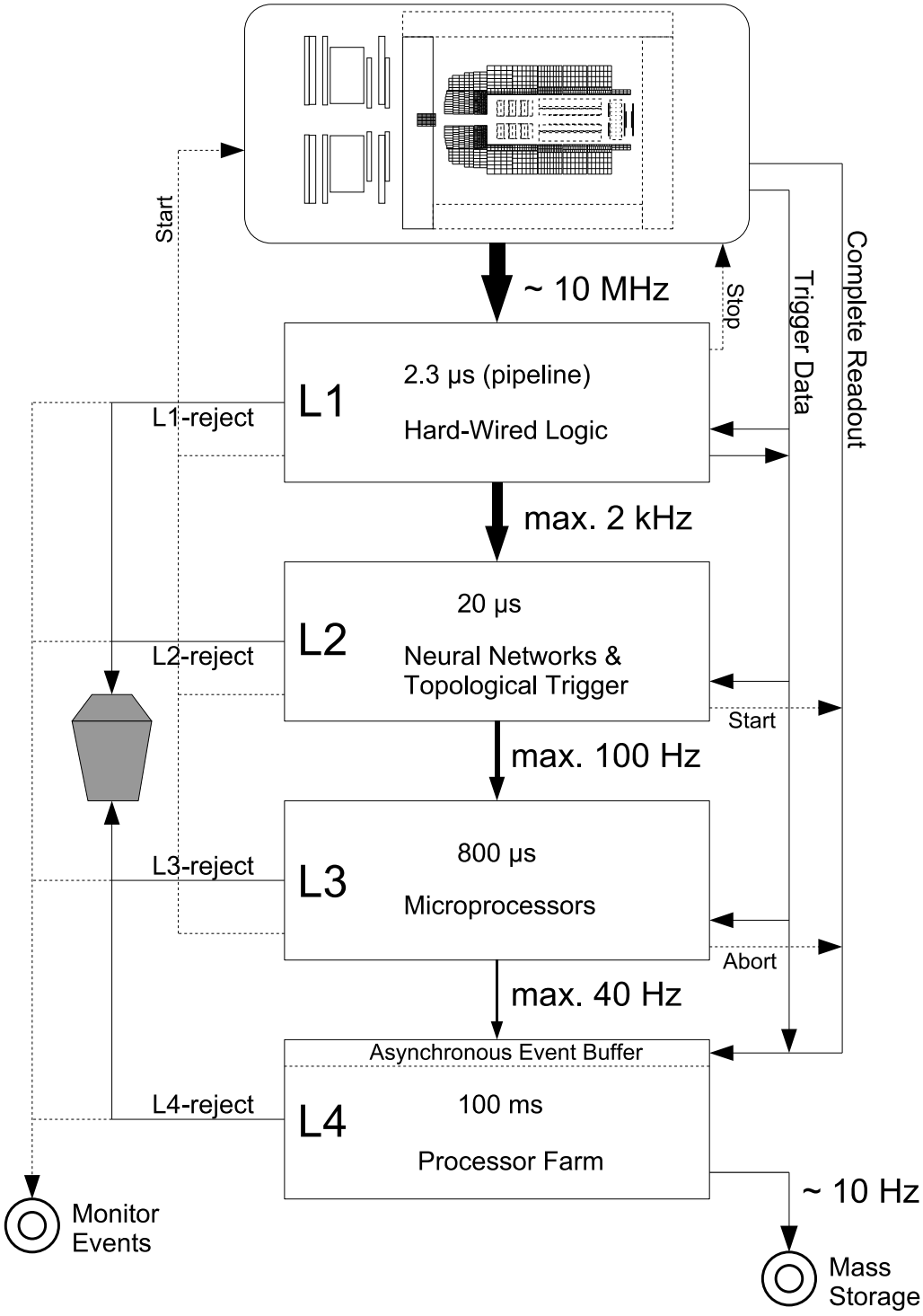

Figure 2.4: The H1 Trigger System: A very important bottleneck is given by the bandwidth with which the complete readout can be transferred to level 4, hence the maximum input rate to this trigger level cannot be increased by using more processors in the farm. Trigger level 3 was not used up to end of 2004.

level 1 decision is derived with a latency of about 2.3 μs. This decision is the logical OR of 128 pre-scaled level 1 sub-triggers each of which is a logical combination of simple queries to the trigger information of any detector component (threshold conditions to energies or counters).

L2: Second Trigger Level

Trigger level 2 has 20 μs to derive its decision. At this level all level 1 information is available, thus correlations between the level 1 trigger information from different detector components can be used to judge the event which was triggered by level 1 in more detail. Two independent systems are used to validate specific level 1 sub-triggers: The neural network trigger (L2NN) and the topological trigger (L2TT). The latter uses matrix operations and logical combinations to come to a conclusion while the neural network trigger evaluates mostly feed forward neural networks as discussed below in detail. If at least one level 1 sub-trigger is validated successfully or a level 1 sub-trigger fired which does not need a validation on level 2 the event is read out and sent to trigger level 4 (trigger level 3 is not used).

L4: Fourth Trigger Level

About 30 parallel processors take over the complete detector read-out for any event that reaches the fourth trigger level. On trigger level 4 the full detector information is available, a full event reconstruction can thus be done and the previous trigger levels are verified. A set of *finders* is then applied to the reconstructed data. Each finder is dedicated to a specific physics class and the event is kept if one of the finders is successful. The fourth trigger level is thus the first software-based level while the first and second level are hardware-based. With a rate of about 10 Hz events are logged to mass storage.

A simple way to limit the trigger rate especially of level 1 sub-triggers with a high rate is pre-scaling. A sub-trigger can be pre-scaled which means that only every nth trigger is processed, all others are ignored. This is the simplest way to reduce a too high trigger rate by a certain factor (from pre-scale 2: factor 0.5 to arbitrarily high pre-scales). Of course it is important to note that pre-scaling throws away physics and background equally – from the trigger point of view the selection is done randomly.

To keep track of the trigger behaviour and to monitor the decisions specific trigger levels can be switched off for some time or can be by-passed for a certain fraction of events:

- Under normal conditions each trigger level is active. For trigger studies **transparent runs** can be taken for which a specific trigger level is ignored. For a level 2/4 transparent run for example all events triggered by the first trigger level are written to tape with a high dead time ignoring the other trigger levels.

- A small percentage of events which would be rejected by trigger level 2 or 4 are kept nevertheless for monitoring purposes – they should allow to watch and control the trigger behaviour. These events are called **override** or **by-pass** events.

The neural network trigger on the second trigger level will now be discussed in detail since the analysis presented in section 7.2 will be based on the development of new neural network triggers in the HERA II period.

2.1.4 The Level 2 Neural Network Trigger

The second trigger level offers the possibility to combine pieces of information from different sub-detectors and evaluate their correlations to get a clearer picture about the actual event. The neural network trigger hosts up to 13 feed forward neural networks which take the values from various sub-detectors as inputs and either deliver or refuse a confirmation of the L1 decision. A neural network is a particular example for a statistical learning method and will be discussed together with other methods in chapter 5 (section 5.4.2). In appendix A we will shortly discuss the hardware which is used to fetch the inputs from the various sub-detectors, to calculate the output of the neural networks and to send it back within only 20 μs.

Especially after the luminosity upgrade, the trigger system as a whole and the level 2 neural network trigger in particular become even more important. The primary rate is increased but there is no possibility to also increase the logging rate (bandwidth limit of the readout hardware). The rate thus has to be reduced within the trigger system more than before the upgrade.

The neural network trigger is perfectly suitable for this task because it is designed to reject the background triggered by those level 1 sub-triggers which have a very high rate. It is important to note that this background rejection cannot be done for example by increasing the number of processors on level 4 because the input rate to level 4 is technically restricted to about 40 Hz. Therefore the rate reduction must be done very early in the trigger system.

The quantities which are available for the level 2 trigger are generated by the level 1 triggers of each subsystem:

- For the calorimeters fast sampled energies (LAr and SpaCal) or counted cells above thresholds (SpaCal) are available. The energies are summed over the electromagnetic and hadronic part and come in four quadrants in ϕ. The liquid argon calorimeter is in addition divided into three regions (central, forward and inner forward).

- The drift chamber $r\phi$-trigger (DC$r\phi$) applies masks to 10 of 56 wire layers and finds thereby track candidates which are grouped to positive and negative and to high and low energy tracks according to their curvature.

- The z-vertex trigger uses the hits in the inner and outer z-chambers (CIP and COP, part of the multi-wire proportional chambers). All combinations of hits in the same ϕ-bin (16 bins in total) are projected back onto the z-axis and fill a histogram of 16 bins covering the region between $-44cm$ and $+44cm$ for the projected z-positions. The track-candidates (called "big-rays") are also counted and grouped according to their z-direction (forward, forward central, backward central, backward).

- The central muon trigger counts in four z-regions (forward end-cap, forward, backward, backward end-cap) how many of the 16 modules have been hit (by muons which passed the liquid argon calorimeter or by cosmic muons).

Table 2.1 shows an overview of the most important quantities available for the level 2 neural network trigger which are built by the level 1 triggers as described above.

The neural network trigger successfully reduces the rate of many different level 1 triggers since its installation in 1996. The analysis presented in section 7.2 will discuss neural

Liquid Argon Calorimeter (electromagnetic and hadronic summed)	
`larife`	Total energy in the inner forward region
`eifq0`	Energy in quadrant 0 of the inner forward region
`eifq1`	Energy in quadrant 1 of the inner forward region
`eifq2`	Energy in quadrant 2 of the inner forward region
`eifq3`	Energy in quadrant 3 of the inner forward region
`larfbe`	Total energy in the forward region
`efbq0`	Energy in quadrant 0 of the forward region
`efbq1`	Energy in quadrant 1 of the forward region
`efbq2`	Energy in quadrant 2 of the forward region
`efbq3`	Energy in quadrant 3 of the forward region
`larcbe`	Total energy in the central region
`ecbq0`	Energy in quadrant 0 of the central region
`ecbq1`	Energy in quadrant 1 of the central region
`ecbq2`	Energy in quadrant 2 of the central region
`ecbq3`	Energy in quadrant 3 of the central region
Spaghetti Calorimeter (electromagnetic and hadronic summed)	
`spcent1`	Number of trigger cells in the central area above a certain threshold (0.5 GeV before 16.03.04, 9 GeV afterwards)
`spcent3`	Number of trigger cells in the central area above a certain threshold (6 GeV, stayed the same)
`espq0`	Energy in quadrant 0 (operable since 05.06.04)
`espq1`	Energy in quadrant 1 (operable since 05.06.04)
`espq2`	Energy in quadrant 2 (operable since 05.06.04)
`espq3`	Energy in quadrant 3 (operable since 05.06.04)
Drift Chambers	
`trtot`	Total number of track candidates
`trlopos`	Number of negative track candidates with low momentum
`trhipos`	Number of positive track candidates with high momentum
`trloneg`	Number of negative track candidates with low momentum
`trhineg`	Number of positive track candidates with high momentum
Multi Wire Proportional Chambers	
`cpvmax`	Maximum number of entries in the z-vertex histogram
`cpvsum`	Total number of entries in the z-vertex histogram
`cpvpos`	Bin with the maximum number of entries in the z-vertex histogram
`nbigray`	Total number of track candidates
`nbigfwd`	Number of track candidates in the forward region
`nbigfce`	Number of track candidates in the forward-central region
`nbigbce`	Number of track candidates in the backward-central region
`nbigbwd`	Number of track candidates in the backward region
Central Muon Detector	
`irontot`	Total number of muons
`ironfe`	Number of muons in the forward end-cap region
`ironfb`	Number of muons in the forward region
`ironbb`	Number of muons in the backward region
`ironbe`	Number of muons in the backward end-cap region

Table 2.1: Most important quantities available for the level 2 neural network trigger: all these quantities are calculated and provided by the level 1 trigger of the respective subdetector.

network triggers developed in 2004 which cope with the task of a high background rejection combined with an equally high efficiency for the interesting physics. In the following we will dicuss those physics channels for which new networks have been developed since the beginning of the HERA II period.

2.1.5 Important Physics Channels for L2NN

In the following sections different kinds of *ep*-interaction will be discussed which are triggered by L1 sub-triggers that need a rate reduction. These physics channels form thus the "signal" which should pass the neural network that validates the respective level 1 sub-trigger. The underlying physics ranges from diffractive events to the production of heavy quarks.

Deeply Virtual Compton Scattering

Deeply virtual Compton scattering (DVCS) is defined as the elastic scattering of a virtual photon off a proton with a real photon in the final state. In the case of H1 photons with virtualities $Q^2 >$ few GeV^2 are emitted by the incoming electron/positron leading to the reaction $ep \rightarrow e\gamma p$. Figure 2.5 shows perturbative approximations: In Quantum Electro Dynamics (QED) the virtual photon scatters off a quark from the proton which is reabsorbed in the proton after emission of the real photon. In Quantum Chromo Dynamics (QCD) the virtual photon interacts via a quark loop with two gluons from the proton. The quark loop also emits the real photon. At HERA energies the QCD graph dominates and thus provides the possibility to study the gluonic structure of the proton.

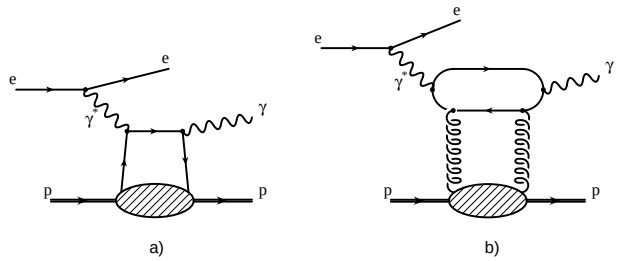

Figure 2.5: Perturbative approximations of deeply virtual Compton scattering: (a) QED and (b) QCD.

DVCS is triggered by the level 1 sub-trigger 41 which requires a significant energy deposition (the electron) in the SpaCal within a certain time window. Figure 2.6 top shows a DVCS event triggered by this condition, the energy deposition in the SpaCal is clearly visible, in addition the energy deposition of the photon in the LAr calorimeter can be seen. On the bottom we see also an energy deposition in SpaCal, this event was also triggered by the level 1 sub-trigger 41. But this event is an upstream beam-gas interaction.

The analysis in section 7.2.3 will investigate the possibility to reduce the rate of level 1 sub-trigger 41 while maintaining a high efficiency for DVCS. The rate of this sub-trigger reached 8-9 Hz, much too high compared to the overall (from all L1 sub-triggers) maximum input rate to level 4 of 40 Hz.

Charged Current Interactions

Figure 2.7 shows an electron-proton interaction in which either a neutral boson (γ or Z^0) or a charged boson ($W^{\pm}$) is exchanged. The interactions are named neutral and

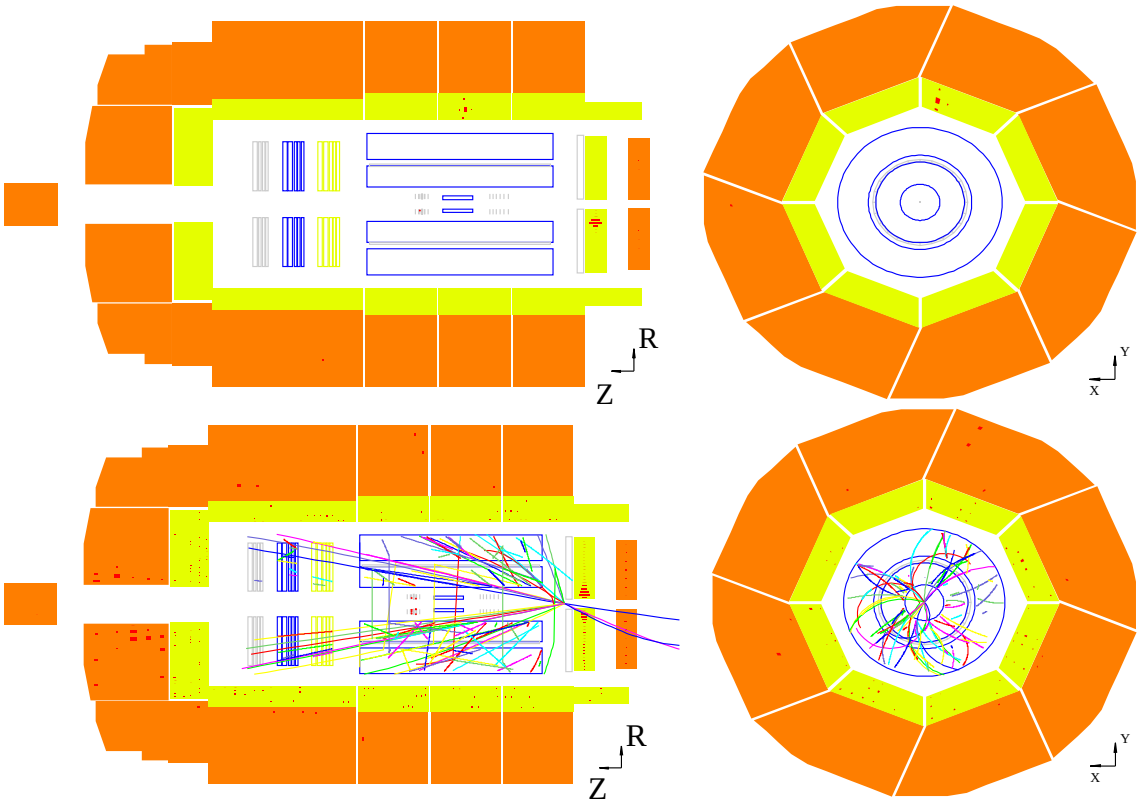

Figure 2.6: Detector view of deeply virtual Compton scattering (top) and competing background, here an upstream beam-gas interaction (bottom).

charged current interactions, respectively. In the case of an exchanged $W^{\pm}$ boson the positron/electron converts into an (anti)neutrino which leaves the detector unobserved. This neutrino carries some part of the transversal energy which leads to an unbalanced observed energy in the detector. Charged current interactions can therefore be triggered by requiring a missing transverse energy in the LAr calorimeter.

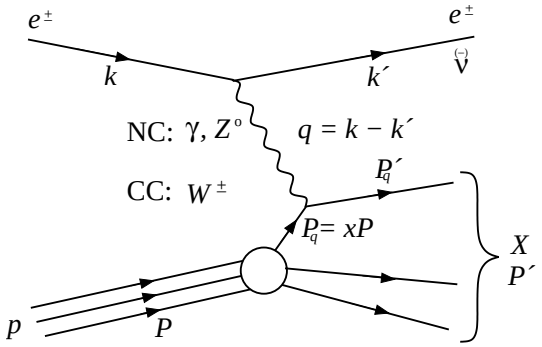

Figure 2.7: Neutral current vs. charged current interactions: important kinematic variables are the momentum transfer $Q^2 = -q^2$ and the fraction of the proton momentum carried by the struck quark x.

Charged current interactions (CC) are very rare compared to neutral current interactions (NC) as can be seen from the cross sections:

$$\frac{d^2\sigma_{NC}^{e\pm}}{dxdQ^2} = \frac{e^4}{8\pi x}\left[\frac{1}{Q^2}\right]^2 \phi_{NC}^{\pm}(x, Q^2) \tag{2.1}$$

where the neutral current "structure function term" $\phi_{NC}^{\pm}$ is a linear combination of the generalised structure functions $\tilde{F}_L$, $\tilde{F}_2$ and $x\tilde{F}_3$.

$$\frac{d^2\sigma_{CC}^{e\pm}}{dxdQ^2} = \frac{g^4}{64\pi x}\left[\frac{1}{Q^2 + M_W^2}\right]^2 \phi_{CC}^{\pm}(x, Q^2) \tag{2.2}$$

where $\phi_{CC}^{\pm}$ consists of the structure functions for CC interactions $\tilde{W}_L$, $\tilde{W}_2$ and $x\tilde{W}_3$ in analogy to NC. The Q^2 dependence of the cross sections mainly results from the propagator terms which is $1/Q^4$ for NC and $1/(Q^2 + M_W^2)^2$ for CC. While the NC cross section decreases very rapidly with increasing Q^2, the CC cross section falls much less steeply until $Q^2 \approx M_W^2$. At the electroweak unification scale ($Q^2 \approx M_Z^2 \approx M_W^2$) the cross sections are similar: $\phi_{NC}^{\pm} \approx \phi_{CC}^{\pm}$. From this point of view the neutral current cross section rises much quicker with decreasing Q^2.

Therefore a very high efficiency of a trigger for charged current interactions is mandatory. The level 1 sub-trigger 77 which needs a validation by a level 2 neural network is based on missing transverse energy in the liquid argon calorimeter. For high transverse energy triggering is no problem but the threshold for sub-trigger 77 is set to a medium level to be sensitive also to the low p_T part of the cross section. This results in a trigger rate which climbs up to the order of 10 Hz for high currents. The rate reduction which is needed in the analysis in section 7.2.4 is only about a factor two (50%) while the main target is a near 100% efficiency.

Figure 2.8 shows how a charged current event and competing background (a cosmic event) look like in the detector. In both cases a missing transverse energy is detected, therefore both types are triggered by level 1 sub-trigger 77.

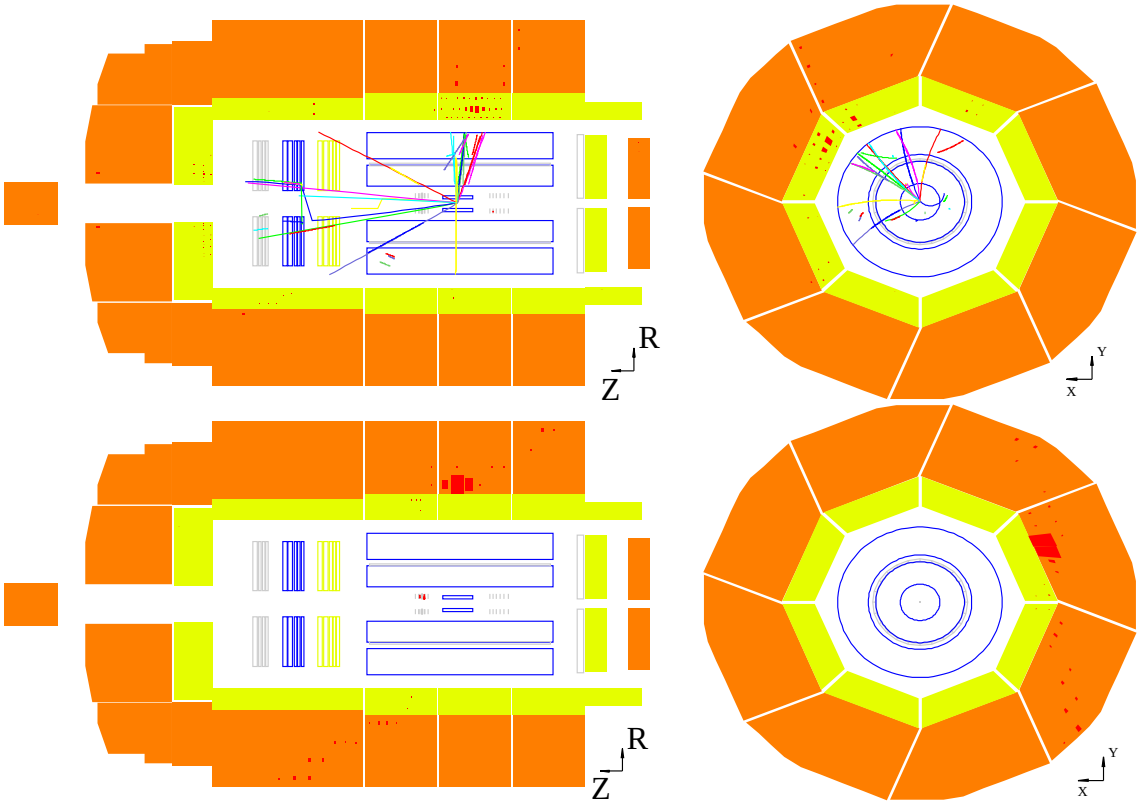

Figure 2.8: Detector view of a charged current interaction (top) and competing background, here an event with a cosmic muon passing the detector from top to bottom (bottom).

J/ψ Production

The electron/positron beam of HERA can be viewed as a source of virtual photons like for DVCS. Figure 2.9 shows that a J/ψ is formed by a virtual photon, e.g. according to the vector meson dominance model, in which the photon is viewed as a composition of an electromagnetic field and hadronic parts, both with the same quantum numbers $\mathcal{J}^{\mathcal{PC}} = 1^{--}$. Hadronic particles with these quantum numbers are vector mesons of which a heavy one is the J/ψ. The J/ψ scatters off the proton and can be detected via its subsequent decay, for example into an e^+e^- or $\mu^+\mu^-$ pair.

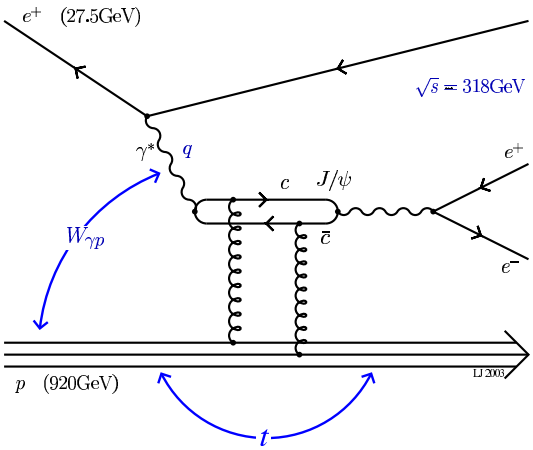

Figure 2.9: J/ψ production and subsequent decay into electron and positron, important kinematic variables are the energy in the photon-proton centre-of-mass system $W_{\gamma p}$ and the four-momentum transfer squared at the proton vertex t.

Depending on the decay type and on the kinematic region different level 1 sub-triggers are designed to trigger the production of J/ψ's. Two level 1 sub-triggers which have a high rate and thus need background suppression on the second trigger level will be discussed in the analysis. Sub-trigger 15 (section 7.2.6) triggers inelastic photoproduction of J/ψ's and their decay into muons. Sub-trigger 33 (section 7.2.5) triggers exclusive photoproduction of J/ψ's and their decay into electrons where one electron leaves a track in the tracking system and ends up in the LAr calorimeter, the other electron produces an energy cluster in the SpaCal (Track-Cluster configuration).

D^* and Dijet Production

A last example of a physics channel, whose level 1 trigger needs background reduction, is the production of heavy quarks, in particular D^*'s, and dijet production. Both of them are triggered by level 1 sub-trigger 83 (analysis in section 7.2.7) which requires activity in the central trackers in combination with the tagged beam electron. Figure 2.10 shows the production mechanism for heavy quarks. D^{*+}'s consist of an charm and anti-down quark, D^{*-}'s thus consist of an anti-charm and down quark.

Like the J/ψ production also this kind of ep-interaction provides direct access to the gluon structure. Whereas the J/ψ production delivers an experimentally very clean signal, the heavy quark production is experimentally difficult because of the fragmentation process. On the other hand the process shown in figure 2.10 is theoretically well described by known QCD matrix elements whereas the bound J/ψ state is difficult to describe theoretically.

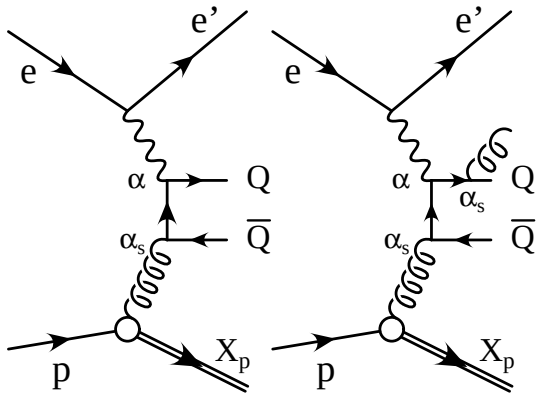

Figure 2.10: Heavy quark production by direct photon-gluon-fusion, left in leading order and right in next to leading order.

2.1.6 Offline Analysis: Instantons

Instanton-induced processes [7] in *ep*-collisions are a good example for a (possible) small signal above a huge background. QCD predicts a rare class of events with special kinematic properties (compare the "fireball-like" structure shown in figure 2.11). Instantons are characterised by the production of a large number of charged particles and an enhanced fraction of strange hadrons. The resulting energy depositions in the calorimeter with the characteristic instanton band is shown in figure 2.12.

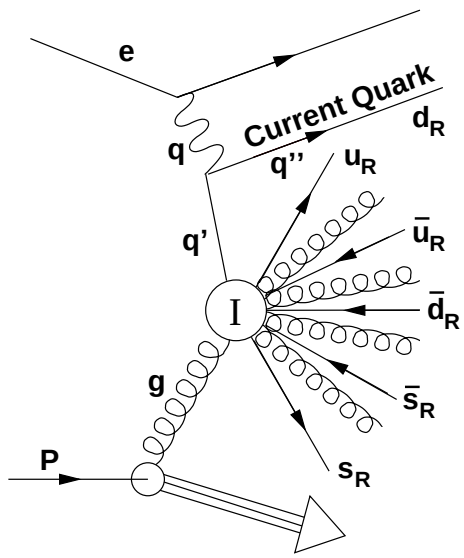

Figure 2.11: Feynman-like diagram of an instanton-induced process in deep inelastic scattering.

Since this kind of search for a rare event type is done offline we have the possibility to choose variables as inputs which describe the instanton-processes best or, to be more precise, which describe the differences between the instanton-like events and the competing QCD background in the best possible way[2]. An even more important requirement is often given by the usage of Monte-Carlo simulations. Only those quantities should be used as inputs for a training with simulated data whose distributions match in simulated and experimental data. Otherwise the derived results may not be valid.

In this analysis (which is a follow-on of an earlier analysis [8]) Monte-Carlo simulations are used for the training of the classifier which should separate instanton-induced

[2]The free choice of inputs is one of the major differences between online and offline applications. The preprocessing (and thus the generation of well-suited inputs) has to be kept at a minimum for online applications due to the tight time restrictions.

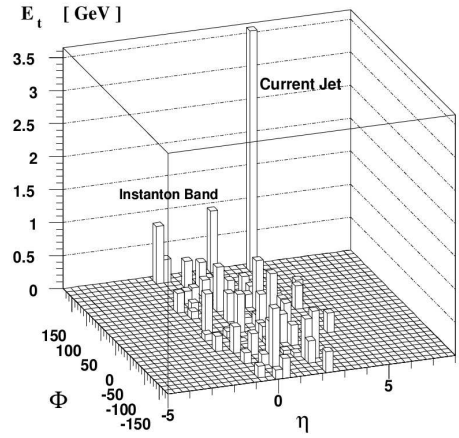

Figure 2.12: Energy depositions in the LAr calorimeter for an instanton-induced event with the characteristic instanton band [8].

events from the competing background. The signal (simulated instanton-induced events) is generated by "QCDINS" [9] and for the background two different simulations for the standard perturbative QCD event classes exist: "MEPS" (Matrix Element plus Parton Shower, calculated with RAPGAP [10]) and "CDM" (Colour Dipole Model, calculated with ARIADNE [11]). The perturbative simulations were checked to reproduce the experimental data and three kinematic quantities were chosen as basic inputs: band sphericity sph_B, band multiplicity n_B, and the virtuality ${Q'_{rec}}^2$. These three inputs can be used to distinguish signal from background, are reproduced well and are thus first choice as inputs for the classifier. Two additional quantities, the transverse energy of the current jet $E_{T,Jet}$ and the transverse energy of the band $E_{T,B}$ can also be used to distinguish signal from background, but they show slightly different behaviour in the simulation with respect to the experiment.

For the further analysis we regard this selection of inputs as given. We now try to find a method which accepts instanton-like events but rejects standard perturbative QCD event classes which look similar. The target of this analysis is to apply this classifying method to the experimental data and compare the number of kept events with the number of kept events in the perturbative background (and with the number of kept instanton-induced events). The instanton hypothesis would be confirmed if the number of kept events in the experimental data shows an excess over the kept perturbative QCD events from the simulation.

A very good background reduction is needed to be able to compare these numbers as the instanton-induced events are expected to be several hundred among over hundred thousand of perturbative events. Statistical uncertainties alone make it impossible to compare these numbers without a prior background reduction leaving systematic uncertainties unconsidered. A high background rejection combined with an equally high efficiency for instanton-like events with the help of statistical learning methods is the main target of the analysis presented in section 7.3.

2.2 Higgs Boson Parity Measurement at a Future Linear Collider

The origin of particle masses is one of the major scientific questions of our time. The experimental verification of the Higgs mechanism as proposed by the Standard Model [12] is therefore an important goal. But not only a discovery is aspired, the determination of all the properties of the Higgs mechanism is among the central tasks for future linear colliders. The Standard Model Higgs boson H is a scalar ($\mathcal{J}^{\mathcal{PC}} = 0^{++}$) while extensions like the Minimal Super-symmetric Standard Model predict in addition a pseudo-scalar partner A ($\mathcal{J}^{\mathcal{PC}} = 0^{-+}$). It is therefore important to be able to distinguish between these two cases.

The analysis on which the following discussion is based [13] uses a future experimental setup which is based on the TESLA proposal [14]. A linear e^+e^- collider with a centre-of-mass energy of 500 GeV is assumed with an accumulated luminosity of 500 fb^{-1}. The Higgsstrahlung production process shown in figure 2.13 left and a Higgs boson mass of $120GeV$ are used in the simulation. The assumptions about the detector lead to certain resolution properties which are respected in the Monte Carlo simulations. Gaussian spreads are, for example, applied to the generated energies as well as angular spreads to the angles. The detector effects have been approximately verified by a full detector simulation [15]. A study of background which competes with the decay channel presented below is planned but not used in the analysis as presented here.

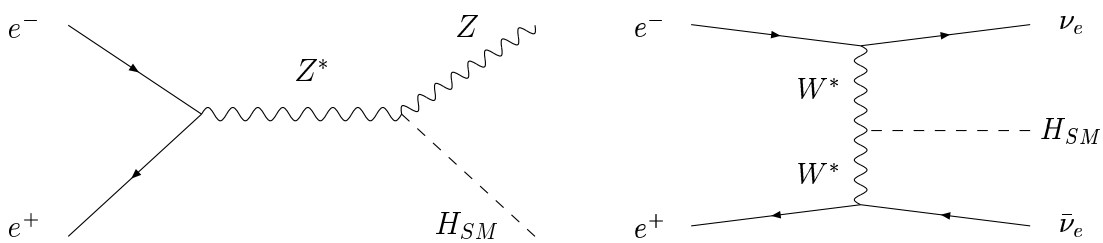

Figure 2.13: Main Higgs boson production mechanisms in an e^+e^- collider: Left Higgsstrahlung and right W^+W^- fusion.

The strategy to recognise which type of Higgs boson was found is to study the decay products. One of the most promising decay channels which allows equal sensitivity to the $\mathcal{CP}$-even and $\mathcal{CP}$-odd components is $H \to \tau^+\tau^-$. Although the decay to $b\bar{b}$ has a much higher branching ratio, the extraction of information about the b polarisation state is very difficult due to depolarisation effects in the fragmentation process. This makes the $\tau^+\tau^-$ decay mode important from the spin correlations point of view.

The propagation of the transversal spin correlations in the subsequent decays $\tau^\pm \to \rho^\pm\bar{\nu}_\tau(\nu_\tau)$ and $\rho^\pm \to \pi^\pm\pi^0$ leads to a characteristic angular distribution which depends on the parity of the Higgs boson. Figure 2.14 shows the distribution of the acoplanarity angle ϕ^* between the plane spanned by $\tau^+\rho^+$ and the plane spanned by $\tau^-\rho^-$ in the Higgs boson rest frame for a scalar and a pseudo-scalar Higgs boson.

Figure 2.15 tries to give an intuitive interpretation why this acoplanarity angle distribution depends on the Higgs boson parity. Since the parity is given by

$$P = (-1)^l P_{\tau^+} P_{\tau^-} = (-1)^{l+1} \tag{2.3}$$

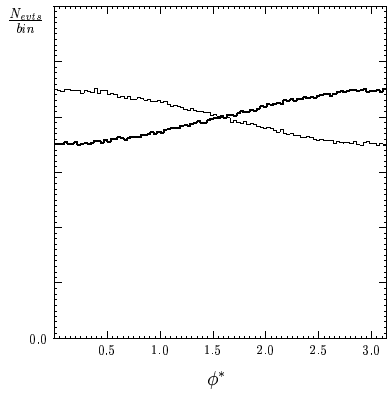

Figure 2.14: Acoplanarity ϕ^* distributions in the Higgs boson rest frame (defined as the angle between the plane spanned by $\tau^+\rho^+$ and the plane spanned by $\tau^-\rho^-$). The thick line denotes the case of a scalar Higgs boson, the thin line the pseudo-scalar one. Taken from [13].

the τ-pair is in a p-wave ($l = 1$) for a scalar Higgs boson H and in an s-wave ($l = 0$) for a pseudo-scalar Higgs boson A. The transversal spin components (in the respective $\tau^\pm$ rest frames) induce a favourite decay direction for the $\rho^\pm$ because the weak decay of $\tau^\pm$ can only result in left-handed neutrinos and right-handed anti-neutrinos. A favourite configuration of back to back ρ's for a scalar Higgs boson H results in the cosinus-distribution shown in figure 2.14 with the maximum at 180 degrees. For a pseudo-scalar Higgs boson A the favourite configuration is parallel with a most probable ϕ^* of 0 degrees.

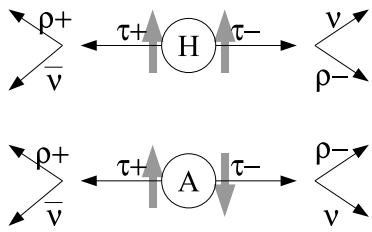

Figure 2.15: Propagation of spin correlations from the transversal $\tau^\pm$ spins to the preference of decay directions of the $\rho^\pm$.

Since the ϕ^* distributions are not directly observable a different angle is defined which can be reconstructed in a similar way. The observable which promises the best distinction between the two cases is defined in the rest frame of the $\rho^+\rho^-$ system [16]: The acoplanarity angle φ^* is the angle between the two planes spanned by the immediate decay products (the $\pi^\pm$ and π^0) of the two ρ's as shown in figure 2.16.

The distribution of φ^* does not differ directly depending on a scalar or pseudo-scalar Higgs boson. But it does depend on the parity after dividing events according to $y_1 y_2 > 0$ and $y_1 y_2 < 0$ where

$$y_1 = \frac{E_{\pi^+} - E_{\pi^0}}{E_{\pi^+} + E_{\pi^0}} \; ; \qquad y_2 = \frac{E_{\pi^-} - E_{\pi^0}}{E_{\pi^-} + E_{\pi^0}}. \tag{2.4}$$

and $E_{\pi^\pm}$ and E_{π^0} are the $\pi^\pm, \pi^0$ energies in the respective (replacement) $\tau^\pm$ rest frames (for a full description of the observables see [16]). Figure 2.17 shows the distributions of φ^* for $y_1 y_2 > 0$ left and right for $y_1 y_2 < 0$. In a simple interpretation y_1 and y_2 determine how the $\rho^\pm$ energies are distributed among the charged ($\pi^\pm$) and neutral (π^0) products. These distributions determine how the angle ϕ^* defined above is propagated. A product $y_1 y_2 \approx 0$ means that the spin correlation gets lost due to almost equal energies for the products of at least one decay. The spin correlation is on the other hand preserved if one decay product gets almost the full $\rho^\pm$ energy in both decays. In this case $y_1 y_2$ is either smaller or greater than zero and the φ^*-distributions differ. The sign of $y_1 y_2$ is then only used to distinguish between the two cases shown in figure 2.17 which result from the definition of the angle φ^*

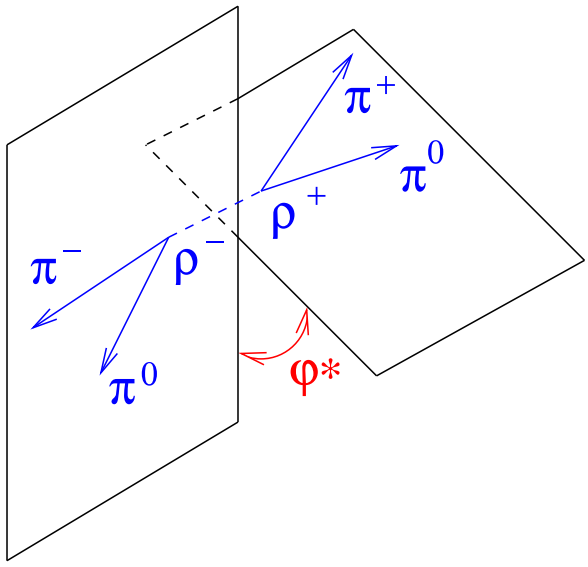

Figure 2.16: Definition of acoplanarity φ^* as the angle between the two planes spanned by the immediate decay products (the $\pi^\pm$ and π^0) of the two ρ's in the $\rho^+\rho^-$ rest frame.

in figure 2.16: The orientation of each normal vector depends on the energy distribution of the two π's and thus on y_1/y_2. In summary, $sign(y_1 y_2)$ determines, whether a phase-shift of 180° has to be applied to the acoplanarity φ^* before the respective event is filled into the histogram.

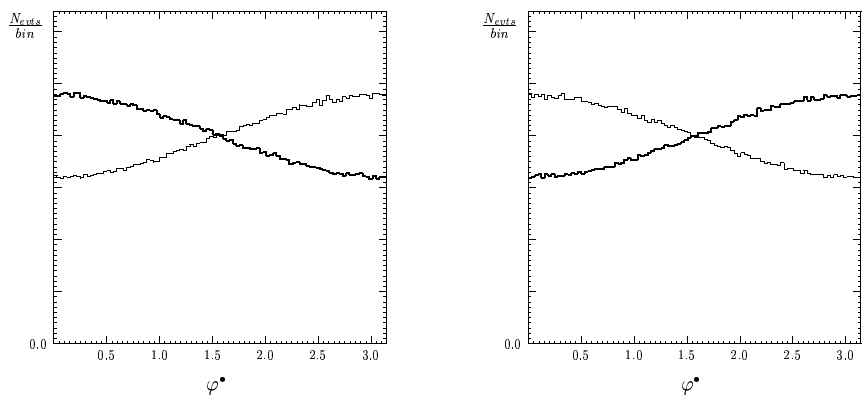

Figure 2.17: Acoplanarity φ^* distributions in the $\rho^+\rho^-$ rest frame. Left events with $y_1 y_2 > 0$ are plotted for a scalar (thick) vs. pseudo-scalar (thin) Higgs boson, right events with $y_1 y_2 < 0$ are plotted in the same way. Taken from [13].

There are some additional ideas which allow to reconstruct y_1 and y_2 with better precision or which select special event types to intensify the significances shown in figure 2.17. They will be explained shortly in section 7.4.

Two competing targets control the significance which allows to decide whether a scalar or a pseudo-scalar Higgs boson is observed: On the one hand as many events as possible are needed in the analysis since the statistics given by the luminosity and cross section of the chosen decay channel is very limited. On the other hand a strong selection on those events that show a clear signal, i.e. a favour for only one of the two parity states, may enhance the significance. The determination of the observed Higgs boson parity at a maximum possible level of significance with the help of statistical learning methods is the main target of the analysis presented in section 7.4.

2.3 Small-Angle Neutron Scattering Detector

Small-angle neutron scattering is an essential tool for the investigation of structural and dynamical properties of condensed matter. Almost any fluctuation in composition, density and magnetisation on a scale between 5 and 5000 Å can be detected by this method [17]. Measurements are usually carried out with large area position sensitive detectors. This allows an efficient exploitation of the expensive beam-time by a simultaneous measurement of the total scattering regime.

The small-angle neutron scattering instrument KWS-1 [18] at the research reactor FRJ-2 at the Forschungszentrum Jülich uses the principle of the Anger camera [19] to detect neutrons by collecting scintillation light in a two-dimensional array of photomultipliers. Figure 2.18 illustrates this principle: A ^{6}Li-glass scintillator doped with Ce as an activator leads via the nuclear reaction

$$n + {}^6Li \rightarrow {}^3H + \alpha + 4.78MeV \tag{2.5}$$

to the production of light which is dispersed on an array of photomultiplier tubes.

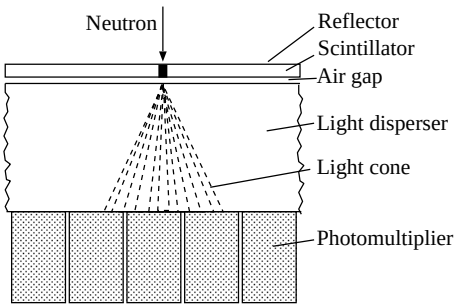

Figure 2.18: Cross sectional view of the detector showing the light distribution of a neutron event.

The photomultiplier tubes are arranged in a two-dimensional grid and capture a large part of the produced scintillation light. Figure 2.19 illustrates that the produced light cone illuminates typically 3×3 photomultipliers. The individual photomultiplier signals undergo an amplification, shaping and digitisation step and can then be used to derive the exact incident position of the neutron as the reconstructed centre of the light cone.

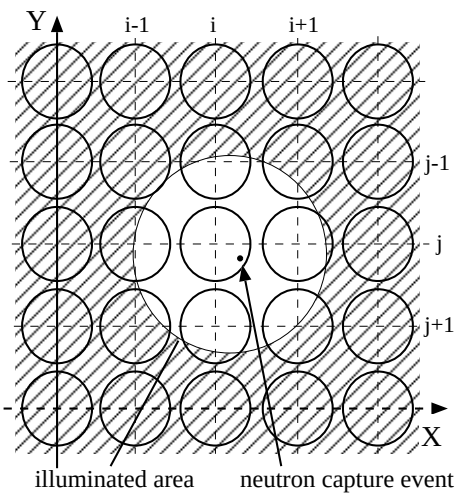

Figure 2.19: Projection of a light cone from a neutron capture event on the plane of the photomultiplier cathodes.

The active area of 60×60 cm^2 is formed by 4×4 glass-scintillator plates behind which 8×8 photomultipliers are placed (see figure 2.20). For the reconstruction of the neutron

incident position the total scintillation surface is divided into 128 × 128 channels. Any reconstructed position is given in these channel coordinates. The resolution given by this binning is sufficient and probably better than the precision which can be achieved by any reconstruction method.

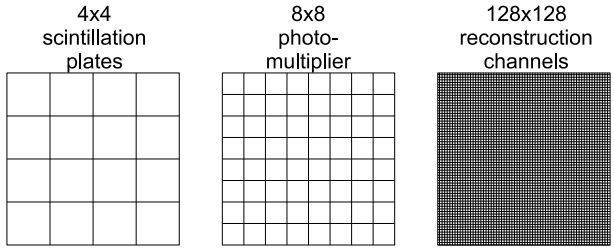

Figure 2.20: Structure of the neutron detector: The detector surface is physically divided by the scintillation plates and the photomultipliers. It is divided into a fine binning for the reconstruction of the neutron incident position.

The currently implemented position reconstruction [20] uses the quotient q_i to describe the light asymmetry around the central photomultiplier signal S_i:

$$q_i = \frac{S_{i+1} - S_{i-1}}{S_{i+1} + S_i + S_{i-1}}. \tag{2.6}$$

S_{i-1} and S_{i+1} are the photomultiplier signals of the neighbouring columns or rows (depending on whether the x or y coordinate is estimated). The shape of q_i, depending on the position coordinates x and y, is assumed to be linear after a calibration measurement was used to equal the signal responses to neutrons of the different photomultipliers. The positions x and y within the central photomultiplier (i, j) are then linear functions of their respective light asymmetry values q_i and q_j.

Another reconstruction method investigated is a maximum likelihood method which will be called "naive Bayes" in its description in section 5.3.4. For any pair (x, y) of the 128×128 positions the expected signals from all 64 photomultipliers have to be determined. For an observed event the 64 photomultiplier signals then determine the correct pair (x, y) as the position for which the observed photomultiplier signals match the memorised expectations best (maximum probability):

$$(x, y) := \mathrm{argmax}_{(x,y)} \left[\prod_{i=1}^{64} P\Big(S_i = E_i(x, y)\Big) \right] \tag{2.7}$$

where E_i are the memorised expectations given a certain "true" position and S_i are the observed signal values. Unfortunately it is difficult to get the "training" data, i.e. the expectation values for any position. The simulation of the detector is not suited well to be used for this because there are some effects observed in experimental data which are not simulated and thus lead to differences in the final photomultiplier values.

The problem of the missing training data will be discussed further in the analysis in section 7.5. The exact reconstruction of the neutron incident position with the help of statistical learning methods is the main target of the analysis presented there.

The time constraints make this application especially interesting. Since only histograms of the reconstructed positions can be stored, the position reconstruction has to be done online. The full detector readout cannot be stored for every event because of count rates of several kHz. The hardware for the position reconstruction currently used in the detector system consists of 16 Digital Signal Processors (DSPs) which all work in parallel (see appendix A).

2.4 The MAGIC Telescope

Figure 2.21: The MAGIC Telescope by the time of its inauguration in October 2003.

The MAGIC (Major Atmospheric Gamma ray Imaging Cherenkov) telescope [21] is located on the Canary island La Palma. Figure 2.21 shows the MAGIC telescope by the time of its inauguration in October 2003. The main goal of this experiment is to close the observational gap in the energy region between 10 GeV and 500 GeV of the astronomical γ-ray flux because this window is not covered well at the moment neither by satellite-based experiments, which measure at lower energies, nor by ground based experiments, which measured up to now at higher energies (compare figure 2.22).

A wide range of astronomical phenomena from the *non-thermal universe* is covered by these observations in the energy range between 10 GeV and 500 GeV, ranging from Active Galactic Nuclei and SuperNova Remnants over Gamma Ray Bursts to Dark Matter. Important observation targets are given by the EGRET catalogue shown in figure 2.23. Unidentified sources need to be identified and the energy spectra of all sources have to be analysed – a cut-off is expected in many of them because there are so much less sources discovered by Cherenkov telescopes (figure 2.24).

There are several key points to reach the 10 GeV energy threshold, which is very low compared to all existing Cherenkov telescopes:

- The large mirror surface of the 17m diameter reflector dish collects enough Cherenkov light even from very small (low energy) showers.
- A high photon detection efficiency is guaranteed by highly reflective mirrors and photomultipliers (PMTs) with a high quantum efficiency.
- A sophisticated background rejection is applied which can identify shower images induced by charged particles, mainly protons, and keeps a large fraction of the γ-induced events.

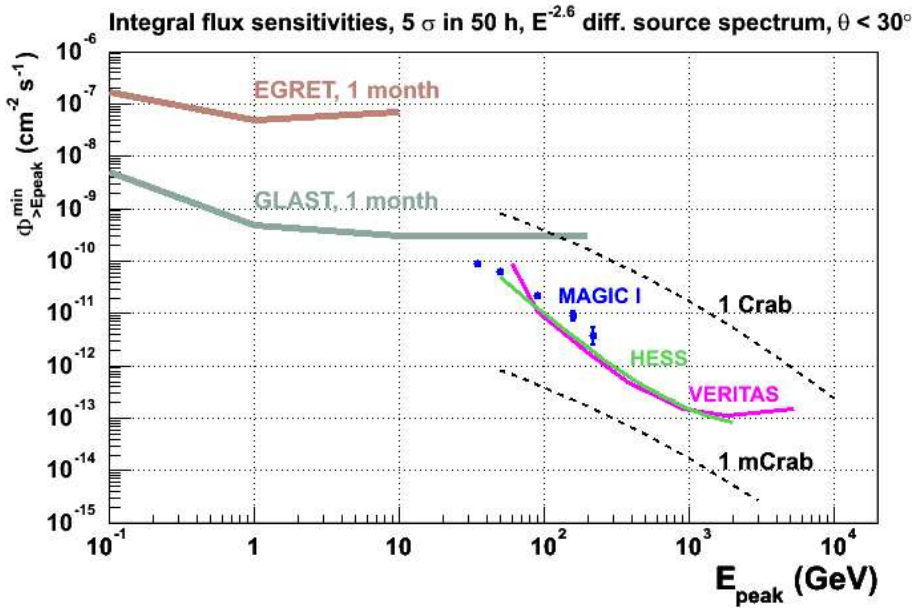

Figure 2.22: The flux sensitivity of MAGIC reaches down to the energy range of the satellite based experiments.

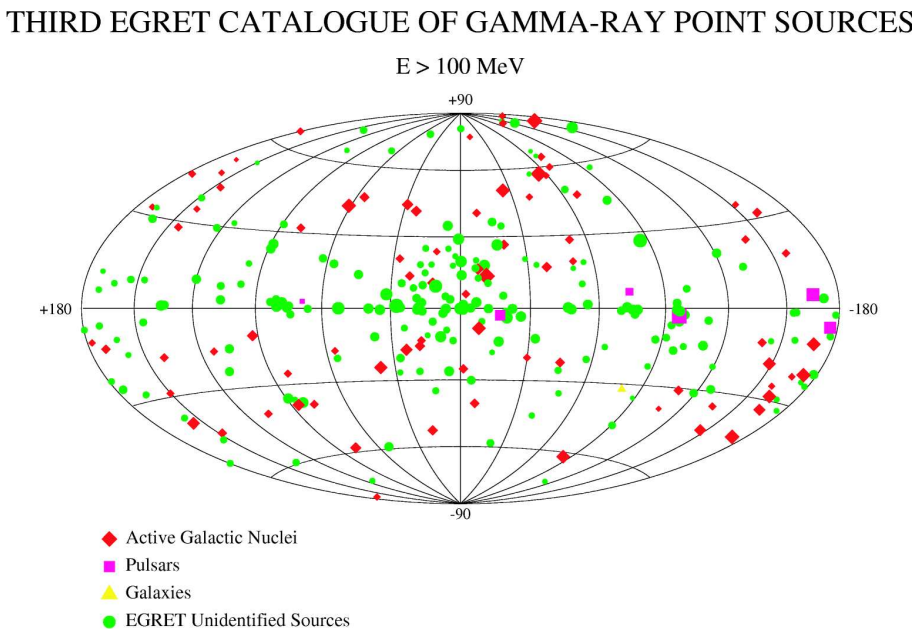

Figure 2.23: Sources detected by EGRET in the energy range $100MeV < E < 10GeV$. Taken from [22].

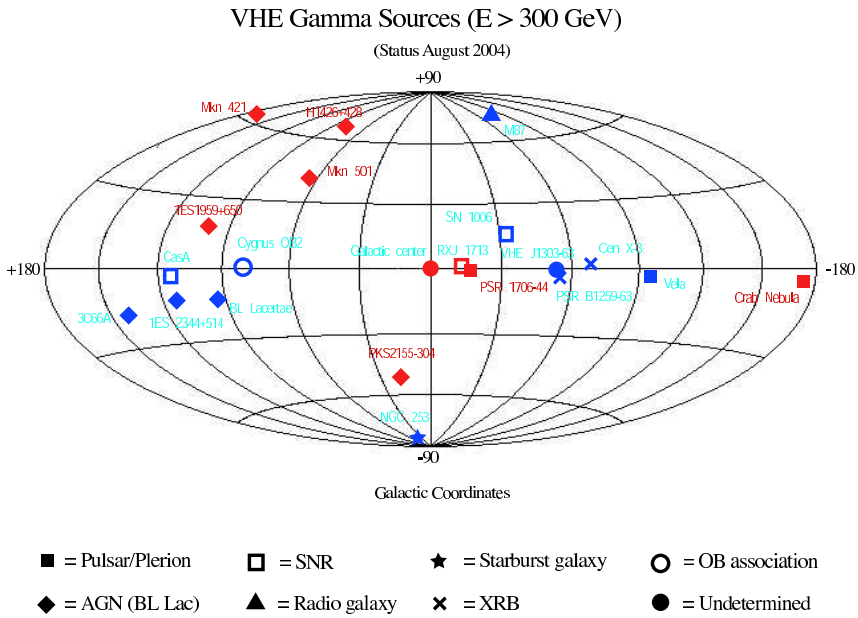

Figure 2.24: Sources detected by Cherenkov telescopes in the energy range $E > 300GeV$. Red symbols indicate sources detected with certainty, blue symbols need further confirmation.

Besides the tasks that are directly related to the collection of Cherenkov light some technical issues have led to innovations:

- The carbon fibre frame is both lightweight and rigid so that the telescope can be positioned to point at any source in the sky within only $30s$.
- The lightweight aluminium mirrors are produced fast and cost effective, they have a longer lifetime compared to conventional glass mirrors and can be heated to prevent the formation of dew and ice.
- The Active Mirror Control allows to counteract small residual deformations of the frame by monitoring a laser spot in a CCD camera.

2.4.1 Extensive Air Showers and Imaging of Cherenkov Light

A high energy γ-ray or cosmic ray nucleus which enters the atmosphere interacts the first time at a height of about 10 to 25 km, producing secondary particles which themselves interact, leading to a cascade called "extensive air shower". Since the energy of the primary particle is distributed over all secondary particles the cascade development stops when the energy of a single particle falls below the threshold for further particle production. The shower then dies out and energy losses due to ionisation processes become dominant.

Figure 2.25 shows the differences of the principal development of extensive air showers, depending on the type of the primary particle.

- A **γ-induced shower** starts with the production of an electron-positron pair in the electromagnetic field of an atmospheric nucleus. Produced electrons or positrons radiate new γ-rays (bremsstrahlung) which themselves produce pairs of electrons and

positrons. Since only electromagnetic interactions take place this kind of shower is called electromagnetic. Below a critical value for the mean energy of the electrons and positrons ($83MeV$ in air) ionisation replaces bremsstrahlung as the dominant process of energy loss. In addition the cross section for pair production becomes of the same order as for Compton scattering and photo absorption at energies of a few MeV. Thus the shower reaches its maximum development at a mean particle energy around 10-100 MeV.

- A **hadron-induced shower** starts with the collision of the cosmic hadron with an atmospheric nucleus producing pions, kaons and nucleons. About 90% of all secondary particles produced in the shower are pions which continue to multiply in nuclear collisions until their mean energy falls below the threshold for pion production (about 1 GeV). Through the quick decay of π^0's into two γ-rays electromagnetic sub-showers are created. Though having a hadronic core of nucleons and mesons even hadronic showers are thus dominated at their tail by photons and electrons.

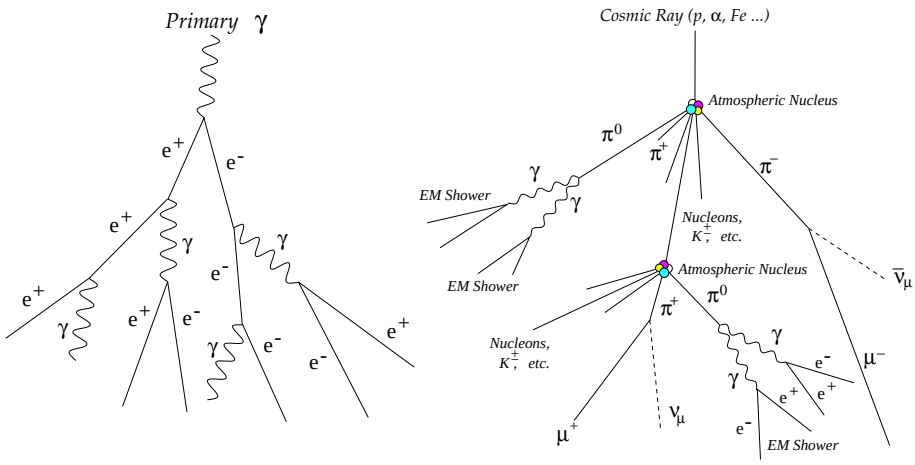

Figure 2.25: Development of extensive air showers induced by a γ-ray (left) or by a charged cosmic ray (right).

Cherenkov light is emitted by charged particles which travel through a medium (here air) with a speed greater than the speed of light in this medium:

$$v = \beta c \text{ with } \beta > \frac{1}{n}. \tag{2.8}$$

A net polarisation of the medium along the trajectory of the particle occurs as shown in figure 2.26 (b) and the superposition of all the wavefronts according to Huygens' principle leads to a fixed radiation angle

$$\cos(\theta) = \frac{1}{\beta n} \tag{2.9}$$

as shown in figure 2.26 (c). To compute the Cherenkov light emitted in extensive air showers the variation of the air density (and thus of he refractive index) with the height has to be taken into account. Calculating the energy thresholds for emission of Cherenkov light shows that nearly all light in an extensive air shower is produced by electrons due to their small rest mass compared to muons or protons.

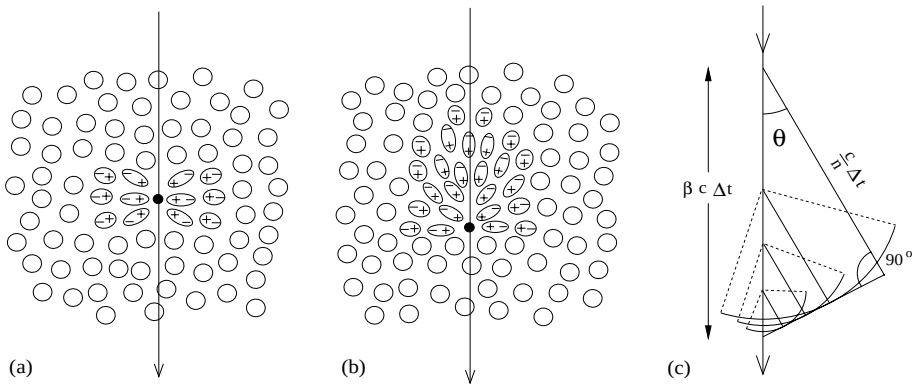

Figure 2.26: Development of Cherenkov light: The polarisation of the dielectric medium is shown for a charged particle with (a) low speed and (b) high speed. The Cherenkov light cone for (b) is derived by Huygens' principle (c).

Figure 2.27 visualises the differences in the shower development between an electromagnetic and a hadronic shower (upper part of the picture) and shows the Cherenkov light as projected into the camera of a Cherenkov telescope (lower part of the picture). The energies of the primary particles were chosen so that the total number of Cherenkov photons is almost the same in both pictures. Nonetheless clear differences in the shape of the projected light can be seen which will be the basis for the suppression of the hadronic background, to be discussed in the next section.

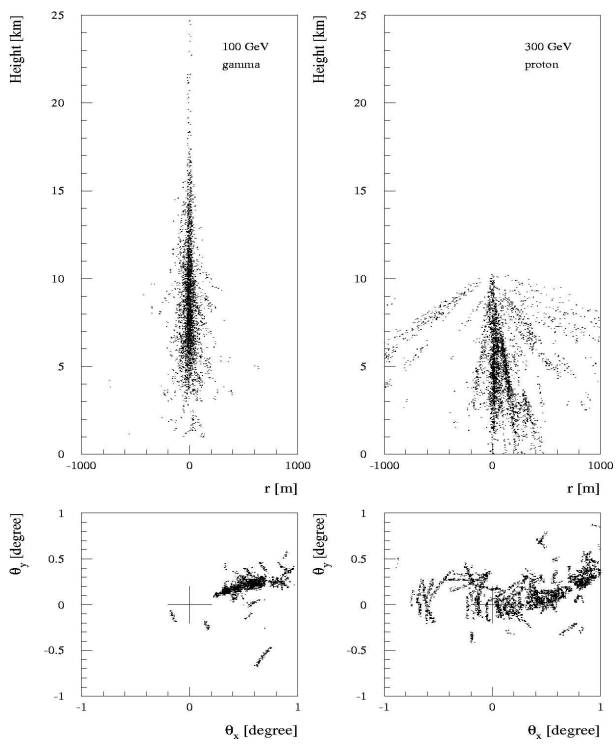

Figure 2.27: Two simulated air showers allow the comparison of a photon-induced shower (left) with a proton-induced shower (right) in terms of their shower development (upper part) and the produced Cherenkov light as seen in the focal plane of a Cherenkov telescope (lower part). The energies are chosen such that the amount of produced Cherenkov light is comparable for both cases.

2.4.2 Shower Image Analysis and Background Rejection

Emphasis was put on the differences between extensive air showers induced by a photon on the one hand and induced by a hadron on the other hand. This difference becomes essential as soon as the measurement of properties of astronomical objects – which was the primary objective – comes into focus. Charged hadrons which enter the atmosphere do not point back to their origin – the information about the location of their emission is lost due to the deflection in the weak galactic and intergalactic magnetic fields[3]. Therefore hadron-induced showers have to be separated from the photon-induced showers which form the basis for any physics analysis. Even though there are large differences in the shower images, a factor 10^4 more hadron-induced showers than photon-induced showers makes sure that there will be enough hadron-showers which look like photon-showers and the rejection of all hadron events will be impossible. A high background rejection combined with a high γ-efficiency by means of statistical learning methods is the main target of the analysis presented in section 7.6.

Figure 2.28 shows in more detail how the Cherenkov light emitted by a photon-induced extensive air shower is projected into the camera. The shower trajectory is parallel to the pointing axis of the telescope which means that the photon comes from the direction the telescope is pointing to. The Cherenkov light emitted at large altitude will therefore be found near the camera centre while the light emitted later (close to the surface) is seen more and more towards the outer part of the camera.

The resulting ellipsoidal shape of the γ-induced Cherenkov light is described in the standard *Hillas* analysis [23] with the geometrical properties of the resulting Cherenkov ellipse as shown in figure 2.29. In addition to the geometrical quantities shown in the picture an important parameter is for example `size`, the total number of photons in the shower image. The calculation of the geometrical quantities is usually not done by fitting but by calculating the first and second moments of the light distribution (pixel-positions weighted by the number of Cherenkov photons counted there).

Since there is usually much noise in the camera image (both from electronics and real light from the *night sky background*) an *image cleaning* procedure is performed before doing the Hillas analysis. The image cleaning rejects those pixels in which the number of photons is in the order of the fluctuations of the noise. Figure 2.30 shows the effect of the image cleaning on photon- and proton-induced simulated events at different energies. It can be seen that only very few pixels "survive" the image cleaning for low energies – too few for a Hillas analysis as discussed above. This suggests that in the low energy region methods which operate on the uncleaned image may be better suited to perform a γ-hadron separation.

A standard method to perform the γ-hadron separation is the *supercuts* method [24]. This method applies a set of cuts in the Hillas parameters which may depend on other Hillas parameters. For example a cut in `width` is done depending on the value of `size` (compare figure 2.29). The dependences are parameterised and the optimal parameters are found in an optimisation process. This optimisation process varies the parameters and thus the dynamical cuts in the Hillas parameters until a (potentially local) maximum or minimum of the optimisation target has been found.

A typical optimisation target is the significance of the detected photon signal [25] which

[3] Only for very high energies ($> 10^{19} eV$) they could point to their sources but the flux in this energy region is extremely small.

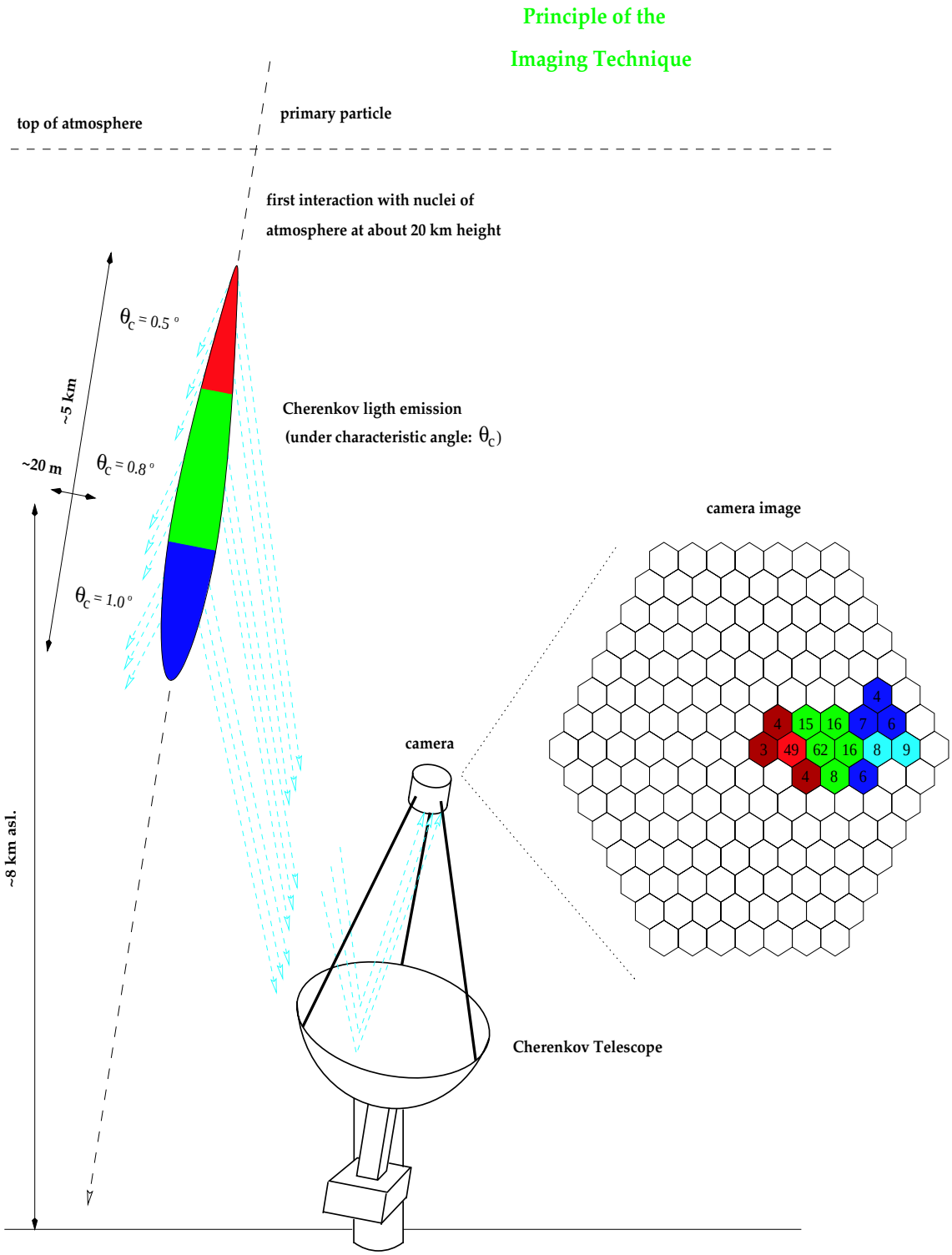

Figure 2.28: The principle of Imaging Air Cherenkov Telescopes. The stated dimensions are typical for a 1 TeV γ-induced shower. The numbers in the camera pixels denote the amount of detected Cherenkov photons.

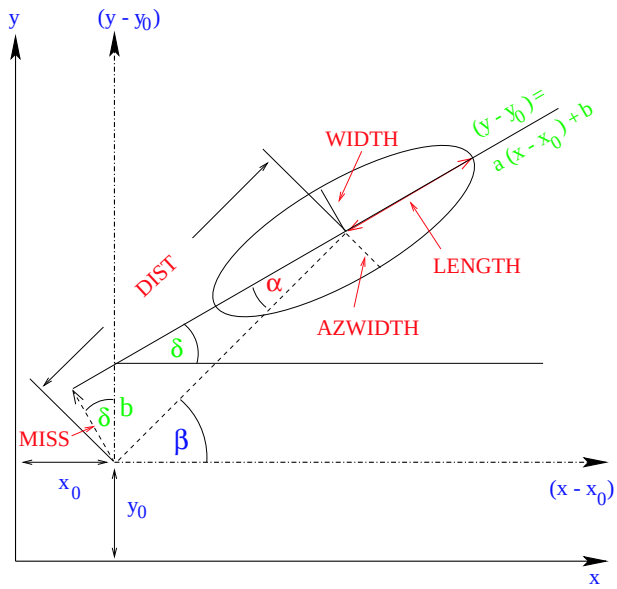

Figure 2.29: Standard *Hillas* analysis of the shower image light distribution. (x, y) are the coordinates in the original camera system and (x_0, y_0) are the coordinates of a reference point like the source position (which mostly is the camera centre).

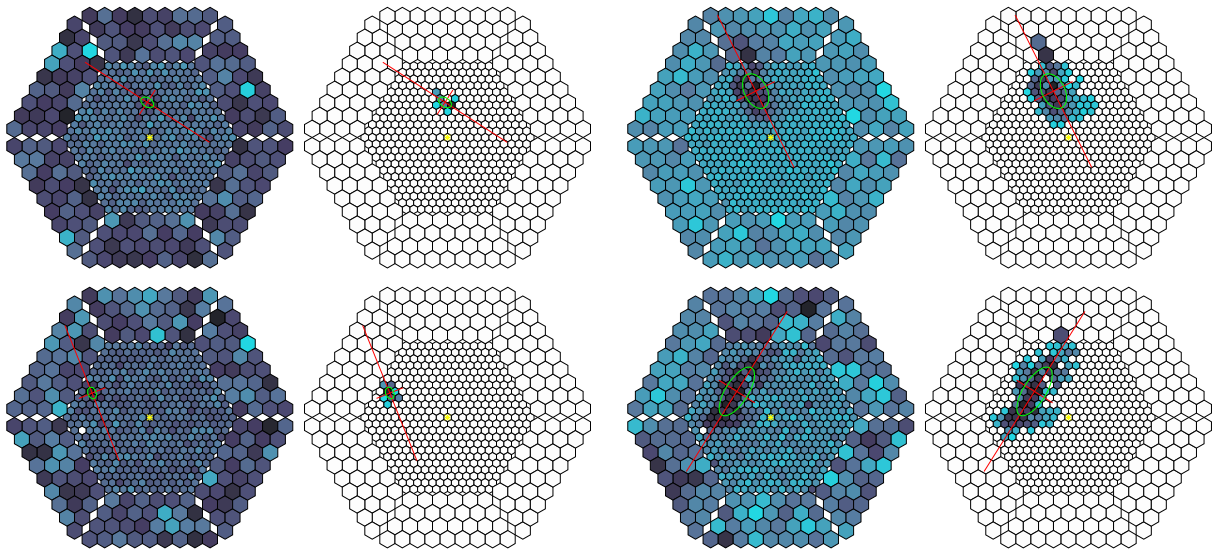

Figure 2.30: Image cleaning at different energies (left below $100GeV$ and right above $300GeV$) for photons (upper row) and protons (lower row).

is derived from the α-*plot* shown in figure 2.31. The Hillas parameter α (see figure 2.29) measures the angle between the shower axis and the line connecting the reference point with the centre of gravity of the shower. For photons this angle α is usually very small as their Cherenkov ellipses point towards the source position. The hadronic background on the other hand produces a quite flat distribution in α as these events do not point to the source position. In figure 2.31 the peak towards small α contains the γ-induced events while the flat background over the whole range of α is generated by the hadron-induced events. The peak is much clearer and thus the significance much higher after application of the supercuts method because the background was dramatically reduced but the amount of γ-events stayed similar.

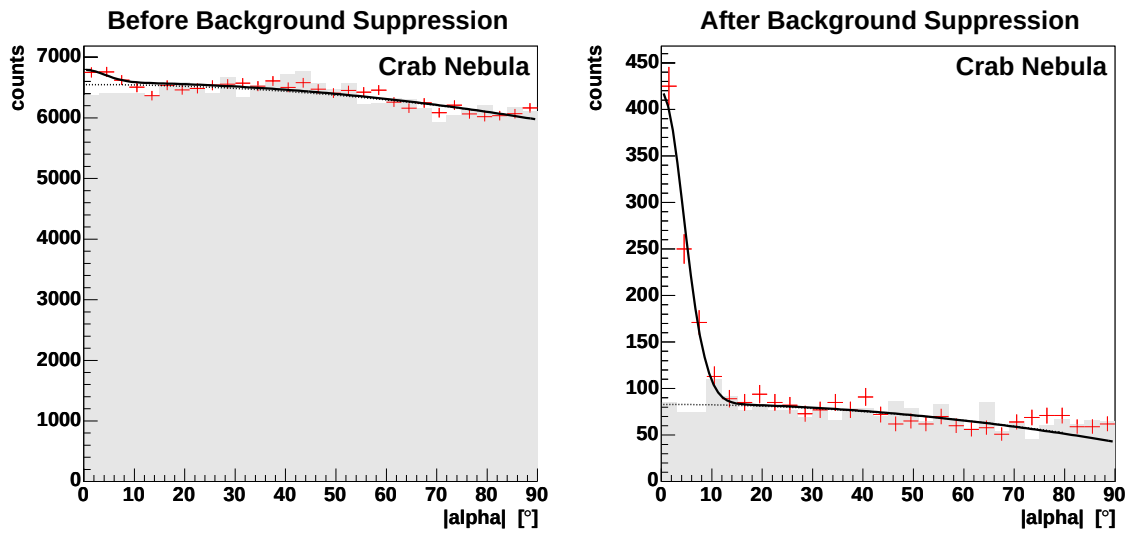

Figure 2.31: α-plots before and after γ-hadron separation: the γ-excess for small α's in the ON dataset (crosses) compared to the OFF dataset (grey bars) is clearly seen only after the background suppression has been done, the background is fitted by a 2 or 4 dimensional polynomial (without odd powers due to symmetry), the signal is fitted in the same way with a Gaussian centred at $\alpha = 0$ added.

Plotting the distribution of α for all events can therefore be used to measure the significance of the photon signal contained in the whole dataset and especially contained in those events that remain after the application of some method for the γ-hadron separation. A few details have to be noted: α must of course not be used in the method which performs the γ-hadron separation because it is used here for an independent check. Furthermore the analysis of the α-plot is more reliable if two datasets are used like in figure 2.31. Although it is possible to check the γ-hadron separation with only one dataset that contains the photon signal and of course the background (called *ON* dataset because the telescope points onto the source) the analysis is more reliable if a second dataset is used in addition (called *OFF* dataset because the telescope does not point onto the source). This *OFF* dataset contains no or only a marginal amount of photons and can thus be used to monitor the background for small α for which the photon excess is seen in the *ON* dataset.

2.4.3 Energy Estimation

After the background has been reduced as much as possible, the remaining events (assuming they have been induced by a photon) have to be analysed. There are two basic quantities of each event which are essential for physics analysis: The arrival time, for measuring variations of the γ-flux with time, and the energy of the primary photon, for measuring the energy dependence of the γ-flux. The arrival time is measured quite easily by logging a time stamp together with the event data. The energy of the primary photon, however, is much more difficult to find out because it is not measured directly but has to be estimated from the Cherenkov light which was collected. One of the most important quantities for this energy estimation is of course the `size` parameter which shows a strong correlation with the energy of the primary photon in the simulation as shown in figure 2.32 (a large `size` – a large number of observed Cherenkov photons – corresponds to a large energy).

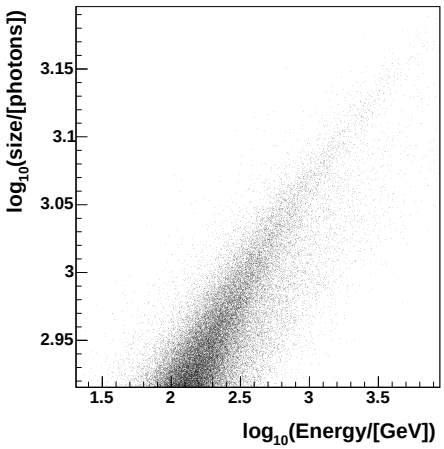

Figure 2.32: Dependence of `size` on the energy of the primary photon from the simulation after passing the trigger of the telescope and the image cleaning procedures. The scatter plot shows the broad range of energies which could belong to a measured `size`.

The estimation of the energy of the primary photon is not possible by using only `size`, as figure 2.32 demonstrates. Complex parameterised relations between the energy of the primary photon and the Hillas parameters are usually used like:

$$\begin{aligned} E_{est} = \quad & a + b \cdot \texttt{size} + c \cdot \texttt{conc} + d \cdot \texttt{length} + e \cdot \texttt{width} \\ & + f \cdot \texttt{dist} + g \cdot \frac{\texttt{width}}{\texttt{length}} + h \cdot \frac{\texttt{size}}{\texttt{width} \cdot \texttt{length}} + i \cdot \texttt{leakage} \end{aligned} \tag{2.10}$$

where `conc` describes the concentration of the Cherenkov light among the illuminated pixels and `leakage` tells about the estimated amount of Cherenkov light which missed the detector.

As already seen in figure 2.30, the determination of the Hillas parameters is quite difficult for low energy events. Thus methods which do not rely on image cleaning and Hillas parameters may be better suited to estimate the energy of the primary photon in the low energy region. A precise reconstruction of the energy of the primary photon with the help of statistical learning methods is the main target of the analysis presented in section 7.6.

2.5 The XEUS Satellite

The X-ray Evolving Universe Spectroscopy (XEUS) mission [26] is a potential follow-up project to ESA's cornerstone X-ray Multi-Mirror mission (XMM-Newton) [27]. The science goal of the XEUS mission is the hot universe at far redshift, the first massive black holes, the first galaxy groups and the evolution of heavy elements. The satellite will probably be launched after 2015.

Figure 2.33 shows an artist's view of the XEUS satellite which will consist of two independent spacecrafts: the mirror spacecraft and the detector spacecraft which fly with a $50m$ focal length distance aligned by a laser tracking system (see also figure 2.34).

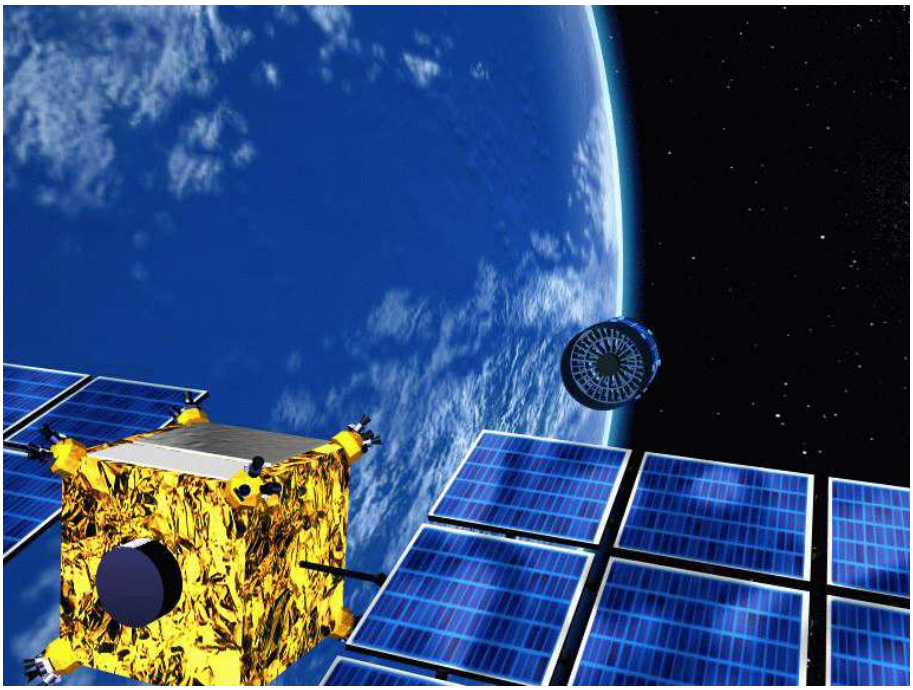

Figure 2.33: The XEUS satellite with its two components (mirror spacecraft and detector spacecraft) in an artist's view.

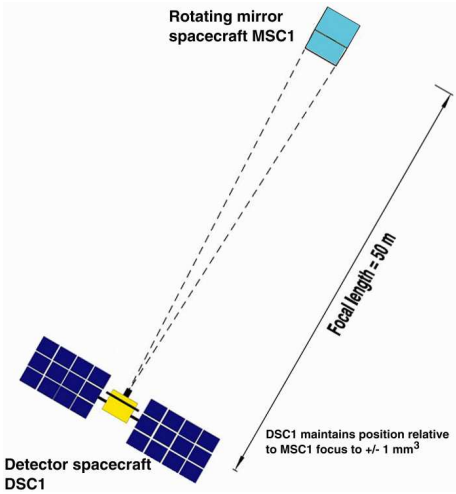

Figure 2.34: Components of the XEUS satellite: The mirror spacecraft and detector spacecraft will be aligned by a laser tracking system.

The collection area is planned to be 30 m^2, corresponding to an increase in the photon count rate of a factor 200, compared to XMM-Newton. Different detectors will be placed in the detector spacecraft which can be rotated into the focal plane. To match the optical requirements for a wide field imager on XEUS, an APS array with 1024 x 1024 pixels of $75 \times 75 \mu m^2$ is being developed [28]. The intrinsically fast read-out of the APS allows frame rates of 500 Hz, corresponding to 0.5 GigaPixel/s.

The specific type of pixel detector will not be relevant in the analysis presented in section 7.7. This analysis works with any kind of pixel detector, charge-coupled devices (CCDs) or active pixel sensors (APSs), which deliver imaging and spectroscopic information about incident radiation. MOS- and *pn*-CCDs [29] e.g. are implemented as focal plane instruments on the XMM-Newton satellite. Together with Active Pixel Sensors (APS) they remain first choice devices also for future X-ray space telescopes like XEUS or, for example, the ROSITA (ROentgen Survey with an Imaging Telescope Array) mission [30].

To transform the raw data stream coming from the pixel detector into scientific information, a processing scheme has to be applied which will be discussed in the following exemplary for a CCD device as shown as schematic cross section in figure 2.35. The analysis presented in section 7.7 will mostly be based on a generic simulation which can simulate different types of pixel detectors from CCDs to APSs.

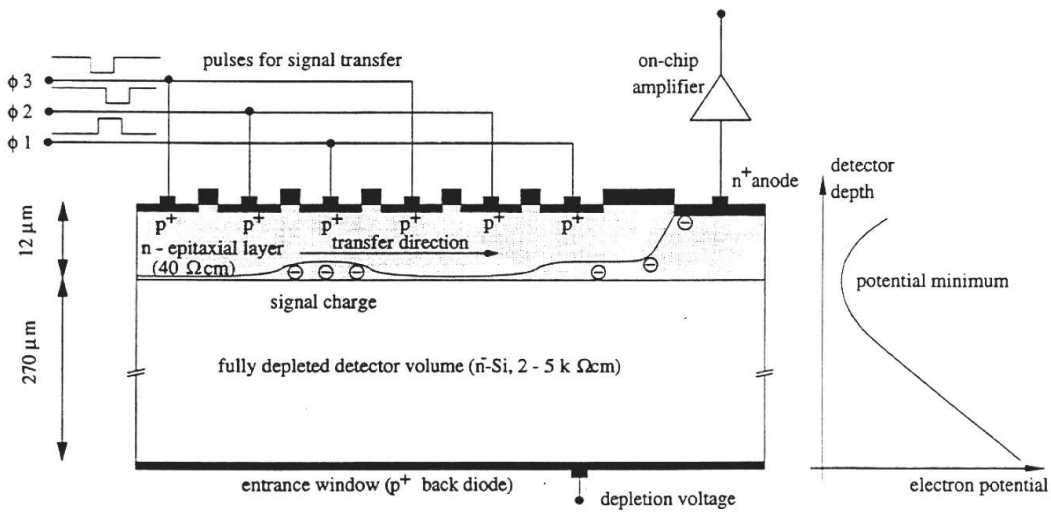

Figure 2.35: Schematic cross section of a fully depleted *pn*-CCD like the one used on the XMM satellite.

An X-ray of a few hundred eV to more than $10keV$ enters the semiconductor from the backside (lower side in figure 2.35) and converts into an electron-hole pair. This pair generates in a small cascade further electron-hole pairs until their mean energy drops below the energy threshold for the production of electron-hole pairs (3.68 eV in Si). The holes drift to the back side while the electrons drift into the potential minimum under the front side. This vertical potential minimum is horizontally divided into pixels so that the electron-cloud generated by one X-ray photon is typically distributed over up to four neighbouring pixels. Figure 2.36 shows which patterns can be generated after a threshold is applied which suppresses the noise.

The main target of the processing of the data stream is to measure precisely position and energy for each single photon. Therefore detector effects have to be corrected and

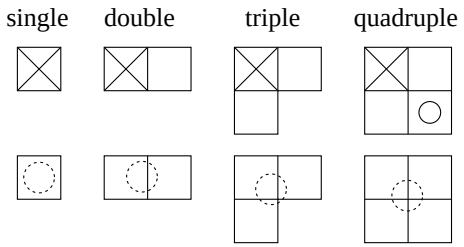

Figure 2.36: Patterns which can be generated by single photons: Shown are the illuminated pixels (above some threshold). In the upper row the cross marks the maximum charge and the circle the minimum charge. In the lower row the generation of the pattern is visualised with the generating charge cloud. Triples can be generated if the fourth signal is below the event detection threshold.

background has to be rejected. The correction of detector effects is an important preprocessing step to any further analysis. This step includes the correction of pixel-wise and row-wise ("common mode") offsets, different column-wise gains and charge transfer inefficiencies (see appendix B.1 for a detailed description). The main background is due to noise which is simply cut away by applying an event detection threshold of e.g. 5σ. The illuminated pixels (those with a value above threshold) are then grouped to patterns. The second kind of background are minimum ionising particles which are identified by the large charge deposition which is far above the energy band of interest. The third kind of background are *pileups* meaning that two photons hit the CCD during the same exposure time and close together so that they are recorded in the same frame and their patterns are at least adjacent or even overlap. Figure 2.37 shows examples of four levels of pileup formation from *pattern pileup* to complete *charge pileup*. At least in the cases (b), (d) and (e) the energy information of each single photon is lost, the event cannot be used in the analysis.

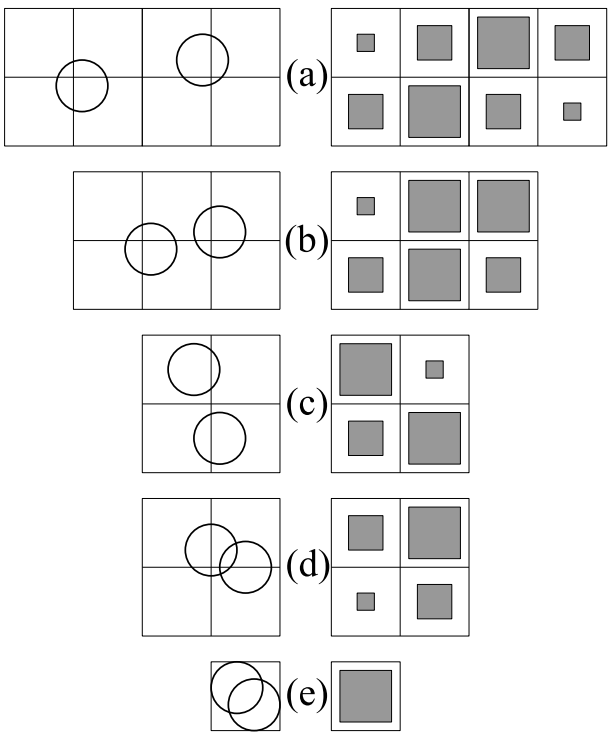

Figure 2.37: Pileups from pure pattern pileup (a) and (c) over mixed pattern and charge pileup (b) and (d) to pure charge pileup (e).

Case (d) in figure 2.37 cannot be filtered out by using simple patterns like those from figure 2.36. A more sophisticated analysis may be able to recognise the differences between a pileup and a single photon whose charge distributions look almost the same. The reliable recognition and rejection of pileups with the help of statistical learning methods is the main target of the analysis presented in section 7.7.

The information about the position and the energy of the event is finally derived from the pixel coordinates and the digitised electron count, respectively. The energy resolution is limited by the noise of the detector. After the above mentioned corrections the energy of the photon is determined by simply adding up the charges of the illuminated pixels. The position, however, can be determined more precisely than just in pixel-coordinates[4] by taking into account the splitting of the generated charge cloud among the neighbouring pixels. The incident position of the photon or to be more precise the exact position of the centre of the generated charge cloud can then be estimated with sub-pixel resolution (compare figure 2.38). The exact reconstruction of the centre of the generated charge cloud with the help of statistical learning methods is the main target of the analysis presented in section 7.7.

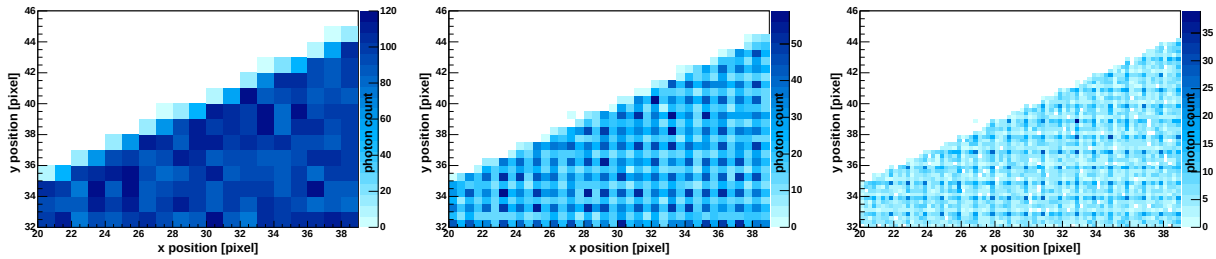

Figure 2.38: Resolution improvements by using the distribution of the charge over up to four pixels. A sharp edge is simulated in a pixel detector with $50 \times 50\mu m^2$ pixel size. On the left only the position in pixel coordinates of the maximum energy deposition per event is used. In the middle a centre-of-mass method is used to estimate the incident position from the charge distribution. On the right the η-method (presented in section 7.7.2) is used to estimate the incident position from the charge distribution. The binnings of the histograms are adapted to the obtained resolutions.

To be able to train a statistical learning method for the reconstruction of the incident position some kind of training data is needed. The problem of generating training data to train a statistical learning method will be discussed generally in section 3.10. The next section presents an experiment with the pixel detector which allows us to generate training data for the problem of the reconstruction of the centre of the generated charge cloud.

[4]The analysis of the XMM data, for example, uses only pixel coordinates i.e. the coordinates of the maximum charge. Sometimes the analysis is even restricted to use those events only which did not split over more than one pixel (for XMM with $150 \times 150\mu m^2$ pixels this is still a fraction of more than 60%) because this improves the energy resolution. Using the coordinates of the maximum charge only is also no problem from the detector point of view: The resolution given by the mirrors is already worse than one pixel. Therefore any determination of a position with sub-pixel resolution seems useless. However this is a general analysis which is also applicable to pixel-detectors which are for example used as tracking devices in a colliding experiment. Any additional resolution is welcome in such an application.

2.5.1 The Mesh-Experiment

The *mesh* in our experiment is a $10\mu m$ thin gold foil with small holes of $5\mu m$ diameter which are arranged in a rectangular grid pattern with spacings equal to or a multiple of the CCD structure e.g. $150\mu m$ in x and y direction for a CCD with $75 \times 75\mu m^2$ pixels. The mesh is placed directly in front of the CCD and blocks the homogenous illumination with X-rays from entering the CCD except for the given holes. A slight rotation of the mesh with respect to the CCD structure leads to different placements of the holes over the respective pixels as shown in figure 2.39. Under the assumption that all pixels behave equally the holes sample thus the whole area of a pixel and define isolated incident positions for photons.

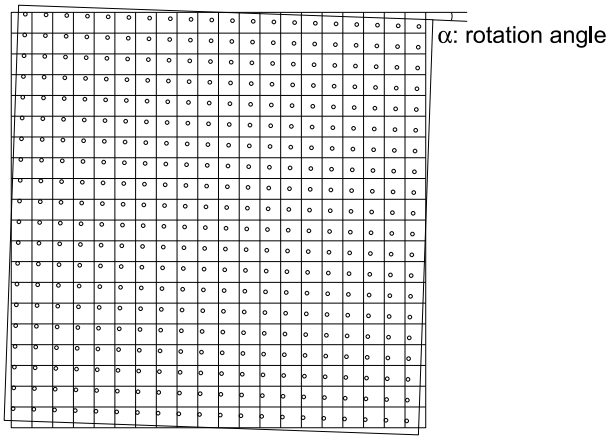

Figure 2.39: The mesh-experiment: A small rotation of the mesh structure with respect to the pixel structure places the holes over different parts of the individual pixels.

The main target of the mesh-experiment is to understand the detector behaviour. The charge splitting is caused by diffusion and electrostatic repulsion of the charge carriers during their drift time. The charge splitting thus also depends on the electric field which affects the electrons as they drift to the front side into the potential minimum. Knowing how the electron cloud is split among different pixels due to the electric fields for each specific incident position will help to understand the detector behaviour.

An interesting byproduct of the mesh-experiment is the possibility to generate training data for the reconstruction of the incident position. Real experimental data could then replace the data coming from Monte Carlo simulations. As will be discussed in section 3.10 the usage of experimental data should of course be preferred. To have training data means to know the correct incident position for each event. This information is then used to teach a reconstruction method. The mesh experiment offers this unique possibility: to see real events in the detector and to know in addition where they come from because the position of the respective mesh hole is known (appendix B.2 will show how the position and angle of the mesh relative to the CCD can be determined).

Chapter 3

Statistical Learning for Physics Experiments

In this chapter the basic concepts and properties of statistical learning will be explained. The emphasis is put on the motivation why and in which case statistical learning methods should be used and on the way they are applied correctly. Most of the mathematical background is skipped here and will be dealt with in the next chapter.

The terminology and all the examples are closely related to the application of statistical learning to physics experiments. Nevertheless all comments and advices apply also to other fields of application.

The next few sections (3.1 to 3.6) start with an introduction to statistical learning and present the most basic concepts and notions. Afterwards (sections 3.7 to 3.10) the classical alternatives to statistical learning and the most common motivations to use statistical learning methods will be discussed. In the following sections (3.11 to 3.13) the correct handling and control of statistical learning methods during the training and in the performance evaluation will be explained including the calculation of statistical und systematic uncertainties. This chapter ends with a description of the data mining capabilities of statistical learning methods (section 3.14) and with a guide to compare learning methods in a statistically correct way (section 3.15).

3.1 Statistical Learning in the World of Artificial Intelligence

Historically the development of artificial neural networks as simplified models of the human brain was one of the fundamental starting points of artificial intelligence and signified the desire to make machines "intelligent", similar to humans. Other starting points made use of knowledge databases or tried to induce logical rules. Whereas the structure of neural networks was derived from the first insights into the structure of the brain (mainly of animals but also of humans), the idea behind the neural network training is clearly induced by psychology. Supervised learning, the interaction between a flexible learner and a precise teacher, is still one of the most fundamental and successful paradigms of today's artificial intelligence (AI). Although the focus of current AI research shifted to agent-based systems, distributed learning and parallel systems, statistical learning – the principle of learning by examples – remains a very important building block [31].

A common problem with statistical learning methods is the need for a teacher (compare the introduction in section 3.3) who gives a feedback for every single decision which is made by the method. In real world applications we often have only a global reinforcement which means that the feedback is given after a time-dependent series of decisions. A major focus of current AI research is to break down the global feedback to the local decisions. Unfortunately the methods which generate the local from the global feedback are mostly application dependent (see for example [32]).

General overviews of the relation between machine learning, artificial intelligence and statistical learning can for example be found in [33, 34, 35, 36, 37].

3.2 Inputs

The common basis of all statistical learning is the desire to extract some kind of information for later usage ("learning") from a dataset consisting of N examples (or *events*, *patterns*) drawn from some ensemble ("statistics"). In other words we want to create and choose among hypotheses on the basis of evidence given by examples. The information coming from these examples is usually written as a list of vectors $\vec{x}_k$, $k = 1 \ldots N$. The vector $\vec{x}$ is called *input* vector and is complemented by the *target* y in the case of supervised learning as discussed in the next section.

The components of the input- or *feature*-vector $\vec{x}$ can represent any information we have about the dataset from which we want to learn. The datatype can be

- categorical, e.g. "which experiment?" (H1—ZEUS—CDF),
- ordinal, e.g. "run quality?" (poor—medium—good) or
- numerical either discrete or continuous, e.g. "energy?" (0-100GeV).

For physics applications, the inputs for the statistical learning method usually characterise an event which was seen in some kind of detector. Inputs can be raw values of different detector components but they can also consist of derived quantities, which have not been measured directly but which describe the structure of the event on a higher level (compare the discussion of preprocessing in section 3.6).

In the following, all inputs will be assumed to have a numerical datatype. While ordinal inputs can be easily transformed by simply numbering the states, the case is more difficult for categorical inputs: Binary representation may be a better choice for some statistical learning methods than simple numbering since this does not create an artificial ordering of the categories.

Example: Categorical Inputs

If the name of the experiment (as mentioned above) should be encoded in the input vector of each event there are basically two possibilities: Either we use one input value with the values 0 (H1), 1 (ZEUS) and 2 (CDF) or three inputs are used which are either 0 or 1 (binary representation): 1-0-0 for H1, 0-1-0 for ZEUS and 0-0-1 for CDF. The first method has the disadvantage that some learning methods may try to interpret the artificial ordering H1<ZEUS<CDF.

Choosing the right inputs is an important task. Usually higher level quantities (after using prior knowledge) are better suited. The data compression step from "raw" data – which usually consists of many inputs – to a higher level of abstraction with usually fewer inputs is called preprocessing (discussed in section 3.6).

Careful consideration of the known meanings and interactions of the inputs that are available are typically combined with many cycles of trial and error in which different combinations of inputs are tried out. For supervised learning (see below) the correlation between each input x_i and the *target* value y

$$\rho_i = \frac{cov(x_i, y)}{\sigma_{x_i}\sigma_y} = \frac{1}{N}\sum_{k=1}^{N}\frac{(x_{ik} - \bar{x}_i)(y_k - \bar{y})}{\sigma_{x_i}\sigma_y} \tag{3.1}$$

gives an indication which input may be suited well to describe the target because of a high correlation. A technique called *relevance* which estimates the importance of each input after the training has been done will be discussed in section 3.14.

Too many inputs with only a moderate number of training examples lead to serious problems. The *curse of dimensionality* means that too few training examples distributed in a very high dimensional space result in such a sparse density that the learning task will be very difficult, especially overtraining (see section 3.11) will be a problem. Section 3.6 will discuss some algorithms which try to reduce the dimensionality of the input space while trying to not loose information.

To deal with missing values for some components of the input vector in some events is a task that is important, for example, in medical applications (e.g. a specific test was not done on some of the patients). Therefore some learning methods have the capability to deal with missing information. In the further discussion, however, the data will be assumed to be complete.

A *weight* might also be part of the information that is available for each event. The weight does not serve the purpose to describe each example but to tell about its importance. Usually all examples have the same importance and thus always weight 1. But for some datasets some events may be regarded as being more important than others resulting in a weight $w_i > 1$ for some and $w_i < 1$ for others. These weights may come from Monte Carlo simulations or may be introduced by purpose to modify the behaviour of the learning method (the boosting method does so as described in section 5.5.4). Weights must not be used like normal inputs to distinguish one event from another but they should be used to steer the learning process and to evaluate the performance (see section 3.12).

3.3 Supervised and Unsupervised Learning

Supervision of a statistical learning method means that each input vector $\vec{x}$ gets a label, the *target* value y. These target values have to be predicted by the learning method and a *teacher* is available who provides feedback about the errors which are made.

For unsupervised learning the only available information is the sampled distribution of data points in the input space. Naturally the task is then to derive statements about the underlying probability distribution from which the data points were sampled. Typical statements that result from unsupervised learning tell about local densities and clusters. Unsupervised learning is sometimes useful as a preprocessing step to supervised learning. Section 3.6 discusses, for example, clustering as a way to reduce the dimensionality of a

problem. Typical unsupervised learning methods are discussed for example in [38, 31]. In the following we will concentrate on supervised learning.

Supervised learning means that N pairs of input and corresponding target $(\vec{x}_i, y_i)$, $i = 1 \dots N$ are given to the learning method. The special case where the learning method itself can decide at which position $\vec{x}$ in the input space the corresponding *target* value y should be given by the teacher is called *active learning*. Throughout this thesis we will always assume the more general case where the N pairs $(\vec{x}_i, y_i)$, $i = 1 \dots N$ are fixed in advance.

The inference from these N pairs of input and target to some kind of output function $out(\vec{x})$ will be called the *training* of the statistical learning method. In chapter 5 we will see that some methods do not require a training step: They derive the output directly from the given examples without building a model in advance. Figure 3.1 shows a scheme of the usual process of training and evaluation for a classification problem.

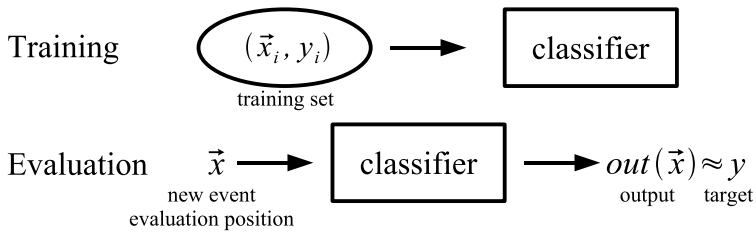

Figure 3.1: Supervised learning methods (here for classification) which build a model are trained resulting in a classifier which is then used to evaluate new events.

Example: Radial Basis Function Neural Networks

Radial Basis Function neural networks (figure 3.2) are usually trained in two steps. The first step consists of unsupervised clustering of the data. The centres and variances of the radial basis functions are determined, for example, with a Gaussian mixture model (see [31]).

The second step consists of supervised learning for which the found clusters remain fixed and only the influence of each cluster on the output is determined. Since only the weights of all basis functions in the final sum have to be determined this can be done by a simple multi-linear regression.

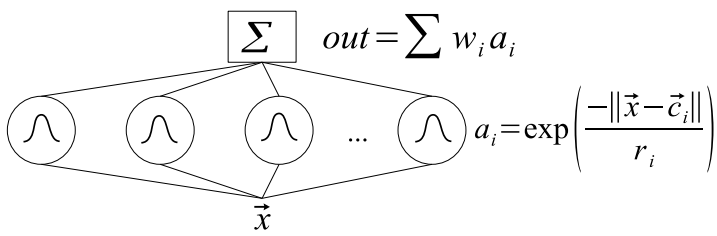

Figure 3.2: Structure of a Radial Basis Function Neural Network: The activation a_i of each *hidden* neuron depends on the distance of the input event to the centre c_i and on the radius r_i. The output is the (positively or negatively) weighted sum of all activations.

3.4 Classification vs. Regression

Depending on the given problem the datatype of the target y can be either dichotomic, for example 1 for the signal class and 0 for the competing background, or continuous, for example in the range $[0, 100]$ referring to the energy range 0-100 GeV.

In the first case we speak of *classification*: The statistical learning method should be able to distinguish between (at least) two classes. Problems with more than two classes are also quite frequent. Some examples where one statistical learning method handles many different classes can be found in literature: The digit database (images showing one of the digits 0-9) of the United States National Institute of Science and Technology is for example discussed in [31]. Nonetheless we will restrict to the case of two classes because the decomposition of a multi-class problem into different two-class problems is easy and often leads to even better performance. Hsu [39] states that the decomposition into "one vs. one" problems ($\frac{k(k-1)}{2}$ classifiers for k classes) performs better than the decomposition into one vs. others problems (k classifiers). The goal for a classification problem is for example given by the minimisation of misclassifications as discussed in section 3.12.

In the second case we speak of *regression*: Here the task is to estimate the value of some quantity of interest. For example an energy could not be measured directly but is hidden in the correlation of geometric quantities of each event. The typical goal for a regression problem is the minimisation of the squared error as discussed in section 3.12.

Example: Classification

A very simple classification problem in two dimensions is shown in figure 3.3. The squares should be separated from the circles – a possible solution (without any misclassification) is shown by the curved *decision boundary*. Any new event given by two coordinates would be classified according to the two sides of this decision boundary.

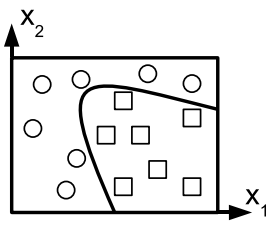

Figure 3.3: A simple two-dimensional example for a classification problem, the circles symbolise the "signal", events with $y = 1$, the squares stand for the "background", events with $y = 0$.

Example: Regression

A one-dimensional regression problem is shown in figure 3.4. The seven crosses represent the data points ("examples") and the smooth curve may be a solution formed by a statistical learning method. Any new event given by an x-coordinate will result in a y-coordinate output according to the learned curve.

3.5 Online vs. Offline

The concept of a trigger is well known in physics experiments. It selects events which should be recorded and skips others (either background or uninteresting events) to reduce the total

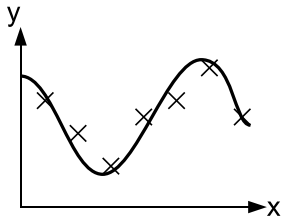

Figure 3.4: A simple one-dimensional example for a regression problem.

rate which must be logged onto mass storage. An online classification is nothing else than a trigger where "classification" reminds us of the fact that the trigger knows only two basic event classes: "good" ones should be kept and "bad" ones should be rejected. "Online" reminds us that the decision must usually be available within a very short timescale (μs to ms).

Offline classification mostly has to do with *purification* which means that a dominant background should be suppressed to make a weak signal visible. All this takes place when the data is stored on mass storage and the time restrictions for the classification are less demanding.

Example: Online vs. Offline – Trigger vs. Purification at the H1 Experiment

The trigger system of the H1 experiment including the neural network trigger was discussed in section 2.1. Within less than $20\mu s$ information from the different level one subsystems is received, the data is preprocessed to form useful inputs, and the decision is made by calculating the outputs of 13 feed forward neural networks in parallel. The trigger decision is very closely related to the hardware of the detector since the fast decision allows only minimal abstraction from the specific detector components.

Things are completely different for an offline analyses: Variables which describe the event after the full reconstruction was done have a much higher level of abstraction. Very specific quantities which describe the kinematics of the interaction can then be taken as inputs for statistical learning methods.

3.6 Preprocessing

Preprocessing data means transforming raw inputs $\vec{x}$ which are directly measured by the detector into new inputs $\vec{x'}$ which are better suited to describe the event in one or more of the following senses:

- The transformed inputs may make use of prior knowledge as we will discuss in section 3.9. Usually such a transformation leads to quantities which describe the event on a higher level and are better suited to distinguish different classes (classification) or different values (regression).

- The transformation may reflect a certain symmetry that is inherent to all events. In such a case resolving the symmetry by a rotation, mirroring, normalisation[1] or other

[1] A normalisation of each input to mean value 0 and standard deviation 1 can already simplify the learning task a lot.

transformation like a Fourier analysis frees the statistical learning method of discovering this property on its own. The learning method is usually capable of discovering such kind of symmetries. Nevertheless, typically an increased performance or shorter training times can be achieved if the job is made easier by preprocessing.

- If the input space is very high-dimensional and it is unknown how to reduce the dimensionality, a transformation based on automatic procedures can be very helpful. Such transformations are, for example, clustering (see for example [38, 31]) or principal component analysis (see for example [40]). These methods reduce the dimensionality of the problem while trying to maintain all the essential characteristics.

Like the search for the best raw inputs, also the search among different preprocessing techniques requires some studies if one wants to be sure that optimal inputs for the statistical learning method have been found.

Example: Symmetry – Charge Distribution in a Pixel Detector
A pixel detector which may be used aboard the XEUS satellite was introduced in section 2.5. An event usually consists of a charge cloud that is distributed over up to 2×2 pixels as shown in figure 3.5. To treat such a structure optimally (given a very homogeneous detector) one can swap left, right, up and down without changing the event topology. Resolving this symmetry, for example by setting the maximum charge always to the lower left corner, will help the learning method.

The fact that the four pixel signals sum up to the total charge generated by the photon can also be regarded as a symmetry. A normalisation to the total charge event by event would resolve this symmetry.

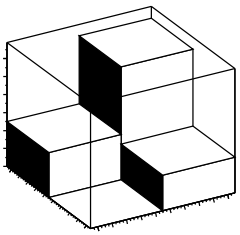

Figure 3.5: The charge generated in the pixel-detector by an X-ray photon is usually distributed over up to 2×2 pixels.

3.7 Standard Approach to Classification: "Cuts"

"Cuts" are on the one hand only a synonym for classification because a cut does nothing else than separate one side from the other, one class from another. On the other hand the word "cuts" is very strongly connected to a sequence of conditions applied to one-dimensional histograms which present the distribution of events depending on different quantities.

This sequence consists of *univariate* classifications whereas one *multivariate* classification is – if possible – always the better decision. The key point is the correlation between the different quantities in which the one-dimensional cuts are done. Projections of the multidimensional distribution onto the various coordinate axes ignore any correlations between the coordinates. Figure 3.6 shows a special example where a simple line – a correlated cut in two dimensions – performs much better than one-dimensional cuts.

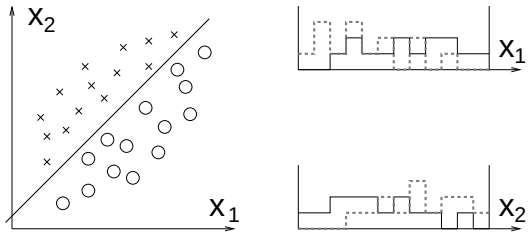

Figure 3.6: In this example the two-dimensional cut performs much better than any sequence of one-dimensional cuts. Almost no sensible cut could be found in the two projections shown on the right. There the crosses (×) are shown as a dotted line and the circles (∘) have a solid line.

In terms of available inputs x_i for two classes "good" ($y = 1$) vs. "bad" ($y = 0$) the standard approach first tries to find a value for x_1 which separates best good from bad events. Events of both classes which "survived" the first cut are filled into a histogram depending on x_2 , a new cut is defined there and so on until all available inputs have been used (see figure 3.7).

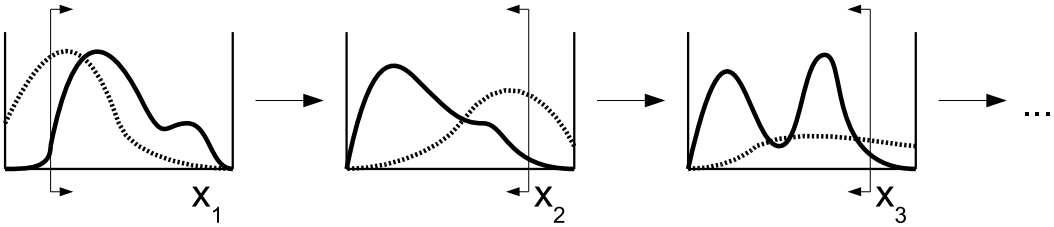

Figure 3.7: Sequence of cuts to reduce background (dotted) while keeping as much signal (solid) as possible.

A correlated cut, meaning a decision depending on all inputs x_i simultaneously, is surely a better decision if one is able to construct it. Specific knowledge about the multidimensional dependences of all inputs are necessary to build such a classifier by hand. As will be discussed in section 3.9, ignorance of these dependences naturally leads to the application of statistical learning methods because they are using correlated cuts and they learn the multidimensional dependences from examples.

Example: Enrichment of Instanton Events

Figure 3.8 shows in three rows three steps of a cutting procedure. The distributions of signal and background for two inputs (virtuality ${Q'_r}^2$ and band multiplicity n_B) of the instanton dataset (introduced in section 2.1.6) are plotted for each step.

The original distributions are plotted in the first row. Since no cut has been performed at this point all events are taken. Black histograms mean "correctly identified" while "misclassified" events are plotted with grey histograms. Because all events are taken at this point all signal events are in a black histogram and all background events in a grey one. In addition to the histograms the first cut which will be done is indicated in the ${Q'_r}^2$ histograms.

In the second row the cut in ${Q'_r}^2$ was done. We see that signal events are misclassified for too low ${Q'_r}^2$ and background is still misclassified for high ${Q'_r}^2$. In the n_B distributions the effect of the cut in ${Q'_r}^2$ can be seen. Especially in the background

distribution for n_B it becomes clear that ${Q'_r}^2$ and n_B are not independent because the shape of the background that is still left over (grey) differs significantly from the shape of the background that was cut out (black). In addition to the histograms the second cut which will be applied is indicated in the n_B histograms.

The third row finally shows the resulting distributions after both cuts have been done. Since the second cut had a quite high rejection rate and low efficiency we see that most of the background is correctly recognised (black) while a still significant amount of signal remains (also black).

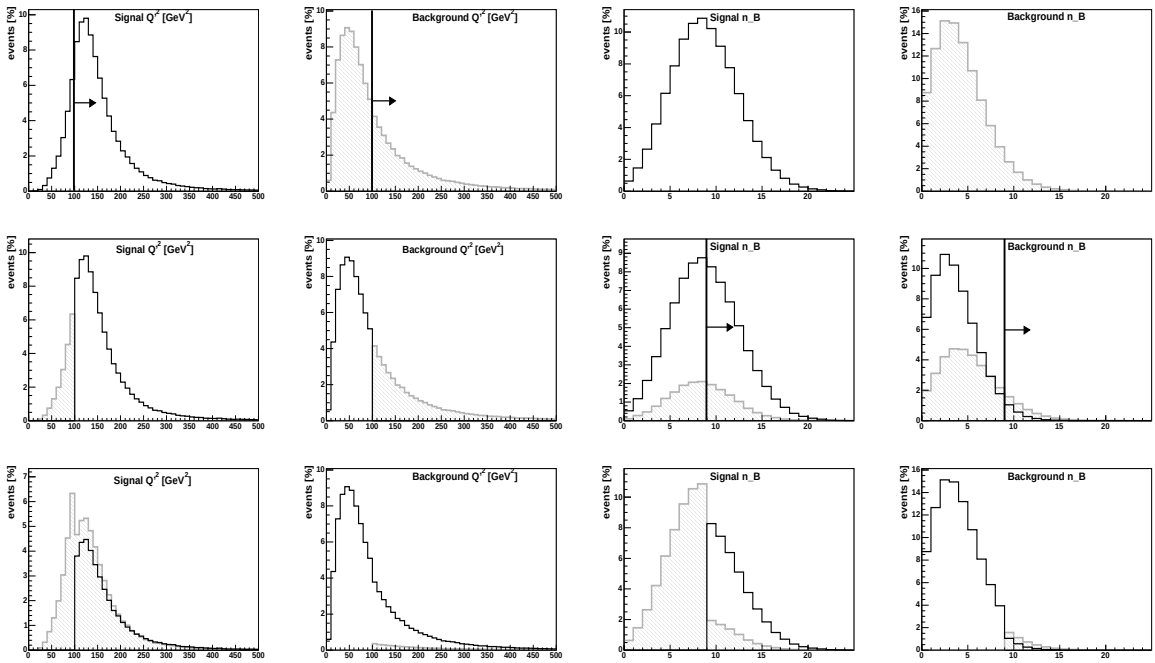

Figure 3.8: A three step cut process with the instanton dataset, two quantities are used here: ${Q'_r}^2$ is plotted in GeV^2 and n_B is the number of tracks. The three rows are explained in the example.

3.8 Standard Approach to Regression: "Fit"

Motivated by theoretical predictions or by observed distributions, a certain functional dependence is mostly the basis to cope with a regression problem. This functional dependence is usually not fully understood and may contain many free parameters whose values have to be determined by a fitting procedure to describe the observed data correctly. Once this is done a formula has been obtained that can predict for a certain input-vector $\vec{x}$ the corresponding target value y. For example a sinus function with a certain y-offset, wavelength, phase and amplitude was fitted to the data shown in figure 3.9.

The main problem of this approach is the necessity that some functional dependence must be known. The fitting cannot be done if either no functional dependence is known or the theoretically predicted relations do not describe the measured data. Besides, the

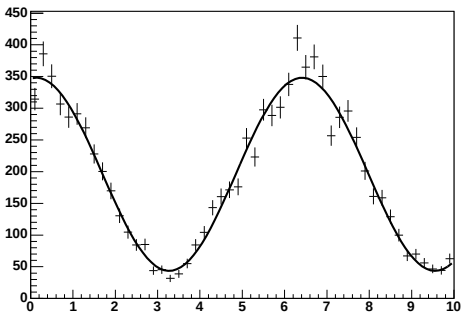

Figure 3.9: A simple one-dimensional example showing a histogram with errors and a fitted sinus function.

known dependence may only include some but not all of the quantities x_i which would be available for discrimination.

In each of these and similar cases it is again ignorance that leads to the application of statistical learning methods as will be discussed in section 3.9 because they will extract the needed functional dependence from the examples given to them.

Example: Energy Estimation in the MAGIC Detector
The Hillas parameters describe the shape of the Cherenkov ellipse in the telescope (see section 2.4). These quantities can be used to determine the energy of the primary photon in a simple linear model:

$$E_\gamma = a + b \cdot \texttt{size} + c \cdot \texttt{size}^2 + d \cdot \texttt{dist} + e \cdot \texttt{length} + f \cdot \texttt{width} + g \cdot \frac{\texttt{width}}{\texttt{length}}.$$

The parameters a to g are determined by minimising the sum of squared relative errors $\left(\frac{E_{est}-E_{true}}{E_{true}}\right)^2$ where the sum runs over many events of different energies. The following table shows the values for the parameters which have been obtained:

parameter	a	b	c	d	e	f	g
value	38.7	0.120	$-4.38 \cdot 10^{-6}$	-0.244	0.621	-1.90	21.1

We choose an event like the one shown in figure 3.10 with a true energy of $269GeV$. Plugging in its Hillas quantities

input	`size`	`dist`	`length`	`width`
value	1352	205	52.0	17.5

leads to an estimated energy of $170GeV$, much lower than the true energy of $269GeV$, but still within the average relative error which is usually around 30%.

3.9 Knowledge and Time

As mentioned above, the amount of knowledge about the dataset which should be analysed is the most important criteria whether a multidimensional cut or function can be constructed that will meet the requirements in terms of both the description of the data and the performance of the resulting classification/regression. If this is not the case, statistical learning methods are for sure worth considering.

This means that statistical learning methods can always only be the second choice. Either one has nothing at hand or one has only some method which does not perform as

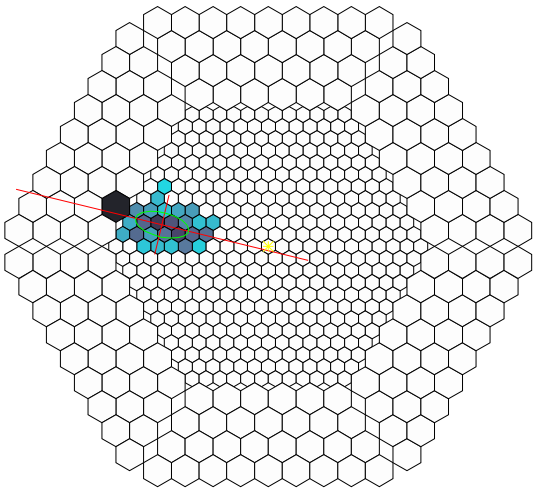

Figure 3.10: The Cherenkov light as projected into the camera of the MAGIC telescope has an ellipsoidal shape for high energies – here about $270GeV$. The energy estimation may be performed with the parameters of the Cherenkov ellipse.

well as wished. Then statistical learning offers the possibility to leave the development of a good classifier/regression to the computer.

In section 3.6 it was discussed how knowledge that is existing (but is not sufficient to completely analyse the data) can be used in statistical learning methods by preprocessing the data. This offers the possibility to make knowledge from the theoretical background or experimental observation directly available to the learning method and may improve the performance drastically.

Example: Prior Knowledge – The Cherenkov Ellipse in the MAGIC Telescope

Section 2.4 described the MAGIC Telescope and how the Cherenkov light forms an ellipsoidal image in the camera. The Cherenkov light is emitted by a shower which was induced by a high energy photon. For energies below $50GeV$ the ellipsoidal shape might not be recognised. But for energies around $1TeV$ the high-level Hillas parameters, which result from the analysis of the ellipsoidal shape, will provide statistical learning methods most probably with better information than raw pixel values (compare figure 3.10).

There is only one reason to apply statistical learning to a problem for which a very good analysis based on prior knowledge exists: time consumption. If the existing analysis is a time consuming offline analysis and a faster decision is needed, for example, to create a trigger, then the well performing offline analysis can act as a teacher for training a fast statistical learning method. Of course, the statistical learning method will not outperform but only approximate the decision of the offline analysis. Nevertheless a very similar classifier can be obtained which is many orders of magnitude faster.

Usually experiments from high energy or astrophysics have some kind of trigger which applies a number of thresholds. An event gets triggered if the thresholds are exceeded. If the thresholds should then be lowered to obtain more events, or to obtain events from a lower energy region, then this trigger system gets into trouble: The trigger rate becomes much too high. Especially in such a case a trigger based on statistical learning methods is worth considering.

Example: Fast Trigger Decision with the Neural Network Trigger at the H1 Experiment

The neural network trigger (see section 2.1.4) was already presented as an example for a statistical learning method working online. It was compared to an offline analysis for the same experiment where high level quantities from the complete reconstruction of the event can be used. These abstract event descriptors are not available for triggering but they can be used as the teaching information for the training of the trigger. Therefore the neural network will approximate the correct decision (which would be made after full reconstruction) on the second trigger level with only the quantities which are available at this point.

3.10 Prerequisite: Training Data

The most important demand of any statistical learning method is, of course, the training data. As discussed in section 3.3, supervised learning requires that examples are available which consist of inputs and their associated target values. The correct target values represent the information coming from the teacher. If such kind of teacher or correctly assigned target values are missing completely then there is no chance of applying a supervised statistical learning method.

However, in practice there are several ways to obtain the needed training examples. A simple example was discussed in the last section where a time consuming offline analysis acts as a teacher for a statistical learning method which will then perform similarly but much faster.

Monte Carlo simulations are also a standard way of generating training examples. But they must be used with care: Both underlying physics and the detector response have to be understood very well to create a simulation which generates events matching the experimental observations. Even very small deviations, "features" – correlations or deviations that exist in the simulation but not in reality (vice versa may be less important) – may result in a trained method that handles simulated events perfectly, but shows a behaviour like random guessing on real data.

Example: Monte Carlo Simulations in the Search for Instantons at H1

Non perturbative QCD predicts a very small number of instanton-induced events which could be detected in the H1 experiment. As discussed in section 2.1.6, two different Monte Carlo simulations, MEPS and CDM, for the standard perturbative QCD event classes can be used. To enrich the number of instanton-like events in the measured dataset only a small fraction of the events from the standard perturbative QCD event classes is selected.

Independent of the used classification method discrepancies between these two simulations are revealed, since the number of selected events (for a fixed instanton efficiency) after the purification differs significantly. Of course, the differences become even larger when the number of inputs for the purification is increased: The more inputs are available for discriminating instantons from perturbative QCD event classes, the more special the selected region in phase space will be.

If different simulations lead to contradictory results no final conclusion can be drawn. Therefore one or more reliable and consistent simulations are really essential if there

are no other ways to generate training data.

Finally, training data can be generated by temporarily modifying the experiment. This means that the correct classification or value of the regression is fixed and known after changing the experimental setup. To obtain examples for all classes of a classification or for all values of a regression problem the experiment may have to be modified several times. It is important that the modification only controls the target (class or value) and does not change the detector response. When enough examples have been collected, the experiment can be run in "normal" mode which then means that any class or value for a regression may appear in the events. The statistical learning method can then make its decision based on the examples from the modified versions.

Example: Modifying a Pixel-detector – The Mesh-Experiment
The pixel-detector which will be part of the XEUS satellite was presented in section 2.5. To improve the position resolution of this pixel-detector examples for the charge distribution, depending on the incident position of the photon, are needed. To obtain events for which the incident position is known, a mask with small holes in a regular rectangular pattern is placed slightly rotated in front of the detector as was described in section 2.5.1. The small angle between the pixel structure and the so-called mesh places the holes over different parts of the underlying pixels as shown in figure 3.11. Since the photons will only come through the holes, their incident positions are known. From different pixels we get different positions covering the whole pixel area.

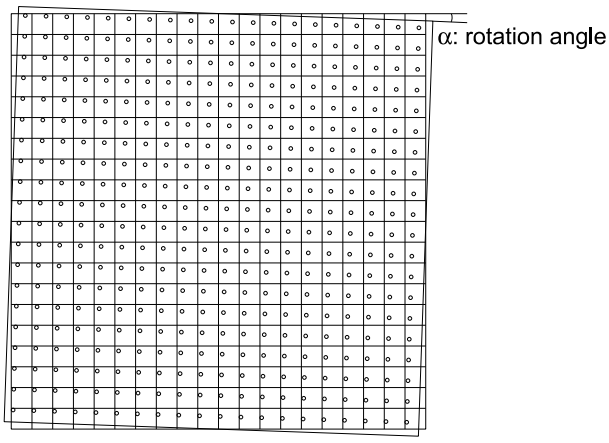

Figure 3.11: The mesh-experiment: A small rotation of the mesh structure with respect to the pixel structure places the holes over different parts of the individual pixels. Assuming that all pixels behave identically, training data can be generated that covers incident positions for the whole area of a pixel.

3.11 Overtraining and Regularisation

Controlling a statistical learning method is quite similar to controlling any analysis: One has to make sure that it does what it should do, which means that one checks whether

everything what one puts into the method is correct, checks the output of the method for a few examples "by hand", compares the behaviour of simulation with real data, compares the results with theoretical predictions and so on.

What is peculiar to controlling statistical learning methods originates from the "statistics part": Only a certain – potentially very low – number of examples is available which can be used to teach a – potentially very complex – target function to the learning method. The obviously ill-defined attempt to learn a very complex function from only a few examples will result in a behaviour called *overtraining*.

Overtraining means that the function space which was searched by the statistical learning method was too large compared to the low number of examples which could define such a function. The result is a method which performs perfectly on all the training examples because they were learned by heart. But no *generalisation* was done and thus the performance for new events will be generally bad.

Example: Overtraining like in Function Fitting

Figure 3.12 shows a fitting example in which in the left picture a function with too many adjustable parameters was chosen. This leads to a perfect fit, but without any generalisation. In the right picture the number of free parameters is appropriate. The number of free parameters and thus the size of the function space which is searched shows here the same influence like for a statistical learning method.

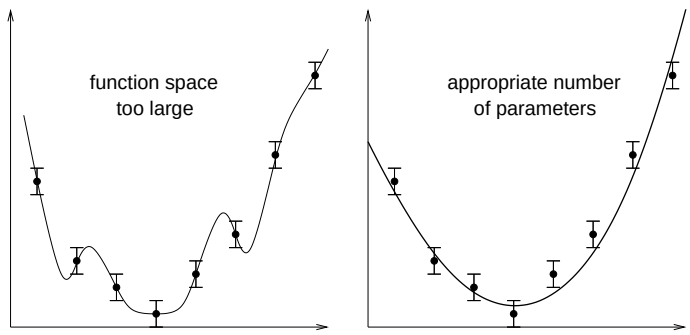

Figure 3.12: Like in function-fitting overtraining is due to too many free parameters. *Regularisation* reduces the searched function space by restricting the parameters. The training examples are fitted no longer exactly after regularisation but the generalisation is much better.

A very basic concept to detect overtraining is the division of the available examples into (at least) two groups which will be called the *training set* and the *test set*. By using only the examples from the training set to train the method, the application to the test set can act as an independent evaluation of how well the learning method performs. Overtraining can then be detected directly as a large disagreement between the good performance on the training set and the bad performance on the test set.

But the detection of overtraining is not enough, we need some handle to control this behaviour: *Regularisation* means that the size and type of function space which should be searched by the statistical learning method can be controlled (penalisation of complexity) by the adjustment of certain parameters of the method. These parameters depend on the method and will be discussed for each method in chapter 5.

By adjusting the regularisation parameters the optimal type of function space can be searched, for which the contradicting demands of precision on the training set and generalisation for the test set lead to an optimal behaviour. But this introduces a new problem: A new bias is introduced if one of the many parameter sets which have been tried out is selected by looking for the best performance on the test set. The behaviour on the test set is then no longer an independent measure of the performance of the method. Therefore we finally need to divide the available examples into three groups: training set, test set and *selection set* which can be used to select the best performance among different parameter sets.

In summary the division into training and test set is enough for the training of only one classifier (with the training set) and the independent measurement of its performance (with the test set). But if the optimal performance should be chosen among many different trainings then the division into three sets is necessary. The training set is used as before only to train the different classifiers. The selection set is then used in between to choose the optimal performance among the different trainings. Finally the test set is also used as before to have an independent measurement of the true performance of the selected classifier/regression model.

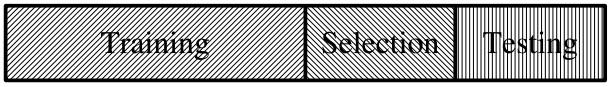

Figure 3.13: Usually the available examples should be split up into three sets: the training set is used to train the statistical learning method, the selection set is used to choose the optimal generalisation behaviour among different parameter settings, and the test set provides an independent estimate for the true performance. The splitting could typically be 50%:25%:25%.

We have seen that model selection (by choosing one of the parameter sets) and model assessment (estimating the true performance) are two independent steps which can both be done with the help of estimates from held-back events. How the selection and test set are used to estimate the true performance will be discussed in the next section. However it is important to note that there are also several theoretical frameworks coping with the tasks of model selection and model assessment that do not require a selection or test set. A short introduction to theoretical aspects will be given in chapter 4.

3.12 Performance Evaluation

Since the training of statistical learning methods depends on the method itself we will discuss in chapter 5 how the different methods work and how in each case the learning of a classification or regression problem is done. What is nearly independent of the learning method is the performance evaluation.

As discussed in the last section, overtraining may lead to a very good performance on the training set but to a very bad performance on the test set. This means that the training set should never be used for the measurement of the true performance (on new events). But there exist methods by which the whole dataset can be used for training AND evaluation in very specific ways.

Three different methods are commonly used to estimate the performance:

- *Bootstrapping* creates k different training sets of the same size as the original training set by drawing with replacement from the original training set. In the limit of infinitely large training sets each of the k new sets consists of about 63% of the events in the original set, the rest are replications (compare the *Bagging* procedure in section 5.5.2). The learning method is then trained for each of the k newly created training sets resulting in k slightly different classifiers[2]. Each of the training events can now be used to evaluate the performance by processing it with exactly those l classifiers for which it did not appear in the training set. The output for each event is thus an average over the outputs of l classifiers. This average output is unbiased since each event is only processed with those l of the k classifiers for which it did not appear in the training set. Since averages over many training results are used bootstrapping tends to overestimate the performance.

- *Cross-Validation* splits the training set into k subsets (the case where k equals the number of training events is called *Leave-One-Out*). k different training sets are then created by leaving out each part once and the learning method is trained k times resulting in k slightly different classifiers. Each of the training events can now be used to evaluate the performance by processing it with that classifier for which it did not appear in the training set. This output is taken to measure the performance.

- *Train-Test* was already presented as a way to discover overtraining. About half of the events are separated from the training set, never presented to the learning method and only used to measure the performance independently. Since the test set blocks about 50% of the available events from being used in the training this method tends to underestimate the performance compared to a training which could use the whole dataset.

The problematic point about bootstrapping and cross-validation is that they do not give a performance estimate of a specific trained classifier or regression method but about the general behaviour of the learning method (using many different trainings). If we want to measure the performance of a specific training result the train-(selection)-test procedure has to be used which will be done throughout this thesis[3].

All that is needed to measure the performance is the output of the learning method for all examples of the test set. From this the performance can be determined depending on the type of application.

3.12.1 Performance Evaluation for Classification

For the performance evaluation of a classifier, histograms of the output distributions for signal and background form the basis (compare figure 3.14). If the weights are not all equal the events are filled with their associated weights into the histograms. Well designed classification methods do not only give outputs 0 or 1 (representing the guess for the class an event belongs to). But they give continuous values in the interval $[0, 1]$ which could be interpreted as a probability. A value of 0.5 then means that this event could be either class with almost the same probability.

[2] When we speak in this section about a classifier it can as well be a regression model.

[3] Concerning the difference of comparing learning methods vs. comparing specific hypotheses see also the discussion in section 3.15

Example: Output Histograms

Figure 3.14 shows an example of output distributions for signal and background. The output for the signal class should peak at 1 while the output for the background class should peak at 0. However, one notices that for a typical dataset from physics analysis there are background events which look almost exactly like signal events (their output is around 1) and there are also signal events which look like background (their output is around 0).

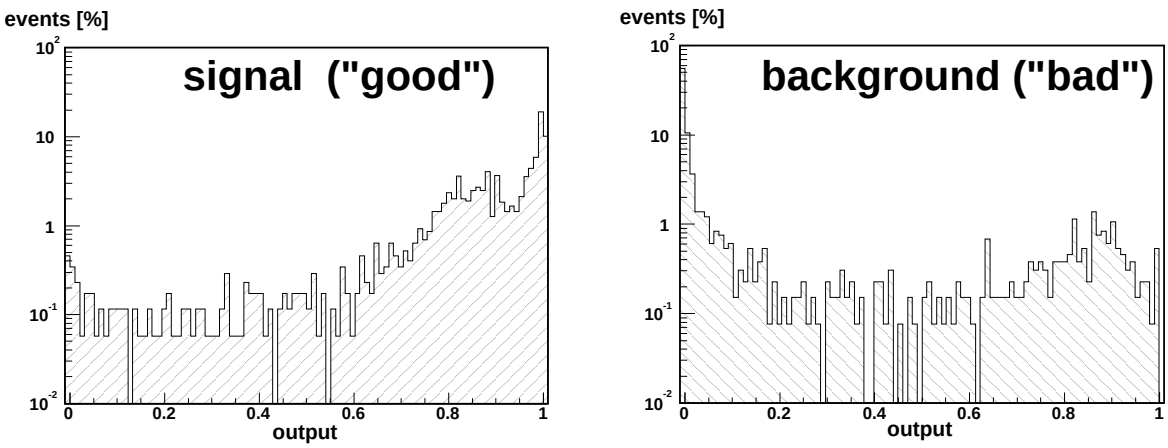

Figure 3.14: An example of output distributions for signal and background on a logarithmic scale.

The great advantage of a continuous output between 0 and 1 shows up when a cut is defined to do the actual classification. The *signal efficiency* ϵ is then given by the percentage of recognised (output>cut) "good" events and the *background rejection* r is given by the percentage of recognised (output<cut) "bad" events. If the outputs are either 0 or 1 the classifier can only be used in one single way.

If the output is distributed in $[0, 1]$ any cut in between defines its own efficiency and rejection. All signal events with an output above the cut and all background events with an output below the cut are correctly classified. Naturally the graph in the efficiency/rejection plane which results from choosing different cut values starts with no rejection and 100% efficiency for $cut = 0$ and ends at 100% rejection and no efficiency for $cut = 1$ (compare figure 3.15). Random guessing or *pre-scaling* would mean that in between $\epsilon + r = 1$, a straight line between the two endpoints (see figure 3.15). Good classifiers try to reach the upper right corner with full efficiency and full rejection.

The free choice of the efficiency/rejection pair makes it also easy to account for weights per class or *cost matrices*, i.e. the relative weighting of an accepted background event to a rejected signal event. While weights per event are respected as mentioned above, the costs for misclassifying signal as background and vice versa are respected by choosing an appropriate pair of efficiency and rejection[4].

[4]Simple algorithms which return only either 0 or 1 have to incorporate these cost matrices in advance because they fix efficiency and rejection by their output.

Example: Efficiency vs. Rejection Graph

Choosing a cut in the output distributions of figure 3.14 results in the efficiency vs. rejection graph shown in figure 3.15. Because of the tails in both output distributions the optimal point of 100% efficiency and 100% rejection cannot be reached. This is the typical case in datasets from physics experiments as they are usually not completely separable: The signal and background classes overlap. Nevertheless, with only a loss of 3% efficiency a rejection of 80% is achieved in the example. The corresponding cut in the output distributions has a value of 0.23. The separation power of 4.9 is calculated as the quotient of signal and background efficiency ($SP = \frac{\epsilon}{1-r}$).

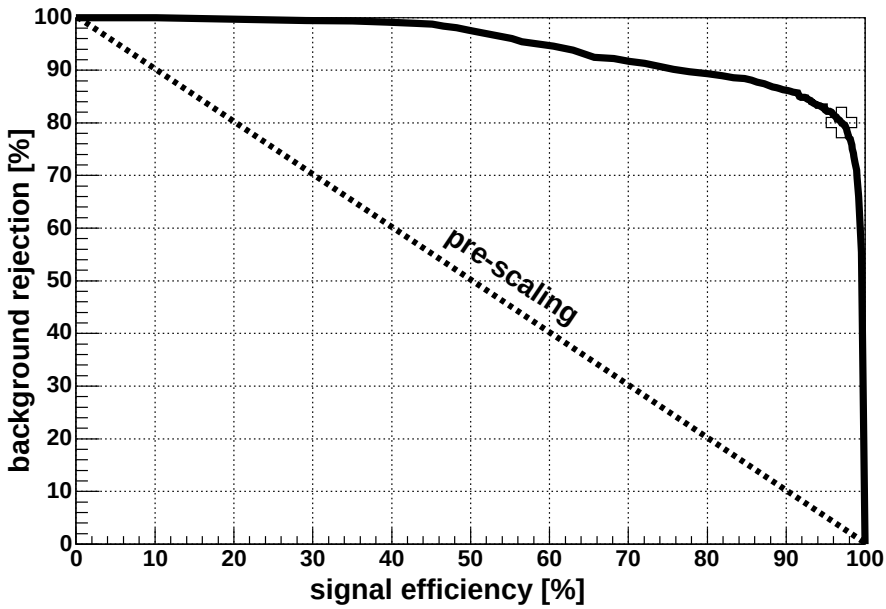

Figure 3.15: The efficiency vs. rejection graph which results from the output distributions shown above – at the marked point an efficiency of 97% is achieved with a cut at 0.23 and results in a background rejection of 80%. The separation power is 4.9.

To finally evaluate a classifier or to compare different classifiers among each other the curves may be compared as a whole or, depending on the application, the most interesting numbers may be extracted like

- the minimum percentage of misclassifications: $\min(1 - \epsilon + 1 - r)$
- the efficiency at a rejection of a predefined percentage
- the rejection at an efficiency of a predefined percentage

Furthermore the separation power as defined above can be used to compare classifiers, where usually a minimum efficiency is required. Each of these criterions fixes a specific cut and sets thus a "working point" for the classifier.

The performance of a classifier might also be measured with a more complicated analysis involving pseudo experiments or calculating a significance. These methods choose a cut by taking into account the signal over background ratio as well as the efficiency itself which scales the significance with $\sqrt{N_{\text{sel}}}$. This is the statistical uncertainty of the number of selected signal events (which passed the cut).

Example: Different Demands to Classifiers
For the H1 Neural Network Trigger (see section 2.1) usually a certain rate reduction is required. The efficiency for a rate reduction of, for example, a factor 20 is determined by a rejection of 95%.

For other applications like for the rejection of pileups in a pixel-detector (see section 2.5) a very high efficiency may be important. Then the rejection with, for example, 99% efficiency is taken as the value to measure the performance.

3.12.2 Performance Evaluation for Regression

To evaluate the solution to a regression problem, different loss functions can be defined which sum up the differences between target value y_i and actual output $out(\vec{x_i})$ for all examples in the test set.

The most commonly used one is the squared error loss function which can sum up absolute or relative errors:

$$E^2_{abs} = \frac{1}{N}\sum_{i=1}^{N}\left(out(\vec{x_i}) - y_i\right)^2 \qquad E^2_{rel} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{out(\vec{x_i}) - y_i}{y_i}\right)^2 \tag{3.2}$$

If the weights are not all equal the squared errors should be weighted by the weight of the event and a normalisation factor $1/\sum w_i$ may be needed.

To visualise the performance, a correlation plot between target y and output $out(\vec{x})$ is helpful (figure 3.16 left) as well as a plot of the absolute or relative error distributions ($out(\vec{x}) - y$ or $\frac{out(\vec{x})-y}{y}$) in different bins of the target value y (figure 3.16 right). To obtain a total error per bin from these distributions one has to add up the squared mean (offset/bias) and the variance[5] ($\langle X \rangle$ denotes the mean value of X):

$$\begin{aligned} E^2_{abs} &= \langle out(\vec{x}) - y\rangle^2 + \mathrm{Var}\left(out(\vec{x}) - y\right) \\ &= \langle out(\vec{x}) - y\rangle^2 + \langle \left(out(\vec{x}) - y\right)^2\rangle - \langle out(\vec{x}) - y\rangle^2 \\ &= \langle \left(out(\vec{x}) - y\right)^2\rangle \\ &= \frac{1}{N}\sum_{i=1}^{N}\left(out(\vec{x_i}) - y_i\right)^2 \end{aligned} \tag{3.3}$$

which matches the first error definition. Often these total errors per bin in the target value y are used as a measure of the performance of the statistical learning method.

3.13 Calculation of Uncertainties for Statistical Learning Methods

How to calculate uncertainties for statistical learning methods is a very important subject for physics analysis. Therefore this question is very frequently posed by physicists who would like to apply this kind of methods but are unsure whether uncertainties can be obtained in a physically correct and uncomplicated way. Unfortunately a broad confusion

[5] The term *Var* is used here for the biased estimate of the variance. For $N \ll 1000$ the unbiased estimate $\mathrm{Var}(X) = \frac{1}{N-1}\sum_{i=1}^{N}\left(X_i - \langle X\rangle\right)^2$ should be used.

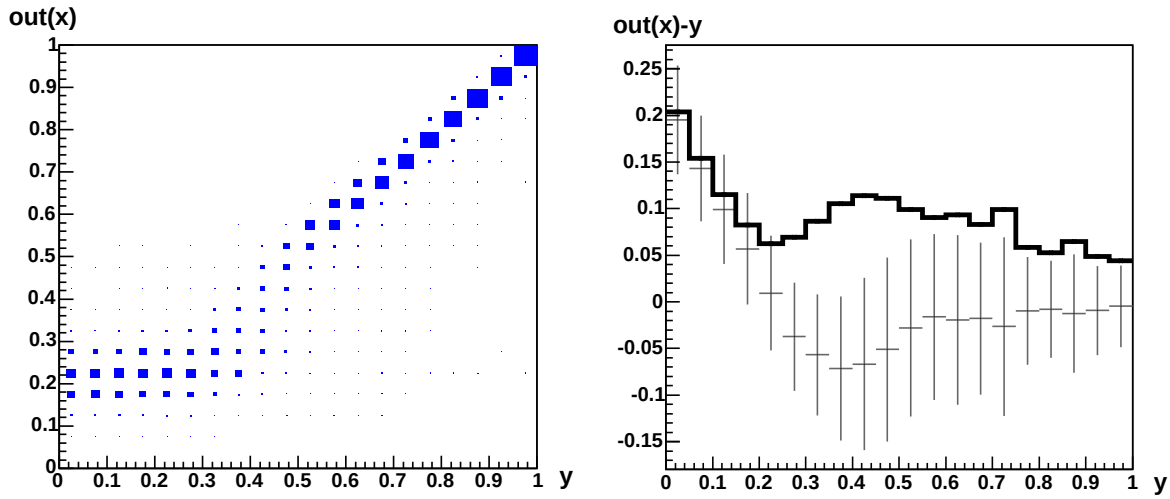

Figure 3.16: For 20 bins of the target $y \in [0, 1]$: Left the correlation between output and target is shown while right the crosses show the mean and variance of the error $out(\vec{x}) - y$. The bold line shows the total error (squared mean and variance added).

about this subject has led to a general fear that with the application of such methods uncertainties are no more under control.

It is important to note that there are very clear straightforward methods to calculate both statistical and systematic uncertainties for any statistical learning method.

3.13.1 Statistical Uncertainties

In this section different methods to calculate the statistical uncertainty of an efficiency (and in the same way of a rejection) will be reviewed and compared. For a small number of events in the test set as well as for very high or very low efficiencies the different methods might give different results while they should agree for large statistics and medium efficiency.

A very rough guess for the statistical uncertainty of the efficiency could be derived from the counting error of the total number of events. The absolute uncertainty of the number of selected events would then be $\sqrt{N}$ where N is the number of signal events for efficiency and the number of background events for rejection. This formula is very conservative and naturally shows a problematic behaviour for high or low efficiencies since then the 1σ interval may include efficiencies below 0% or above 100%.

1. A typical basis to calculate the uncertainty of the efficiency is to study the propagation of the counting errors for the numbers of events which pass or do not pass the selection. This propagation is described by

$$\sigma_\epsilon^2 = \sum \left(\frac{\partial \epsilon}{\partial x_i} \right)^2 \sigma_{x_i}^2 \tag{3.4}$$

where the efficiency ϵ depends on quantities x_i which have to be independent. Since the efficiency is the percentage of selected events

$$\epsilon = \frac{N_{sel}}{N_{sel} + N_{cut}} \tag{3.5}$$

its uncertainty can be calculated by plugging in the counting uncertainties of the independent numbers of cut and selected (passed) events

$$\begin{aligned}\sigma_\epsilon^2 &= \left(\frac{\partial\epsilon}{\partial N_{sel}}\right)^2 \sigma_{N_{sel}}^2 + \left(\frac{\partial\epsilon}{\partial N_{cut}}\right)^2 \sigma_{N_{cut}}^2 \\ &= \left(\frac{N_{cut}}{(N_{sel}+N_{cut})^2}\right)^2 N_{sel} + \left(\frac{-N_{sel}}{(N_{sel}+N_{cut})^2}\right)^2 N_{cut} \\ &= \frac{N_{sel}N_{cut}}{(N_{sel}+N_{cut})^3}. \end{aligned} \tag{3.6}$$

A formula for the same problem taking into account weighted events can be found, for example, in [41].

2. The efficiency and its uncertainty can also be derived from its probability density given the number of selected and total events. The binomial distribution describes the probability that a certain number N_{sel} of totally N_{tot} events are selected, given a fixed efficiency (for each event):

$$P(N_{sel}|N_{tot},\epsilon) = \binom{N_{tot}}{N_{sel}} \epsilon^{N_{sel}}(1-\epsilon)^{N_{cut}}. \tag{3.7}$$

The probability for the number of selected events given the efficiency $P(N_{sel}|N_{tot},\epsilon)$ is thus known. But we ask for the probability of a certain efficiency given the number of selected events $P(\epsilon|N_{sel},N_{tot})$. Bayes theorem transforms the first probability distribution into the second if we know the prior probability for the efficiencies. Due to ignorance we choose a flat distribution:

$$f(\epsilon)d\epsilon = d\epsilon. \tag{3.8}$$

Bayes theorem then gives

$$f(\epsilon|N_{sel},N_{tot})d\epsilon = \frac{P(N_{sel}|N_{tot},\epsilon)f(\epsilon)d\epsilon}{\int_0^1 P(N_{sel}|N_{tot},\epsilon')f(\epsilon')d\epsilon'} = \frac{\epsilon^{N_{sel}}(1-\epsilon)^{N_{cut}}}{\int_0^1 \epsilon'^{N_{sel}}(1-\epsilon')^{N_{cut}}d\epsilon'}d\epsilon. \tag{3.9}$$

The integral in the denominator can be calculated by successive partial integrations [42]. We obtain

$$f(\epsilon|N_{sel},N_{tot})d\epsilon = \frac{(N_{tot}+1)!}{N_{sel}!N_{cut}!}\epsilon^{N_{sel}}(1-\epsilon)^{N_{cut}}. \tag{3.10}$$

The expectation value of this probability density is an estimation of the true efficiency, its variance is an estimation of the variance of the efficiency:

$$\epsilon_{est} = \frac{N_{sel}+1}{N_{tot}+2} \tag{3.11}$$

$$\sigma_{est}^2 = \epsilon_{est}\left(\frac{N_{sel}+2}{N_{tot}+3} - \epsilon_{est}\right). \tag{3.12}$$

This estimate of the efficiency is biased towards medium efficiencies due to the prior assumption (flat prior probability for the efficiencies). For large N_{tot} and N_{sel} the bias vanishes and also the variance becomes identical to the result of the first method.

3. A third method to estimate the uncertainty of the efficiency is to use statistical tests to derive a 68% confidence interval. Such tests are based on the binomial distribution. To determine the lower bound of the confidence interval we look for an ϵ_0 which is just too low to generate N_{sel} from N_{tot}. We therefore have to decide for a given ϵ_0 between two hypotheses H_0 and H_1:

$$H_0 : \epsilon \geq \epsilon_0 \text{ vs. } H_1 : \epsilon < \epsilon_0. \tag{3.13}$$

H_0 has to be rejected if the probability for a lower number of selected events (than actually found) is too high:

$$\sum_{k=0}^{N_{sel}-1} B(k, N_{tot}, \epsilon_0) = \sum_{k=0}^{N_{sel}-1} \binom{N_{tot}}{k} \epsilon_0^k (1-\epsilon_0)^{N_{tot}-k} > 16\%. \tag{3.14}$$

For the upper bound of the confidence interval H_0 reads $\epsilon \leq \epsilon_0$ and has to be rejected if the integral of the binomial distribution from $N_{sel}+1$ upwards given ϵ_0 is greater than 16%. The 16% confidence levels are the remaining integrals left and right of the 68% (1σ) confidence interval. A binary search in ϵ_0 determines the values for the lower and upper boundary of the confidence interval at which the acceptance of H_0 changes into its rejection. This is the only method allowing to calculate asymmetric uncertainties.

Figure 3.17 gives an impression of how the three different methods to estimate the statistical uncertainty of the efficiency behave. For the third method the mean value of σ_{up} and σ_{down} is plotted. The unification of the three methods for high N_{tot} is clearly visible, the third method tends to give lower uncertainties for low statistics, the first two methods behave very similar, they even change their places between 70% and 90% efficiency. We will use the first method for all evaluations done in chapter 7.

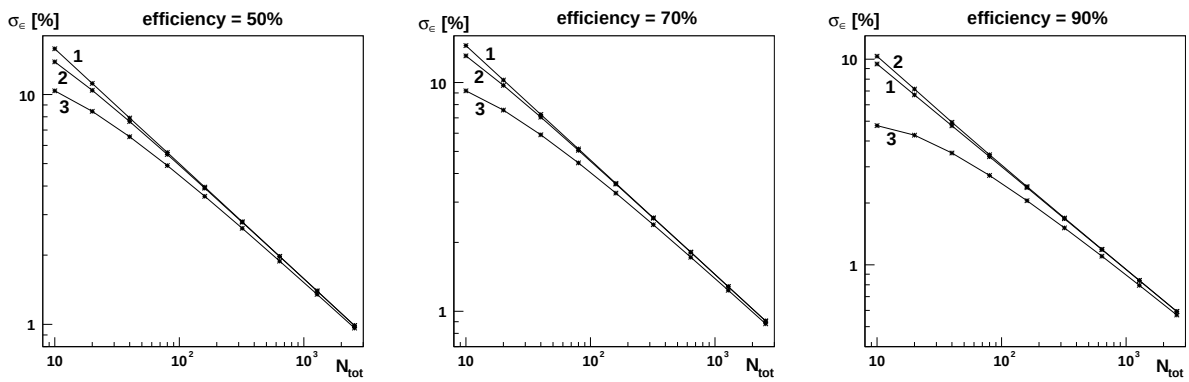

Figure 3.17: Comparison of the three estimates for the statistical uncertainty, shown are the absolute uncertainties of the efficiency (in %) for different numbers of events and different efficiencies.

For a regression problem it was shown in section 3.12.2 that the squared error loss function can be decomposed into offset and variance of which the uncertainties can be estimated. Defining the error per event $X = out(\vec{x}) - y$ we get an estimate of the mean

$\mu = \langle X \rangle$ and an estimate of the variance $S^2 = \frac{1}{n-1}\sum_{i=1}^{N}(X_i - \mu)^2$. The uncertainties of mean and variance are given by their variances

$$\mathrm{Var}(\mu) = \frac{S^2}{N} \tag{3.15}$$

$$\mathrm{Var}\left(S^2\right) = \frac{1}{N}\left(\mu_4 - \frac{N-3}{N-1}S^4\right) \tag{3.16}$$

where μ_4 is the fourth moment

$$\mu_4 = \langle (X-\mu)^4 \rangle = \langle X^4 \rangle - 4\mu\langle X^3 \rangle + 6\mu^2\langle X^2 \rangle - 3\mu^4. \tag{3.17}$$

The uncertainty of the squared error loss function is then calculated by summing up the uncertainties of mean and variance:

$$\mathrm{Var}(E^2_{abs}) = \mathrm{Var}\left(\mu^2\right) + \mathrm{Var}\left(S^2\right) \tag{3.18}$$

$$\approx 4\mu^2\mathrm{Var}\left(\mu\right) + 2\mathrm{Var}^2\left(\mu\right) + \mathrm{Var}\left(S^2\right). \tag{3.19}$$

The last approximation was derived with simulations. It transforms the variance of the squared mean into the variance of the mean which is known.

Example: Statistical Uncertainties
Assuming we determine efficiency and rejection of a trained classifier with a test set of (only) 300 examples – 100 signal and 200 background events. If 80 of the 100 signal events pass the cut and 180 of the 200 background events are rejected by the cut then the efficiency is (80±4.0)% and the rejection is (90±2.1)% (first method used).

3.13.2 Systematic Uncertainties

Unfortunately the opinion that statistical learning methods have an uncertainty in themselves which needs to be added somehow to the total uncertainty is met quite frequently. There exist ideas like varying the behaviour of a classifier by small amounts and observing the variation of the outputs (e.g. by varying the weights of a neural network). But what will that tell us? This only tells us that indeed the output of a classifier depends on the values which represent the learned hypothesis.

An simple way to derive the true systematic uncertainties is to think of a trained classifier which solves a classification problem by applying a cut in its output distribution. That is all it is. Like any well known one-dimensional cut this multidimensional cut does nothing else but propagate the systematic uncertainties of its inputs to the output. There is no uncertainty from the classifier itself (a cut has no uncertainty). There is also no uncertainty from the learning procedure since we want to evaluate only the classifier we obtained. The question whether a training with different parameters would have resulted in a different classifier has nothing to do with the systematic uncertainties of the classifier we obtained. Again: We have a fixed classifier with a fixed cut in the output distribution and all we have to calculate is the propagation of the systematic uncertainties of the inputs.

For one-dimensional cuts this propagation is often simulated by a variation of the cut. This is a legal procedure but only in exactly this one-dimensional case. Let us assume that

a quantity z has an uncertainty δz and a cut $z < c$ is applied. Varying the cut by δz is legal because this variation is identical to the variation of all events according to δz.

For statistical learning methods we cut in the output distribution which depends on all inputs simultaneously. We have no information about any uncertainty there but what we do know are the uncertainties of the inputs. To calculate the propagation of these uncertainties several modified test sets have to be created for which the input quantities are varied according to their own uncertainty. These modified test sets are passed through the statistical learning method without any changes in the method itself, also the cut stays the same. The resulting output distributions changed according to the uncertainties of the inputs. The same cut as before will result in new efficiencies and rejections. Calculating the variation over the modified test sets finally results in an estimate of the systematic uncertainty which corresponds to the propagated uncertainties of the inputs.

The systematic uncertainties of the inputs can for example be due to an energy calibration which is varying over time, a varying noise contribution, a detector efficiency which is degrading over time or a movement of parts of the detector. Systematic uncertainties are generally found as any variation which may appear after the training set has been fixed and which affects the performance on future events by changing the inputs (or underlying quantities from which the inputs are derived) systematically in one direction (which may vary over time).

The following procedure assumes that the inputs are independent so that the variations can be added up in quadrature. One could also imagine that two or more of the inputs are correlated because they depend on a set of underlying quantities. Then the correct procedure would be to vary these underlying quantities according to their uncertainties and using their propagation over the actual inputs finally to the output of the statistical learning method.

Example: Systematic Uncertainties

Figure 3.18 shows how an original test set showing an efficiency of 80% and a rejection of 90% is modified six times, with a variation upwards and downwards for each of the three inputs. The known systematic uncertainties may, for example, be the following: x_1 has a Gaussian error distribution with an absolute sigma of $\sigma_1 = 0.1$, x_2 and x_3 have relative errors of $\frac{\sigma_2}{x_2} = 5\%$ and $\frac{\sigma_3}{x_3} = 10\%$.

The usual way to create the modified test sets is that exactly two sets are created per input (for every 1σ variation, up and down). In our example we would have six modified test sets: Starting with a set in which in every event x_1 is replaced by $x_1 + 0.1$ and ending with a set in which in every event x_3 is replaced by $0.9 \cdot x_3$. The individual inputs are here assumed to be independent, the six differences in efficiency and rejection (modified vs. original) can thus be added in quadrature. This is done by adding up the squared differences in positive direction on the one hand and all squared differences in negative direction on the other hand. This procedure covers the case that sometimes both modification directions for the same input lead to a change (of efficiency or rejection) into the same direction[6].

As can be seen in figure 3.18 the same classifier and the same cut as for the original test set are used for all the modified sets. The variation of the efficiency is shown as

[6] One can understand this effect if one thinks of a Gaussian input distribution for signal events from which the centre part, say $[\mu - \sigma, \mu + \sigma]$, is selected by the learning method. Any shift in this distribution will result in a loss of efficiency.

$80^{+5}_{-6}\%$ and the rejection varied like $90^{+2}_{-2}\%$. These are the systematic uncertainties of efficiency and rejection which were propagated from the input quantities through the statistical learning method.

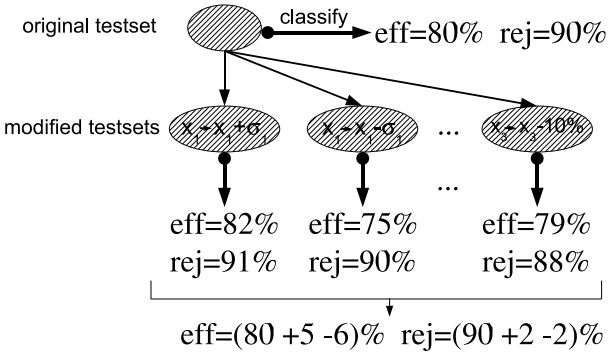

Figure 3.18: An example how to calculate the systematic uncertainties of efficiency and rejection by processing test sets which have been modified according to the systematic uncertainties of the inputs.

For regression the principle is exactly the same. There the variation of the test set according to the input uncertainties results in different values, for example, for the squared error function. The visualisation would be identical to the classification case shown in figure 3.18 with the exception of the error measure. Here the squared error as defined above is calculated for the original and for the modified test sets. The variations up and down are added up and lead to the final systematic uncertainty of the squared error.

3.14 Data Mining

Often the solution to the classification or regression problem itself is not the only goal. But physicists typically want to learn more about their datasets. Mining in a dataset means finding out more about the properties of its multidimensional distribution. Of course designated algorithms exist for this purpose [43, 44, 45] but already training statistical learning methods as discussed so far can reveal insights into the structure of the data.

Statistical learning methods can, for example, be used to derive the *relevance* of each input that is used for the classification or regression. A standard definition of the relevance of input x_r is

$$R(x_r)^2 = \frac{1}{N}\sum_{i=1}^{N}\Big(out(\ldots,x_{r-1},x_r,x_{r+1},\ldots) - out(\ldots,x_{r-1},\overline{x_r},x_{r+1},\ldots)\Big)^2 \tag{3.20}$$

where $\overline{x_r} = \frac{1}{N}\sum_{i=1}^{N} x_r$ is the mean value of x_r for the whole test set. This relevance is no absolute number but can be compared among all inputs giving the highest values for those inputs on whose variation the output depends most.

Example: Relevance in a Toy Dataset
Figure 3.19 shows a toy dataset for which a classifier is trained. The calculated relevances $R(x_1) = 0.41$ and $R(x_2) = 0.14$ reflect the fact that a cut in x_1 could obviously separate the two Gaussian distributions much better than a cut in x_2.

For classification problems with more than two classes statistical learning methods can also give hints about the multi-class-proximities which are involved. A simple procedure to

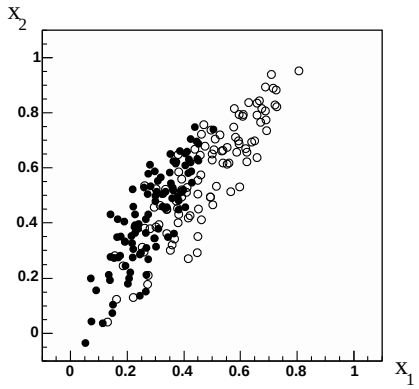

Figure 3.19: A toy dataset where both signal and background follow a simple two-dimensional Gaussian distribution.

measure the multi-class-proximities could train a statistical learning method for all pairs of classes and take the percentage of misclassifications as a measure for the proximity of each pair of classes.

Example: Proximity in a Toy Dataset

A simple two-dimensional dataset with three classes is shown in figure 3.20. The three combinations of "one vs. one" are used to train three classifiers. The minimum number of misclassifications are: • vs. ∘ 28%, ∘ vs. × 44%, • vs. × 10%. They are well in agreement with the intuitive guess from figure 3.20.

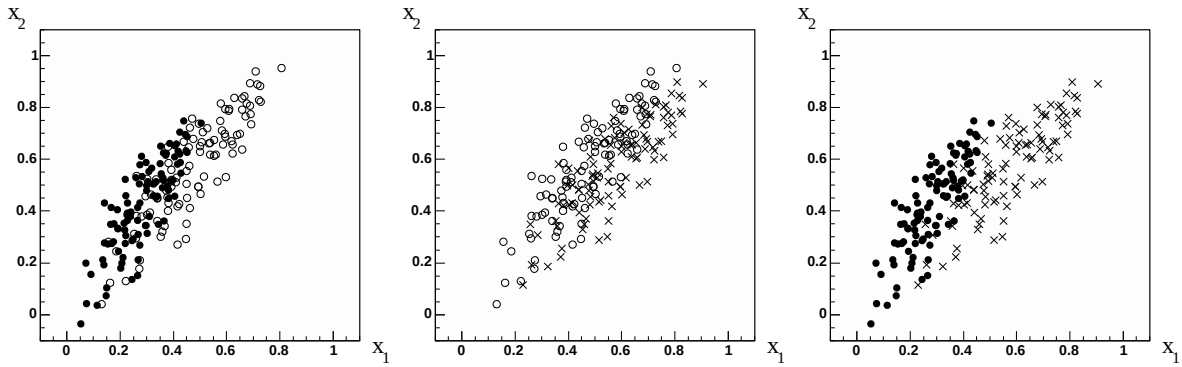

Figure 3.20: A toy dataset with three classes.

Outliers can also be detected easily with statistical learning methods that are implemented as local density estimators which will be discussed in section 5.3. Outliers are characterised by a far-beyond-average distance to the next few neighbours. As the name says, local density estimators know about the density of training examples in the vicinity of any position and can easily warn about outliers that do not resemble any training point.

Example: Outliers Identified by 2-Nearest-Neighbour-Search

The two outliers (•) in figure 3.21 have much larger distances to their two nearest examples from the training set (∘) – around 5 units whereas a typical neighbourhood is around 2 units apart – and are thus easily identified during the classification or regression process.

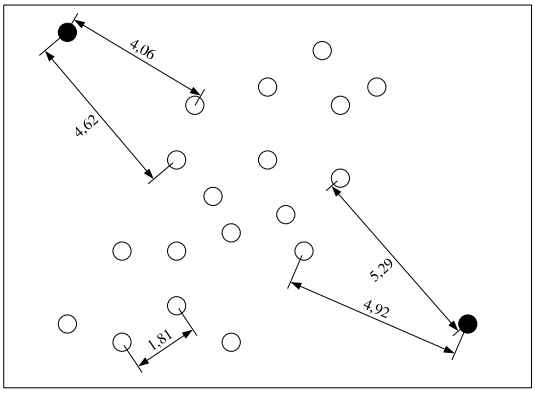

Figure 3.21: The two outliers (•) are easily identified by their large distance to the nearest training points (○) in this simple two-dimensional example.

3.15 Comparison of Statistical Learning Methods

The multitude of statistical learning methods motivates a competition among the different methods and, of course, between statistical learning methods and those algorithms which try to solve a given problem by using prior knowledge (compare the discussion in section 3.9).

There exist, however, several hurdles on the way to a valid and meaningful comparison:

- First one has to face the theorem [46] that all learning methods are equal in the sense that – given no prior information about the multidimensional probability distributions involved in the problem – any method is as good as random guessing ON AVERAGE if the error is measured for all possible events which are not in the training set. This means that averaging uniformly over all possible probability distributions (and really all since no prior information was given) makes any bias (the preference for a certain distribution) useless.

  This theorem appears as part of the "no free lunch" theorems in the literature and will be further discussed in section 4.5. From these theorems the claim to omit useless comparisons of statistical learning methods was derived. For our situation a simple argument banishes these theoretical constructs: We know that there is a very special prior which restricts the probability distributions we may face. This results from the fact that our datasets come from physics experiments (and not from random distributions).

  The interesting question which is perfectly valid to pose is which bias and therefore which learning method is the best one for these datasets coming from physics experiments.

- Once several classifiers (or regression methods) have been trained and their performances on the test set have been measured, it is not enough to simply state which one performs best. If one wants to claim that a certain method is better than all others, a decent statistical test must be applied.

- The alpha error of such a statistical test limits the probability that the claim of a discovery of a better performing method is wrong. Typically 95% confidence intervals are constructed which means that the alpha error is restricted to 5%. The problem in this context is that usually many different methods have to be compared which results in the most general case in $\frac{n(n-1)}{2}$ tests for n methods. Then the comparison-wise alpha error may be restricted to 5%, but the experiment-wise alpha error may

be much larger due to the many comparisons. If, for example, one test with a 95% confidence interval is replaced by two independent tests with 95% confidence intervals each, then the total alpha error for the two tests is no longer 5% but 9.75% since $0.95 \cdot 0.95 = 0.9025$ (probabilities can be multiplied under the assumption of independent tests).

In the next sections the statistical test for comparing hypotheses will be introduced and will then be generalised to allow the comparison of learning methods.

3.15.1 Comparing Hypotheses

The first step towards a comparison of learning methods is the comparison of hypotheses. We assume that two or more hypotheses are given (usually as the results of different learning methods) in the form of a list of output values for all events of the test set. The test set is partitioned into k disjoint sets each of which has at least around 30 events[7].

Feelders [47] suggests to construct 95% confidence intervals for the difference of the mean performances $\Delta\mu$. If the confidence interval does not contain zero we have found a significant difference between two hypotheses. The statistical test on which the calculation of the confidence interval is based is the t-test [48]. By comparing the errors of two hypotheses for each of the k parts of the test set we collect statistics (usually the differences in the number of misclassifications for classification, and the differences of errors $(out_1(\vec{x}) - y)^2 - (out_2(\vec{x}) - y)^2$ for regression) and the t-test decides whether to reject $H_0 : \Delta\mu = 0$ on the basis of the mean difference $\Delta\mu$ and its standard deviation $\sigma_{\Delta\mu}$[8].

To limit the experiment-wise alpha error Dunn [49] suggests to reduce the comparison-wise alpha error which makes the tests less powerful but the experiment-wise alpha error is then kept to, for example, 5% in total. Table 3.1 shows a list of z-values which construct a 95% confidence interval $[\Delta\mu - z\sigma_{\Delta\mu}, \Delta\mu + z\sigma_{\Delta\mu}]$ for a given number of means that have to be compared (number of learning methods). The values in the table are based on the conservative assumption that really all $\frac{n(n-1)}{2}$ tests are done.

n	3	4	5	6	7	8	9	10	11	12	13	14	15
z	2.39	2.64	2.81	2.94	3.04	3.14	3.20	3.26	3.32	3.37	3.41	3.45	3.49

Table 3.1: For a given number n of means (methods) which have to be compared, the shown z-value constructs 95% confidence intervals $[\Delta\mu - z\sigma_{\Delta\mu}, \Delta\mu + z\sigma_{\Delta\mu}]$.

Example: Comparing Three Hypotheses on a Toy Example

Let us assume that all three hypotheses A, B and C are combined with an appropriate cut so that they all give the same efficiency of 90%. Now we compare the performance on the ten background examples of the test set which is shown in table 3.2. Because of the low number of events this toy example violates the rule that each part of the test

[7] This is the typical number of events given in statistics books, which is sufficient to replace the binomial distribution by the approximated gaussian distribution.

[8] The variance obtained from the distribution accounts for statistical uncertainties. Systematic uncertainties may be added in quadrature. However, this only makes sense if the performance of the different hypotheses should be compared among different systematic settings (it just makes the test less powerful).

set should contain at least 30 events – here we divide the test set into ten parts, one event per part. A plus sign means that the classification as background is correct. For the comparison one obtains -1 if hypothesis one was wrong and hypothesis two not, 0 if both came to the same result, $+1$ if hypothesis one was correct and hypothesis two not. The confidence intervals show clearly that A is statistically significantly superior to B and C (intervals above 0). Though B is a bit better than C this difference is not significant (interval contains 0).

i	A	B	C	A vs. B	B vs. C	A vs. C
1	+	+	-	0	+1	+1
2	+	-	-	+1	0	+1
3	+	+	-	0	+1	+1
4	+	-	+	+1	-1	0
5	+	-	-	+1	0	+1
6	+	+	-	0	+1	+1
7	+	-	-	+1	0	+1
8	+	-	+	+1	-1	0
9	+	-	+	+1	-1	0
10	+	+	-	0	+1	+1
$\Delta\mu$				+0.60	+0.10	+0.70
$\sigma_{\Delta\mu}$				0.15	0.26	0.14
95% CI				[+0.24,+0.96]	[-0.52,+0.72]	[+0.37,+1.03]

Table 3.2: Example for the comparison of three hypotheses.

3.15.2 Comparing Learning Methods

After clarifying the procedure of the comparison of several hypotheses, it is important to note that the comparison of learning methods goes one step further. Here we must take into account that the construction of a hypothesis depends on the training examples. These examples are randomly sampled from some underlying probability distribution. Therefore the same strategy as above is applied but now the t-test is applied to mean and variance which result from different training sets.

In section 3.12 we already discussed k-fold cross-validation. We apply k-fold cross-validation to each of the learning methods which should be compared. For each learning method we thus create k different training sets and measure the errors of the resulting k hypotheses on their k test sets. These k values are used to calculate the mean performance and the variance for each learning method. The differences of the means of the different learning methods are then compared just like above.

Chapter 4

Statistical Learning Theory

In this chapter an overview of the theory of statistical learning is given. We will review the basics of three frameworks in which probability theory is used to derive theorems about the performance of specific classes of learning algorithms. In section 4.6 the possible application of these theorems to the learning methods discussed in chapter 5 will be analysed. Although some parts of this chapter may fit to the problem of regression, classification with two classes is the basis of the whole chapter. Overviews of statistical learning theory covering at least parts of the frameworks presented here can be found, for example, in [31, 38, 46, 50, 51, 52].

The key question in the following discussion will be in which way a small training error can guarantee a small true error if certain conditions apply to the learning method. Recall the problem of overtraining (section 3.11) and consider the two following scenarios: On the one hand, the learning method may have a very large hypotheses space from which it can choose a hypothesis that fits the training data very well. On the other hand, the learning method may have a small hypotheses space from which it can choose only a hypothesis that fits the training data fairly well. The basic assumption is now that in the latter case the true error is more likely to be as low as the training error (see section 4.5 for a criticism of this assumption). The frameworks presented in this chapter try to quantify this relationship between training error and true error without using an estimate from a test set.

The first section 4.1 will start with some basic comments about how the error of a learning method is usually measured within theoretically oriented frameworks. Section 4.2 will give a short introduction to the Bayesian framework, section 4.3 continues with the PAC framework and section 4.4 concludes with the VC-framework. A general criticism of the concepts behind these frameworks will be discussed in section 4.5. Afterwards the possible applications of theoretical results to learning methods will be investigated in section 4.6.

4.1 Error Measurement

As discussed in chapter 3, the difference between the performance (often the terms *error* or *risk* are used) on the training and on the test set is very important. The test error was important in chapter 3 as an estimation of the true error. In this chapter, however, we want to derive bounds for the true error directly from the training error and the properties of the learning algorithm. Therefore the test error will not be used here.

We will call the training set T ($|T| = n$). All elements $x \in T$ are drawn *independently identically distributed* (IID), which describes the normal sampling from one underlying probability distribution P_x in some vector space X.

We define the training error as

$$E_T(h) = \frac{1}{n} \sum_{x \in T} L\Big(h(x), y(x)\Big) \tag{4.1}$$

where the general *loss-function* L measures the difference between hypothesis h (output of the learning method) and target function y. Different loss functions are discussed below.

For the true error we need to sum (or even integrate[1]) over all x that could be drawn (IID) according to the probability distribution P_x:

$$E_P(h) = \sum_{x \in X} L\Big(h(x), y(x)\Big) P_x(x). \tag{4.2}$$

The index P reminds us that the true error is measured according to the probability distribution of x.

Different loss functions lead of course to different performance measurements. But the choice of the loss function is also essential for the theoretical framework. A very common loss function that was already used in section 3.12 is the quadratic difference. Also the absolute error is a typical choice:

$$L(h, y) = \begin{cases} (h-y)^2 \\ |h-y| \end{cases}. \tag{4.3}$$

A nice property of the squared error loss function is the possibility to decompose it into two components: *bias* and *variance*[2]. Assuming that the target function y results from a noisy readout of a function f: $y(x) = f(x) + \epsilon$, where the noise ϵ has expectation $E[\epsilon] = 0$, we can write for any position x_0 (with $h = h(x_0)$, $f = f(x_0)$ and $y = y(x_0)$):

$$\begin{aligned}
& E\Big[\Big(h - y\Big)^2\Big] \\
= \; & E\Big[\Big(h - f + \epsilon\Big)^2\Big] \\
= \; & E\Big[h^2 + f^2 + \epsilon^2 - 2hf\Big] \\
= \; & E\Big[h^2\Big] + E\Big[f^2\Big] - 2E\Big[hf\Big] + E\Big[\epsilon^2\Big] \\
= \; & E\Big[h\Big]^2 + f^2 - 2E\Big[hf\Big] + E\Big[h^2\Big] - E\Big[h\Big]^2 + \mathrm{Var}(\epsilon) \\
= \; & \Big(E\Big[h\Big] - f\Big)^2 + E\Big[\Big(h - E[h]\Big)^2\Big] + \mathrm{Var}(\epsilon).
\end{aligned} \tag{4.4}$$

The first term is the square of the bias. The bias measures the distance of the average hypothesis (averaged over all possible samplings from P_x, i.e. over all training sets) to the true function. The second term is the variance of the hypotheses around their mean. The third term is the variance that results from the noise that is inherent to y – and thus irreducible.

[1] Since no input quantity is really continuous but is probably digitised with a fixed precision one can argue that the input space X is countable and even finite.

[2] A similar decomposition was shown in section 3.12.2 by using a squared error function to evaluate the performance of a regression model.

The problem of overtraining and regularisation can be interpreted directly in terms of bias and variance: Low bias and high variance mean usually a very complex hypotheses space and potentially overtraining while high bias and low variance signify a small hypotheses space with low possibility of overtraining.

The loss function which will be (implicitly) used in the next three sections – the theoretical frameworks depend on it – is the *0-1 loss*

$$L(h, y) = I(h \neq y) = 1 - \delta(h - y) \tag{4.5}$$

which is designed for classification with two classes: It equals 0 if h and y agree, 1 if not.

4.2 Bayesian Learning

As an introduction towards hypotheses spaces and probabilities we have a look at Bayesian learning [53, 54, 55]. Bayesian learning is founded on the conditional probabilities linking evidence (data) and hypotheses. Given a finite hypotheses space $H = \{h_1, h_2, \ldots, h_n\}$ Bayesian learning selects an optimal combination of these hypotheses by applying Bayes theorem:

$$P(h_i|T) = \frac{P(T|h_i)P(h_i)}{P(T)}. \tag{4.6}$$

The probability of hypothesis h_i given the training data T will then be used in the construction of the optimal hypothesis. This probability consists of

- the *likelihood* $P(T|h_i)$ which can be measured,
- the hypotheses prior $P(h_i)$ which reflects the assumed probability of each hypothesis without taking into account any measurement,
- the inaccessible probability $P(T)$ with which the training data was sampled from the underlying distribution (but this factor is the same for each h_i and can thus be neglected).

The training events are believed to be independently identically distributed. The probability of the whole training set given a specific hypothesis can then be decomposed into a product over all training events $x \in T$:

$$P(T|h_i) = \prod_j P(x_j|h_i) \tag{4.7}$$

Bayesian learning selects a combination of all hypotheses taking into account their probabilities given the observed data. Bayesian learning selects therefore

$$h = \{x \mapsto \arg\min_y \sum_i L(h_i(x), y)P(h_i|T)\} \tag{4.8}$$

as the optimal hypothesis which means that for any x a value of y should be chosen which minimises the average loss, where the average is over all h_i according to their posterior probabilities $P(h_i|T)$. It is important to note that the prior $P(h_i)$ is needed here, some kind of assumption must be made.

One simplification is very common: *MAP* (maximum a posteriori) learning means that not a combination of hypotheses is chosen but the one hypothesis that fits the training data (a posteriori) best (maximum) is selected:

$$h_{MAP} = \arg\max_{h_i} P(h_i|T) = \arg\max_{h_i} \prod_j P(x_j|h_i)P(h_i) \tag{4.9}$$

4.3 PAC Learning (Probably Approximately Correct)

For PAC Learning [56, 57, 58, 59] we now assume a finite hypotheses space H of boolean functions (like classification) which includes the target function y. The input space X is at most countable. The true error of a hypothesis in a classification problem is the sum over the probabilities of all misclassified events (0-1 loss):

$$E_P(h) = \sum_{x \in X} I(h \neq y) P_x(x) = \sum_{x \in X : h(x) \neq y(x)} P_x(x). \tag{4.10}$$

Definition
We say that h is approximately (except for ϵ) correct if $E_P(h) < \epsilon$, where ϵ is the accuracy parameter.

Depending on the training set T which is randomly sampled according to P_x, the output of a learning method may sometimes be approximately correct (given some ϵ) and sometimes not.

Definition
We say that the output of a learning method h is probably (except for δ) approximately (except for ϵ) correct, if it is approximately correct with a probability greater than $1 - \delta$, where δ is the confidence parameter.

The goal is now to assure that under certain conditions the output of a learning method is probably approximately correct which means that we need to connect the error on the training set (whether h and y match on these points) with the true error (whether h and y match on points drawn according to P_x).

We start by regarding the errors per event $\delta(h(x) - y(x)) \in \{0, 1\}$ for $x \in T$ as Bernoulli random variables X_i which have the same expectation $p = E_P$. The following theorem limits the tail of the binomial distribution that results from the n Bernoulli experiments:

Theorem (Chernoff/Hoeffding) [60]
Let $\{X_i\}$ be independent Bernoulli random variables each with expectation $p \in [0, 1]$. Then $\forall n$

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - p\right| > \epsilon\right) < 2\exp(-2n\epsilon^2). \tag{4.11}$$

This translates for our purpose directly into

$$P\left(|E_T(h) - E_P(h)| > \epsilon\right) < 2\exp(-2n\epsilon^2), \tag{4.12}$$

and it can also be shown [60] that

$$P\left(E_P(h) - E_T(h) > \epsilon\right) < \exp(-2n\epsilon^2). \tag{4.13}$$

Since we want a statement uniformly over $h \in H$ we sum up the probabilities that some $h \in H$ has $E_P(h) - E_T(h) > \epsilon$:

$$\sum_{h \in H} P\left(E_P(h) - E_T(h) > \epsilon\right) \leq |H| \exp(-2n\epsilon^2). \tag{4.14}$$

Setting this probability to be at most δ results directly in the following theorem:

> THEOREM
> Let H be any finite set of hypotheses, T be a set of n training examples drawn independently according to some distribution P_x, y be any target function in H, and $\epsilon > 0$, $\delta > 0$. Given $n \geq (1/2\epsilon^2)\left(\ln|H| + \ln(1/\delta)\right)$ the probability that $E_P(h) - E_T(h) > \epsilon$ is at most δ $\forall h \in H$.

This result means that the true error of the hypothesis is probably (except for δ) bounded by $E_T + \epsilon$ if n is sufficiently large (scaled by ϵ, δ and $|H|$). If one wants to estimate the true error from the error on the training set one has to look for the following quantities: Naturally a large number of training events is desired but most important the hypotheses space must not be too large.

The hypotheses spaces of the learning methods presented in chapter 5 are not finite a priori because continuous parameters are used both to steer the training and in the representation of the classifier. Nevertheless one can argue that the zero-one loss function and the finite input space imply finite hypotheses spaces independent of the formulation of the algorithm.

It can already be seen that a not too large hypotheses space points directly to regularisation as discussed in section 3.11: The softer a decision boundary and the coarser the granularity of a decision region, the closer a learning method comes to the condition of a small $|H|$. Again the contradictory demands of precision on the training set (small error required) and generalisation (no learning by heart) show up.

Theorems going beyond the above one replace, for example, the size of the hypotheses space $|H|$ by a description length $2|L(h)|$. There $|L(h)|$ is the length of the binary string $L(h)$ describing h in the shortest possible way. This links to the Bayesian formalism discussed above: Any assumed prior $P(h_i)$ induces a certain description language L.

> EXAMPLE: LINEARLY SEPARABLE FUNCTIONS IN A BOOLEAN VECTOR-SPACE
> The number of linearly separable functions in the vector space $X = \{0,1\}^k$ is $|H| \leq 2^{k^2}$, thus $n \geq (1/2\epsilon^2)(k^2 \ln 2 + \ln(1/\delta))$. For $dim(X) = k = 50$, $\epsilon = 0.01$ and $\delta = 0.01$ we obtain $n \geq 8.7 \cdot 10^6$ which ensures the *PAC learnability*.

4.4 The VC-Framework (Vapnik-Chervonenkis)

The starting point of the VC-Framework [61, 62] is the attempt to find a measure for the complexity of a space of hypotheses even if it is not finite. Also the input space X may now be continuous. If H is very large (in some still to be defined way) probably any distribution of n events could be classified in all 2^n ways (assigning labels 0 or 1 to each of the n events).

Definition
We say that H *shatters* a given set of n events if functions from H can dichotomise the n events in all 2^n ways. The VC-Dimension of a hypotheses space VCdim(H) is defined as the largest number of events n (in arbitrarily easy configuration) which are shattered by H.

The two important points about the VC-Dimension are that the events may be placed in arbitrarily easy (for the separation of the two classes) configuration, called *general* configuration. But then there must exist members in H which classify these events according to any labelling with 0 and 1, which means that H must contain at least these 2^n different hypotheses.

Example: VC Dimensions
The VC-Dimension of separating hyperplanes (a linear decision surface) is VCdim(linear) = $dim(X) + 1$. Figure 4.1 shows as an example the case $dim(X) = 2$ where 3 but not 4 points are shattered by separating lines.

Simple cuts were presented in section 3.7. We now take the more general case of axis-parallel hyper-rectangles meaning two cuts per input which exclude some outer region. For this setup one can calculate $dim(X) \leq$ VCdim(cuts) $\leq 2dim(X)$.

A classical example that the VC-dimension is not necessarily related to the number of free parameters is the function $\sin(\alpha x)$. Here the input space is one-dimensional and $\sin(\alpha x) > 0$ predicts class 1 and $\sin(\alpha x) < 0$ predicts class 0. Although this function has only one free parameter, values for α can be found to dichotomise any number of points on the real axis.

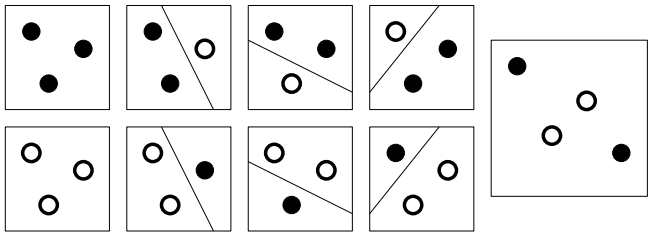

Figure 4.1: The VC-Dimension of a linear decision in two dimensions is at least 3 because all $2^3 = 8$ dichotomies shown on the left (small boxes) are linearly separable. It is exactly 3 because no linear decision surface is found for the example with four points on the right where the points are already in a general position. This example of the inability of linear decision boundaries to solve simple problems like this XOR-configuration became famous as the Minsky-Papert criticism on the perceptron (see section 5.4.2).

The following theorem looks much like the result for PAC learning because it has the same basis. The only difference is now that H is allowed to be infinite, but needs to have a finite VC-Dimension. The measure of complexity has therefore changed.

Theorem
Let $H \subseteq 2^X$ have a finite VC-Dimension d, T be a set of n training examples drawn independently according to some distribution P_x and $\epsilon > 0$, $\delta > 0$. Given $n \geq (c/\epsilon)\,(d + \ln(1/\delta))$ the probability that $|E_P(h) - E_T(h)| > \epsilon$ is at most δ $\forall h \in H$ where c is a constant.

Extensions of the VC-framework make use of this result by developing tight bounds for the VC-Dimensions of certain learning algorithms. A tight upper bound for the VC-Dimension can be calculated by using the *margin* of any hypothesis that is put out by the learning algorithm. The margin is the least distance of training events to the decision boundary in the separable case (compare the discussion of the support vector algorithm in section 5.4.3).

4.5 Criticism: No-Free-Lunch Theorems

Wolpert [46] uses an error quantity that is different from the IID true error E_P as defined above. He wants to emphasise the importance of the generalisation property of learning algorithms and thus restricts the true error to all points of the input space X which are not in the training set. The *off-training-set error* is therefore

$$E_O(h) = \sum_{x \in X, x \notin T} L\Big(h(x), y(x)\Big) P_x(x). \tag{4.15}$$

This definition makes again clear that the theoretical framework is commonly based on a finite or at least countable input space X, compare the footnote on page 72.

Since the PAC and VC frameworks provide results about the relation between training error E_T and IID true error E_P the question arises whether also for off-training-set error one could provide similar bounds.

> No-Free-Lunch Theorem (NFL)
> For any two learning algorithms A and B the mean off-training-set errors averaged over all target functions y are exactly the same for any training set T: $\overline{E_O(h_A)} = \overline{E_O(h_B)}$.

It is important that the off-training-set errors are averaged over all target functions y. Since one usually has no a priori information about the target function it seems reasonable to average over all of them. Intuitively this theorem tells us that the target function may – theoretically – have any value for all the points which have not been in the training set. This means further that any guess for these values is as good as random guessing if the resulting errors are averaged over all target functions. An extreme example is shown in figure 4.2.

Concerning strategies like "low training error and low VC-dimension gives high probability for low true error" or even concerning the simple strategies presented in section 3.12 like Cross-Validation or Bootstrapping, the No-Free-Lunch theorem states that these strategies work in as many cases as they fail. The No-Free-Lunch theorem makes every attempt to have low true error seem useless. To clarify the apparent contradiction of this theorem and the frameworks presented above we will discuss one particular aspect for each framework:

- The Bayesian formalism makes use of **prior probabilities**. The optimality of the Bayesian approach seems to be questioned by the NFL theorem because its algorithm to select the optimal hypothesis seems to be just a learning method to which the NFL theorem applies. But the NFL theorem does not apply because no averaging over target functions would make sense since the Bayesian formalism makes an assumption

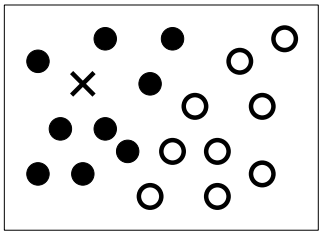

Figure 4.2: The No-Free-Lunch theorem makes local density estimation seem useless: In this example the point marked by a cross could be an open or a filled circle with equal probability. Despite the fact that this position is surrounded by filled circles so that it seems obvious that the cross is also a filled circle, both possible target functions have to be taken into account with equal probability since no prior information is given. Such a prior for local density estimators would favour locally constant target functions.

about the prior probabilities of the target functions as the first step. This makes clear that any assumption about the prior probabilities of the target function forbids the application of the NFL theorem. Such kind of priors in the form of favour for "simple" functions as discussed below in section 4.6.1 are criticised by Wolpert to be not justified from the probabilistic point of view.

- PAC-Learning (and also the VC-framework) uses **IID error** and not off-training-set error. The results that connect the training error with the true error in the PAC framework rely on the IID definition of the true error. In contrast, for the off-training-set definition of the true error the NFL theorems apply which means that the value of the training error and the size of the hypotheses space H cannot give any hint about the off-training-set error. Again it is important to note that the PAC framework works with a countable input space which ensures that the rising number of training events n makes the input space more and more known in terms of covered probability density P_x. Therefore the bound on IID error is naturally connected to the number of training events. For off-training-set error of course there is no such connection.

- Understanding the **conditional probabilities** in the VC framework is important. This understanding is not directly related to the NFL theorem. Nevertheless it belongs to the general criticism of the VC-Framework as it questions the most common interpretation of the resulting theorem. This interpretation assumes a high probability guaranteed that, given a certain value for the training error, the true (IID) error is not too far away. However, Wolpert emphasises that the correct conditional probabilities imply the following statement (and only this one): Given a fixed value of the true (IID) error there is a high probability that the training error is not too far away. This situation does not reflect the application in real life where a training error is measured and the true error should be estimated. For a conversion of the conditioned probability $P(E_T|E_P)$ into the needed $P(E_P|E_T)$ one would use Bayes theorem. But then the unknown probabilities $P(E_P)$ (which in fact matter) would be needed (compare the discussion in section 3.13.1).

4.6 Regularisation Schemes

The Bayesian Framework would result directly in a learning algorithm if prior assumptions about the target function could be justified. However, this is almost never the case. Both PAC-Learning and the VC-Framework give bounds for the true error depending on certain properties of the learning algorithm – either depending on the size of its finite hypotheses space or on the VC-Dimension of its possibly infinite hypotheses space. We now want to study different attempts to incorporate these theorems into practice-oriented learning algorithms.

4.6.1 Occam's Razor

William of Occam (1285-1349) stated "Pluralitas non est ponenda sine necessitate" (plurality should not be stated without necessity) as a basic principle of logic reasoning. This principle became a core part of the scientific method but its application has also been criticised [63].

For our context of statistical learning this principle coincides well with the results from PAC-Learning and the VC-Framework. While it might be difficult to clarify what on the one hand complexity (plurality) – or on the other hand simplicity – exactly means, the theorems discussed above speak more precisely about the size of the hypotheses space or the VC-dimension.

Still the application to existing learning algorithms is quite difficult. In chapter 5 we will see that each statistical learning method has some kind of parameter steering the hypotheses space from which the best hypothesis is learned. In most cases neither the size (mostly infinite) nor the VC-Dimension of these spaces are controlled directly but merely some kind of complexity that intuitively agrees well with the reasoning behind Occam's razor.

As mentioned in section 3.11, these regularisation parameters are tuned to provide the optimal balance between accuracy (minimising the training error) and generalisation (minimising the difference between training and true error): A hypotheses space is needed that is as complex as needed and as simple as possible.

4.6.2 MDL Principle (Minimum Description Length)

One step further to a specific measure of complexity in existing learning algorithms we find the attempt to assign a specific description language to its hypotheses space. The description length was discussed as an extension to PAC-Learning which gives the theoretical foundation.

A simple derivation of the basic idea of the MDL principle [64] is possible by applying information theory to Bayes theorem in the form

$$P(h|T) = \frac{P(h)P(T|h)}{P(T)}. \tag{4.16}$$

Maximising $P(h|T)$ (finding the best hypothesis) means therefore maximising $P(h)P(T|h)$ or $\log(P(h)) + \log(P(T|h))$. Information theory assigns as information of an event the negative logarithm of the probability: $I(X) \propto -\log(P(X))$. Thus maximising $P(h|T)$ also means minimising $I(h) + I(T|h)$. This is known as the minimum description length

principle: The optimal hypothesis is given by the shortest total description length including both the description of the training data using the hypothesis and the description of the hypothesis itself. For some learning algorithms certain description languages have been developed allowing this principle to be used directly in the regularisation process [65].

4.6.3 Structural Risk Minimisation

Structural Risk Minimisation [61] finally is the best example for a direct link between a theoretical result and its application to a learning algorithm. In fact the respective learning algorithm was not existing but has been developed on the basis of the VC-Framework: The support vector machine, discussed in more detail in section 5.4.3.

As mentioned above, extensions of the VC-theorem prove low bounds for the true error based on large margins. Therefore the support vector machine tries to find the decision boundary with the maximum margin. Of course this is only well-defined for a separable problem and, as discussed in section 5.4.3, a parameter has to be introduced for the non-separable case which steers the balance between maximising the margin and minimising misclassifications.

Structural Risk Minimisation creates a nested sequence of hypotheses spaces $H_1 \subset H_2 \subset H_3 \subset \ldots$ with rising VC-Dimension $\mathrm{VCdim}(H_1) < \mathrm{VCdim}(H_2) < \mathrm{VCdim}(H_3) < \ldots$. The hypothesis with the lowest true error is found by searching through the hypotheses spaces in the given order and stopping when the bound for the true error is minimal (Theorem from page 76):

$$E_P < E_T + c\sqrt{\frac{d + \ln \frac{1}{\delta}}{n}}. \tag{4.17}$$

The true error is bounded by the sum of training error and an additional term called here "generalisation error" which depends on the VC-dimension. Searching for the lowest true error thus means that the sum of training error and the additional "generalisation error" (depending on the VC-Dimension – thus depending on the margin) has to be minimised (see figure 4.3).

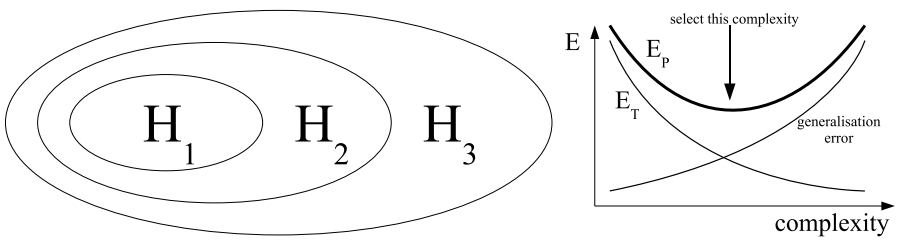

Figure 4.3: Structural Risk Minimisation: The nested sequence of hypotheses spaces allows searching for the best hypothesis which minimises the sum of training and "generalisation" error.

Chapter 5

Statistical Learning Methods

Statistical learning methods try to model a functional dependence only by looking at some examples. This requires some basic assumptions about how this function could look like, called the *bias* of the learning method. Each learning method implements its own bias, the preference for a specific kind of target function, by its internal representation of the learned hypothesis.

In fact there are two kinds of bias in each learning method: The *absolute bias* restricts the hypotheses space to those functions that can be represented by the learning method and the *preference bias* represents the wish to find the best hypothesis in a specific part of the hypotheses space because of regularisation.

The most widely used statistical learning methods can be grouped into three categories depending on the underlying idea of classification (regression is always done by generalising classification). The three categories of methods come from different basic classification concepts:

- **Decision trees**: Consecutive cuts in different inputs result in an input space which is divided into small hypercubes reflecting the class distributions. Classification is done by majority voting inside each hypercube.

- **Local density estimators**: A small environment around a new event is set and classification is done by majority voting inside this small environment.

- **Methods based on linear separation**: A hyperplane is used to divide the input space into two regions. A new event is classified according to the half space it falls into.

Based on these simple ideas many different strategies and variants evolved for each concept and each of these variants itself has different algorithmic implementations. In this chapter we will discuss the most popular methods in the three categories and restrict to as few algorithmic details as possible. Some will be given in appendix B.3.

Overviews of frequently used statistical learning methods and their basic concepts can be found, for example, in [31, 38, 50, 51, 52].

The notations which have been introduced in chapter 3 will be used here: The training examples consist of input vectors $\vec{x}_i$ complemented by the corresponding target value y_i. The output of a learning method for a specific input $\vec{x}$ will be written as $out(\vec{x})$.

5.1 Model-Based vs. Instance-Based Methods

Statistical Learning Methods can also be distinguished by their model-making behaviour. Some methods extract information from the training data, build a model and classify a new event only on the basis of that model without looking at the training data any more. Others do not build any model but classify a new event by comparing it to all or at least to a random subset of the training data.

These two different kinds of behaviour directly results in different time and memory consumptions for training and evaluation.

Model-based methods will usually spend much time on finding the correct model that fits, but does not over-fit, the training data. The number of parameters in the model is usually small compared to the amount of training data. Classifying a new event according to this model is usually fast. All decision trees and methods based on linear separation work like this.

Instance-based methods have no training time. They need the full training set for each evaluation of a new event and loop at least over a significant portion of it. Therefore each evaluation takes long. Most local density estimators work like this.

5.2 Decision Trees

Decision trees and rule-based expert systems are closely related. Both try to apply a number of queries and finally arrive at a unique solution. Non-continuous attributes (even not ordinal) play an important role in these systems. But in our context only the decision rule $x_i \leq t$ vs. $x_i > t$ will be used. The binary tree built up by consecutive tests like this is called a decision tree and the leaves of this tree carry the information about the final classification that should be made. Figure 5.1 shows an example.

The various decision tree implementations [66, 67, 68] differ mainly in the way how the tree is built, i.e. how the pair of input x_i and cut value t is selected for each node of the tree. One typical approach [67] is, for example, to define the information of a dataset D with two classes 0/1 as

$$\mathrm{Info}(D) = -p_0(D) \cdot \log_2\Big(p_0(D)\Big) - p_1(D) \cdot \log_2\Big(p_1(D)\Big) \tag{5.1}$$

and to compare this with the information after the cut (expectation over the two branches). Here $p_0(D)$ is the fraction of events from class 0 in the set D, while $p_1(D)$ is the fraction of events from class 1. The cut with the largest *information gain* (difference between expectation over the two resulting branches and previous information) is selected.

Another approach defines the *Gini index* [66] as:

$$\mathrm{Gini}(D) = 1 - (p_0(D)^2 + p_1(D)^2) \tag{5.2}$$

where the probabilities are like above approximated by the sampled events which are used in the training procedure. The Gini Index has its minimum 0 for a pure set of only one class (best case) and its maximum 0.5 for equal contributions from both classes (worst case). The cut with the smallest sum of resulting Gini indices is selected.

The basic structure of these algorithms is recursive: The rule which constructs the query by searching exhaustively through all possible pairs (x_i, t) is applied to each terminal node

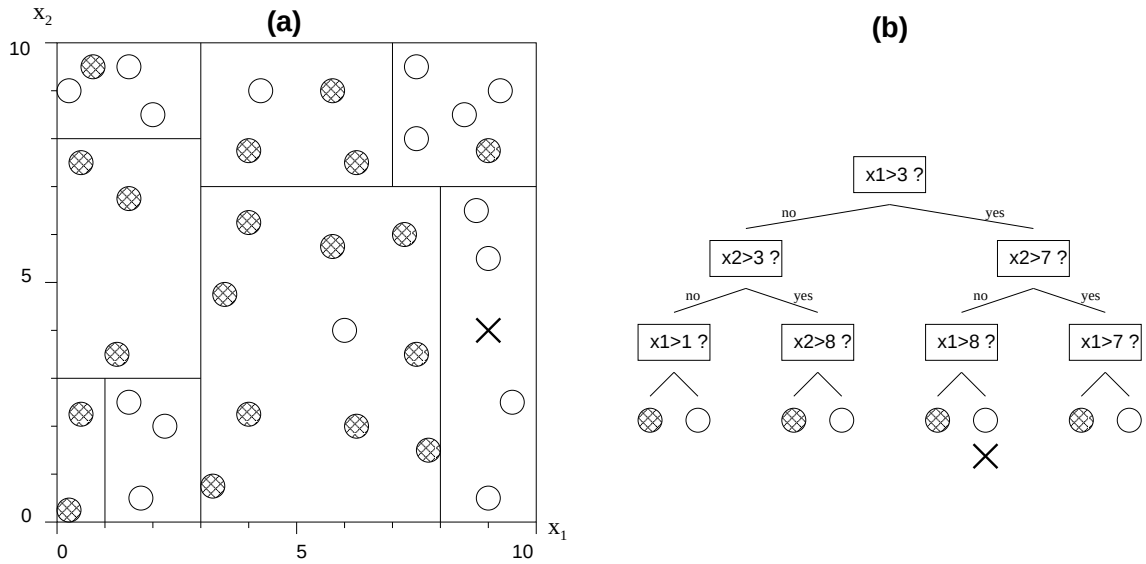

Figure 5.1: An example of a decision tree – in two dimensions the hypercubes which result from the axis-parallel cuts are rectangles. A decision for the event at (9,4) is derived by following the branches until a leaf is reached. We start at the root node and find $9 = x_1 > 3$. We follow the branches over $4 = x_2 < 7$ to $9 = x_1 > 8$ and end up in a leaf with the classification as $\circ$.

(beginning with the root node) until only nodes are left which give a final classification result and need no further branching[1]. Whether a node needs further branching may for example be defined by the purity of the remaining event sample or by the number of remaining events.

The output for a new event is found by descending in the tree structure until a leaf is reached. The leaves of the tree contain the final output associated with the corresponding slice of input space (see figure 5.1).

Regression

The generalisation from classification to regression is done by replacing the binary information in the leaves by an appropriate value (e.g. the mean target value of the remaining events) and by defining a new rule how to select the pair of input and cut for each node, for example by minimising the sum of the two variances after the cut.

Parameters and Regularisation

Parameters for decision trees usually control the stopping behaviour of the splitting algorithm. As mentioned above a minimum number of events sets a stopping condition (in addition to the obvious conditions of a pure sample or only identical inputs). Other parameters may control the details of how the best split is searched for.

In a last step, the obtained tree is pruned which means that some leaves are merged to make the tree simpler (regularisation). A confidence level steers which branches should be pruned and which not.

[1]Some of the older tree building algorithms constructed it bottom-up by a stepwise simplification of complex rules instead of top-down like shown here.

Execution Times and Variants

Decision trees are usually grown very fast and also the pruning step does not take long. For evaluation one needs to go once from the root node to the leaves which is also done quickly.

Variants of decision tree algorithms are related to the different splitting algorithms [66, 67, 68] as discussed above. Also the pruning step has different implementations (see references above). Recent developments for decision trees [69] do not take place in the algorithms themselves but in meta learning strategies which create and act upon several decision trees. These strategies will be discussed in section 5.5.

5.3 Local Density Estimators

This class of statistical learning methods is named after their strategy to look into the vicinity of the given event in input space. For classification the events of both classes are counted in this region and their numbers are compared. The class with the majority of events is the best guess for the class of the new event. For regression usually a (weighted) mean of the target values is calculated. Local density estimators differ in the way they decide whether a training event resembles the new one.

5.3.1 k-Nearest-Neighbours

This algorithm searches for the k training points with smallest distance to the new event. The size of the spherical region surrounding the new event is therefore (theoretically) adapted such that exactly k training points are taken into account (see figure 5.2). For classification the events in this region are counted for both classes, resulting in N_{good} (target $y = 1$) and N_{bad} (target $y = 0$) of the k events and the output is

$$out = \frac{N_{\text{good}}}{k}. \tag{5.3}$$

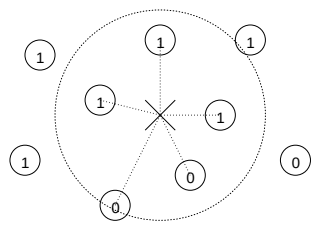

Figure 5.2: An example for a 5-nearest-neighbours evaluation with result $\frac{3}{5}$.

Regression

For regression the (weighted) mean target value of the k events is calculated:

$$out = \frac{1}{k}\sum_{i=1}^{k} w_i y_i \qquad (\sum w_i = k). \tag{5.4}$$

Parameters and Regularisation

A free parameter of the k-nearest-neighbours search is k which determines the overtraining behaviour. A very local, fine granularity decision is generated by a small k while a large k means a very soft decision insensitive to very local properties.

Another important parameter is the metric with which the distances are calculated. One can, for example, insert scaling factors along each axis into a standard Euclidean metric so that

$$\|\vec{x}\| = \sqrt{\sum \gamma_i x_i^2}. \tag{5.5}$$

Execution Times and Variants

As k-nearest-neighbours determines its output directly from the input data without building a model no training time is required. For the evaluation of a new event the distances to all training points need to be checked and the smallest k distances need to be found. This makes evaluation times very long. Many strategies have been invented to deal with this problem, ranging from smart indexing schemes to algorithms creating a condensed version of the training set with less events but trying to keep all information [70].

Variants also emerge from the distance calculation. As mentioned above, the metric can be varied and the next section introduces the idea to use the general concept of kernels instead of a specific metric.

5.3.2 Kernel Methods

As a generalisation of the free choice of the metric in the k-nearest-neighbours search the concept of kernels is now introduced. A simple local density estimator is, for example, *Parzen's window* [71]:

$$\text{local density}(\vec{x}) = \frac{1}{n}\frac{1}{V}\sum_{i=1}^{n}\phi\left(\frac{\|\vec{x}-\vec{x}_i\|}{\lambda}\right) \qquad \phi(u) = \begin{cases} 1 \text{ if } |u_j| \leq \frac{1}{2}\forall j \\ 0 \text{ else} \end{cases} \tag{5.6}$$

where the summation over ϕ simply counts the events within a region of volume V scaled by λ. Since $\|\vec{z}\|^2 = \vec{z}\cdot\vec{z}$, any distance measure can be modified by exchanging the definition of the dot product. A very general density estimation is therefore given by

$$\text{local density}(\vec{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\vec{x},\vec{x}_i) \tag{5.7}$$

where the kernel K represents any strategy for distance measurement (and thus weighting) among all events. For example the Gaussian kernel

$$K(\vec{x},\vec{x}_i) = \frac{1}{\left(\sqrt{2\pi}\lambda\right)^p}\exp\left(-\frac{1}{2}\sum_j\left(\frac{x_j - x_{ji}}{\lambda}\right)^2\right) \tag{5.8}$$

could have been used instead of Parzen's window above. Like for Parzen's window, λ scales the size of the window (with soft borders in the kernel) around the point for which the density should be estimated.

Regression

If each label shows a real value instead of 0 and 1 for the two classes one does not compare the two local densities anymore but one weights the target value y_i with the distance measurement resulting from the kernel:

$$out = \frac{1}{n}\sum_{i=1}^{n} K(\vec{x}, \vec{x}_i) y_i. \tag{5.9}$$

Parameters and Regularisation

Any chosen kernel has its own parameters, some of which may need to be set manually for each dataset. Parameters like λ above should always be available because they control the overtraining behaviour: A small λ makes the window very narrow resulting in a classifier sensitive to very local density differences. A large λ will create a very smooth classifier which is only sensitive to more global density differences.

Execution Times and Variants

The same problem as for k-nearest-neighbours search applies here: In principle all training events have to be processed. In the same way the different workarounds can be used. Variants clearly come from using different kernels, some more will be mentioned in section 5.4.3.

5.3.3 Range Search

To deal with the problem of high computing costs for large datasets (for each evaluation all training points need to be taken into account) range-searching [72] inserts all training points into a binary tree structure and needs to traverse this tree only partially for one evaluation. For this a box is placed around the position which should be evaluated and data points outside are ignored, see figure 5.3 for an illustration.

The output of the range search method for a given evaluation position is the fraction of the "good" training events lying inside the box:

$$out = \frac{N_{\text{good}}}{N_{\text{good}} + N_{\text{bad}}}. \tag{5.10}$$

Regression

Like for the methods discussed so far in the case of regression the output is the (weighted) mean value

$$out = \frac{1}{N}\sum_{i=1}^{N} w_i y_i \qquad (\sum w_i = N). \tag{5.11}$$

Parameters and Regularisation

The crucial parameter in the range search method is the size of the surrounding region which should be looked at. If it is too small the statistics is bad and overtraining will be seen, if it is too large the performance will drop. To find the right size – in the most general case two (asymmetric) values for each input – is an involved task (compare section 7.2.7).

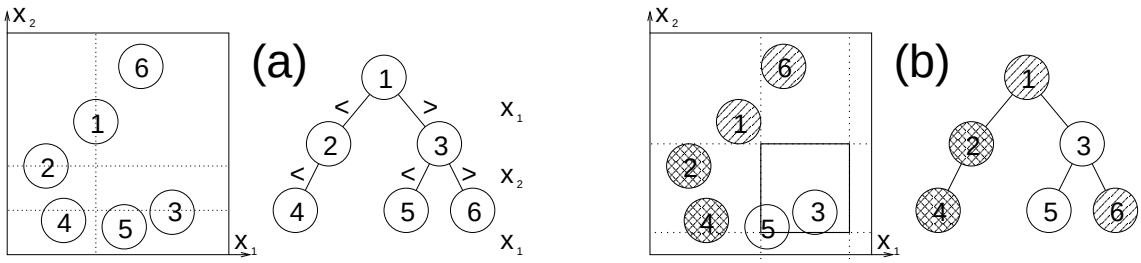

Figure 5.3: (a) The binary tree structure is built up by changing the coordinate which is used for ordering in every level. The events are filled into the tree as they are read in. Event 1 is read in first. The binary split on the first level is determined by the x_1 coordinate, thus the events 2 and 3 are filled into the two branches of event 1 because 2 is found left of 1 and 3 right. Event 4 is left of 1 and is thus filled somewhere in the branch of event 2. The binary split on the second level is determined by x_2 and event 4 is found to be below event 2. Event 5 is found right of event 1 and below event 3 while event 6 is right of event 1 and above event 3. (b) The search for the points inside the box can exclude whole branches of the tree by comparing the coordinates with the bounds of the box. The shown box is completely right of event 1 and can therefore exclude event 2 and the whole branch below it. Events 3 and 5 are found inside the box, event 6 has to be checked since the box is not completely below event 3.

Exection Times and Variants

Like k-nearest-neighbours search, range search is not either a model-based algorithm: The decision is directly derived from the training set. Therefore no training times exist. But evaluation can take quite long because the tree needs to be partially traversed for each evaluation position.

A modification of the basic algorithm may improve the performance: A slower but more effective way of setting a box size could be to use an adaptive box size. This means that the hyper-box surrounding any evaluation position is scaled to fit the local density of data points so that the number of training points in the box is of the same order for any evaluation position.

5.3.4 Naive Bayes

The naive Bayes classifier (also called maximum likelihood) [73] derives its decision – like Bayesian learning discussed in section 4.2 – from Bayes theorem

$$P(C|\vec{x}) = \frac{P(\vec{x}|C)P(C)}{P(\vec{x})} \tag{5.12}$$

which calculates the conditional probability $P(C|\vec{x})$ for class C (0 or 1) given the input vector $\vec{x}$. Maximising $P(C|\vec{x})$ (finding the correct class) means maximising $P(\vec{x}|C)P(C)$. The naive Bayes assumption is $P(\vec{x}|C) = \prod_i P(x_i|C)$ (independentness) and thus $P(C|\vec{x}) \propto P(C)\prod_i P(x_i|C)$. The probabilities $P(x_i|C)$ are estimated from histograms as described below. The prior class probability $P(C)$ may also be estimated from the data. In the following however, they are assumed to be equal for both classes.

Two histograms per input are needed to estimate $P(x_i|0)$ and $P(x_i|1)$. These histograms are projections onto the axes x_i (assumption of independentness) of the two

multidimensional sampling distributions given by the training data. By the projection process, however, the correlation between the input variables gets completely lost. This uncorrelated approach is frequently used, while the large amounts of data required for the correlated, not projected version commonly forbid its usage in practice[2]. The projection can be regarded as a good work-around as long as correlations between input variables are negligible. Unfortunately this is almost never the case in real data (figure 3.6 showed a problematic example).

As shown in figure 5.4, naive Bayes builds $2 \cdot \dim(\text{input})$ histograms (distributions for "good" and "bad" in each input dimension). The histograms are filled with the training data and normalised to present a probability density with integral one. A decision for any evaluation position is made by calculating the product of the single probabilities for the "good" histograms on the one side and for the "bad" histograms on the other side. A continuous output in the range $[0, 1]$ is then given by

$$out = \frac{\prod p_{\text{good},i}}{\prod p_{\text{bad},i} + \prod p_{\text{good},i}}. \tag{5.13}$$

In the case where no information is available ($\prod p_{\text{bad},i} + \prod p_{\text{good},i} = 0$) $out = 0.5$ is taken.

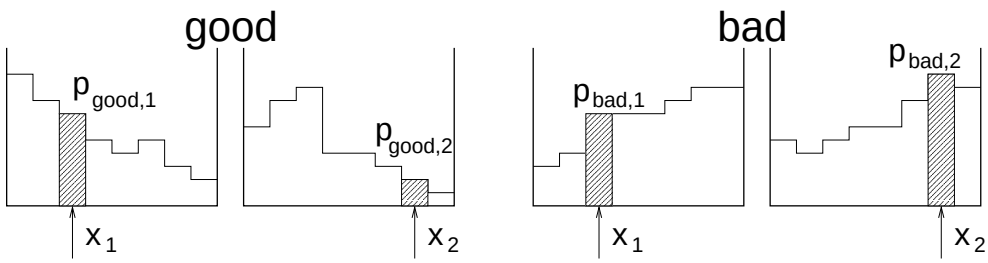

Figure 5.4: An example where the position (x_1, x_2) should be evaluated as "bad" because $p_{\text{good},1} \cdot p_{\text{good},2} < p_{\text{bad},1} \cdot p_{\text{bad},2}$.

Regression

The generalisation of this method to regression is problematic but possible. Instead of a pair of histograms for each input which represent the two classes we then need a large number of histograms for each input covering the range of the target value with a sufficiently fine granularity. If the range of the target value is binned into k intervals, the output is, for example, calculated as the target value in the bin with the maximum likelihood:

$$out = y[\text{bin} = \arg\max_k \prod_i p_{ki}]. \tag{5.14}$$

The granularity of the binning of the target value can be a serious problem as it demands a very high number of training examples if k grows large.

[2]Four input variables with 20 bins each build a multidimensional histogram of $20^4 = 160000$ bins each of which should be filled with a meaningful amount of data, say at least 10 in average. This requires 1.6 million training points at least.

Parameters and Regularisation

The binning properties are the only free parameters of this method. The size of the bins is directly related to the statistics (number of events per bin) and thus to the overtraining behaviour.

Execution Times and Variants

Naive Bayes is very fast: During training only histograms have to be filled and for any evaluation some probabilities have to be multiplied. For the case that no information is available ($\prod p_{\text{bad},i} + \prod p_{\text{good},i} = 0$) one could extend the algorithm to dynamically increase the number of bins taken into account to always arrive at a non-zero sum of probabilities.

5.4 Methods Based on Linear Separation

Another very intuitive way to classify data points is to try to separate the two classes in the mathematically simplest way: by a hyperplane. In a one-dimensional input space this is nothing else than a one-dimensional cut. In a two-dimensional input space we have a separating line, for three dimensions a plane, and so on. Instead of a series of one-dimensional cuts, one multidimensional cut has the advantage of taking into account the correlations between the inputs. A typical example was shown in figure 3.6.

The following methods make use of one or more separating hyperplanes of the form

$$\vec{w} \cdot \vec{x} + b = 0 \tag{5.15}$$

where $\vec{w}$ is the normal vector of the separating hyperplane and $b/||\vec{w}||$ is the signed distance from the origin. The formula above can also be interpreted as a regression formula with coefficients w_i and b.

5.4.1 Linear Discriminant Analysis

Linear Discriminant Analysis [74] calculates one separating hyperplane in the input space, for example, by performing a multidimensional linear regression. The regression is done by assigning $t = 0$ to one class and $t = 1$ to the other and calculating the coefficients b and w_i in

$$y_k = b + w_0 x_{k,0} + \ldots + w_l x_{k,l} + \sqrt{v}\xi_k \tag{5.16}$$

by minimising

$$\sum \xi_k^2. \tag{5.17}$$

In the linear model the error terms $\sqrt{v}\xi_k$ are assumed to have a Gaussian distribution with mean 0 and standard deviation $\sqrt{v}$. The coefficients b and w_i can be calculated by the matrix operations (see, for example, [48] for details):

$$A = \begin{pmatrix} 1 & x_{10} & \cdots & x_{1l} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n0} & \cdots & x_{nl} \end{pmatrix} \tag{5.18}$$

$$\begin{pmatrix} b \\ w_0 \\ \vdots \\ w_l \end{pmatrix} = \left(A^T A\right)^{-1} A^T \cdot \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}. \tag{5.19}$$

Notice the two-fold interpretation: b and w_i are the coefficients of a linear function (regression) but can also be seen as distance and normal vector of the separating hyperplane (classification).

Parameters and Regularisation

Since only one separating hyperplane is used, overtraining is nearly impossible. In this basic algorithm there are no free parameters to be tuned.

Execution Times and Variants

Linear discriminant analysis is a very fast method: the matrix inversion scales with the number of inputs, not with the number of training points. A variant of this method is Logistic Discriminant Analysis [75] where the separating hyperplane is not calculated by optimising a quadratic cost function but by maximising a conditional likelihood. An extension to linear discriminant analysis is Quadratic Discriminant Analysis [76] where the separating surface is then allowed to be quadratic.

5.4.2 Neural Networks

Neural networks [77, 78] have become a standard tool in many fields of application. Based on the idea to model a machine learning algorithm similar to the human brain, neural networks are now used as a precise classifier (and regression method) in many commercial and scientific applications. We will focus here on the so-called multilayer perceptron or feed forward network[3].

The elementary unit of a neural network is called *neuron* (figure 5.5 (a)). It computes the function

$$out = \sigma(\sum_j x_j w_j - b) \qquad \text{where } \sigma(a) = \frac{1}{1+e^{-a}}. \tag{5.20}$$

The sigmoidal transfer function σ is plotted in figure 5.5 (b).

The argument to the sigmoid function is called *activation*. Geometrically, the interpretation of the functionality of such a neuron is straightforward as soon as the activation is recognised as a distance measure for a separating hyperplane in the input space defined by the normal vector $\vec{w}$ and the threshold b. Applying σ to the sum results in a small value (near 0) for the one side of the hyperplane and a large value (near 1) for the other with a soft transition region in between. The length of the weight vector $\vec{w}$ scales the steepness of the threshold function and thereby the size of the transition region as shown in figure 5.6.

Historically the first kind of feed forward neural network consisted of only one neuron and was called *perceptron* [79]. A simple training rule was developed to apply this single neuron to real-life problems. The criticism of Minsky and Papert [80] eliminated most of the enthusiasm which came along with this first attempt to create an artificial neural

[3] Radial basis function neural networks have been discussed in section 3.3.

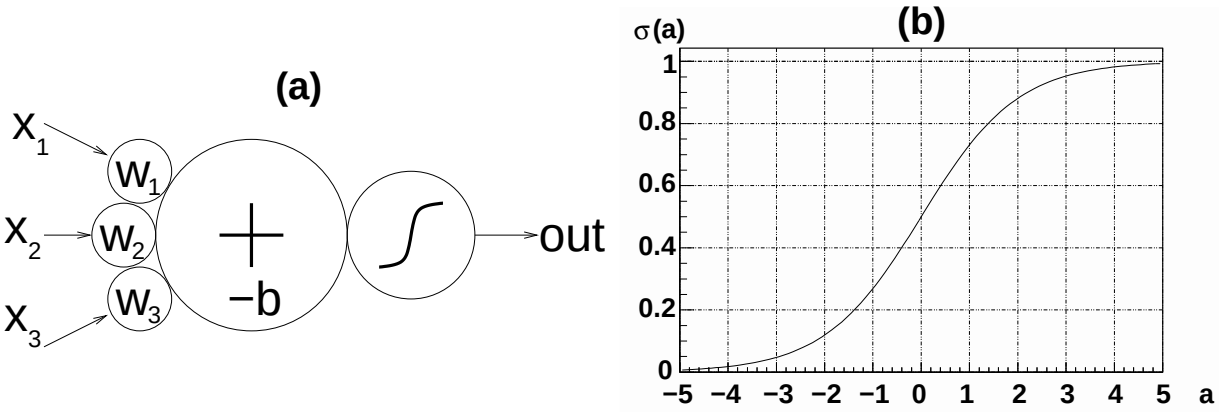

Figure 5.5: A neuron sums up weighted inputs, subtracts the threshold (a) and passes the sum through the soft threshold function (b).

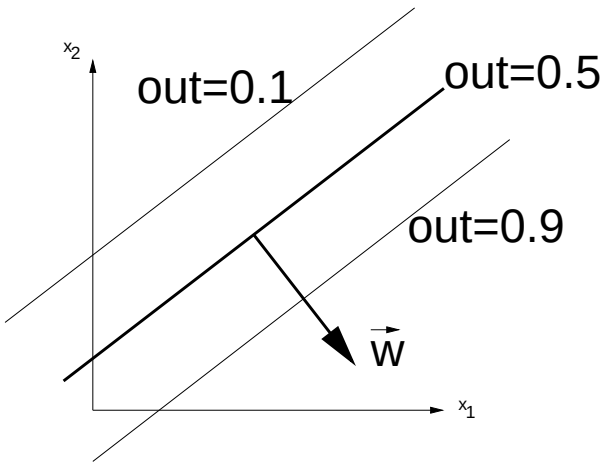

Figure 5.6: The length of the weight vector scales the width of the transition region in input space. The separating hyperplane is a line in this two-dimensional example. On the line the activation is 0 and the output thus 0.5. The longer the weight vector, the smaller is the width of the band from, for example, $out = 0.1$ to $out = 0.9$.

network that performs similarly to the human brain. Minsky and Papert remarked that this linear classifier is not able to solve some very simple problems like the XOR configuration shown in figure 4.1.

It took several years to find a training rule for the multi-layer network structure shown in figure 5.7 which offers the possibility to model much more complex functions than just a linear separation. This is achieved with the help of a sufficiently large number of neurons in the so-called *hidden*[4] layers. Throughout this thesis only one hidden layer will be used since a general theorem [81] guarantees that any continuous function can be expressed already by only one hidden layer with a sufficiently large number of neurons.

The output of a network with one hidden layer and a single output is calculated by

$$out = \sigma\left(\sum_i \sigma\Big(\sum_j x_j w_{ij} - b_i\Big) \cdot \tilde{w}_i - \tilde{b}\right) \tag{5.21}$$

where $\vec{x}$ is the input, w_{ij} are the weights connecting neuron i in the hidden layer with the jth input and $\tilde{w}_i$ are the weights connecting the output neuron with the ith neuron in the hidden layer. b_i are the thresholds of the hidden neurons and $\tilde{b}$ is the threshold of

[4]Often the input-layer is counted as the first layer despite the fact that no neuron is calculated there. Figure 5.7 would then be a three-layer network.

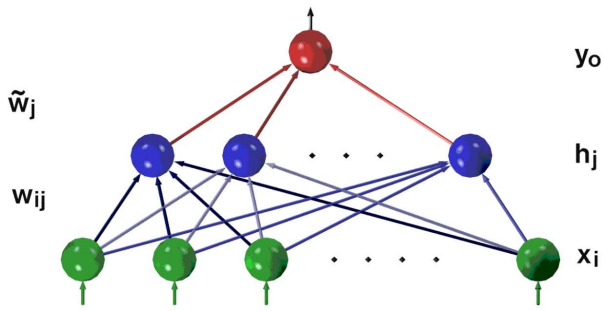

Figure 5.7: Architecture of a feed forward neural network with one hidden layer.

the output neuron. From the classification point of view a combination of the separations done with the hidden neurons is calculated in the output neuron.

Regression

From the regression point of view any arbitrarily complex function can be formed by overlaying the sigmoid functions from a sufficiently large number of hidden neurons. The soft threshold function of the output neuron is often omitted for regression but can still be used if the range of the target is within the interval $[0, 1]$.

Parameters and Regularisation

Both weights and biases are optimised to fit the given classification task during the training phase. In principle any optimisation technique can be used to find the best weights. Historically the "back-propagation" algorithm [82, 83] was used in the first applications and is still used frequently. It will be used here as an example to discuss controlling parameters and mechanisms for regularisation.

Given the cost function per event

$$\text{Cost} = \frac{1}{2}\left(out(\vec{x}) - y\right)^2 \tag{5.22}$$

a gradient descent approach

$$\Delta w \propto \frac{\partial \text{Cost}}{\partial w} \text{ with } w \in \{\tilde{w}_i, \tilde{b}, w_{ij}, b_i\} \tag{5.23}$$

leads to the update rule

$$\Delta w(k) = -\eta \frac{\partial \text{Cost}}{\partial w} + \mu \Delta w(k-1) \text{ with } w \in \{\tilde{w}_i, \tilde{b}, w_{ij}, b_i\}. \tag{5.24}$$

The partial derivatives can be calculated directly for $\tilde{w}_i$ and $\tilde{b}$ and via the chain-rule (back-propagation) also for w_{ij} and b_i. In the update rule 5.24 different parameters can be used to steer the step width of the gradient descent (η) and the scaling of a momentum term (μ) both of which control mainly how fast the algorithm converges (trying to not get stuck in secondary minima). These parameters can be set in various ways and can even be varied during the training (compare the details given in appendix B.3).

Regularisation is done with the number of hidden neurons – the more separating hyper-planes are used the more complex the decision boundary can be. But also the lengths of the

weight vectors have influence on the overtraining behaviour. The shorter the weight vectors are, the softer the threshold function is (small a in equation 5.20, compare figure 5.6). Soft threshold functions are combined to soft decision boundaries, while long weight vectors induce sharp thresholds and sharp decision boundaries. A weight decay term added in the update rule of the back-propagation algorithm can be used to penalise large weights and by this control overtraining (compare the details given in appendix B.3). A weight decay for the output neuron can also be interpreted as maximisation of the margin (compare the support vector algorithm in section 5.4.3).

The gradient descent is a local optimisation process and depends on the starting point given by a random initialisation of the weights and biases. Usually multiple networks with different initialisations are trained to avoid local minima.

Execution Times and Variants

The training times depend on the chosen strategy but are usually minutes to hours. Once a network is trained the evaluation for any given input is done very fast. Hardware implementations making use of the inherent parallelism of neural networks have been constructed. In recent implementations the calculation of a large digital neural network may take only $400ns$ [84]. Hardware implementations of neural networks will be discussed in appendix A.

Variants of the presented method of a feed forward neural network with one hidden layer naturally extend the architecture to more hidden layers (despite the theorem discussed above) and specific interconnections of these layers (also recurrent). Frequently more than one output neuron is used. As mentioned above, there are many different possibilities to train the network besides back-propagation, ranging from conjugate gradient [85] to genetic algorithms [86]. Extensions to the basic training procedure implement dynamic construction (adding) and dynamic pruning (removing) of neurons [87].

5.4.3 Support Vector Machines

Support vector machines [88, 89] are an application resulting from modern statistical learning theory (see chapter 4). Many learning tasks where, for example, neural networks were used are now in the process of being reconsidered with the help of support vector machines.

When talking about support vector machines it is convention to give the two classes which should be separated the target values $y = \pm 1$ instead of 0 and 1. The basic concept of the support vector machine is to find the optimal separating hyperplane

$$\vec{w}\vec{x} + b = 0 \tag{5.25}$$

with the maximum margin (distance to the nearest data points, see figure 5.8).

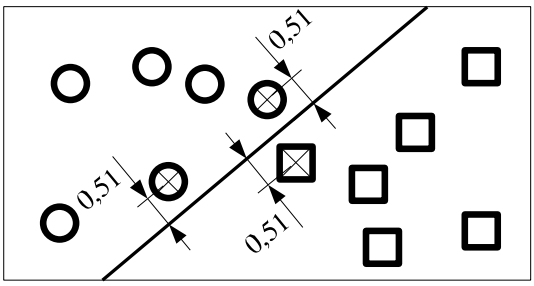

Figure 5.8: The maximum margin classifier is defined by the maximised distance to the nearest data points (here 0.51).

To find this maximum margin hyperplane we choose a minimum distance of $1/||\vec{w}||$ and search the appropriate $\vec{w}$ and b by setting the conditions

$$\vec{w}\vec{x}_i + b \geq +1 \quad \text{if } y_i = +1 \quad \text{and} \quad \vec{w}\vec{x}_i + b \leq -1 \quad \text{if } y_i = -1 \tag{5.26}$$

or simply

$$y_i(\vec{w}\vec{x}_i + b) \geq +1 \tag{5.27}$$

and minimising $||\vec{w}||$. In other words, the correct classification is ensured by equation 5.27. From the remaining subset of $(\vec{w}, b)$ (constrained by 5.27) we choose the solution where $||\vec{w}||$ is minimal (structural risk minimisation, see section 4.6.3) so that indeed $\vec{w}\vec{x}_k + b = \pm 1$ for certain k's which are then the nearest points to the separating hyperplane. The minimisation leads to the Lagrangian

$$L = \frac{1}{2}||\vec{w}||^2 - \sum \alpha_i(y_i(\vec{w}\vec{x}_i + b) - 1). \tag{5.28}$$

The solution of this optimisation problem is – in contrast to the gradient descent for neural networks – unique. The Karush-Kuhn-Tucker theorem [90] states that only for those vectors $\vec{x}_k$ fulfilling condition 5.27 with equality (thus have minimal distance to the separating hyperplane) the respective α_k will be non-zero: these are the support vectors.

By substituting $\vec{w} = \sum \alpha_i y_i \vec{x}_i$ (which is a necessary condition for an optimal L) into the hyperplane $\vec{w}\vec{x} + b$ we can determine the output for a new event using the soft threshold function from section 5.4.2 modified appropriately to return the range $[-1, +1]$:

$$out = -1 + 2\sigma(\sum \alpha_i y_i \vec{x}_i \vec{x} + b). \tag{5.29}$$

As for neural networks in the case of regression the threshold function is usually omitted for the output.

Note that the sum needs to run over support vectors only, and only their dot products with the new vector are calculated. As we can substitute $\vec{w} = \sum \alpha_i y_i \vec{x}_i$ in equation 5.28 like above, the optimisation process can also be done by evaluating only dot products between training events.

This offers the possibility to generalise the so far only linear discrimination to arbitrary decision boundaries induced by a certain kernel: The normal dot products are replaced by dot products in some feature space F. The mapping $\Phi : R^N \rightarrow F$ is not explicitly calculated but hidden in the kernel:

$$K(\vec{x}, \vec{y}) = \Phi(\vec{x})\Phi(\vec{y}). \tag{5.30}$$

Typical kernels contain some free parameters which need to be set appropriately, like the polynomial kernel $K(\vec{x}, \vec{y}) = (\gamma\vec{x}\vec{y}+n)^d$ and the Gaussian kernel[5] $K(\vec{x}, \vec{y}) = \exp(-\gamma(\vec{x} - \vec{y})^2)$.

The interpretation of the support vector classification is straightforward in feature space: Only one hyperplane is calculated there and, as described above, the support vectors are those points that are closest to the border of the two classes. Projecting back the hyperplane into input space reveals the non-linear decision boundary, as shown in figure 5.9.

[5]The factor γ could also be a vector giving a different scale to each input.

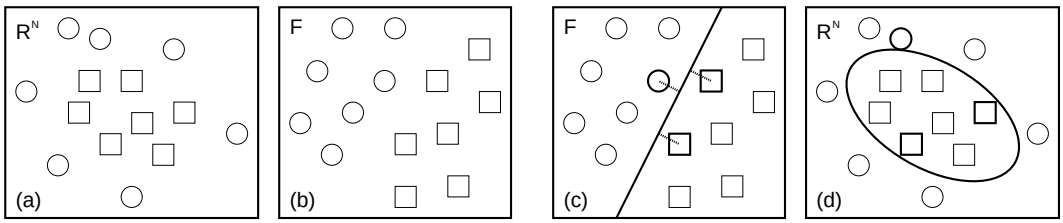

Figure 5.9: Support vector classification: Separating circles from boxes (a) is easier in feature space (b). The separating hyperplane and the support vectors with their minimal distance (c) are projected back into input space (d).

Parameters and Regularisation

If equation 5.27 cannot be fulfilled by all events the problem is not separable. In this case slack variables ξ_i have to be introduced which describe the violation of the optimality contraints:

$$y_i(\vec{w}\vec{x}_i + b) \geq +1 - \xi_i \qquad \text{with } \xi_i \geq 0. \tag{5.31}$$

The Lagrangian then becomes

$$L = \frac{1}{2}||\vec{w}||^2 - \sum \alpha_i(y_i(\vec{w}\vec{x}_i + b) - 1) + C\sum \xi_i - \sum \mu_i \xi_i \tag{5.32}$$

where the factor C determines the balance between generalisation (minimisation of $\|\vec{w}\|$) and precision (minimisation of the slack variables) and therefore controls the overtraining behaviour. The last sum with the Lagrange multipliers μ_i only enforces positivity of the ξ_i.

Additional parameters are part of the used kernel and may also have influence on the overtraining behaviour like the scaling factor γ in the Gaussian kernel.

Execution Times and Variants

The training times are comparable to neural networks but evaluation can take longer for support vector machines. Despite their similar structure the number of support vectors is usually much higher than the number of hidden neurons in a corresponding network.

Looking back to section 5.3.2 we see that *kernelisation* can be done with many different types of learning methods. Wherever the training examples appear only in the form of dot products a great variety of kernels can be plugged in and tried out.

Implementations of support vector machines mostly differ in the way the optimisation of equation 5.28 is done. Many improvements to the basic quadratic optimisation algorithm have been invented to make the procedure fast enough, even for very large datasets [91, 92].

5.5 Meta Learning Strategies

To improve the performance of any kind of statistical learning method different strategies were invented to combine several differently trained methods. Depending on the strategy the different classifiers acting together may have been trained by the same learning method or may have even been trained by different learning methods. The aim is to create one well performing classifier from several less well performing. We will speak in the following only

about classifiers. All meta learning strategies can, however, be applied also to regression problems.

5.5.1 Stacking

Stacking [93] means that one classifier acts on top of several different classifiers. The different classifiers forming the basis result from different learning methods and differ naturally in some way in their decision making. Or they are all of the same type and some kind of modification leads to different kinds of behaviours.

To combine these classifiers their differing outputs are taken as inputs to another learning method. Like a director who acts depending on the different votes of his advisors the last combining learning method is trained to judge and weight correctly the outputs from the different first level methods (see figure 5.10).

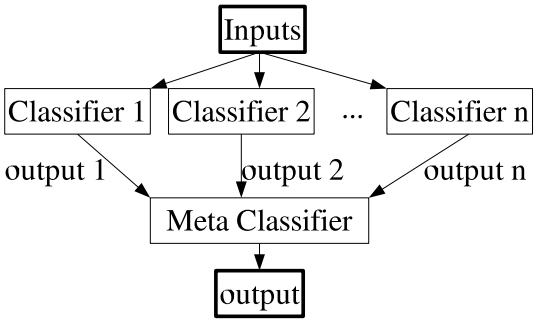

Figure 5.10: The stacking scheme for a classification problem.

5.5.2 Bagging

Bootstrap **agg**regat**ing** or *bagging* [69] modifies the training set several times by randomly drawing events from the original set with replacement (compare the bootstrapping strategy discussed in section 3.12). With this procedure new training sets of the same size are formed. With these sets several classifiers are trained in parallel (see figure 5.11).

The differences in the training sets guarantee that there will be some disagreements in the outputs. The final output is calculated, for example, by averaging over the individual outputs. The aim is to create one well-performing method by averaging over many approximately correct but independent outputs.

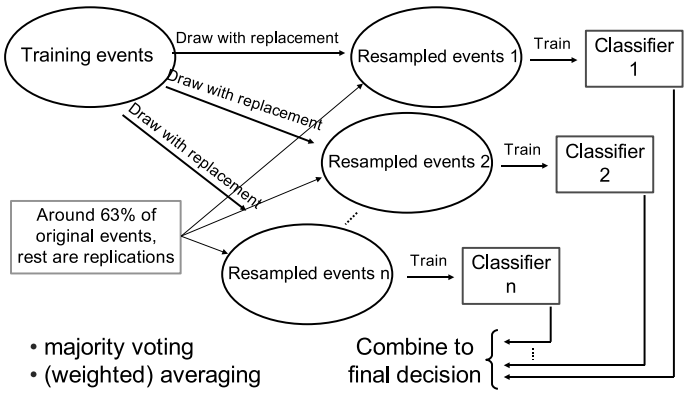

Figure 5.11: The bagging scheme for a classification problem.

Example: Random Forest

A favourite learning method based on the bagging principle is the random forest method [94]. The basic method is a decision tree algorithm called CART [66] with slight modifications introducing random behaviour in the splitting algorithm. On top of this basic method bagging is used to create several decision trees resulting in a *random forest*.

Although decision trees were always regarded as learning methods returning very meaningful information about the data ("structural relationship"), for example as a list of rules, bagging generally, and in particular the random forest method, do not allow any more this kind of interpretability. The performance gain from one CART tree to a random forest is however quite remarkable (compare the study in section 7.2.5 where the performance difference between the decision tree C4.5 and random forest is visualised). The remarkable gain in performance from one CART tree to a random forest has also been documented in the literature [94].

5.5.3 Random Subspace Method

The *random subspace method* [95] works very much like bagging: several modified training sets are created and the outputs of all the trained classifiers are averaged. The only difference is in the way in which the modified training sets are created. While for bagging the inputs were always the same and only the events were changed, it is the other way round for the random subspace method. As the name indicates, a subspace of the whole input space is chosen randomly by ignoring all but the selected inputs. By this procedure new training sets are formed with the same events but different sets of inputs (see figure 5.12).

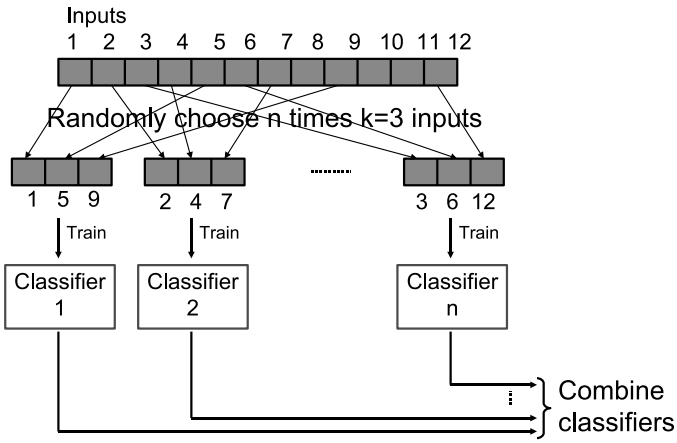

Figure 5.12: The random subspace method for a classification problem, here three inputs are selected for each classifier.

5.5.4 Boosting

In contrast to the other meta learning methods, *boosting* [96] trains serially and changes the weights of the training events according to the accuracy of the classifier trained before. We define the performance:

$$\beta = \frac{1-E}{E} \qquad \text{where} \quad E = \frac{1}{\sum w_i} \sum w_i |out(\vec{x}_i) - y_i|. \tag{5.33}$$

The adaptation of the weights from one step to the next is done by

$$w_i \mapsto w_i \times \beta^{E_i} \qquad \text{where} \quad E_i = w_i |out(\vec{x}_i) - y_i|. \tag{5.34}$$

Boosting tries to "boost" the performance of the underlying method by giving higher weights to those examples that were misclassified. The final output is not that of the last iteration but again a weighted average of all trained methods. The weight for each classifier is given by its performance β (see figure 5.13).

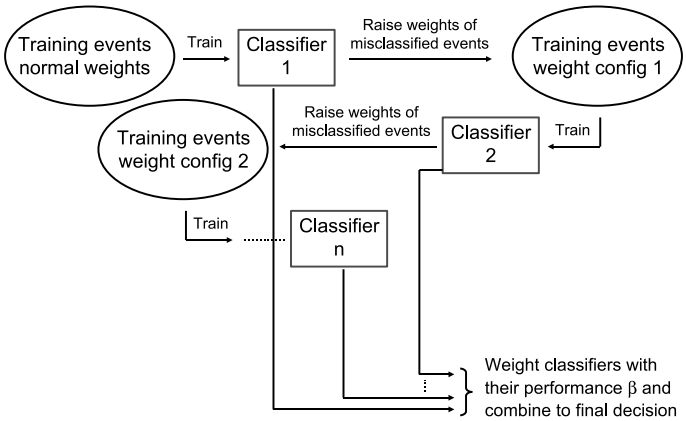

Figure 5.13: The boosting scheme for a classification problem.

5.6 Typical Properties of Different Classification Methods

To get an intuitive feeling of how the presented learning methods behave, three artificial two-dimensional datasets will be used. All three problems are classification problems since regression would be more difficult to visualise.

In the three examples the datasets will be presented and the two-dimensional output distribution will be shown for each method which has been discussed in this chapter. We will present additional plots regarding the internal structure of the algorithms whenever meaningful.

5.6.1 Toy Example 'Hole'

This dataset describes a square with a square hole in it. Without overlap of the two classes and with a structure induced by the coordinate axes it is an artificial and rather simple classification problem. The test-part of the dataset is plotted in figure 5.14. Here and in the following examples 10000 datapoints have been generated which are divided into training, selection and test set with the fractions 50%:25%:25% (compare the discussion in section 3.11).

The cut-based approach (see section 3.7) cannot handle well the correlation in this problem. As shown in figure 5.15 left, only the outer parts can be cut out. But the whole convex square remains as signal region[6]. The decision tree C4.5 had no problems detecting

[6]The output value is defined here and in the following results for simple cuts as 2^{-c} where c is the number of cuts an event does not fulfil.

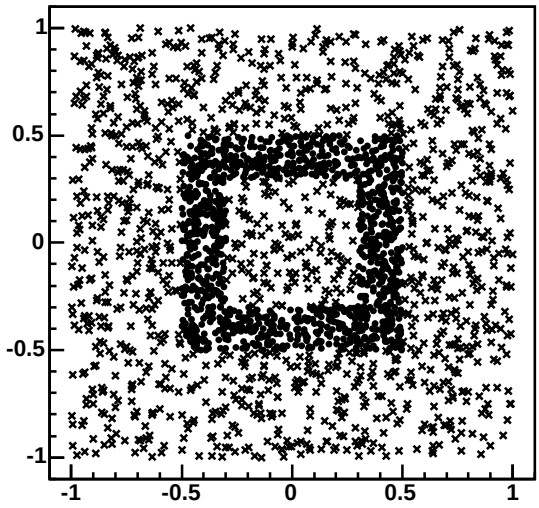

Figure 5.14: The test set of the toy example 'Hole'.

the rectangular decision boundaries as can be seen in figure 5.15 second from left. The k-nearest-neighbours-search returned the least error on the selection set for $k = 1$. The resulting output distribution is as spiky as one would expect, shown in figure 5.15 third from left. The range search method returned the least error on the selection set for a box size of 0.05 for x and 0.1 for y. Since the problem is symmetric this is clearly a statistical effect. The output distribution in figure 5.15 right shows the resulting smoothing-effects in the regions of the decision boundary. It also shows the beginning of the data-free regions on the borders of the plot where an output value of 0.5 shows that without training data in this region no decision is possible.

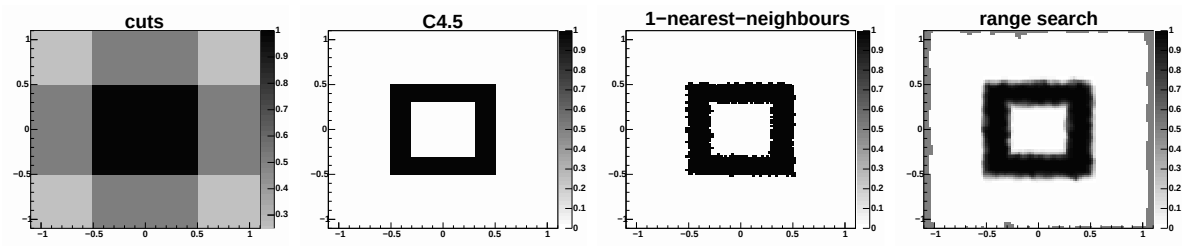

Figure 5.15: 'Hole': Output distributions for simple cuts, C4.5 tree, 1-nearest-neighbours search and range search.

For the naive Bayes method this toy problem is ideal. The histograms showing the normalised distributions of both classes along the x-axis are plotted in figure 5.16. Twenty bins are sufficient to get the bin borders matched to the borders of the square and its hole. Therefore the output distribution in the same figure shows a perfect selection of the signal region with a cut at about 0.7.

Neural networks with 8 to 12 hidden neurons were trained and the least error on the selection set was found for a network with 10 hidden neurons whose graphical representations as separating hyperplanes (lines[7] in this two-dimensional case) are shown in figure 5.17. Eight of the hidden neurons form the boundaries of the inner and outer part of the signal

[7]The thickness of the line is proportional to the weight of the hidden neuron to the output neuron.

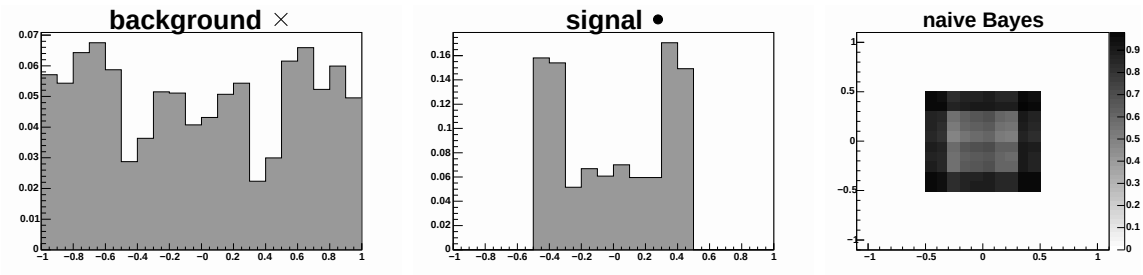

Figure 5.16: 'Hole': Naive Bayes training result – the one-dimensional histograms show the normalised distributions of both classes along the x-axis (y-axis identical due to symmetry) – the output distribution is on the right.

region as expected while one non-aligned line with a small output-weight is visible, the other one lies beyond the boundaries of the plot.

The combination of the "hidden decisions" in the output neuron results in the presented output distribution which shows the signal region with some artefacts in the inner and outer corners. These artefacts are due to the finite length of the weight vectors (normal vectors) of the separating lines. As discussed in section 5.4.2 this corresponds to the broadness of the sigmoid transfer function – infinite long weight vectors would result in a theta-function (and would remove the artefacts here). In real-life examples, however, almost always a weight decay term is useful in the training procedure which translates into the preference of short weight vectors and thus rounded transition regions.

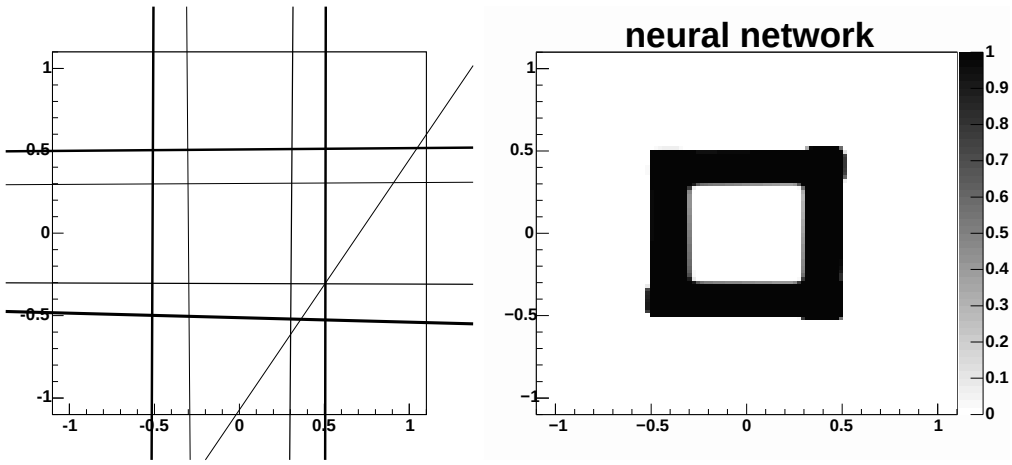

Figure 5.17: 'Hole': Neural network training result – the separating lines as expressed by the weight-vectors of the hidden neurons (left) and the final output distribution (right).

Also the support vector machine prefers a rounded solution[8]. The chosen 350 support vectors and the resulting output distribution are plotted in figure 5.18. The few outer support vectors were chosen despite the fact that they are not near to any decision boundary (though in feature space they probably are). They are producing the symmetric artefacts extending from the corners of the square.

The random forest method uses 10 trees and profits like the decision tree algorithm C4.5 of the rectangular structure. The output distributions of two of the ten trees are

[8]This is directly related to the Gaussian kernel $K(\vec{x}, \vec{y}) = \exp(-\gamma(\vec{x} - \vec{y})^2)$ which was used.

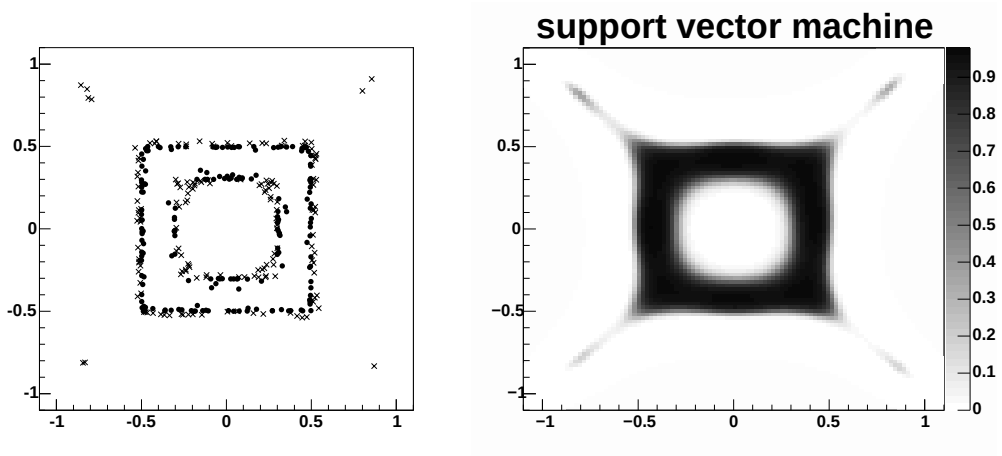

Figure 5.18: 'Hole': Support vector machine training result – the chosen support vectors and the final output distribution.

shown in figure 5.19 and the average over all 10 trees forms the final output distribution which is also shown there. The principle of creating many, by purpose noisy, classifiers which then form a smooth and well behaving classifier in average is clearly visible.

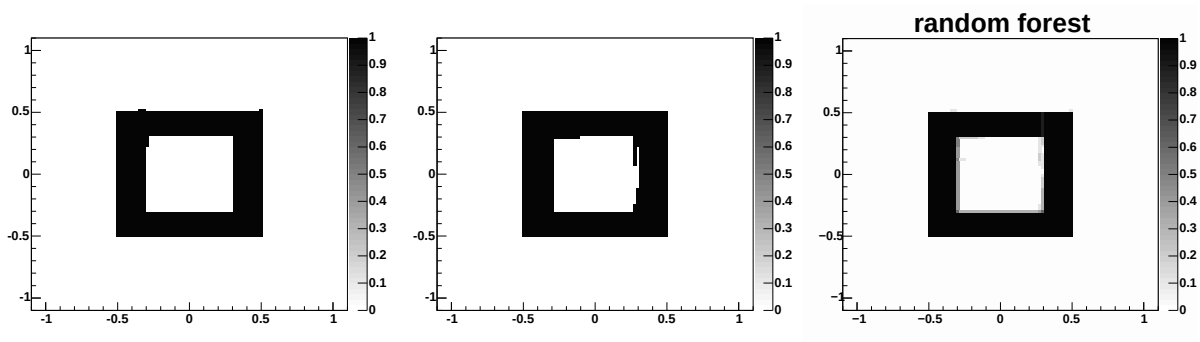

Figure 5.19: 'Hole': Random forest training result – two of ten trees (left and middle) building up the final output distribution (right) by averaging.

5.6.2 Toy Example 'Rings'

This dataset describes two rings with the same centre and radius but different Gaussian widths. For one ring the radius has a Gaussian distribution around $\mu = 1$ with $\sigma = 0.05$ while the other has $\mu = 1$ and $\sigma = 0.2$. This dataset seems quite constructed and difficult to solve because of the large overlap of the two classes. Nevertheless it has two important properties: On the one hand the optimal solution given infinite statistics is perfectly known (two circles with varying radii enclose the smaller distribution). On the other hand it is obvious that this problem cannot be solved by one separating hyperplane nor by two or three. It thus can be regarded as a rigorous test for the capabilities of the various statistical learning methods and in particular of neural networks. The test set is plotted in figure 5.20 and the numbers are the same as for the 'Hole' dataset.

Like for the 'Hole' example the cut method cannot do more than cutting away the outer part. This leaves a square as signal region as shown in figure 5.21 left which is worse for

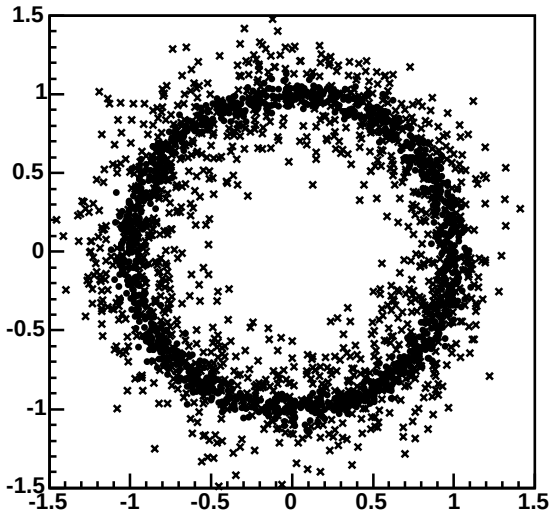

Figure 5.20: The test set of the toy example 'Rings'.

this circular shape than it was for the square 'Hole' example. The circular structure is also much more difficult for the decision tree C4.5 which tries to approximate it by rectangles (figure 5.21 second from left). Unfortunately the granularity chosen by the algorithm is too coarse and the output range between 0.1 and 0.9 is hardly used[9]. For the k-nearest-neighbours decision a value of $k = 25$ showed the least error on the selection set. As expected the decision boundaries in figure 5.21 third from left are smoothed out but still noisy. We see a similar, less noisy output distribution for the range search method. Even more obvious than for the 'Hole' example we see the regions without training data where no decision can be made (figure 5.21 right).

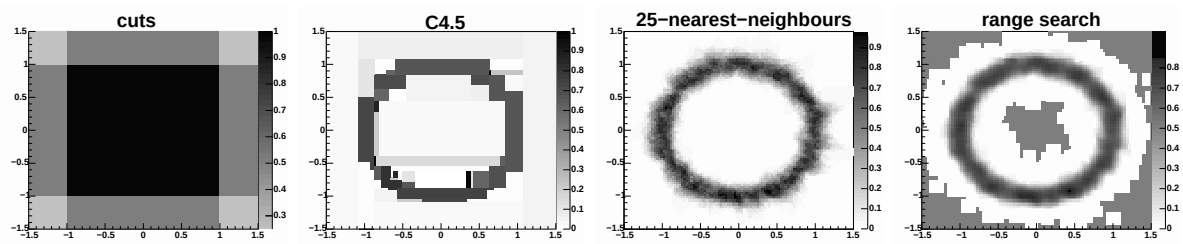

Figure 5.21: 'Rings': Output distributions for simple cuts, C4.5 tree, 25-nearest-neighbours search and range search.

The assumption of independentness which is inherent to the naive Bayes method transforms the circular structure into a rectangular one. This behaviour generates artificial corners as seen in figure 5.22 where the misclassification rate is very high.

Neural networks need to approximate the circular structure with many separating lines. Although still linear decision boundaries are visible in the output distribution in figure 5.23 right – especially if one compares to the alignment of the separating lines (left) – the circular shape is nevertheless well approximated by 13 of the totally 16 hidden neurons (the other three neurons have a very small weight towards the output neuron and do not contribute much to the final decision). The circular shape is approximated well because any corner

[9]This makes it also difficult to choose in other examples the efficiency and rejection which are needed.

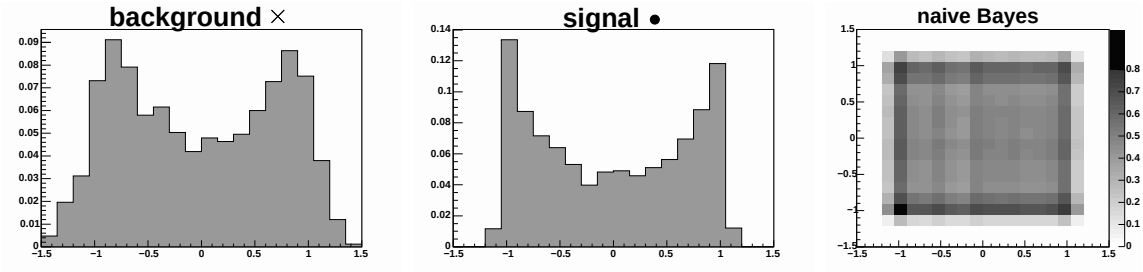

Figure 5.22: 'Rings': Naive Bayes training result – the one-dimensional histograms show the normalised distributions of both classes along the x-axis (y-axis identical due to symmetry) and the output distribution on the right.

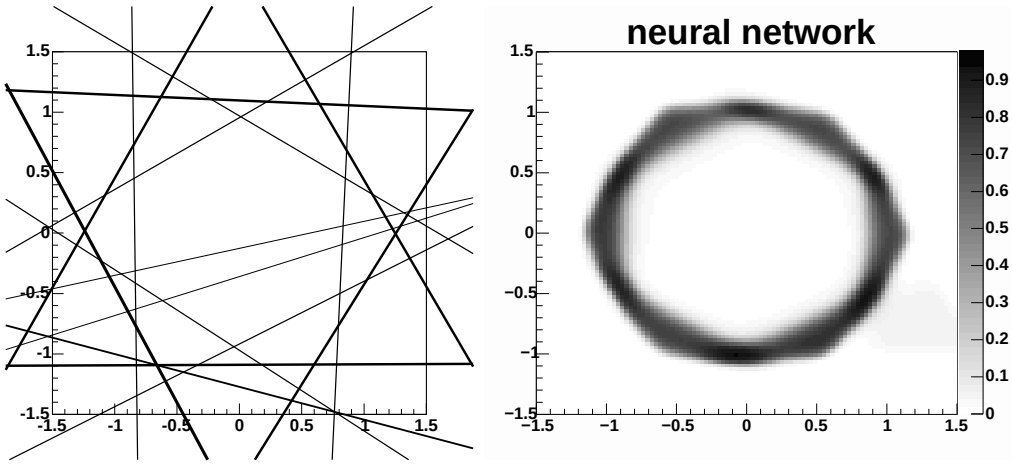

Figure 5.23: 'Rings': Neural network training result – the separating lines as expressed by the weight-vectors of the hidden neurons and the final output distribution.

where two separating lines meet can be rounded nicely by their short weight vectors which lead to a very soft threshold function.

With 1311 support vectors the support vector machine produces almost perfect circles as decision boundaries which can be seen in figure 5.24. The preference for circular structures due to the chosen Gaussian kernel was already visible in the 'Hole' example. Hence support vector machines with a Gaussian kernel perform better on circular than on linear structures in contrast to most of the other methods. However, any real dataset will have neither only a circular nor only a linear structure.

More obvious than in the 'Hole' example the single trees in the random forest method are very noisy (figure 5.25). Still much of this noise is visible in the final output distribution which is the average over 200 such trees.

5.6.3 Toy Example 'Gaussians'

The third toy example is a dataset with two classes each of which consists of two Gaussian distributions. The signal class dominates in the left and upper region of the unit square while the background covers the lower right area. The two classes have also a large overlap as shown in figure 5.26. This toy example is the most realistic of the three examples as it consists of multiple Gaussian distributions which have a large overlap but also regions

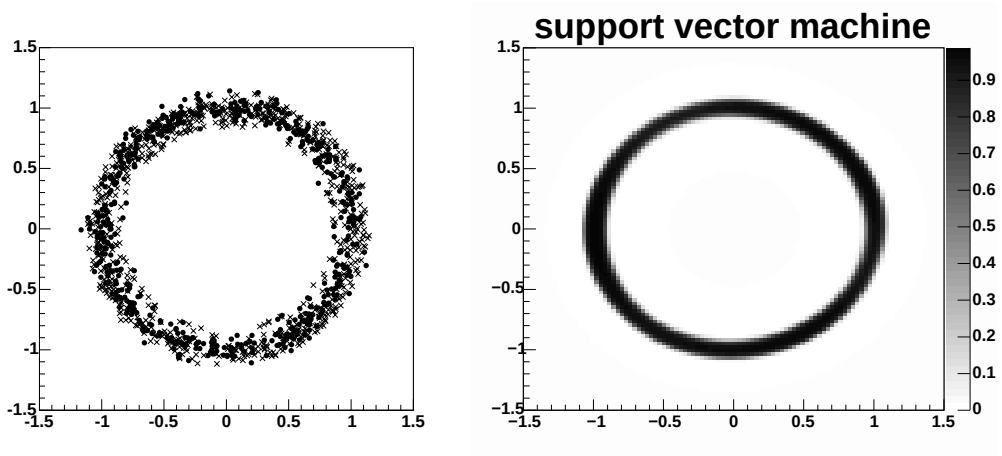

Figure 5.24: 'Rings': Support vector machine training result – the chosen support vectors and the final output distribution.

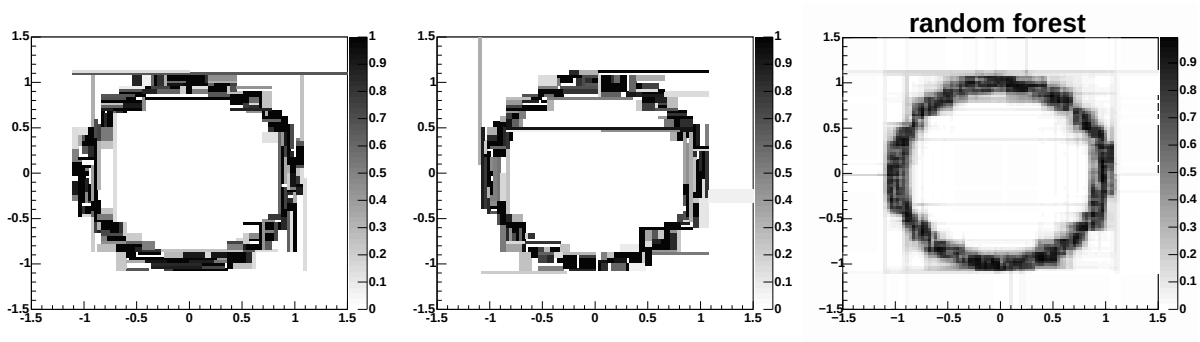

Figure 5.25: 'Rings': Random forest training result – two trees (left and middle) of 200 trees building up the final output distribution (right) by averaging.

which are easy to classify. Furthermore the density of events differs significantly within the unit square.

Figure 5.27 shows the output distributions of four different methods. The first output belongs to the cut method. Simple cuts try to select the left and upper region but they cannot model a good decision boundary because no correlated cut is done.

The decision tree C4.5 has the same problem here as with the 'Rings' example: The approximation of the linear and curved decision boundaries with rectangles is too coarse and the output range between 0.1 and 0.9 is hardly used (figure 5.27 second from left). Nevertheless it is interesting to see that the decision tree shows exactly the same behaviour as the simple cut method in the lowest and rightmost part of the unit square.

Although the k-nearest-neighbours search averages over $k = 50$ neighbours, the decision boundary still looks very noisy. The extension of the signal region to the right side in the lowest part of the plot (figure 5.27 third from left) is common among the local density estimators – they have difficulties with the low density of events in this region.

These low density regions are even more problematic for the range search method. It shows strange behaviour not only in the lowest part of the plot but also in two places in the left part (figure 5.27 right). In addition we see again the large regions where the decision should be clear but cannot be determined by the range search method because statistics

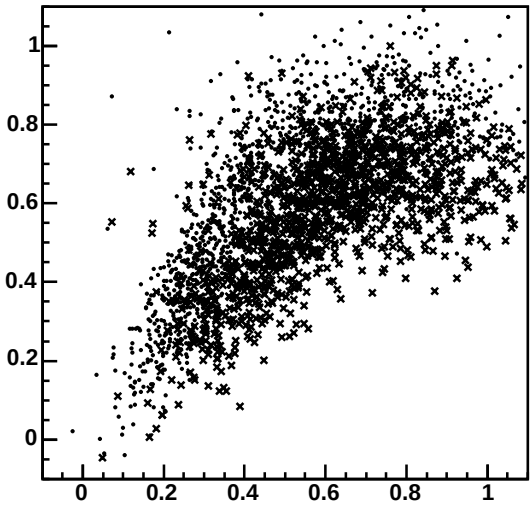

Figure 5.26: The test set of the toy example 'Gaussians'.

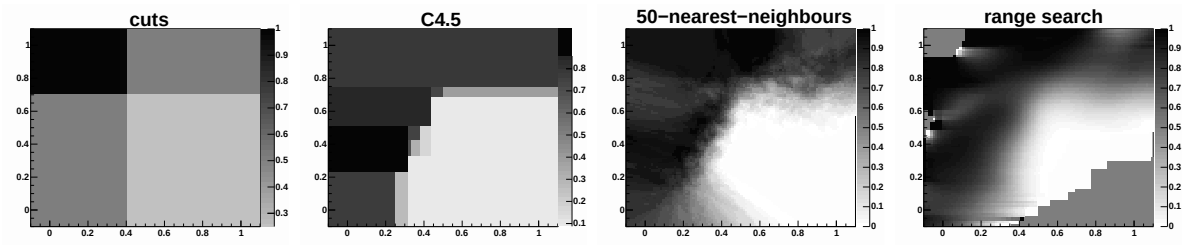

Figure 5.27: 'Gaussians': Output distributions for simple cuts, C4.5 tree, 50-nearest-neighbours search and range search.

was too small there. Nevertheless, the main decision boundary is modelled well.

For the naive Bayes method the smallest misclassification rate on the selection set was found for a coarse binning in x (10 bins) and a fine binning in y (80 bins). Figure 5.28 shows the probability histograms in x-direction (upper row) and y-direction (lower row). Using these probability estimates the naive Bayes method runs into problems with the strong correlation in the left part of the unit square, similar to the simple cuts, as shown in the output distribution (right plot).

This problem is an easy one for neural networks because the mathematically optimal decision boundary between two Gaussian distributions is a separating hyperplane. As both classes consist of two overlapping Gaussian distributions two or three combined decision lines should do the job. Indeed the neural network uses three of four given hidden neurons to perform the separation as shown in figure 5.29. The output distribution shows a very nice and smooth decision boundary.

As discussed above, support vector machines with a Gaussian kernel prefer rounded decision boundaries over linear ones. In this toy example this leads to a nicely curved decision boundary. An artificial background region in the upper part of the output distribution in figure 5.30 is however generated by the desire to curve the two signal regions – which were correctly recognised as the two Gaussians contributions. This artifical background region is not motivated by the training data, it is only induced by the chosen kernel (compare the chosen support vectors in figure 5.30 with the test events in figure 5.26). Changing the

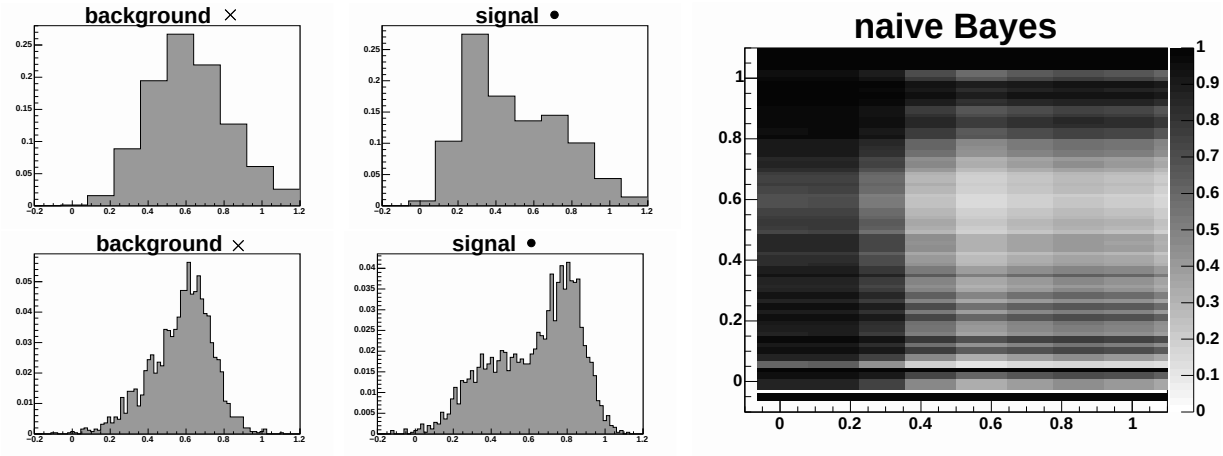

Figure 5.28: 'Gaussians': Naive Bayes training result – the one-dimensional histograms show the normalised distributions of both classes along the x-axis and y-axis, the output distribution on the right.

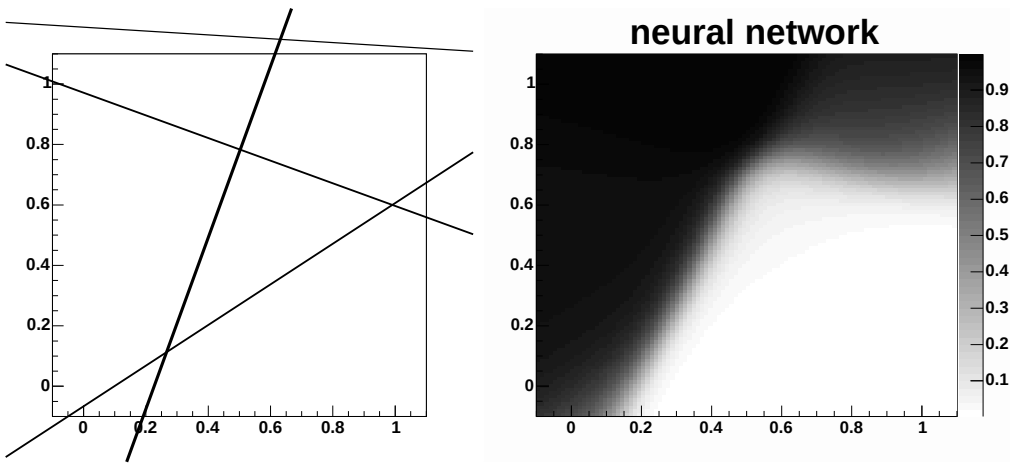

Figure 5.29: 'Gaussians': Neural network training result – the separating lines as expressed by the weight-vectors of the hidden neurons and the final output distribution.

kernel would help here for sure, but this means additional human intervention.

The random forest method is also one of the methods showing a behaviour on this dataset similar as for the 'Rings' example. This is because both require rounded or at least not axis-parallel decision boundaries. As seen in figure 5.31 the single trees are very noisy which is intended by the algorithm. But we also see that in the average among 200 trees there is still very much noise and the decision boundaries are by far not as smooth as they should be.

5.6.4 Summary of Typical Properties of Different Classification Methods

Taking into account the results from the three toy examples we want to summarise shortly some characteristic properties of the discussed learning methods:

- The simple cut method was not suited for the first two toy examples since each of

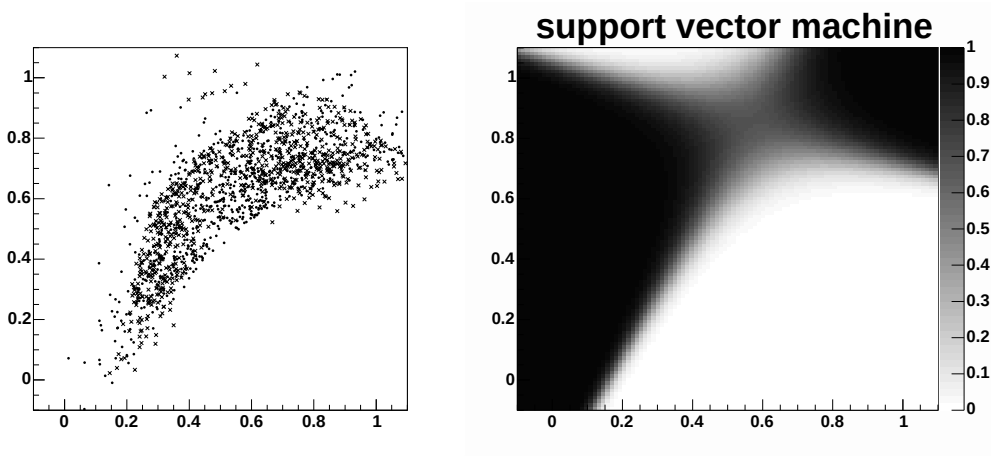

Figure 5.30: 'Gaussians': Support vector machine training result – the chosen support vectors and the final output distribution.

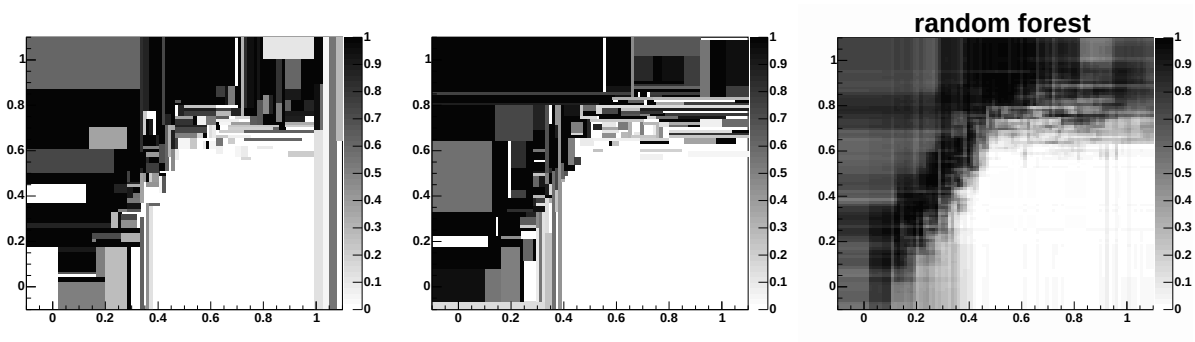

Figure 5.31: 'Gaussians': Random forest training result – two trees (left and middle) of 200 trees building up the final output distribution (right) by averaging.

them had some kind of enclosed region which could not be described with simple cuts. On the last example the simple cuts performed fairly well but the axis-parallel cuts have great difficulties with any kind of correlation.

- Naive Bayes, C4.5 and random forest often have the same problem as the simple cuts: They are based on axis-parallel cuts. If a structure with decision boundaries parallel to the axes is given (like the 'Hole' example), they perform well. Any kind of correlation or rounded decision boundaries are problematic for these three methods. The random forest can cope with these problems best due to its averaging strategy.

- k-nearest-neighbours and range search show the typical properties of local density estimators: They describe the decision boundary quite well, independent of its structure. They have no problems with correlations or rounded regions. However, the derived decision boundaries are always much noisier than they should be. Range search shows in addition the problem of the regions where no decision can be made due to missing training events, even if the classification would be simple.

- Neural networks have no problems with correlations and are especially well suited for piecewise linear decision boundaries. However, the approximation of rounded decision boundaries with several hidden neurons is not perfect.

- Support vector machines with a Gaussian kernel show a behaviour complementary to neural networks. The description of rounded decision boundaries works excellent with support vector machines. In contrast, linear decision boundaries are less well described and often lead to artefacts and thus to misclassifications.

We see that each of the learning methods prefers a specific kind of problem for which it then gives very good results. This closes the circle and brings us back to the discussion at the beginning of this chapter about the different kinds of biases which are implemented inside each learning method. As mentioned in section 3.15, the remaining question is which kind of bias is suited best for the datasets which will be faced in physics analysis. Chapter 7 will try to answer this question by comparing different learning methods on many different dataset from high energy and astrophysics experiments.

Chapter 6

Software Development

In this chapter the software framework will be described which was developed to allow the application and evaluation of many different statistical learning methods to all kinds of datasets from physics experiments. Also the toy problems discussed in the last chapter have been processed within this framework. On the one hand a large fraction of the software development depends on the data sources and therefore on the specific experiment. It is important to realise that the access to and the management of data coming from a large experiment is always a demanding and time consuming task. On the other hand there is also a large fraction of the software development which is independent of the specific data source.

Generally all implemented learning methods and their automation as well as the prior analysis and the posterior evaluation work independently of the underlying experimental situation. There is, however, again a part of the evaluation which does depend on the application since the quantity which should be optimised may be defined depending on the application.

The programs described in this chapter are mainly implemented in `C++` using its type-safe object oriented style and its fast executables. These programs make use of the ROOT library [97] mainly for the graphics tools and basic mathematical operations. Several scripts which operate on top of the basic programs are programmed in `perl` since this language provides an excellent interface to the shell and very comfortable text processing.

The framework of learning methods and the programs for the evaluation of learning methods contain about 15.000 lines of code. In contrast, the experiment dependent programs like data access and special performance evaluation contain about 55.000 lines. This discrepancy confirms the comments above that large parts of the work and thus also large parts of the programs have to be dedicated to the interaction with the specific experiment.

6.1 Data Access and Preprocessing

The first step towards the application of statistical learning methods to a new problem from physics analysis depends much on the hardware and software which is used in this experiment. Usually some kind of analysis chain already exists and has to be used to obtain the training data for a statistical learning method. Tools implementing an interface between the experiment dependent software and the framework of learning methods (which will be described below) have been created for each of the experiments described in this

thesis. They range from simply reading in text tables or binary data files to the usage of very complex software frameworks.

> Example: Data Access in the Large Software Frameworks of H1 and MAGIC
> The training data for the level 2 neural network trigger consists of the level 1 trigger quantities which are sent to the level 2 system for each event (see section 2.1.4). This data stream is stored on mass storage in a special "bank", unfortunately only for every second event due to a limitation in the readout system. Libraries in `fortran` and since recently also in `C++` allow the access to the H1 data files and in particular to the neural network data. Several hundreds of megabytes have to be processed to extract the neural network inputs for some hundred events.
>
> A large software framework called MARS copes with the data analysis for the MAGIC telescope. This `C++` framework has to be used to be able to extract detector data from the MAGIC data files. Depending on the abstraction level of the data (information from all pixels vs. Hillas parameters, compare section 3.9), typical file sizes are tens of megabytes to several gigabytes.

The preprocessing step also depends strongly on the experiment from which the data is taken. Some datasets may suggest only one specific input vector which is obtained in a trivial way from the available data stream. Much more often, however, the given detector data can be preprocessed in different ways including not only standards like normalisation but specific operations based on knowledge about the detector.

> Example: Preprocessing for the MAGIC telescope and the XEUS pixel-detector
> Substantial preprocessing was, for example, done for the MAGIC datasets since different sets of inputs were formed which depend on completely different levels of abstraction. The basic analysis presented in section 7.6 works on the level of the Hillas parameters. First steps towards an analysis based on pixel information have been presented in section 7.6.3.
>
> For the XEUS datasets the preprocessing step is essential since it can already filter out background. A framework which transforms uncalibrated data files into various analysis formats has been developed and will be presented in appendix B.1.

Additional tools have been developed which make typical data manipulation tasks easier. Among them one tool "squares" the input dimension (adding $x_i x_j \ \forall i, j$ to the inputs $x_1 \dots x_n$). Another one selects or removes specific inputs from a datafile, others help managing weights and IDs which are used to distinguish between training, selection and test sets.

6.2 Data Visualisation

Plots of the output distributions in two dimensions were already helpful in section 5.6 to understand the typical behaviour of different learning methods. It is also important to plot one- and two-dimensional distributions of the possible inputs of statistical learning methods for signal and background events to get a feeling which combination of inputs

will be suited best for the classification (like for example in section 7.2.3). Therefore tools to plot the input distributions in one and more dimensions and also in combination with already applied cuts have been developed.

6.3 Statistical Learning Methods and their Automation

Several of the statistical learning methods presented in chapter 5 were newly implemented while others have been incorporated into the framework by writing wrappers for existing implementations. The framework of learning methods used for this thesis consists of the following programs which have been developed "from scratch":

- The **simple cuts** procedure discussed in section 3.7 was easily incorporated into the framework.
- The **k-nearest-neighbours search** has been implemented according to its description in section 5.3.1, including the option of scaling factors in the Euclidian metric.
- The **range search** method has been implemented as described in section 5.3.3, including the possibility to weight events according to their distance from the evaluation position, and the possibility to use an adaptive box size.
- The **naive Bayes** method has been implemented as presented in section 5.3.4, but only for classification. The regression version was not implemented due to its inherent problems discussed in section 5.3.4. A specialised version was programmed for the neutron detector application (as described in section 2.3).
- **Linear discriminant analysis** has been implemented using the matrix inversion described in section 5.4.1.
- The **feed forward neural network** was programmed as described in section 5.4.2 using the classical back-propagation algorithm with some extensions regarding the automatic variation of the learning parameters. The back-propagation algorithm is described, for example, in [82, 83], the extensions implemented in this thesis are described in appendix B.3.
- The **random forest** method was implemented in three parts: The bagging procedure was programmed as described in section 5.11 and the splitting rule for classification is a `C++` copy of the original `fortran` code available on the Internet [98]. A splitting rule for regression was newly developed, it uses the variances as described in section 5.2.
- The meta learning strategies **bagging**, **random subspace** and **boosting** (sections 5.5.2, 5.5.3 and 5.5.4) have been programmed and can be applied to any of the basic learning methods.

The following learning methods have been incorporated into the framework using existing implementations which were freely available on the Internet:

- The decision tree **C4.5** can be downloaded from the Internet [99]. The source code fitted quite easily into the existing framework.

- For the **support vector machine** the library `libsvm` was used which can be downloaded from the Internet [100]. This source code was also easily combined with the existing framework. The only drawback with this implementation is the lack of support for weighted events. The existing Gaussian kernel was extended to support scaling factors per input instead of one global scaling factor. The Gaussian kernel was generally used since it performed well, the polynomial kernel was sometimes used as a cross check.

Two learning algorithms were downloaded and tried out since they have been praised to perform well: DIPOL, a hybrid of clustering and neural networks, and CAL5, a decision tree algorithm, both available on the Internet [101]. Unfortunately the source code of both methods was in such a bad condition that the incorporation into the existing framework was quite difficult. In addition, some basic studies showed quickly that they did not perform at all as well as it was advertised. They will therefore not be used in the comparative studies in the analysis chapter.

As described in section 3.11, the selection set is used to find the optimal balance between performance on the training set and generalisation among trainings with different parameter settings (different regularisations). Each of the learning methods mentioned above has some kind of free parameters which can be varied to control overtraining. These parameters have been discussed for each method in chapter 5. To arrive at a well performing learning method without too much human intervention it is important to automate the training and evaluation of different parameter settings.

In the implemented framework the variation of the training parameters has been automised for each learning method. The trainings with different parameter sets can automatically be done and evaluated in parallel which makes computing times less serious. Appendix B.3 gives an overview of the parameters which were varied for neural networks, support vector machines and random forests.

6.4 Performance Evaluation and Control

To be able to compare the different learning methods, common evaluation routines have to be used. For classification this amounts to calculating pairs of efficiency and rejection with the respective cut values (see section 3.12.1). For regression we calculate bias, variance and the total error in bins of the target value (as mentioned in section 3.12). All results are visualised and can also be overlaid for different methods to allow direct comparison. The statistical analysis necessary to compare the performance of learning methods and the assessment of the inputs which were used in the training are done in additional routines.

For applications which demand a performance measurement which is more involved than just calculating efficiencies and rejection rates typically some kind of significance is calculated with a program which depends on the specific experiment.

> Example: Performance Evaluation for the γ-hadron Separation (MAGIC) and for the Higgs Boson Parity Measurement
> After the γ-hadron separation the significance of the photon-excess is determined with the help of the α-plot as described in section 2.4. All events passing the γ-hadron separation are filled into the α-histogram and the significance is derived in quite a complicated formula from the relation between γ-excess for small α and the

background in this region. This formula takes into account the signal over background ratio as well as the relative counting errors for both numbers which get large for a very high background suppression.

The significance of the Higgs boson parity is derived from a large number of pseudo-experiments in each of which a small number of events is simulated as if the experiment was really performed. The mean significance together with its RMS can then be used to make a probability statement about the performance for the true measurement which will be possible when the linear collider will have been built.

Special programs which check the behaviour of statistical learning methods provide the control which is needed to make sure that the obtained results are trustworthy. Among them are the handling of the division into training, selection and test set, tools to calculate statistical and systematic uncertainties and programs which analyse efficiencies for example in dependence of an important observable.

Chapter 7

Analysis and Results

7.1 Overview

Chapter 3 showed that there are many specific topics concerning the application of statistical learning methods which are worth to be discussed in detailed examples. Furthermore, chapter 5 introduced a long list of learning methods which could in principle be applied to and compared on any problem.

Since not all presented methods can be compared on every physics dataset there will be local focuses for each analysis subject. The applied learning methods are motivated either by the problem itself, by the experimental situation, by the history of the analysis or simply by the preference of the analysers.

We will focus on the neural network learning method as this is one of the most heavily used methods in today's physics analysis and has become a standard. It is important to clarify whether this method deserves this special rank.

The discussion of problems which typically arise with the application of statistical learning methods will be distributed among the different applications. Different aspects from chapter 3 will be discussed for different physics cases in addition to the main physics results.

The following sections will present the analysis performed experiment by experiment. In chapter 8 a discussion will follow summarising the results presented here under different aspects which are important for the application of statistical learning in physics analysis.

The next section 7.2 will start with the neural network trigger of the H1 experiment. Many practical aspects like the determination of efficiencies, the choice of inputs, statistical and systematic uncertainties, the comparison of learning methods and many more will be discussed with the different datasets resulting from different types of *ep*-interactions to be triggered. Within this thesis, several new networks have been developed. Their good performance will be shown here as well as astonishing signs of artificial intelligence in the behaviour of neural networks. Most important, these neural networks have been deployed in the H1 experiment. They are active in the trigger system and help collecting physics events.

Section 7.3 continues with an offline analysis performed within the H1 experiment. In contrast to the event information provided by the trigger system, the events are fully reconstructed here so that high-level kinematical quantities can be chosen as inputs. The target of the analysis is to enrich certain classes of events in the experimental data – a typical pattern recognition task for statistical learning methods: standard perturbative

QCD events have to be distinguished from the instanton-like events. A comparison with the predicted number of events from a perturbative QCD simulation may then give a hint whether instanton-induced events indeed exist in the data.

In section 7.4 the possibility to determine the Higgs boson parity at a future linear collider will be analysed. Whereas the standard model Higgs boson has positive parity minimal super-symmetric theories predict also a Higgs boson with negative parity. In addition to the classical approach which fits a special angular distribution to determine the parity, a completely new and very successful method based on the direct discrimination of both parity states will be presented.

The reconstruction of the neutron incident position in the scintillation detector will be analysed in section 7.5. This task is an example for the typical problem of missing training data. How statistical learning methods can nevertheless successfully be used in the reconstruction process will be presented in this part of the analysis.

In section 7.6 two applications of statistical learning methods to MAGIC telescope data will be analysed. For the γ-hadron separation as well as for the energy estimation the results will clearly show that statistical learning methods perform better than the respective classical methods which are based on simple parameterisations.

Finally, section 7.7 will present the results for the second astrophysical analysis, for the XEUS satellite. Again a significantly better performance of statistical learning methods compared to the classical method results from the pileup rejection analysis. The reconstruction of the incident position of the X-ray photon can be used to study detector effects.

7.2 Applications for H1: The Level 2 Neural Network Trigger

Two classification problems will be discussed for the H1 experiment: The neural network trigger as one of the few existing online applications of statistical learning methods, and the search for instanton-induced events which will be discussed in the next section as an example for the need to purify an event sample as much as possible in the search for an especially rare event type.

The neural network trigger was described in section 2.1.4 and its hardware is discussed shortly in appendix A. In the following, recently developed networks for specific physics channels will be presented. The main target of these developments is to implement new level 2 triggers in the form of neural networks which reduce the rate of the level 1 trigger to an acceptable amount. As discussed in chapter 2, the maximum input rate to the fourth trigger level is the bottleneck of the trigger system since this rate is limited to totally (over all physics channels) 40-50 Hz because a full readout of the detector is needed for the level 4 trigger decision and this readout bandwidth is limited by the hardware. On the way to this implementation many of the issues common to all statistical learning methods as discussed in chapter 3 are faced. The network development thus offers the possibility to learn about subjects like generating training data, input selection, avoiding overtraining, controlling the learning method, measurement of the performance, calculating uncertainties and so on. Finally, studies on the comparison of alternatives to neural networks will be presented: Is there a classical algorithm or another statistical learning method which performs at least as good as neural networks and would it be possible to implement this algorithm in hardware to match the tight time constraints?

After two general comments about the generation of training data (section 7.2.1) and performance checks (section 7.2.2), the recent neural network trigger development will be presented for different physics channels in the following sections: DVCS in section 7.2.3, charged current in section 7.2.4, $J/\psi \to e^+e^-$ in section 7.2.5, $J/\psi \to \mu^+\mu^-$ in section 7.2.6 and D^*'s and dijets in section 7.2.7. In the end a summary of the newly developed neural network triggers will be given.

7.2.1 Training Data

The training data is generated in a generic way for all physics channels which will be discussed in the following sections. The physics ("good") events are selected for a specific physics channel (i.e. for a specific kind of *ep*-interaction) by a member of the corresponding physics working group. He or she knows all the details how such an event is characterised and thus how such a kind of event should be selected in an offline analysis. Very high-level algorithms are available for the analysis of the fully reconstructed events. It can thus be made sure that the training set will consist of a very clean sample of the desired kind of *ep*-interaction.

To characterise the background ("bad") events we have to remember that each level 2 trigger validates a specific level 1 trigger. Thus anything which is triggered by the selected level 1 trigger, but is not physics (compared to the selection of the good events), is labelled as background. Practically all events coming from the selected level 1 trigger can be regarded as background because the rate of real physics events is still extremely low. Because of this low rate of real physics the selection of good events covers usually a period

of several months of data taking while one "transparent run" (i.e. all events triggered by any level 1 trigger are logged ignoring the other trigger levels) of half an hour is enough to provide the background events.

7.2.2 Performance Check

Controlling the behaviour of a trigger, in particular the neural network trigger, is an indispensable task for deriving correct cross section measurements. Most importantly, the trigger efficiency must be measured precisely as the cross section σ is calculated by

$$\sigma = \frac{N}{\epsilon \cdot L} \tag{7.1}$$

where N is the number of remaining events after all cuts, L is the luminosity and ϵ is the total efficiency of all applied cuts, including all trigger efficiencies.

The usual way of monitoring trigger behaviour is implemented in the H1 trigger system (see figure 2.4) by logging a small fraction of events which the trigger would otherwise reject ("by-pass"). Unfortunately even a quite large fraction of by-pass events would usually contain much too few signal events to tell anything about the true efficiency (remember that the signal/physics events are typically hundreds to thousands in several months of detector operation while the trigger rate is about 100 Hz).

One strategy to evaluate the trigger efficiency uses so-called orthogonal triggers. This strategy requires that the same physics class is triggered by different triggers which should be as independent as possible. Suppose we have a level 1 trigger A for our physics class whose decision is verified by a neural network NN on level 2 and that we also have an independent level 1 trigger B for the same physics class. The efficiency of the neural network which acts on level 1 trigger A is then calculated as the fraction of selected physics events which were triggered by both A and B and have a positive decision from the neural network. Figure 7.1 illustrates this strategy: The basic assumption is that B is independent of A and that therefore the efficiency under the B-condition $\frac{A\&B\&NN}{A\&B}$ (which is measured), equals the true efficiency $\frac{A\&NN}{A}$.

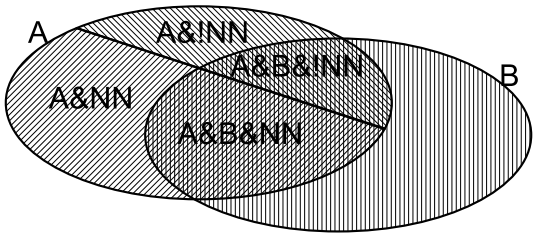

Figure 7.1: Efficiency studies with orthogonal triggers: If A and B are independent, then the efficiency of NN under condition A equals the efficiency of NN under condition A & B.

A very different approach to measure the trigger efficiency is to use a Monte Carlo simulation. Simulated events are usually numerous enough to obtain a precise measurement of the efficiency with a low statistical uncertainty. However, the drawback with simulated events is always the same: One has to understand the detector behaviour very precisely. Otherwise the efficiency obtained with the simulation may be strongly biased.

The key to the correct determination of the neural network trigger efficiency is the correct description of the level 2 inputs (coming from the level 1 trigger-systems). Usually much effort is needed to obtain distributions of the level 2 input quantities in the simulation which match those measured in real data. [102] presents an analysis where this

was successfully done. Once the one-dimensional distribution of each input is correctly described by the simulation, also the correlations between the inputs can be assumed to match. This assumption is based on the fact that the correlations, for example between tracks and energy deposits in the calorimeter, are determined by the underlying physics, i.e. by the particles which create the tracks and energy depositions.

The calculation of the neural network itself is no problem. It is "simulated" just by calculating the network output (compare section 5.4.2). Care must be taken only for the integerisation of the calculation performed in the hardware.

For the background rejection rate the monitoring is quite easy because in a good approximation all events coming into the neural network can be regarded as background as discussed above. This allows to calculate the background rejection simply by counting the number of events from the corresponding level 1 sub-trigger which pass the neural network, and the total number of events coming from the mentioned level 1 trigger.

In summary, the rate reduction and thus the background rejection is well controlled while the efficiency is problematic to handle. The efficiency calculated with the test set can be taken as a good estimation if the true efficiency does change with time. To check the trigger efficiency independently, the method of orthogonal triggers can be applied if such a trigger with a sufficiently large overlap exists. Finally a simulation can be taken to calculate the efficiency but the simulation of the level 2 trigger quantities is difficult.

7.2.3 Deeply Virtual Compton Scattering

Trigger Development

The first studies on the recognition of deeply virtual Compton scattering ("DVCS", see section 2.1.5) with the level 2 neural network trigger started beginning of 2003. As a first approach a selection of J/ψ events from 1997 were used as pseudo-DVCS events to train a neural network. The selected type of J/ψ events show an electron-positron pair from the decay of the J/ψ in *Track-Cluster* configuration where one of them is found in the SpaCal (cluster) and the other in the LAr calorimeter (with a track in the central tracking system). This configuration looks pretty much like DVCS where the photon also deposits its energy in the electromagnetic part of the LAr calorimeter and the beam electron/positron goes to the SpaCal. Figure 7.2 shows a DVCS event on the top and a J/ψ event on the bottom, their similarity concerning the energy deposits in the calorimeters is convincing.

However, the background which was used in this first study was the background of the J/ψ events (taken from a transparent run in 1997). Since the level 1 sub-trigger for J/ψ's does not only require the energy in the SpaCal but also at least one fired mask in the tracking system (which cannot be generated by the photon in DVCS), the background is different. Nevertheless the achieved pair of 93% signal efficiency and 90% background rejection for a test training of neural networks with this data was already a hint that the task is manageable.

The next steps in triggering DVCS have been taken with "real" DVCS events, first with a selection from the year 2000 and finally in May 2004 with a DVCS selection from 2004. The event selection is done by requiring an electromagnetic cluster in SpaCal above 15 GeV and one in the electromagnetic part of the LAr calorimeter above 1 GeV. No other energy deposition above the noise level (0.5 GeV) in the LAr and less than 2 tracks per event are allowed.

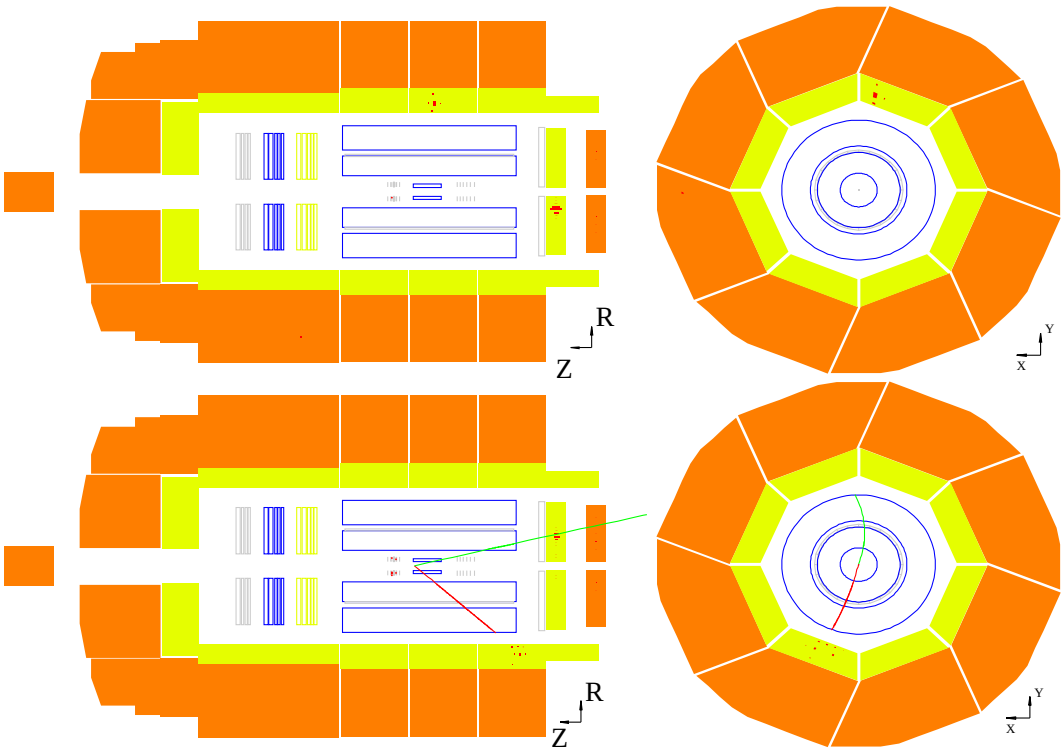

Figure 7.2: Deeply virtual Compton scattering (top) compared to a J/ψ decaying into two electrons in *Track-Cluster* configuration (bottom).

For the final network which was developed in May 2004 the inputs consist of the twelve LAr quadrant energies `eifq0-3, efbq0-3, ecbq0-3`, the four SpaCal quadrant energies `espq0-3` and the z-vertex quantities `cpvsum`,`cpvmax` and `cpvpos` (compare table 2.1). It was verified that the track quantities `trlopos,trloneg,trhipos,trhineg` as additional inputs do not improve the classifier. Figures 7.3, 7.4 and 7.5 show the distributions of the chosen inputs for the DVCS selection (black) and the competing background from level 1 sub-trigger 41 (grey). The following observations connect the observed distributions with the underlying physics:

- As expected, the four quadrants of each region of the LAr calorimeter and of the SpaCal show very similar behaviour due to the rotational symmetry of the events.
- The inner forward region of the LAr calorimeter (figure 7.3 a-d) gives a clear indication of background formed by upstream beam-gas interactions. All particles produced by the collision of a beam proton with a gas nucleus are seen in the inner forward region of the LAr calorimeter. For DVCS events this region is almost always empty.
- The photon is emitted mostly to the central region of the LAr calorimeter, sometimes to the forward region. Accordingly similar energies for signal and background are seen in the forward region but the signal is dominant in the central region (figure 7.3 e-h and figure 7.4 a-d).
- The integral and the maximum of the z-vertex histogram scale with the number of track candidates which are usually less for DVCS events than for the background (figure 7.5).

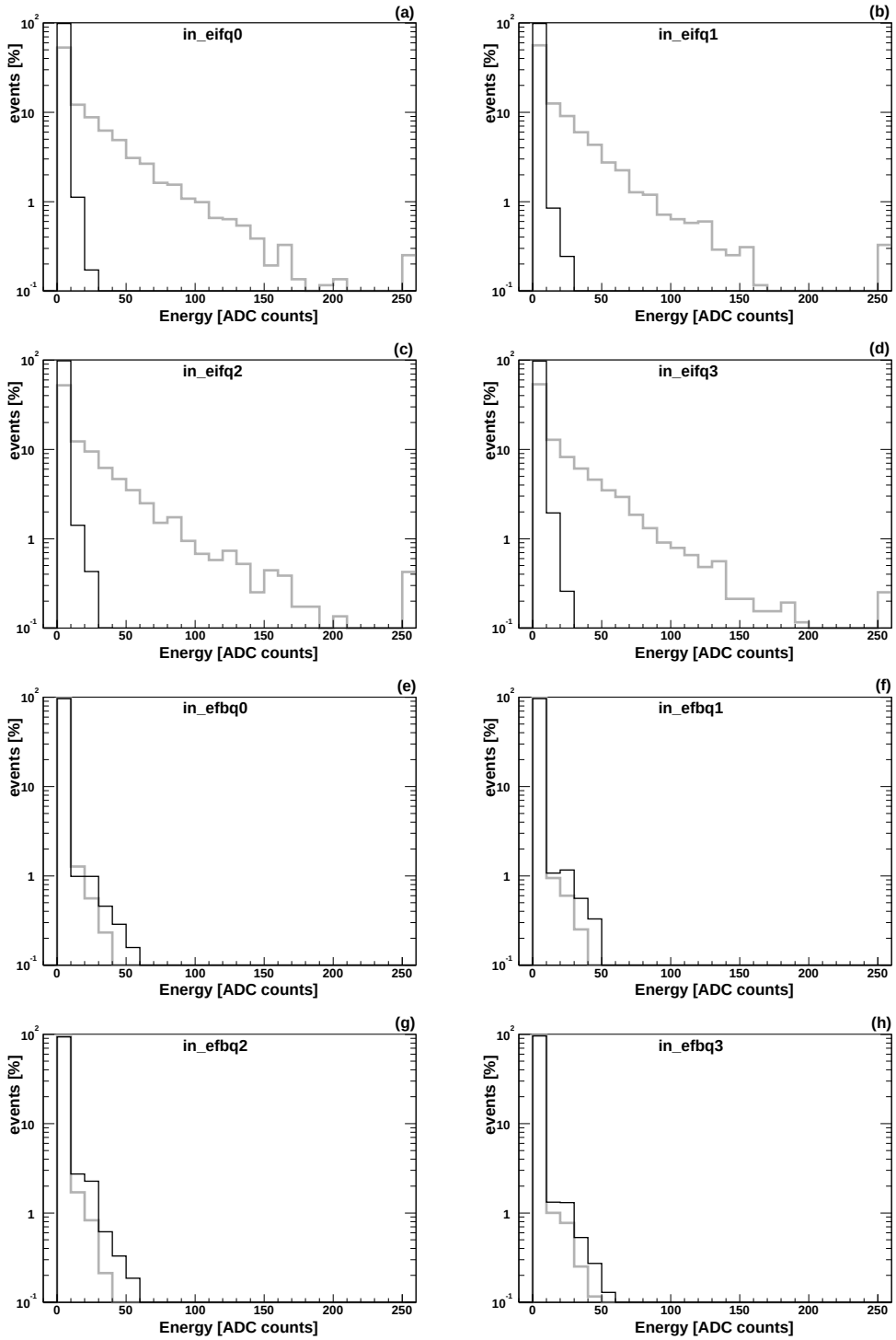

Figure 7.3: Distributions of trigger quantities for DVCS events (black,thin) vs. background (grey,bold): energy deposits in the eight quadrants from the inner forward (`if`, a-d) and forward region (`fb`,e-h) of the LAr calorimeter. 10 ADC counts correspond to $10GeV$.

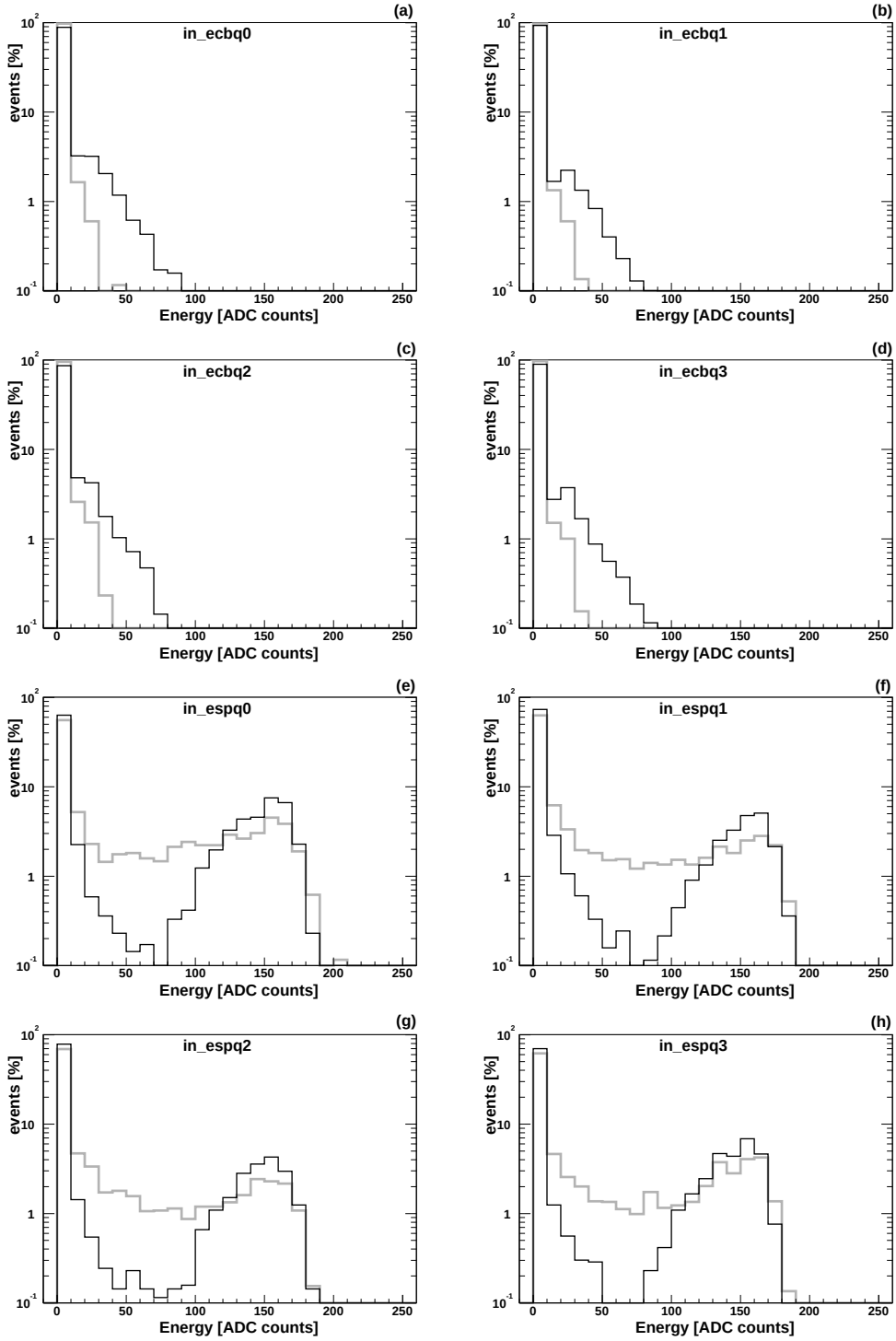

Figure 7.4: Distributions of trigger quantities for DVCS events (black,thin) vs. background (grey,bold): energy deposits in the four quadrants from the central region (`cb`, a-d) of the LAr calorimeter and in the four quadrants of the SpaCal (`sp`,e-h). In the SpaCal 100 ADC counts correspond to $18GeV$.

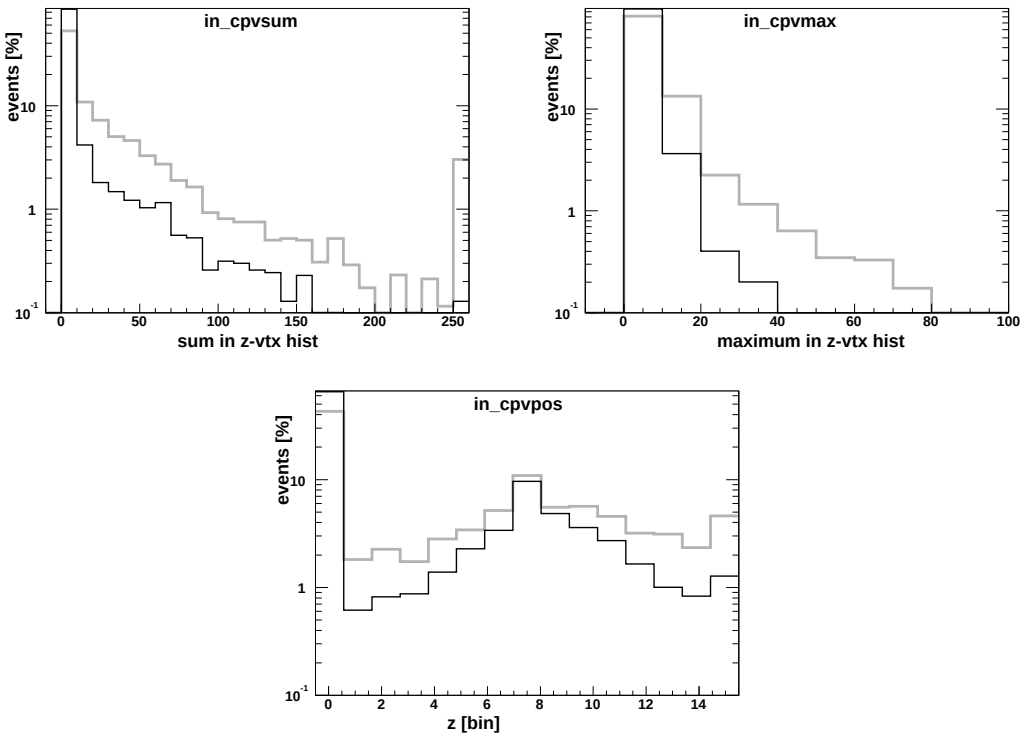

Figure 7.5: Distributions of trigger quantities for DVCS events (black,thin) vs. background (grey,bold): z-vertex quantities which give the sum of entries in the z-vertex histogram (`cpvsum`), the maximum of the z-vertex histogram (`cpvmax`) and the position of the maximum (`cpvpos`).

input	$\rho(\texttt{input}, \texttt{target})$
eifq0	-0.4132
eifq1	-0.3962
eifq2	-0.4116
eifq3	-0.4100
efbq0	+0.1653
efbq1	+0.1215
efbq2	+0.1552
efbq3	+0.1463
ecbq0	+0.0439
ecbq1	+0.0501
ecbq2	+0.0835
ecbq3	+0.0456
espq0	+0.0324
espq1	+0.0057
espq2	+0.0145
espq3	+0.0157
cpvsum	-0.2638
cpvmax	-0.2651
cpvpos	-0.2093

Table 7.1: Correlation coefficients $\rho(\text{input}, \text{target})$ for all inputs of the DVCS dataset.

The observed relationships between signal and background for each input are confirmed by the correlation coefficients for the correlation between each input and the target value (equation 3.1 on page 43) shown in table 7.1. A negative value means that a high value is likely for background, a positive correlation means that the signal has usually higher values and a correlation around zero means that from this input alone no clear dependence can be derived. But still inputs with a correlation around zero may contribute much to the final multidimensional classification since here only projections of the multidimensional distribution are used (compare the discussion in section 3.7).

The division of the available data into training, selection and test set was done with the ratio 50% : 25% : 25% as proposed in section 3.11. Table 7.2 shows how many events went into which group. With 7000 DVCS events and 5000 background events this dataset has large statistics compared to other physics channels. This allows on the one hand to loosen the regularisation constraints because overtraining is not too likely to happen. On the other hand the statistical error for the designed efficiency and rejection will be small.

	background	DVCS	sum
train	2570	3494	6064
select	1292	1745	3037
test	1315	1737	3052
sum	5177	6976	12153

Table 7.2: The DVCS-dataset: DVCS-selection from 2004 and transparent run (background) from February 2004.

The rate of the level 1 sub-trigger 41 for DVCS was between 8 and 9 Hz, too high for an allowed rate budget of about 2 Hz for trigger level 4. A background rejection on level 2 should therefore reduce the rate by a factor of 5, i.e. with a rejection rate of 80%. Multiple neural networks were trained as described in appendix B.3.1 and the best efficiency for a rejection of 80.0% was found to be 96.0% on the selection set for a network with 20 hidden neurons. The performance on the training and test set is shown as histograms of the output and efficiency vs. rejection graphs in figure 7.6. The target rate reduction of 80% leads to a designed efficiency of 97.0% (now measured on the test set) at the "working point" with a cut at 0.23. Since beginning of June 2004 this network is operating successfully.

As mentioned in the introduction of this analysis section, the performance check for the neural network trigger is easy for the background rejection: The monitoring system of the neural network trigger allows to count the fraction of rejected events and this fraction is a good estimate of the background rejection as discussed above. Figure 7.7 shows the obtained background rejection for real data coming from the detector. The designed rejection rate of 80% is confirmed.

The signal efficiency is more complicated than the background rejection. To verify the designed DVCS efficiency of 97% the level 1 sub-trigger 17 is used. This trigger is on level 1 identical to sub-trigger 41 but has a different verification on level 2: For sub-trigger 17 a "topological" trigger based on the LAr energies is used. DVCS events have been selected for the period where the neural network was active and rejecting background. 85% of all DVCS events found were triggered by both level 1 sub-triggers 17 and 41. For 99.6% of these events the neural network operating on sub-trigger 41 kept the event, only for 0.4% the event would have been rejected if it would not have been triggered also by sub-trigger 17. Under the assumption of an "orthogonal sub-trigger" 17 (compare the discussion above) this result confirms the designed efficiency (it is even slightly higher than estimated). Figure 7.8 shows the efficiency dependence on energy and angle of the

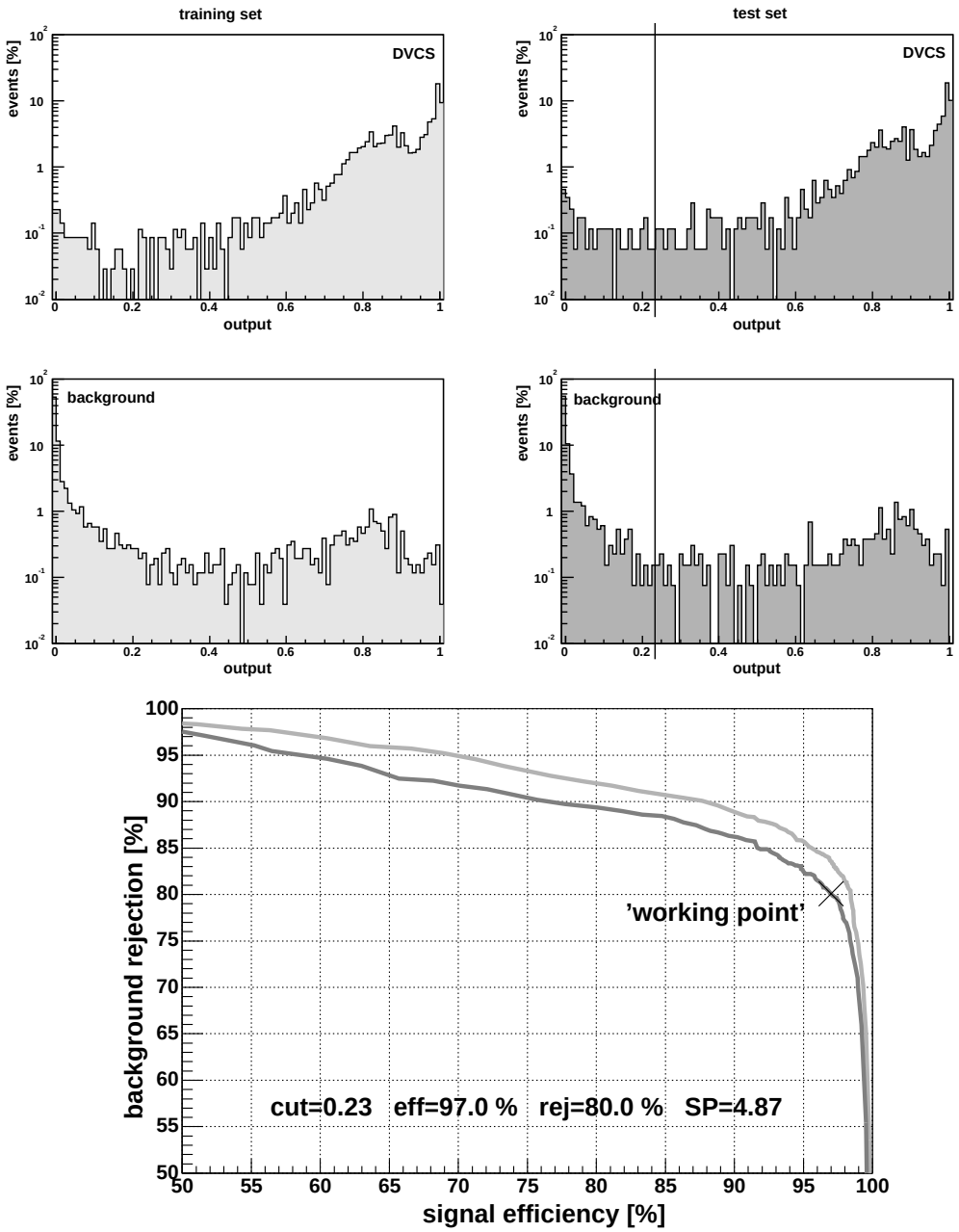

Figure 7.6: Output distributions and efficiency vs. rejection graph for the DVCS network: The light grey histograms show the output on the training set (top left) and result in the light grey curve in the efficiency vs. rejection plot. The dark grey curve shows a slightly worse performance as it comes from the output distributions on the test set (top right). The term signal means here the DVCS selection, the background comes from level 1 sub-trigger 41. The cut at 0.23 is shown in the output histograms of the test set.

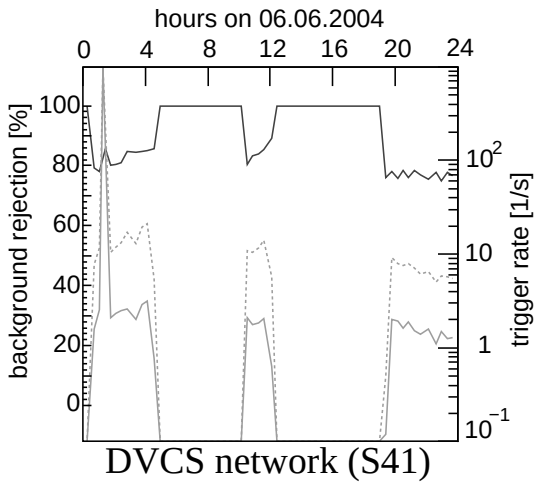

Figure 7.7: Online monitoring of the DVCS-network: The background rejection (dark grey) is around 80% as calculated from the reduction of the level 2 input rate (light grey, dotted) to the level 2 output rate (light grey, solid). In the display of 24 hours 3 data taking periods are visible (there have been no beams from 4 to 10 and from 12 to 19).

photon in the LAr calorimeter and of the electron in the SpaCal. All four plots show a flat efficiency, taking into account the statistical errors. This means that no bias is introduced by the neural network which could make an analysis of the triggered DVCS events difficult.

In summary the DVCS network with its rate reduction by a factor 5 and almost no loss in efficiency is a big success in comparison to the alternative pre-scaling. This would mean that the efficiency is only 20% for a factor 5 rate reduction.

Two targets for an analysis which goes beyond the development of the neural network trigger will be discussed in the next two sections. First the statistical and systematic uncertainties of the efficiency of the implemented neural network will be analysed and secondly a search will be performed for any method which performs similar to the implemented neural network and which can cope with the tight time constraints.

Statistical and Systematic Uncertainties

The way how the systematic uncertainty of the signal efficiency can be calculated in principle was introduced in section 3.13. It was emphasised that the correct calculation of the uncertainties is an important step for the application of statistical learning methods in physics analysis. The DVCS dataset will act as an example of how this calculation is practically done.

The principle to calculate the systematic uncertainty of the output is to propagate the systematic uncertainties of the inputs through the network. The first step is therefore to determine the systematic uncertainties of the inputs. For the calorimeter energies we adopt the known uncertainties of the calibration: A relative uncertainty of 4% for the LAr energies and 1% for the SpaCal energies. All inputs from the same calorimeter (12 from LAr and 4 from SpaCal) depend on the calibration in the same way and are thus correlated. The correct systematics is therefore derived from a common variation of all input energies for each calorimeter up and down by 1σ.

The three z-vertex quantities are derived from the z-vertex histogram which itself is filled by the projections of all hit combinations from the outer and inner z-chambers. The two systematic effects affecting this histogram are a variation in the efficiency and in the position along the z-direction. They lead to a change in each bin content (up/down) and to a shift of the entries (left/right), respectively. The uncertainty in the bin heights is estimated as 4% (from the efficiency of 96% which is typically assumed). The systematic

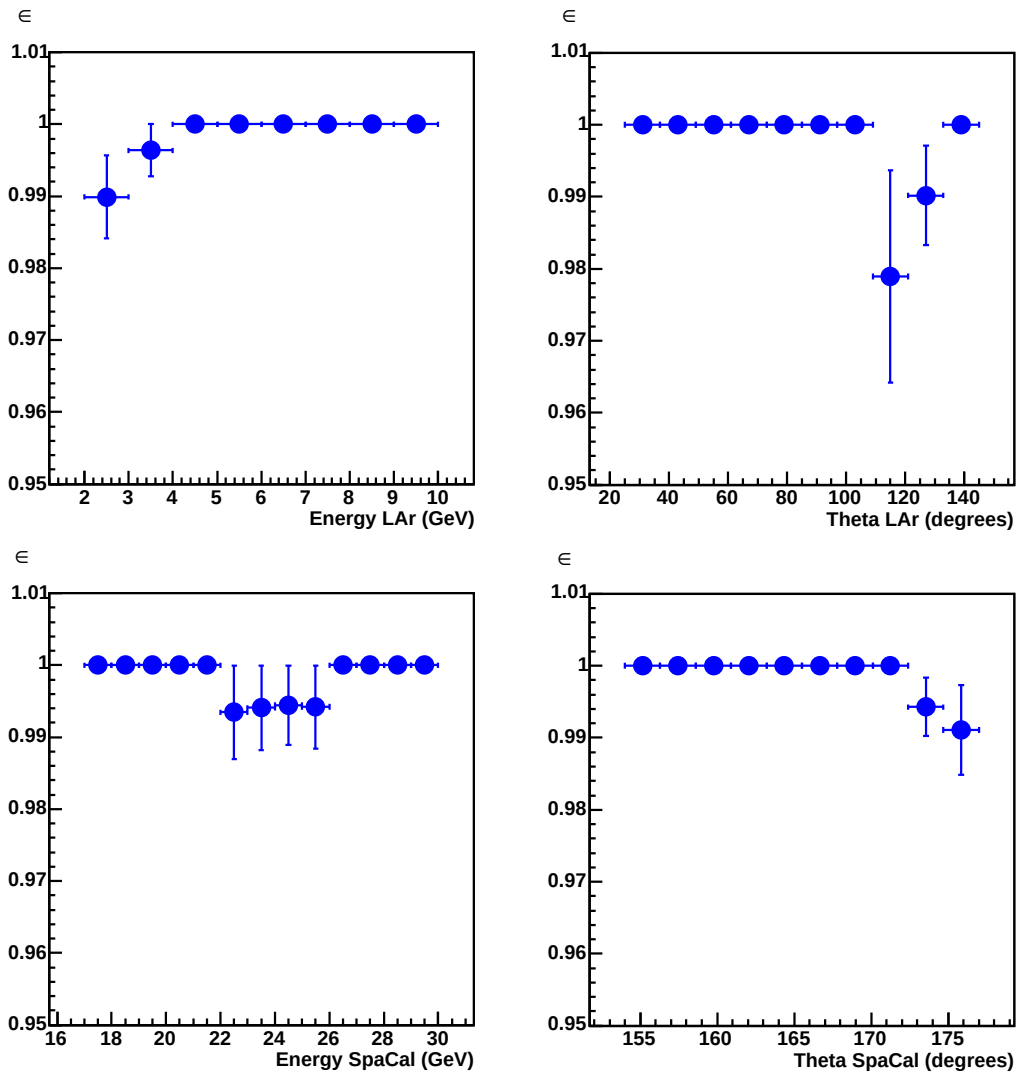

Figure 7.8: Efficiency check for the DVCS-network: Dependence of the efficiency on the energy- and angular distributions in the LAr calorimeter (photon) and SpaCal (electron).

uncertainty in the bin shift is estimated as a quarter of a bin ($1.4cm$).

To study the propagation of the input uncertainties to the output uncertainty the above estimates will be taken as a basic medium version of uncertainties. In addition a "low" and "high" version of uncertainties will be studied where all mentioned values are halved or doubled, respectively, as shown in summary in table 7.3. This will allow to estimate the general degree of influence of systematic uncertainties on the network behaviour.

Variable	uncertainty level		
	low	medium	high
LAr calibration	2%	4%	8%
SpaCal calibration	0.5%	1%	2%
z-vtx hist. shift [bins]	0.125	0.25	0.5
z-vtx efficiency	2%	4%	8%

Table 7.3: Sources for systematic uncertainties for the DVCS dataset. Three scenarios are displayed (low, medium, high), the medium scenario uses the standard uncertainties determined by the H1 experiment.

The test set used to derive the systematic uncertainties is different from the original test set since the data had to be reprocessed in between[1]. Efficiency and rejection are almost the same as for the original test set but the number of available events is lower: 1182 background and 881 signal events (compare to table 7.2). The statistical uncertainty of the efficiency presented below is therefore about 40% larger.

The systematic uncertainties of the inputs propagate to efficiency and rejection as shown in table 7.4. The three tables contain the effects on efficiency and rejection in terms of the strength of the variation from "low" to "high". Inside each table variations upwards and downwards are performed for the four sources of systematic uncertainties.

It is interesting to see that there is almost no degradation in the efficiency. If at all, a negative effect can be seen in a lower rejection. All variations seen in efficiency and rejection, even for the doubled uncertainties ("high"), are still smaller than the statistical uncertainties induced by the low number of test events. Table 7.5 summarises the total uncertainties for efficiency and rejection for the DVCS neural network. The systematic uncertainties even in the very conservative case are still below the statistical uncertainties. The "fault tolerance" of neural networks leads here to the effect that uncertainties of a few percent for the input quantities are transformed into uncertainties less than one percent for the efficiency. In summary, the neural network behaves very stable and reliable as the statistical and systematic uncertainties are well under control.

Comparison of Hypotheses

In order to take a first step towards the question which statistical learning method is optimally suited for application in physics analysis we will start here with a comparison of those methods that can cope with the tight time constraints given by the level 2 trigger system. For a few methods it is obvious that they are fast enough to be calculated in hardware within the H1 requirement of 10 μs. Among these are the naive Bayes method, linear discriminant analysis (LDA) and, of course, simple cuts. Neural networks and support vector machines also have a low time consumption due to their parallel architecture (as long as the *hidden* units are few enough to be calculated in parallel). Hardware imple-

[1]The z-vertex histogram quantities have not been available in the original dataset and are now needed here to get the uncertainties of the three derived quantities.

"low"		
variation	eff. [%]	rej. [%]
without	97.6	79.4
LAr +	97.6	79.4
LAr −	97.6	79.4
SpaCal +	97.7	79.3
SpaCal −	97.7	79.4
z-vtx shift right	97.6	79.2
z-vtx shift left	97.7	79.2
z-vtx efficiency +	97.6	79.3
z-vtx efficiency −	97.6	79.3
"medium"		
variation	eff. [%]	rej. [%]
without	97.6	79.4
LAr +	97.6	79.3
LAr −	97.8	79.2
SpaCal +	97.7	79.0
SpaCal −	97.7	79.5
z-vtx shift right	97.6	79.3
z-vtx shift left	97.7	79.1
z-vtx efficiency +	97.6	79.0
z-vtx efficiency −	97.6	79.3
"high"		
variation	eff. [%]	rej. [%]
without	97.6	79.4
LAr +	97.4	79.6
LAr −	98.0	78.5
SpaCal +	97.7	78.7
SpaCal −	97.7	79.8
z-vtx shift right	97.7	79.3
z-vtx shift left	97.6	79.2
z-vtx efficiency +	97.6	79.2
z-vtx efficiency −	97.6	79.4

Table 7.4: Evaluation of systematic uncertainties for the three different levels of underlying systematic uncertainties for the DVCS network. The medium level corresponds to the standard uncertainties determined by the H1 experiment. The efficiencies and rejections are calculated with the same neural network and the same cut as for the original test set. For each uncertainty level a variation upwards and downwards for all four sources of systematic uncertainties are taken into account. These sources are assumed to be independent so that the differences in efficiency or rejection can be added up in quadrature.

Efficiency [%]	97.62	±0.51 (stat.)	− 0.00 + 0.20 (syst.,low) − 0.00 + 0.30 (syst.,medium) − 0.23 + 0.39 (syst.,high)
Rejection [%]	79.36	±1.18 (stat.)	− 0.28 + 0.08 (syst.,low) − 0.59 + 0.17 (syst.,medium) − 1.11 + 0.49 (syst.,high)

Table 7.5: Total uncertainties for the three different levels of underlying systematic uncertainties for the DVCS network.

mentations will be discussed in appendix A and for now these five methods are assumed to be fast enough.

For the very basic LDA method which has very few parameters, a larger input space can be allowed. The input space is "squared" by adding $x_i x_j \ \forall i, j$ to the inputs $x_1 \dots x_n$. This generates in total 209 inputs which are then used for the linear discriminant analysis. For the other four methods the original 19 inputs are kept.

Figure 7.9 shows a comparison of the five fast classification methods. The methods were trained with different parameter sets (except LDA) and the best efficiency for a rejection rate of 80% was selected with the selection set. The efficiency vs. rejection graphs shown in the figure present the performance on the test set. Linear discriminant analysis clearly profits from the additional inputs. With only the usual 19 inputs its performance would be worse than for naive Bayes but still better than for simple cuts.

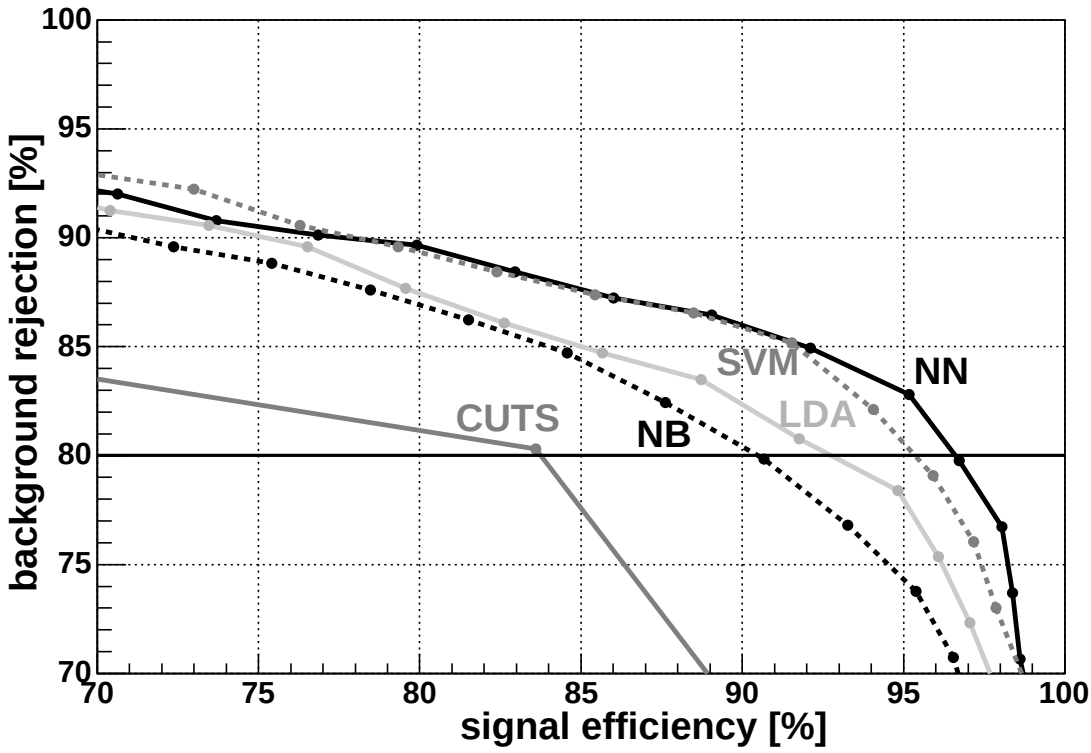

Figure 7.9: Comparison of fast classification methods, neural network (NN), support vector machine (SVM), linear discriminant analysis (LDA), naive Bayes (NB) and simple cuts (CUTS). The selection criterion was the best efficiency at a rejection of 80%.

The statistical analysis needed to compare different hypotheses as described in section 3.15 is performed here with the five chosen methods. 95% confidence intervals are constructed for each difference in the performance-ordered list. The result is shown in table 7.6. Exclamation marks signify a significant performance difference (95% CL) between two methods in the list which is sorted by efficiency for a required rejection of 80%.

Table 7.6 reveals four groups of significantly different performance. The neural network and the support vector machine perform best, then follow linear discriminant analysis and naive Bayes, and the simple cuts perform worst. As discussed in section 3.15 this is not a

Neural Network	96.5%
$\Delta = 0.7\% \pm 0.9\%$	
Support Vector Machine	95.7%
$\Delta = 3.3\% \pm 1.5\%$ (!)	
Linear Discriminant Analysis	92.5%
$\Delta = 2.2\% \pm 2.1\%$ (!)	
Naive Bayes	90.5%
$\Delta = 6.8\% \pm 2.5\%$ (!)	
Cuts	83.6%

Table 7.6: Statistical analysis of the efficiencies at 80% rejection for different methods on the DVCS dataset, statistically significant differences are marked by an exclamation mark (95% CL).

full comparison of learning methods, but only of these specific five hypotheses. Nonetheless it is encouraging that the neural network hypothesis – which was really implemented in the hardware and has been used to select events – is one of the best classifiers available for this problem. From the worst to the best classifier, from simple cuts to the neural network, the percentage of lost events (inefficiency) is reduced by a factor 4.7, which underlines the necessity to study and choose among different classification methods.

7.2.4 Charged Current Interactions

Trigger Development

In the following the rate reduction of the level 1 trigger rate for charged current ("CC") reactions (see section 2.1.5) is studied. The chosen level 1 sub-trigger 77 sets a medium threshold for the missing transverse energy in the LAr calorimeter and showed a trigger rate low enough to be passed on to the fourth trigger level without pre-scaling (at most 1 Hz) before the HERA 2 upgrade. The higher trigger rates after the upgrade may make it necessary to reduce the level 4 input rate, rates of up to 2 Hz have already been reached.

There is a particular problem with charged current events which may also be faced in many other applications for statistical learning methods: The number of training events is very small. For an integrated luminosity in 2004 of $21pb^{-1}$ and $25pb^{-1}$ for negatively and positively polarised positrons[2], respectively, and a cross section of roughly $14pb$ and $36pb$, respectively, one can expect around 1000 events per year.

Like for the first try for the DVCS problem also in this case some kind of pseudo signal events could help. As discussed in section 2.1.5, charged current interactions are closely related to neutral current interactions. The main difference from the detector point of view is the electron in the final state which creates a track and an energy deposition in the calorimeter for neutral current interactions. For charged current interactions the incoming electron/positron is converted into an (anti) neutrino which leaves the detector unobserved. The hadronic final state is expected to be identical in neutral current and charged current reactions. This means that a neutral current event where the electron is masked out, can be used as a "pseudo charged current" event. Neutral current events are fortunately very numerous (compare the discussion in section 2.1.5).

The conversion of neutral current events into charged currents by removing the electron information from the detector is done by a program which even converts the level 1 trigger quantities. This is important to be able to determine the efficiencies on L1 for charged

[2]The polarisations were about -40% ("left handed") and +34% ("right handed").

current events. Unfortunately the masking of the electron is not yet working for the quantities used on the second trigger level.

As an alterative we make a selection of inputs which hides the differences between neutral and charged current. This means that the `FB` and `CB` region of the LAr calorimeter cannot be used as inputs because there the electron energy is usually seen. At this point it was unclear whether the track of the electron has a significant influence on the trigger quantities. All track quantities were used as they are. It turned out, however, that the number of tracks in the backward region (`nbigbwd`) differs between pseudo and real charged current. Figure 7.10 shows that for charged current the probability for no track in the backward region is around 50%. In contrast, for neutral current this probability is only 12% due to the electron in the final state. Nevertheless, the following analysis shows that this difference is not essential for the network behaviour: a sign of the "fault tolerance" of the neural network.

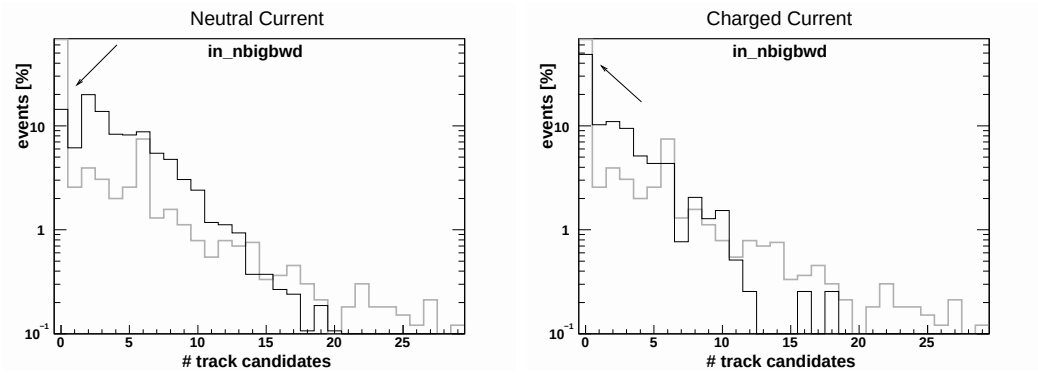

Figure 7.10: Track candidates in the backward region for neutral current (left) and charged current (right). The probability of no track candidate (first bin) is 4 times higher for charged current due to the missing electron in the final state.

The choice of inputs contains therefore any available quantity which is related to charged current interactions (excluding the SpaCal and the muon quantities) and which is not directly related to the electron (which excludes the two regions of the LAr calorimeter). This leaves four LAr quadrants in the inner forward region `eifq0-3`, four drift chamber track quantities `trlopos,trloneg,trhipos,trhineg`, four multi wire proportional chamber track quantities `nbigfwd,nbigfce,nbigbce,nbigbwd` and the z-vertex quantities `cpvsum,max,pos` as inputs (figures 7.11 and 7.12).

Table 7.7 summarises the data sets which are available to train and test a neural network trigger for charged current interactions. The training, selection and estimation of performance is done with the large sets of pseudo charged current events and background. The estimated efficiency and rejection can then be cross-checked with real charged current events (and the background test set). The division into training, selection and test set was done 60% : 12.5% : 27.5% to have more events for training and testing.

The optimal relation between efficiency and rejection ("working point") for charged current is different from the previously discussed DVCS dataset. Here, a very high efficiency, as close to 100% as possible, is the primary objective and the corresponding rejection rate is secondary since a high rejection is not required. About a factor 2 rate reduction, i.e. around 50% rejection, is desired.

The trainings were done using different parameter settings. A quick overview showed

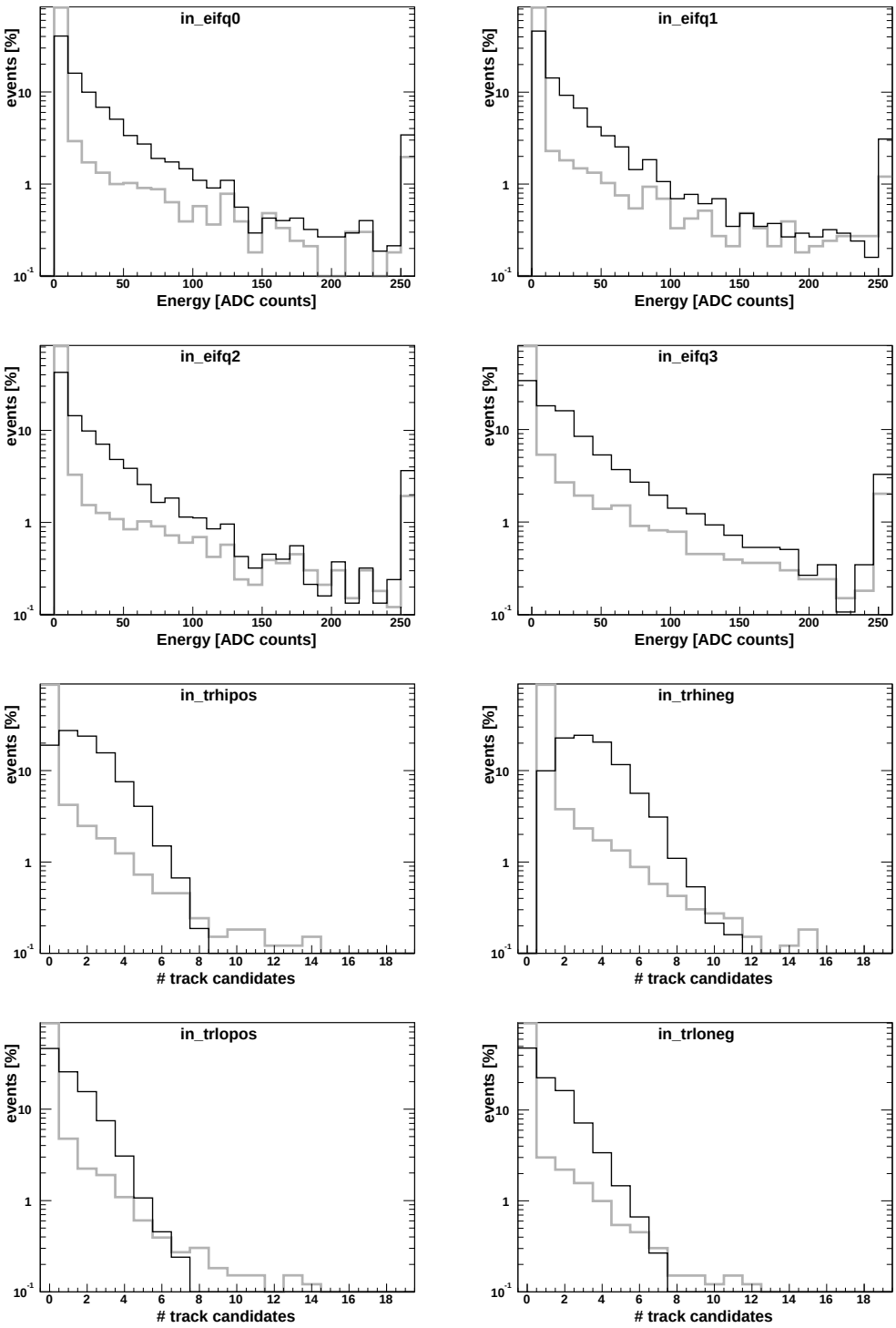

Figure 7.11: Distributions of trigger quantities for pseudo-CC events (black,thin) vs. background (grey,bold): energy deposits in the four quadrants from the inner forward region of the LAr calorimeter and track candidates from the drift chamber.

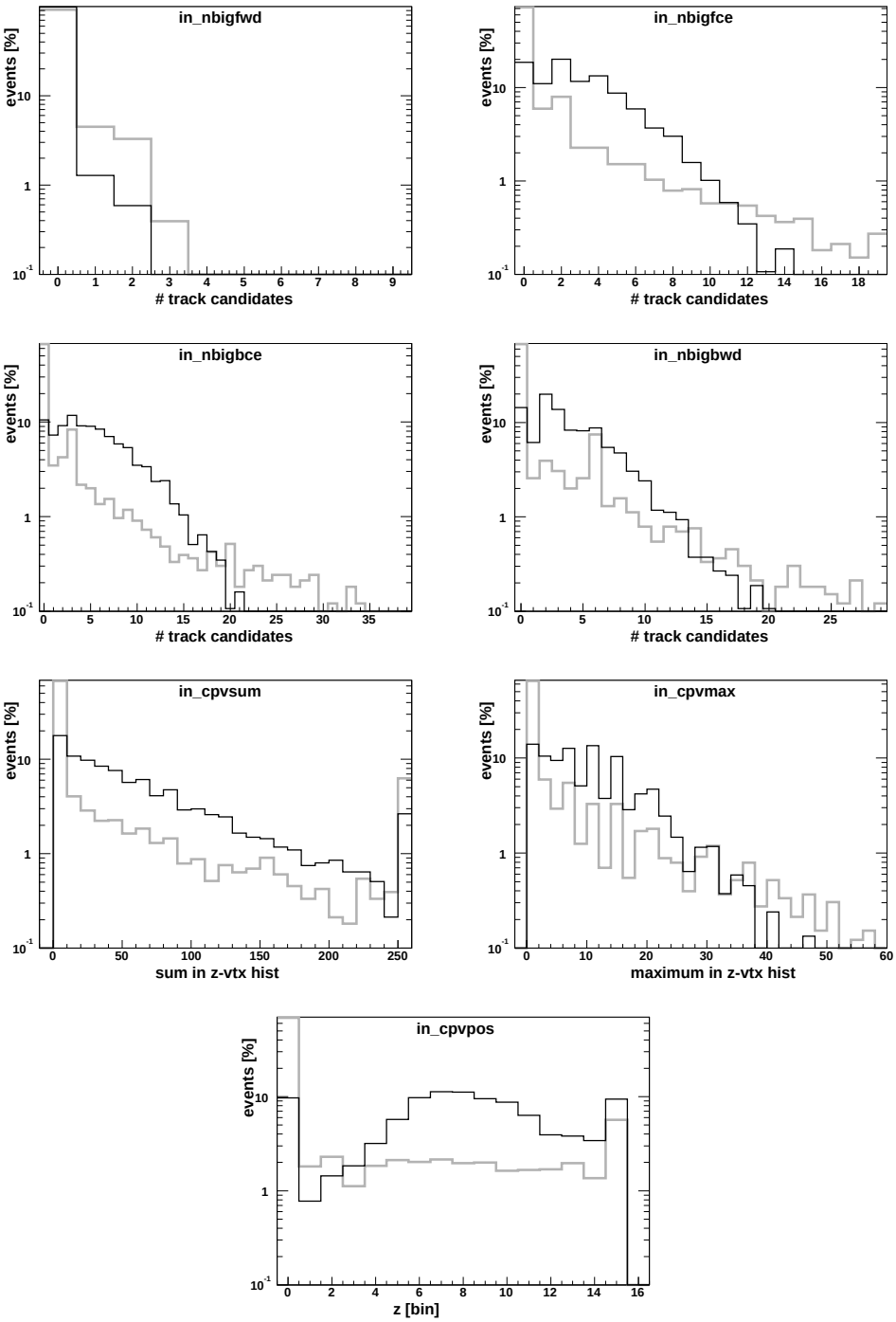

Figure 7.12: Distributions of trigger quantities for pseudo-CC events (black,thin) vs. background (grey,bold): tracks from the z-vertex trigger in four regions (`nbigfwd` to `nbigbwd`) and the z-vertex quantities which give the sum of entries in the z-vertex histogram (`cpvsum`), the maximum of the z-vertex histogram (`cpvmax`) and the position of the maximum (`cpvpos`).

	back-ground	pseudo-CC	sum	real CC
train	1985	2258	4243	
select	415	425	840	
test	909	1063	1972	391
sum	3309	3746	7055	391

Table 7.7: The CC-datasets: Neutral current selection from 2004 used as pseudo charged current vs. transparent runs from April 2004 and real charged current events used for testing only.

that efficiencies above 99% can be obtained with a rejection rate above 50%. The highest efficiency was thus searched on the selection set for a fixed rejection of 50%. The best efficiency found for the selection set was 100% (no signal event lost) for a network with 12 hidden neurons. On the test set, the corresponding network shows the performance presented in figure 7.13. An efficiency for pseudo charged current interactions of 99.9% (1 of 1063 pseudo-CC events gets lost) is achieved with a background reduction of 58%.

This neural network was integrated into the trigger hardware to check the rejection rate online on new events. Figure 7.14 shows the graphs for input rate, output rate and their quotient, the background rejection, coming from the monitoring system of the level 2 neural network trigger. The background rejection of about 50% is slightly below the designed 58%, but still in good agreement since exact equality cannot be expected due to changing running conditions.

To verify the designed efficiency the 391 events from the charged current selection were used. These events have not been included in the training and can thus serve as a test set. Figure 7.15 proves the success of the pseudo charged current replacement. The classifier looses only one out of the 391 real charged current events. Why and what is special about this event will be discussed after the determination of the uncertainties of the network.

Statistical and Systematic Uncertainties

The determination of the statistical and systematic uncertainties proceeds according to the scheme introduced for the DVCS network in section 7.2.3. Two groups of input quantities are used in the charged current network which have not been used in the DVCS network. Both groups describe the numbers of track candidates, one with information from the z-vertex trigger (`nbig*` quantities) and one with information from the DC$r\phi$-trigger (`tr*` quantities).

The uncertainty for the "big-ray"-quantities is coupled to the z-vtx histogram height since both depend on the track efficiency of the z-vertex trigger (CIP and COP). The systematic variation of this track efficiency therefore affects the z-vertex histogram as well as the number of tracks (`nbig*` quantities).

The uncertainty for the DC$r\phi$ track efficiency is estimated to be similar to the track efficiency of the z-vertex trigger. The same medium level of uncertainty of 4% is therefore assumed. Table 7.8 summarises the sources of systematic uncertainties which are taken into account.

The statistical errors are based on the whole real charged current test set and a part of the background set which again had to be reprocessed to include the z-vertex histogram quantities. A total of 391 real charged current events and 944 background events are used to study the systematic uncertainties. Table 7.9 summarises the variations in efficiency and rejection resulting from the systematic uncertainties of the inputs.

If the "high" systematic uncertainties are assumed, one additional event gets lost (0.26%

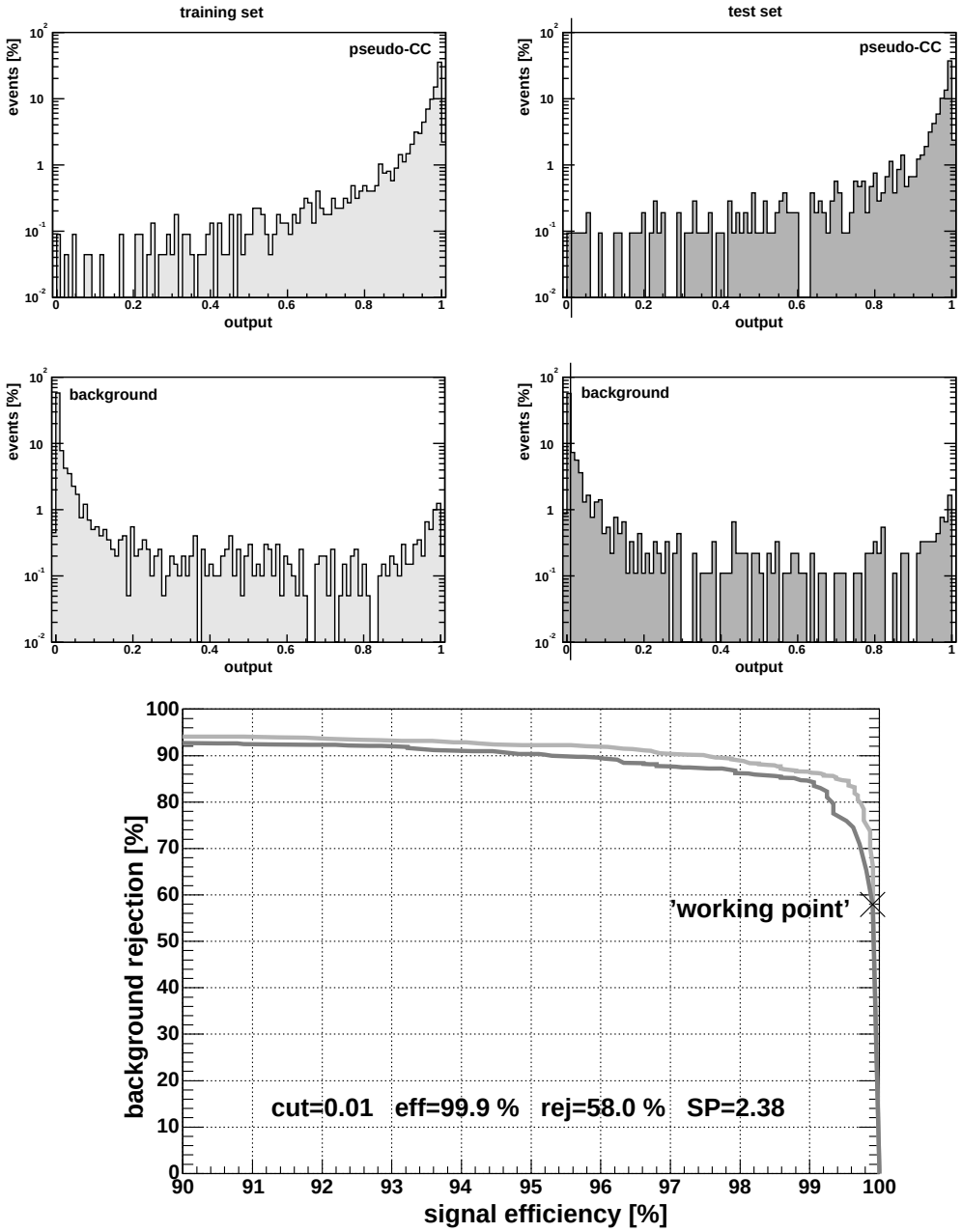

Figure 7.13: Output distributions and efficiency vs. rejection graph for the pseudo-CC network: The light grey histograms show the output on the training set (top left) and result in the light grey curve in the efficiency vs. rejection plot. The dark grey curve shows a slightly worse performance as it comes from the output distributions on the test set (top right). The term signal means here the pseudo-CC selection, the background comes from level 1 sub-trigger 77. The signal efficiency of 99.9% means that only one of the 1063 pseudo-CC events is rejected with the cut at 0.01 which is shown in the output histograms of the test set.

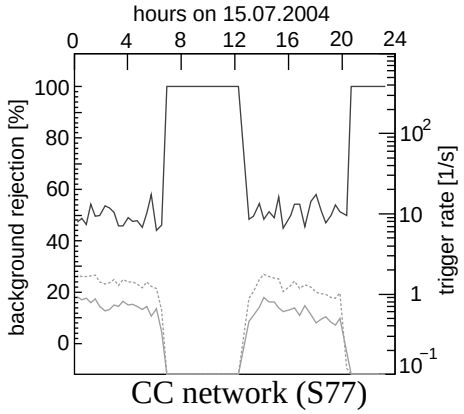

Figure 7.14: Online monitoring of the CC network: The background rejection (dark grey) is around 50% as calculated from the reduction of the L2 input rate (light grey, dotted) to the L2 output rate (light grey, solid). Like in figure 7.7 two periods without beams can be seen.

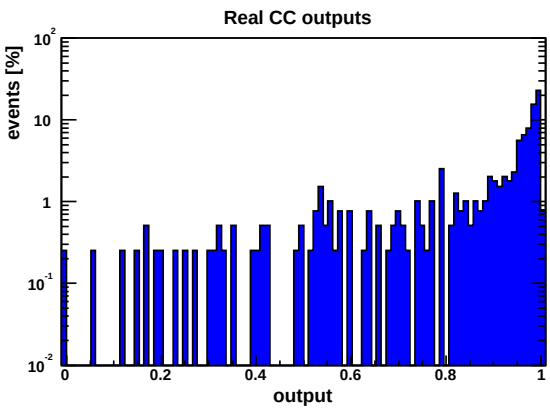

Figure 7.15: Efficiency check for real charged current events, one event is lost, all other outputs are above the cut at 0.01.

Variable	uncertainty level		
	low	medium	high
LAr calibration	2%	4%	8%
z-vtx efficiency	2%	4%	8%
z-vtx hist. shift [bins]	0.125	0.25	0.5
DC$r\phi$ track efficiency	2%	4%	8%

Table 7.8: Sources for systematic uncertainties for the CC dataset.

Efficiency [%]	99.74	±0.26 (stat.)	− 0.00 + 0.00 (syst.,low) − 0.00 + 0.00 (syst.,medium) − 0.26 + 0.00 (syst.,high)
Rejection [%]	60.88	±1.60 (stat.)	− 0.90 + 0.39 (syst.,low) − 1.74 + 0.71 (syst.,medium) − 4.78 + 1.35 (syst.,high)

Table 7.9: Total uncertainties for the three different levels of underlying systematic uncertainties calculated for the real charged current selection.

corresponds to 1 of 391). This event will be shown and discussed in the next section as well as the one which anyway would not have passed the trigger. With only a slight variation in the efficiency for high uncertainties the neural network behaves very stable over longer running periods. Even large systematical uncertainties of the inputs lead to only small systematical uncertainties of the output, again a sign of the "fault tolerance" of neural networks. Although the uncertainty in the rejection rate is higher than for the DVCS network even the variations for a high-level of underlying uncertainties are still acceptable.

Artificial Intelligence

The event which was part of the charged current selection but would have not passed the trigger is shown in figure 7.16. The energy depositions in the LAr calorimeter in both views suggest that they have not been generated by particles coming from the interaction region but by a high energy cosmic ray passing the detector from top to bottom (high energy cosmic rays, i.e. muons, can initiate electromagnetic showers in the steel plates of the hadronic LAr calorimeter if their energy exceeds the "critical" energy of about $300GeV$). The remaining parts of the event structure then show some kind of ep-interaction but not a charged current event since this presumption was based on the unbalanced LAr energy. Thus this event should not have been part of the charged current selection.

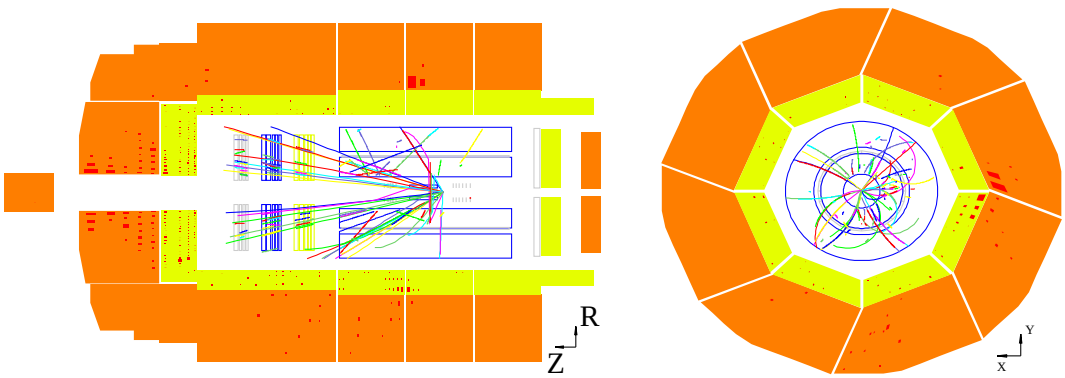

Figure 7.16: Event from the charged current selection which would have been rejected by the neural network. It turned out to be most probably an overlay event with a high energy cosmic muon passing through the detector.

The other interesting event of the charged current selection is the one which is rejected in addition if a large variation downwards in the DC$r\phi$ track efficiency is assumed. This event originally had an output of 0.25 and falls slightly below the cut at 0.01 if the DC$r\phi$ track efficiency is lowered by 8%. The reason can be deduced directly from the modified inputs of this event: A statistical process simulates the conversion of floating point numbers (which result from the change in efficiency) into integers. This event had the bad luck that the input `trhineg` (number of high energy negative tracks) turned from 2 to 1 and `trloneg` (low energy negative tracks) from 1 to 0. All other inputs of this event were zero and of course stayed so. This drastic change in the number of tracks leads to the drop in the output.

Figure 7.17 shows this special event with apparently two opposite charged particles coming from the vertex. One of them points to the energy deposition in the LAr calorimeter. A closer look, however, reveals that both particles have the same momentum. Therefore

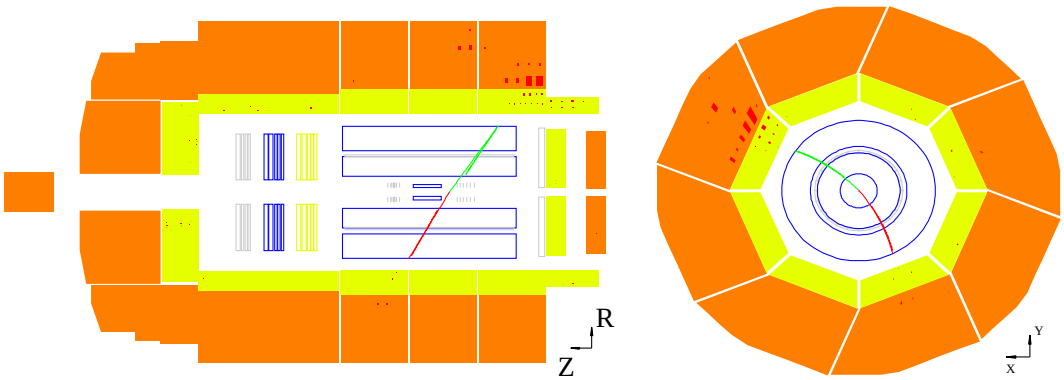

Figure 7.17: One event from the charged current selection which might be rejected due to high systematic variations. It turned out to show a cosmic muon which generated the energy deposit in the LAr calorimeter and then passed through the interaction region.

the two tracks describe most likely one particle coming from the outside and not from the interaction region. Since the track is linked to the energy deposition in the LAr calorimeter it describes most likely the cosmic muon which has generated the energy in the calorimeter. Also, the energy deposition in the calorimeter is certainly due to a cosmic muon which means that again this event should not have been part of the charged current selection.

In summary, we have optimised a new neural network for the triggering of charged current events in the HERA II period. This network has proven to be very efficient: no single event has been rejected which showed undoubtfully a charged current interaction. In addition, the rejection of over 50% is sufficient for the needs of the trigger system. A detailed analysis of the network behaviour revealed two events which should not have been part of the charged current selection. This means that the neural network was able to detect a weak point of a physics selection by intelligent pattern recognition even based on only the coarse information delivered to the second trigger level.

7.2.5 Exclusive J/ψ Photoproduction – $J/\psi \to e^+e^-$

Trigger Development

Level 1 sub-trigger 33 is dedicated to heavy vector mesons (J/ψ's and Υ's) (see section 2.1.5), decaying into electrons for high γp-centre-of-mass energies. In this case the decay electrons are boosted into the backward region, thus one is found in the SpaCal and the other in the liquid argon calorimeter. Neural networks for the rate reduction of this trigger have been used since long time reducing the rate from about 10 Hz to less than 1 Hz. An update of the existing network was necessary in 2004 because some input quantities changed (`spcent1`, compare section 2.1.4). Training data was available from two independent J/ψ selections from the year 2000 (A) and 2004 (B). The background was taken from a transparent run in May 2004. The inputs used in the final training are the four SpaCal quadrant energies, the four DC$r\phi$ track quantities, three of the four z-vertex track quantities (without `nbigfwd`) and the three z-vertex histogram quantities, totally 14 inputs.

The training of the neural networks was done using the selection B from 2004 and cross-checking the efficiency on selection A from 2000. Several networks were trained with different parameter settings (compare appendix B.3.1), the best performance on the

selection set was found for a network with 20 hidden neurons. Interestingly, the trained network with a background rejection of 94% showed an efficiency of 95% on selection B but over 99% on selection A. Comparing the output distributions of the two selections (figure 7.18) reveals that there might be a special class of events in selection B which look very much like background (output close to 0). This class is not present in selection A.

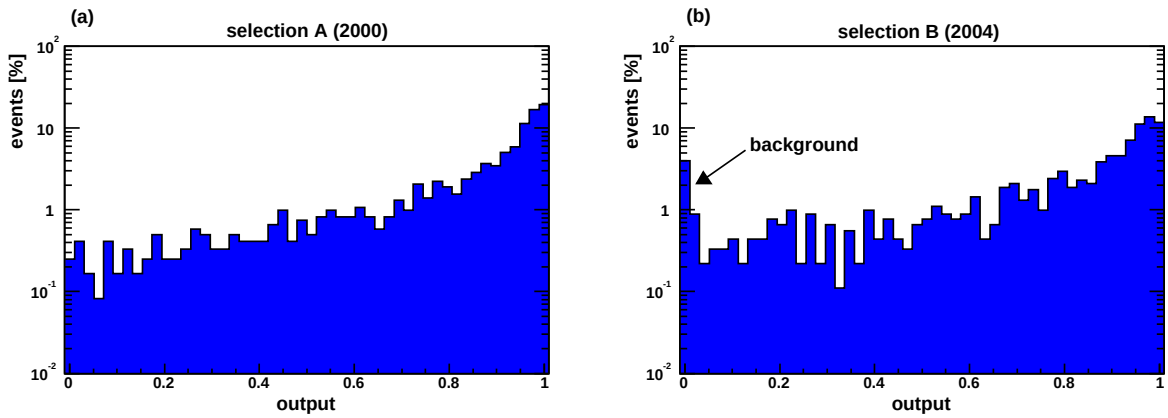

Figure 7.18: Output distribution for the two $J/\psi \to e^+e^-$ selections: (a) selection A (2000) and (b) selection B (2004). A special event class in selection B with outputs around 0 was identified as background by the neural network.

Indeed, simply by scanning through the selected events from 2004 which get a very low output a weak point of the event selection was revealed: One of the selection cuts was chosen very soft by purpose but turned out to be much too loose since a lot of background events were kept then. Exactly these background events were recognised and rejected by the neural network – a very nice demonstration of the pattern recognition capabilities of neural networks and statistical learning methods in general.

Influence of the Training Set Size

The number of available events for training and testing is one of the crucial factors for successful statistical learning. A small number of training events may lead to overtraining or, if overtraining is reduced by regularisation, the regularisation will allow only a very small function space, leading probably to a trained method not performing too well. A small number of test events will result in a very rough guess of the performance because the statistical uncertainties will be large.

The $J/\psi \to e^+e^-$ dataset will be used in this section to study the dependence of the classifier performance on the training set size for different learning methods. Table 7.10 shows the different sizes for training, selection and test set which are studied. It is important to decrease the size of the selection set in the same way as the training set because a significant bias can be created if the best classifier is chosen among different parameter sets on the basis of a large selection set. The remaining test set is used to measure the true performance of the selected classifier, its size influences the statistical uncertainty of the performance measurement.

Figure 7.19 shows the performance on the test set for four selected statistical learning methods (neural network, support vector machine, random forest and decision tree C4.5). The markers show the minimum percentage of misclassifications M which is calculated as

Training [%]	1	2	4	8	10	15	20	25	30	35	40
Selection [%]	1	2	4	7	9	13	16	25	28	33	30
Test [%]	98	96	92	85	81	72	64	50	42	32	30

Table 7.10: Different sizes of training, selection and test sets: The percentages are applied independently to 6892 background and 906 $J/\psi \rightarrow e^+e^-$ events.

$M = (1-\epsilon) + (1-r)$ where ϵ is the efficiency and r is the rejection. Sometimes simply the number of misclassified events divided by the total number of events is calculated. But this definition neglects the possibility of very different set sizes for signal and background. The uncertainty shown is the combined uncertainty of efficiency and rejection added in quadrature (see section 3.13).

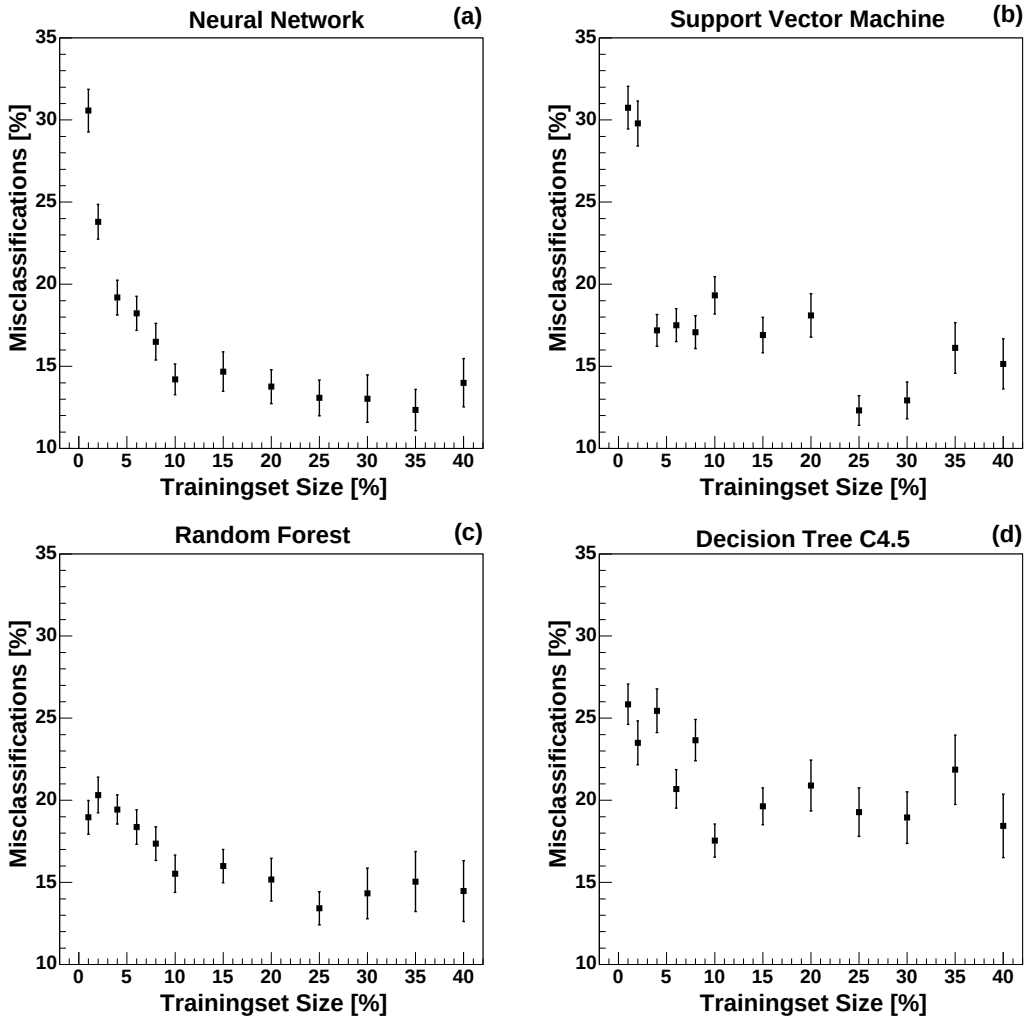

Figure 7.19: Dependence of the misclassification rate on the training set size for four different learning methods, measured with the $J/\psi \rightarrow e^+e^-$ dataset.

For each learning method and for each size of the training set several classifiers have been trained with different parameter settings. This is very important in this context since it allows to choose optimal regularisation parameters. A good regularisation is especially needed for the small training sets. Details about which parameters are varied for which

learning method can be found in appendix B.3 for the learning methods neural network, support vector machine and random forest. The selection set is used to choose the best classifier among the different trainings for each training set size and for each learning method. The data points shown in figure 7.19 have been measured with the best classifier of each group.

As expected, the misclassification rate is higher for the very small training sets. It seems, however, that the performance does not increase much if the training set becomes larger than 10% of the total data. Neural networks (figure 7.19 a) show a smooth behaviour for the misclassification rate as a function of the training set size. As expected the misclassification rate rises strongly towards very small training set sizes and reaches a flat minimum towards large sizes. The performance of the support vector machines (figure 7.19 b) is quite unstable, the random forest method (figure 7.19 c) shows a nice stable behaviour even for very small training set sizes and the decision tree C4.5 (figure 7.19 d) is again a little unstable with a generally higher misclassification rate.

This study confirms the necessity to have as many training events available as possible. However, even more importantly, one observes unstable behaviour for the support vector machines and the decision tree C4.5. In prospect to the analysis of the parameter optimisation process, which will be demonstrated using the dijets dataset (section 7.2.7), we see a clear indication here that other effects besides the size of training set scale the achieved performance. This makes it necessary to train several times in different ways and then choose the most successful training by selecting the optimal performance with the selection set.

7.2.6 Inelastic J/ψ Photoproduction – $J/\psi \to \mu^+\mu^-$

Trigger Development

The training data used here consist of a signal selection of J/ψ's decaying to $\mu^+\mu^-$ (see section 2.1.5) and background from a transparent run, both from 2004. Since a medium rate reduction of 30% was considered sufficient for the level 1 sub-trigger 15 (with rates around 6 Hz) a high efficiency can be obtained and the neural network was selected accordingly. The inputs used for this network consist of the LAr energies (`if` region in four quadrants, `fb` and `cb` regions summed), the three z-vertex histogram quantities and the muon quantities `ironfe`, `ironfb`, `ironcb` and `ironbb` (compare section 2.1.4), totally 13 inputs. The neural network selected among the different parameter settings has 30 hidden neurons. The trained neural network shows an efficiency of 96% at a rejection of 30% on the test set. These designed values have been verified online: The rejection rate measured with the monitoring system was 30% and the efficiency check with orthogonal triggers resulted in 56 kept of 56 total events in good agreement with the designed value of 96% within the statistical uncertainty.

Level 1 sub-trigger 15 will be taken exemplary to demonstrate that of course the development of neural network triggers has a direct impact on the physics analysis and its results. Figure 7.20 presents a preliminary result of the selection of J/ψ events [103] which have been triggered after the neural network trigger had been activated. The mass peak at $3.1GeV$ is clearly visible and even the $\psi(2S)$ at $3.7GeV$ can be recognised. This selection of $J/\psi \to \mu^+\mu^-$ events covers all level 1 sub-triggers. An easy calculation shows that the neural network increases the J/ψ's triggered by sub-trigger 15 by 37% with respect to the

alternative pre-scaling. This number is derived from the comparison of the efficiency of the neural network (96%) to the efficiency of pre-scaling (70%) at the same rate reduction (30%).

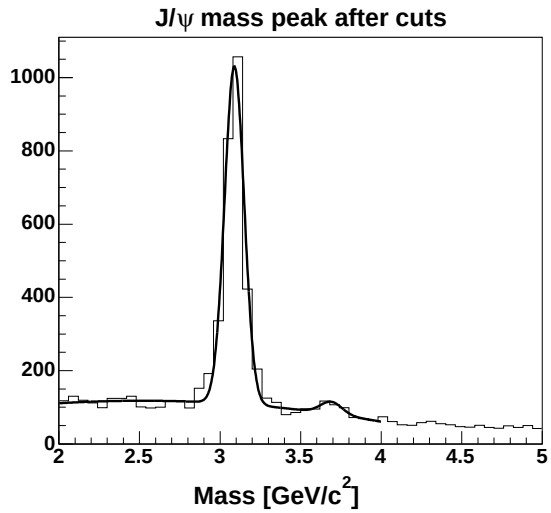

Figure 7.20: A preliminary J/ψ mass peak [103] as seen after selection for the events which have been triggered after the neural network trigger had been activated.

Input Selection

The selection of appropriate inputs is a central task for statistical learning methods. Quite frequently the original structure of the available data is very high-dimensional (in the order of several hundred input quantities) and is thus unsuitable to be used directly. Section 3.6 discussed some ways of preprocessing such datasets. Here we will show how a subset of inputs can be selected from an initially large number of available inputs so that the performance of the learning method is optimised. As mentioned in section 3.2, presenting all available inputs to the learning method is mostly not optimal because some inputs tend to confuse more than help – or if this statement seems inappropriate in the context of statistical learning: the more unnecessary inputs are used the more likely overtraining will happen and the more regularisation must be done which finally decreases the performance.

We start with a training with all inputs which potentially can be used to distinguish between a J/ψ decaying to $\mu^+\mu^-$ and its competing background from level 1 sub-trigger 15. Table 7.11 includes all detector components delivering signals to the neural network trigger. In principle all available inputs are listed, only a few restrictions have been made a priori: the energies in the forward and central region of the LAr calorimeter for example are not presented in four quadrants but only as a sum due to the low energy values there.

Table 7.11 presents the correlation values ρ (equation 3.1 on page 43) for all inputs and the mean absolute correlation $\bar{\rho}$ per detector component. These values can be obtained without any training whereas the relevance R (equation 3.20 on page 65) is calculated after a learning method has been trained with all available inputs (the random forest method is used here and in the following as an example). Finally, the table shows the performance results for the specific sub-detectors: The misclassifications M are calculated for trainings in which only inputs from the specified detector component are used.

All the presented values give a hint about the importance of each input quantity and the respective detector component. At a first glance the correlation coefficients ρ, the relevances R and the misclassifications M agree that the SpaCal quantities and the muon

quantities are probably candidates which could be removed. The assessment of the other inputs is more difficult, at least the drift chamber quantities and the z-vertex-histogram quantities look promising. It could also be discussed whether the sum of relevances or correlations is a better measure of the importance of a whole group of inputs than the mean value.

Group	Liquid Argon Energies $\bar{\rho} = 0.10$ $\bar{R} = 4.9$ $M = 60.3\%$					
Inputs	`eifq0`	`eifq1`	`eifq2`	`eifq3`	`larfbe`	`larcbe`
Correlations ρ	0.05	0.02	0.04	0.07	0.22	0.18
Relevances R	8.6	4.5	5.1	5.5	2.8	2.9

Group	SpaCal Energies $\bar{\rho} = 0.03$ $\bar{R} = 0.9$ $M = 83.4\%$			
Inputs	`espq0`	`espq1`	`espq2`	`espq3`
Correlations ρ	0.02	0.00	0.05	0.04
Relevances R	1.2	0.8	0.8	0.7

Group	Drift Chamber Tracks $\bar{\rho} = 0.26$ $R = 5.2$ $M = 57.0\%$			
Inputs	`trhineg`	`trhipos`	`trlopos`	`trloneg`
Correlations ρ	0.37	0.24	0.15	0.28
Relevances R	8.9	4.6	2.5	4.8

Group	Muon Candidates $\bar{\rho} = 0.07$ $R = 2.1$ $M = 82.6\%$			
Inputs	`ironbe`	`ironfe`	`ironfb`	`ironbb`
Correlations ρ	0.05	0.06	-0.13	-0.03
Relevances R	2.0	1.6	2.0	2.7

Group	z-vtx Tracks $\bar{\rho} = 0.21$ $R = 4.9$ $M = 71.6\%$			
Inputs	`nbigfwd`	`nbigfce`	`nbigbce`	`nbigbwd`
Correlations ρ	-0.05	0.22	0.27	0.30
Relevances R	0.5	6.7	6.5	6.0

Group	z-vtx Histogram $\bar{\rho} = 0.21$ $\bar{R} = 6.1$ $M = 60.7\%$		
Inputs	`cpvsum`	`cpvmax`	`cpvpos`
Correlations ρ	0.26	0.24	0.14
Relevances R	5.0	3.6	9.7

Table 7.11: Assessment of single inputs and detector components for the $J/\psi \rightarrow \mu^+\mu^-$ dataset: The correlations with the target value and the relevances in a training which includes all inputs are given for all available inputs. These values are also summarised for each detector component in a mean value of the absolute correlations and a mean relevance. In addition the percentage of misclassifications M for a training with the respective detector component is given.

The search for the subset which performs best is done here by successively dropping the input quantity with the least relevance. This strategy hopes to improve the performance step by step by removing unnecessary inputs until the performance decreases again when inputs get removed which contained useful information. Table 7.12 shows which input

got removed in which step and what its relevance was in the last training. The resulting misclassifications show indeed the predicted behaviour: The minimum percentage of misclassifications is reached after removing 5 inputs (the number of tracks in forward direction and all SpaCal quantities).

Input removed	none	`nbigfwd`	`espq1`	`espq3`	`espq2`	`espq0`	`ironfe`
Relevance R	–	0.5	0.7	0.8	0.8	1.2	1.4
Misclassifications M	43.9%	43.7%	42.6%	42.0%	42.5%	**41.7%**	45.1%
	`ironbe`	`ironfb`	`trlopos`	`ironbb`	`larcbe`		
	1.1	1.9	2.4	2.5	2.9		
	44.1%	42.1%	44.8%	46.5%	45.5%		

Table 7.12: Successive removal of inputs for the $J/\psi \to \mu^+\mu^-$ dataset: For each step the removed input, its relevance in the last training and the percentage of misclassifications after removing it are given. The minimum is reached after removing five inputs (the number of tracks in forward direction and all SpaCal quantities).

7.2.7 D^* and Dijet production

Rate Reduction for a trigger with two different physics classes

The two physics channels, D^* and dijet production (see section 2.1.5), are discussed together since they are both triggered by the same level 1 sub-trigger 83. This sub-trigger needs a rate reduction of about a factor 2 (the rate was around 4 Hz). At the same time, the trigger should have a sufficiently high efficiency for both physics channels. This is ensured by training two neural networks, one for each physics channel. The trigger decision is then the logical OR of both networks, leading to a total background rejection which might be lower than the two single rejection rates. Accordingly, the two total efficiencies might by higher than the designed single efficiencies. To control and estimate these "crosstalk" effects is an important part of the trigger design.

The chosen inputs are the same for both networks. They consist of the four LAr inner forward quadrant energies and the energy sums for the forward and central barrel. Further inputs are the four DC$r\phi$ track numbers, the three quantities describing the z-vertex histogram and finally three of the four z-vertex track quantities (without `nbigfwd`), totally 16 inputs. Like in the previous sections multiple trainings have been done with different parameter settings (compare appendix B.3.1). The networks with the best performances on the selection sets have 8 hidden neurons for both datasets. Table 7.13 summarises the performance values measured for the two networks independently and measured for the logical OR of the two networks.

	D^* efficiency	Dijet efficiency	background rejection
D^* network	95%		43%
Dijet network		95%	50%
Combined (OR)	96%	97%	37%

Table 7.13: Efficiencies and rejections for D^* and dijet production and the combination of both classifiers.

The first two lines are determined by using the test set of each training independently in the usual way. The last line is derived from the combination of both networks:

- The test part of the D^* signal is passed through the dijet network. Any event which is kept there but not in the D^* network has to be added to the D^* efficiency.
- Analogously any dijet event kept by the D^* network has to be added to the dijet efficiency.
- The total rejection is given by those events of the test part of the background which pass neither the D^* nor the dijet network.

Comparison of Learning Methods

The D^* dataset is used here to perform a comparison of the four learning methods k-nearest-neighbours, naive Bayes, neural network and random forest. This comparison is done as described in section 3.15.2. A first step towards the comparison of learning methods was done in section 7.2.3 where specific hypotheses have been compared. The next step is to include the study of a varying training set which leads to the cross-validation structure (see section 3.12). We choose five-fold cross-validation dividing the 3500 D^* and 1500 background events into five equal sets which are used to train five classifiers. These five classifiers are then evaluated to derive the performance variations for each learning method as described in section 3.15.2.

The learning methods were chosen to have at least one algorithm from each of the three basic groups of classifying techniques (see chapter 5): The random forest method is based on a decision tree algorithm[3], k-nearest-neighbours and naive Bayes are both local density estimators and the neural network is based on linear separation.

Figure 7.14 shows the result of the comparison of the four selected methods. A rejection of 60% was fixed for each method and for each of the five cross-validation sets. The variations in the efficiency were used to derive a 95% confidence interval for each performance difference as described in section 3.15.2. The mean efficiencies for the rejection of 60% are shown in the table sorted from best to worst If two neighbouring methods perform significantly different (95% CL) an exclamation mark signifies the gap in performance.

Random Forest	88.8%
$\Delta = 0.7\% \pm 1.8\%$	
Neural Network	88.1%
$\Delta = 1.8\% \pm 1.6\%$ (!)	
Naive Bayes	86.2%
$\Delta = 7.9\% \pm 2.4\%$ (!)	
k-Nearest-Neighbours	78.3%

Table 7.14: Comparison of the efficiencies of different learning methods for 60% rejection on the D^* dataset, statistically significant differences are marked by an exclamation mark (95% CL).

We see that the random forest method and the neural network perform best and almost equally well. The naive Bayes method is significantly worse and the k-nearest-neighbours search again performs significantly worse. However, we have to note that this comparison and the resulting sequence of performance cannot be generalised but applies only for this

[3]As mentioned in section 5.5.2 the good performance of the random forest method is not due to the underlying decision tree but due to the bagging structure.

specific D^* dataset. Datasets from different applications may lead to different conclusions. A summary for the experiments used in this thesis will be given in section 8.5.

Parameter Optimisation

The selection of dijet events will be used to study the parameter optimisation for eight different learning methods: Neural network, support vector machine, random forest, decision tree C4.5, k-nearest-neighbours, naive Bayes, simple cuts and range search. It was emphasised in section 3.11 that different parameter sets should be tried out to regularise the training, i.e. to find an optimal balance between performance on the training set and generalisation which is measured (estimated!) with the test set. Trying out different parameter sets also helps to find a well performing classifier for an unstable learning method (compare the observations from section 7.2.5). The selection set is used to choose the best training from the various parameter settings while the test set is preserved to measure the true performance without any bias.

An especially difficult aspect of the parameter optimisation process comes with learning methods which need some kind of scaling factor per input. The k-nearest-neighbours search, for example, can use a metric with different scaling factors for each input quantity, the range search method as a second example needs to know the extensions of the hyperbox in which the events should be counted. To optimise these parameters for each input means generally to try out all combinations, i.e. already $3^4 = 81$ trials for 3 possible values for 4 inputs. For a large number of inputs this strategy can of course not be used. In this example we restrict the dijets dataset to its four most promising inputs (`larfbe`,`trhineg`,`trhipos` and `nbigbwd`) and allow thereby the extensive search for the best combination of parameters.

To be able to compare the parameter optimisation processes for different learning methods, between 50 and 100 different sets should be tried out for each method. This number includes pre-studies in which a rough guess for good parameters is obtained, for the support vector machine, for example, an acceptable value for the factor γ in the Gaussian kernel may vary over several orders of magnitude. Appendix B.3 describes the parameter variations in more details for neural networks, support vector machines and random forests. The free parameters and their variation for the other methods have been shortly described in the respective sections of chapter 5.

Figure 7.21 gives an overview of the effects of the 50 and 100 parameter variations for the different learning methods. For each method the distributions of training and test error (misclassifications) are plotted for the different parameter settings. The thin black line in the test error distribution marks the true performance estimate which is obtained for the classifier which was chosen by selecting the best performance on the selection set.

We see that overtraining is a very natural effect common to all learning methods, even the combinatorial search for optimal simple (i.e. one-dimensional) cuts shows overtraining (marker A in figure 7.21). The larger the distance between the mean training error and the mean test error the more the learning method tends to overtrain its classifier. This may be a natural and unproblematic behaviour like for the random forest method where some parameter settings show strong overtraining and some less[4], yet they result all in more or

[4]The important parameter for this behaviour is the number of events below which a branch of a tree is no more split. If this number is set equal to 1, the performance on the training set is naturally near "optimal".

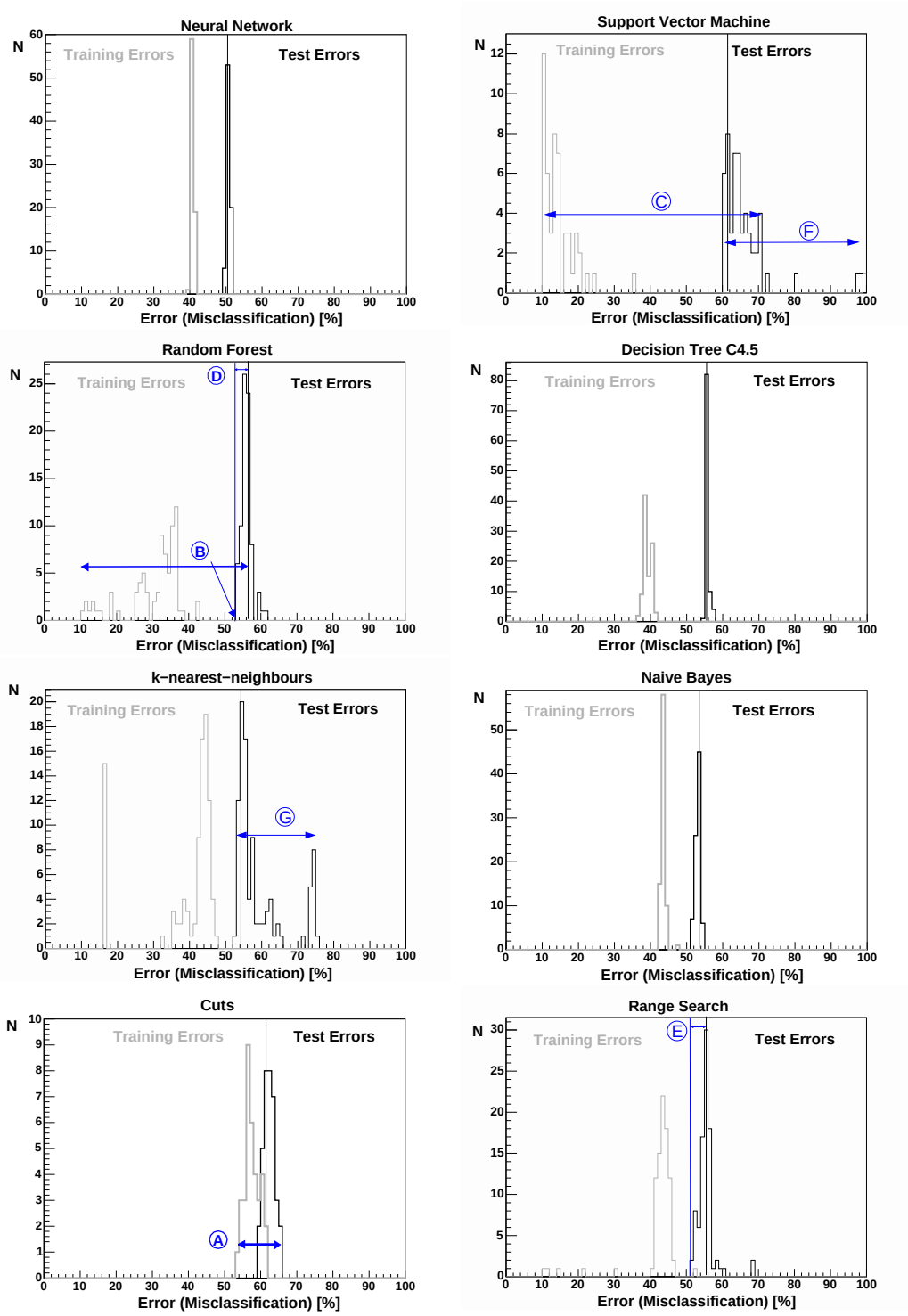

Figure 7.21: Evaluation of the parameter optimisation process with the dijets dataset: The error distributions for the training and test errors are plotted and a thin black line marks the test error which is obtained with the classifier which has the least error on the selection set. The blue letters mark special properties and are referred to in the text.

less the same performance on the test set (marker B in figure 7.21). Strong overtraining is seen for the support vector machines (marker C in figure 7.21).

A second important aspect is the width of the test error distribution. The rule proposed here is to choose the best parameter set with the selection set and not with the test set. Neglecting this rule would result in a significant bias for those methods which show a large variance for the test error. The thin black line shows which true performance (estimated with the test set) can be expected when the best parameter set is chosen with the selection set. A significant bias would have resulted, for example, for random forests (marker D) and range search (marker E). A few percent bias due to the incorrect handling of this problem (i.e. using the test set to choose the best classifier) may produce significant problems in the subsequent analysis.

Finally it has to be emphasised that the goal of studying different parameter sets is to find an optimal set which performs significantly better than the others. For support vector machines (marker F) and the k-nearest-neighbours search (marker G) it is obvious that this parameter optimisation is needed and successful (large spreads of the test errors). For other methods like random forests, combinatorial cuts search, and range search this effect is still visible in the plots and probably significant. For the other methods this whole procedure might seem useless. However, we used here only one dataset as an example, for other datasets (different number of training events, more or less inputs and, of course, different physics) the situation will change definitely (e.g. the support vector machine performs significantly better in other datasets like for DVCS in section 7.2.3).

7.2.8 Summary of Newly Developed Neural Networks

The following table summarises the neural network triggers which have been discussed in the last sections. The first column mentions the underlying physics and the second column gives the level 1 sub-trigger which triggers this physics (mostly one among others). The rate of this sub-trigger will be reduced by the verification with the neural network. Efficiency and rejection of the neural network trigger are stated as the designed values but have been confirmed as discussed in the previous sections.

physics	L1 ST	eff.	rej.	signal selection	comment
DVCS	41	97%	80%	L.Favart [104]	used since 06.2004
CC	77	100%	57%	R.Placakyte [105]	test trigger
$J/\psi \to e^+e^-$	33	95%	94%	R.Lopez (04) [106]	test trigger
				L.Janauschek (00) [102]	
$J/\psi \to \mu^+\mu^-$	15	96%	30%	H.Lüders [107]	used since 06.2004
D^*	83	95%	43%	M.Göttlich [108]	OR dijets: 96% / 37%
					used since 07.2004
Dijets	83	95%	50%	S.Schätzel [109]	OR D^*: 97% / 37%
					used since 07.2004

7.3 Applications for H1: Instanton Purification

The search for instanton-induced events has been introduced in section 2.1.6. If a significant excess of instanton-like events in the H1 data compared to the predictions from the perturbative simulations could be detected, then the instanton hypothesis would be confirmed. The main goal of the analysis presented in the following sections is to enrich the number of instanton-like events in the data as much as possible and to perform then the comparison to the perturbative predictions taking into account statistical and systematic uncertainties.

7.3.1 Datasets

The datasets used in this analysis are provided by an earlier analysis [8]. In the following we will refer to this analysis and its results as "analysis I". The provided datasets consist of Monte-Carlo simulations for instanton-induced events "QCDINS", and two different simulations for perturbative QCD events "MEPS" and "CDM" (compare section 2.1.6). The datasets include also H1 experimental data. All datasets went through a preselection step described in [8] resulting in the numbers shown in table 7.15.

Source	# Events	Mean Weight
DATA	354600	1.056
QCDINS	165301	0.797
MEPS	301722	5.816
CDM	163930	5.029

Table 7.15: Datasets for the instanton analysis, all simulations (QCDINS, MEPS, CDM) are specific to the years 1996 and 1997 from which the experimental data, covering 21.1 pb^{-1}, were taken.

In addition to these basic datasets there exist also datasets for the two perturbative QCD simulations which were modified according to the systematic uncertainties of the underlying observables. This means that the inputs sph_B, n_B, ${Q'_{rec}}^2$, $E_{T,Jet}$ and $E_{T,B}$ (compare section 2.1.6) have been modified according to the systematic uncertainties which are caused by model uncertainties in the simulations, resolution effects in the detector and limitations in the knowledge of the detector calibration. The eleven sources for systematic uncertainties taken into account are listed in table 7.16.

The uncertainties shown in this table are taken from analysis I and unfortunately do not agree totally with the data files actually provided by the author of analysis I [8]. Compared to the analysis I, the data files used here contain a more conservative version of systematics in the sense that larger uncertainties are assumed for the SpaCal electron energy scale, the electron scattering angle and the track efficiency. In analysis I these uncertainties were regarded as too large and have been decreased in a newer version which is unfortunately no longer available.

To obtain systematic uncertainties which can be compared to the results of analysis I we will therefore apply correction factors to the propagated systematic uncertainties obtained for the three quantities mentioned above. These correction factors decrease the obtained conservative uncertainties by up to one order of magnitude to obtain values compatible to analysis I. How these correction factors are derived will be discussed in section 7.3.4.

Liquid argon hadronic energy scale	±4%
SpaCal electron energy scale	±2%
Electron scattering angle	±2mrad
Track momentum scale (FSCOMB)	±3%
SpaCal hadronic energy scale	±7%
Track momentum scale	±3%
Track azimuth angle	±2mrad
Track polar angle	±2mrad
Track efficiency	95/98%
Luminosity uncertainty	1.5%
F_2 uncertainty	3.0%

Table 7.16: Sources for systematic uncertainties for the instanton analysis. These values describe the uncertainties used in analysis I [8]. Here, however, slightly different uncertainties are used (see text).

Figures 7.22 and 7.23 show the kinematic quantities used in the discussed datasets. The plots always show the distributions of the MEPS simulation vs. QCDINS simulation. The one-dimensional plots present all five quantities which are available to distinguish instantons from perturbative QCD background, the two-dimensional plots show any combination of two of the three first-choice inputs sph_B, n_B and ${Q'_{rec}}^2$ (compare the discussion in section 2.1.6). In the two-dimensional projections of the input distribution clear rounded decision boundaries between the light (background) and dark (signal) regions are visible. These slightly rounded decision boundaries encourage to try out neural networks on this problem. There are also regions in which no events were found resulting in an undefined decision state (0.5). Yet it is mostly obvious that they can only belong to the signal region (dark). These undefined regions are dangerous for local density estimators like the range search method but they can be handled well by linear separators, for example by neural networks.

7.3.2 Evaluation Strategy

A high separation power is essential in this analysis, since a small signal is searched in an overwhelming background. The definition of the separation power is

$$SP = \frac{\epsilon}{1-r} \tag{7.2}$$

where ϵ is the QCDINS efficiency and r is the MEPS/CDM rejection. This shows that arbitrarily high separation powers can be reached towards stronger and stronger cuts with the price of a very low efficiency.

For the evaluation of the developed classifiers we follow the convention of analysis I. There the separation power at an efficiency of 10% is measured to compare the performances. Furthermore the uncertainties (statistical and systematic) in the number of selected events from the perturbative simulations have to be taken into account. A high separation power and thus a potentially large fraction of instanton-induced events in the measured data may be compensated by large uncertainties in the number of selected background events.

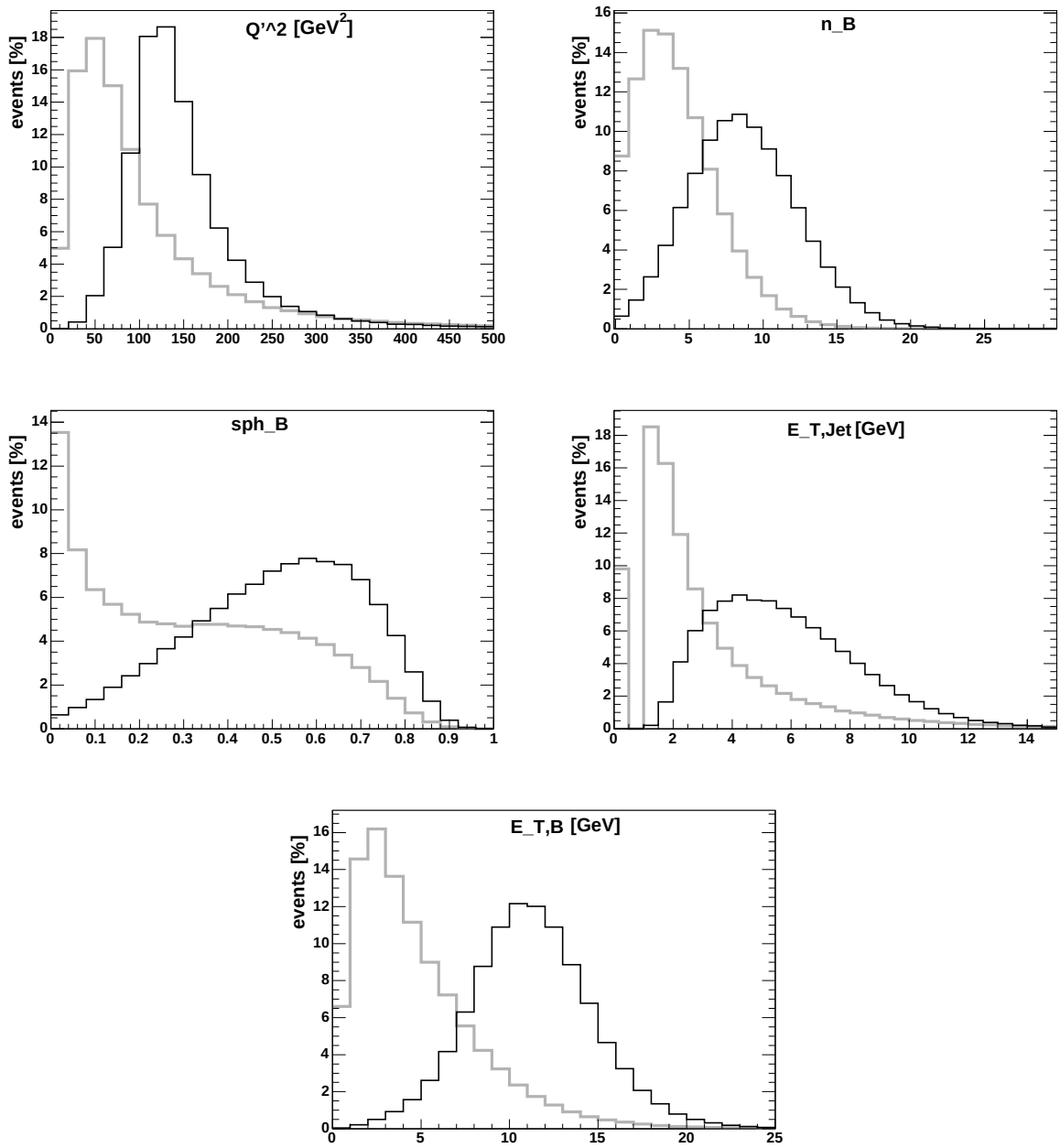

Figure 7.22: Input distributions of the instanton dataset in one-dimensional projections, the signal (QCDINS) in thin black and the background (MEPS) in thick grey.

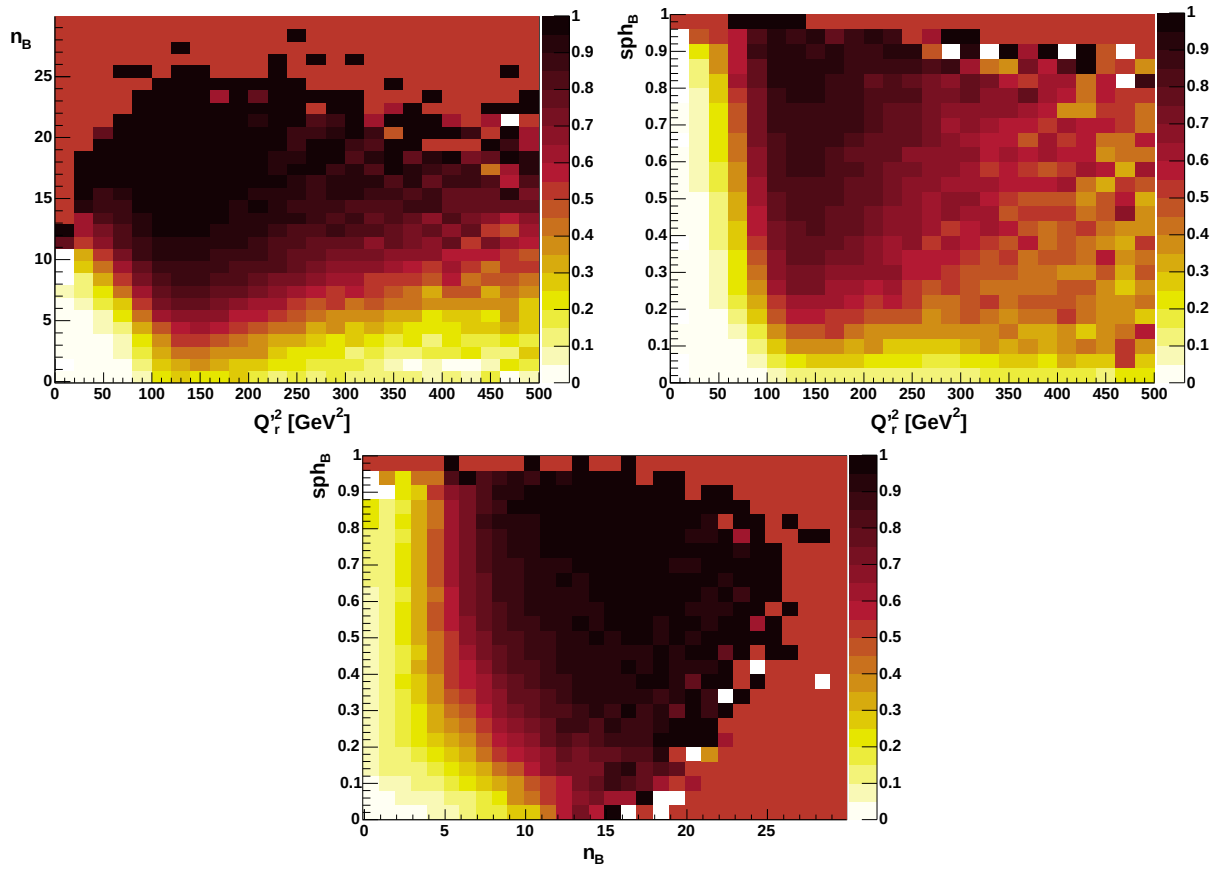

Figure 7.23: Input distributions of the instanton dataset in two-dimensional projections: Plotted is the ratio of QCDINS events vs. all events (QCDINS plus MEPS events) in each bin which can be interpreted directly as the optimal output of a learning method. Signal and background have been normalised to the same number of events. Regions in which no event is found take the value 0.5 (undecided).

The agreement between measured data and simulation in combination with the uncertainty of the simulation will be taken to derive a significance of the instanton hypothesis with the following equation:

$$s = \frac{N_{Data} - (N_{Pert} + N_{Ins})}{\sigma_{N_{Pert}}}. \tag{7.3}$$

The difference between selected events in the measured data N_{Data} on the one hand and selected events from the background simulation (MEPS or CDM) N_{Pert} and from the signal simulation (QCDINS) N_{Ins} on the other hand is divided by the total uncertainty of the background simulation $\sigma_{N_{Pert}}$. The result is a significance s whose absolute value is the larger the lower the probability of the instanton hypothesis is. The sign of the significance tells whether too many (+) or too few events (−) have been found in the measured data compared to the simulated prediction. To be precise, a very large significance means that there are many more events in the data compared to the simulated prediction, which only means that the absolute scaling of the prediction for the number of instanton-induced events may be wrong, still there is "room" for them.

The evaluation is done two times for any classifier: Once with the MEPS simulation and once with the CDM simulation to estimate the fraction of the perturbative QCD events in the experimental data. As we will see, the result of the whole analysis depends strongly on which simulation is chosen.

7.3.3 Training and Search Strategy

There exist many ways to generate a classifier which can be evaluated as described above. The simulated instanton-induced events are, of course, always used as the signal. But the background can be formed by different datasets. Both background simulations MEPS and CDM can be used as well as the experimental data (since the possible contamination with instanton-induced events is most likely negligible).

As mentioned above, a high separation power may mean that a classifier also has large uncertainties which makes the analysis less significant in total. We are interested in both aspects, in the ability to enrich the instanton signal as much as possible and in the significance resulting from the uncertainties.

As discussed in section 3.11, the parameters of the different learning methods have to be varied, resulting in many different classifiers with different overtraining behaviours. We will then choose the classifiers with an optimal separation power at 10% efficiency and evaluate the significances of the instanton hypothesis. For details of the parameter optimisation process, see appendix B.3.1.

7.3.4 Verification of Previous Results

We start our analysis with a verification of the results presented in analysis I [8]. Two different methods were used there: A standard cut-based approach (compare section 3.7) and the range search method (see section 5.3.3). The optimisation of the cuts and the training of the range search method were both done with the MEPS simulation as background and the QCDINS simulation as signal.

As shown in table 7.17 the number of selected events for the cuts developed in analysis I ($95GeV^2 < {Q'_{rec}}^2 < 200GeV^2$, $n_B > 11$ and $sph_B > 0.4$) can be confirmed. However, there

		Analysis I		This Analysis		corr.
		CDM	MEPS	CDM	MEPS	fac.
No Variation		443	304	443	304	
LAr had. E	up	−1.22%	+0.23%	−1.22%	+0.23%	
	down	+2.55%	+2.81%	+2.55%	+2.82%	
SpaCal el. E	up	+0.07%	+1.38%	+0.43%	+0.53%	0.10
	down	−0.33%	−0.35%	−0.31%	−0.32%	0.07
Elec. scatt.	up	−1.32%	−1.27%	+1.12%	+2.02%	0.90
	down	+2.64%	+2.13%	−1.28%	−1.31%	0.43
Track (FSCOMB)	up	+0.93%	−0.56%	+0.92%	−0.56%	
	down	−0.75%	+3.02%	−0.75%	+3.03%	
SpaCal had. E	up	+0.00%	−0.39%	+0.00%	−0.37%	
	down	+0.00%	+0.00%	−0.01%	−0.01%	
Track mom.	up	+0.19%	+0.22%	+0.19%	+0.22%	
	down	−3.14%	−3.74%	−3.14%	−3.74%	
Track azimuth	up	−0.21%	−0.03%	−0.21%	−0.03%	
	down	+0.37%	+0.18%	+0.37%	+0.18%	
Track polar	up	+0.84%	+0.94%	+0.84%	+0.94%	
	down	−1.06%	−1.65%	−1.06%	−1.65%	
Track eff.		−3.62%	−4.28%	−2.62%	−4.15%	1.00
Lumi uncert.		±1.5%				
F_2 uncert.		±3.0%				

Analysis I					
CDM			MEPS		
Syst.	Stat.	Total (1σ)	Syst.	Stat.	Total (1σ)
−27.6 +19.5	±21.5	−35 +29	−21.4 +16.5	±13.0	−25 +21

This Analysis					
CDM			MEPS		
Syst.	Stat.	Total (1σ)	Syst.	Stat.	Total (1σ)
−26.7 +21.1	±21.5	−34 +30	−21.7 +18.5	±13.0	−25 +23

	Analysis I			This Analysis		
	selected	Sep.Pow.	significance	selected	Sep.Pow.	significance
Data	484			484		
QCDINS	81			82		
CDM	443 $^{+29}_{-35}$	86	−1.1	444 $^{+30}_{-34}$	86	−1.2
MEPS	304 $^{+21}_{-25}$	125	+4.7	304 $^{+23}_{-25}$	125	+4.3

Table 7.17: Verification of the **cut-based approach** in 3 dimensions (optimised for MEPS): The first table shows the relative change in the number of selected events according to the different systematic uncertainties ($\pm 1\sigma$). The second table summarises these relative changes to total changes in the absolute number of events. The third table compares the number of selected events for data vs. simulations and derives the significance s for the instanton hypothesis (equation 7.3).

are obvious differences in the uncertainties, as expected from the discussion above. For those uncertainties that need a correction factor to make the results compatible (compare the discussion in section 7.3.1), the shown propagated uncertainties for this analysis have already been corrected by the shown factor. The correction factors have been calculated to transform the mean (MEPS and CDM) propagated uncertainty obtained in this analysis into the mean propagated uncertainty from analysis I. The correction factors chosen are not able to reproduce the previous systematics exactly, but the total uncertainties agree well.

For the electron scattering angle there is, in addition, a sign problem. If the up/down variation is switched the numbers agree fairly well. The systematic uncertainties are combined with the statistical uncertainties in the second table and the total uncertainties are used in the third table to derive the significance s for the instanton hypothesis (equation 7.3).

As discussed above a large absolute significance means that the predicted and observed numbers of events disagree. In particular, the significances from the cut-based approach show that there are too few events in the data in comparison with the CDM simulation (significance -1.1 / -1.2) and much too many in comparison with the MEPS simulation (significance $+4.7$ / $+4.3$). In terms of selected events, without taking into account the predicted number of instanton-induced events, this means that both simulations predict less events than found in the data which can be interpreted as an evidence for instanton-induced events. The excess in the data, however, differs significantly for the two simulations: Whereas the excess is smaller than predicted by QCDINS for the CDM simulation, it is much larger for the MEPS simulation. Still, there seems to be "room" for instanton-induced events in both simulations.

Table 7.18 shows the results for the range search method[5]. The same parameters for the range search algorithm were used in analysis I and in this analysis: The box size is fixed symmetrically with a length of 65 GeV^2 in ${Q'_{rec}}^2$, 0.0875 in sph_B and asymmetrically 1 downwards and 5 upwards in n_B. Although the range search algorithm implemented in this thesis is based on the same concept as in analysis I, there are obviously some details in each implementation which generate differences[6]. In table 7.18 many examples are found where the variations up and down of one quantity both lead to a change in the number of selected events in the same direction, compare the footnote on page 64.

The differences in the number of selected events and in the uncertainties make a direct comparison of the two calculations useless. They lead to slightly different conclusions: While both significances in the old analysis (-1.0 for CDM and $+1.2$ for MEPS) are compatible with the instanton hypothesis the significances in this analysis (-0.4 for CDM and $+2.1$ for MEPS) show an even better match for the CDM simulation but the disagreement for the MEPS dataset is larger.

[5]The fact that in analysis I the propagated uncertainties for the electron scattering angle are exactly the same as for the cut-based method makes these numbers questionable.

[6]The implementation of the range search method in [8] uses for example the rule that an event should be classified as background if there are less events in the range search box than a given threshold. This rule can only worsen the classification and is an example for the fear to lose control over "statistical uncertainties".

		Analysis I		This Analysis		corr.
		CDM	MEPS	CDM	MEPS	fac.
No Variation		354	299	429	348	
LAr had. E	up	+2.73%	+3.80%	−3.60%	+3.65%	
	down	+4.60%	−1.42%	−0.46%	+1.28%	
SpaCal el. E	up	+4.98%	−6.29%	+0.87%	+0.15%	0.10
	down	+4.70%	−1.07%	−0.36%	−0.18%	0.07
Elec. scatt.	up	−1.32%	−1.27%	+0.54%	−1.27%	0.90
	down	+2.64%	+2.13%	−1.76%	−0.60%	0.43
Track (FSCOMB)	up	+2.78%	−0.55%	+0.07%	+3.89%	
	down	+2.00%	+1.13%	−2.91%	+0.19%	
SpaCal had. E	up	+0.09%	+0.00%	+0.00%	−0.01%	
	down	+0.09%	+0.00%	+0.01%	−0.02%	
Track mom.	up	+0.09%	+0.20%	+0.00%	−0.34%	
	down	−3.04%	−5.88%	−2.57%	−4.94%	
Track azimuth	up	−0.32%	−0.22%	−0.75%	−0.12%	
	down	+0.08%	+0.20%	+0.12%	+0.36%	
Track polar	up	+0.80%	+0.61%	+0.51%	+0.64%	
	down	−0.61%	−2.99%	−1.50%	−3.22%	
Track eff.		−3.53%	−4.05%	−2.80%	−2.97%	1.00
Lumi uncert.		±1.5%				
F_2 uncert.		±3.0%				

Analysis I					
CDM			MEPS		
Syst.	Stat.	Total (1σ)	Syst.	Stat.	Total (1σ)
−18.4 +35.6	±17.9	−26 +40	−31.5 +15.7	±18.9	−37 +25

This Analysis					
CDM			MEPS		
Syst.	Stat.	Total (1σ)	Syst.	Stat.	Total (1σ)
−33.0 +18.9	±19.2	−38 +27	−26.5 +22.6	±21.4	−34 +31

	Analysis I			This Analysis		
	selected	Sep.Pow.	significance	selected	Sep.Pow.	significance
Data	410			496		
QCDINS	81			83		
CDM	354 $^{+40}_{-26}$	106	−1.0	429 $^{+27}_{-38}$	89	−0.4
MEPS	299 $^{+25}_{-37}$	126	+1.2	348 $^{+31}_{-34}$	110	+2.1

Table 7.18: Verification of the **range search approach** in 3 dimensions (trained on MEPS): The first table shows the relative change in the number of selected events according to the different systematic uncertainties ($\pm 1\sigma$). The second table summarises these relative changes to total changes in the absolute number of events. The third table compares the number of selected events for data vs. simulations and derives the significance s for the instanton hypothesis (equation 7.3).

7.3.5 New Results

The focus of investigation for a potential improvement of the previous analysis will be placed on the combinatorial cut search, the range search method and neural networks. While the first two methods have been chosen in the previous analysis, neural networks have been regarded as too time consuming and not worth considering regarding possible improvements in performance. This assessment will be challenged here. In addition the previous analysis fixed the background set to the MEPS simulation while here all the different training strategies which have been discussed in section 7.3.3 will be tried out.

We first address the question why the search for instanton-induced events was restricted to three inputs in the previous analysis. One part of the answer is that the differences between simulation and data in the two additional quantities $E_{T,Jet}$ and $E_{T,B}$ are larger than the differences in the other three quantities sph_B, n_B and ${Q'_{rec}}^2$. Closely related but more important, however, is the observation that the simple cut-based approach in five dimensions leads to too many selected events in the CDM simulation compared to the data. Whereas a lower number of perturbative events would be expected in a scenario where instanton-induced events are produced, more perturbative events in the prediction than in the data clearly points to a problem in the simulation.

Analysis I has not been able to realise that this problem is not specific to the five input quantities. The effect can also be observed with the three first-choice inputs, however, only when larger efficiencies are required. Since the study in analysis I was restricted to the 10% efficiency the effect plotted in figure 7.24 could not be observed.

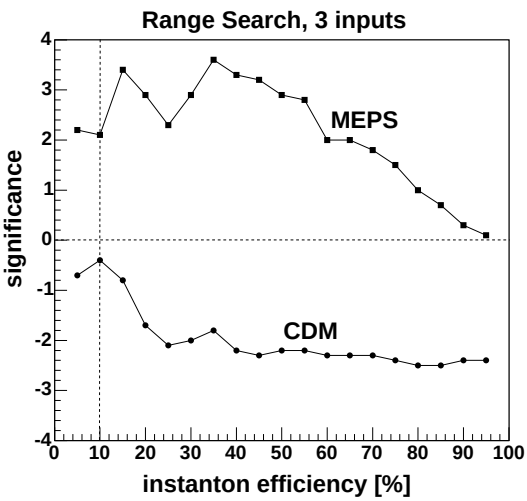

Figure 7.24: Significances in dependence of the instanton efficiency: This curve has been measured with the range search method with three inputs, shown in table 7.18 (trained with MEPS as background). The dotted line marks the special case of 10% efficiency for instanton-induced events.

When the required instanton efficiency is increased by loosening the cut in the output, the separation power decreases and also the differences between simulations and data are expected to disappear: They are explicitly normalised to return the same number of selected events if no cut is done[7]. Since statistical and systematic uncertainties become large compared to the number of selected instanton-induced events, the significance for high efficiencies can be directly taken as a measure for the disagreement between data and simulation. In other words, for high efficiencies it is not important whether the few hundred instanton-induced events are taken into account, the only significant differences there are due to disagreements between experimental data and simulation.

[7]This implicitly assumes that the fraction of possible instanton-induced events is small.

Figure 7.24 demonstrates that the evaluation of the significances at the efficiency of 10% is a very special case. In contrast to intuition, the significances do not decrease but increase with higher efficiencies. This means that much stronger results regarding the disagreement of the two simulations could have been obtained already in analysis I by evaluating higher efficiencies. In particular, it can be seen for high efficiencies that the CDM simulation predicts much more events than seen in the data. This effect will play an important role in the following.

Table 7.19 summarises all the different results from different training strategies. The first column describes which method was used: neural network (NN) or combinatorial cut search (CUTS). It also identifies the training set used: either the three standard inputs (sph_B, n_B and ${Q'_{rec}}^2$) were used ("3") or all five inputs ($E_{T,Jet}$ and $E_{T,B}$ in addition) were used ("5"). The background was formed either by the MEPS simulation ("M"), by the CDM simulation ("C"), or by the experimental dataset ("D"). The second column ("DATA Sel.") shows the number of selected events in the experimental dataset. The following two blocks show the same information for the two simulations CDM and MEPS: The separation power ("SP"), the number of selected events ("Sel.") and its total (statistical plus systematic) uncertainty and the significance (equation 7.3). If the instanton hypothesis holds then the number of selected events in the experimental dataset ("DATA Sel.") should be the sum of the selected instanton-induced events ("INS Sel.") plus the standard perturbative events ("CDM Sel." or "MEPS Sel.").

The following cuts have been selected by the combinatorial cut search with five inputs since they gave the best separation power:

- For training with background from MEPS: $80GeV^2 < {Q'_{rec}}^2 < 200GeV^2$, $n_B > 10$, $sph_B > 0.38$, $E_{T,Jet} > 1.5GeV$ and $E_{T,B} > 12GeV$.
- For training with background from CDM: $80GeV^2 < {Q'_{rec}}^2 < 180GeV^2$, $n_B > 9$, $sph_B > 0.40$, $E_{T,Jet} > 1.0GeV$ and $E_{T,B} > 11.4GeV$.

	Training	DATA	INS	CDM				MEPS			
		Sel.	Sel.	SP	Sel.	Uncert.	Sign.	SP	Sel.	Uncert.	Sign.
1	NN-3-D	388	81	109	345	$^{+25}_{-34}$	-1.1	141	267	$^{+17}_{-23}$	$+2.3$
2	NN-3-C	424	82	99	380	$^{+32}_{-34}$	-1.1	135	280	$^{+17}_{-25}$	$+3.6$
3	NN-3-M	387	81	104	363	$^{+26}_{-34}$	-1.7	142	265	$^{+22}_{-28}$	$+1.9$
4	NN-5-D	261	83	101	377	$^{+69}_{-49}$	-4.1	160	237	$^{+37}_{-41}$	-1.4
5	NN-5-C	347	83	112	343	$^{+46}_{-27}$	-2.9	127	300	$^{+18}_{-38}$	-1.0
6	NN-5-M	260	82	110	346	$^{+43}_{-43}$	-3.9	215	177	$^{+26}_{-20}$	$+0.1$
7	CUTS-5-M	288	82	97	387	$^{+36}_{-36}$	-5.0	193	196	$^{+23}_{-20}$	$+0.5$
8	CUTS-5-C	286	82	103	372	$^{+48}_{-35}$	-4.8	190	199	$^{+15}_{-25}$	$+0.3$

Table 7.19: Summary of new results for the instanton search – the columns are described in the text.

Two technical observations have been made during the training and evaluation of the different classifiers:

- The extensive studies summarised in table 7.19 and in figure 7.25 allow rigorous consistency checks. They have only been possible by the consequent automation of the training and evaluation procedures (compare section 6.3). The calculation of systematic uncertainties plays a very important role in the evaluation process. For each significance 17 test sets have to be evaluated (see table 7.18). Whereas the training times for neural networks and the range search method are similar, the evaluation times are about two orders of magnitude smaller for neural networks. The fast evaluation of the modified test sets with neural networks allows thus to perform the extensive studies shown in table 7.19 and in figure 7.25.

- The comparison of the separation powers from table 7.19 with the range search result in table 7.18 shows that the highest separation powers are achieved with neural networks (142 in line 3 of table 7.19 vs. 126 in table 7.18). The highest separation power is equivalent to the best possible enrichment of instanton-induced events in the data. Whatever the conclusion about the obtained numbers will be, higher separation powers will inevitably lead to more meaningful results.

What is immediately evident from a scan through table 7.19 is that no clear conclusion about the instanton hypothesis can be drawn. Like in the previous analysis [8] the numbers of selected events in simulations and data are sometimes consistent with the instanton hypothesis, sometimes they show deviations. These deviations have unfortunately two different directions for the two simulations CDM and MEPS: The CDM simulation shows almost as many selected events or even more than found in the data leaving little or no room for the predicted number of instanton-induced events. In contrast, the MEPS simulation shows mostly much less selected events than found in the data which leaves room for instanton-induced events. But these would have to be much more than predicted by the simulation, leading to interesting physical implications [7].

A closer look into table 7.19, however, reveals some important details about the mismatch of the two simulations. It was not possible to obtain these details in analysis I because only one specific training was done there (the range search training with three inputs and MEPS as background shown in table 7.18).

The first observation is that the separation powers ("SP") for the CDM simulation are generally much lower than those for the MEPS simulation. This means that it is more difficult to distinguish between CDM events and instanton-induced events (QCDINS) than to distinguish between MEPS events and instanton-induced events.

A second important observation is that the numbers of selected events in the data agree well for the trainings performed with data and MEPS as backgrounds ("DATA Sel." in lines 1 and 3 and in lines 4 and 6). In contrast, the number of selected events in the data is always much higher if the CDM simulation was used as background in the training ("DATA Sel." in lines 2 and 5). This suggests that the phase space regions selected by the training with data and MEPS are similar while the training with the CDM simulation selects a different phase space region. This difference is confirmed, in particular, by the training of the neural network with five inputs: Trainings with data or MEPS as background result in much too many events predicted by CDM ("CDM Sel." compared to "DATA Sel." in lines 4 and 6 vs. line 5). In contrast, training with CDM as background results in a prediction of CDM which matches the seen data.

Figure 7.25 finally shows again that the possible validation of the instanton hypothesis with the MEPS simulation depends strongly on the background which is used in the training as well as on the efficiency for instanton-induced events which is chosen. The regions in phase space selected in the training with the CDM simulation are different from those selected in the training with the MEPS simulation. Above, this was observed with the number of selected events in the CDM simulation. Here, this can be observed by the behaviour of the significance for the MEPS simulation. Whereas the significance decreases towards lower efficiencies for the training with MEPS it stays constantly very high for the training with CDM.

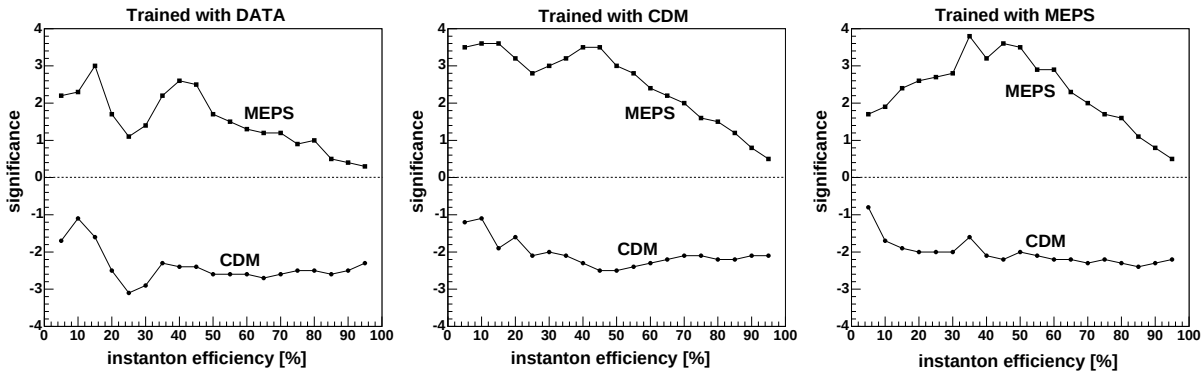

Figure 7.25: Significances in dependence of the instanton efficiency: All curves have been measured with the neural network trainings with three inputs shown in table 7.19.

In summary, the contradictory results of the simulations make a conclusion about the instanton hypothesis difficult. If one believes that the MEPS simulation describes the perturbative QCD event classes correctly, then the instanton hypothesis is confirmed. In particular, the difference between simulated and observed events is significantly larger than the number of predicted instanton-induced events in the three-dimensional study, while the predicted number of instanton-induced events matches well in the five-dimensional case. If one believes that the CDM simulation describes the perturbative QCD event classes correctly, then the instanton hypothesis cannot be confirmed as the results in five dimensions leave no room. Even more important, the CDM simulation predicts more events than found in the data which points to a problem in the simulation. Finally, we have two simulations claiming to describe the perturbative QCD event classes whose predictions differ significantly in the selected region of phase space. Improvements in the understanding of these differences are clearly needed, some aspects which could help have been discussed here.

7.4 Higgs Boson Parity Measurement at a Future Linear Collider

While the next generation of hadron colliders aims, among other "new physics" topics like super-symmetry and large extra dimensions, at the detection of the Higgs Boson, the determination of its properties, in particular of its parity, is a task for a future linear collider. Whereas the standard model Higgs boson must have positive parity (scalar particle), super-symmetric models predict in addition a Higgs boson with negative parity (pseudo-scalar particle). It will be very important to be able to distinguish between these two cases. Section 2.2 already introduced the basic ideas how such a measurement could be done.

7.4.1 Data Source and Preprocessing

The simulated Higgs production events which are used in this analysis are generated by a simulation package already used in many previous studies (see [13] and references therein). Because a simulation is used to generate both the training and the test set arbitrarily many events can be generated for both purposes. For the training of statistical learning methods around 20.000 events will be used (10k from a scalar and 10k from a pseudo-scalar Higgs boson) and the test set consists of 320.000 events (160k for each parity). This large set is used to derive mean significances for the Higgs parity measurement over many pseudo experiments (see below).

The inputs which are available to distinguish scalar-like from pseudo-scalar-like events are calculated within the simulation considering the decay $H \rightarrow \tau^+\tau^- \rightarrow \rho^+\bar{\nu}_\tau\rho^-\nu_\tau \rightarrow \pi^+\pi^0\bar{\nu}_\tau\pi^-\pi^0\nu_\tau$. Among them are the acoplanarity φ^* and the quotients y_1 and y_2 describing the energy distribution among the pions (see section 2.2). The same quotients can also be derived with an alternative reconstruction method which uses the $\tau^\pm$ impact parameters (they are then called y_1^i and y_2^i). This method is described in [13] and returns in addition the IDs id_1 and id_2 describing the geometrical configuration of the τ momenta and pion directions. Furthermore the $\rho^\pm$ energies can be used as inputs as well as the measured four vectors of all pions. Different choices of input combinations have been tried out and the one that performed best in the tests will be used throughout the following analysis. It consists of the nine inputs φ^*, y_1, y_1^i, y_2, y_2^i, id_1, id_2, E_{ρ^+} and E_{ρ^-}.

7.4.2 Training and Evaluation Strategy

Two different analysis strategies will be presented in the next two sections. The first one has been used in the previous studies [13] and underwent many refinement steps [110]. It is based on the acoplanarity (φ^*) distribution which matches a cosine whose phase-shift is determined by the Higgs boson parity. The second strategy is independent of such theoretical prerequisites as it measures directly the difference in the physical observables of both parity states.

Analysis based on a Cosine fitted to the Acoplanarity

It was shown in section 2.2 that the distribution of events in the acoplanarity angle follows a cosine function with some vertical offset if events are selected according to $y_1y_2 > 0$ or $y_1y_2 < 0$, the phase is shifted by 180 degrees for the second case. The main target of the

analysis methods presented here is to generate the clearest possible cosine signal in the acoplanarity distribution. Two components have to be taken into account: On the one hand the amplitude of the cosine should be maximised (by rejecting events which only contribute to a flat background). On the other hand as many events as possible should be used to fit the cosine function because the uncertainty of the amplitude is high with too low statistics. A fitting procedure determines the significance of the cosine signal for a given event set by taking both factors into account: The significance is defined as

$$s = \frac{A}{\sigma_A}, \tag{7.4}$$

the amplitude of the fitted cosine function divided by its uncertainty which is derived from the fit (compare figure 7.28). For the fit the vertical offset of the cosine function is fixed to the mean value (mean number of events per bin) and no phase shift of the cosine function is allowed. A shift would correspond to a mixing of states but this is a different analysis and will not be discussed here [111].

Different training targets and preprocessing cuts will be compared in the next section. The starting point of the analysis is the acoplanarity histogram with "scalar events" for which $y_1 y_2 > 0$ or equivalently with "pseudo-scalar events" for which $y_1 y_2 < 0$. The remaining half of the events can be used in the same way, but the acoplanarity is shifted by 180 degrees (compare the discussion in section 2.2). Using the $\tau^{\pm}$ impact parameters in the reconstruction process means replacing y_1 and y_2 by y_1^i and y_2^i, respectively, and improves the significance slightly [13].

Different preselections will be used in different combinations to improve the significance which is obtained from the histogram described above:

(a) A significant improvement can be achieved by selecting events according to $y_1 y_1^i > 0$ and $y_2 y_2^i > 0$. This means that those events generate a very clear cosine signal for which both reconstruction methods return the same sign.

(b) The significance can be improved further if the pion energies from $\rho^{\pm}$ are compared in a second way. $y_1 y_2 > 0$ meant that for both ρ's the major energy part (relative to their energy sum) should be either in the charged or in the neutral pion. We define

$$z_1 = E_{\pi^+} - E_{\pi^0} \qquad z_2 = E_{\pi^-} - E_{\pi^0} \tag{7.5}$$

where the π_0's are from the respective $\rho^{\pm}$ decays. A selection $sign(z_1 z_2) = sign(y_1 y_2)$ improves the significance by demanding the same relation between the pion energies not relatively but in terms of absolute energies as measured in the laboratory frame.

According to the basic selection with $sign(y_1 y_2)$ (or $sign(y_1^i y_2^i)$) the training targets for statistical learning methods are chosen. The source for the target values is the simulation which knows about the generated energies and momenta, in contrast to the experimental situation where only reconstructed quantities are available. The values x_1 and x_2 are derived from generated values and correspond to y_1 and y_2, respectively. A near optimal significance should therefore be obtained by selecting according to $sign(x_1 x_2)$. The statistical learning methods will thus use $sign(x_1 x_2)$ as the basis for a training target. The different strategies in detail are:

(A) Preselect according to selection (a) above, then use the selection according to $sign(y_1^i y_2^i)$ if this selection is validated (output > cut) by the statistical learning method (else discard the event). The training target is therefore $sign(y_1^i y_2^i) = sign(x_1 x_2)$ (defined as 0 if false and 1 if true).

(B) Preselect according to selection (b) above and then select according to (output > cut) or (output < 1−cut) since the training target is then $(x_1 x_2 > 0)$.

(C) No preselection, select directly according to (output > cut) or (output < 1−cut), the training target is again $(x_1 x_2 > 0)$.

The output distributions are usually symmetric so that indeed (output > cut) performs like $y_1^i y_2^i > 0$ and (output < 1−cut) like $y_1^i y_2^i < 0$ in (B) and (C). To control this behaviour all event types of the test set are used. This means that we use scalar and pseudo-scalar events to test the classification. To clarify the event selection according to $sign(y_1^i y_2^i)$ (or the respective cut in the output) we write down case by case which events are entered into the histogram to calculate the significance:

- scalar events with $y_1^i y_2^i > 0$ (or the respective cut in the output),
- scalar events with $y_1^i y_2^i < 0$ (or the respective cut in the output) with a phase shift in acoplanarity of 180 degrees,
- pseudo-scalar events with $y_1^i y_2^i < 0$ (or the respective cut in the output),
- pseudo-scalar events with $y_1^i y_2^i > 0$ (or the respective cut in the output) with a phase shift in acoplanarity of 180 degrees.

Effectively a phase-shift needs to be applied for each event depending on its simulated parity and depending on $sign(y_1^i y_2^i)$ or the output of the statistical learning method. Using all these event types assures that any detection method is sensitive to both parity types.

Analysis based on a direct Discrimination of Parity States

In contrast to the analysis strategy described above, the training target can be directly defined by the parity: 0 for scalar events and 1 for pseudo-scalar events. The inputs used for the training are the same as mentioned above. The resulting output distributions like the one in figure 7.26 show an accumulation around the mean value 0.5 with different tails to the left and right side. By construction the left tail is higher for scalar events because they were trained to 0 and the right tail is higher for pseudo-scalar events because they were trained to 1.

The significance of a small (N events) event set is

$$s = \frac{\Delta\mu}{\sigma_{\Delta\mu}} = \frac{\mu_N - \mu_{tot}}{\sigma_{\mu_N}} = \frac{\mu_N - \mu_{tot}}{\frac{\sigma_{\mu_{0/1}}}{\sqrt{N}}}. \tag{7.6}$$

Here the difference $\Delta\mu$ of observed mean μ_N to the overall mean μ_{tot} (from both classes, close to 0.5) is divided by its uncertainty $\sigma_{\Delta\mu}$ which is calculated by transforming the standard deviation of each class $\sigma_{\mu_{0/1}}$ to the lower statistics N. The values μ_{tot} and $\sigma_{\mu_{0/1}}$ are measured with large statistics while μ_N is measured for the small event set.

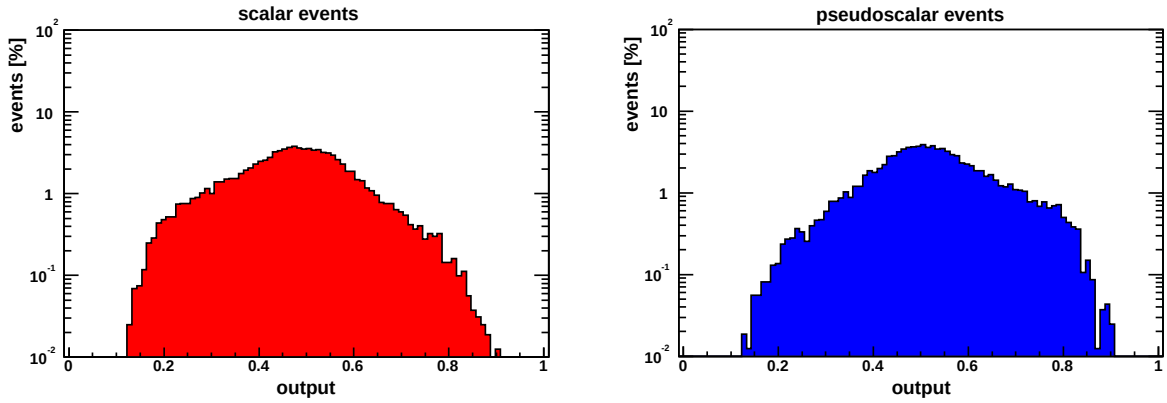

Figure 7.26: Direct discrimination of scalar and pseudo-scalar events: Example of output distributions after training (high statistics).

Technically, the overall mean value μ_{tot} and the variances of the output distributions for scalar and pseudo-scalar events $\sigma_{\mu_{0/1}}$ are calculated by using the whole test set consisting of 160.000 events of each type. As described below several pseudo experiments are done with small parts of the test set. These have their own mean value μ_N and determine thus the difference to the overall mean $\Delta\mu$ and its uncertainty $\sigma_{\Delta\mu}$ given by the number of events N.

Compared to the analysis based on the acoplanarity angle this direct discrimination has the advantage that much less theoretical input is necessary. In addition, no fit is necessary, only means and variances have to be calculated.

Significance obtained from the Pseudo Experiments

The significances are obtained by performing a number of pseudo experiments (300 with scalar and 300 with pseudo-scalar events in the results presented below): A total number of observed events is simulated which has a Gaussian distribution around its mean given by the luminosity, cross section and detection efficiency assumptions (this mean will be 500 events in the results presented below). The events of each of these pseudo experiments are passed through the selection cuts and a significance for the Higgs parity (equation 7.6) is calculated. The events for the pseudo experiments are taken from the large test set which has not been used in the training. Performing many pseudo experiments does not only result in a mean significance but also in a standard deviation of the distribution of significances (compare figure 7.27). With this information a confidence interval can be constructed for the significance to be determined in a future linear collider experiment.

7.4.3 Results

In the next two sections the results for the described training and evaluation methods will be presented. The first section presents significances obtained with the cosine fit in acoplanarity. This method was used in all previous analyses (see [13] and references therein), and is thus important to have a comparison but also as a base line, from which improvements by statistical learning methods can be judged. The second section presents results for the direct discrimination of parity states which shows in all its simplicity an

even greater potential than the "analytical" method.

The choice of different strategies is dominated by the different preselection techniques. They will be combined with two statistical learning methods: neural networks and random forests which have both proven their high potential in the previous analysis sections. All mean significances and standard deviations (RMS) shown below are calculated from 600 pseudo experiments with 500 events each on average, as discussed above. Figure 7.27 shows an example how 600 pseudo experiments result in a distribution of significances of which the mean and RMS can be determined.

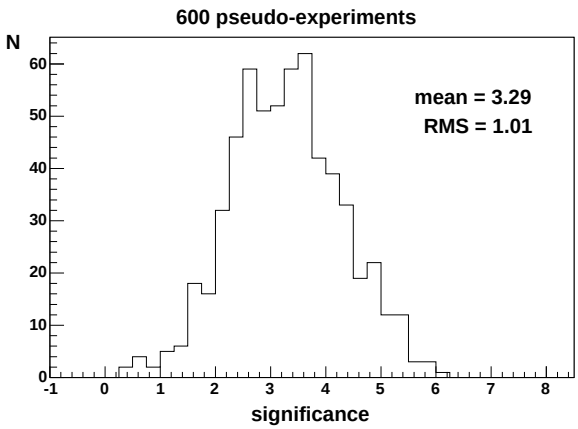

Figure 7.27: Example for a distribution of significances derived from 600 pseudo experiments.

Results based on a Cosine fitted to the Acoplanarity

Figure 7.28 shows an example how the significance of the cosine signal is determined from the histogram. There a χ^2-fit is used on a coarsely binned histogram. In practice – but less suitable to be shown – the fit is performed with a maximum likelihood technique on a histogram with fine binning.

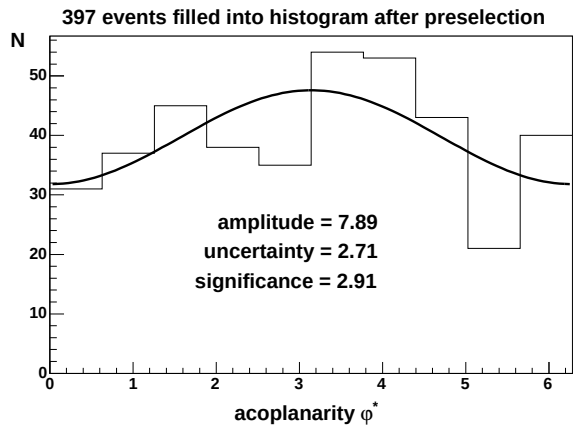

Figure 7.28: Example for an acoplanarity (φ^*) histogram with fitted cosine for one pseudo experiment. The 397 events in the histogram (of totally 500 events) are those that survived the preselection cuts. The amplitude and its uncertainty are derived from the fit, the significance is the quotient.

Table 7.20 summarises the significances and their RMS which can be obtained by using the classical selection with the preselections (a) and (b) discussed above. The first two lines show theoretical limits as they use the generated values x_1 and x_2. The other significances yield the basic performance which should be outbid by the statistical learning methods.

The neural networks and random forests which are used were selected from multiple trainings with different parameter sets by selecting the least number of misclassifications on the selection set. Details about the parameter optimisation can be found in appendix B.3.

Preselection	Phase-shift based on	significance	RMS
–	$sign(x_1x_2)$	7.38	1.18
$sign(z_1z_2) = sign(y_1^iy_2^i)$	$sign(x_1x_2)$	6.67	1.27
–	$sign(y_1^iy_2^i)$	3.64	1.03
$y_1y_1^i > 0$ & $y_2y_2^i > 0$	$sign(y_1^iy_2^i)$	4.89	1.12
$y_1y_1^i > 0$ & $y_2y_2^i > 0$ & $sign(z_1z_2) = sign(y_1^iy_2^i)$	$sign(y_1^iy_2^i)$	5.09	1.14

Table 7.20: Classical results for the significance of the parity determination: The first two lines show theoretical results based on the generator level values x_1 and x_2. They represent the maximum possible significances which could be obtained by any of the strategies.

Table 7.21 presents the significances and their RMS which can be obtained by using a neural network trained according to the strategies (A)-(C) (see above). The optimal significances are about 9% better than the best classical strategy (5.56 for preselection (b) and training strategy (A) compared to 5.09 for preselection (b) and "classical" selection according to $sign(y_1^iy_2^i)$). It is obvious that the neural network profits from the preselection in the same way as the classical strategies do.

Preselection	Training Strategy	significance	RMS
$y_1y_1^i > 0$ & $y_2y_2^i > 0$	A	5.37	1.26
$y_1y_1^i > 0$ & $y_2y_2^i > 0$ & $sign(z_1z_2) = sign(y_1^iy_2^i)$	A	5.56	1.33
$y_1y_1^i > 0$ & $y_2y_2^i > 0$	B	5.40	1.33
$y_1y_1^i > 0$ & $y_2y_2^i > 0$ & $sign(z_1z_2) = sign(y_1^iy_2^i)$	B	5.51	1.38
$y_1y_1^i > 0$ & $y_2y_2^i > 0$	C	4.87	1.19
$y_1y_1^i > 0$ & $y_2y_2^i > 0$ & $sign(z_1z_2) = sign(y_1^iy_2^i)$	C	5.53	1.35

Table 7.21: Neural network results for the significance of the parity determination – based on a fit in the acoplanarity φ^*.

Table 7.22 presents the significances and their RMS which can be obtained by using a random forest trained according to the strategies (A)-(C). Only one preselection is used here as this one showed the best performances for the classical strategy and for the neural networks. The increase in significance is almost identical to the neural network results.

Preselection	Training Strategy	significance	RMS
$y_1y_1^i > 0$ & $y_2y_2^i > 0$ & $sign(z_1z_2) = sign(y_1^iy_2^i)$	A	5.52	1.38
$y_1y_1^i > 0$ & $y_2y_2^i > 0$ & $sign(z_1z_2) = sign(y_1^iy_2^i)$	B	5.54	1.39
$y_1y_1^i > 0$ & $y_2y_2^i > 0$ & $sign(z_1z_2) = sign(y_1^iy_2^i)$	C	5.56	1.32

Table 7.22: Random forest results for the significance of the parity determination – based on a fit in the acoplanarity φ^*.

Results based on a direct Discrimination of Parity States

The neural networks and random forests which are used in the tables below were selected from multiple trainings with different parameter sets by selecting the least squared error on the selection set. The inputs are still the same, as mentioned in section 7.4.1. Details about the parameter optimisation process can again be found in appendix B.3. Table 7.23 shows the neural network results for the different preselections. The increase of 23% in significance compared to the best classical strategy is a real success.

Preselection	significance	RMS
–	4.28	1.03
$y_1 y_1^i > 0$ & $y_2 y_2^i > 0$	6.15	1.26
$sign(z_1 z_2) = sign(y_1^i y_2^i)$	5.65	1.14
$y_1 y_1^i > 0$ & $y_2 y_2^i > 0$ & $sign(z_1 z_2) = sign(y_1^i y_2^i)$	6.26	1.27

Table 7.23: Neural network results for the significance of the parity determination – based on direct discrimination.

The results for the random forest method shown in table 7.24 are very similar to the significances obtained with the fitting method and do not show the significant increase like for the neural networks. However these numbers confirm again the relative magnitude of the significances for the different preselection techniques.

Preselection	significance	RMS
–	3.67	1.01
$y_1 y_1^i > 0$ & $y_2 y_2^i > 0$	5.34	1.20
$sign(z_1 z_2) = sign(y_1^i y_2^i)$	5.12	1.17
$y_1 y_1^i > 0$ & $y_2 y_2^i > 0$ & $sign(z_1 z_2) = sign(y_1^i y_2^i)$	5.57	1.33

Table 7.24: Random forest results for the significance of the parity determination – based on direct discrimination.

Summary of Results

If one wants to derive statements with as much predictive contents as possible it makes sense to calculate a confidence boundary for the significance based on the standard deviations (RMS). This means that the significance which will actually be found in the experimental data will most probably be larger than a certain level of significance. A 90% one-sided confidence interval for a Gaussian probability distribution is bounded by $\mu - 1.3\sigma$. We can therefore derive from the numbers above that one can expect (90% confidence) from the classical methods a significance larger than 3.61. For a neural network trained on the direct discrimination of parity states one can expect (90% confidence) a significance larger than 4.61 which is an increase of 28% and comes very close to the "magic barrier" of 5σ which is often taken as the minimum significance needed to claim the detection of a signal. The assumptions regarding the accumulated luminosity have been discussed in section 2.2.

7.4.4 Future Research

Now that the discrimination of the scalar and pseudo-scalar Higgs bosons with statistical learning methods was that successful, the analysis can be extended. A very interesting subject for future research on this topic is the question for a mixing angle of scalar and pseudo-scalar Higgs bosons. The impact of a mixing angle on the measured quantities is easily seen in the acoplanarity distribution: There the mixing angle is identical to a shift of the cosine function (as we have seen in section 2.2 that the distributions of scalar and pseudo-scalar events have a shift of 180 degrees). The first method to determine the parity type which performs a fit of the acoplanarity distribution could take a variable shift into account. Of course, a direct discrimination of mixing angles, either based on several classifications or based on a regression, has to be studied as well since this strategy was shown to lead to the significant improvements.

7.5 Position Measurement for a Small-Angle Neutron Scattering Detector

The neutron scintillation detector presented in section 2.3 has to cope with a high event rate. Therefore it is not possible to log the full detector information for each event. Instead the incident position of the neutron is reconstructed online and stored in a histogram. This offers the possibility for the application of fast statistical learning methods.

7.5.1 Data Sources

The desire to test the capabilities of statistical learning methods for the reconstruction of the incident position of the neutron faces one main problem: It is very difficult to generate training data for this application. Of course, there exists a simulation describing the generation of scintillation light and the electronics up to the final ADC count value. However, there might be details in the experimental data which are not included to a sufficient level in the simulation so that a statistical learning method for the position reconstruction may be misled.

To modify the experiment to generate training data directly is a very difficult and time consuming task: A mask with only a few holes could be placed in front of the detector (compare the mesh experiment in section B.2). But it then needs to be moved into many different positions and data has to be collected for each of them. Up to now the desire to do analysis with the detector has left no time window to perform such a measurement.

In section 2.3 the current standard reconstruction method was introduced. In principle this reconstruction method can be used to create a training set for statistical learning methods. The problem is that the statistical learning method is not likely to perform any better than the standard reconstruction. Independent of the performance one target of this analysis was to study the possibility whether the standard reconstruction method could be replaced by a fast statistical learning method implemented in hardware. For this target, the standard reconstruction method can be used as a teacher and neural networks are a perfect candidate to cope with this task.

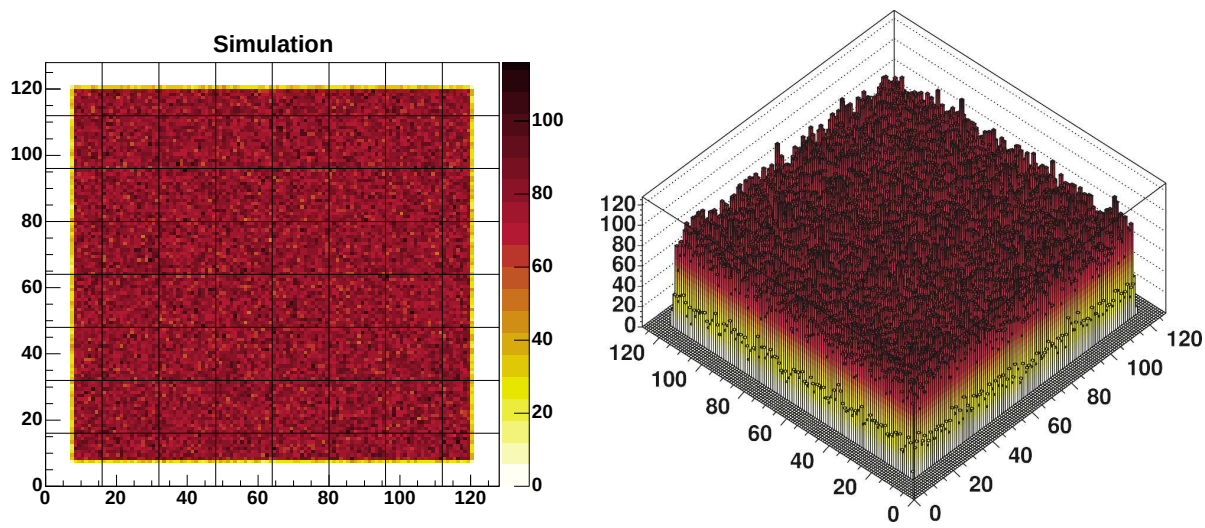

Figure 7.29: Dataset of simulated neutron events with a flat illumination.

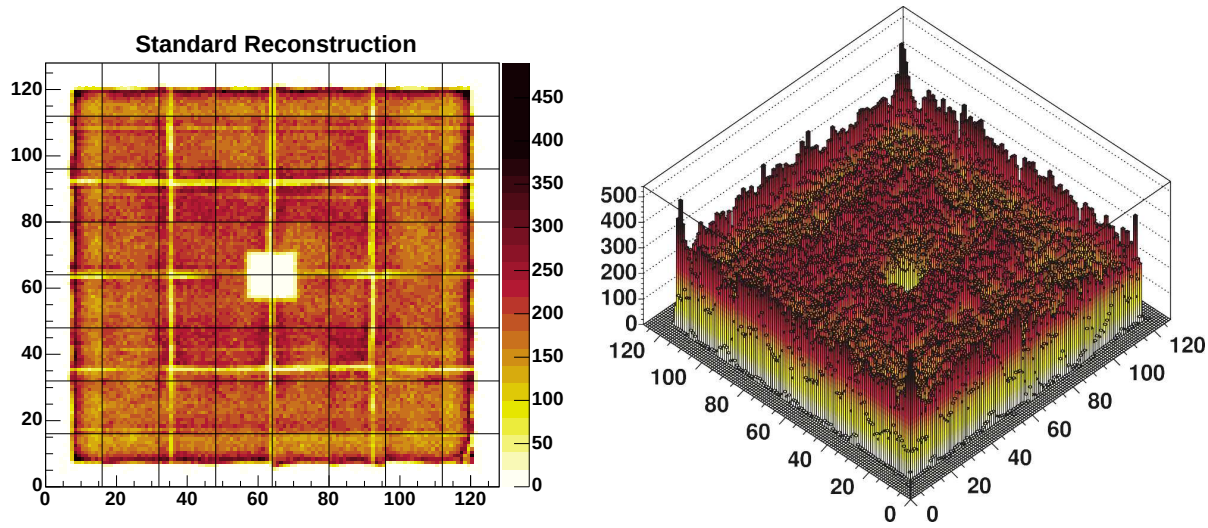

Figure 7.30: Standard reconstruction of the calibration dataset (flat illumination). The observed blind regions are due to the detector structure (the beam stop in the centre and the small gaps between the scintillation plates).

Figure 7.29 shows a simulated dataset with a homogenous illumination of the detector by neutrons. The structure of 8 times 8 squares show the approximate positions of the photomultiplier tubes (compare figure 2.20). Figure 7.30 also shows a homogenous illumination of the detector but here for experimental data for which the positions have been reconstructed using the standard reconstruction. This plot shows some detector effects, for example the blind regions between the 4 times 4 scintillator plates (compare figure 2.20). This homogenous illumination can be used as a calibration dataset by checking the flatness of the reconstructed event distribution.

7.5.2 Preprocessing

Two statistical learning methods will be in the focus of this analysis: The naive Bayes method ("maximum likelihood") was already discussed for this special application in section 2.3. The neural network method will be studied as mentioned above in particular because of the possible online application. Both methods localise the reconstruction by taking into account only 3 times 3 photomultiplier values with the maximum value centred. This means that the maximum likelihood method builds only a product over these at most 9 photomultipliers and the neural network uses only these at most 9 inputs. This helps significantly to reduce the noise coming mainly from the photomultipliers which did not collect scintillation light. Figure 7.31 illustrates this problem with one event from the calibration dataset and one simulated event (compare figure 2.20).

7.5.3 Results

The first attempt to create a position reconstruction with statistical learning methods is based on the hope that the simulated data can be used without bias. Figure 7.32 shows the reconstructions of the simulated dataset with the neural network and the maximum likelihood method. Both plots show that the homogenous illumination can be reconstructed

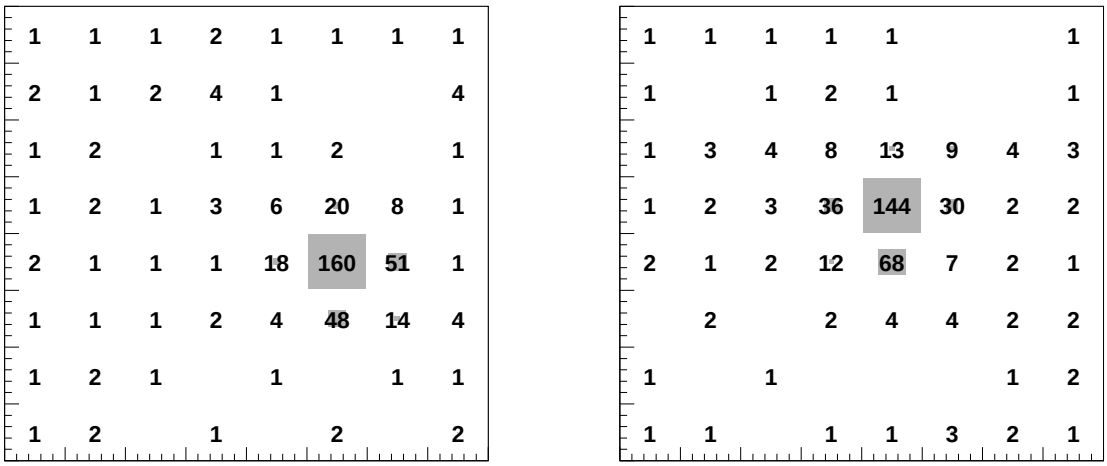

Figure 7.31: Event structure in the neutron detector: Left one event from the calibration dataset, right a simulated event.

very well. The average reconstruction error (RMS) for the neural network is 0.53 channels (2.8cm) and 0.69 channels (3.6cm) for the maximum likelihood method.

The application of the trained reconstruction methods to the experimental data, however, reveals that some detector effects are not included in the simulation (figure 7.33). Artefacts in the form of "too dark" or "too bright" regions appear in the reconstruction of the experimental data which should show a homogenous illumination. The maximum likelihood method shows in addition a very noisy behaviour which makes it inacceptable for the reconstruction. The neural network reconstruction would be an interesting alternative to the standard reconstruction if the problematic areas could be corrected by small corrections to the simulation.

As an alternative to the training with the simulation, the standard reconstruction can be used as the training information. By this the statistical learning methods can be trained with experimental data, problems due to an incomplete simulation are then no problem. As said above, however, one cannot expect superior performance compared to the target "standard reconstruction". Figure 7.34 shows the reconstructions of the experimental data with the neural network and the maximum likelihood method. Both plots show reconstructions without obvious differences to the standard reconstruction except a few "ghost signals" in the central blind area generated by the maximum likelihood method. The flat distributions demonstrate the capability of the two statistical learning methods to reproduce the standard reconstruction. The average reconstruction error (RMS) measured with respect to the standard reconstruction is 0.29 channels (1.5cm) for the neural network and 1.59 channels (8.3cm)for the maximum likelihood method.

A direct comparison of the reconstruction of the calibration dataset with neural networks trained with the simulation and with the standard reconstruction is shown in figure 7.35 in the form of two projections onto the y-axis. The blind regions of the detector have been cut away for this projection so that the mean value and its variance are measured per y-channel only for the pixels which should be uniformly illuminated.

One can see that the simulation-trained reconstruction shows for example two rows

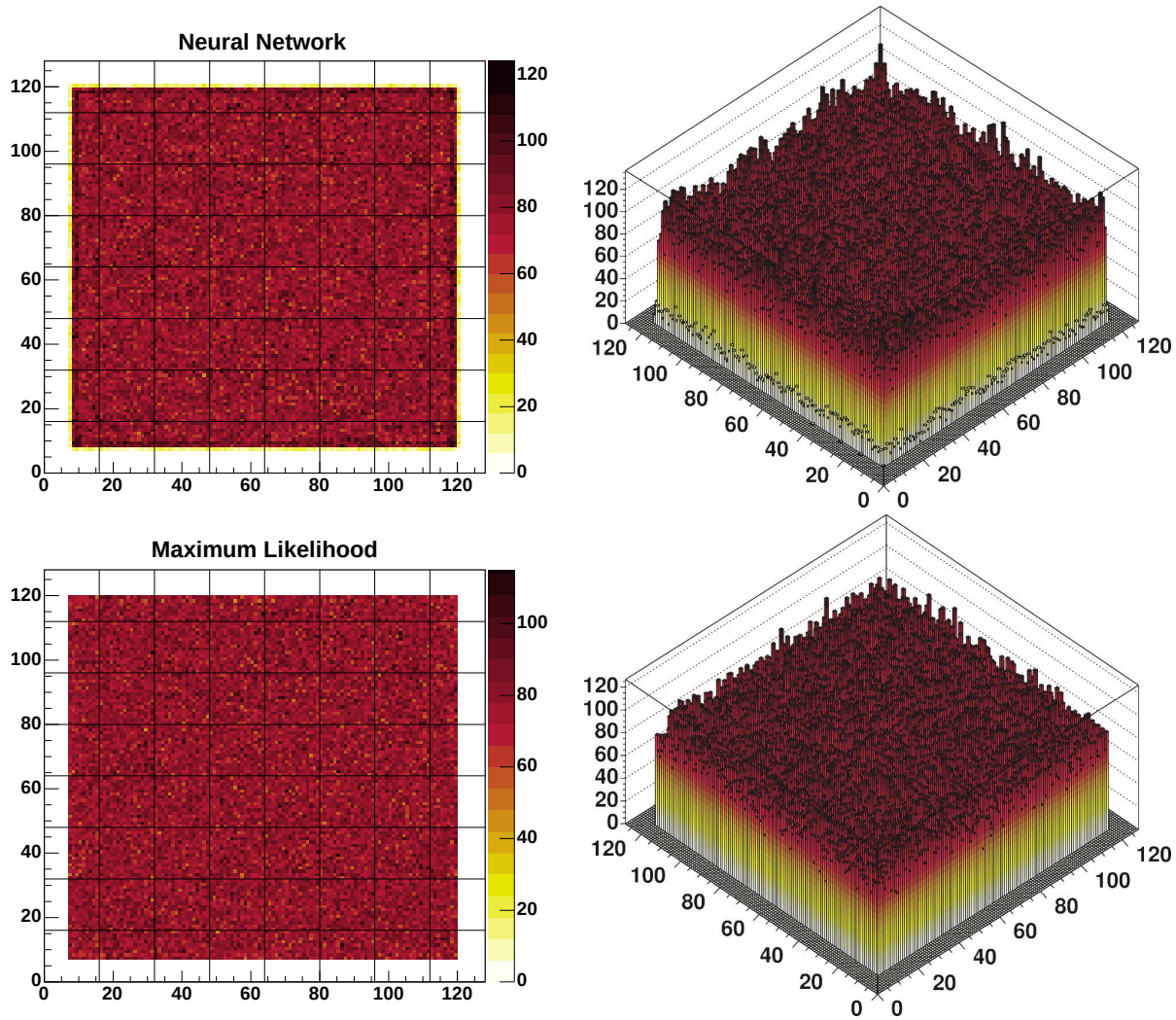

Figure 7.32: Reconstruction of the simulated dataset with statistical learning methods.

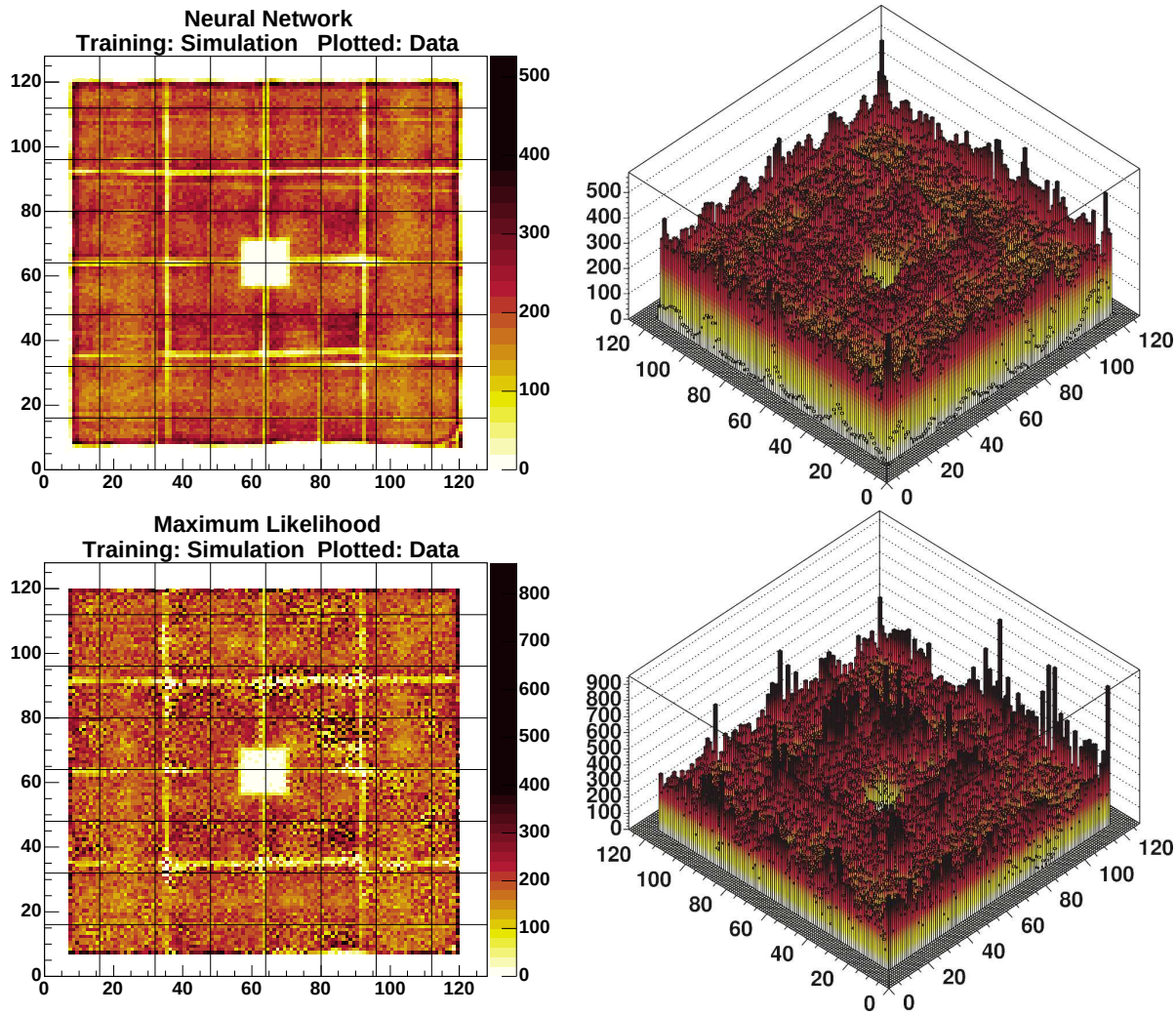

Figure 7.33: Reconstruction of the calibration dataset with statistical learning methods which were trained with the simulation.

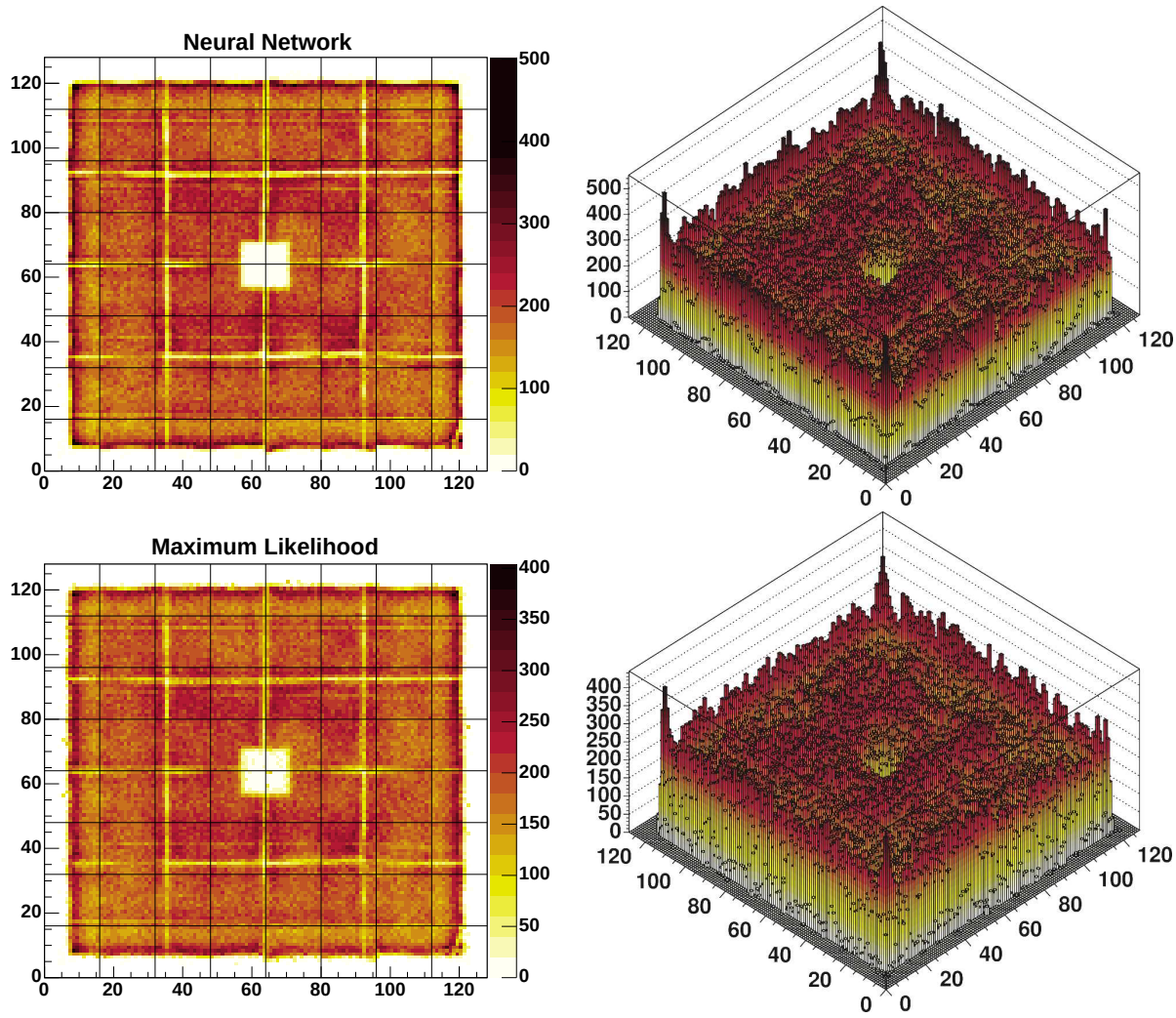

Figure 7.34: Reconstruction of the calibration dataset with statistical learning methods.

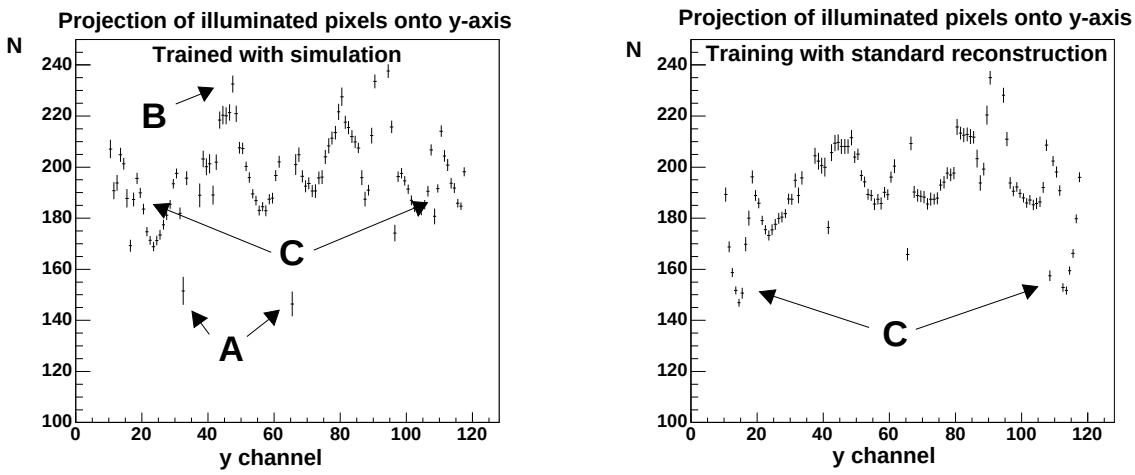

Figure 7.35: Mean and variance in the projections onto the y-axis of the illuminated pixels for neural networks which were trained with the simulation (left) or with the standard reconstruction (right).

with too few events (marker A in figure 7.35) and a too high peak in between (marker B). But it shows a nicer behaviour towards the outer borders where it is flatter than the standard-trained reconstruction (markers C). This means that the simulation is in principle perfectly suited to act as teacher for this reconstruction if only the problematic areas could be fixed.

7.6 Applications for the MAGIC Telescope

The data analysis for the MAGIC telescope offers many possibilities for the application of statistical learning methods. The trigger system, for example, could make use of very fast classification techniques implemented in hardware to yield a sufficiently low rate but still with very high efficiency for specific classes of events. In particular, this would allow observations towards a very low energy threshold where unfiltered trigger rates dramatically increase. The image cleaning as a second example could also make use of statistical learning methods: The number and arrival time of Cherenkov photons in a pixel as well as the same quantities from the neighbouring pixels could be used to distinguish between noise and signal.

The two applications which are currently the most important issues in the MAGIC data analysis have been introduced in section 2.4 and will be discussed in the following: The suppression of the hadronic background, which has a several orders of magnitude higher rate, is essential to see a clear photon signal. Secondly, the estimation of the energy of photon signals is important to derive any spectral information.

The focus in this section will lie on the comparison of standard methods used in MAGIC [112] and the statistical learning methods random forest and neural network. This focus reflects the current software situation in the MAGIC analysis framework: Standard methods like those which were already used in the predecessor experiment CT1 are of course implemented, the random forest algorithm was also implemented quite early and led to nice improvements compared to the standard methods. Neural networks finally have always been discussed as an alternative but they have not found their way into the software framework so that no comparisons were available.

7.6.1 Gamma-Hadron Separation

The standard method for the separation of events which originated from either photons or hadrons is called "supercuts". This is a parameterised model based on the Hillas parameters (see section 2.4). The parameters are optimised for a given pair of ON and OFF datasets by maximising the significance of the γ-excess obtained from the α-plot after applying the supercuts.

Two different approaches towards γ-hadron separation will be discussed and results for each of them will be presented. The first approach originates from the idea to train a statistical learning method without having to use a Monte Carlo simulation since the data has not been described well by the simulations in the starting phase of the experiment (see also the discussion in the previous section on the neutron detector). The classical approach to train photons from a simulation against hadrons from OFF data is presented as the second approach.

Training without Simulation

The Monte Carlo simulation of very high energy photon events in the MAGIC telescope is difficult and showed some inconsistencies in the starting phase of the experiment. To perform the γ-hadron separation with statistical learning methods examples of photon induced events must be available. This section presents results from an approach which circumvents the simulation by taking an ON dataset enriched with γ-showers, to represent

the photon-induced events. Since the whole ON dataset would have too many hadrons (order of 10^4 more than photons), we take a subset of the data on which the supercuts method was already applied. In the α-plot a photon signal is then seen somewhere below 15 degrees. The photon sample for the training is extracted after application of supercuts and the condition $\alpha < 15°$ (see figure 7.36). The background is formed by hadron-induced events from an OFF dataset which also survived the supercuts. The hope is that the characteristics of the photons can be found by the statistical learning methods even if there is still background in the "signal" selection. Thus for both training and evaluation the supercuts method is always applied first, before applying a statistical learning method in addition. This leads to a higher background suppression and – with a sufficient photon efficiency of the learning method – to a gain in significance.

With this method the number of excess events can only be reduced, as the excess after applying the supercuts is an estimate for the true number of selected photons. Any additional cut can only cut away more photons. Any increase in the number of excess events has to be an artefact. Nevertheless a higher significance with a similar number of excess events is a valuable target.

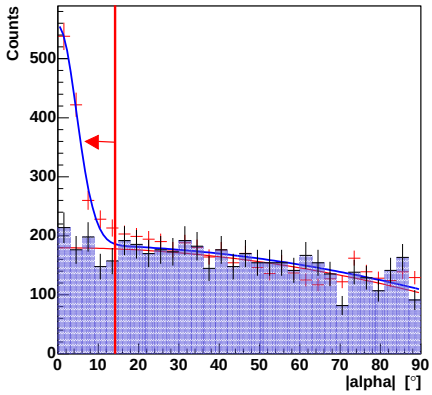

Figure 7.36: Supercuts and α-cut principle: The photon signal for the training is taken from the shown region after application of the supercuts. There is still significant hadronic background but the photons are hopefully recognised by the learning method.

The inputs for the statistical learning methods consist of several parameters from the Hillas analysis (see section 2.4.2) and some extensions: `length`, `width`, `dist` and `size` are known, furthermore `newdist` is used, a variation of `dist`, calculated in a different way [112], furthermore `asym` which describes the asymmetry of the ellipse with respect to the pointing direction, `conc` describes the concentration of the Cherenkov light among the illuminated pixels and `leakage` tells about the estimated amount of Cherenkov light which missed the detector.

The observational data which is used here to demonstrate the power of the statistical learning approach was obtained on 15th February 2004 during the commissioning phase of the telescope. The observation target was Markarian 421 for which 105 min ON data and 50 min OFF data were taken in a zenith angle range of 9° to 28°. Both datasets were divided into a training and test set to be able to detect overtraining effects – which is also important for the supercuts method. The datasets and the status of the telescope during this period are described in the analysis [24] on which the following results are based.

For the training of the random forest method and the neural network the sample representing the photons was defined by the ON training set after application of the supercuts and for $\alpha < 15°$ as described above. Accordingly the sample representing the background was taken from the OFF training set after application of the supercuts (for all values of

α)[8]. Trainings with different parameter settings were done (compare appendix B.3) and a neural network with 10 hidden neurons and a random forest with 20 trees were selected by checking the significance using the training data (compare the discussion at the end of the section about the γ-hadron separation).

Figure 7.37 shows the dependence of significance [25] and number of excess events depending on the cut in the output of the statistical learning method on top of the background suppression given by the supercuts method. Only the test set is used here. As discussed in section 2.4, the significance derived from the α-plot takes into account the relative excess of the ON data to the OFF data (measured here below 6°) as well as the statistical uncertainties which are relatively larger the more the background is suppressed. The number of excess events is

$$N_{ex} = N_{ON} - N_{OFF} \tag{7.7}$$

where N_{ON} is the number of selected ON events and N_{OFF} is the number of selected OFF events (both here below 6°). N_{ex} gives an estimate for the true number of selected photons. The combination of a high significance and a high number of excess events is important for the derivation of an energy spectrum as both influence directly the uncertainty of the measured photon flux.

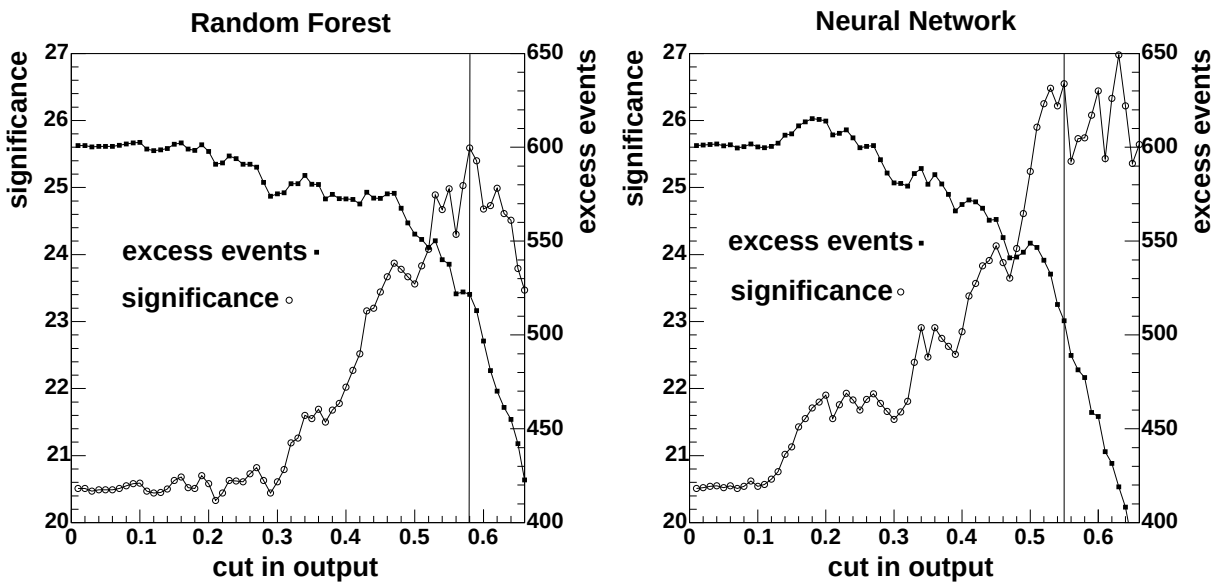

Figure 7.37: Significance and number of excess events for Markarian 421, depending on the cut in the output for the random forest method and for the neural network.

It is clearly visible that both methods start with the supercuts result for a cut at 0 (no additional background suppression). With a larger cut the significance rises due to the better background suppression while the number of excess events falls (with a small rising artefact for the neural network).

[8]It was also tried out to restrict the background to $\alpha < 15°$ but then the background in the region $\alpha > 15°$ was not rejected well.

Figure 7.38 shows the α-plots for the test set for the selected cuts (shown by the lines in figure 7.37, corresponding to the highest significance[9]). The stated significance is measured for $\alpha < 6°$ like in the previous analysis [24].

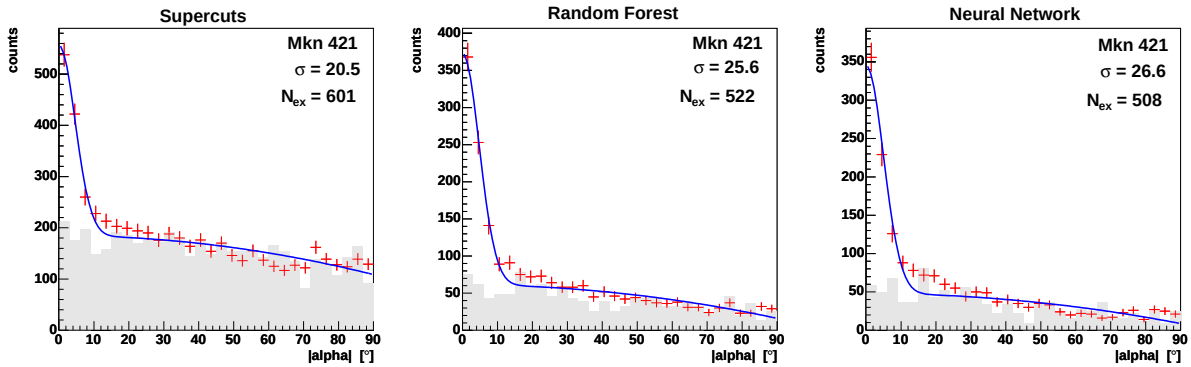

Figure 7.38: Different significances in the alpha plot (Markarian 421, 15.02.2004) for supercuts, random forest and neural network. Shown is the background (OFF data) in light grey and the ON data as crosses with a fitted polynomial plus Gaussian excess.

Supercuts are always applied and lead to a significance of 20.5 by themselves. Applying the random forest method in addition improves the significance by 24%, applying instead the neural network afterwards leads to a remarkable improvement of 30%. The loss in the number of excess events is 13% for the random forest and 15% for the neural network. Although the gain in significance may be worth the additional background suppression, the loss in the number of excess events is motivation enough to train with simulated photons as soon as the Monte Carlo simulation fairly matches the experimental conditions.

Training with Simulation

Two observations from September and October 2004 will be used in this section. The observation times are for the Crab Nebula ON data 140 min, for the 1ES1959 ON data almost 7 hours. The OFF data which is shared for both analysis was taken for the Crab Nebula and has almost 3 hours observation time. All datasets share the same zenith angle range of 35° to 47°.

The γ-hadron separation for the two observation targets will be performed with statistical learning methods which have been trained with simulated photons as signal and part of the 1ES1959 ON data as background. As mentioned, the ON dataset consists to a good approximation only of background. Discarding the Crab OFF data for the training has the advantage that no overtraining effects can appear so that always the whole ON and OFF datasets can be used as a test set in the analysis[10]. The supercuts method has been optimised with simulated photons and Crab OFF data and thus may show some overtraining[11]. All numbers derived from α-plots in the following are based on a preselection which

[9]For the neural network the optimum significance at the cut 0.63 was ignored because of the noisy behaviour of the significance in this region and because of the much lower number of excess events.

[10]In addition only less than 1% of the events from the 1ES1959 ON dataset have been taken for training so that overtraining effects are very unlikely.

[11]Normally the supercuts are directly optimised with the ON and OFF dataset. For 1ES1959, however, the photon excess is very small so that a different technique was chosen. The optimisation of the parametric

includes some technical cuts to create a clean event sample, most important `size`>2000 photons.

The inputs used for the statistical learning methods are the Hillas parameters or quantities directly derived from them: Like above, `length`, `width`, `size` and `conc` are used, as well as the two combinations log(`size`)/log(`length`) and log(`size`)/log(`width`). In addition, `M3long` is used which describes the third moment of the Cherenkov ellipse. The random forest results shown in this section are based on a classifier which has been trained with the random forest implementation of the MAGIC analysis software [113].

Figure 7.39 starts with the results for the Crab Nebula by showing the important relationship between significance and the number of excess events. As mentioned above, the combination of a high significance and a high number of excess events is important for the derivation of an energy spectrum as both influence directly the uncertainty of the measured photon flux.

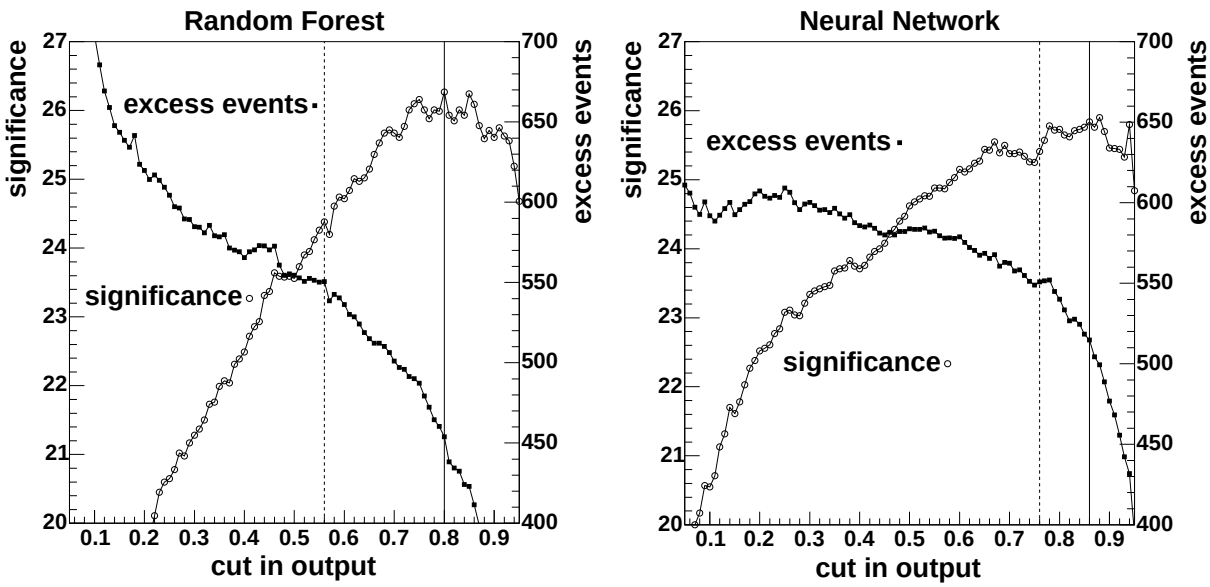

Figure 7.39: Significance and excess events vs. cut in output for Crab Nebula: The cut 0 would mean no background suppression, the solid line marks the cut for optimal significance and the dotted line marks the cut for the background suppression which matches the supercuts result (98.5%).

The behaviour of significance and number of excess events in figure 7.39 depending on the cut in the output of the statistical learning method is typical: A cut at 0 would mean that we take all events without any γ-hadron separation. The resulting significance would be below 5 (not shown here). With an increasing cut the background suppression becomes stronger and the significance rises while the number of excess events naturally falls because the true number of selected photons (for which the number of excess events is an estimate) can only decrease with a stronger cut. For very strong cuts finally, the significance starts to fall again since then the loss in statistics outweighs the better background suppression.

models inside the supercuts method is done by generating an artificial ON dataset mixing simulated photons into an OFF observation. The supercuts are then optimised to give the largest significance of this artificial ON dataset.

The cuts in the output which are marked by a solid line are used for figure 7.40 to show the resulting α-plots. In addition to the random forest and the neural network results the supercuts result is shown. The supercuts method was optimised to give the largest significance, the cuts for statistical learning methods have also been chosen to give the best significance.

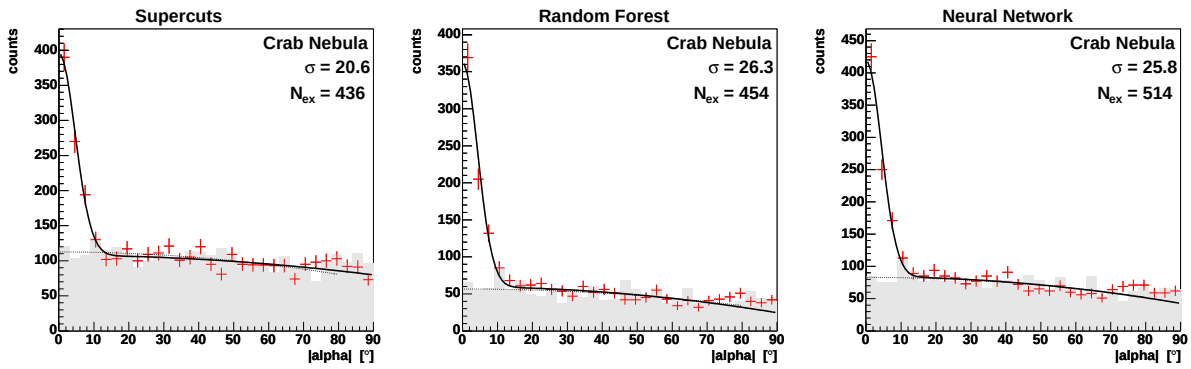

Figure 7.40: Crab results with cuts which are optimised for significance: Both statistical learning methods lead to a higher significance and more excess events than the supercuts method.

Both statistical learning methods show a similar optimal significance (random forest slightly better than neural network) which is improved by 28% with respect to the supercuts result. In addition, the number of excess events is higher if the γ-hadron separation is done with statistical learning methods. The number of excess events is increased by 18% with the neural network.

The dotted lines in figure 7.39 mark the cuts which are used to produce the α-plots shown in figure 7.41. In the latter, the background suppressions of random forest and neural network were set to match the background suppression of the supercuts which is about 98.5%.

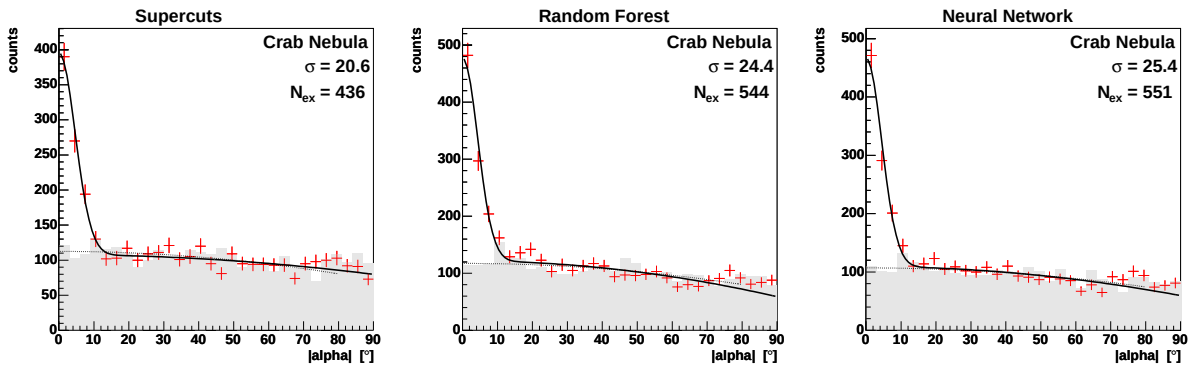

Figure 7.41: Crab results for identical background suppressions: Both statistical learning methods lead to a higher significance and more excess events than the supercuts method.

Again both statistical learning methods show a similar significance (neural network slightly better than random forest) which is now improved by 23% with respect to the supercuts result. The slightly smaller improvement in the significance is compensated by a gain in excess events which is about 25% for both methods.

The tradeoff of significance vs. number of excess events shown in figure 7.39 as well as both cut-selections in figure 7.40 and 7.41 show clearly that the statistical learning methods not only allow to choose which relation between significance and photon excess is appropriate[12]. They also result in a combination of a higher significance and an increase in the number of excess events which makes the statistical learning methods clearly superior to the classical supercuts method.

In contrast to the Crab Nebula, 1ES1959 is a very weak source. Despite the long observation time the significance of the photon excess is hardly above the "detection threshold" 5σ. Figure 7.42 shows like for the Crab Nebula the dependence of significance and number of excess events on the cut in the output for the two statistical learning methods. These curves are much noisier than for the Crab Nebula, especially for the random forest, because of the weakness of the signal.

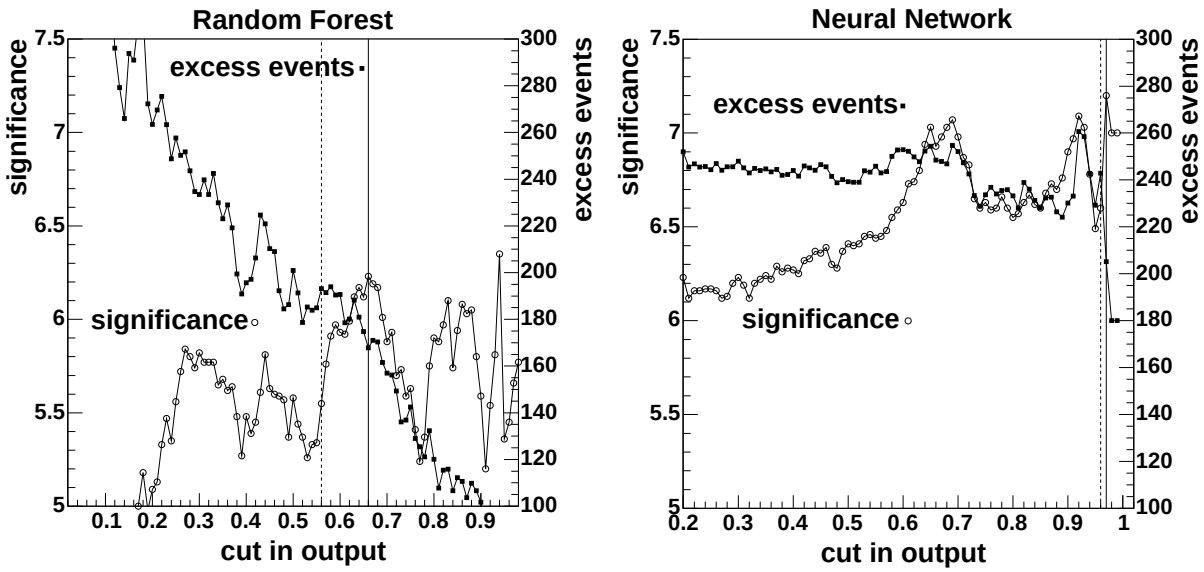

Figure 7.42: Significance and excess events vs. cut in output for 1ES1959: The cut 0 would mean no background suppression, the solid line marks the cut for optimal significance and the dotted line marks the cut for the background suppression which matches the supercuts result (98.5%).

The chosen cuts for the optimal significance are again marked by a solid line and the corresponding α-plots are shown in figure 7.43, together with the supercuts result.

Like for the respective Crab result, the statistical learning methods show a better background suppression combined with a higher photon efficiency which leads to a higher significance and more excess events. The random forest method improves the supercuts significance by 44%, the neural network by 67%. This sign for a performance difference between random forest and neural network is confirmed by the different number of excess events: While the random forest shows an increase of 17% with respect to the supercuts result, the neural network has 43% more excess events.

Like for the Crab dataset also here the dotted lines in figure 7.42 mark the cuts for which the background suppression of all three methods is equal to 98.5% (fixed by the

[12]This tradeoff between significance and excess events is not possible with the supercuts method.

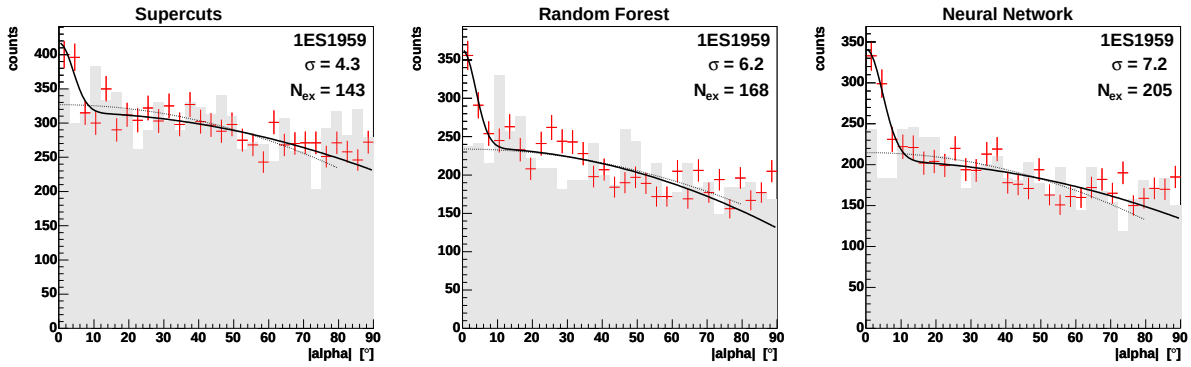

Figure 7.43: 1ES1959 results with cuts which are optimised for significance: Both statistical learning methods lead to a higher significance and more excess events than the supercuts method. The neural network performs better than the random forest method.

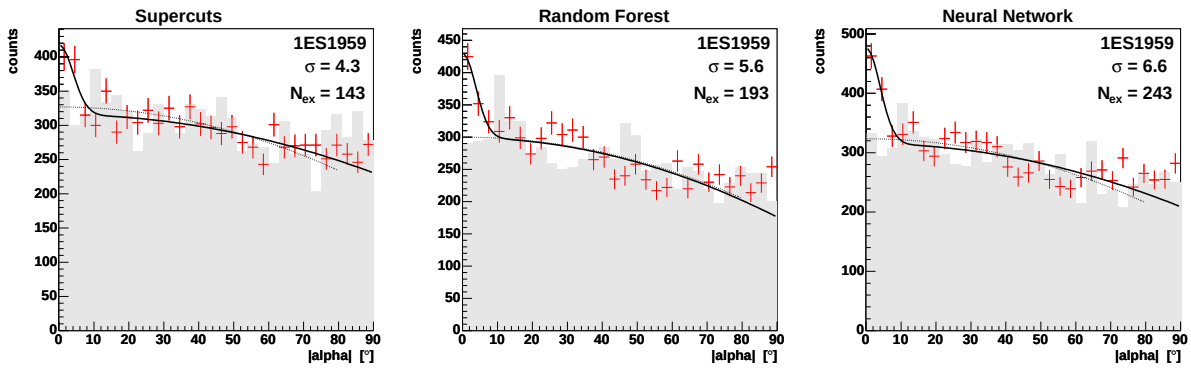

Figure 7.44: 1ES1959 results for identical background suppressions: Both statistical learning methods lead to a higher significance and more excess events than the supercuts method. The neural network performs better than the random forest method.

supercuts optimisation). Figure 7.44 shows the corresponding α-plots.

The same observation as for the Crab dataset can be made here: We still see major improvements in significance and in the number of excess events but now the increase in significance is slightly smaller (30% for random forest and 53% for neural network) and the increase in the number of excess events greater (35% for random forest and 70% for neural network).

In summary, the weak source 1ES1959 shows very clearly – more dramatically than the strong Crab Nebula – that statistical learning methods have to be considered as a very remunerating alternative to the classical supercuts method. The improvements in significance and number of excess event – which are the most important quantities to determine the photon flux of a source with small uncertainty – are significant.

The analysis of source 1ES1959 is a good opportunity to present a new physics result which has been obtained with statistical learning methods: Figure 7.45 presents the photon flux of 1ES1959 which has been measured by the MAGIC telescope in the above mentioned periods. The difference between the two plots is the method for the γ-hadron separation. The two statistical learning methods neural network and random forest are used. The

supercuts method would result in much larger uncertainties since its γ-hadron separation performs so badly (compare figure 7.43).

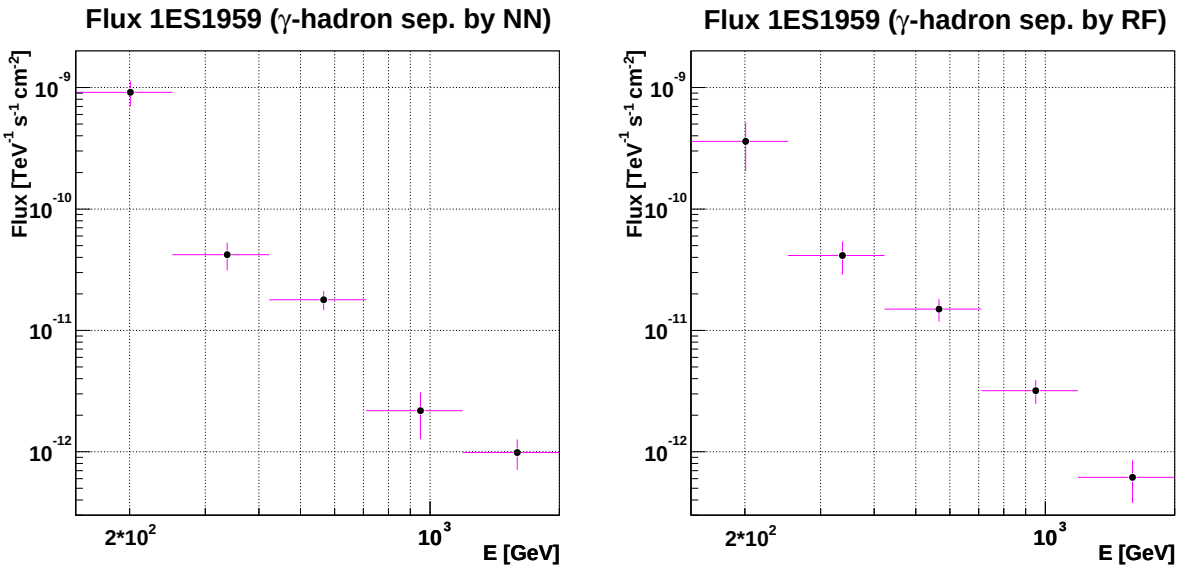

Figure 7.45: Flux measurement for 1ES1959: Left a neural network is used for the γ-hadron separation, right a random forest is used.

This analysis is very preliminary. The differences between the two flux spectra obtained will thus not be discussed. To have a first estimate of the flux spectrum for this weak source is already a real success. These spectra give first hints towards the conjecture that the spectrum of the AGN 1ES1959 is steeper than that of the Crab Nebula in the same energy range.

Concluding the study of the γ-hadron separation, we want to compare the neural networks with the random forests in a special way which also allows insights into the relation between Monte Carlo simulation and real data: Figure 7.46 shows the relation between the efficiency for simulated photons and the achieved significance on real data. For each classifier (neural network or random forest) the efficiency for simulated photons is measured (with the test set) for a fixed background rejection of 98%. The significance is the maximum significance which can be obtained for each classifier by varying the cut like in figure 7.40. The significances cannot be compared directly to the analysis above since the dataset used here had a different preprocessing (`size`>800 photons instead of 2000 above). The different classifiers have been obtained by training the two methods with several different parameter settings like in the study of the parameter optimisation process in section 7.2.7. Details about the variation of training parameters can be found in appendix B.3.

The two most important observations from figure 7.46 are

- that the performance spread (for both axes) is much larger for differently trained neural networks than for differently trained random forests and
- that the correlation between the performance on simulated photons (x-axis) and performance on real data (y-axis) shows a behaviour opposite to what is desired.

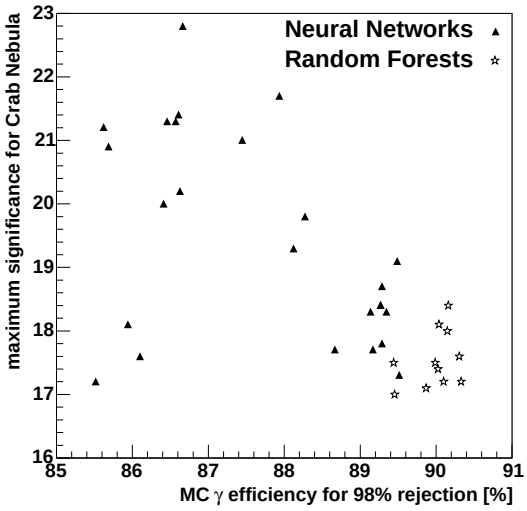

Figure 7.46: Monte Carlo efficiency vs. significance on real data for neural networks and random forests which have been trained with different parameter settings.

The explanation of the observed effects is, however, quite simple: There are some differences between the simulated and the real photons. These differences are small but can still be detected by the statistical learning methods. The random forest method always (nearly independent of changes in the training parameters) tends to adapt itself very well to the training dataset, i.e. to the simulated photons. Its performance on the real data is therefore always similar and always significantly below the optimum. The neural networks, in contrast, show a wide spectrum of performances on the simulated photons and thus on the real data. The important correlation is, as mentioned, that a better adaption to the simulated photons leads to a worse performance on real data. This is some kind of overtraining effect since the higher generalisation (observed with the significance on real data) has to be achieved by a lower performance on the training set (observed with the efficiency for simulated photons).

The conclusion regarding the comparison of neural networks and random forests is that as long as differences between simulation and real data exists (probably they will never disappear), neural networks clearly provide the better means to train classifiers which generalise better. Generalisation is meant here as "from the simulation to the real data". This generalisation can be achieved for neural networks by strong regularisation of the training process (compare appendix B.3), for random forests such an effect could not be stimulated.

Looking back to the descriptions of neural networks (section 5.4.2) and random forests (section 5.5.2) and to their typical behaviour on the toy examples in section 5.6 gives an intuitive explanation for the observation made here: The decision tree structure (even if averaged by bagging) creates a partitioning of the input space into potentially very small hypercubes, creating thus very local decisions which can follow any target function defined by the simulated photons. Neural networks, in contrast, are "more physical" in the sense that they create more globally oriented decision boundaries built up by joined hyperplanes. They will automatically ignore all small fluctuations of the target function which are artificially induced by the simulation.

7.6.2 Energy Estimation

The only possibility to train a method for energy estimation of very high energy photons observed in the MAGIC telescope – this also applies to the standard method which optimises parameters in a simple linear model – is to use events from a Monte Carlo simulation. Only there we have access to the information about the true energy of the photon. As mentioned in section 2.4, the estimation of the energy of the primary γ-ray photon is much more complicated than just a direct relation to the number of detected Cherenkov photons. The variables for the parametrisation as well as the inputs for the learning methods are: `size`, `conc`, `length`, `width`, `dist`, `width/length`, `size/(width*length)` and `leakage` (compare the definitions in section 2.4 and the descriptions in the last section). This choice is based on first studies of energy estimation performed in the MAGIC collaboration. The parametrisation is then a simple linear model [114]:

$$\begin{aligned} E_{est} = \quad & a + b \cdot \texttt{size} + c \cdot \texttt{conc} + d \cdot \texttt{length} + e \cdot \texttt{width} \\ & + f \cdot \texttt{dist} + g \cdot \tfrac{\texttt{width}}{\texttt{length}} + h \cdot \tfrac{\texttt{size}}{\texttt{width} \cdot \texttt{length}} + i \cdot \texttt{leakage} \end{aligned} \tag{7.8}$$

where the parameters a to i are determined by minimising the relative errors

$$\left(\frac{E_{est} - E_{true}}{E_{true}} \right)^2 . \tag{7.9}$$

It is expected that the quality of the energy estimation in dependence of the true energy will depend strongly on the energy spectrum which is used in the fitting or training step. For a very low analysis threshold, i.e. many photons with very low energy ($< 100GeV$) will be in the training sample, the energy resolution will be better for these events and worse for the higher energies. The opposite will be true for a spectrum which results from a high analysis threshold with less low energy photons. This difference is relevant not only for statistical learning methods but also for the simple parametrisation as illustrated in figure 7.47: Together with the original energy spectrum the reconstruction quality is shown there. The crosses show bias and variance of the energy estimation per bin in true energy, the additional histogram presents the total resolution which is obtained by summing up bias and variance in quadrature (compare section 3.12.2).

It can be seen that the energy resolution for the threshold `size`>2000 is much worse in the low energy region ($100GeV < E_{true} < 200GeV$) but slightly better for high energies ($> 500GeV$). This effect is mainly due to a simple shift of the global bias towards higher energies. We also see that the bias always plays an important role as low energies are systematically over-estimated and high energies are systematically under-estimated. This effect can be seen directly in the energy spectra reconstructed by the parametrisation, as shown in figure 7.48: Low and high energies are suppressed and medium energies are enhanced.

An alternative to the resolution plots shown in figure 7.47 is to change the true energy on the x-axis into the estimated energy. The two possibilities to present the energy resolution are shown in figure 7.49 next to each other (for the `size`>1000 spectrum). The left plot (in bins of the true energy) has the advantage that the behaviour of the reconstruction method is monitored in dependence of an objective value. The right plot (in bins of the estimated energy) is important from the calculational point of view, since an energy resolution has to be given for observed γ-ray photons for which then only the estimated energy is known.

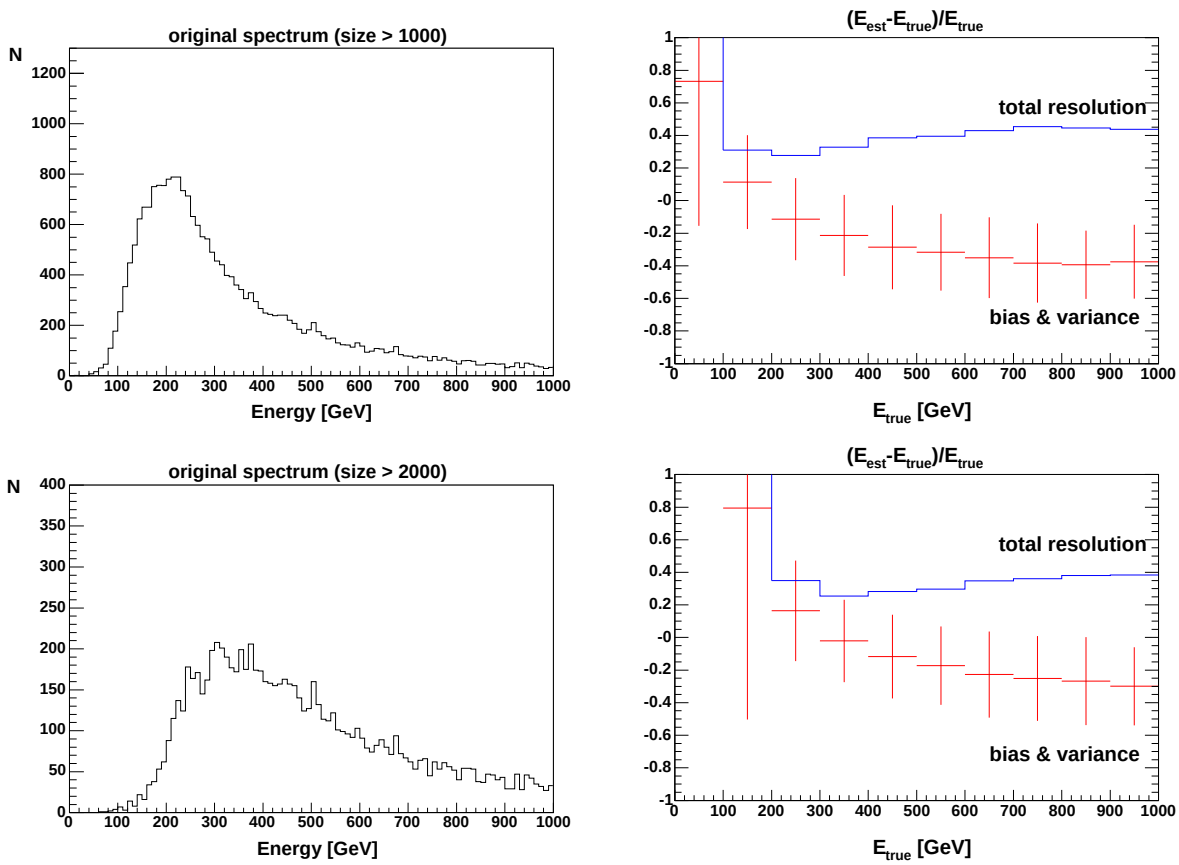

Figure 7.47: Differences in the behaviour of the energy estimation induced by different cuts in `size`: upper row `size`>1000, lower row `size`>2000, left the original spectra and right the behaviour of the parametrisation with bias, variance and total resolution in bins of the true energy.

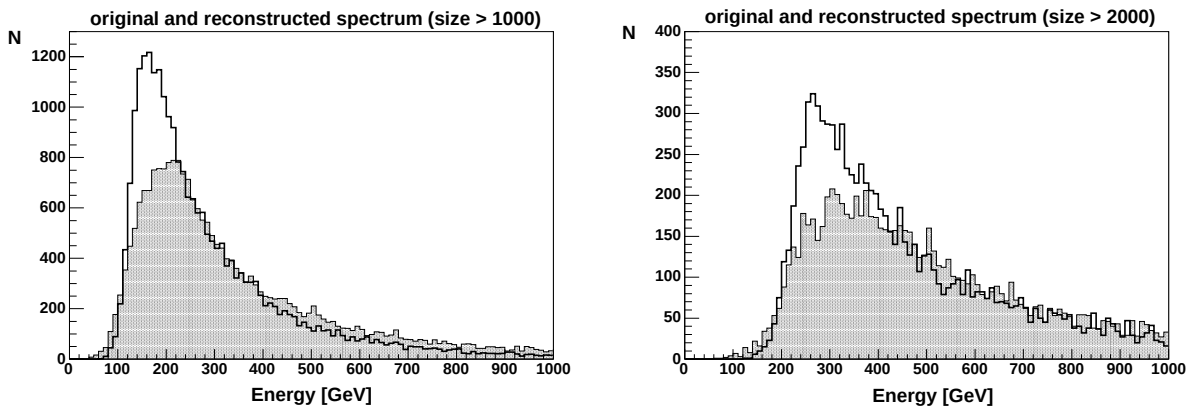

Figure 7.48: Original (shaded) and reconstructed (thick line) energy spectra for different size cuts, the method used to estimate the energies is the parametrisation.

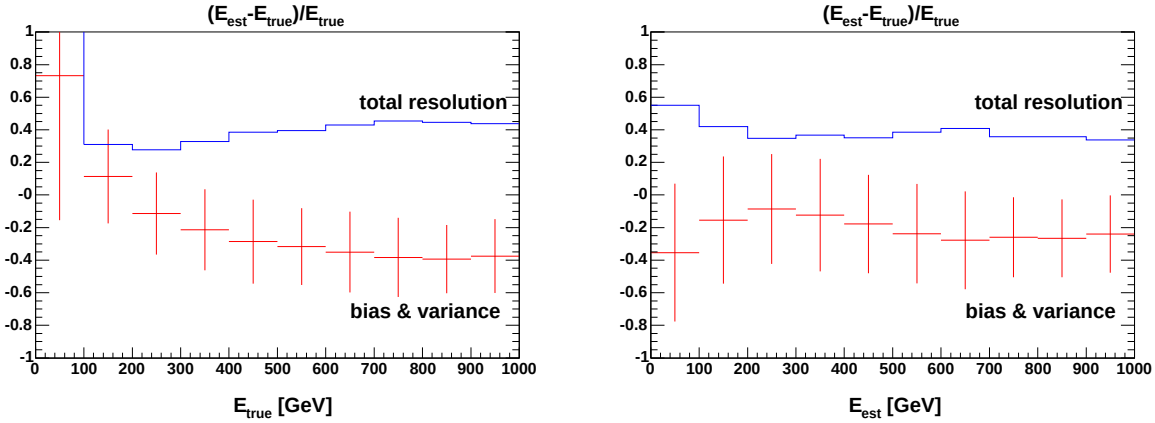

Figure 7.49: Binning in true energy vs. binning in estimated energy: Two different aspects of the energy resolution. The energy estimation is done by the parametrisation. The spectrum with `size`>1000 was used for the optimisation of the parametrisation and in the resolution plots. The left plot is identical to figure 7.47 top-right.

To keep this analysis tight, the parametrisation will only be compared to the random forest method (it has been verified that the neural network performs very similar to the random forest). Instead of a comparison of learning methods, a comparison of training targets will be performed in the following. The standard target for the energy estimation would of course be simply the energy itself. The training process of e.g. neural networks or random forests[13] implicitly optimises $\sum_i (E_{est} - E_{true})^2$ where i runs over all training events[14]. Since we want to optimise relative errors instead of absolute errors we choose $\log(E)$ as the training target for the regression.

A very different approach has emerged from the software development for the MAGIC telescope [112]: Instead of a normal regression a classification scheme was implemented there, called bin-wise classification or binning-method in the following. The energy estimation is done there by training different classifiers for several energy bins (with a logarithmic binning). Each classifier is constructed by training the contents of one energy bin against all others. Therefore each classifier should recognise a specific energy region. The evaluation is done by weighting the output of each classifier by the respective energy (centre of the bin):

$$E_{est} = \frac{\sum E_i \cdot out(i)}{\sum out(i)}. \tag{7.10}$$

The resolutions obtained by the two different training targets are shown in figure 7.50. Like above for the parametrisation, also here bias and variance are summed to get the total resolution.

Comparing the results from the two different training procedures we see that the resolution for the true regression is better for energies with a low true energy (the energy bin below $100GeV$ has too little statistics to allow any reasonable statement), for higher true energies the resolution of both methods is slightly above 20%. In dependence of the estimated energy we see a more stable behaviour for the logarithmic regression with res-

[13]The regression version of decision trees was discussed in section 5.2.

[14]This statement applies to the implementations discussed in chapter 5. Of course, different optimisation rules could be implemented.

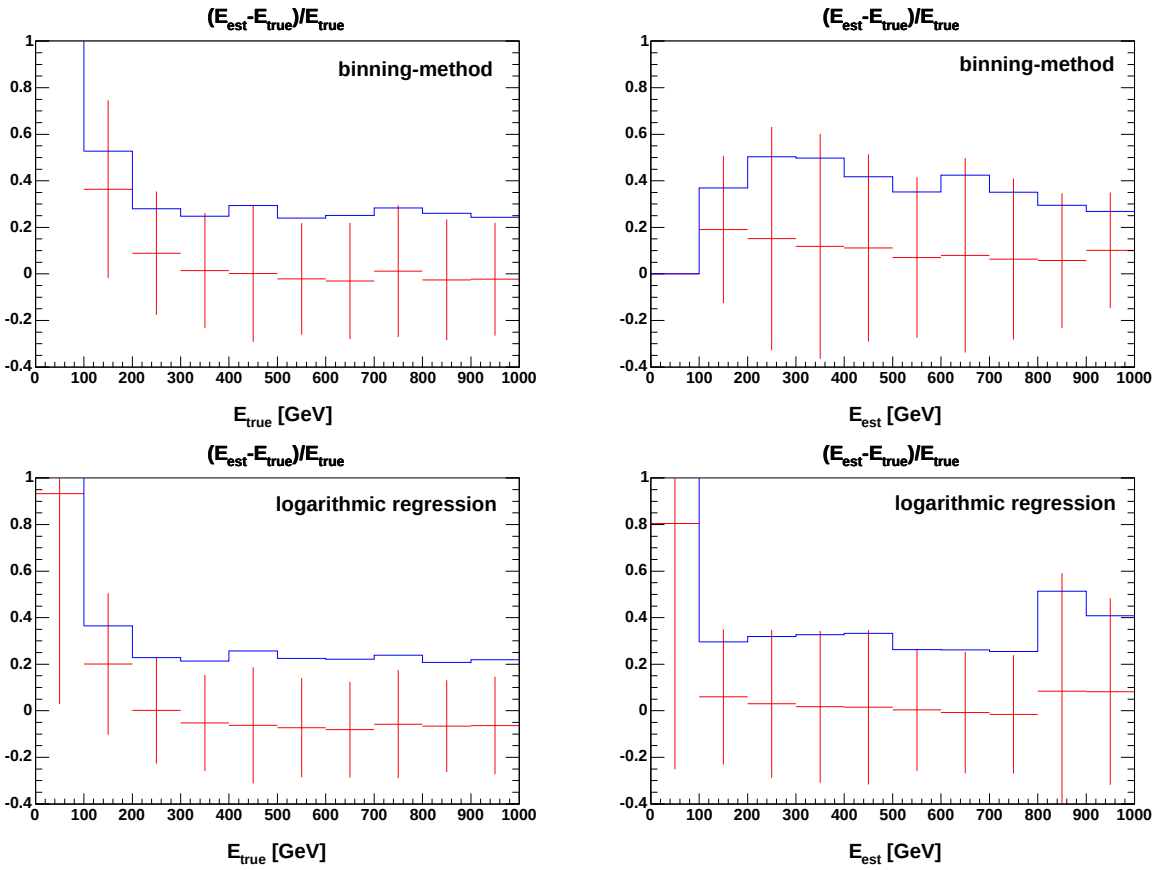

Figure 7.50: Resolutions obtained with the random forest method: The upper row shows the results for the bin-wise classification and the lower row for the logarithmic regression. Left the resolution is plotted in bins of the true energy and right in bins of the estimated energy. The spectrum with `size`>1000 was used in the training of the random forest and in the resolution plots.

olutions mostly around 30% while the classification technique shows resolutions varying between 30% and 50%. These results suggest that a true regression is preferable over the bin-wise classification.

A second sign for a problematic behaviour of the classification technique is found by comparing the spectra reconstructed by the respective methods. Figure 7.51 shows the original spectrum as well as the reconstructed spectra of the binning method (bin-wise classification), the logarithmic regression with training target $\log(E_{true})$ and the linear regression with training target E_{true}.

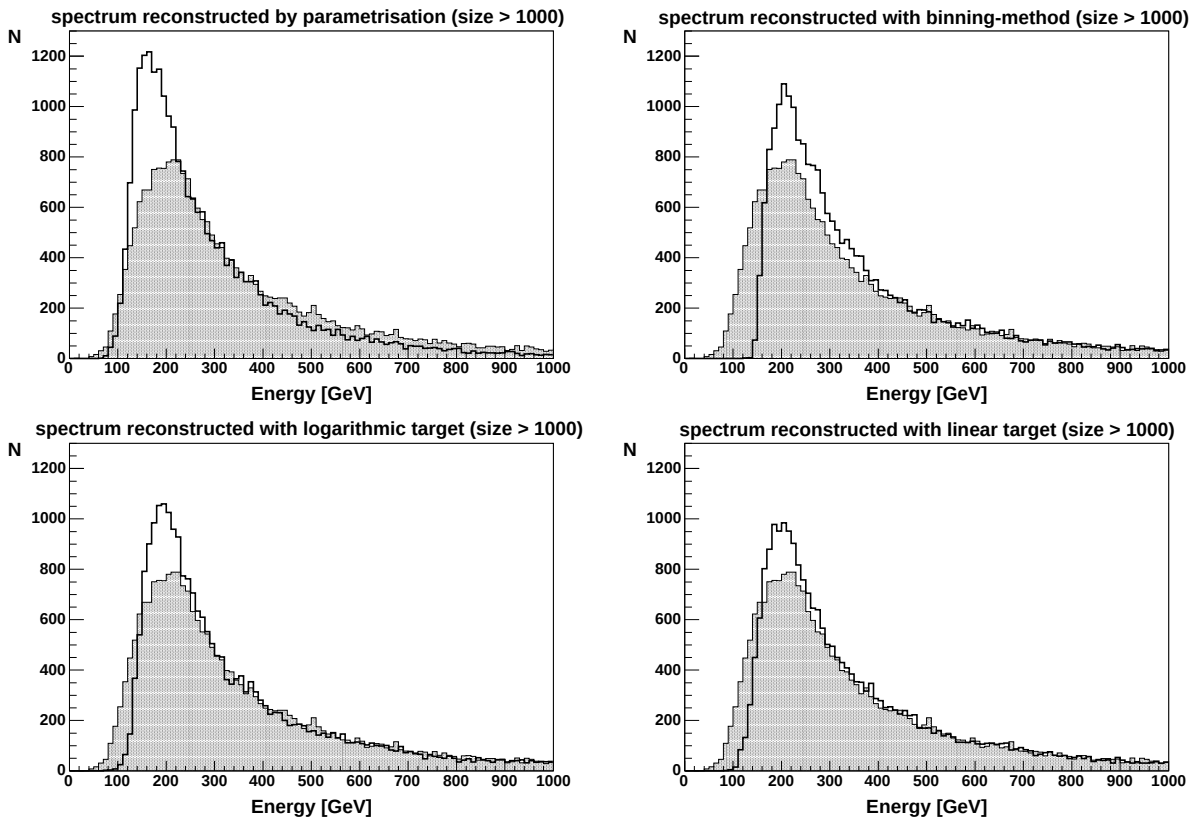

Figure 7.51: Reconstructed spectra obtained with different training methods, the shaded original spectrum is always plotted to allow the comparison. Shown are the spectrum reconstructed by the parametrisation top-left, the spectrum reconstructed by the bin-wise classification top-right and in the lower row the two regression methods, with a logarithmic target left and with a linear target right.

We see that the true energy spectrum reaches down to below $50GeV$. Both regression techniques reconstruct energies down to about $100GeV$ but the bin-wise classification reaches only $150GeV$. This points to an upward bias for very low energy events which is stronger for the bin-wise classification than for the regression techniques (as already seen in the resolution plots in figure 7.50). This bias is most probably introduced by the weighting procedure shown in equation 7.10. Regarding the goal of the MAGIC analysis, i.e. to reach down to 50 or even 30 GeV, the bias in the reconstructed energy plays a very important role.

The energy resolution plots shown so far are complemented by the correlation plots shown in figure 7.52. Each E_{true} bin (i.e. each column) has been normalised to 1 there

so that the spectrum appears to be flat in E_{true}. This makes it easier to evaluate the high energy part of the spectrum where events become rare. These plots complement the ones shown in figure 7.51 because here the absolute reconstruction errors are visualised in contrast to the relative errors in the plots before. One can see that the parametrisation (a) performs badly for higher energies since it is optimised for the lower energies where the majority of the events is found. The problems of the bin-wise classification for very low energies can be seen in plot (b) while the logarithmic regression (c) performs better in this energy region.

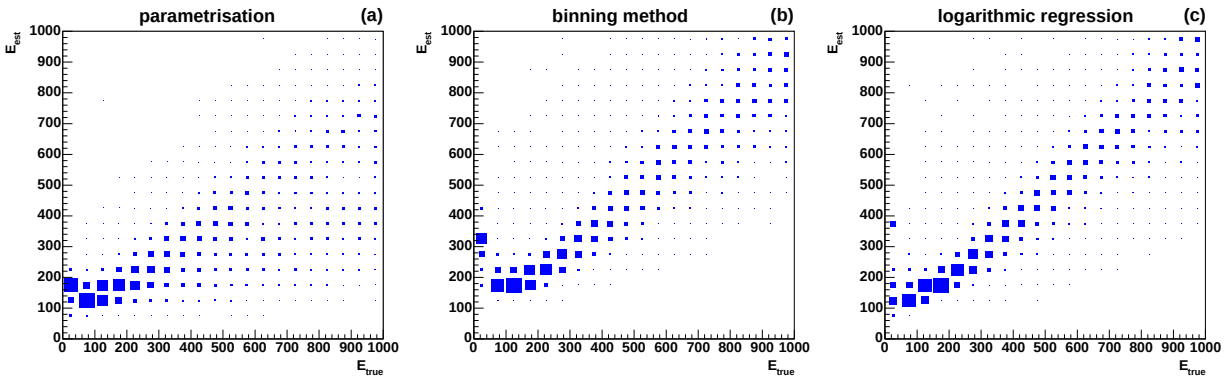

Figure 7.52: Correlation between true and estimated energy for the `size`>1000 spectrum. Each E_{true} bin (i.e. each column) has been normalised to 1 so that the spectrum appears flat in E_{true}. In (a) the parametrisation is used for the energy estimation, in (b) the binning-method (bin-wise classification) with random forests and in (c) the logarithmic regression with a random forest are used.

Finally, the resolutions of the random forest method (using the logarithmic target) and the parametrisation have to be compared. Regarding the reconstructed energy spectra it has to be noted that the parametrisation reconstructs energies down to almost $50GeV$ while the random forest reaches only $100GeV$ (figure 7.51). The resolution in dependence of the estimated energy looks very stable for the parametrisation as it stays about 40% for the whole energy spectrum (figure 7.49). The random forest method (logarithmic regression) shows a slightly noisier behaviour with resolutions around 30% (figure 7.50). In dependence of the true energy, however, the parametrisation shows resolutions between 30% and 45% (even rising towards higher energies due to the significant bias) while the random forest shows an almost constant resolution of 20% to 25%.

In summary, the energy estimation with statistical learning methods (shown here with the random forest method) gives better results than the classical parametrisation with resolutions improved by about 10% to 15% (absolutely). The comparison of the bin-wise classification and the logarithmic regression, both with random forests, has shown that the "true" regression is preferable due to the better behaviour in the very low energy region and also due to the improved resolutions.

7.6.3 Future Research

It was mentioned already in section 2.4 that for very low energy γ's the reconstructed Hillas ellipse may not be suited to describe the event since the number of collected Cherenkov photons is low and only very few pixels are illuminated. The reconstruction of the Hillas parameters, i.e. the geometry of the Cherenkov ellipse, is very inaccurate in this case. As an alternative the direct use of each pixel value together with the timing information is suggested.

The reconstruction of the shower image with the Hillas parameters is mandatory for the standard approaches: supercuts for the γ-hadron separation and the parametrisation for the energy estimation. The statistical learning methods could, however, also use inputs like the number of Cherenkov photons and their arrival time for all pixels. It is not wise to simply use all pixels as inputs for the learning method, a clever preprocessing is usually more advantageous. One could, for example, extract a circular region of pixels around the centre of gravity of the Cherenkov shower.

The following study shows that a change in the inputs from Hillas-based quantities towards pixel-based quantities reveals problems in the Monte Carlo simulation of γ-events which have to be studied and corrected before this new approach can lead to improvements. We study the γ-hadron separation in the low energy region by applying a cut in `size` of 200 to 400 photoelectrons which corresponds to roughly 1000 to 2000 Cherenkov photons. The γ-hadron separation for these events is then trained by using simulated photons as signal and OFF-data as background. The Hillas-approach is compared to the pixel-approach by using two different input sets: On the one hand a training is done like in section 7.6.1 with the Hillas-quantities. On the other hand, a second training is done with the pixel signals and timings which are extracted in a circular structure around the centre of gravity of the shower as mentioned above.

The trained neural networks are compared in two ways (exactly like at the end of section 7.6.1): The test set (consisting of simulated γ's and OFF-data) can be used to get the estimated efficiency for simulated γ's for a reasonable background suppression (99% taken in the following). The networks can also be applied to the Crab Nebula ON and OFF datasets and the α-plot for a reasonable background suppression (similar to the 99% taken above) should lead to the well-known photon excess for low α's.

The results obtained show a clear disagreement between the two evaluation methods: Whereas the efficiency for simulated γ's (for 99% rejection, measured with the test set) is only 17% for the Hillas-approach, it is 38% for the pixel-approach. One expects a photon excess twice as high in the α-plots for the latter approach. But as figure 7.53 shows, only for the Hillas-approach a photon excess is visible, there is no signal at all for the pixel-approach.

The lesson to learn is that the adaption of the Monte Carlo simulation to the data is quite good with respect to the higher abstraction level: The Hillas quantities are described well, with the exception mentioned at the end of section 7.6.1. But there remains a lot of work to be done for the lower abstraction level concerning the individual pixel signals and timings. With a better description of the data the pixel-approach is probably better suited for a γ-hadron separation and for energy estimation in the low energy region than the Hillas-approach.

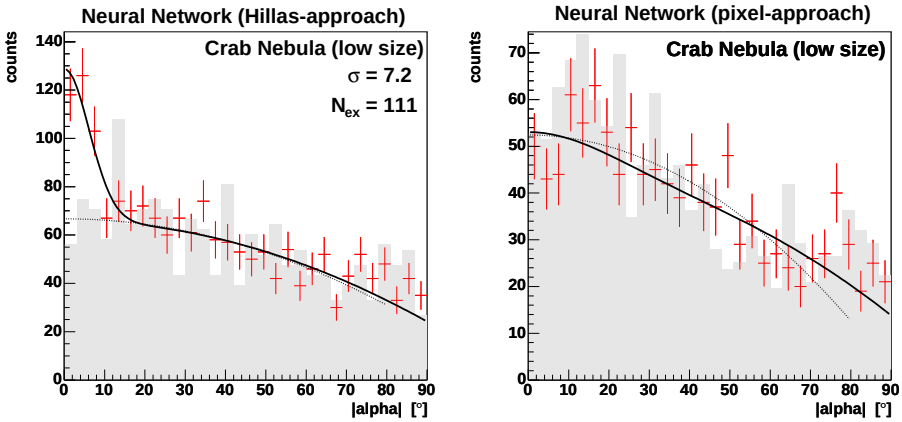

Figure 7.53: Two different input sets for the low size region: For `size` between 200 and 400 photoelectrons the Hillas-approach still shows a small signal while the pixel-approach shows no signal. In contrast to these α-plots the γ-efficiency is much higher for the pixel-approach when measured with the Monte Carlo simulation.

7.7 Applications for the XEUS satellite

The next sections will present results for pileup rejection and sub-pixel resolution with statistical learning methods (see section 2.5 for an introduction into the subject). The main data-source for this analysis is a Monte Carlo simulation, for the sub-pixel resolution also the mesh experiment was used to obtain experimental data for training and testing.

As already discussed in section 2.5 the preprocessing of the data coming from a pixel-detector is a very important step. The newly implemented algorithm for the calibration and signal extraction for the XEUS satellite will be presented in appendix B.1. A second important algorithm needed for the analysis presented here is the determination of the mesh parameters in the mesh experiment. The position and angle of the mesh relative to the CCD lead to the position of each hole above the detector. The newly developed algorithm will be presented in appendix B.2.

7.7.1 Pileup Rejection

The first application of statistical learning methods for the XEUS satellite is a classification problem. As mentioned in section 2.5 high photon rates will lead to patterns which contain two or more photons. The spectroscopic information gets lost completely in these *pileup* events, therefore they should be filtered out. In addition pileup events become the dominant event-type for very large photon fluxes, depending of course on the integration time of the detector. A rejection of pileup events already done online aboard the satellite may be a very important step to cope with the data rate which may be too large to be sent down to earth for the observation of bright X-ray sources.

Figure 7.54 shows which charge distributions may occur if two photons form a pileup. All patterns larger than 2×2 pixels (a,b) can be filtered out easily while complete charge pileup in only one pixel (e) cannot be distinguished from one single photon. Pileups which form case (c) or (d) or similar patterns could be distinguished from single photon events by a sophisticated analysis.

A simple pattern oriented analysis [115] uses the four pattern types shown in figure 7.55. Only the relative position of maximum and minimum charge are used in this method. This pattern method is used, for example, for the analysis of the XMM data [115] and will be used here as a basic classical algorithm to which any statistical learning method can be compared.

Data Source and Preprocessing

This analysis is based on data generated by a Monte Carlo simulation developed at the MPI semiconductor laboratory [116]. The basic detector parameters used in the simulation are listed in table 7.25.

depth	temperature	noise	voltage	channel depth
300 μm	200 K	6 electrons	180 V	290 μm

Table 7.25: Basic detector parameters used in the simulation.

Since it is no problem to detect pileup in a scenario with only one energy line (pileups then have double or triple energy) we choose a white spectrum from $200eV$ to $20keV$. The

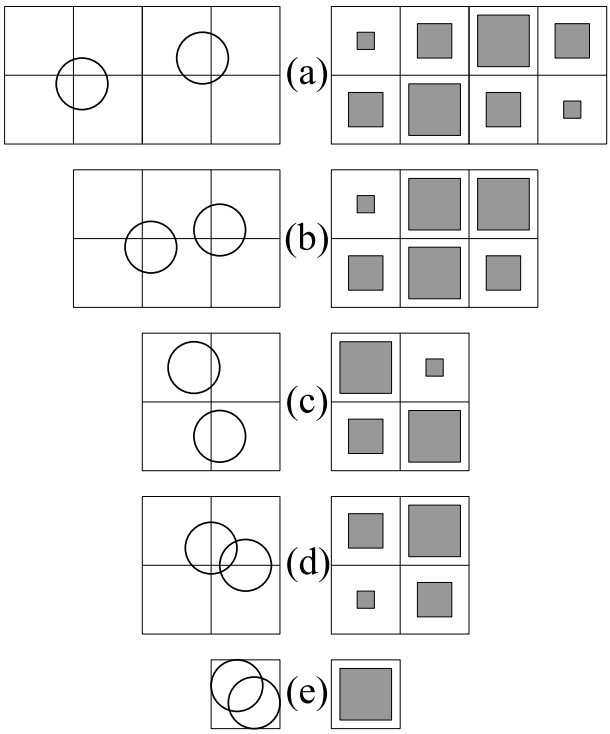

Figure 7.54: Pileups from pure pattern pileup (a) and (c) over mixed pattern and charge pileup (b) and (d) to pure charge pileup (e).

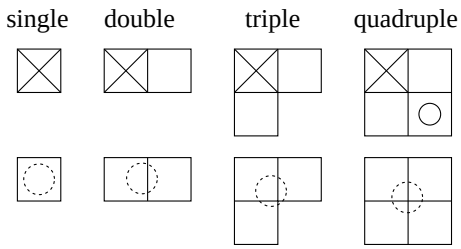

Figure 7.55: Allowed patterns for the "XMM" algorithm: Shown are the illuminated pixels (above some threshold). In the upper row the cross marks the maximum charge and the circle the minimum charge. The lower row shows how the patterns can be generated by a charge cloud.

resulting reconstructed spectrum is shown in figure 7.56 where the energy is measured in ADC counts. $20keV$ correspond to 5405 ADC counts, the conversion factor of $3.7eV$ is the average energy needed to create an electron-hole pair in silicon. The use of a flat spectrum reflects the natural necessity to detect pileups which are generated by any combination of photon energies.

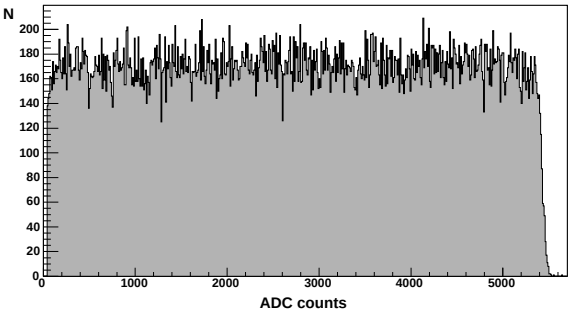

Figure 7.56: The reconstructed white spectrum used in the datasets, $20keV$ correspond to 5405 ADC counts, the conversion factor of $3.7eV$ is the average energy needed to create an electron-hole pair in silicon.

The splitting behaviour strongly depends on the pixel size. We will therefore test simulations with three pixel sizes of $50 \times 50\mu m^2$, $75 \times 75\mu m^2$ and $150 \times 150\mu m^2$. While "singles" dominate the events seen for the largest pixel size they form only a minor contribution for the smallest pixel size.

The events (generated by one or more photons) formed by the clustering procedure of the preprocessing are first filtered: Only events which fit into 2×2 pixels are written out, others are identified as pileups (patterns (a) and (b) in figure 7.54). In addition, a cut at 40 ADC counts total energy should suppress all noise events. The remaining events are divided to form the training and test events for the neural networks as well as for the XMM pattern method.

There is no big choice which inputs should be used. For the XMM algorithm the pixels above threshold are taken and compared to the patterns shown in figure 7.55. For the neural network always four signal values are given as input even if only one to three pixels are above threshold. This is done as shown in figure 7.57: The direct neighbours of the maximum charge are checked for their signals and the 2×2 box which will be extracted is determined by the larger signal in x-direction and in y-direction. Giving always these four values as inputs, the neural network is able to determine by itself which charge value is noise and which not.

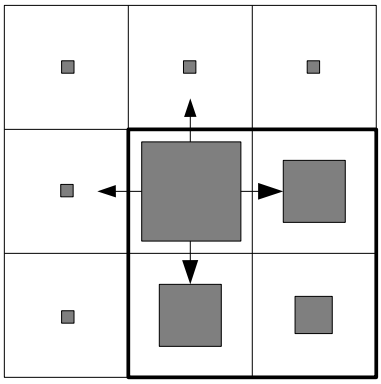

Figure 7.57: Inputs for the neural network: The 2×2 box which will be extracted is determined by the larger signal in x-direction and in y-direction.

The symmetry naturally given by the two coordinate axes is resolved by mirroring the 2×2 box until the maximum charge lies in a prefixed corner. In addition the second

maximum is mirrored to a prefixed position so that in effect the four values get sorted. Two normalisation schemes have been tested. The first (figure 7.58 a) applies the same normalisation factor on each of the four pixel signals to bring them into the range $(0, 1)$. The second one (figure 7.58 b) normalises the three smaller pixel signals with the value of the maximum charge and uses a fixed normalisation on the pixel with the maximum charge. This second method performed better in the tests and will be used further on.

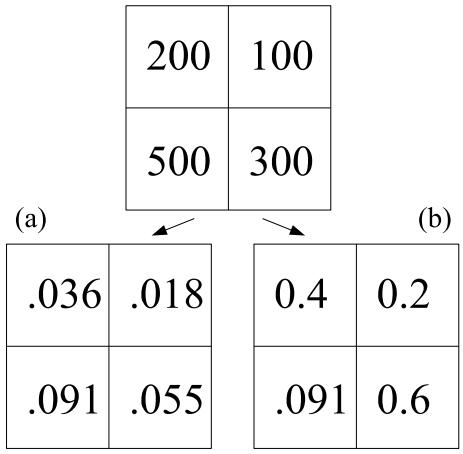

Figure 7.58: Normalisation (a) divides all four values by the same factor (5500), normalisation (b) divides the maximum by 5500 and the remaining three values by the maximum.

The XMM Algorithm

We will start this analysis by studying the typical behaviour of the standard pileup rejection, the "XMM" algorithm. The algorithm naturally has a very high efficiency (for single photons) but may also have only a low background (pileup) rejection. Only one parameter steers the behaviour of the XMM algorithm: the threshold which is used in the preprocessing step affects the pattern type by defining the acceptance of small signals. The lower the threshold the better small signal fractions are recognised but the more also noise is mistaken as signal. Thus the threshold setting leads to a specific efficiency and rejection of the XMM algorithm depending on the dataset. In the following the well-performing standard of a 5σ threshold will be used.

Typical events which originate from single photons but which are nevertheless rejected by the XMM algorithm are shown in figure 7.59. The usual problem is that a noise contribution changes the pattern type: The quadruple type of figure 7.55 is transformed by additional noise into a pattern where the minimum is no longer opposite to the maximum.

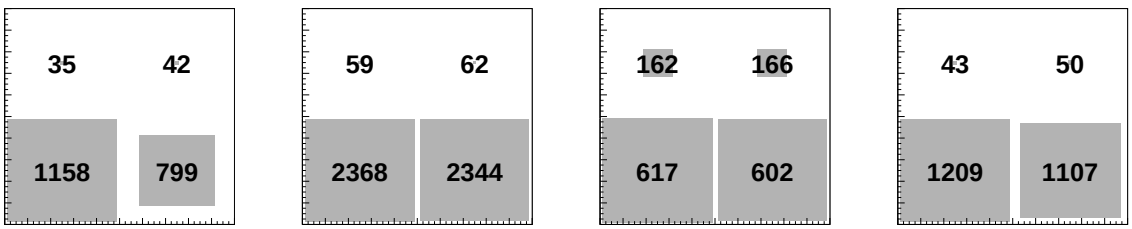

Figure 7.59: Patterns from single photons which are not recognised by the XMM algorithm: The quadruple (4-split) pattern type is modified by noise so that the minimum is no longer opposite to the maximum, as required by the algorithm for genuine photons (figure 7.55).

Concerning the rejection of pileups, the XMM algorithm is rather conservative. Figure 7.60 shows some examples of pileups which are not recognised by the XMM algorithm. They can be identified as pileups if one takes into account that they cannot be generated by the Gaussian charge cloud of one single photon with the given detector parameters.

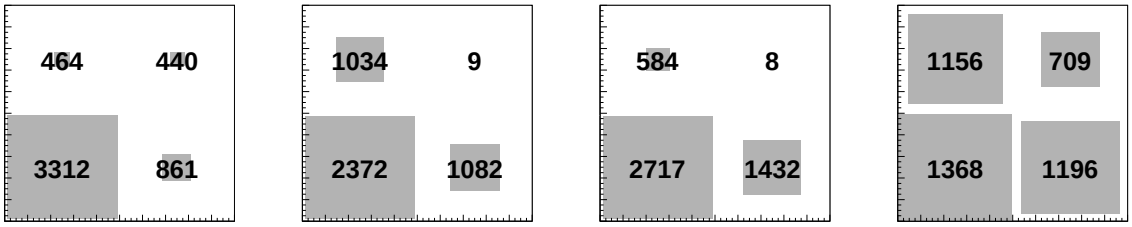

Figure 7.60: Patterns from pileups which are not recognised by the XMM algorithm: None of these events can originate from a single photon in this specific dataset because in the first event the minimum is too large compared to the other three charges, in the next three events the minimum is always too small compared to the other charges.

Comparison of the XMM Algorithm to Neural Networks

In this section the efficiency for single photons and the rejection rate for pileups will be analysed as function of the total energy of an event. The total energy should not have any influence on the behaviour of the XMM algorithm since it uses only topological information, but a neural network may show an energy dependence.

The two pileup rejection methods are compared by first determining the efficiency of the XMM algorithm and then setting the cut for the neural network to obtain the same efficiency. These efficiencies are 99.9% for 150 μm pixel size, 99.5% for 75 μm pixel size and 98.9% for 50 μm pixel size. The efficiency of the XMM algorithm decreases towards smaller pixel sizes because the fraction of quadruples increases. The quadruples may be transformed by noise as in the examples in figure 7.59. The higher the fraction of quadruples the higher thus the inefficiency.

Figure 7.61 summarises the spectral dependence of efficiency and rejection for the three different pixel sizes. The identical mean efficiencies close to 100% are shown as well as the rejection rates which differ most for small pixel sizes. For smaller pixel sizes the neural network clearly performs better. As shown, it rejects more pileups at the same efficiency, for example, all events shown in figure 7.60 are correctly recognised as pileups by the neural network. For a different cut it shows a pileup rejection similar to the XMM algorithm but a much higher efficiency with which, for example, all events shown in figure 7.59 are correctly recognised as single photons.

A clear conclusion can be drawn from these plots: Whereas it does obviously not matter which method is used to reject pileups for a pixel-detector with pixels as large as $150 \times 150 \mu m^2$ it does matter for small pixel sizes like $50 \times 50 \mu m^2$. A factor two larger pileup rejection for 75 μm pixel size and a factor three larger pileup rejection for 50 μm pixel size as visible in figure 7.61 is a significant improvement. Taking into account the trend to smaller pixel sizes (giving better angular resolutions) one has to conclude that future missions will profit from the advanced analysis with statistical learning methods. Seen from a different viewpoint one has to keep in mind that the statistical learning method

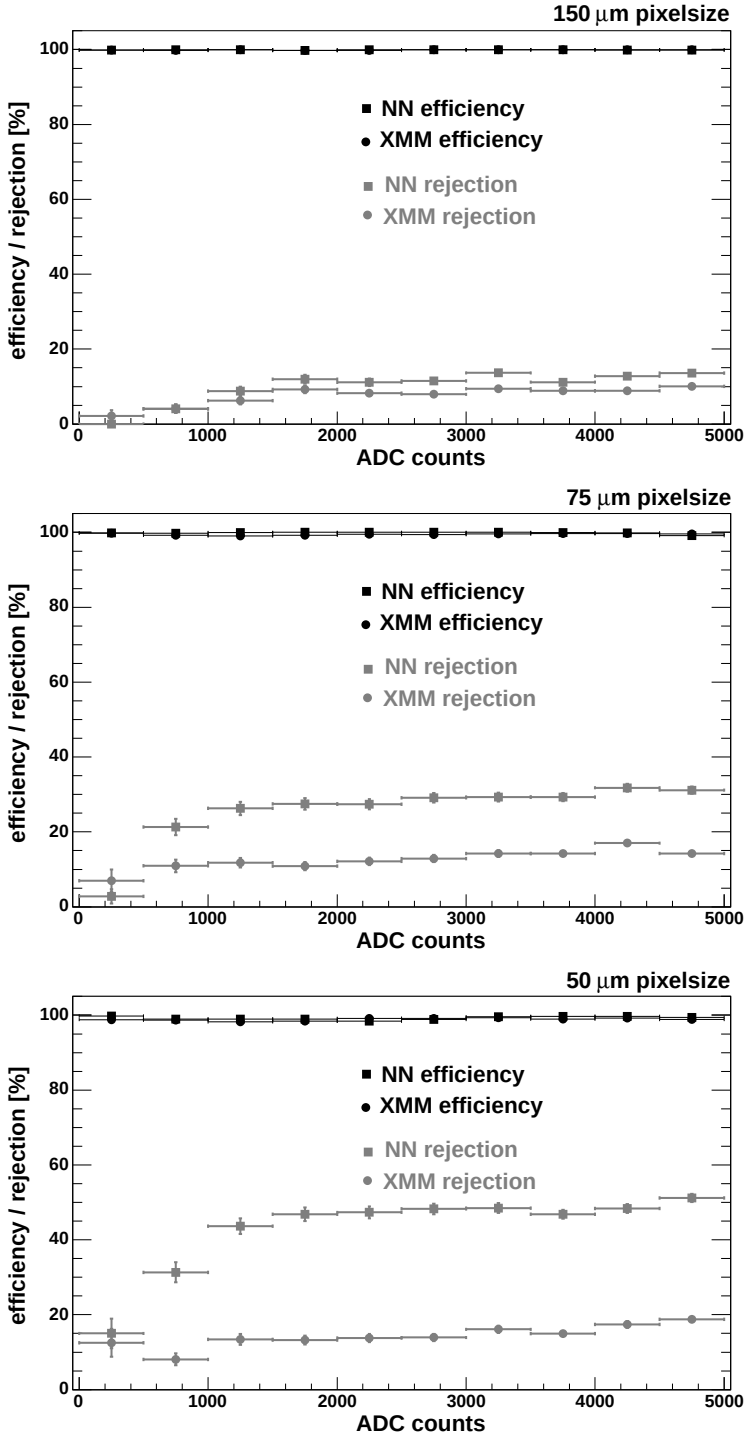

Figure 7.61: Pileup rejection of XMM algorithm and neural network: For three different pixel sizes the efficiencies are designed to match. While no significant difference for the rejection rate is visible for 150 μm pixel size, the neural network gives a factor 3 better rejection at the same efficiency for 50 μm pixel size.

of course allows to trade efficiency vs. rejection which means that one could also decide to choose an efficiency higher than 98.9% for 50 μm pixel size and accepting some more pileup in return. For example, an efficiency of 99.9% can be chosen (to have the same photon efficiency as for the 150 μm pixel size), the pileup rejection is then about 20%: still slightly higher than the XMM algorithm and the single photon loss is reduced from 1.1% to 0.1%, by a factor 11.

Systematic Uncertainties

The pileup rejection task is a good candidate to study systematic effects which may influence the efficiency and rejection. Four different systematic effects will be studied as examples:

- A filter is placed in front of the X-ray detecting device to filter out non X-ray light. Starlight from the optical part of the spectrum which passes the filter may generate a few electrons inside the pixel detector. With a sufficiently high flux, a significant energy can be deposited in the detector. The amount depends on the observation target and on the filter type used. The systematic variation of the starlight background can be simulated by a correlated additional local offset in the pixel-detector.

- The electronic noise might change over time since it depends on the operation mode, temperature etc. Its variation can be simulated by random noise added to each pixel.

- Crosstalk between neighbouring pixels is an example for an effect generated in the readout electronics. It may happen in the amplifier when the amplification of a pixel depends on the previous state of the amplifier i.e. on the strength of the previous signal. A large signal generates by this a small signal in the neighbouring (right) pixel in the order of a percent of the original signal.

- A second systematic effect which may happen due to an unstable amplification of the signals is a variation in the linearity: Higher order effects may lead to a smaller amplification of large signals.

To study the stability of the efficiency and rejection of the two methods under varying conditions, three different levels of uncertainty are created, called "low", "medium" and "high". Starlight is simulated by a two-dimensional Gaussian distribution with a maximum drawn from a flat distribution between 0 and 15, 30 or 45 eV respectively. Electronic noise is drawn from a Gaussian distribution with a $\sigma = 6$, 12 or 18 eV, respectively. Crosstalk is simulated by adding 1%, 2% or 3% of a pixel's signal to its right neighbour and the non-linear amplification is simulated by adding a quadratic term with a coefficient of $-1 \cdot 10^{-5}$, $-2 \cdot 10^{-5}$ or $-4 \cdot 10^{-5}$ (for signals in ADC counts).

Table 7.26 summarises the influence of the systematic uncertainties on efficiency and rejection for photons below 5000 ADC counts[15] for the XMM algorithm for 50 μm pixel size. For the neural network table 7.27 presents the detailed list of systematic variations and their propagation to efficiency and rejection and table 7.28 summarises these uncertainties.

The significantly largest contribution to the total systematic uncertainty is always due to the crosstalk effect as seen in table 7.27. This matches the intuition that the charge ratios

[15] As mentioned above this restriction is necessary to cut away the edge effect since single photons are only simulated up to 20 keV, corresponding to roughly 5400 ADC counts.

Efficiency	98.88	±0.07 (stat.)	− 0.08 + 0.00 (syst.,low) − 0.20 + 0.01 (syst.,medium) − 0.21 + 0.03 (syst.,high)
Rejection	15.64	±0.35 (stat.)	− 0.32 + 0.10 (syst.,low) − 0.80 + 0.15 (syst.,medium) − 1.11 + 0.32 (syst.,high)

Table 7.26: Total systematic uncertainties for the pileup rejection (XMM algorithm) for 50 μm pixel size.

"low"		
variation	eff. [%]	rej. [%]
without	98.9	49.0
starlight	98.9	48.9
noise	98.9	48.7
crosstalk	98.9	48.5
amplification	98.9	49.1
"medium"		
variation	eff. [%]	rej. [%]
without	98.9	49.0
starlight	98.8	48.7
noise	98.9	48.6
crosstalk	98.9	47.6
amplification	98.9	49.1
"high"		
variation	eff. [%]	rej. [%]
without	98.9	49.0
starlight	98.8	48.6
noise	98.9	48.5
crosstalk	99.0	46.7
amplification	98.9	48.8

Table 7.27: Detailed systematic uncertainties for the pileup rejection (neural network, 50 μm pixel size).

Efficiency	98.88	±0.07 (stat.)	− 0.01 + 0.06 (syst.,low) − 0.05 + 0.07 (syst.,medium) − 0.12 + 0.13 (syst.,high)
Rejection	49.00	±0.48 (stat.)	− 0.55 + 0.11 (syst.,low) − 1.47 + 0.08 (syst.,medium) − 2.41 + 0.00 (syst.,high)

Table 7.28: Total systematic uncertainties for the pileup rejection (neural network) for 50 μm pixel size.

of neighbouring pixels are very important for the discrimination process. Both methods tend to loose a few percent of their background rejection (relatively, the XMM algorithm loses more) while the efficiency stays in the order of 99%. Both methods therefore show a rather stable behaviour which can cope with even very large systematic effects.

7.7.2 Sub-pixel Resolution

The second application of statistical learning methods for a pixel detector aims at improving the positional resolution. A natural basis for this resolution is given by the pixel size of the detector. Taking into account the charge splitting among neighbouring pixels can improve the resolution significantly. Figure 7.62 shows how the relative incident position is defined within one pixel.

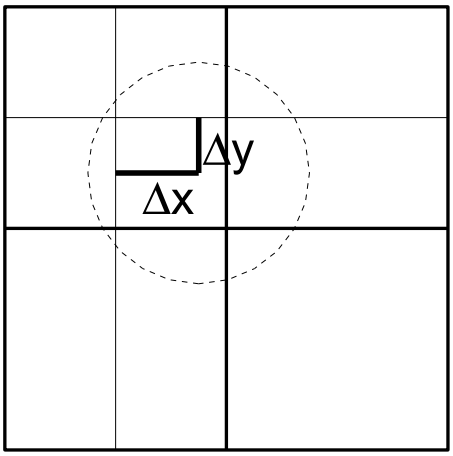

Figure 7.62: Definition of the relative incident position: The distances Δx and Δy are measured from the centre of the pixel with the maximum charge to the centre of the charge cloud. The maximum value for Δx and Δy is half of the pixel size. The normalisation divides by half of the pixel size and returns thus distances between 0 and 1.

Data Source and Preprocessing

As for the pileup rejection, the Monte Carlo simulation developed at the MPI semiconductor laboratory [116] is used to generate data for training and evaluation. Again three different pixel sizes from 50 to 150 μm will be tested for their influence on the position reconstruction.

To exclude noise and pileups from this analysis the events coming from the clustering procedure are filtered by the standard XMM filter described above. A cut of 40 ADC counts minimum total energy has been applied. As for the pileup rejection, the 2×2 box is extracted for each event and the maximum is mirrored into a fixed position. Care must be taken that the x and y coordinates have to be adopted accordingly. The position of the second maximum cannot be fixed here since this would lead to a measurement of the resolution for the direction with the larger splitting. This would be a significant bias compared to the true resolution in x- or y-direction.

For the following analysis, using simulated data, the detector is assumed to behave identically in the x and y coordinates. Therefore the above preprocessing makes sense and all results for the x direction will apply equally for the y direction.

The second data-source for the study of sub-pixel resolution is the mesh experiment (see section 2.5.1). A measurement with a gold-mesh in front of a 75 μm pixel size detector will be used as training data for a learning method which can then be optimised to the experimental situation.

Standard Reconstruction Methods

A straightforward method to obtain the centre of a distribution is the "centre of mass" method (COM)[16] which weights each pixel position with its charge:

$$x_{COM} = \frac{\sum x_i c_i}{\sum c_i}. \tag{7.11}$$

Since the charge cloud has a Gaussian shape this estimate does not perform very well because it assumes a linear splitting of the charges. A correction table can help to map the COM values to the correct position. This table needs training data in the sense that a mean correction value for each COM value (in a fine binning) has to be determined by taking into account all events from the training set. The position resulting from the combination of COM value and the correction table will be called "corrected centre of mass" method (CCOM). In summary, both COM and CCOM use the ratios $\frac{c_{right}}{c_{total}}$ for x and $\frac{c_{down}}{c_{total}}$ for y. Figure 7.63 (a) shows the translation table from the charge ratio to the position derived with the CCOM method.

Another reconstruction method with sub-pixel resolution is called "η-method" [117]. This method also uses the two charge ratios mentioned above and derives directly the corresponding positions x and y by inverting the function

$$f(x) = \frac{c_{right}}{c_{total}} \tag{7.12}$$

so that

$$f^{-1}\left(\frac{c_{right}}{c_{total}}\right) = x \tag{7.13}$$

can be calculated. The inversion is done numerically by sampling the charge ratios of equally distributed values of x. A histogram (called η-function) for the charge ratios is created and filled with events which have a flat distribution in x. The running integral of this histogram returns the inverse function f^{-1} because

$$\begin{aligned} (f^{-1})'\left(\frac{c_{right}}{c_{total}}\right) &= \frac{1}{f'(x)} \\ f^{-1}\left(\frac{c_{right}}{c_{total}}\right) &= \int \frac{dx}{df} \end{aligned} \tag{7.14}$$

where the last term is the integral over the histogram. Figure 7.63 (b) shows the translation table from the charge ratio to the position derived with the η-method. One can see that the two methods behave very similarly.

The translation tables are built with all events from the mentioned white spectrum. To give the methods the opportunity to adapt themselves to energy dependent effects six such tables will be used in the analysis below for six energy regions up to 6000 ADC counts. However, the differences in the tables turn out to be negligible.

[16] more appropriately called "centre of gravity"

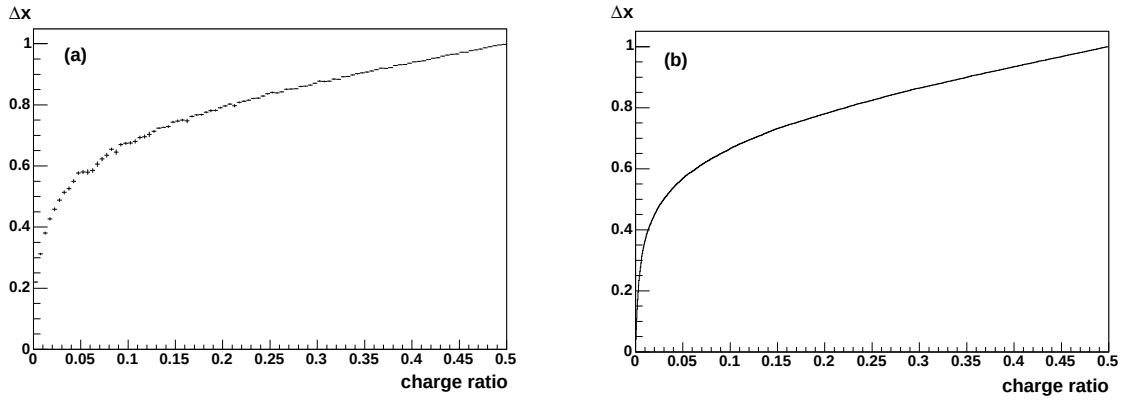

Figure 7.63: CCOM and η-method: Final functional dependence between charge ratio $\frac{c_{right}}{c_{total}}$ and position as derived by the CCOM method (a) and the η-method (b).

Results

To make the following results more transparent we will start with some basic investigations regarding the splitting behaviour as function of the photon energy. Figure 7.64 shows the typical geometry of the conversion point and the following drift of the electron cloud for low energy and high energy photons. Photons with a lower energy convert close to the backside, have a long drift time and therefore create, by diffusion and electrostatic repulsion, a large transverse extension of the charge cloud reaching the readout structure. For higher energies the absorption length is larger and the conversion point moves towards the front side of the semiconductor device which leaves less drift time and creates smaller extensions in the transverse direction.

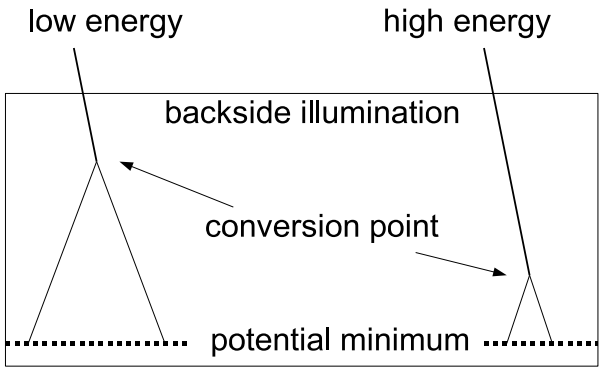

Figure 7.64: The photon energy determines the final charge cloud size: The higher the energy the larger is the absorption length and the smaller is the drift time from the conversion point to the potential minimum. The drift time finally determines the transverse extension of the charge cloud by diffusion and repulsion.

This effect directly translates into the probability to create a split event. Figure 7.65 shows the probabilities for different splitting types depending on the energy of the photon for 50 μm pixel size. We see the clear trend that the probability of the largest splitting (quadruple – 4-split) decreases from medium to high energies while the probability for smaller splittings – especially also for singles – rises due to the discussed absorption length effect. The theoretical extrapolation of this behaviour is shown in dotted lines. Towards lower energies, however, a different effect becomes relevant.

The decreasing probability for large splitting towards very low energy is only a reconstruction artefact since the number of illuminated pixels is determined by applying a threshold to each pixel. Even in a very low noise scenario some threshold must be set to

distinguish noise from real signal. Applying a threshold necessarily leads to the loss of some signal fractions if, for example, the pixel with the smallest part of a quadruple just falls below the threshold. This effect is seen in figure 7.65. It will become difficult for any reconstruction method to recover small signals as we will see in the following.

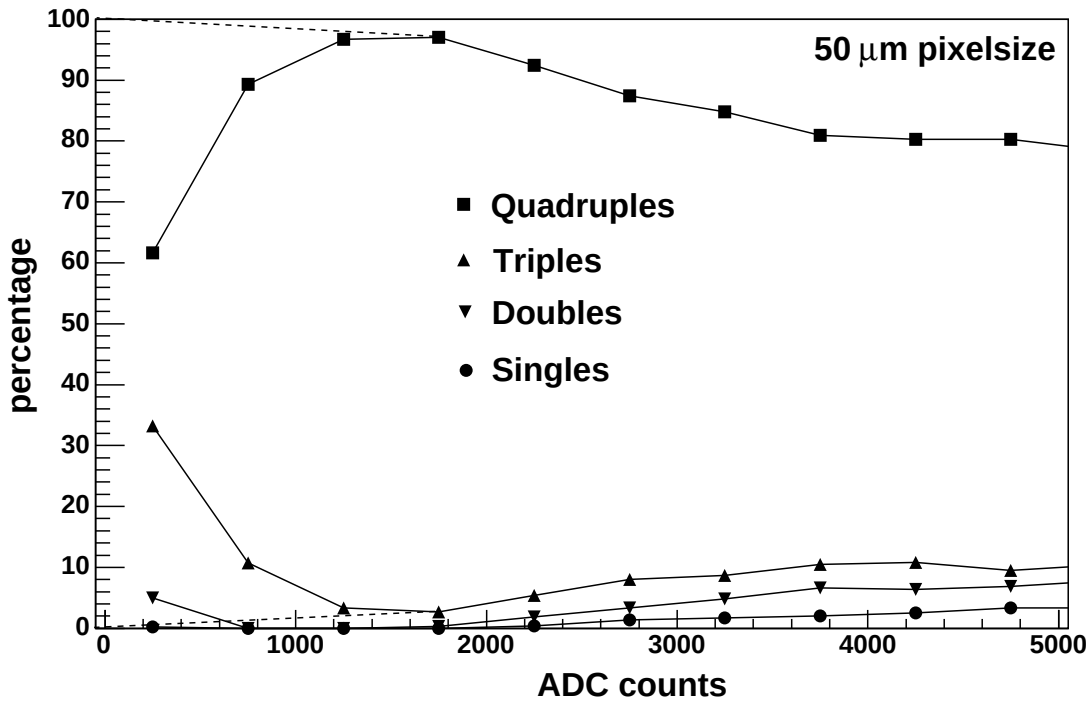

Figure 7.65: Probability for different splitting types depending on the photon energy. The dotted lines visualise the trend towards lower energies which would be visible in a no-noise scenario (see text).

The resolution – no matter with which method the reconstruction is done – confirms the two effects discussed. For a very low noise situation, as shown in figure 7.66, a best resolution (minimum reconstruction RMS) is found for low energies. For lowest energies the noise – even if very small – affects the reconstruction. For higher energies the splitting effect becomes smaller.

Finally, the results for a normal noise scenario are plotted in figure 7.67 for the three different pixel sizes. Although the neural network always gives the best resolution the differences of the three methods are negligible taking into account the statistical uncertainties: Only for the 150 μm pixel size the η-method seems to perform significantly worse than the CCOM method and the neural network. The improvement in resolution is much greater than the refinement of the pixels: The factor 3 between 150 μm pixels and 50 μm pixels has to be compared to the improvement in the optimal resolution at about 5.5 keV (1500 ADC counts): From 12 μm (for 150 μm pixels) to 0.8 μm (for 50 μm pixels) the improvement is roughly a factor 15.

The dependence of the resolution on the photon energy is, however, not the only interesting conclusion which can be drawn. Plotting the resolution as function of the true incident position clearly shows the different splitting types. Figure 7.68 shows how a neural network reconstructs the incident position from the pixel centre (0) to the pixel border (1).

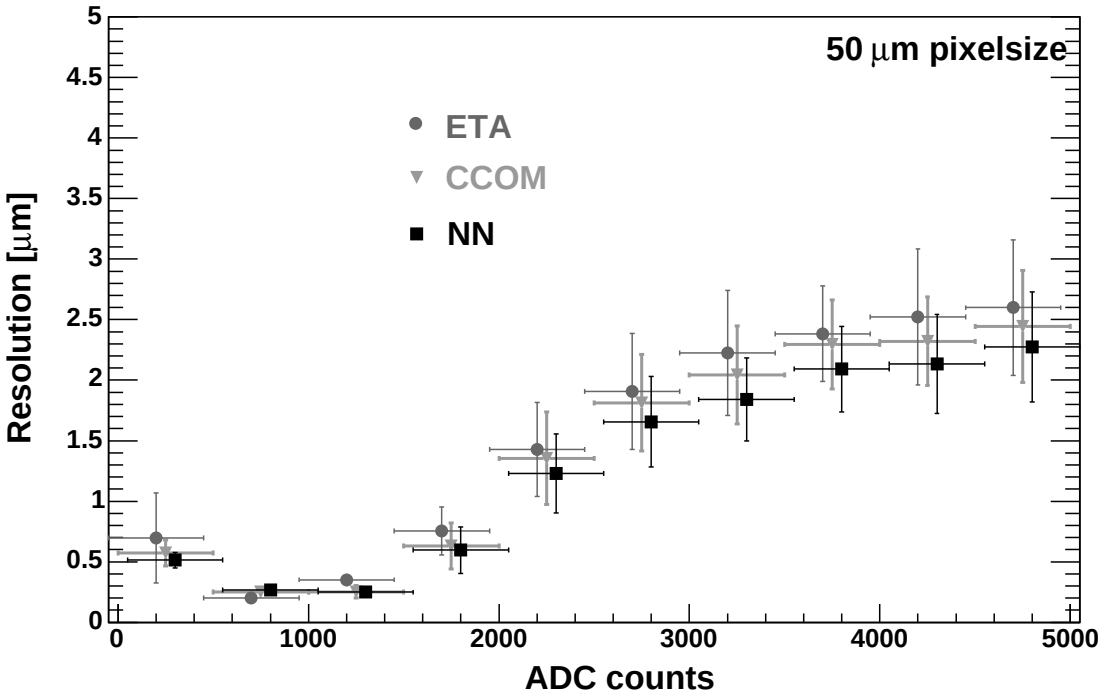

Figure 7.66: Resolution depending on the photon energy in a low noise scenario.

There is obviously a region around the centre of the pixel for which no helpful information is available. This region is directly related to the pixel size, it covers those incident positions for which single events are generated[17]. The reaction of the neural network (and of the CCOM method) is to assign the mean position to all these events which minimises the quadratic distance. One can see, for example, for the 150 μm pixel size that the mean value is around $0.35 = 26\mu m$), the total region is about $0.70 = 52\mu m$).

Figure 7.69 presents the effect just discussed, but this time with the reconstruction bias and the reconstruction variance. Added up in quadrature, these two components result in the total error giving the true resolution. It is clearly visible at least for the CCOM method and the neural network that the resolution becomes very good at the border of the pixel, as expected.

To conclude the analysis of the sub-pixel resolution, two-dimensional plots of the reconstruction errors will be shown giving a clear view of the resolutions which can be obtained for the experimental dataset. Fortunately, we can use the mesh hole positions to train the reconstruction methods directly with the experimental data and to test their performance.

Figures 7.70 and 7.71 show that differences between the CCOM method and the neural network can be seen primarily for singles and doubles where the neural network tries to get additional information from the supposedly not illuminated neighbours in y-direction. The smaller reconstruction errors of the neural network in y-direction for a part of the events are compensated by larger reconstruction errors in y-direction for other events. The expected behaviour is shown by the CCOM method which means that a flat error distribution is

[17] As discussed before, the splitting behaviour also depends on the conversion height and therefore on the photon energy. Hence the mesh experiment enables us in principle to perform a three-dimensional scan of the potential structure inside the semiconductor device.

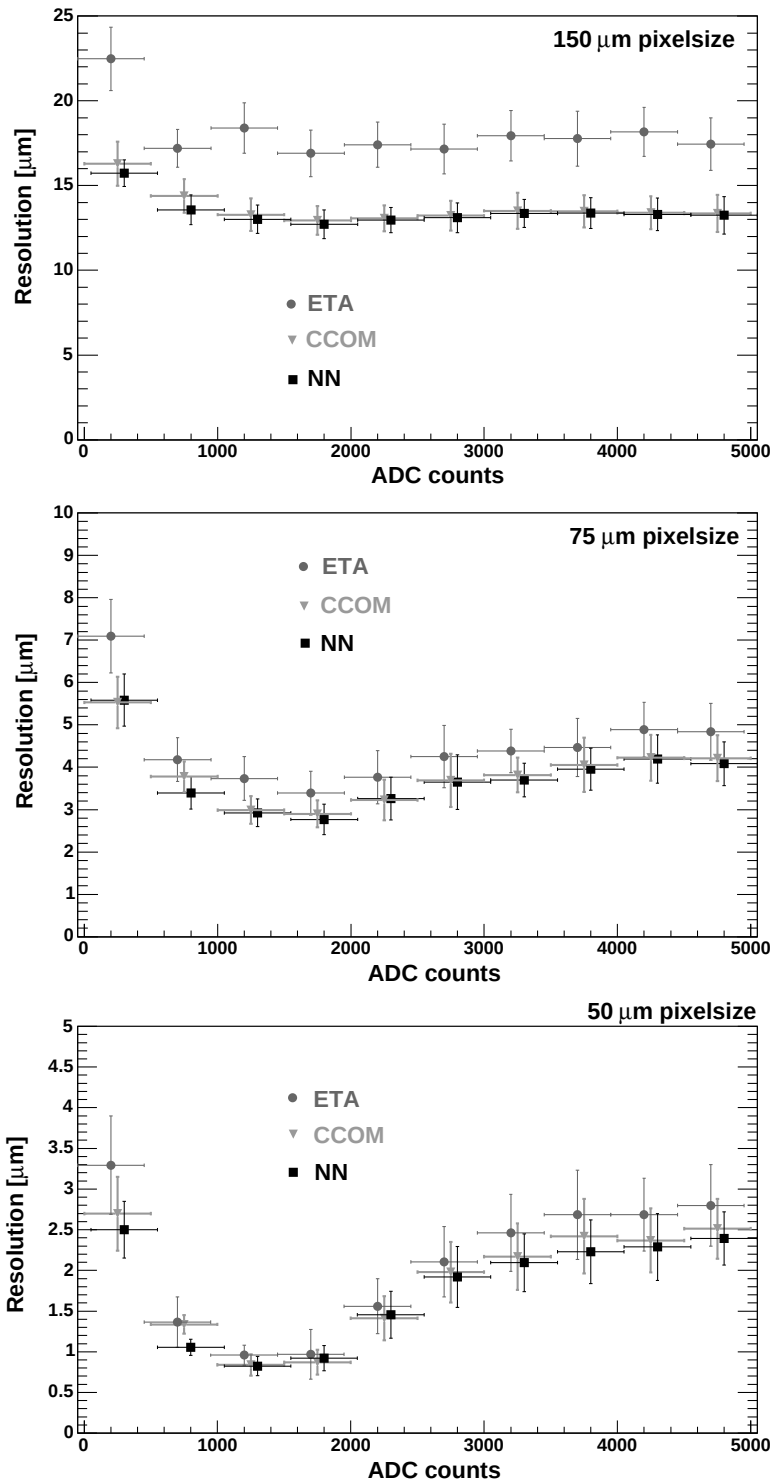

Figure 7.67: Resolution of the three reconstruction methods corrected centre of mass method (CCOM), η-method (ETA) and neural network (NN) in dependence of the photon energy for three different pixel sizes.

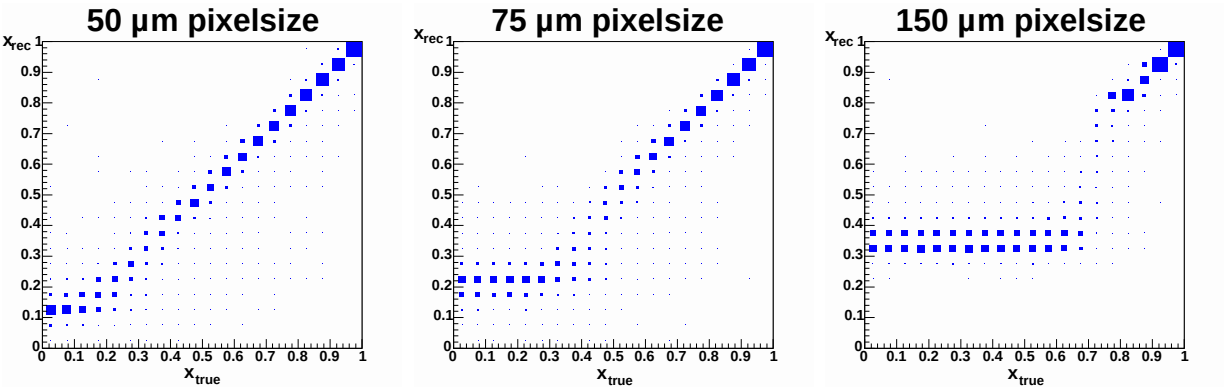

Figure 7.68: The correlation between true position (on the x-axis, 0 is the pixel centre and 1 is the pixel border) and reconstructed position (on the y-axis) is plotted for three different pixel sizes. The region in which no information is available (only singles) is clearly visible for the larger pixel sizes and disappears almost for the smallest pixel size.

obtained for singles covering the whole area within one pixel for which singles are generated. In comparison to the neural network result the width of the error distribution in y-direction is smaller but also the peak with small errors is missing.

The error distributions for triples have widths of around 4.2 μm in x- and 2.9 μm in y-direction (σ of a Gaussian fit). For quadruples the widths are around 3.0 μm in x- and 2.6 μm in y-direction. These widths apply to the neural network reconstruction and to the CCOM reconstruction, their differences are marginal. The larger widths in x-direction are due to the larger splitting effect in y-direction.

Systematic Uncertainties

As for the pileup rejection the systematic uncertainties will be applied to the detector information and the propagated uncertainties in the resolution will be studied. The systematic uncertainties are the same as discussed before, they will again be applied in three levels from "low" to "high". Table 7.29 summarises the results for 50 μm pixels and the three reconstruction methods.

method	resolution [μm]	stat.	syst. (low-medium-high)		
ETA	2.3	± 0.5	-0.2 $+0.0$	-0.2 $+0.4$	-0.1 $+0.8$
CCOM	2.0	± 0.4	-0.1 $+0.1$	-0.1 $+0.4$	-0.1 $+0.8$
NN	1.9	± 0.3	-0.1 $+0.1$	-0.1 $+0.4$	-0.1 $+0.8$

Table 7.29: Systematic uncertainties for the overall resolution in 50 μm pixels.

We see again that the differences in performance are negligible with respect to the statistical and systematic uncertainties. The largest contribution to the systematic uncertainties comes, as for the pileup rejection, from the crosstalk effect. This is evident since all reconstruction methods are very sensitive to the charge splitting ratio $\frac{c_{right}}{c_{total}}$.

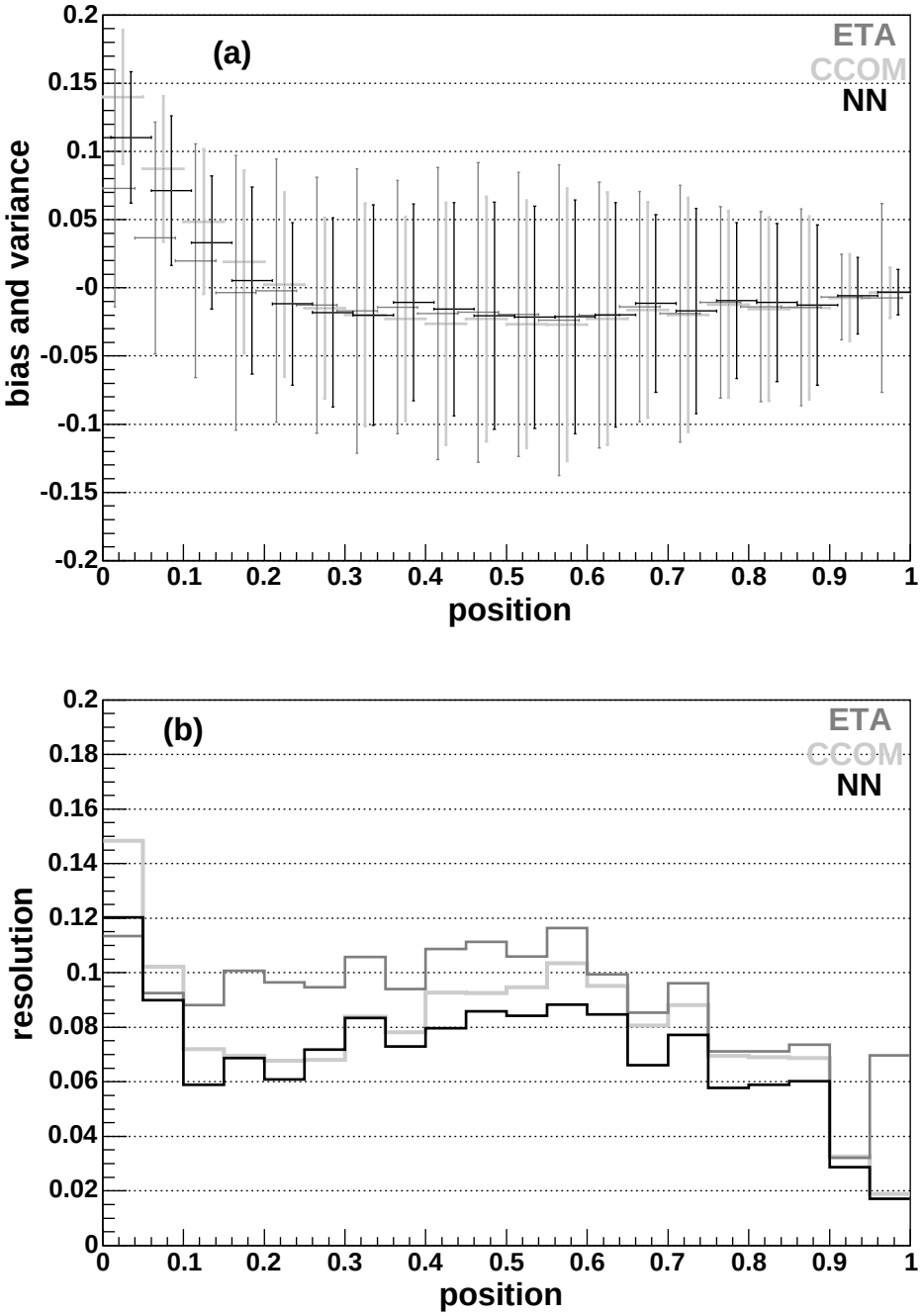

Figure 7.69: Comparison of resolution for corrected centre of mass method (CCOM), η-method (ETA) and neural network (NN) for $50\mu m$ pixels: (a) In bins of the true position (0 is the pixel centre and 1 is the pixel border) bias and variance, (b) the total resolution. The y axis is normalised like the x axis to 25 μm.

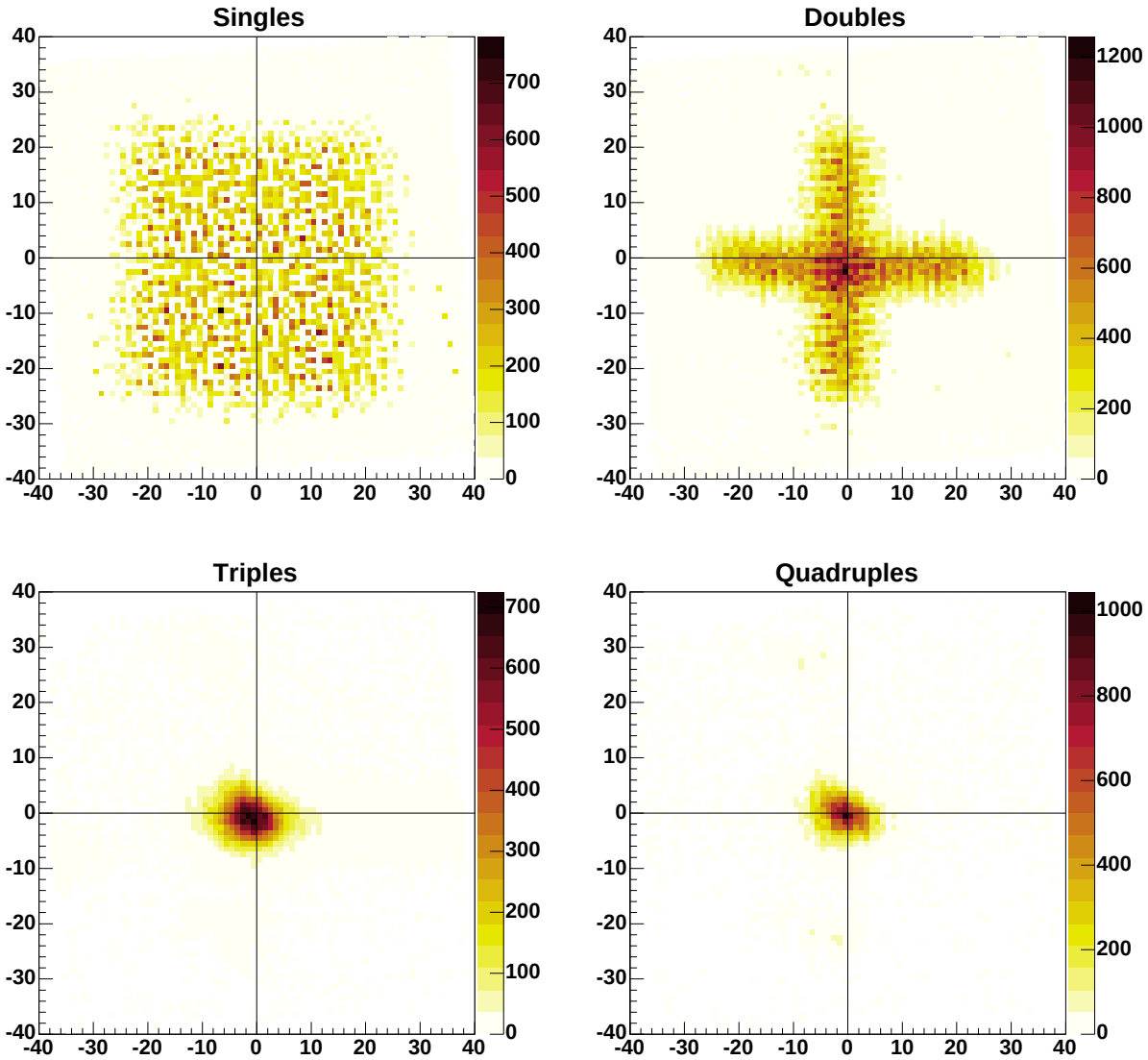

Figure 7.70: Reconstruction error from the reconstruction of experimental data (CCOM method) (75 μm pixel size).

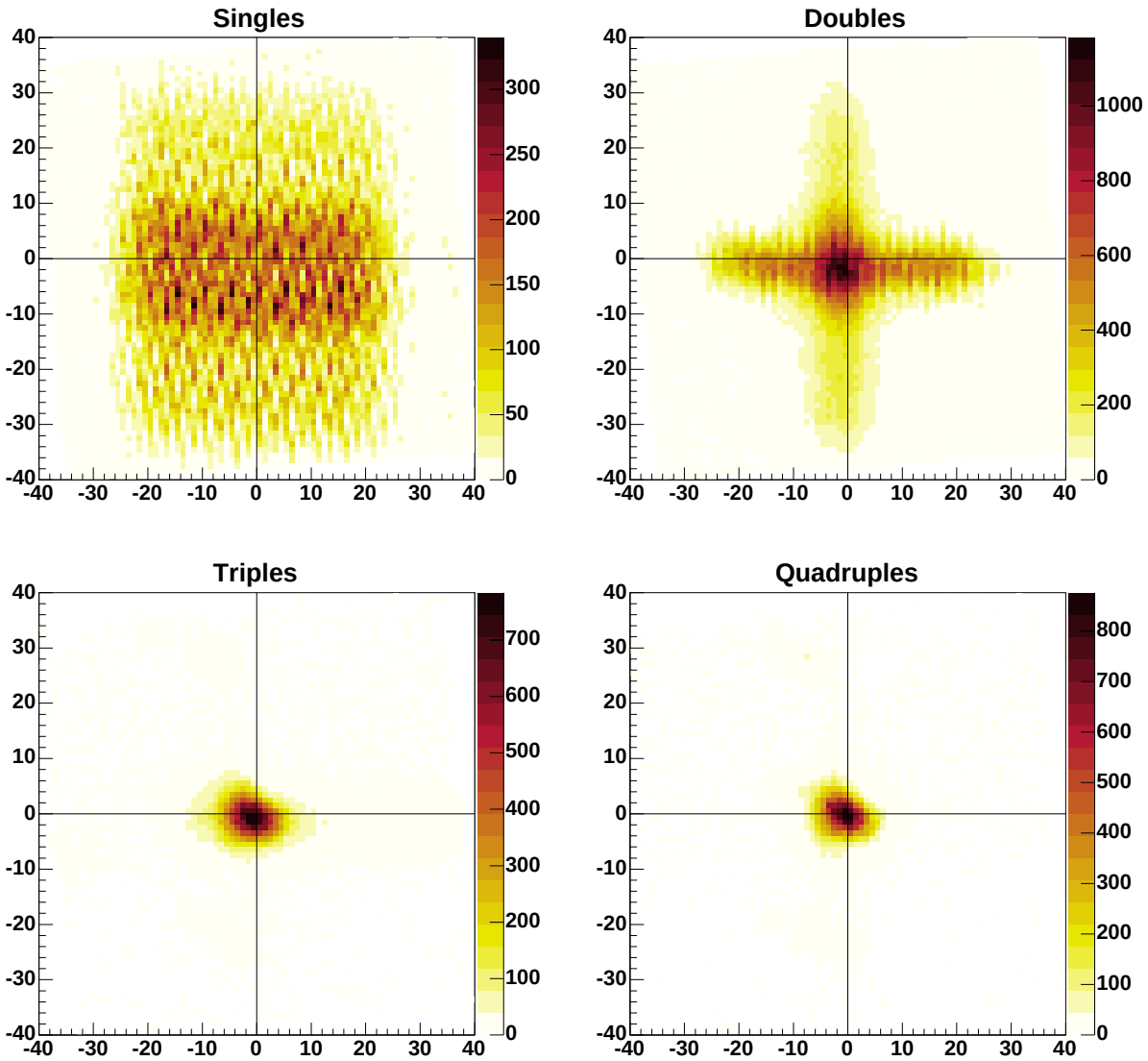

Figure 7.71: Reconstruction error from the reconstruction of experimental data (neural network) (75 μm pixel size)

.

Chapter 8

Discussion

In this chapter a summary of the most important aspects from the various analyses in the last chapter will be given. There, the results were grouped by experiment. Here, the main aspects of the application of statistical learning methods to physics analysis are summarised in an overview, taking into account the many results presented in the last chapter.

The following section 8.1 will discuss which kind of physics results can profit from the application of statistical learning methods. The question whether the performance of these methods really motivates to replace a classical method by one of the statistical learning methods will be addressed in section 8.2.

Another important question is whether statistical learning methods can be controlled well enough to make them a reliable and trustworthy part of a physics analysis. The frequently expressed fear of lacking control will be discussed in section 8.3. Continuing with the practical point of view we will then summarise different aspects of the correct handling of statistical learning methods in section 8.4.

Of course one wants to use the best learning method available. We will discuss in section 8.5 which method may be called "the best one" and in which sense. Some pieces of evidence will be summarised in section 8.6 which underline that statistical learning methods are not only a useful tool. They sometimes also show surprising artificial intelligence which turns out to be very helpful in physics analysis. Finally, the future of statistical learning methods in physics analysis will be shortly discussed in section 8.7 from a personal point of view.

8.1 Physics Results with Statistical Learning Methods

The applications presented in this thesis clearly show that statistical learning methods have a very broad spectrum of possible applications in high energy and astrophysics. Along with the variety of applications, the many different underlying physics subjects can be studied directly or indirectly with statistical learning methods.

The first kind of physics results which are made possible by statistical learning methods results indirectly from the improvement of the detector performance by these methods. The last chapter showed a variety of applications of this type. Usually the detector optimisation and the physics analysis are not connected very much so that the influence of an optimised detector is often not directly visible in the final physics analysis.

A first example for an impact of statistical learning on detector level on the subsequent analysis has been shown in section 7.2 for the neural network trigger at the H1 experiment. Previous analyses [102] already made use of the neural network trigger for level 1 sub-trigger 33 ($J/\psi \to e^+e^-$). Here a $J/\psi \to \mu^+\mu^-$ analysis was shown for data which was obtained while the neural network trigger was used to trigger these events. A significant increase in the number of J/ψ events triggered by the respective sub-trigger 15 has been calculated there. This increase in the number of events propagates directly to the J/ψ event yield shown in section 7.2.6 and to the final analysis results, for example to the statistical and systematic uncertainties of the measured cross section.

A second example of a detector which may be optimised with statistical learning methods is the pixel-detector aboard the XEUS satellite (section 7.7.1). If very bright sources are observed the photon fluxes may become far too high to send all observed pixel patterns down to earth. Most of them will be pileups anyway which are not useful in the analysis. The background suppression proposed in this thesis would be able to cope with a three times higher background rate at the same photon efficiency compared to the classical method. Again the impact on the physics analysis is indirectly visible in the higher X-ray photon count rate which scales, for example, the statistical uncertainties in the X-ray spectra.

The second kind of physics results which are made possible by statistical learning methods results directly from the improvement of the analysis by statistical learning methods. This analysis improvement is mostly related to an efficient background suppression. The goal may either be to test a hypothesis or to decrease the statistical uncertainties.

The search for instanton-induced events at H1 (section 7.3) is an example where the question given in the form of two different simulations turned out to be ill-posed if one wants to take into account both simulations equally. The strongest results show no room for instantons if the data is compared to the CDM simulation whereas the MEPS simulation would confirm the instanton hypothesis, even in terms of the absolute number of events predicted by the QCDINS simulation.

A second example for the test of hypotheses is given by the measurement of the Higgs boson parity (section 7.4). It was shown that the significance of the parity measurement is clearly larger if the discrimination is done with statistical learning methods compared to the classical approach. This opens up the possibility for additional studies like the determination of a mixing angle for two Higgs bosons, e.g. in a super-symmetric extension of the standard model.

Finally the γ-hadron separation for the MAGIC telescope (section 7.6.1) is an example in which a clever background suppression makes a flux measurement possible even for very weak sources. The flux spectrum of the weak source 1ES1959 was presented as an example of the new analysis possibilities opened up by the application of statistical learning methods.

8.2 Performance of Statistical Learning Methods

The results from chapter 7 confirm the general statement from section 3.9: Statistical learning methods are often the only possibility to perform a certain analysis because there is no knowledge on which a classical algorithm could be built. However, even if classical algorithms exist statistical learning methods have proven worth to be considered as an

alternative to classical algorithms because of their performance.

That statistical learning methods show indeed significantly better performance than their classical competitors has been proven by the results in the last chapter. A few highlights from the whole set of analyses presented in this thesis are:

- The combination of simple one-dimensional cuts might be regarded as an alternative to the neural network trigger at the H1 experiment at least for some special sub-triggers. The sub-trigger for deeply virtual Compton scattering (section 7.2.3) is one where this approach works surprisingly well. Yet, a comparison of the efficiencies at a fixed rate reduction of 80% reveals that with the simple cuts only 83.6% efficiency can be achieved while neural networks give 96.5%. This is a reduction of inefficiency (lost signal events) by a factor 4.7.

- The research on the determination of the Higgs boson parity (section 7.4) at a future linear collider was up to now focused on fitting a cosine function to a certain angular distribution. The significance of this method was improved by applying different preselections. Yet, another increase of the significance by 23% could be achieved by changing the strategy from the theory-oriented cosine-fit to a simple discrimination of both parity states by a neural network. The lowest significance which may be obtained in the real experiment (with 90% confidence) has been increased by 28% from 3.61σ to 4.61σ.

- The supercuts ("dynamical cuts") method is like a historical standard for the γ-hadron separation for the MAGIC telescope (section 7.6.1). In comparison to statistical learning methods the supercuts show a lower background suppression. The strong effect on the significance of the photon signal is visible for the weak source 1ES1959. There the significance of the signal could be improved by 67% using a neural network to perform the γ-hadron separation. The number of excess events was increased simultaneously by 43%.

- The rejection of pileup events in the pixel-detector aboard the XEUS satellite (section 7.7.1) may be an important subject for future X-ray missions even online, i.e. aboard the satellite before transmission to earth. The classical algorithm used for this purpose up to now shows a good efficiency but only a low rejection of pileups. An increase of the rejection rate by a factor 3 at the same efficiency can be gained by applying a neural network instead.

8.3 Control of Statistical Learning Methods

Because of the significant performance improvements obtained with statistical learning methods, they are and will continue to be applied in many new problems in physics analysis. However, some physicists hesitate to make use of statistical learning methods. They show some kind of general fear that the application of these methods might disturb or even destroy the whole analysis ("black box syndrome").

Different aspects of this fear have been discussed in this thesis:

- It is often said that statistical learning methods are black boxes, that one cannot look inside and cannot understand how they are working. Chapter 5 gave an introduction

to all popular learning methods, showed the ideas on which they are founded and how typical implementations of these methods work. The mathematical background and the geometrical interpretation of each method has been presented. In addition some basic aspects of statistical learning theory have been discussed in chapter 4.

- Even though the learning methods can really be understood and nicely interpreted, nothing forbids their application as black boxes. The remaining fear is mostly based on the question whether statistical learning methods can be controlled in the right way. In the context of a physics analysis, statistical and systematic uncertainties provide the most important means of controlling the results. Section 3.13 discussed both statistical and systematic uncertainties in a detailed way and showed that there is no uncertainty in the learning method itself. The systematic uncertainty of the statistical learning method is obtained by a propagation of systematic uncertainties of the inputs to the output. This propagation can be calculated for any statistical learning method. Some examples from the last chapter are:
  - Statistical and systematic uncertainties have been calculated for the neural network trigger of H1 (section 7.2). The networks have shown a very stable behaviour. Even if very high systematic uncertainties for the inputs are assumed, the propagated uncertainties of efficiency and rejection are well under control (fault tolerance).
  - Systematic uncertainties have been very important for the search for instantons, as only they can give the probability of the instanton hypothesis (section 7.3). However, the calculated uncertainties finally revealed that the two underlying simulations of standard sources (QCD) are not compatible.
  - For the pileup rejection aboard the XEUS satellite (section 7.7.1) the calculated uncertainties showed that the very high efficiency for single photons is stable and that the pileup rejection may decrease but only within a relative change of 5%.
- A second important step towards a trustworthy analysis with statistical learning methods is the control of any statistical effect which may lead to a bias in the predictions. On the one hand the overtraining effect has to be controlled. The division of the available examples into training, selection and test set to have an independent estimation of the true performance was discussed in section 3.11. This technique has been exercised in many examples in chapter 5 and in the analysis in chapter 7. On the other hand the comparison of learning methods has to follow statistical rules. Otherwise one might wrongly claim a significant performance difference. The statistical techniques presented in section 3.15 have been used explicitly in the case of the DVCS dataset in section 7.2.3 and for the D^* dataset in section 7.2.7.
- The control which is needed in addition to the above topics covers exactly the same questions which have to be thought about for any physics analysis like:
  - Could the results be crosschecked with an alternative data selection? An example was discussed in section 7.2.5 for the $J/\psi \to e^+e^-$ dataset. A too loose cut in a recent selection was detected by a comparison to the output distribution of an older selection.

- Which events are rejected by a certain cut and how do they look like? For the charged current selection in section 7.2.4 events which would have been rejected by the neural network trigger have been scanned. Two events have been revealed by this which should not have been part of the charged current selection.
- Is there an artificial bias of the efficiency of the selection cuts in dependence of a certain observable? The efficiency of the neural network trigger for DVCS event (section 7.2.3) has been evaluated as function of the angles and energies of the photon/electron. All four histograms showed a nicely flat efficiency distribution. As a second example the pileup rejection and the position reconstruction in the pixel detector have been checked for their spectral dependence (sections 7.7.1 and 7.7.2). As expected both tasks show problems towards lowest energies due to the noise. The worse performance of the position reconstruction for very high energies has been explained by the energy dependence of the absorption length.
- How can one determine the efficiency of a trigger when it is already rejecting? The principle of orthogonal triggers was used as an example for the DVCS and the $J/\psi \to \mu^+\mu^-$ dataset in sections 7.2.3 and 7.2.6. In both cases the efficiency determined with this method confirmed the estimation from the test set. As mentioned in section 7.2.2, in some cases a Monte Carlo simulation needs to be instrumented to determine the correct efficiency of the trigger.

8.4 Handling of Statistical Learning Methods

Even if a statistical learning method is well controlled there might still arise some problems from the complex handling. There are several tasks which may make it difficult to obtain a well-performing classifier or regression model: The data has to be preprocessed and/or preselected in an optimal way, the best inputs have to be selected and the best training target has to be defined. Finally, each learning method has its own parameters which have to be varied in order to avoid overtraining and to find the optimal performance. All these topics have been discussed in chapter 3 and have been illustrated by examples in the last chapter.

The importance of a good preselection was discussed on the Higgs boson parity problem (section 7.4) and a good preprocessing was needed for the position measurement for the neutron detector (section 7.5), for the shower image in the MAGIC telescope (section 7.6) and for the pixel cluster in the X-ray pixel-detector (section 7.7). The procedure of selecting the optimal subset from a given set of inputs was demonstrated with the $J/\psi \to \mu^+\mu^-$ dataset from the H1 trigger (section 7.2.6). The difficulty to choose also the right training target was faced for the Higgs boson parity problem (section 7.4) and for the energy estimation in the MAGIC telescope (section 7.6.2). The parameter optimisation problem was discussed with the help of the dijets dataset from the H1 trigger (section 7.2.7).

Finally, many examples for tricky details have been presented like the multi-class problem for the H1 level 1 sub-trigger 83 where two different physics channels come from the same sub-trigger and both need a high efficiency (section 7.2.7). Another typical problem is that the training is done with a simulation but the test should of course be done with real data. Examples for the problems arising from this combination have been discussed for the search for instantons (section 7.3), for the position reconstruction in the neutron detector (section 7.5) and for the shower image in the MAGIC telescope (section 7.6).

8.5 The "Best Learning Method"

The summary of performance highlights in the second section of this chapter already made clear that neural networks have indeed earned their status as the most often applied statistical learning method in physics analysis. Throughout the results presented in chapter 7 neural networks have been among the best performing learning methods. This can also be seen in recent publications about applications of statistical learning methods in physics analysis [118, 119]. However, one cannot state that generally the neural network is the best learning method since datasets may exist where other learning methods are suited better. A comparison of different methods should always be done where possible.

Section 3.15 made clear that the comparison of learning methods has to follow some specific rules. The statistically correct comparison of some learning methods was performed as an example on the D^* dataset from the H1 experiment (section 7.2.7). There the random forest method and neural networks performed best in comparison to naive Bayes and k-nearest-neighbours. More hints concerning performance differences can be taken from the comparison of fast classification techniques on the DVCS dataset (section 7.2.3) where the neural network and the support vector machine performed best in comparison to linear discriminant analysis, naive Bayes and combinatorial cuts. Further information can be derived from the separation powers observed in the search for instantons (section 7.3) where neural networks showed better results than the range search method or the combinatorial cut search. Additional results come from the Higgs parity measurement (section 7.4) where the random forest method performed similar but not as good as the neural network, and from the γ-hadron separation for the MAGIC telescope (section 7.6.1) where neural networks performed again slightly better than the random forest method.

In summary the three methods support vector machines, random forests and neural networks generally show the best results. There might be datasets where other methods perform better but usually the three mentioned methods perform particularly well in physics analysis. A ranking among these three methods is difficult to obtain and certainly has to take into account a very specific application. One of them will be better suited for a specific application than the others although all three of them show generally good results nearly independent of the dataset.

8.6 Artificial Intelligence in Statistical Learning Methods

There have been some examples in the last chapter which lead back to the roots of statistical learning methods in artificial intelligence. We have seen for example, that there is a specific class of events in the selection of J/ψ events from the year 2004 which are classified as background (section 7.2.5). Indeed, a scanning of these events revealed that one of the selection cuts was too loose and let through a lot of background. Although these events were used in the training procedure as "signal" they were clearly identified as background.

An even more important detection with neural networks was made in the selection of charged current events (section 7.2.4). This selection contains only a few hundred events for the whole year 2004 which makes a clean event sample even more important. All the events used to check the neural network performance have been scanned by a physicist. One event from the charged current selection would have been rejected by the neural network

and this event turned out to be most probably an overlay event which should not have been in the charged current selection. A second event would be rejected if large systematic shifts are applied. This event turned out to be a cosmic event which also should not have been part of the charged current selection.

In summary, statistical learning methods (here neural networks) showed an (un)expected intelligent behaviour which revealed yet unknown details of certain physics selections. The results from the statistical learning methods were surprising and led to insights on the human side which lacked before. This is more artificial intelligence than the methods were designed to show. The results are even more encouraging if one takes into account that the physics selections are done on the analysis level, i.e. with fully reconstructed events. The neural network, in contrast, made its surprisingly clever decision only based on the coarse detector information which is available on the second trigger level.

8.7 The Future of Statistical Learning in Physics Analysis – A Personal View

The number of possibilities to apply statistical learning methods to problems in physics analysis will clearly grow further. The rising complexity of each experiment, the completely computerised analysis frameworks and the hunt for clever algorithms which can cope with the enormous amounts of data clearly point towards new applications for statistical learning methods. The key issue in future applications of statistical learning methods will be the background suppression.

Neural networks have been some kind of the standard statistical learning method in physics analysis for several years. Two important new methods have been developed during the 1990s: The random forest method and the support vector machine. Both of them have now to be taken into account as possible alternatives to neural networks as the discussion about the best learning method above has shown. Whereas the possibility exists that some new design leads to an even better kind of learning method the analysis presented here has demonstrated that the performance differences among the best learning methods are often negligible, in contrast to the differences to simple learning methods or to classical algorithms. The feeling that one can almost always reach the theoretical optimum performance with at least one of the methods presented in this thesis is not misleading.

But not only the performance is an important criterion for the future application of statistical learning methods in physics analysis. Even more important is the existence of a working implementation of the algorithm within the standard framework which is used for a specific experiment. The optimum would be, of course, to have several different methods implemented so that one can compare their performance on the different kinds of analyses. This was attempted in this thesis.

Most important for the future application of statistical learning methods in physics analysis is the knowledge and experience of those people who want to apply these methods to a specific problem in their analyses. New generations of diploma and PhD students will try out different types of learning methods on different problems. They are hopefully equipped with basic knowledge about statistical learning methods and the typical problems arising from the application in high energy and astrophysics like discussed in this thesis. This will not only make the work easier but will also help to avoid common mistakes and misconceptions.

Chapter 9

Conclusion

The various analyses presented in this thesis cover typical applications of statistical learning methods in high energy and astrophysics experiments. They have shown that these methods lead directly or indirectly to very interesting physics results. Different analysis results have been discussed which could not have been obtained without the help of statistical learning methods.

These intriguing results have been made possible by the convincing performance of statistical learning methods. Numerous examples have been discussed in this thesis where a statistical learning method significantly outperforms the competing classical algorithm. In addition, statistical learning methods sometimes show more artificial intelligence than expected as they lead to insights which have not been asked for. The comparative study among different learning methods has shown that neural networks have earned their position as a standard tool in physics analysis.

Despite the interesting new physics results and the remarkable increase in performance obtainable with statistical learning methods, physicists often hesitate to make use of these methods. A strong emphasis was therefore put onto the understanding and interpretation of statistical learning methods. Furthermore clear guidelines for the correct application of these methods have been given.

A special focus has been put on the controlling techniques important for physics analysis: Methods for the unbiased estimation of efficiencies have been presented and applied to many examples. The correct calculation of statistical and systematic uncertainties has also been exercised on different examples. It was thereby shown that statistical learning methods can be well controlled.

Knowing how to control statistical learning methods in the physically correct way and knowing how to apply them in the best possible way to a given analysis problem enables the physicist to profit from their extraordinary performance and to obtain thus new, interesting physics results.

Appendix A

Statistical Learning Methods in Hardware

Naturally some kind of hardware has to be used to calculate the output of statistical learning methods. Programs running on standard PCs (like those mostly used in this thesis) are considered as a "software" solution. They are only software in the sense that a piece of hardware has a significant speed advantage if it is dedicated to the evaluation of a specific kind of statistical learning method.

However, today's processors also allow the fast calculation of the output of statistical learning methods, even in parallel in a farm of processors. Execution times in the order of a few microseconds can easily be reached, for example, for the calculation of a medium sized neural network on a single processor. A bottleneck in the application of this solution to a real online application may be the I/O speed of the PC.

Hardware which is dedicated to the calculation of a specific statistical learning method can still be very flexible like, for example, a digital signal processor (DSP) or a field programmable gate array (FPGA). The structure of the program which has to be loaded to the hardware first fixes its behaviour. 16 parallel DSPs are, for example, used in the neutron detector discussed in this thesis to perform the reconstruction of the neutron incident position [20, 120]. The corresponding electronics for analog and digital processing of the detector data is shown in figure A.1. A clever FPGA design for the fast execution of large feed forward neural networks ($400ns$!) is presented in [84]. The CNAPS (Connected Network of Adaptive processorS) VME cards [121] used in the level 2 neural network trigger of the H1 experiment have been designed for the fast execution of parallel algorithms, e.g. in feed forward neural networks. These chips have been commercially available but are no longer. Figure A.2 shows a diagram of the hardware used in the level 2 neural network trigger.

Figure A.1 and A.2 make clear that the implementation of the statistical learning method in some piece of hardware is one of the minor steps towards a working online system. The data processing which needs to be done to provide the desired inputs for the statistical learning method is an important and costly step.

The main advantage of a dedicated piece of hardware in comparison to a classical CPU is the parallel implementation of statistical learning methods. Thus some of the learning methods discussed in chapter 5 are especially well suited to be used in an online application where a short execution time is mandatory. The naive Bayes method, linear discriminant analysis, and simple cuts can be implemented in parallel algorithms or may even be simple

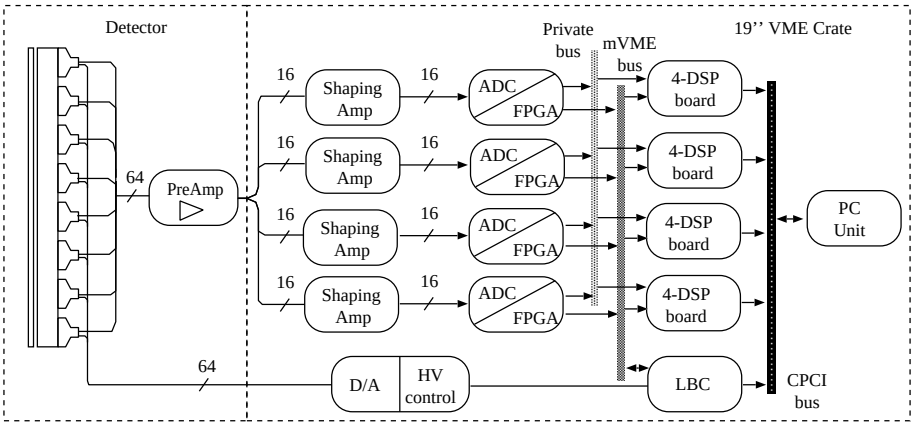

Figure A.1: Block diagram of the analog and digital signal processing used in the neutron scintillation detector system.

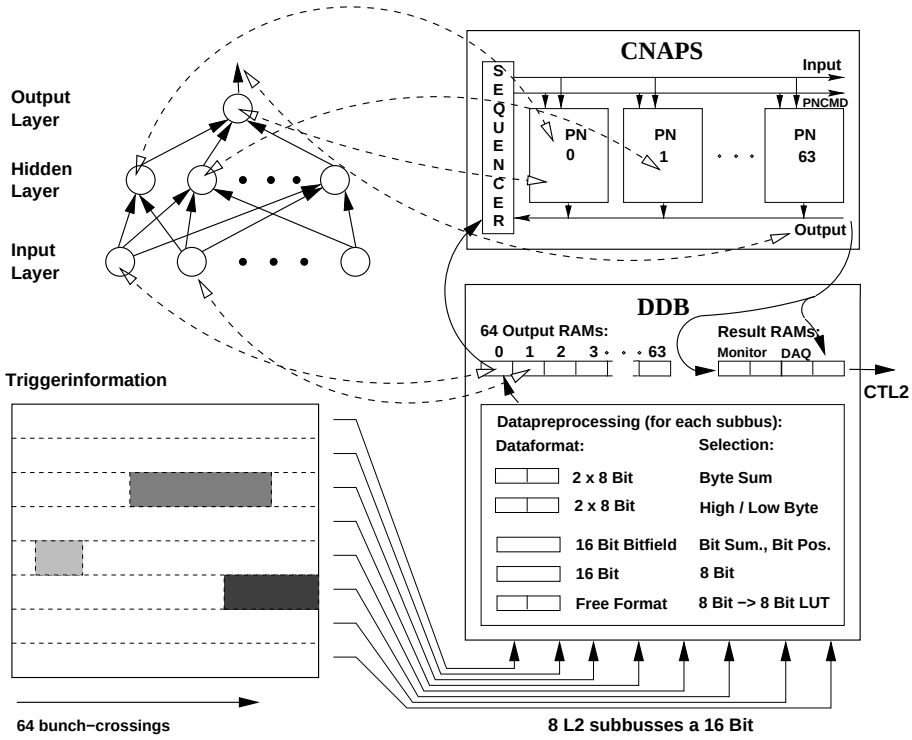

Figure A.2: Diagram of one L2NN trigger-box: Data arriving on the different L2 subbusses is preprocessed by the Data Distribution Boards (DDB) which then manages the interaction with the CNAPS chip where the actual computation of the network is performed in parallel.

enough to be implemented serially. Feed-forward neural networks and support vector machines are special candidates for the parallel architecture since the calculation of all outputs in the hidden layer (for neural networks) and the comparison with all support vectors (for the support vector machine) naturally parallelise. Typical numbers of parallel units are 10 to 60 for neural networks and 100 to 1000 for support vector machines.

Other learning methods like local density estimators which compare a new event in some way to all the training events are less suited for hardware implementation. They might have an implementation which uses parallel algorithms up to some degree but generally show a more serially oriented processing. This would lead to considerably long execution times in an online application. For decision trees one has to proceed serially from the first cut at the root of the tree up to the last decision in one of the branches. The time consumption is difficult to be determined in advance since the number of cuts which will have to be done is not known in advance.

Meta-learning strategies like bagging, boosting and the random subspace method do parallelise but they multiply the needed parallel processors by the number of classifiers. Bagging, for example, is typically applied with 100 trainings. This would mean that 100 classifiers have to be calculated in parallel.

Appendix B

Implementation Details

B.1 Preprocessing for the Pixel-detector

In this section the preprocessing scheme will be presented which has been newly implemented for the analysis of the X-ray pixel detector data in section 7.7. The preprocessing of the detector data is a two step process: First known detector and readout effects have to be corrected ("calibration") and afterwards the signals need to be extracted. As discussed in section 2.5 the charge cloud generated by one photon is usually distributed over up to four pixels. Thus once the illuminated pixels have been identified, they have to be grouped into clusters ("patterns") and the total charge of each cluster has to be determined.

This whole preprocessing procedure is essential for any following analysis. Because such an analysis did not exist in an object oriented programming language it was developed in this thesis. This implementation will be described in the two following sections. Many new possibilities compared to the existing implementations make it a powerful tool.

B.1.1 Correction of detector and readout effects

The correction of effects which modify the measured pixel signals is done in three steps:

1. First the *common modes* for all rows in all frames have to be determined. The common mode is an offset which is shared by the pixels in one row and varies with time. It is probably caused by a time dependent amplifying mechanism which acts in parallel on all pixels of one row during the readout process. The common mode is usually determined as the mean of all the pixels of one row. Care must be taken that the line is "dark" which means that illuminated pixels (with charge, e.g. induced by a photon) have to be omitted. Subtracting the calculated common mode leaves an offset value which is independent for each pixel.

2. The offsets per pixel are calculated as the mean value over several frames. Again care must be taken that no signal charge disturbs this calculation. The variation around the offset value is used to derive a standard deviation. A pixel is considered as illuminated if its value after subtraction of common mode and offset exceeds n standard deviations where n is typically 4 or 5.

3. The first two calibration steps use dark frames in which no X-ray photon should appear. These first two calibration steps are enough to recognise events. The last

calibration step is done with X-ray photons of a specific energy to correct for two detector effects which affect the readout of deposited charges: The gain correction applies a factor to each column of the pixel matrix to compensate the slightly different amplifications per channel. The charge transfer efficiency describes the loss of charge in the transfer process and is thus a function of the row in which the charge was deposited. The charge transfer efficiency is usually described as a linear function either for all columns or column-wise.

To estimate the gain factors and the charge transfer efficiencies, events from a calibration source need to be extracted for this step as described in the next section. Linear fits of the measured charges along each column are done and result in a slope and an offset value per column. The offset values can be transformed into the gain factors since they represent the charges measured without transfer losses. The slopes of the fits describe to a first approximation the charge loss per row and thus the charge transfer efficiency.

Different methods to derive common mode, offset and variance as well as gain and charge transfer efficiency can be easily plugged into the calibration procedure. Additional detector effects like hot and cold pixels are taken into account by masking them out and ignoring events which would have illuminated this pixel.

B.1.2 Signal Extraction

The signal extraction is performed on the detector data which was calibrated according to the above procedure. As mentioned above, pixels can be recognised as illuminated if the deposited charge exceeds a 4σ or 5σ threshold. After a zero-suppression step these pixels are clustered together. The clusters found form the events which are processed further. Events generated by minimum ionising particles (MIP) can be filtered out due to the very large energy deposition (above the "MIP-threshold") and, in most cases, also by their geometry (if the MIP passes the detector almost horizontally). Events generated by one X-ray photon occupy at most 2×2 pixels for the detector types discussed in this thesis. Thus larger patterns can be directly identified as background. Event-clusters which remain after this background suppression form the input for any method determining the final position and energy information per event.

Different event filters, depending on charge, geometry or both, can be chosen and combined in the newly created preprocessing framework. New filters are easily added to the framework. For example, the pileup-filters presented in section 7.7.1 can be used here. Any method which calculates the final information about each event from the extracted clusters can also be easily plugged into the framework. Usually a raw output is produced consisting of a list of all pixels belonging to a cluster with their individual calibrated charges. Instead a final charge and position information can be produced. The position could be estimated with methods like those presented in section 7.7.2.

Figure B.1 shows the reconstructed energy spectrum of a test measurement, the titanium K-α and β lines are visible. The energy spectrum obtained with the newly implemented preprocessing scheme described here is compared to two spectra which have been obtained with two preprocessors developed in the MPI semiconductor laboratory [116, 115]. The negligible differences in the energy spectra show that the newly implemented preprocessor works fine.

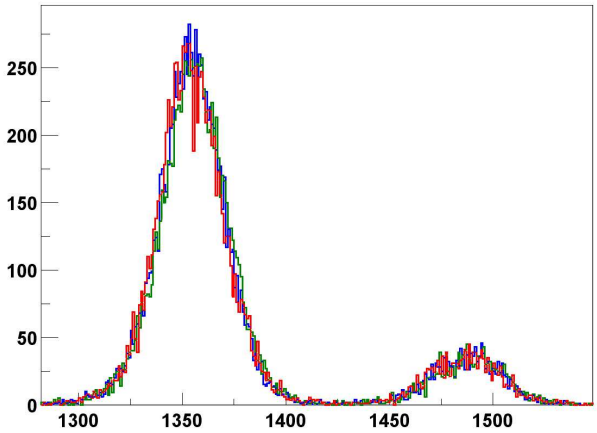

Figure B.1: Reconstructed spectra obtained with the preprocessor described here (blue) compared to two preprocessors which have been developed in the MPI semiconductor laboratory (red and green).

B.2 Analysis of the Mesh Experiment

The principle of the mesh experiment was adopted from research done in Japan [122]. The importance of the mesh experiment for a better understanding of the detector was already mentioned in section 2.5.1. Here we will focus on the first step of the analysis of the mesh experiment: The determination of the mesh parameters, i.e., how the mesh is aligned relative to the CCD pixel structure. Once the angle and the offsets are found, the positions of all holes with respect to the CCD pixels are known. Any photon event can then be identified with a specific hole position which determines its incident position.

To reconstruct the mesh parameters a new method has been invented, implemented and checked on simulated and experimental data. The principle used for the reconstruction is visualised in figure B.2: The moire pattern formed by single events can be used to calculate the rotation angle and offsets in x- and y-direction.

The rotation angle is closely related to the angle formed by the grid of the moire cells with respect to the CCD coordinate system. In the same way the offsets of the mesh are related to the offsets of the grid of moire cells. The lines drawn into the moire pattern of single events in figure B.2 on the right visualise the grid of moire cells which needs to be determined. This is done by projecting the moire pattern row-wise or column-wise onto the x- or y-axis, respectively, and fitting the periodic pattern. The periodicity and offset of the periodic functions obtained for all rows and columns are then used to extract the rotation angle and the offsets of the mesh.

The reconstruction method used in the Japanese experiment [122] tries to estimate the mesh parameters by minimising a cost function. This cost function measures the misplacement of the mesh with the currently assumed parameters. The misplacement is determined with the help of the single events: The single events should appear (only) in those pixels over which a mesh hole is placed centrally.

Compared to this minimisation of a cost function the fitting method developed here does not depend on the uniformity of the single events and has no problems with secondary minima. The precision of the reconstructed mesh parameters is as good as for the minimisation procedure and the application is much easier since no secondary minima need to be excluded.

A comparison of both reconstruction techniques in a simple simulation of a mesh with $150\mu m$ spacings over a CCD with $150\mu m$ pixels showed that the method described here can reconstruct the rotation angle within $0.04mrad$. The minimisation of the cost function

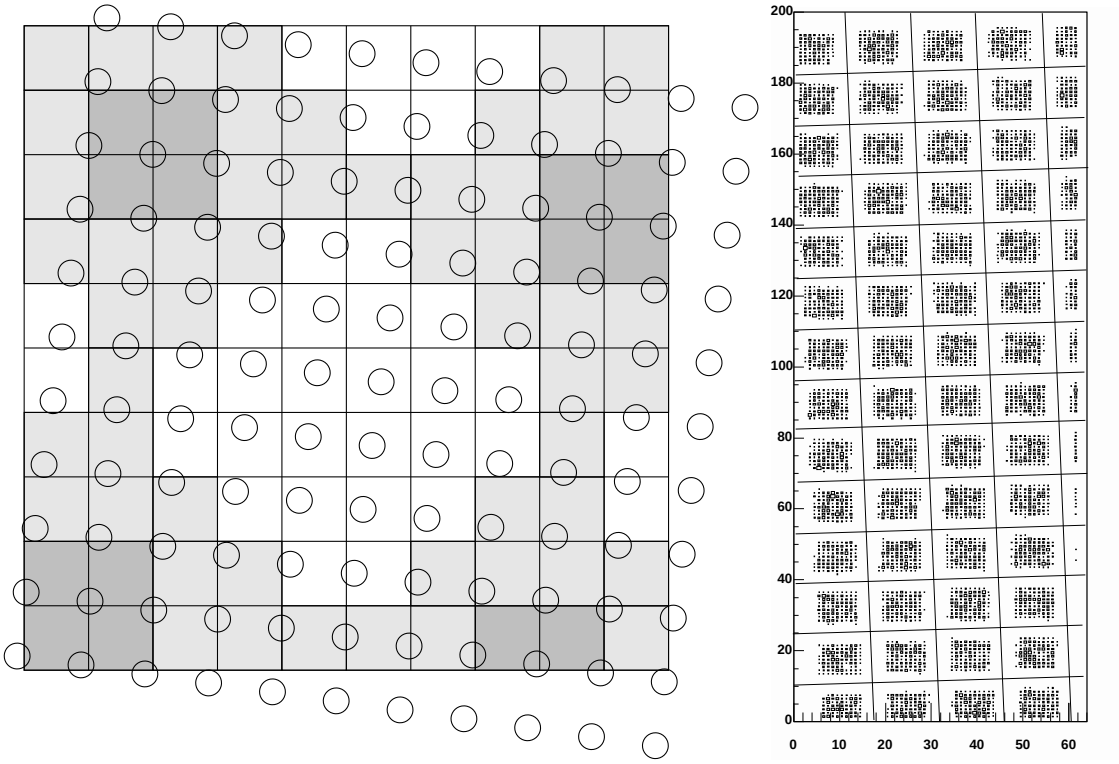

Figure B.2: The mesh-experiment: Single events are likely to happen whenever the mesh holes lie over the centre of the pixel. If the hole is close to a border the charge will split among multiple pixels. The moire pattern of single events (right) is induced by the slight rotation of the mesh relative to the CCD pixel structure. The lines which have been added manually in the right plot visualise the grid structure of the moire cells.

(with starting values from from the our method) could reconstruct the rotation angle only within $0.11mrad$. The offsets are better reconstructed with the minimisation technique resulting in an average resolution of $1.6\mu m$. The method described in this section resulted in an average resolution of $2.3\mu m$.

B.3 Automatic Parameter Optimisation

In the following three sections details about the programs will be presented which automatically generate trainings for different parameter settings. This is done for the learning methods neural network, support vector machine and random forest. These three methods are discussed in more detail than the other methods presented in chapter 5 because they have been found to be always among the best performing learning methods (compare the discussion in section 8.5).

B.3.1 Parameters for Neural Networks

The training of feed forward neural networks is very often done with the back-propagation algorithm. It is, for example, discussed in [82], [83] or [121]. Here the most important parameters are summarised and the strategy for their setting is presented.

An adaptive algorithm has been developed for the steering of the parameters in the back-propagation algorithm: The learning rate η and the momentum μ in the update rule

$$\Delta w(k) = -\eta \frac{\partial \text{Cost}}{\partial w} + \mu \Delta w(k-1) - \eta \delta w(k) \text{ with } w \in \{\tilde{w}_i, \tilde{b}, w_{ij}, b_i\} \tag{B.1}$$

are both initialised to 0.8 and decreased down to 10^{-4} by multiplying with 0.5 after each group of training epochs. One training epoch uses all training events once and one group of training epochs ends if the cost measured on the training set either rises or has stabilised after 50, 100, 150 or 200 training epochs. A new group starts anyway if 250 training epochs have been reached. The training is stopped before η and μ have reached 10^{-4} if overtraining has been detected ("early stopping"). This is done by keeping track of the changes of the errors on the training set and on the selection set. The selection set is used here to have an error measurement independent of the training set, the test set is not used during the training to have the independent performance measurement. Early stopping ensures that complex networks which would show clear overtraining if trained too long can still show a very good generalisation. The described procedure shows a quicker and more stable descent to the minimum cost compared to the standard of fixed learning parameters and compared to a prefixed decrease of the learning parameters within a given number of training epochs.

The parameter δ in the update rule above is called *weight decay* and results directly from a penalisation of large weight vectors. As mentioned in section 5.4.2 this penalisation is a common regularisation procedure. The other regularisation parameter is the number of weights in the network. Although it is in principle possible to build complex networks with many feed forward layers we use always only one hidden layer with a varying number of neurons. It was proven theoretically [81] that one hidden layer with a sufficient number of neurons is able to model any continuous output function.

The only free parameters which remain with the described procedure are the number of hidden neurons, the seed for the random initialisation of the weights and the weight

decay[1]. The training of multiple neural networks with different parameter settings is then easily done by calling one program which performs the trainings for a given range of hidden neurons, for a given number of different random initialisations and for a given set of weight decay parameters (tried out in every combination).

Typical values are 4 to 12 hidden neurons for many inputs or few training events, 20 to 40 hidden neurons for few inputs and enough training events (order of a few thousand). Typically two or three random initialisations are tried out and the weight decay parameter is varied in powers of 10 from 10^{-10} to 10^{-5}.

B.3.2 Parameters for Support Vector Machines

The Gaussian kernel for the support vector machine was introduced in section 5.4.3. It is modified for the framework of this thesis to allow scaling factors for each input individually:

$$K(\vec{x}, \vec{y}) = \exp\left(-\sum \gamma_i (x_i - y_i)^2\right). \tag{B.2}$$

A set of scaling factors is given to a program which then performs trainings in which each factor is tried out for each input in every combination. This "grid" approach is only suitable for sufficiently few inputs, since the number of combinations is k^n where n is the number of inputs and k is the number of factors which are tried out.

A second parameter which is varied by this program in a given region and with a given step-size is ν. ν determines the fraction of events which are taken as support vectors and is directly connected to the parameter C described in section 5.4.3. It can therefore be used to adjust the overtraining behaviour.

In total, every ν is automatically tried out with every combination of scaling factors for the inputs. Typical values are 5% to 30% for ν and scaling factors in different powers of 10 between 10^{-2} to 10^{2}.

B.3.3 Parameters for Random Forests

As discussed in section 5.11, the random forest method applies the bagging strategy to a slightly modified version of the CART decision tree algorithm. A very fundamental change in the behaviour of the decision tree CART is given by the variation of its splitting rule. Two different splitting rules have been implemented in the framework used in this thesis: The standard based on the Gini index (compare section 5.2) and an alternative especially for the regression case minimising the sum of the variances of the target values which remain in the two branches after the cut.

The other parameters do not really steer algorithmic behaviour but change some numbers like for the learning methods discussed above. The first of them steers the randomness of the splitting rule: In usual decision trees the search for the best pair of input and cut-value is exhaustive but for the random forest only a certain number of inputs is searched for a good split. The free parameter determines how many times an input is chosen randomly to be searched for the best split. A second parameter steers the stopping behaviour: An absolute number of events can be given below which a branch is not split any more. A third parameter determines the number of trees which should be grown, i.e. the number of

[1] The values for start and end of the η and μ decrease as well as the number of epochs per group can be changed but they are usually used with their default values.

basic decision trees produced by the bagging strategy. Finally, a last parameter is simply an initialisation for the random number generator.

All these parameters are varied within a given region and with a given step-size. Each combination of the different parameter values is automatically tried out by the managing program. Typical values for the number of inputs which are searched for the best split are 1 to the total number of inputs. The number of minimum events per leaf is typically 1 to a few tens. The number of trees which should be grown is usually in the order of 50 to 100.

Bibliography

[1] P. Schmüser. The electron proton colliding beam facility HERA. *Nuclear Instruments and Methods*, A 235:201–208, 1984.

[2] I. Abt et al. The H1 detector at HERA. *Nuclear Instruments and Methods*, A 386:310–347, 1997.

[3] J. Bürger et al. The central jet chamber of the H1 experiment. *Nuclear Instruments and Methods*, A 279:217–222, 1989.

[4] P. Marage et al. Construction of a cylidrical MWPC for the central tracking detector of H1. *Nucl. Phys. Proc. Suppl.*, 16:518, 1990.

[5] B. Andrieu H1 Calorimeter Group et al. The H1 Liquid Argon Calorimeter System . *Nuclear Instruments and Methods*, A 336:460–498, 1993.

[6] R.-D. Appuhn H1 Spacal Group et al. The H1 lead/scintillating-fibre calorimeter . *Nuclear Instruments and Methods*, A 386:397–408, 1997.

[7] A. Ringwald and F. Schrempp. *Phys. Lett.*, B 503:331, 2001.

[8] B. Koblitz. *Search for Instanton-Induced Processes with the H1 Detector Deep-Inelastic Electron-Proton Collisions at HERA*. PhD thesis, FB Physik, Univ. Hamburg, June 2002.

[9] A. Ringwald and F. Schrempp. *Comp. Phys. Comm.*, 132:267, 2000.

[10] H. Jung. *Comp. Phys. Comm.*, 86:147, 1995.

[11] L. Lönnblad. *Comp. Phys. Comm.*, 71:15, 1992.

[12] F. Halzen and A.D. Martin. *Quarks and Leptons: An Introductory Course in Modern Particle Physics*. John Wiley & sons, 1984.

[13] M. Worek. *Spin correlations of the heavy flavours as signal for Higgs bosons*. PhD thesis, Institute of Physics, University of Silesia, June 2003.

[14] T. Behnke et al. *TESLA Technical Design Report Part IV: A Detector for TESLA*. DESY-01-011.

[15] M. Pohl and H.J. Schreiber. *SIMDET - Version 4: A parametric Monte Carlo for a TESLA detector*. hep-ex/0206009.

[16] G.R. Bower et al. *Phys. Lett.*, B543:227–234, 2002. hep-ph/0204292.

[17] J. Schelten et al. Recent developments in X-ray and neutron small-angle scattering instrumentation and data analysis. *J. Appl. Cryst.*, 11:297–324, 1978.

[18] W. Gläser et al. The new neutron source FRM II. *Physica B*, 276–278:30–32, 2000.

[19] H.O. Anger. Scintillation camera. *Rev. Sci. Instr.*, 29:27–33, 1958.

[20] G. Kemmerling et al. Performance measurements of a new large area neutron scintillation detector system. *IEEE Transactions on Nuclear Science*, 51(3):1098–1102, 2004.

[21] J. Barrio et al. The MAGIC telescope. Technical report, 1998.

[22] R. Hartman et al. The third EGRET catalog of high-energy gamma-ray sources. *The Astrophysical Journal Supplement Series*, 123:79–202, 1999.

[23] A. Hillas. Cherenkov light images of EAS produced by primary gamma. *Proceedings of 19th International Cosmic Ray Conference*, 3:445–448, 1985.

[24] D. Paneque. *The MAGIC Telescope: development of new technologies and first abservations*. PhD thesis, Fakultät für Physik der Technischen Universität München, August 2004.

[25] Li and Ma. Analysis methods for results in gamma-ray astronomy. *The Astrophysical Journal*, pages 317–324, 1983.

[26] A.N. Parmar. XEUS - the X-ray evolving universe spectroscopy mission. *X-Ray and Gamma-Ray Telesopes and Instruments for Astronomy*, 4851:304, 2003.

[27] F. Jansen et al. XMM Newton observatory. *Astronomy and Astrophysics*, 365(1):L1–L6, 2001.

[28] P. Holl et al. Active pixel matrix for X-ray satellite missions. *IEEE Transactions on Nuclear Science*, 47(4):1421–1425, 2000.

[29] L. Strüder et al. The european photon imaging camera on XMM-Newton: The pn-CCD camera. *Astronomy and Astrophysics*, 365(1):L18–L26, 2001.

[30] P. Predehl. ROSITA - scientific goal and mission concept. *X-Ray and Gamma-Ray Telesopes and Instruments for Astronomy*, 4851:314, 2003.

[31] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach SE*. Englewood Cliffs, New York, 1995.

[32] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[33] L. Fu. *Neural Networks in Artificial Intelligence*. McGraw-Hill, New York, 1994.

[34] J. Hertz et al. *Introduction to the Theory of Neural Computation*. Addison-Wesley, New York, 1991.

[35] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, San Francisco, 1996.

[36] B. Natarjan. *Machine Learning: A Theoretical Approach.* Morgan Kaufmann, San Francisco, 1991.

[37] S. Weiss et al. *Computer Systems that Learn.* Morgan Kaufmann, San Francisco, 1991.

[38] N.J. Nilsson. Introduction to machine learning, 1996.

[39] C.-W. Hsu et al. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

[40] I.T. Jollife. *Principal Component Analysis.* Springer, New York, 1986.

[41] B. Laforge et al. *Elements of Statistical Methods in High Energy Physics Analyses.* Service de Physique des Particules, DAPNIA, CEA Saclay. H1 08/97-528.

[42] J. Dichtl. Neuronale Trigger für Heavy Quarkonium-Produktion bei HERA. Diplomarbeit an der Fakultät für Physik der Ludwig-Maximilians-Universität München, August 1999.

[43] R. Srikant and R. Agrawal. Mining generalized association rules. *Future Generation Computer Systems*, 13(2–3):161–180, 1997.

[44] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a database perspective. *IEEE Trans. On Knowledge And Data Engineering*, 8:866–883, 1996.

[45] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In Jorgeesh Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, pages 144–155, Los Altos, CA 94022, USA, 1994. Morgan Kaufmann Publishers.

[46] D. Wolpert. The relationship between PAC, the Statistical Physics framework, the Bayesian framework and the VC framework.

[47] A. Feelders et al. On the statistical comparison of inductive learning methods. *Learning from Data: AI and Statistics V. Edited by D. Fisher and H.J. Lenz (=Lecture notes in statistics 112)*, 112:271–279, 1996.

[48] H.-O. Georgii. *Stochastik: Einführung in die Wahrscheinlichkeitstheorie und Statistik.* de Gruyter, Berlin, 2002.

[49] O.J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[50] T. Hastie et al. *The Elements of Statistical Learning.* Springer, New York, 2001.

[51] T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[52] D. Michie et al. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994. The Statlog Project.

[53] J. Berger. *Statistical decision theory and Bayesian analysis.* Springer Verlag, 1985.

[54] T. Loredo. From Laplace to supernova 1987a: Bayesian inference in astrophysics. In P. Fougere, editor, *Maximum Entropy and Bayesian Methods*, pages 81–142.

[55] D. Haussler et al. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. *Machine Learning*, 14:83–113, 1994.

[56] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[57] D. Haussler. *The probably approximately correct (PAC) and other learning models.* Kluwer, 1994.

[58] D. Haussler. Probably approximately correct learning. *Proc. Eighth Nat. Conf. on AI*, pages 1101–1108, 1990.

[59] A. Blumer et al. Learnability and Vapnik-Chervonenkis dimensions. *Journal of the ACM*, 36:929–965, 1989.

[60] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13–30, 1963.

[61] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[62] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[63] P. Domingos. The Role of Occam's Razor in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999.

[64] P.M.B. Vitányi et al. Minimum Description Length Induction, Bayesianism, and Kolmogorov Complexity. *IEEE Transactions on Information Theory*, 46(2):446–464, 2000.

[65] J. Ross Quinlan. MDL and categorial theories (continued). In *International Conference on Machine Learning*, pages 464–470, 1995.

[66] L. Breiman et al. *Classification and Regression Trees.* Chapman & Hall, New York, 1984.

[67] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[68] W. Müller et al. The decision tree algorithm CAL5 based on a statistical approach to its splitting algorithm. In G. Nakhaeizadeh and C.C. Taylor, editors, *Machine Learning and Statistics, The Interface*, pages 45–65. John Wiley & Sons, Inc., 1997.

[69] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[70] Y. Lamdan and H. Wolfson. Geometric hashing: a general and efficient model-based recognition scheme. In *Proceedings ICCV*, volume 88, pages 238–249, 1988.

[71] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat.*, 33:1065–1076, 1962.

[72] T. Carli et al. A multi-variate discrimination technique based on range-searching. *Nuclear Instruments and Methods*, A 501:576–588, 2003.

[73] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[74] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[75] T. Hastie, R. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89(428):1255–1270, 1994.

[76] P. McCullagh and J.A. Nelder. *Generalized Linear Models.* London: Chapman and Hall, 1989.

[77] C. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press Oxford, 1995.

[78] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley New York, 1973.

[79] F. Rosenblatt. The perceptron: A probabilistic model for information storage nad organization in the brain. *Psych. Rev.*, 65:386, 1958.

[80] M. Minsky and S. Papert. *Perceptrons.* MIT Press Cambridge Massachusets, 1969.

[81] R. Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *IEEE, First Annual Int. Conf. on Neural Networks*, pages paper III–11, 1987.

[82] D.E. Rumelhart et al. Learning representations by back-propagating errors. *Nature*, 323:533, 1986.

[83] P.J. Werbos. Backpropagation through time: What it is and how to do it. In *IEEE Proceedings*, volume 78, pages 1550–1560, 1990.

[84] J.-C. Prévotet. *Etude des systèmes électroniques pour les réseaux connexionnistes appliqués à l'instrumentation en temps réel.* PhD thesis, Thèse de Doctorat de l'Université P. et M. Curie, 2002.

[85] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards*, 49:409–436, 1952.

[86] L. Jain et al. Neural network training using genetic algorithms. *Machine Perception and Artificial Intelligence*, 26, 1996.

[87] R.G. Parekh et al. Constructive Neural Network Learning Algorithms for Multi-Category Pattern Classification. Technical report, Department of Computer Science, Iowa State University, Ames, Iowa, 1995.

[88] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery Nuclear*, 2(2):121–167, 1998.

[89] B. Schölkopf et al. Introduction to support vector learning. In *Advances in Kernel Methods: SVM*, pages 1–15. MITPress, Cambridge, MA, 1998.

[90] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, Inc., 2nd edition, 1987.

[91] T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[92] C.J.C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 375. The MIT Press, 1997.

[93] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.

[94] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[95] M. Skurichina and R.P.W. Duin. Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis and Applications*, 5:121–135, 2002.

[96] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[97] R. Brun et al. *ROOT, An Object-Oriented Data Analysis Framework*. http://root.cern.ch.

[98] *http://stat-www.berkeley.edu/users/breiman/rf.html*.

[99] *http://www.cse.unsw.edu.au/~quinlan/*.

[100] *http://www.csie.ntu.edu.tw/~cjlin/libsvm*.

[101] *http://ki.cs.tu-berlin.de/software/DIPOL_CAL5.html*.

[102] L. Janauschek. *Elastic Photoproduction of J/psi Vector Mesons at high Photon-Proton Centre-of-Mass Energy at the H1 Experiment at HERA*. PhD thesis, LMU München, 2005.

[103] F. Salvaire. private communications.

[104] L. Favart. private communications.

[105] R. Placakyte. private communications.

[106] R. Lopez. private communications.

[107] H. Lueders. private communications.

[108] M. Göttlich. private communications.

[109] S. Schätzel. private communications.

[110] K. Desch et al. Measuring the higgs boson parity at a linear collider. *Eur. Phys. J. C*, 29:491–496, 2003.

[111] K. Desch et al. Probing the CP nature of the Higgs boson at linear colliders with tau spin correlations; the case of mixed scalar-pseudoscalar couplings. *Physics Letters B*, 579, 2004.

[112] T. Bretz et al. The MAGIC Analysis and Reconstruction Software. In *Proceedings of the 28th International Cosmic Ray Conference*, volume 5, pages 2947–2950, 2003.

[113] N. Tonello. private communications.

[114] R. Wagner. private communications.

[115] J. Englhauser. private communications.

[116] P. Holl. private communications.

[117] J. Ulrici. *Bildgebung mit DEPFET*. PhD thesis, Universität Bonn, 2003.

[118] *Proceedings on the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research, Moscow*, 2002.

[119] *Proceedings on the IX International Workshop on Advanced Computing and Analysis Techniques in Physics Research, Tsukuba*, 2003.

[120] G. Kemmerling et al. A new two-dimensional scintillation detector system for small angle neutron scattering experiments. *IEEE Transactions on Nuclear Science*, 48(4):1114, 2001.

[121] J. Möck. *Untersuchung diffraktiver J/ψ-Ereignisse im H1-Experiment bei HERA und Entwicklung neuronaler Triggeralgorithmen*. PhD thesis, Technische Universität München, 1997.

[122] E. Miyata et al. Application of the mesh experiment for the back-illuminated CCD: I. experiment and the charge cloud shape. *J. Appl. Phys.*, 41:5827–5834, 2002.

Index

This index is intended to provide a quick guide to explanations and examples of basic concepts and notions of statistical learning methods and their application in high energy and astrophysics. Links to definitions and basic explanations have normal style, links to *examples* and *applications* are italic.

Symbols

0-1 Loss 73

A

Activation of a Neuron 44, 90
Active Learning 44
Artificial Intelligence . . . 41, *138, 139*, 218
Automation of Stat. Learn. Meth. . . . 111

B

Back-Propagation Algorithm 92, 231
Bagging 96
Bayes Theorem 73, 87
Bayesian Learning 73
Bias and Variance 59, 72
Bias of Learning Methods 77, 81, 108
Boosting 97
Bootstrapping 55

C

Classification 45
Clustering 43, 47
Comparison of Hypotheses 68, *128*
Comparison of Stat. Learn. Meth. 67, *146*, 218
Control of Stat. Learn. Meth. . . . 53, 112, *118*, 215
Correlation of Input and Target . 43, *124*, *143*
Cost Matrices 57
Cross-Validation 56
Curse of Dimensionality 43, *140*
Cuts (simple Classifier) 47, *128*, *154*

D

Data Mining 65
Decision Boundary 45, 106
Decision Trees 82, *140*
Density Estimation . . . . *see* Local Density Estimators

E

Early Stopping 231
Efficiency of a Classifier 57
Enrichment of Rare Event-types . 48, *151*
Error Measurement 71

F

Features . . . . . *see* Inputs
Feed-Forward . . . . . *see* Neural Networks
Fit (simple Regression) 49, *187*

G

Generalisation . . . . . *see* Regularisation
Gini Index 82

H

Handling of Stat. Learn. Meth. 217
Hardware for Stat. Lern. Meth. . . . . *see* Online Application
Hidden Neurons 44, 91, 231
Hyperplane as Decision Boundary 89

I

IID (Indep. Ident. Distributed) 72
Information Gain 82
Inputs for Stat. Learn. Meth. 42, 46, *143*
Instance-Based Methods 82

K

K-Nearest-Neighbours Search 84, *146*
Kernel Methods 85
Kernelisation 95
Knowledge Integration 50

L

Linear Discriminant Analysis 89, *128*
Local Density Estimators 84
Loss-Function 59, 72

M

MAP (Maximum A Posteriori) 73
Maximum Likelihood .... *see* Naive Bayes
Maximum Margin Classifier 80, 93
MDL (Minimum Description Length) . 79
Meta Learning 95
Model Assessment ..... *see* Performance Evaluation
Model Selection ..... *see* Parameter Optimisation
Model-Based Methods 82
Multivariate Classification 47

N

Naive Bayes 87, *128*, *171*
Nearest Neighbours ..... *see* K-Nearest-Neighbours Search
Neural Networks 90, *117*, *185*
Neuron 90
No Free Lunch Theorem 67, 77
Normalisation of Events 46, *198*

O

Occam's Razor 79
Off-Training-Set Error 77
Offline Application 45
Online Application 45, *117*, *170*, *223*
Outliers in a Dataset 66
Over-fitting ..... *see* Overtraining
Overtraining 53, *140*, *147*, *185*

P

PAC (Probably Approx. Correct) 74
Parameter Optimisation 54, *147*
Parzen's Window 85
Perceptron 90
Performance Evaluation 55, 112, 214
Pre-Scaling instead of Classification .. 57, *142*
Preprocessing 46, 109, *227*
Principal Component Analysis 47
Propagation of Uncertainties ..... *see* Systematic Uncertainties
Proximity of Event-Classes 65
Pruning of a Decision Tree 83
Purification ..... *see* Offline Application

R

Radial Basis Function Neural Network 44
Random Forests 97, *185*
Random Subspace Method 97
Range Search 86, *156*
Regression 45, *170*, *187*, *203*
Regularisation 53, 79, *185*
Reinforcement Learning 43
Rejection of a Classifier 57
Relevance of an Input 65, *143*

S

Selection Set 54, *147*
Separation Power 58, *154*
Shattering of a Dataset 75
Sigmoid Function 90
Simulation ..... *see* Training Data
SRM (Structural Risk Minimisation) . 80, 94
Stacking 96
Statistical Uncertainties 60, *126*, *135*, *201*, *209*
Supervised Learning 43
Support Vector Machines 93, *128*
Symmetry of Events 46
Systematic Uncertainties ... 63, *126*, *135*, *201*, *209*

T

Target Value 43, *187*
Teacher ..... *see* Supervised Learning
Test Set 54, *147*
Time Restriction . *see* Online Application
Train-Test Arrangement 56
Training Data 52, *117*
Training of Stat. Learn. Meth. 44
Training Set 54, *140*
Trigger ..... *see* Online Application

U

Uncertainties of Stat. Learn. Meth. . . . 59
Univariate Classification 47
Unsupervised Learning 43

V

Variance and Bias 59, 72
VC (Vapnik-Chervonenkis) 75

W

Weight Decay 231
Weight of Events 43

List of Figures

2.1 The HERA collider . . . 6
2.2 HERA currents . . . 7
2.3 The H1 experiment . . . 8
2.4 The H1 Trigger System . . . 10
2.5 Perturbative approximations of deeply virtual Compton scattering . . . 14
2.6 DVCS and competing background . . . 15
2.7 Neutral current vs. charged current interactions . . . 16
2.8 Charged current interaction and competing background . . . 17
2.9 J/ψ production . . . 18
2.10 Heavy quark production . . . 19
2.11 Feynman-like diagram of an instanton-induced process in DIS . . . 19
2.12 Energy depositions in the calorimeter for an instanton-induced event . . . 20
2.13 Main Higgs boson production mechanisms in an e^+e^- collider . . . 21
2.14 Acoplanarity ϕ^* distributions . . . 22
2.15 Propagation of spin correlations . . . 22
2.16 Definition of acoplanarity φ^* . . . 23
2.17 Acoplanarity φ^* distributions . . . 23
2.18 Cross sectional view of the detector . . . 24
2.19 Projection of a light cone from a neutron capture event . . . 24
2.20 Structure of the neutron detector . . . 25
2.21 The MAGIC Telescope . . . 26
2.22 MAGIC flux sensitivity . . . 27
2.23 Sources detected by EGRET . . . 27
2.24 Sources detected by Cherenkov telescopes . . . 28
2.25 Development of extensive air showers . . . 29
2.26 Development of Cherenkov light . . . 30
2.27 Simulated air showers for photon and proton . . . 30
2.28 The principle of Imaging Air Cherenkov Telescopes . . . 32
2.29 Standard *Hillas* analysis of the shower image light distribution . . . 33
2.30 Image cleaning at different energies . . . 33
2.31 α-plots before and after γ-hadron separation . . . 34
2.32 Dependence of `size` on the energy of the primary particle . . . 35
2.33 The XEUS satellite . . . 36
2.34 Components of the XEUS satellite . . . 36
2.35 Schematic cross section of a fully depleted *pn*-CCD . . . 37
2.36 Patterns which can be generated by single photons . . . 38
2.37 Pileups from pattern pileup to charge pileup . . . 38

2.38 Resolution improvements by using the distribution of the charge 39
2.39 The mesh-experiment 40

3.1 Training and Evaluation 44
3.2 Structure of a Radial Basis Function Neural Network 44
3.3 A simple two-dimensional example for a classification problem 45
3.4 A simple one-dimensional example for a regression problem. 46
3.5 The charge splitting in a pixel-detector 47
3.6 The advantage of multidimensional cuts 48
3.7 A sequence of cuts 48
3.8 A three step cut process with the instanton dataset 49
3.9 A fitting example 50
3.10 Energy estimation in the MAGIC detector 51
3.11 The mesh-experiment 53
3.12 Overtraining in function fitting 54
3.13 Training set – selection set – test set 55
3.14 Output histograms 57
3.15 Efficiency vs. rejection graph 58
3.16 Correlation and mean and variance in bins of the target y 60
3.17 Comparison of the three estimates for the statistical uncertainty 62
3.18 An example for systematic uncertainties 65
3.19 A toy dataset with two classes 66
3.20 A toy dataset with three classes 66
3.21 An example for outliers 67

4.1 VC-Dimension of a linear decision in two dimensions 76
4.2 No-Free-Lunch theorem vs. local density estimation 78
4.3 Structural Risk Minimisation 80

5.1 An example of a decision tree 83
5.2 An example for a 5-nearest-neighbours evaluation with result $\frac{3}{5}$. 84
5.3 The range search method 87
5.4 The naive Bayes method 88
5.5 A Neuron and the soft threshold function 91
5.6 Transition region 91
5.7 Architecture of a feed forward neural network with one hidden layer. . . . 92
5.8 The maximum margin classifier 93
5.9 Support vector classification 95
5.10 The stacking scheme for a classification problem. 96
5.11 The bagging scheme for a classification problem. 96
5.12 The random subspace method for a classification problem 97
5.13 The boosting scheme for a classification problem. 98
5.14 The test set of the toy example 'Hole'. 99
5.15 'Hole': Output distributions 99
5.16 'Hole': Naive Bayes training result 100
5.17 'Hole': Neural network training result 100
5.18 'Hole': Support vector machine training result 101
5.19 'Hole': Random forest training result 101

5.20 The test set of the toy example 'Rings'. 102
5.21 'Rings': Output distributions 102
5.22 'Rings': Naive Bayes training result 103
5.23 'Rings': Neural network training result 103
5.24 'Rings': Support vector machine training result 104
5.25 'Rings': Random forest training result 104
5.26 The test set of the toy example 'Gaussians'. 105
5.27 'Gaussians': Output distributions 105
5.28 'Gaussians': Naive Bayes training result 106
5.29 'Gaussians': Neural network training result 106
5.30 'Gaussians': Support vector machine training result 107
5.31 'Gaussians': Random forest training result 107

7.1 Efficiency studies with orthogonal triggers 118
7.2 DVCS compared to J/ψ 120
7.3 Distributions of DVCS vs. background 121
7.4 Distributions of DVCS vs. background 122
7.5 Distributions of DVCS vs. background 123
7.6 Output distributions and efficiency graph for the DVCS network 125
7.7 Monitoring of the DVCS-network: The background rejection 126
7.8 Efficiency check for the DVCS-network 127
7.9 Comparison of fast classification methods on the DVCS dataset 130
7.10 Track candidates in the backward region 132
7.11 Distributions of pseudo-CC vs. background 133
7.12 Distributions of pseudo-CC vs. background 134
7.13 Output distributions and efficiency graph for the CC network 136
7.14 Monitoring of the CC network: The background rejection 137
7.15 Efficiency check for real charged current events 137
7.16 Rejected charged current event 138
7.17 Event from the charged current selection which might be rejected 139
7.18 Output distributions for the two $J/\psi \to e^+e^-$ selections 140
7.19 Dependence of the misclassification rate on the training set size 141
7.20 A preliminary J/ψ mass peak 143
7.21 Evaluation of the parameter optimisation process 148
7.22 Input distributions of the instanton dataset in one-dimensional projections 152
7.23 Input distributions of the instanton dataset in two-dimensional projections 153
7.24 Significances in dependence of the instanton efficiency (range search) . . . 158
7.25 Significances in dependence of the instanton efficiency (neural network) . . 161
7.26 Direct discrimination of scalar and pseudo-scalar events 165
7.27 Distribution of significances 166
7.28 Acoplanarity histogram with fitted cosine 166
7.29 Dataset of simulated neutron events with a flat illumination 170
7.30 Standard reconstruction of the calibration dataset 171
7.31 Event structure in the neutron detector 172
7.32 Reconstruction of the simulated dataset with statistical learning methods . 173
7.33 Reconstruction of the calibration dataset with statistical learning methods 174
7.34 Reconstruction of the calibration dataset with statistical learning methods 175

7.35 Mean and variance in the projections onto the y-axis 175
7.36 Supercuts and alpha principle 178
7.37 Significance and excess events depending on the cut in the output 179
7.38 Different significances in the alpha plot (Markarian 421, 15.02.2004) 180
7.39 Significance and excess events vs. cut in output for Crab Nebula 181
7.40 Crab results (optimised significance) 182
7.41 Crab results (identical background suppressions) 182
7.42 Significance and excess events vs. cut in output for 1ES1959 183
7.43 1ES1959 results (optimised significance) 184
7.44 1ES1959 results (identical background suppressions) 184
7.45 Flux measurement for 1ES1959 185
7.46 Monte Carlo efficiency vs. significance on real data 186
7.47 Different energy estimation induced by different cuts in `size` 188
7.48 Original and reconstructed energy spectra for different size cuts 188
7.49 Binning in true energy vs. binning in estimated energy 189
7.50 Resolutions obtained with the random forest method 190
7.51 Reconstructed spectra obtained with different training targets 191
7.52 Correlation between true and estimated energy 192
7.53 Two different input sets for the low size region 194
7.54 Pileups from pattern pileup to charge pileup 196
7.55 Allowed patterns for the "XMM" algorithm 196
7.56 The reconstructed white and power law spectrum used in the datasets . . . 197
7.57 Inputs for the neural network 197
7.58 Two different normalisation schemes 198
7.59 Single photons which are not recognised by the XMM algorithm 198
7.60 Patterns from pileups which are not recognised by the XMM algorithm . . 199
7.61 Pileup rejection of XMM algorithm and neural network 200
7.62 Definition of the relative incident position 203
7.63 CCOM and η-method 205
7.64 The photon energy determines the final charge cloud size 205
7.65 Probability for different splitting types 206
7.66 Resolution in a low noise scenario 207
7.67 Resolution in dependence of the photon energy 208
7.68 The correlation between true and reconstructed position 209
7.69 Comparison of resolution for three reconstruction methods 210
7.70 Reconstruction error from the reconstruction of experimental data (CCOM) 211
7.71 Reconstruction error from the reconstruction of experimental data (NN) . . 212

A.1 Block diagram of the analog and digital signal processing 224
A.2 Diagram of one L2NN trigger-box 224

B.1 Reconstructed spectra obtained with different preprocessors 229
B.2 The mesh-experiment 230

List of Tables

2.1 Most important quantities available for the level 2 neural network trigger . 13

3.1 Z-values to construct confidence intervals 68
3.2 Example for the comparison of three hypotheses. 69

7.1 Correlation coefficients of the DVCS dataset 123
7.2 The DVCS-dataset 124
7.3 Sources for systematic uncertainties for the DVCS dataset 128
7.4 Evaluation of systematic uncertainties for the DVCS dataset 129
7.5 Total uncertainties for the DVCS network 129
7.6 Statistical performance analysis with the DVCS dataset 131
7.7 The CC-datasets 135
7.8 Sources for systematic uncertainties for the CC dataset 137
7.9 Total uncertainties for the real charged current selection 137
7.10 Different sizes of training, selection and test sets 141
7.11 Assessment of single inputs and detector components 144
7.12 Successive removal of inputs 145
7.13 Efficiencies and rejections for D^* and dijet production 145
7.14 Comparison of learning methods on the D^* dataset 146
7.15 Datasets for the the instanton analysis 150
7.16 Sources for systematic uncertainties for the instanton analysis 151
7.17 Results for cut-based approach 155
7.18 Results for range search approach 157
7.19 Summary of new results for the instanton search 159
7.20 Classical results for the significance of the parity determination 167
7.21 Neural network results for the significance of the parity determination . . . 167
7.22 Random forest results for the significance of the parity determination . . . 167
7.23 Neural network results for the significance of the parity determination . . . 168
7.24 Random forest results for the significance of the parity determination . . . 168
7.25 Basic detector parameters used in the simulation 195
7.26 Total systematic uncertainties for the pileup rejection (XMM algorithm) . 202
7.27 Detailed systematic uncertainties for the pileup rejection (neural network) 202
7.28 Total systematic uncertainties for the pileup rejection (neural network) . . 202
7.29 Systematic uncertainties for the overall resolution 209

Danksagung

Mein besonderer Dank gilt Herrn Prof. Dr. Christian Kiesling, der mir die Möglichkeit dieser Doktorarbeit geboten hat. Die kontinuierliche Betreuung, sein Interesse an meiner Arbeit und anregende Diskussionen waren mir eine wertvolle Hilfe.

Herrn Prof. Dr. Paul Tavan danke ich herzlich für die Übernahme der Zweitkorrektur.

Ich bedanke mich besonders beim Forschungszentrum Jülich für die Finanzierung des Promotionsstudiums, bei Herrn Klaus Zwoll, Herrn Hubert Gorke und insbesondere bei Herrn Günther Kemmerling für den Austausch über den Neutronendetektor.

Allen Mitgliedern der H1-Gruppe, Ana Dubak, Ludger Janauschek, Ringailė Plačakytė und Biljana Vujičič, gilt mein Dank für die gute Zusammenarbeit und die sehr angenehme Arbeitsatmosphäre.

Bei Pratik Majumdar, Daniel Mazin, David Paneque, Nadia Tonello und besonders bei Wolfgang Wittek aus der MAGIC-Gruppe bedanke ich mich herzlich für die hilfreichen Gespräche und die gemeinsamen Studien zu den MAGIC-Daten.

Ich möchte mich herzlich bei den Mitarbeitern des MPI Halbleiterlabors, Jakob Englhauser, Peter Holl, Nils Kimmel, Norbert Meidinger und Lothar Strüder, bedanken für die guten Gespräche und gemeinsamen Studien zu den Pixeldetektoren.

Mein Dank gilt auch Małgorzata Worek für die erfolgreiche Zusammenarbeit bei der Higgs-Parity Analyse.

Für das geduldige und sorgfältige Korrekturlesen des Entwurfs dieser Arbeit bedanke ich mich bei Sonja Niedermaier ganz herzlich.

Schließlich danke ich meiner Frau Monika für ihre Geduld und beständige Unterstützung.