TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Machine-learning assisted track finding in the Silicon Vertex Detector of the Belle II experiment

ausgeführt am

Institut für Hochenergiephysik (HEPHY)
der Österreichischen Akademie der Wissenschaften (ÖAW)

unter der Leitung von

Univ.Doz. Dipl.-Ing. Dr.techn. Rudolf Frühwirth

eingereicht an der Technischen Universität Wien
Fakultät für Physik

durch

Thomas Madlener
Matrikelnummer 0926219
Schumanngasse 3/16
1180, Wien, Österreich

Wien, im Oktober 2015

Abstract

The Standard Model of particle physics (SM) is one of the most successful and most thoroughly tested theories of physics. However, it is not able to answer all questions satisfactorily. One of these questions concerns CP violation, which is, according to the Sakharov conditions for baryogenesis [1], necessary for the universe being able to exist in its current form.

An important contribution towards understanding CP violation in the SM has been made by the Belle experiment [2, 3], where CP violation was discovered in the B-meson system. The Belle II experiment, successor of the Belle experiment, plans to further investigate these phenomena. Another aim is the measurement of rare decay processes, made possible by a forty-fold increase of the instantaneous luminosity.

To handle the increased physics and background rate the detector has to be upgraded accordingly. As a part of the upgrade a silicon based Vertex Detector (VXD) is under construction. Aside from improving the vertex resolution the main purpose of the VXD is finding and reconstructing low momentum particle tracks. The goal of this thesis is to test the feasibility of employing machine learning techniques to improve the current track finding algorithm of the VXD.

The current algorithm relies on the division of the VXD into small sectors. Relations between these sectors can be utilized to apply different predefined cut-off filters which allow to classify combinations of hits into either signal or background. This allows to discard a large fraction of background hits without affecting the efficiency. However, finding the relations between the different sectors and tuning the different filters requires considerable computational resources.

Machine Learning has the prospect of generalizing from relatively small data sets. The approach pursued in this thesis is to exploit the generalization capabilities by exchanging a part of the filters together with their large number of different cut-off values by a small number of sophisticated, machine learned filters.

Due to a major redesign of the current implementation of the tracking algorithm only stand-alone tests for filters using combinations of three hits are possible. The thesis can thus be regarded as a preliminary study. Nevertheless, the results are promising and an implementation into the tracking algorithm is foreseen as the next step once the redesign is finished.

Kurzfassung

Das Standardmodell der Teilchenphysik (SM) ist eine der erfolgreichsten und meist getesteten Theorien der Physik. Trotzdem gibt es noch Fragen, die innerhalb des SM nicht zufriedenstellend beantwortet werden können. Dazu zählen Fragen zum Verständnis der CP Verletzung, die laut den Sacharov-Kriterien [1] nötig ist, damit das Universum in seiner jetzigen From überhaupt existieren kann.

Die Entdeckung der CP Verletzung im B-Meson System durch das Belle-Experiment [2, 3] hat dazu einen wichtigen Beitrag geliefert. Das Nachfolge Experiment, Belle II, soll diese Phänomene weiter untersuchen und auch die Untersuchung sehr seltener Zerfälle, dank einer um den Faktor 40 erhöhten Luminosität, ermöglichen.

Um die dadurch wesentlich größere Rate an Physik und Untergrund Ereignissen zu bewältigen, muss der Detektor entsprechend aufgerüstet werden. Im Zuge dieser Aufrüstung wird auch ein auf Siliziumsensoren basierender Vertex Detektor (VXD) gebaut. Neben der Verbesserung der Vertex-Auflösung ist die Hauptaufgabe des VXD das Auffinden und Rekonstruieren von Teilchenspuren mit niedrigem Impuls. Das Ziel der vorliegenden Arbeit ist die Anwendung von Techniken des maschinellen Lernens, um die Leistung des derzeitigen Algorithmus zur Spurfindung im VXD zu verbessern.

Der derzeitige Algorithmus beruht auf der Einteilung des gesamten VXD in kleine Sektoren. Beziehungen zwischen den Sektoren können genutzt werden, um verschiedene vordefinierte Filter anzuwenden, die es erlauben, Kombinationen von Messpunkten als Signal oder Untergrund zu klassifiziern. Dadurch kann ein großer Anteil des Untergrunds verworfen werden, ohne dabei die Effizienz zu beinträchtigen. Allerdings benötigt das Auffinden der Relationen zwischen den Sektoren und der Grenzwerte für die Filter beträchtliche Ressourcen.

Maschinelles Lernen bietet die Chance, aus kleinen Datensätzen zu generalisieren. Der hier verfolgte Ansatz nützt diese Generalisierungsfähigkeit aus, indem ein Teil der Filter mit ihren vielen verschiedenen Grenzwerten durch eine kleine Zahl von komplexen, maschinell erlernten Filtern ersetzt wird.

Da sich die derzeitige Implementierung des Algorithmus in einer größeren Umstrukturierungsphase befindet, konnten nur unabhängige Tests mit Kombinationen aus drei Messpunkten durchgeführt werden. Diese Arbeit kann daher als Machbarkeitsstudie gesehen werden. Allerdings sind die Ergebnisse vielversprechend, und eine Implementation in der Softwareumgebung von Belle II wird erfolgen, sobald die Umstrukturierung abgeschlossen ist.

Contents

1 Introduction

Particle physics today is mainly concerned with describing the fundamental particles of our universe and the interactions between them. The current state of particle physics, the Standard Model, has been strongly probed and no strong evidence of an effect or particle violating its predictions has been found. However, there are some phenomena that cannot be sufficiently explained by the Standard Model, and new theories or extensions capable of describing these effects have been developed.

These theories have been coined as *new physics* (NP) or *physics beyond the Standard Model* (BSM), and current high energy physics experiments strive to confirm or disprove such models. These experiments are situated at particle colliders that accelerate particles to the energies necessary to probe predicted effects or to discover new effects. The experiments at such facilities are detectors built around the interaction points, where the particles are brought to collision. The main purpose of these detectors is to measure and identify the particles emerging from physics processes occurring in the collisions. An integral part of this task is the measurement of the momenta and the origins of charged particle tracks which is commonly referred to as track and vertex reconstruction. Track reconstruction consists of two steps that can, however, not be clearly separated: track finding and track fitting. Track fitting is concerned with the actual estimation of the track parameters, whereas track finding is concerned with finding and grouping together the hits belonging to tracks. Track finding can also be seen as a classification process that on its most basic level divides hits into signal hits stemming from particle tracks and background hits that must not be used for track reconstruction.

With the rise of computational power in recent years machine learning has been used successfully in a wide area of applications: from pattern recognition (covering image and speech recognition) [4, 5] to prediction and forecasting of stock market or weather trends [6, 7] to classification tasks and physics analysis [8, 9]. In this thesis the possibilities of applying supervised machine learning techniques to track finding in the inner tracking system of the Belle II experiment are investigated. The inner tracking system is a crucial part of the Belle II experiment, and a sophisticated algorithm for track finding has been developed. The current approach is based on dividing the sensors into small sectors and then using relations between these sectors to filter hit combinations. This thesis describes an approach exploiting different machine learning approaches trying to improve the current algorithm.

The current track finder and the Belle II experiment in general are outlined in Chapter 2. Following, in Chapter 3, a brief description of the physics program

pursued at Belle II and a brief description of the Standard Model can be found together with the underlying physics principles of the detector. Chapter 4 provides an introduction to supervised machine learning and the techniques that have been used to obtain the results. A detailed description of the learned classifiers and the results of a preliminary study are reported in Chapter 5. Finally some conclusions and and outlook to the next steps to be taken can be found in Chapter 6.

2 The Belle II Experiment

The Belle II experiment, the successor of the Belle experiment which was shut down in 2010, is located at the SuperKEKB collider in Tsukuba, Japan. SuperKEKB is a major upgrade of the KEKB collider and will have an instantaneous luminosity that is forty times larger [10]. Belle II is designed to cope with the larger physics and background rates to be expected at SuperKEKB. The asymmetric electron-positron collider will be operated mainly at the $\Upsilon(4S)$ resonance. Commissioning is planned to start in 2016, and first data taking is planned for the year 2018 [11]. The aspects of the collider and the experiment that are relevant for this thesis will be described briefly in Secs. 2.1 and 2.2. Further information and a more detailed description can be found in [12].

The much higher data rate is not only a challenge for the detector hardware and the data acquisition system, but also for the software framework that is used for data processing. Due to the number and the extent of the modifications that would have been necessary to adapt the *Belle AnalysiS Framework* (BASF) of the Belle experiment, a complete rewrite from scratch was decided [13]. The new framework, called BASF2, is currently under development. It is written mainly in C++11 and the Python programming language. As the central part of the software used by Belle II it is employed for a variety of tasks including on-line and off-line data handling, Monte Carlo (MC) event generation, detector simulations, track reconstruction and physics analysis. A short introduction to the basic design and working principles of BASF2 can be found in Sec. 2.3.

2.1 SuperKEKB

SuperKEKB is an asymmetric electron–positron collider. The electrons in the high-energy ring (HER) are accelerated to 7 GeV before injection, the positrons in the low-energy ring (LER) to 4 GeV [12], resulting in a boost of the particles produced in a collision. The resulting center of mass energy is

$$E_{\mathrm{CMS}} = 2\sqrt{E_{\mathrm{HER}} E_{\mathrm{LER}}} = 10.58\,\mathrm{GeV} = m_{\Upsilon(4S)}. \tag{2.1}$$

To achieve the design instantaneous luminosity of $8 \cdot 10^{35}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ the so called *nano beam* scheme will be employed, along with a doubling of the beam currents in both rings compared to KEKB [10, 12]. The adaptions that are necessary for the nano beam scheme give rise to increased backgrounds and to a shortened beam

lifetime due to the Touschek effect (seeSec. 3.3.1). To mitigate the latter, the beam energy of the LER is larger than it was in KEKB [12]. As a result, the asymmetry of the beam energies is smaller, leading to a smaller Lorentz boost of $\beta\gamma = 0.28$, as compared to $\beta\gamma = 0.42$ for KEKB.

	KEKB		SuperKEKB	
	e^+	e^-	e^+	e^-
Beam energy (GeV)	3.5	8.0	4.0	7.0
Beam current (A)	1.19	1.64	3.6	2.61
Beam lifetime (min)	150	200	10	10
inst. Luminosity ($10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$)	2.11		80	
Vertical beta function β_y^* (mm)	5.9	5.9	0.41	0.27
Beam emittance (μm)	2100	300	10	20
Number of bunches	1293		2503	

Table 2.1: Comparison of different machine parameters of KEKB and SuperKEKB [12, 14]. The vertical beta function is taken at the interaction point. β_y^* and the beam emittance are measures for the spatial spread of the particles in a beam.

KEKB was and SuperKEKB is currently being installed in the underground tunnel of the former TRISTAN storage ring [15]. The collider has a circumference of about 3 km. There is a single interaction point (IP) at which the Belle II detector is operated. Both beams are accelerated to their full energies by a linear accelerator before they are injected into their respective rings. A schematic overview of SuperKEKB can be found in Fig. 2.1.

2.2 The Belle II Detector

The Belle detector is currently being upgraded to Belle II in order to cope with the increase of luminosity. The most important objectives of the upgrade program are the following: handling the increased physics and background rates, improved radiation hardness, and similar or better vertex resolution despite the smaller Lorentz boost compared to Belle [10].

Due to the asymmetric beam energies and the resulting Lorentz boost the particles produced in a collision and their decay products tend to fly in the direction of the electrons in the HER, which is called the *forward* region. Hence, the Belle II detector has a distinct forward direction and is designed slightly asymmetrically in order to capture as many particles as possible and to lower the material budget.

The detector is divided into several different subdetectors that are arranged radially symmetrically and cover the acceptance region $0° \leq \phi < 360°$ in azimuthal and $17° < \theta < 150°$ in polar angle (Fig. 2.2). The different subdetectors serve different purposes for tracking, vertexing and particle identification.

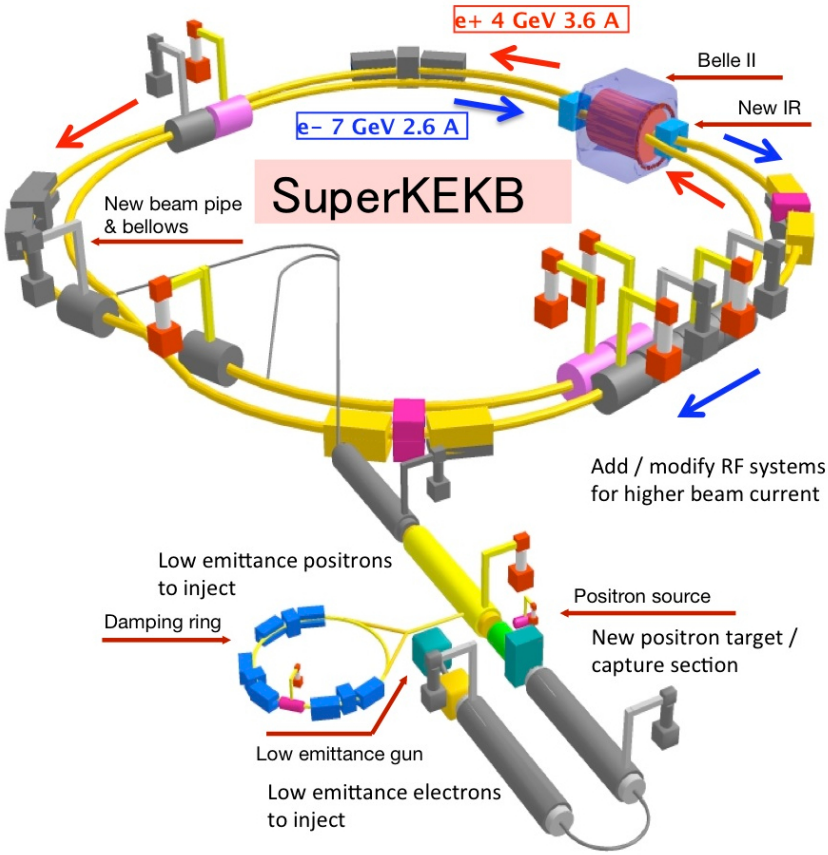


Figure 2.1: Schematic layout of the SuperKEKB [16]

The various subdetectors are described in more detail in the following subsections. The main tracking devices are the Central Drift Chamber (CDC) and the Vertex Detector (VXD). The latter consists of the Silicon Vertex Detector (SVD) and the Pixel Detector (PXD), which is responsible for achieving the best possible vertex resolution. Particle identification is done with the TOP and ARICH detectors, which are assisted by the ECL and the KLM (see below). The latter also serves as flux return for the almost perfectly homogeneous 1.5 T magnetic field in which the tracking detectors are immersed.

2.2.1 VerteX Detector - VXD

The VXD is a silicon based detector consisting of the PXD and the SVD. Its main purposes are increasing the vertex resolution and finding low momentum tracks that do not reach the CDC or other parts of the detector. To this end, a standalone track finding algorithm for tracks originating from the IP has been developed. This algorithm will be described briefly in section 2.4. Tracks that are found in the CDC

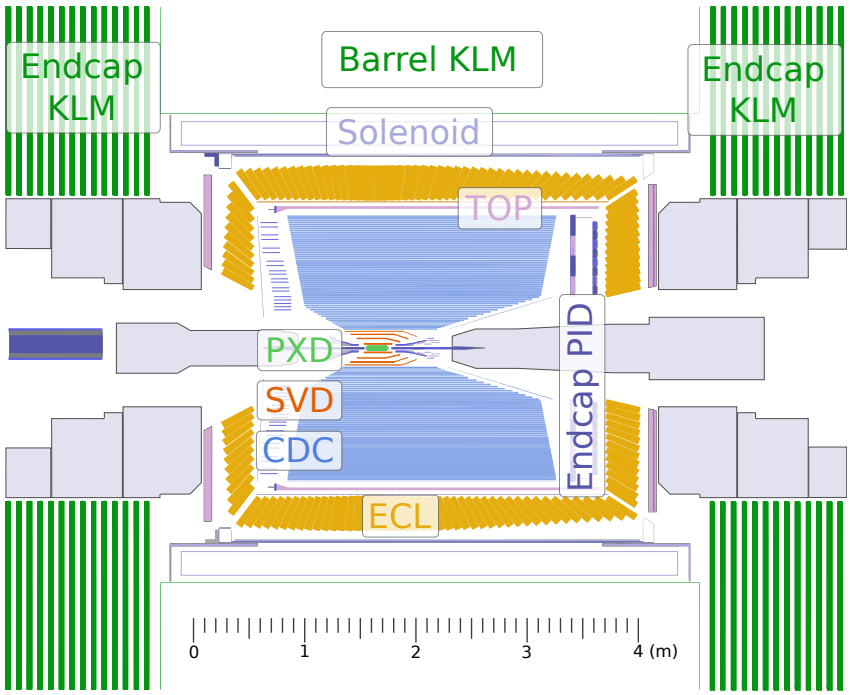


Figure 2.2: Cross section of the Belle II detector with differently colored subdetectors [17]

are matched with the VXD tracks in order to improve their position resolution at the production vertex.

The VXD is organized in six layers, two in the PXD and four in the SVD. Each layer consist of so called ladders. The ladders are arranged in a windmill layout to prevent dead regions and to increase the number of hits found per track (see Fig. 2.4).

PiXel Detector - PXD

The PXD consists of two layers of silicon pixel detectors built using DEpleted P-channel Field Effect Transistors (DEPFET). Placed directly outside the beam pipe at $r = 14\,\text{mm}$ and $r = 22\,\text{mm}$, their main purpose is to provide measurements that make a high resolution vertex reconstruction possible [12]. To minimize multiple scattering, the sensors are designed to be only about $75\,\mu\text{m}$ thick. The pixel size is roughly $50 \times 50\,\mu\text{m}^2$ and $75 \times 50\,\mu\text{m}^2$ [10, 12].

Due to the low power consumption of the DEPFETs, only the readout electronics, which is located outside the acceptance region of the detector, needs active cooling. The duration of a full readout cycle of the over eight million pixels organized in 1600 pixel rows is about $20\,\mu\text{s}$ for an entire frame [12]. The high background rates at the small distance from the IP lead to an estimated occupancy of about 1 %.

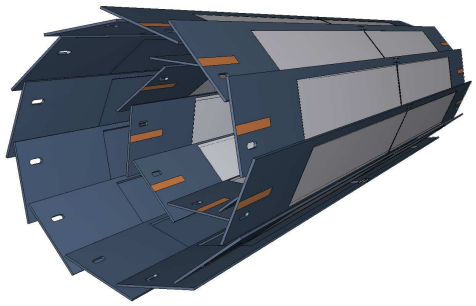


Figure 2.3: Schematic view of the arrangement of the PXD. The gray areas are the sensitive detector areas consisting of DEPFET pixels [12].

The worst case scenario that can be handled by the readout system is an occupancy of about 3 %.

In order to reduce the amount of data that has to be stored for each event only so called *regions of interest* (ROI) will be read out from the PXD. These ROIs are determined via an extrapolation of tracks found in the SVD and CDC by the High Level Trigger (HLT) [17].

Silicon Vertex Detector - SVD

The SVD is a 4-layer silicon strip detector located between the PXD and the CDC, with layer radii between 38 mm and 140 mm. The SVD consists of double-sided silicon strip detectors (DSSD). To cover the whole θ acceptance range and to lower material costs and budget, the SVD sensors are slanted with respect to the beam axis in the forward direction [12] (see Fig. 2.4).

At 20 ns the readout time is faster by a factor of 1000 compared to the PXD. The faster readout and the larger distance to the IP allows the SVD to discard more background. However, silicon strip detectors are prone to so-called *ghost hits*. If more than one particle is passing the sensor during one readout frame, the assignment of strip intersections to particle hits becomes ambiguous, leading to ghost hits (see Fig. 2.5). To resolve this ambiguity, information from other layers and from the PXD is used, as the probability of several ghost hits lining up to a track is low.

2.2.2 Central Drift Chamber - CDC

The CDC, surrounding the SVD, is the largest tracking detector of Belle II . It is a gaseous detector with an outer radius of $r = 1130$ mm, filled with a 50 % Helium, 50 % Ethane gas mixture. Over 55 000 sense and field wires are spanned between

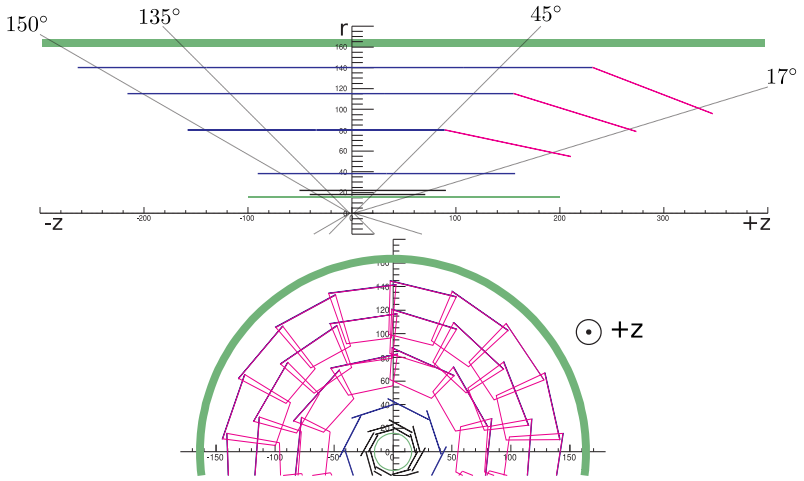


Figure 2.4: Geometric arrangement of the VXD into four layers of the SVD (blue barrel parts and slanted pink parts) and two layers of the PXD (black) as seen from the side (top) and from the forward direction (bottom). All dimensions except the angles in mm. Adapted from [12].

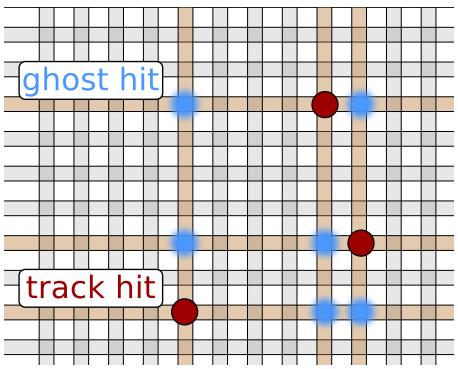


Figure 2.5: Formation of ghost hits on silicon strip detectors. Three particle hits activate strips of the sensor. All intersections of activated strips have to be considered as hit candidates resulting in ghost hits.

the forward and backward end plate inside the gas volume [12]. They are organized in wire cells where one sense wire is surrounded by field wires (see Fig. 2.6).

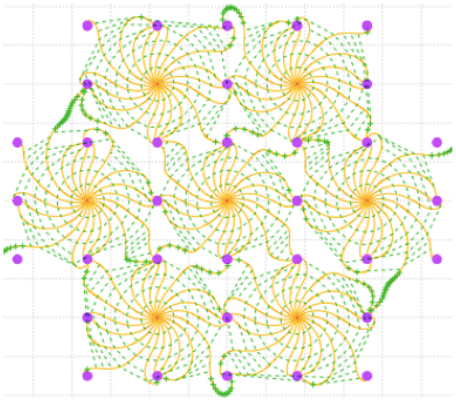


Figure 2.6: Organization of the field wires (purple) and the sense wires (orange) in the CDC together with the field lines (yellow) and isochores (green dashed) [18]

Charged particles passing through the gas volume of the CDC lose energy by ionizing gas molecules. The ions and electrons are separated by the electric field produced by the field wires and are driven to the sense wires. The number of the primary electrons is increased by gas amplification. The measured signal in the sense wire is proportional to the number of collected electrons and thus proportional to the energy loss of the incident particle in the wire cell.

The accelerations of the electrons towards the sense wires is compensated by collisions with the gas atoms. Hence, the propagation of the electrons can be described as a diffusion process with a nearly constant drift velocity. Thus, provided a reference time t_0, the time stamp of the wire signal can be used to calculate the so called *drift time*. As the electrons quickly reach the final drift velocity, the drift time can be used to calculate the distance between a particle's path and the sense wire, given a well-calibrated time-distance mapping (see Fig. 2.7). Although every single measurement has a rather large uncertainty in the order of $150\,\mu$m, the combination of many such measurements allows to achieve a spatial resolution which is orders of magnitude smaller than the size of the wire cells. In the CDC a typical track produces over 50 wire hits, allowing for a resolution in the x-y plane below $100\,\mu$m [12].

The wires are grouped into so called superlayers, where six layers (eight layers for the innermost superlayer) are grouped together. To allow for a measurement of the z-coordinate, the wires of some of these superlayers are slightly tilted with respect to the beam axis (see Fig. 2.8). These superlayers are called *stereo* superlayers, whereas superlayers in which the wire direction is parallel to the z-axis are called *axial* superlayers. The accuracy of the z-information that can be obtained by this setup depends on the stereo angle and is in the order of $1.3 - 2.2\,$mm [12], therefore

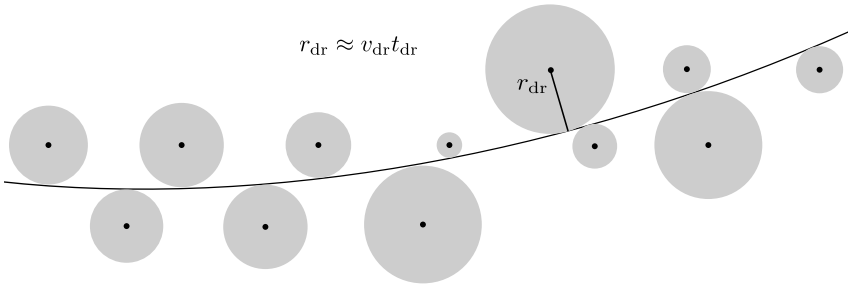


Figure 2.7: Illustration of a track in the CDC with drift lengths calculated from the drift time and the drift velocity shown as circles around the field wires.

the CDC needs to be assisted by the VXD for a precise measurement of the z-coordinate.

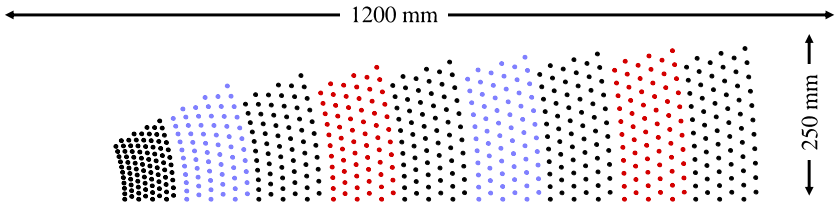


Figure 2.8: Cross section of the CDC wire organization with color coded information on the angle between the z-axis and the wires of a superlayer: axial superlayers parallel to the z-axis (black), stereo superlayers with positive (blue) and negative (red) stereo angles. Adapted from [12].

Besides reconstructing tracks and measuring their momenta the CDC is also used to gain information for particle identification via the specific energy loss of the particle inside the gas volume [19]. The fairly simple construction allows for fast readout electronics offering a low dead time. Furthermore the relatively low production costs make it possible to cover a wide radial distance, which improves the estimate of the transverse momentum p_T.

2.2.3 Particle Identification - PID

The particle identification system of Belle II is based on the Cherenkov effect (see Sec. 3.1.4). It uses different detectors for the barrel region and the forward end cap. One of the main goals is the discrimination of pions from kaons as well as the discrimination of other particles [12].

Time-of-Propagation Counter - TOP

The time-of-propagation (TOP) counter is situated immediately outside the CDC and is responsible for particle identification in the barrel region. It consists of a quartz radiator and combines the measured time of arrival with the Cherenkov angle θ_c to reconstruct a three dimensional Cherenkov image. The Cherenkov photons are reflected internally and measured by photo multipliers (PMT) at the end surface of the quartz bars (see Fig. 2.9). From the Cherenkov image the velocity of the particle can be calculated, which in combination with the momentum measurement allows to compute the likelihoods of different particle hypotheses.

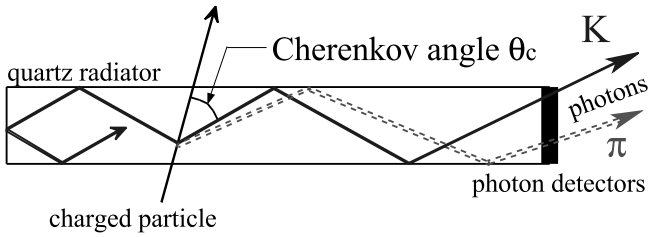


Figure 2.9: Schematic cross-section of a quartz radiator in the TOP counter showing the paths of the internally reflected photons created by different particles passing through the counter [12].

Aerogel Ring Imaging Cherenkov Detector - ARICH

Particle identification in the forward end cap is performed with a proximity focusing Aerogel Ring Imaging Cherenkov Detector (ARICH). Cherenkov photons are produced by passing particles in an aerogel radiator. An array of position sensitive photon detectors is placed behind an *expansion volume* to detect the rings that are formed by the Cherenkov photons [12].

To reduce the spread of the ring image the aerogel radiator is split up into two parts with different refractive indices (see Fig. 2.10). The Cherenkov angle for each detected photon is reconstructed using the particle trajectory provided by the CDC and the position of the photon. The velocity is estimated by combining the Cherenkov angles of all detected photons of a given track.

2.2.4 Electromagnetic Calorimeter - ECL

Enclosing the TOP counter and the ARICH, the ECL consists of 8736 scintillating crystals made of CsI(Tl) (caesium iodide doped with thallium), with a total weight of 43 tons. The main purpose is the energy and position measurement of photons and electrons, as well as the identification of electrons and K_L^0. The latter task is done in conjunction with the KLM [12].

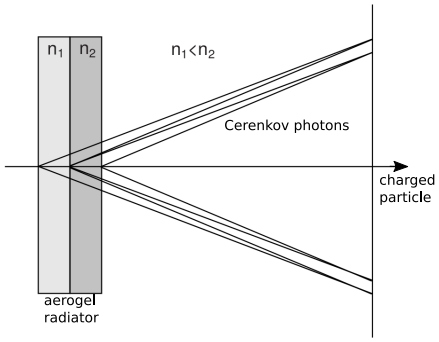


Figure 2.10: Schematic side view of the proximity focusing ARICH used for particle identification in the forward end cap of Belle II. Adapted from [12].

While electrons are absorbed in the ECL or the iron plates of the KLM, K_L^0 leave traces in both detectors. By associating charged tracks to clusters in the ECL and the KLM it is possible to discriminate between electrons and K_L^0. Clusters in the ECL that cannot be associated to a charged track stem from absorbed photons which cannot be detected by the tracking detectors.

The detection principle relies on the production of electromagnetic cascades by the incident particles. While electrons produce these cascades directly, photons produce them via secondary processes such as pair production or the Compton effect. The accompanying excitation of the scintillator material is dissipated through low energy photons which are detected and amplified by photo multipliers. The number of photons is directly dependent on the deposited energy. Hence, the PMT signal is a measure for the energy deposition in the ECL crystal. In order to achieve good energy resolution, a careful calibration of the device is crucial.

2.2.5 K-Long and Muon Detector - KLM

Beyond the ECL and the solenoid the KLM is responsible for the identification of muons and the detection of K_L^0. In the barrel region the KLM is made of a sandwich structure of iron plates, serving as the flux return yoke, and glass electrode resistive plate chambers (RPC). In the end caps the KLM is instrumented with scintillator strips, as the dead-time of RPCs is too long to allow an efficient operation in regions with high background [12].

The discrimination between muons and K_L^0 and other particles is based on the penetration capabilities of muons. All particles except K_L^0 and muons are stopped in the ECL or in the solenoid before they reach the KLM . Clusters in the KLM that can be associated to a track are assumed to stem from muons, whereas clusters that cannot be associated to a track but possibly to a cluster in the ECL are taken as belonging to a K_L^0 [17].

2.3 The Belle AnalysiS Framework 2

Despite its name BASF2 is not only used for off-line analysis but for almost every software task at Belle II that is run on general purpose computers instead of specialized hardware[1]. Several third party libraries popular in particle physics such as EvtGen [20], Geant4 [21] and ROOT [22] are employed.

The basic building blocks of BASF2 are so-called *modules*. Written by developers and users they are used to accomplish a given task within BASF2 by building a chain of modules, a so called *path*, defined by *steering files* written in Python. The modules share data objects via the *DataStore* and employ functionalities from different libraries (see Fig. 2.11). It is possible to group modules into different paths that are executed conditionally based on the return value of other modules.

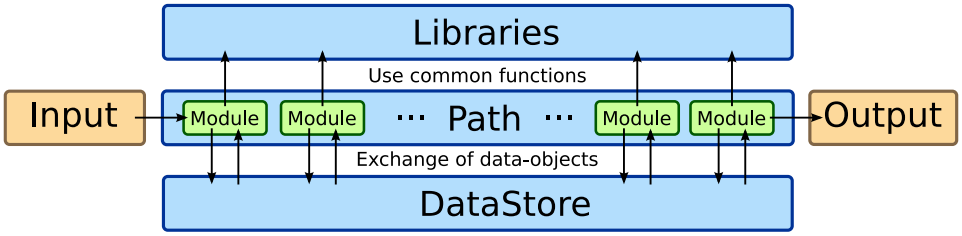


Figure 2.11: Schematic overview over the execution of a path consisting of different modules sharing data via the DataStore and using functionality provided by different libraries [19].

The DataStore is one of the essential parts of BASF2, as it allows to share information between different modules. Only well defined data objects can be placed on the DataStore, which in turn can be accessed by every module for reading and writing. The DataStore not only takes care of managing the required resources, but also tracks relations between data objects. These relations can be used to associate different data objects. Once registered by one module, the relation between two data objects can be used by another module in a later step.

Data objects are stored in the DataStore in so called *StoreArrays*, where every StoreArray holds one type of data objects. However, it is possible to create more than one StoreArray for a given type, which can then be differentiated by the name of the StoreArray. The objects in a StoreArray have a well defined lifetime: They exist either persistently throughout the whole execution of the path or only for one event.

Exemplary data objects that have a lifetime of an event are, e.g., `SVDCluster` and `PXDCluster`, which are the objects that are used for track finding in the VXD. Although these data objects contain no information from the simulation, they are

[1] e.g. readout electronics and the low level trigger

related to the objects `SVDTrueHit` and the `PXDTrueHit` containing truth information from simulation. With the help of these objects it is possible to analyze the performance of the track finding algorithm.

2.4 Track Finding in the VXD

The VXD TrackFinder (VXDTF) serves as a stand-alone track finder designed for reconstructing tracks within a broad momentum range, independently of other detectors. However, the main purpose of the VXDTF is to find low momentum tracks with transverse momenta as low as $p_T \approx 50\,\mathrm{MeV/c}$ [23]. As these tracks do not leave the VXD, no external information is available[2], making the VXDTF the only option to find such tracks. The VXDTF has to cope with several requirements and challenges [23]:

- Energy loss and multiple scattering have an influence on the particle trajectory. Both effects are more pronounced for low momentum tracks (see Sec. 3.1). Thus an algorithm that presupposes a perfectly circular track, for instance *conformal mapping* [24], is infeasible.
- As the SVD has only four layers, the redundancy is low.
- The expected number of real tracks is about 10 per event, but there are additional tracks from machine background.
- The geometry of the detector has to be incorporated as well as possible misalignment of sensors.
- Missing hits due to detector inefficiencies have to be compensated.
- Ghost hits (see Fig. 2.5) increase the number of hits that have to be considered and have to be effectively discarded.
- Background hits increase the occupancy and a signal to noise ratio of up to 1:10 in the SVD and up to 1:1000 in the PXD is expected.
- The reconstruction time budget is limited as it will be run on-line in real time.

2.4.1 Track Finding Strategy

The core algorithm for track finding in the VXD is a combination of a cellular automaton (CA), a Kalman Filter (KF) [25] and a Hopfield neural network

[2] e.g. by extrapolating tracks from the CDC into the VXD

(HNN) [26]. This approach allows to maintain the advantages of the Kalman Filter for track fitting while mitigating the combinatorial problem that arises in track finding [27].

The approach is to use the CA for track finding in conjunction with a prefiltering done by the so called *SectorMap* (see Sec. 2.4.4). This first step considerably lowers the combinatorics and produces sets of hits that are likely to form a track. The KF uses these hits to create track candidates together with estimated track parameters. However, these track candidates are not necessarily "clean". This means that it is possible that two or more track candidates share one or more hits. For further processing a clean, non-overlapping set of tracks has to be found. To find the optimal set of such tracks a HNN using a quality index calculated by the KF, will be employed in the VXDTF [23]. A schematic view of the workflow can be found in figure 2.12.

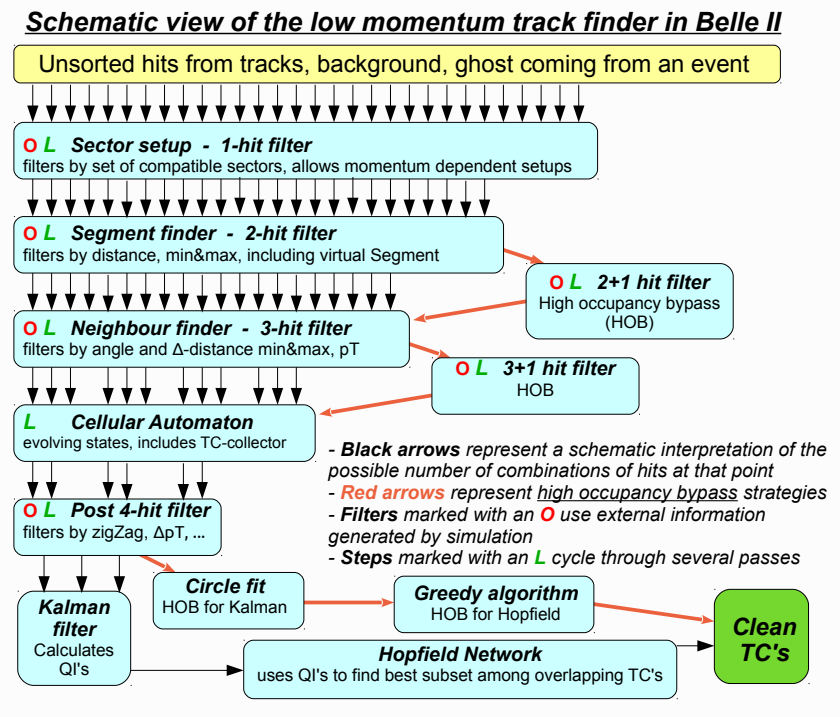


Figure 2.12: Schematic view of the workflow of the VXDTF and how the SectorMap approach is used to reduce the number of hit combinations that have to be considered. Adapted from [27].

2.4.2 Cellular Automaton

A cellular automaton is a "discrete dynamical system whose behavior is completely specified in terms of a local relation" [28]. It consists of discrete cells, with each of

these cells having a discrete state which evolves in discrete time steps depending on the the environment or the neighborhood of the cell. By defining appropriate rules for the cell-state evolution it is possible to use the final states to identify hit combinations that probably belong to the same track [27].

Applied to the VXDTF the definition of a cell usually is a segment connecting two hits with integer values as states. The state values are initialized to 0 at the start of the CA and after that evolve by obeying the following set of rules [27]:

1. Two cells are called connected if they share a hit.

2. If two connected cells have the same state and fulfill some criteria qualifying them as possible part of a common track, they are called *neighbors*.

3. Every cell having at least one neighbor on the inside with the same state raises its state by one at the end of the current step.

4. These steps are repeated until no cell-state changes occur anymore.

This set of rules has the consequence that the final state of each cell denotes the length of the chain of compatible cells from the inside out. Thus, higher states indicate longer chains. By choosing cells with a high state as starting points for collecting track candidates it is possible to significantly reduce the combinatorial background [27].

In the VXDTF the conditions that two hits have to fulfill to be connected as well as the conditions that have to be fulfilled by two cells to become neighbors can be position specific. This problem is addressed by the SectorMap (see Sec. 2.4.4), which stores different, position specific filter cuts and also allows to define a neighborhood that is no longer dependent of the detector geometry [27].

2.4.3 (Combinatorial) Kalman Filter

One of the standard tools for event reconstruction and track fitting nowadays is the extended *Kalman Filter* [25]. The Kalman Filter is a recursive version of the least-squares method for adding measurements to a track starting from a *track seed*. A track seed is a combination of hits that can either be chosen randomly or by a suitable pre-filtering technique.

The Kalman Filter can be regarded as a statistically optimal track following procedure in the sense that it is the best linear unbiased estimator of the parameters describing the current state of the system [25]. While it was originally derived for linear systems, the extended Kalman Filter can also be applied to non-linear systems.

The Kalman Filter consists of two steps: A *prediction step* and an *update step*. During the prediction step, information of the current state is used to predict the

state in the next sensor. The state consists of parameters, for instance position, direction and curvature. In order to do propagate a state a model is required that describes how the state evolves. In a nearly homogeneous magnetic field, such as the one in Belle II , a helix model is used to propagate the state. The covariance matrix of the state is propagated by linear error propagation. The actual computation of the helix model can be done analytically or numerically, for instance by a Runge-Kutta-method. The latter is a general numerical method for solving differential equations, in this case the equations of motion of a charged particle in a magnetic field

In the update step the predicted state is combined with the measurement to form the new state (see Fig. 2.13). In this way the Kalman Filter proceeds from one layer of the VXD to the next until no more measurements can be added or the track quality gets too bad.

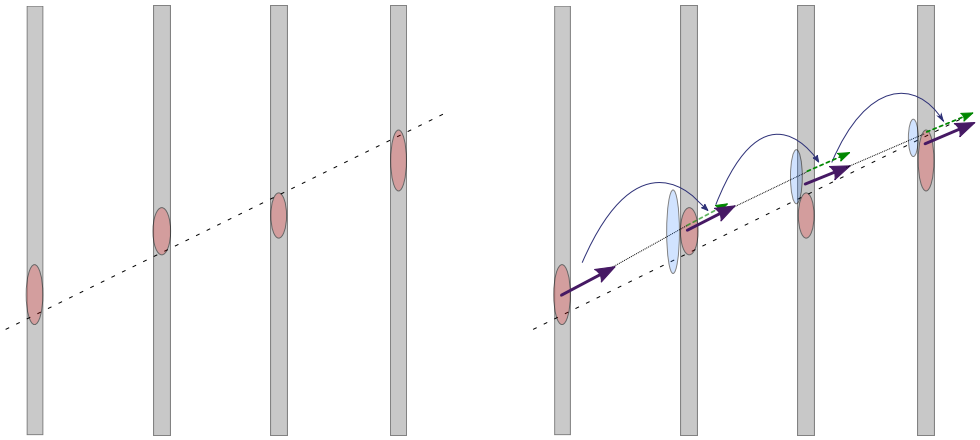


Figure 2.13: Schematic working principle of the Kalman Filter. A straight trajectory (dashed black line) creates a hit with additional measurement error (light red) on each layer (a). Starting from a track seed (left most purple solid arrow) an iteration step is performed leading to a predicted state (green dashed arrows) with an extrapolation error (light blue). Combining predicted state and measurement leads to the updated state (purple solid arrow). Pictures provided by Tobias Schlüter.

A more robust implementation of the Kalman Filter is the *Combinatorial Kalman Filter* (CKF). Instead of choosing only the best possible measurement to add to the current state the CKF clones itself for every measurement that can be added. In this way the effect of picking up a wrong hit can be mitigated. Nevertheless, both the ordinary Kalman Filter and the CKF have to check all hits within a certain window in the next layer if no suitable reduction is applied. Two reasons can be stated why reducing the number of hits to be checked is imperative. First, the

necessary calculations for checking a hit are complex and thus, time consuming. Second, and more importantly, a combinatorial problem arises from the increasing number of possible combinations of hits.

2.4.4 The SectorMap Approach

To reduce the combinatorial problem a so-called *SectorMap* is employed. A more detailed description of the approach can be found in [27] and [23] where also the implementation of the VXDTF is described briefly.

The general idea of the Sector Map is to reduce the number of hits that have to be considered for track candidates stepwise by fast filtering techniques before applying the slow track following algorithm of the (combinatorial) KF. In this way the advantages of the Kalman Filter can be maintained, while the combinatorial problem can be mitigated.

The workflow consists of several steps (see Fig. 2.12). First, the filter rules are applied to the hits of the current event. In the next step, compatible hits are assembled into chains that represent promising track candidates by the CA. The CKF operates on these track candidates before they are handed to the Hopfield Neural Network, which produces a set of clean track candidates. It has to be noted that currently the CKF is not implemented and that a circle fit is used to estimate the track quality.

As the name suggests, the SectorMap relies on the subdivision of sensors into smaller sectors. The filters of increasing complexity are simple geometrical filters with cut-offs. The division into sectors allows for a finer grained tuning of the necessary cut-off values compared to the usage of whole sensors. Furthermore the definition of allowed sector combinations already serves to reduce the number of hit combinations that have to be considered.

The filtering process can be broadly classified into different steps depending on the number of hits that are addressed by a filter:

- Determining compatible hits for the following steps by sorting them according to the list of allowed sector combinations. This is essentially a one-hit-filter without cut-off values.

- Determining pairs of compatible hits, so called *segments*, by applying different cut-off filters to all pairs of hits that pass the previous step. These so called two-hit filters check different distances such as the distance in 3D, the distance in the x-y-plane or the z-direction.

- Finding compatible three-hit combinations by combining segments with their so-called *neighbors*. Segments sharing a hit are checked by three-hit filters. These filters calculate angles between the segments on different planes in the

coordinate system, offset in the z-direction as well as the distance of the IP to a simplified circle fit using the three hits.

- After collecting chains of compatible hits that have passed previous steps with the cellular automaton, four-hit filters are applied to rule out the remnants of physically impossible track candidates by comparing the curvatures calculated from the possible three-hit combinations.

The determination of the allowed sector combinations and the cut-off values for the respective filters have to be determined in a separate step beforehand. This "training" of the SectorMap is done by collecting sufficient amounts of data obtained from simulated events. The cut-off values for each filter and sector combination are determined from quantiles of the distributions of values recorded during training.

To ensure a high track finding efficiency over a wider range of track momenta, several SectorMaps with different sector combinations and cut-off values can be used. Each SectorMap is trained with data in the corresponding momentum range. For track finding, a set of track candidates produced by any of the SectorMaps is processed to find a clean set of tracks.

3 A Short Physics Primer

As the main focus of this thesis is track finding a brief overview on the interactions of charged particles and matter will be given in Sec. 3.1. They represent the basic physics interactions relevant for the detectors used in the Belle II detector. A special focus will be put on the creation of electron-hole pairs in the silicon detectors of the VXD.

The second part of this chapter provides a brief introduction to the Standard Model of particle physics (SM) and to the physics goals pursued at Belle II . These introductions are deliberately held short as a more detailed description [29, 30] would go beyond the scope of this thesis. Additionally the main background sources at Belle II are sketched.

3.1 Interactions of Charged Particles with Matter

Particles moving through matter lose energy mainly due to ionization and excitation of atoms. The creation of electron-hole pairs in silicon detectors and the ionization in drift chambers is an example of such processes. For electrons and for other high energetic particles bremsstrahlung is another important energy loss mechanism which is exploited in the electromagnetic calorimeter (see Sec. 2.2.4). Finally the last process that will be described in more detail is Cherenkov radiation, which is used for particle identification in the TOP and the ARICH detectors (Sec. 2.2.3).

3.1.1 Ionization and Excitation

The energy loss of heavy charged particles moving through matter is described by the Bethe formula [31, Sec. 32]:

$$\left\langle -\frac{dE}{dx}\right\rangle = Kz^2\frac{Z}{A}\frac{1}{\beta^2}\Big[\frac{1}{2}\ln\frac{2m_ec^2\beta^2\gamma^2W_{\max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2}\Big]. \tag{3.1}$$

$\langle -\frac{dE}{dx}\rangle$ is the mean energy loss per unit length in units $\mathrm{MeV g^{-1}\,cm^2}$, $K = 4\pi N_A r_e^2 m_e c^2$, with N_A being Avogadro's constant, r_e the classical electron radius, and m_e the electron mass. The energy loss depends on the charge of the incident particle z, on the atomic number Z and the atomic mass A of the absorber, on the particle velocity encoded by the Lorentz factor $\beta\gamma$, on the maximum energy transfer in a single col-

lision $W_{\max}$, and on the mean excitation energy of the material I. The relativistic rise is suppressed by the so called *density effect* which is accounted for by $\delta(\beta\gamma)/2$.

The Bethe formula (Eq. 3.1) describes the mean energy loss for particles with $0.1 \lessapprox \beta\gamma \lessapprox 1000$ and is relatively independent on the incident particle mass and its initial energy. The mean energy loss reaches a minimum at $\beta\gamma \approx 3$ (see Fig. 3.1). Incident particles at this velocity are thus called minimum ionizing particles.

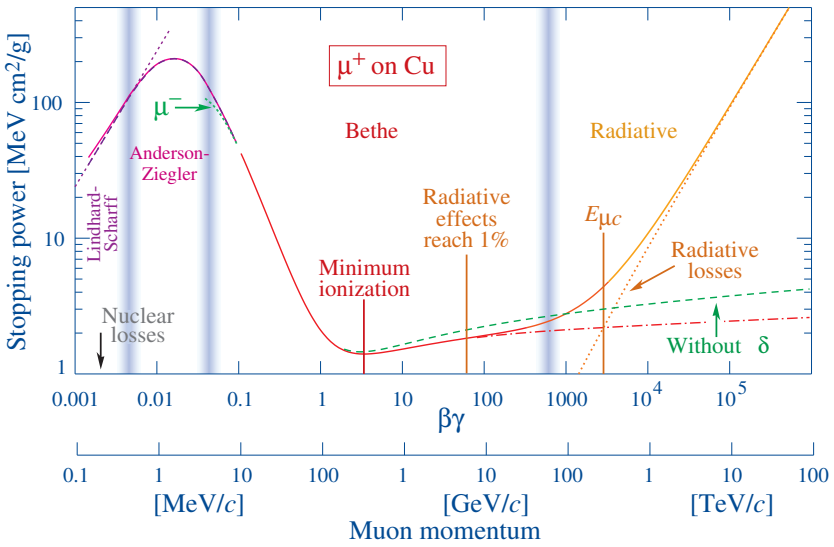


Figure 3.1: Stopping power ($= \langle -\frac{dE}{dx} \rangle$) for μ^+ in copper as function of $\beta\gamma = p/Mc$. The vertical bands indicate boundaries between different approximations valid for different regions of $\beta\gamma$ [31, Sec. 32].

The mass M of the incident particle enters the Bethe formula only indirectly via the maximum energy transfer in a single collision [31, Sec. 32]:

$$W_{max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2\gamma m_e/M + (m_e/M)^2}. \tag{3.2}$$

For $2\gamma m_e \ll M$ the "low energy" approximation is applicable and Equation (3.2) simplifies to $W_{max} = 2m_e c^2 \beta^2 \gamma^2$.

The Bethe formula is not applicable to describe the energy loss of electrons and positrons when passing through matter, as because of their low mass some of the assumptions used in the derivation of the Bethe formula no longer hold true. Furthermore the identity of the incident electrons and the electrons of the absorbing matter has to be considered in the quantum mechanical description of the scattering process.

The energy loss of electrons and positrons is energy dependent. For low energetic particles the main contribution to energy loss is ionization, whereas radiative losses

become dominant over about 10 MeV. The adaptions that have to be made to the Bethe formula are *Møller scattering* for e^-e^- processes and *Bhabha scattering* for e^-e^+ interactions (see Fig. 3.2).

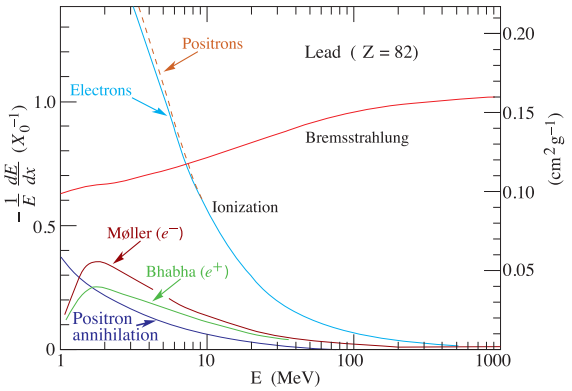


Figure 3.2: Different contributions to the energy loss of electrons and positrons depending on their energy [31, Sec. 32].

3.1.2 Bremsstrahlung

For high-energy electrons moving through matter the energy loss is dominated by radiative losses. One mechanism is e^+e^- pair production and subsequent annihilation of the positron, producing a high energy photon. The second mechanism is bremsstrahlung, where the incident particles lose energy by interaction with the Coulomb field of the matter nuclei. As bremsstrahlung is inversely proportional to the fourth power of the particle mass its effects are negligible for all particles but electrons. The energy loss by bremsstrahlung is dependent on the incident particle energy and can be written as

$$-\frac{dE}{dx} = \frac{E}{X_0}.$$

Thus, the energy of high-energy electrons decreases exponentially with the *radiation length* X_0 when passing through matter. The radiation length is a measure for (a) the mean distance over which a high-energy electron loses $1/e$ of its energy and (b) $\frac{7}{9}$ of the mean free path for pair production [31, Sec. 32].

3.1.3 Multiple Scattering

The interaction of a particle with the Coulomb field of the traversed medium not only leads to an energy loss but also to a change of its trajectory due to many

small-angle scatters[3] (Fig. 3.3). For many small-angle scatters the distributions of the resulting displacement and scattering angle in a projection perpendicular to the particle trajectory are approximately Gaussian. The standard deviation of the distribution of the projected scattering angle according to the Highland formula [31, Sec. 32] is given by:

$$\sigma_\theta = \frac{13.6\,\mathrm{MeV}}{\beta cp} z\sqrt{x/X_0}[1 + 0.038\ln(x/X_0)]. \tag{3.3}$$

Here p is the particle momentum, βc its velocity, z its charge number, and x/X_0 the thickness of the passed medium in units of radiation lengths. The standard deviation of the displacement in a direction perpendicular to the initial particle direction is [31, Sec. 32]:

$$\sigma_y = \frac{1}{\sqrt{3}} x\sigma_\theta. \tag{3.4}$$

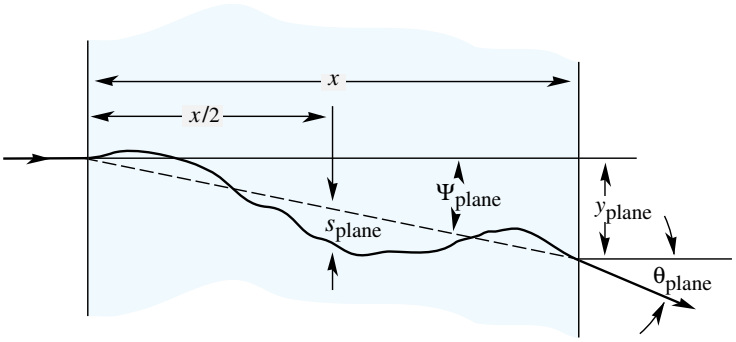


Figure 3.3: Multiple Scattering due to interactions with the Coulomb field of the traversed medium [31, Sec. 32].

The Gaussian approximation is only valid if some assumptions are met: There are no large angle scatters, and the the medium must not be too thin, as then the central limit theorem is no longer applicable. "Hard" collisions with large angle changes lead to tails in the Gaussian distribution, but are less frequent [31, Sec. 32].

3.1.4 Cherenkov Radiation

Although the energy loss of particles by Cherenkov radiation is negligible, the emerging radiation can be used to determine the velocity of the incident particle [31, Sec. 32]. Cherenkov radiation occurs when a charged particle moves through matter at a velocity greater than the local phase velocity of light.

[3] for hadronic particles strong interactions contribute to multiple scattering as well

If a charged particle moves through a medium, the medium gets electrically polarized by the particles' electric field. For particles with velocities exceeding the local speed of light this polarization relaxes by radiation of a coherent shockwave. The opening angle of this shockwave can easily be calculated from the particle velocity βc and the local phase velocity of light in the medium $\frac{c}{n}$, where n is the refractive index. During a time interval of length t the incident particle covers the distance βct, whereas electromagnetic waves travel as far as $\frac{c}{n}t$. In order to interfere constructively the wave fronts created along the track and the direction of the emitted photons must form a right angle (see Fig. 3.4). Thus, the Cherenkov angle θ_C can be calculated to

$$\cos\theta_C = \frac{\frac{c}{n}t}{\beta ct} = \frac{1}{\beta n}.$$

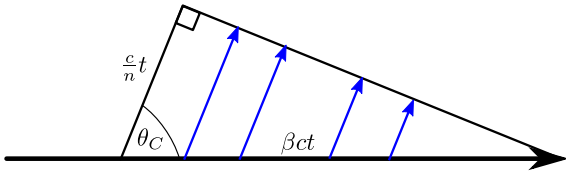


Figure 3.4: Illustration of the formation of Cherenkov radiation (blue arrows) created by an incident particle (thick arrow) with velocity βc in a non-dispersive medium.

Thus the Cherenkov angle of the emitted photons depends only on the refractive index n of the medium and on the incident particle velocity. Thus, in combination with other measurements, such as the particle momentum, it can be used for particle identification. The particle rest mass m_0 is a strong discriminator for different particle types and can be calculated directly from the particle momentum $p = m_0\gamma v$ and its velocity $v = \beta c$.

3.2 The Standard Model of Particle Physics

The Standard Model of particle physics is one of the most successful theories in physics. Developed in the early 1970s it has since been able to describe and predict a variety of phenomena, including the discovery of the long missing piece in the puzzle, the *Higgs* boson, in 2012 [32–34].

The SM is a field theory describing all currently known elementary particles and the interactions among them: the strong interaction, the weak interaction and the electromagnetic interaction. The only fundamental interaction not integrated into the SM is gravity. Neglecting gravity is justified by the fact that its strength is smaller by over 30 orders of magnitude compared to the other three fundamental interactions.

The SM classifies the elementary particles into two groups: the *fermions* with half-integer spin partaking in the interactions, and the *gauge bosons* with integer spin mediating them. The fermions are divided into *quarks* and *leptons*. Pairs of these are grouped together to so called *generations*. The bosons mediating the strong interaction are the *gluons*. The weak interaction is transmitted by the Z^0 and the $W^{\pm}$ bosons and the electromagnetic interaction by the photons. The Higgs boson plays a special role in the SM, as it does not convey an interaction in the sense of the SM, and is thus not considered a gauge boson. For an overview over the particles and interactions described by the SM see Fig. 3.5.

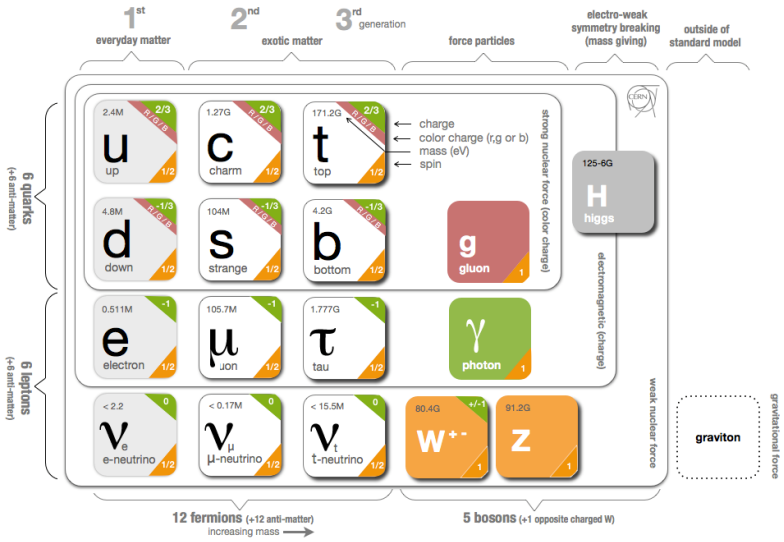


Figure 3.5: The elementary particles described by the Standard Model. Picture from taken from [35].

Despite its huge success the SM has some shortcomings [29, 30]:

- As mentioned above, it does not include gravity and is unable to unify the three fundamental interactions.
- It does not explain why there are three and only three generations of particles.
- The SM has 19 free parameters, the numerical values of which have to be measured by experiments.
- A hierarchy problem emerges from the Higgs mechanism. The measured Higgs mass of about $125\,\mathrm{GeV/c^2}$ requires an unnatural fine-tuning.
- The CP violation described by the SM (Sec. 3.2.1) cannot explain the observed matter-antimatter asymmetry in the universe.

There exist several models that try to solve these puzzles. However, current experimental results do not allow to dismiss all of them or to pick one among them. With the projected amount of data collected at Belle II several more stringent constraints should be possible, and, in combination with other experiments, like the ones at the Large Hadron Collider (LHC), physics beyond the SM hopefully becomes accessible.

3.2.1 CP-Violation in the Standard Model and the CKM Matrix

The strong interaction, described by quantum chromodynamics (QCD), conserves the number of quarks of each flavour separately. The weak interactions allows for transitions between different quark flavours, conserving only the total number of quarks.

The quark eigenstates participating in the weak interaction are not the mass eigenstates which appear in the SM (see Fig. 3.5). However, the eigenstates of the weak interaction can be described as a mixture of the mass eigenstates:

$$\begin{pmatrix} \mathrm{d}' \\ \mathrm{s}' \\ \mathrm{b}' \end{pmatrix} = V_{\mathrm{CKM}} \begin{pmatrix} \mathrm{d} \\ \mathrm{s} \\ \mathrm{b} \end{pmatrix} = \begin{pmatrix} V_{\mathrm{ud}} & V_{\mathrm{us}} & V_{\mathrm{ub}} \\ V_{\mathrm{cd}} & V_{\mathrm{cs}} & V_{\mathrm{cb}} \\ V_{\mathrm{td}} & V_{\mathrm{ts}} & V_{\mathrm{tb}} \end{pmatrix} \begin{pmatrix} \mathrm{d} \\ \mathrm{s} \\ \mathrm{b} \end{pmatrix} . \tag{3.5}$$

Here the primed quark eigenstates participate in the weak interaction, whereas the non-primed are the mass eigenstates. The 3×3 unitary matrix V_{CKM} is the *Cabibbo-Kobayashi-Maskawa* matrix describing the quark mixing and CP violation in the SM [36]. The CKM matrix can be parameterized as a rotation matrix around three Euler angles and one additional complex phase which is responsible for CP violation [37].

3.3 Physics at Belle II

Experiments at particle colliders can be broadly classified into two different approaches: On the one hand, the energy frontier experiments, such as the LHC experiments at CERN, and on the other hand, the precision frontier experiments, such as Belle II . The two approaches serve different purposes. While at the energy frontier the direct discovery of new particles is the major goal, at the precision frontier the signatures of new particles or processes are accessible via the precise measurement of flavour physics at lower energies. The approach of Belle II is to find evidence for deviations from SM predictions that can be interpreted in terms of new physics models [12].

The previous B-factory experiments Belle and BaBar were already successful in confirming CP violation in the B meson system [2, 3], which ultimately lead to the 2008 Nobel Price for physics awarded to M. Kobayashi and T. Maskawa. One of the main goals of Belle II is to further constrain the parameter space both of the SM and of models extending the SM. While some deviations from SM predictions were observed already at Belle, a larger amount of data is needed to thoroughly investigate these effects [38]. This can be achieved at Belle II due to the much larger instantaneous and integrated luminosity (Chapter 2).

Other important questions that are addressed by Belle II will be the search for new CP violating phases, the search for new flavour symmetries that can explain the CKM hierarchy, and the search for new flavour violating processes, such as lepton flavour violation [38]. A more detailed description of the physics possibilities at Belle II can be found in [12] and [38] and the references therein.

3.3.1 Background Sources at Belle II

The increased luminosity of SuperKEKB leads to an increased background rate as well. The main sources of background at SuperKEKB and Belle II are the following [12, 39–41].

Synchrotron Radiation Synchrotron radiation (SR) is created when electrons are accelerated radially by, for example, a magnetic field. Due to SR the stored particles lose energy in every revolution, which has to be compensated. However, radiation causing background in the detector originates from the bending of the electrons and positrons at the focusing magnets. As the effects scales with E^2 it is dominated by the electrons in the HER.

Beam-Gas Scattering Although the beam pipe is held at vacuum some residual gas atoms are still present. These atoms can change the direction of beam particles or induce energy loss by bremsstrahlung. If scattered particles hit the beam pipe or the vacuum chamber they induce showers which create background hits in the surrounding detectors.

Touschek Scattering Particles in a bunch exchange momentum between their transversal and longitudinal directions. The exchange is enhanced by relativistic effects, leading to beam particles hitting the walls of the vacuum chamber and magnets. The induced showers can reach the detector and produce background hits. The rate of Touschek scattering is proportional to E^{-3}. Thus, the major source are the positrons of the LER. The Touschek effect is the reason why the energy of the positrons in the LER was increased to 4 GeV/c.

Radiative Bhabha Scattering The scattering of electrons and positrons can lead to the creation of a photon which propagates along the beam axis direction. If this photon hits the iron of the magnets a neutron can be produced via the giant photo-nuclear resonance mechanism. These neutrons are a major background source in the KLM. The rate of Bhabha scattering is proportional to the luminosity, which can be exploited to measure the luminosity.

QED Background Electron-positron pairs are produced via two-photon processes: $e^+e^- \to \gamma\gamma \to e^+e^-e^+e^-$. The produced pairs have low momenta, rendering this the main background source for the PXD.

Beam-Beam Interaction The particles of the colliding beams interact which leads to changes of the trajectory of the beam particles. This is the least well understood source of background due to its non-linear nature, and precise studies are difficult.

4 Machine Learning Basics

The basic idea of machine learning (ML) is to avoid specifying a predefined set of rules that are followed by a system, by having an algorithm that allows a system to learn this set of rules by itself. One implication of this is that even large and complicated sets of rules can be learned. But the more important consequence is that no or little prior knowledge of these set of rules is necessary in order for them to be learned if they actually are learnable.

Machine learning algorithms can be broadly classified into three categories [42]:

- **supervised learning**: For every training input the desired output is known. Hence, the aim is to learn a set of rules or a general rule that maps inputs to outputs.
- **unsupervised learning**: Only unlabeled inputs are given and the aim is to find possibly hidden structures among these inputs and label the inputs accordingly.
- **reinforcement learning**: The learning system is provided with information in the form of a scalar reinforcement factor measuring how well the system performs [43]. The system has to discover which action yields the highest rewards autonomously.

Because of the particular classification problem studied in this thesis only supervised learning will be described in more, but still by no means exhaustive, detail. Furthermore only a subset of Artificial Neural Networks (ANN), the so called Multilayer Perceptrons (MLP, Sec. 4.2), will be treated together with Boosted Decision Trees (BDT, Sec. 4.3) as an alternative classifier algorithm.

This introduction is deliberately kept short and rather technical, and the reader is provided with some references to more thorough and biologically inspired introductions [5, 42, 44–46] and some recent review articles [47–50].

4.1 Supervised Learning

Mathematically speaking supervised ML is a function approximation problem. The aim of training is to learn some kind of mapping from inputs X to outputs Y:

$$\mathbf{g} : X \to Y.$$

The mapping is learned by minimizing the expectation value of a predefined loss function (or cost function) L for all possible values of inputs and outputs [51]. A loss function maps the outputs of the system onto a scalar value in a way that deviations from the desired output are penalized. For some supervised learning algorithms the loss function has to meet certain criteria, e.g. differentiability or convexity. A typical loss function is the quadratic loss function

$$L_{\mathrm{Q}} = c \sum_i (t_i - y_i)^2, \tag{4.1}$$

where c is some constant usually set to $\frac{1}{2}$, t_i is the desired output for sample i and y_i is the output produced by the system.

4.1.1 Overtraining

As during training only a subset of all possible input/output pairs is known, the possibility arises that the expected loss is minimized on the training set, but not on the entire set. In this case the system performs better on the training set than on other sets, and one speaks of *overtraining* or *overfitting*.

Since one of the main goals of ML is to generalize from possibly small training sets to larger sets, avoiding overfitting is crucial. No general best way for avoiding overfitting exists. However, different approaches exist to stop training before overfitting occurs.

Cross validation and special forms like *hold out validation* are commonly used techniques. The training set is split up into two (hold out validation) or more (cross validation) subsets and different systems are trained with all but one of these subsets, which can then be used as a testing sample to check if a system performs worse on it than on the subsets used for training. If the performance is worse on the set which was not used in training it can be assumed that the system is overtrained.

Generally speaking, if two systems perform (almost) equally after training on a test set, the system with fewer free parameters, or casually spoken the simpler system, is assumed to generalize better.

4.2 Artificial Neural Networks and Multilayer Perceptrons

Inspired by and trying to resemble the data processing of the human brain, ANNs have been proposed almost as early as the first programmable computers in the 1940s [52]. Based on the *Hebbian Theory* of learning [53], algorithms have been developed to train ANNs to accomplish given tasks. According to Hebb, the effi-

ciency of connections between neurons in the (human) brain is increased for such neurons that are often activated at the same time. After an initial golden time of ANN research, it experienced a sudden decline in the late 1960s, when major drawbacks of the then current approaches were revealed [54]. Benefiting from increasing computing power and improved concepts, the area experienced a renaissance in the late 1970s and 1980s, one that has been lasting until today.

4.2.1 Components of an Artificial Neural Network

ANNs consist of abstract representations of its biological counterparts, the neurons, often calles nodes in the context of machine learning (see Fig. 4.1a), and the weights connecting them, representing the axons, dendrites and synapses. The neural network can be represented as a (directed) graph in which the neurons are the vertices and the connections between them are the edges (see Fig. 4.1c).

The model of the neuron is completed by specifying rules on how input signals are combined and how the combined input signals are transformed into an output signal which is then passed to other connected neurons [45]. Most commonly the inputs are combined by a weighted sum:

$$z_j = \sum_{i=1} x_i w_{ij}. \tag{4.2}$$

The weight w_{ij} represents the connection from neuron i to neuron j and is commonly stored in the weight matrix W. This matrix can be considered the memory of the ANN, since it holds all parameters that are adjusted during training [47]. The transformation to an output signal is done via the so-called activation function $S(z)$. The activation functions used in this thesis are (Fig. 4.1b):

- **linear**: $S(z) = c \cdot z$, with c being a constant usually set to 1
- **tanh**: $S(z) = \tanh(z)$, the hyperbolic tangent
- **logsig**: $S(z) = 1/(1+\exp(-z)) = \frac{1}{2}(1+\tanh\frac{z}{2})$, a special case of the logistic function

The latter two are sigmoid functions and are commonly used in MLPs. They fulfill the conditions of the *Universal Approximation Theorem* (Sec. 4.2.3) and are suitable for *Backpropagation Training* (Sec. 4.2.4). The linear activation function is standardly used for the input nodes but can also be used for other nodes. However, using only linear nodes imposes rather strict constraints on the possibilities of ANNs (Sec. 4.2.3).

All presented activation functions have a fixed output for $z = 0$. Sometimes, however, a different value is desired for $z = 0$. To overcome this limitation a

so-called *bias unit* with a fixed output value of 1 and variable weight w_{0j} is introduced. Thus the combined input of a neuron results to $z_j = \sum_{i=0} x_i w_{ij}$, with $x_0 = 1$ (see Fig. 4.1a). This allows to implement the desired output for $z = 0$ and furthermore hand over its determination to the training algorithm.

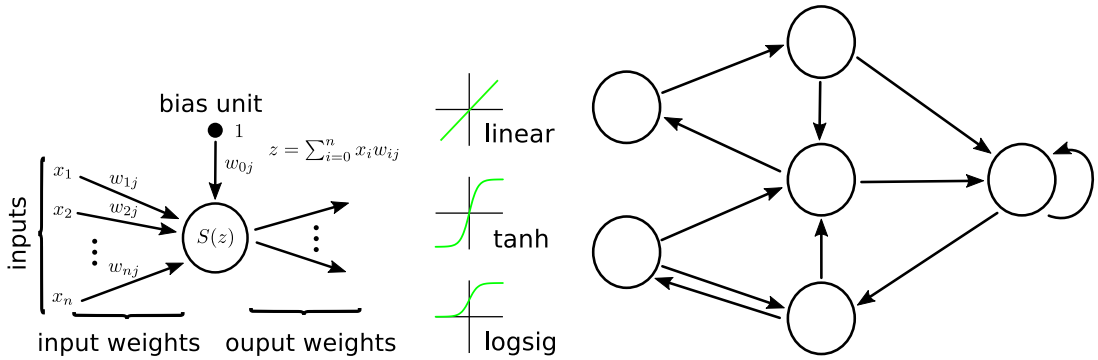


Figure 4.1: Model of a neuron in an ANN with a bias unit and n inputs (a). Graphical representation of different activation functions used in the thesis (b). Exemplary representation of an ANN as directed graph.

4.2.2 Multilayer Perceptrons

One of the earliest ANN approaches were so called Multilayer Perceptrons. A MLP is a fully connected feed-forward neural network (FFN), where the neurons are organized in layers and every neuron in one layer is connected with all neurons in the following layer. Thus, the information flow is unidirectional from the input layer through one or more hidden layers to the output layer (see Fig. 4.2). The name *hidden layers* is chosen because they neither receive input from, nor do they transmit output to the environment directly.

The connections between two separate layers can be stored in separate weight matrices. Hence, for N layers (including input and output layers) $N - 1$ weight matrices have to be determined in the training process.

4.2.3 Universal Approximation Theorem

If the activation functions of all nodes are linear functions, the output is a composition of linear functions of the inputs and therefore again a linear function of the inputs. Such a network may learn local linear approximations to non-linear functions, but is certainly not capable of learning non-linear functions globally.

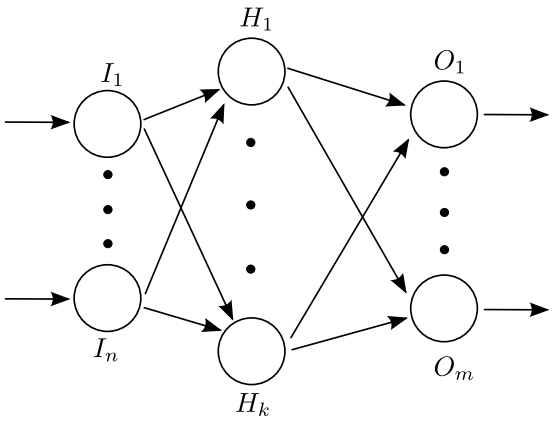


Figure 4.2: Schematic layout of a fully connected feed-forward neural network, a Multilayer Perceptron, with n input nodes, k hidden nodes in one hidden layer and m output nodes, thus implementing a mapping $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$.

Replacing the activation functions in the hidden layer with non-linear ones removes this restriction, turning MLPs into a powerful tool possessing an important theoretical property: the *Universal Approximation Theorem*. The theorem states that any FFN with a single hidden layer containing a sufficiently large, but finite number of neurons is capable of approximating continuous functions to arbitrary precision on compact subsets of $\mathbb{R}^N$ [55, 56].

The theorem sets only mild constraints on the activation function: it has to be bounded, non-constant and monotonically increasing. As a result even simple ANNs are capable of representing a wide variety of functions and forming arbitrarily shaped disjoint decision regions, if they are given appropriate parameters [57, 58]. However, the proof of the Universal Approximation Theorem is not constructive; it does neither provide a general way of learning the weights and biases nor make a statement on the ability of the ANN to learn them during the training process.

4.2.4 Backpropagation Training

Based on the *delta rule* [59] the *backpropagation* (BP) algorithm is one of the most popular techniques for supervised learning of ANNs. While the delta rule is only applicable for perceptrons consisting solely of an input and an output layer, the backpropagation algorithm implements a generalized delta rule that allows adjustment of weights in hidden layers as well. Both algorithms implement a gradient descent on the loss surface (or error surface) generated by the loss function [59].

After initialization of all weights with small random numbers, a training step for both algorithms consists of the following parts:

1. Present the input samples to the ANN and propagate them through the network to calculate the output of each sample.

2. With the help of the loss function calculate a so-called *error signal* for the output layer.

3. Recursively determine the error signal for each layer in backward direction (i.e. from output to input) and apply weight changes (Eq. 4.3).

4. Repeat 1.–3. until convergence or the desired performance is reached.

After presentation of one sample the change to be applied to a given weight w_{ij} according to the (generalized) delta rule reads [59]:

$$\Delta w_{ij} = \eta \delta_i o_j, \tag{4.3}$$

where η is the *learning rate* that can be used to control the convergence of the learning process, δ_i is the error signal encoding the deviation of the desired and the actual output and o_j is the j-th input to the node.

Obviously the error signal has to be defined differently for the output layer, in which external information is available, and the hidden layers, in which such information is not directly available. In output nodes the error signal is defined as

$$\delta_i = \frac{\partial L}{\partial y_i} S'(z_i), \tag{4.4}$$

where L is the loss function, y_i is the i-th component of the output for input sample $\vec{x}$ and $S'(z_i)$ is the derivative of the activation function evaluated at z_i (Eq. 4.2).

In hidden nodes the error signal reads:

$$\delta_i = S'(z_i) \sum_k \delta_k w_{ik}. \tag{4.5}$$

Hence, it incorporates the error signals of all connected nodes in the succeeding layer (from a signal processing point of view). Backpropagation requires the activation function to be differentiable and monotonous [59].

Depending on the loss function as well as on the chosen learning rate η, convergence to a minimum can take a rather long time. As BP implements a gradient descent and supervised learning is a function optimization problem (Sec. 4.1), several algorithms have been developed to speed up convergence: e.g. *Scaled Conjugate Gradient* (SCG) [60], which tries to minimize the number of training steps needed to reach a minimum by carefully choosing the direction and the step size of one step by using second order information [60].

4.2.5 Limitations and Capabilities

Although ANNs are suitable tools a vast variety of tasks (Sec. 4.2.3), some drawbacks have to be mentioned. One of the most prominent problems is that models produced by ANNs are in general not intuitively interpretable, thus rendering them a black box for the user. While this is no real problem if the ANN has been trained properly, it can become tedious to spot flaws. Furthermore, since even small changes of one weight can result in an entirely different model, manual adjustment of a trained network is almost impossible.

Due to their composition of independent neurons ANNs can be evaluated in parallel, resulting in fast execution speeds. Nevertheless, considerable time can be spent in training, especially with increasing numbers of neurons. However, this is a problem not exclusive to ANNs and is generally of only minor concern, since once trained, the parameters are fixed and evaluation time becomes the relevant characteristic.

As mentioned before, supervised machine learning is nothing but the minimization of a predefined loss-function (Sec. 4.1). In MLP training this is frequently achieved by Backpropagation Training (Sec. 4.2.4) which implements a gradient descent. As always with gradient descent, it is not guaranteed that the global minimum is reached or that the final local minimum is sufficient for the targeted purposes. Although reaching the global minimum is rarely required, reaching a sufficient minimum can require several training runs with different initializations.

4.3 Decision Trees and Boosting

One of the drawbacks of ANNs is that they can only handle numerical but not categorical data. Although categorical data can be mapped to numerical data to be used by ANNs, such a mapping has no “natural” or canonical definition and introduces arbitrariness into the task. *Decision Trees* (DT) and *Boosted Decision Trees* (BDT) are capable of dealing with categorical and numerical data directly without use of any transformation.

4.3.1 Decision Trees

DTs are a form of so called logical learning methods that classify input samples by sorting them based on feature values or *features* [47]. A DT can again be represented as a (directed acyclic) graph, where the vertices are divided into nodes and leaves. The latter are vertices with no outgoing edges. On every node of a DT the data is split according to the value of one feature until a leaf is reached which is then the output of the DT (Fig. 4.3). In general this procedure is not repeated until every training sample is correctly labeled but is instead limited to a maximum

number of splits or to a certain depth of the tree. A DT can be translated into a set of rules by creating a rule for each path from the root to a leaf in the tree [61].

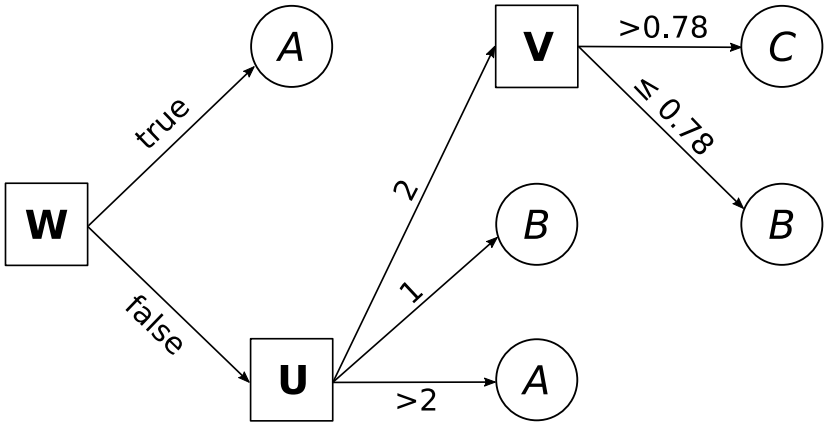


Figure 4.3: Exemplary decision tree labeling input based on three features, **U**, **V** and **W** into three classes A,B and C.

Decision Tree Construction

Training (or constructing) a DT is done by choosing the variable that best splits the data at the root node of the tree and repeating this procedure for every branch of every node in a top-down manner. As constructing an optimal binary DT is a NP-hard[4] problem [62], different measures exist to heuristically determine which variable best splits the data [61].

4.3.2 Boosting

Although DTs can be powerful classifiers, they are generally prone to overfitting. To counteract this tendency, the depth of a DT can be limited. While this generally avoids overfitting, it also affects the classification performance. If the depth is limited to very shallow trees or even tree stumps[5], DTs are considered to be so-called *weak learners*, which by definition have to label data only slightly better than a random labeling does. Despite this apparently major drawback it is possible to combine several weak learners in order to form a strong learner [63]. One way of combining trees is the *boosting* method.

Boosting is done by iteratively training an ensemble of weak learners and applying weights to the training samples according to the deviation of learner output and

[4] NP-hardness is a notion from complexity theory. Put simply, a solution to an NP-hard problem can be verified quickly, however, no efficient way is known for finding a solution.

[5] tree consisting of only the root node

desired output. While mislabeled samples gain weight, correctly labeled samples lose weight such that the weak learners after every boosting step concentrate more and more on samples that were misclassified by previous instances. To compensate for the changed weights of the training samples each weak learner of the ensemble is assigned a weight for evaluation.

Several boosting algorithms exist. However, only *AdaBoost* [64] and *Stochastic Gradient Boost* [51] (SGB) will be briefly described here. Both algorithms share some properties and parts of the initialization process as well. All weak learners are assumed to be DTs and the output is assumed to be binary[6] for the description of the algorithms. The two classes are called signal and background in accordance with the overall topic of the thesis.

Mathematical Basics

As mentioned before, supervised learning can be regarded as function approximation (Sec. 4.1). Boosting achieves this approximation by an expansion of the form [51]

$$g(x) = \sum_m \alpha_m h_m(x), \tag{4.6}$$

where the $h_m(x)$ are the hypotheses from base learners and α_m are the weights assigned to them[7]. Training starts with an initial guess $g_0(x)$, and the parameters of h_m and α_m are then determined by means of adding a new base learner to the previously trained ensemble and choosing the optimal values for them. The approximation after adding a new base-learner is

$$g_m(x) = g_{m-1}(x) + \alpha_m h_m(x). \tag{4.7}$$

Boosting algorithms differ in the way in which they choose the optimal values for the parameters of the base learners and α_m.

For the following description let N be the total number of (training) events, in which the i-th event consists of an input vector $\vec{x}_i$, a target value t_i and a corresponding weight w_i. The ensemble consists of M trees, where the hypothesis of the m-th tree is denoted by:

$$T_m(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{x}_i \text{ lands on signal leaf} \\ 0 & \text{if } \vec{x}_i \text{ lands on background leaf} \end{cases}$$

At the beginning, all events are weighted with $w_i = \frac{1}{N}, i = 1...N$.

[6] i.e. AdaBoost.M1

[7] The parameter-dependence from [51] is dropped in this basic description

AdaBoost

After training a tree, its hypothesis is tested by applying it to the training events and calculating the error[8]

$$e_m = \frac{\sum_{i=1}^{N} w_i I(t_i \neq T_m(\vec{x}_i))}{\sum_{i=1}^{N} w_i},$$

where I is the indicator function which is 1 if its argument is true and 0 otherwise. This yields e_m denoting the weighted fraction of misclassified events. The weight for the m-th tree in the BDT is calculated as

$$\alpha_m = \beta \ln \frac{1 - e_m}{e_m},$$

where $\beta > 0$ ($\beta = 1$ for the standard AdaBoost algorithm [8]) can be seen as a parameter governing the learning speed. The weights are updated according to

$$w_i \leftarrow w_i e^{\alpha_m I(t_i \neq T_m(\vec{x}_i))}$$

and afterwards renormalized such that $\sum_{i=1}^{N} w_i = 1$. In this way, misclassified events gain weight[9] whereas properly labeled events lose weight in the normalization process. The score of an event $\vec{x}$ is finally calculated as

$$T(\vec{x}) = \sum_{m=1}^{M} \alpha_m T_m(\vec{x}) \tag{4.8}$$

yielding a real number that can then be used to define cuts for sorting a presented event into either signal or background. The loss function minimized by AdaBoost is the exponential loss L_E [65]:

$$L_E = \sum_{i=1}^{N} e^{-I(t_i = T(\vec{x}_i))}. \tag{4.9}$$

Stochastic Gradient Boosting

The optimization problem of choosing the weights α_m and the parameters of a base learner h_m as described in Sec. 4.3.2 is solved by Gradient Boosting by splitting the process into two parts, that can be solved more easily [51].

[8] This description follows the one found in [8] but is equivalent to the original of [64]
[9] Since $e_m \leq \frac{1}{2}$ by the requirement of each tree being a weak learner $\alpha_m \geq 0$

First the base learner h_m is determined form a least-squares fit to the so called *pseudo residuals*

$$\tilde{y}_{im} = -\Big[\frac{\partial L(t_i, g(\vec{x}_i))}{\partial g(\vec{x}_i)}\Big]_{g(\vec{x})=g_{m-1}(\vec{x})}. \tag{4.10}$$

Then the optimal value for α_m is determined via a parameter optimization by minimizing the expected loss over the training sample after adding the previously determined base learner.

To further improve generalization capabilities of Gradient Boosting, Stochastic Gradient Boosting randomly draws a sample of fixed size form the training set for each boosting step. Another technique to avoid overfitting is so called *shrinkage* which modifies the update rule (Eq. 4.7) as follows:

$$g_m(x) = g_{m-1}(x) + \nu\alpha_m h_m(x)$$

Here ν is called the shrinkage or learning rate and is usually set to small values to yield the best results [66].

4.4 Data Selection and Processing

Aside from picking an algorithm, selecting the data for training has to be done with greatest care. It has to be representative for the distribution on which the system has to perform after training. If availability of data is only a minor concern it is feasible to generate different sets for training and testing to check if both follow the same distribution and to do cross or hold-out validation to avoid overtraining (Sec. 4.1.1).

Pre- or post-processing of the data is another handle to improve the performance of supervised machine learning systems. Common pre-processing techniques include some form of statistical analysis to ensure the identification of the relevant features (e.g. decorrelation) or some numerical transformations to meet the requirements of the systems optimal performance range (e.g. normalization or transformation to a uniform distribution) as well as identifying missing values and outliers. Post-processing includes (among others) the determination of optimal cut values on the system outputs for classification purposes.

4.4.1 Decorrelation

In a typical input for a machine learning system the features are correlated, implying that some information is stored in the input features redundantly. Analyzing these correlations can help to identify non-relevant features that can be excluded from training. Aside from a possible performance enhancement (Chapter 5) there is

the possibility of lowering the number of necessary free parameters in a system by applying a transformation to decorrelate the input features.

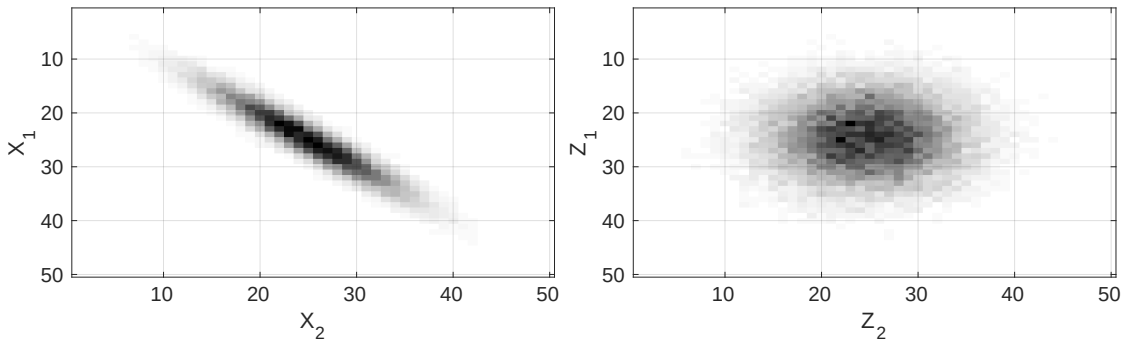
The transformation to decorrelate a random variable X can be obtained from an eigendecomposition of the covariance matrix [66]

$$\mathrm{Cov}[X] = \mathbf{U}\mathbf{D}\mathbf{U}^T, \tag{4.11}$$

where $\mathbf{D}$ is a diagonal matrix containing the eigenvalues of $\mathrm{Cov}[X]$ and $\mathbf{U}$ is an orthogonal matrix whose column vectors are the eigenvectors corresponding to the eigenvalues in $\mathbf{D}$. The transformation from X to a non-correlated variable Z reads

$$Z = \mathbf{D}^{\frac{1}{2}}\mathbf{U}^T X, \tag{4.12}$$

where $\mathbf{D}^{\frac{1}{2}}$ is for normalization purposes only. The covariance matrix $\mathrm{Cov}[Z]$ is diagonal, and if normalized, unity. Hence, Z is by definition not linearly correlated. Geometrically speaking, the transformation is a rotation of the inputs followed by a rescaling (see Fig. 4.4)



(a) original (b) decorrelated

Figure 4.4: Geometrical interpretation of decorrelation as rotation and rescaling in the input space.

4.4.2 Transformation to Uniform Distribution

According to the *Probability Integral Transformation* any random variable X can be transformed to a random variable X' having a uniform distribution given the *cumulative distribution function* (CDF) F_X [67] (see Fig. 4.5):

$$X' = F_X(X) \tag{4.13}$$

While this not only is a common approach in statistics when dealing with multivariate distributions, it furthermore allows to optimize the training and execution

procedures of BDTs by operating on bin numbers rather than on actual float values [19].

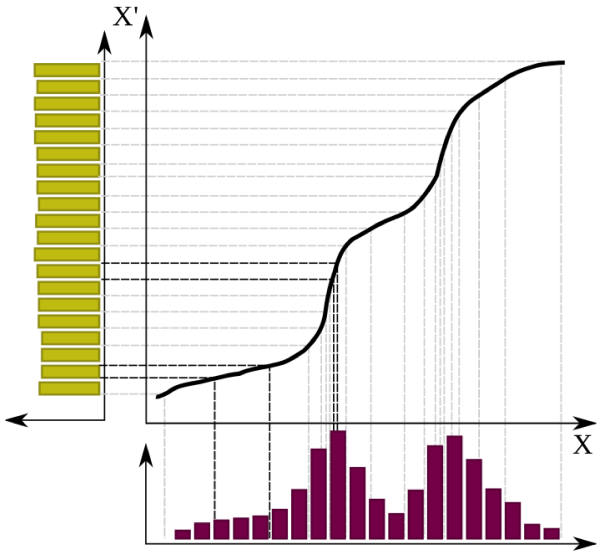


Figure 4.5: Transformation to a uniform distribution using the CDF. Adapted from [68]
.

5 Machine Learned Tracklet Filters

The current SectorMap approach in the VXDTF (Sec. 2.4) requires tuning a large number of cut off values for the different filters. As this number is in the order of 10^6, this tuning requires considerable amounts of computational resources to gather the necessary quantities of data. While tuning these cut off values can be seen as a very basic machine learning approach, applying more advanced supervised machine learning techniques (Chapter 4) to the problem has several possible benefits. These are primarily connected to the generalization capabilities of such systems:

- A smaller number of needed sectors yields a smaller number of filters that have to be tuned.
- The size of the required training data set can be reduced, lowering the necessary amount of computational resources
- The number of SectorMaps needed for different momentum ranges can be reduced.

Furthermore including signal and background in the training process potentially yields a better separation of signal and background. The resulting lower number of hit combinations that have to be checked in a later stage might allow for faster execution of the track finder.

5.1 Chosen Approach and Goals

Stepping in at the *two hit filter* stage of the SectorMap is probably unfeasible due the high occupancy and the highly imbalanced data. The ratio of background to signal is very high, rendering training an efficient classifier difficult. Thus, the approach taken in this thesis is to train a classifier that acts as a *three hit filter* in the SectorMap. At this stage the data is almost balanced (see Tab. 5.1), and the occupancy has been considerably reduced by the previous stage. Furthermore, a third hit offers additional information which might allow for a better discrimination between signal and background. The gained advantages can then be used to loosen the requirements on the previous filter stage.

The main goal of the SectorMap approach is to keep as much signal as possible while discarding as much background as possible. The measures used to assess the different machine learning approaches against each others are defined as follows:

- **signal efficiency**

$$r = 1 - \alpha \tag{5.1}$$

  where α is the ratio of signal samples misclassified as background. The prefix *signal* will be omitted most of the time.

- **signal-to-noise ratio (SNR)**

$$\text{SNR} = \frac{\text{number of signal samples in data set}}{\text{number of background samples in data set}} \tag{5.2}$$

- $\mathbf{SNR_{in}}$, the SNR in the input data set
- $\mathbf{SNR_{out}}$, the SNR in the data set that is labeled as signal by the classifier

$$\text{SNR}_{\text{out}} = \frac{\text{number of signal samples classified as signal}}{\text{number of background samples classified as signal}} \tag{5.3}$$

- **SNR gain**, the ratio of input to output SNR .

$$\text{SNR gain} = \frac{\text{SNR}_{\text{in}}}{\text{SNR}_{\text{out}}} \tag{5.4}$$

The definition of the SNR gain yields a measure that is independent of the SNR in the input data and thus allows to compare classifiers that have been trained and tested on different data sets more easily than by just comparing the output SNR.

To discard as little signal as possible the desired efficiency of the classifiers is set to $r \geq 0.99$ if not stated otherwise in this chapter. Hence, more than 99 % of all signal samples have to be classified as such. This chapter addresses the standalone performance of the machine learned three hit filters only. This allows to determine the proper parameters of the ML classifiers like tree depth, number of boosting steps and number of hidden nodes (Sec. 4.3 and 4.2). However, statements on how the performance of the VXDTF is affected are limited.

5.2 Generating Data Sets

As mentioned in section 4.4, generating the training data has to be done very carefully to ensure proper operation after training. In a later stage the framework will allow to directly receive all three hit combinations that passed the two hit stage of the SectorMap. At the moment, however, the VXDTF has to be "misused" to obtain training data.

To have data that as closely as possible resembles the data that will later be filtered by the classifier, only the two hit filters that are turned on for standard

reconstruction are enabled in data generation, while all three and four hit filters are disabled. Any combination of segments that is allowed by the SectorMap is connected to a track candidate. No further processing (i.e. no CKF and no HNN) is done to filter out possible background. The track candidates are afterwards split up into tracklets containing three hits each and each of these tracklets is either assigned a signal or a background label.

To be labeled as a signal sample, each of the three hits has to be related to the same MCParticle[10]. If one or several hits are related to more than one MCParticle, but there is one MCParticle shared by all of them, the sample is still considered as signal sample. The term *background* is used as a collective in this thesis as no discrimination between machine background and combinatorial background is made.

5.2.1 Properties of Data Sets

For reasons of comparability all classifiers have been trained with the same training data set and afterwards tested with a test data set. The training data set is randomly chosen from a data set in which the samples obtained from 100,000 simulated events are contained. All sample not used in the training process are used for testing. Every sample

$$X = (\vec{x}, t), \quad \vec{x} \in \mathbb{R}^9, t \in \{0, 1\} \tag{5.5}$$

consists of the three dimensional spatial coordinates of the three hits $\vec{x}$ and an associated target value t indicating if the sample is background (0) or signal (1). Incorporating further information can readily be done if it is kept in mind that no MC information, other than determining the target value, can be used as input, since such information is absent in the actual experimental situation. Nevertheless this information can and will be used in performance analysis (Sec. 5.6). The complete data set is split into three subsets (see Tab. 5.1):

- consecutive set: The hits of the samples appear on consecutive layers with no skipped or repeated layers.
- overlapping set: At least two hits in the sample are on the same layer. For non-curling tracks this is only possible in the overlapping parts of the detector.
- skipping set: The hits in the sample 'skip' one or more layers. The most frequent reason for this to happen is when a particle passes an insensitive part of the detector.

[10] see Glossary for a short explanation

data set	# signal samples	# bg samples	# total samples	SNR	ξ
consecutive	1,279,539	1,680,919	2,960,458	0.7612	0.25
overlapping	180,069	337,204	517,273	0.5340	0.5
skipping	48,450	79,862	128,312	0.6066	0.75
complete	1,508,058	2,097,985	3,606,043	0.7188	0.25

Table 5.1: Number of signal and background (bg) samples in the different data sets (training and test data sets combined) and fraction ξ that is used as training set.

Since the data sets are not of the same size, different fractions ξ are chosen for randomly drawing the training sets from the complete sets. This ensures each the training set is large enough to provide enough information for a successful training.

To ensure the resemblance of training and test data sets the distributions of each component are compared via a Kolmogorov-Smirnov (KS) test [69] (see Fig. 5.1).

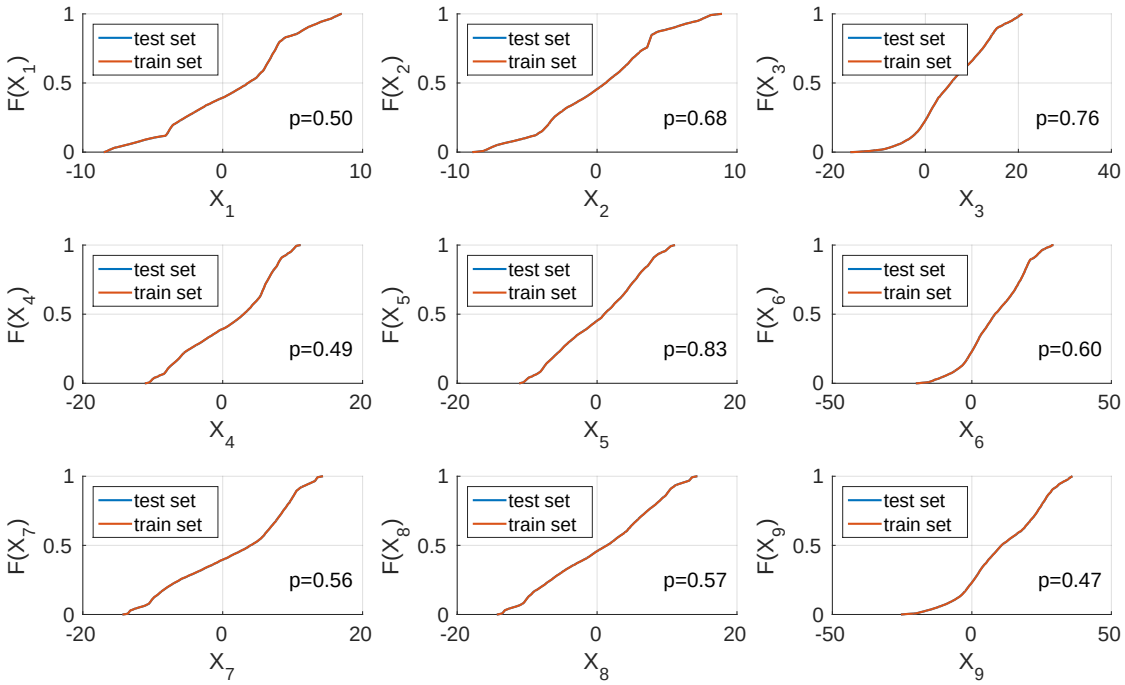


Figure 5.1: Comparison of the CDFs for each component of the training and the test data set of the complete data set. The p-values are obtained from a KS test.

The original data set is highly correlated due to the filtering of the previous two hit filter stage (see Fig. 5.2a). To decorrelate these data sets (see Sec. 4.4.1) the transformations are determined from the training data sets and subsequently applied to the test data sets. Due to the size of the training sets this should

only yield marginal differences compared to transformations determined from the complete sets [70].

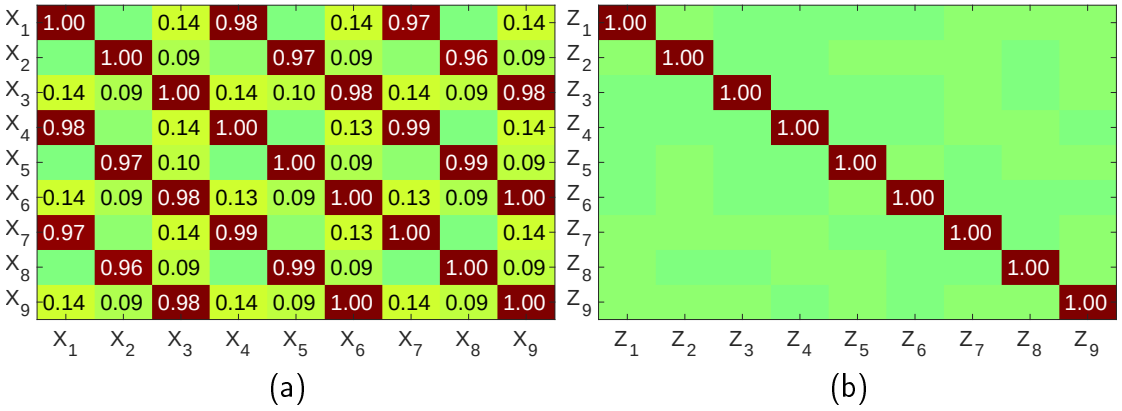


Figure 5.2: Correlation matrix of the complete test data set before (a) and after (b) decorrelation. Every entry that has no numerical inset has a correlation coefficient $\rho \leq 0.01$.

For determining the ML classifier parameters (Sec. 5.3–5.5) and for a detailed performance analysis (Sec. 5.6) only the consecutive set will be used. All three subsets will be used in Sec. 5.7, where possible benefits of incorporating the ML classifiers into the SectorMap are investigated.

5.3 Multilayer Perceptron Classifiers

To train and evaluate different MLPs MATLAB [71] is used, which offers a variety of different BP training algorithms. However, only the scaled conjugate gradient algorithm [60] is used. Nevertheless no significant difference to other algorithms is expected, since all BP algorithms implement a gradient descent (Sec. 4.2.4). The parameters that are varied are the number of hidden nodes N and the activation function of the output node. The activation functions of all hidden neurons are fixed to *tanh* and the activation functions of the input neurons are all *linear* (see Fig. 5.3). The activation function of the output neuron is either *logsig* or *linear* (Sec. 4.2).

In order to limit the number of necessary training runs, the size N of the hidden layer is restricted to $N \in \{10, 50, 100, 200\}$. As the number of necessary training steps tends to increase with increasing numbers of adjustable weights, initialization and decorrelation effects are only examined for $N = 50$.

To avoid overtraining MATLAB uses so called *Validation Stop*, which is a variation of hold out validation (Sec. 4.1.1). The training data set is split into a set which is actually used for training and a validation set, which is used to check the

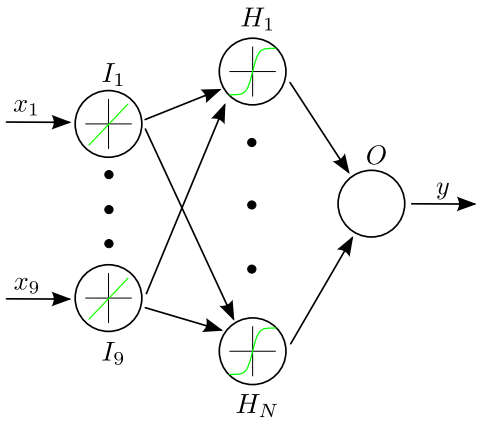


Figure 5.3: Basic structure of the MLPs used in this section with all fixed parameters.

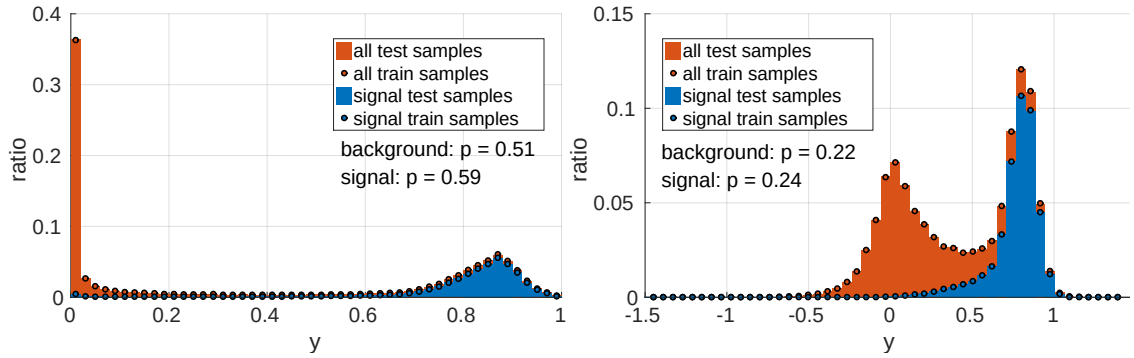
generalization capabilities of the MLP. Once the loss calculated from the output of the validation set stops to decrease or starts to increase the training is stopped. The validation set is chosen randomly from the training set and contains 15 % of all samples in the training data set.

5.3.1 Hidden Layer Size

As mentioned in Sec. 4.2.1, the size of the hidden layer determines the amount of information that can be stored by a MLP. In the previously described setup each additional hidden neuron adds $9 + 1 + 1 = 11$ adjustable weights: one weight for connecting each of the 9 inputs, one bias weight and one weight for the connection to the output. Although the computational complexity of evaluation and backpropagation is the same for standard backpropagation [59], a higher number of hidden neurons implies longer training, since more parameters have to be adjusted. It is thus desirable to find a hidden layer size with an acceptable performance while still being small enough for reasonable training times. Furthermore restricting the number of adjustable parameters reduces the risk of overtraining (Sec. 4.1.1).

Depending on the activation function of the outpout node the output y of an MLP is either limited to the interval $[0, 1]$ (*logsig*) or unbounded (*linear*) (see Fig. 5.4). To check if the MLP is overtrained, a KS test is performed comparing the output distributions of the training and test data sets. Due to the continuous performance checks during training (Sec. 5.3), no overtraining can be detected in any of the trained MLPs, even with $N = 200$ neurons in the hidden layer (see Fig. 5.4).

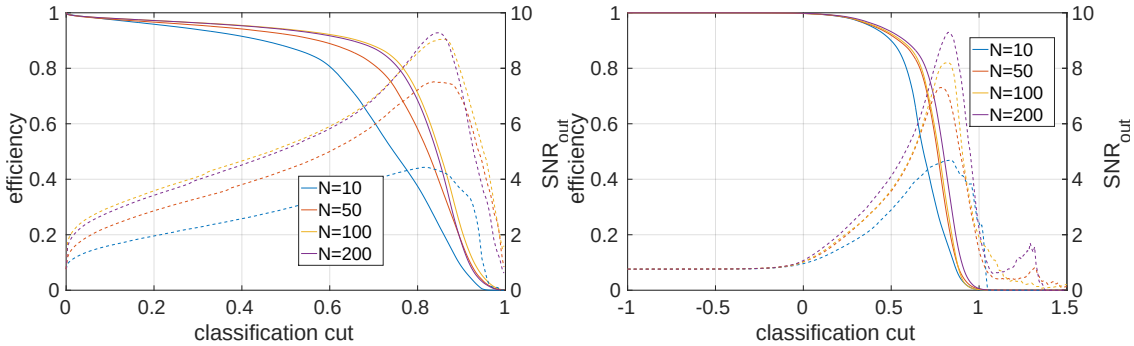
A well trained classifier produces an output distribution that can be easily divided into a signal and a background region by simply defining cut values. The desired performance is then achieved by tuning the cut value (see Fig. 5.5). At the lowest cut value all samples are classified as signal, implying an efficiency of 1 and no



(a) *logsig* (b) *linear*

Figure 5.4: Output of a MLP with $N = 200$ hidden neurons with a *logsig* output neuron (a) and a *linear* output neuron (b). The output distributions for the training and the test data set are compared via a KS test (p-value in plots).

discrimination capabilities. By increasing the cut value, the number of samples classified as signal decreases. This leads to a decrease of the efficiency as more and more signal samples are classified as background. However, the rejection of background samples leads to an increase of $\mathrm{SNR}_{\mathrm{out}}$, which reaches a maximum at a certain point. At approximately this point the output distribution of the signal samples reaches its maximum, whereas the tail of the output distribution of the background samples is approximately constant. Further increasing the cut value ultimately results in $r = 0$, as all samples are being rejected by the MLP (see Fig. 5.5). The visible fluctuations in $\mathrm{SNR}_{\mathrm{out}}$ at high cut values for the linear



(a) *logsig* (b) *linear*

Figure 5.5: Efficiency (solid lines, left y axis) and $\mathrm{SNR}_{\mathrm{out}}$ (dashed lines, right y axis) depending on the classification cut used for dividing the output distribution into a signal and a background region for different numbers of hidden neurons N for MLPs with a *logsig* output neuron (a) and a *linear* output neuron (b).

output node (see Fig. 5.5b) are due to the low occupancy of these bins and are of no interest in this thesis since the focus is on the high efficiency range.

While the maximum SNR gain that can be achieved is in the same range for both output nodes for all N, the efficiency where this maximum is reached differ (see Fig. 5.5 and Fig. 5.6a). Notably the SNR gain in the high efficiency range reaches only about a third of its maximum value for all MLPs. The MLP with a *logsig* output neuron with $N = 100$ hidden neurons shows the highest value (see Fig. 5.6b). While MLPs with a *linear* output neuron generally perform worse than its *logsig* counterparts this is reversed for the smallest hidden layer with $N = 10$. However, it is not possible to rule out initialization effects. MLPs with a *logsig* output neuron show better performance in the high efficiency range than MLPs with a *linear* output neuron, and their efficiency is slightly more robust to variations of the cut value.

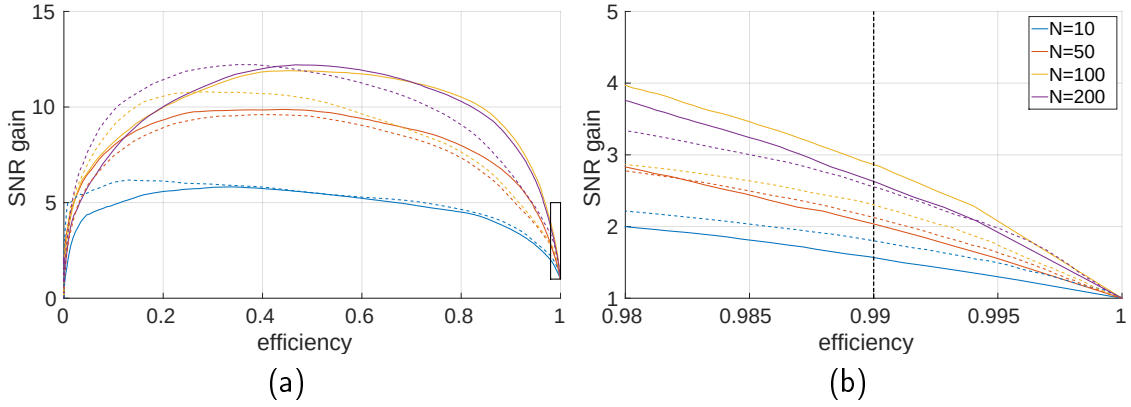


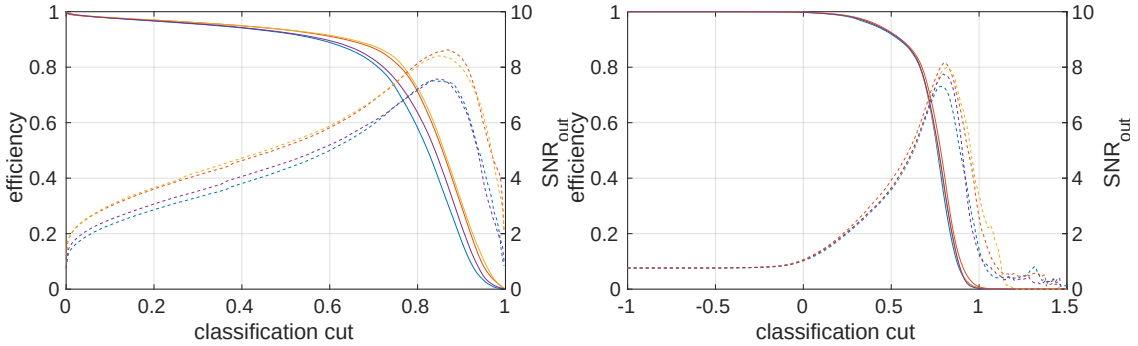
Figure 5.6: SNR gain vs. efficiency for different numbers of hidden neurons N and output functions: *logsig* (solid lines), *linear* (dashed lines). (a) shows the full efficiency range, (b) shows the high efficiency range only (rectangle in (a)).

5.3.2 Initialization Effects

As with every gradient descent method, the starting point strongly influences the minimum that can be reached. Hence, different random initializations, representing different starting points, are expected to yield different MLPs. A second source of randomness, the drawing of the validation samples from the training set, is considered to be part of the random initialization. The effects of differently initialized MLP classifiers are investigated with a fixed hidden layer size of $N = 50$ and four different initializations.

Comparing the efficiency and $\mathrm{SNR}_{\mathrm{out}}$ that can be reached by applying different classification cuts reveals that initialization effects have a considerable influence

on the performance. Especially for MLPs with a *logsig* output neuron the BP training reaches minima that yield MLP classifiers with a large performance spread. Although MLPs with a *linear* output neuron as well show a spread in performance for different initializations it is more confined compared to MLPs with a *logsig* output neuron (see Fig. 5.7).



(a) *logsig* (b) *linear*

Figure 5.7: Efficiency (solid lines, left y axis) and SNR_{out} (dashed lines, right y axis) depending on the cut value for different initializations for MLPs with $N = 50$ and a *logsig* (a) and a *linear* (b) output neuron.

These findings hold for the high efficiency range as well. While the achieved SNR gain at an efficiency of $r = 0.99$ is in the range of $\approx 2 - 2.8$ for MLPs with a *logsig* output neuron, the region covered by MLPs with a *linear* output neuron is confined to $\approx 2.2 - 2.5$ (see Fig. 5.8). Notably the best performing MLP performs almost as well as MLPs with $N = 100$ or $N = 200$ that have been tested previously in section 5.3.1 (see Fig. 5.6b).

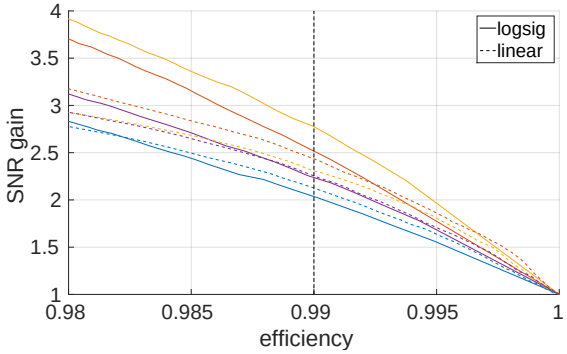


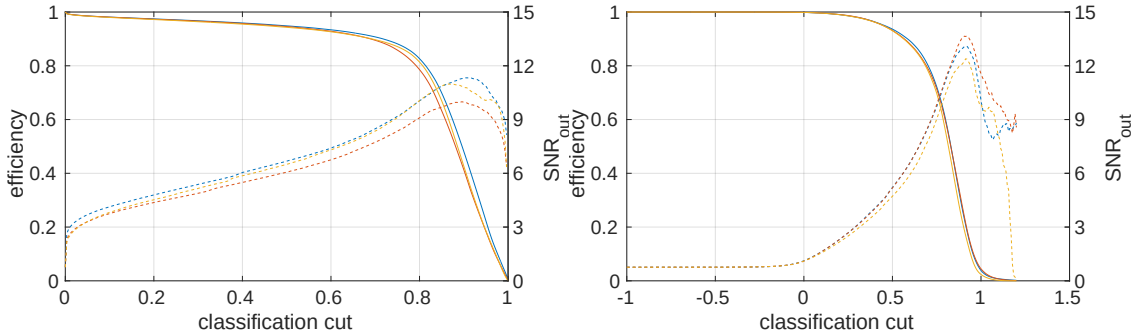
Figure 5.8: SNR gain vs. efficiency for different initializations for MLPs with $N = 50$ and *logsig* (solid lines) or *linear* (dashed lines) output neurons.

These results imply that finding the best performing MLP requires training with different initializations. Moreover, finding a suitable hidden layer size requires multiple trainings with different initializations for every number of hidden neurons to make comparisons viable.

5.3.3 Input Decorrelation

Although MLPs are in general able to handle correlated data, decorrelating the input data can alleviate the task of learning an appropriate mapping. To test the effects on the performance, MLPs with $N = 50$ hidden neurons have been trained with decorrelated input data. Furthermore initialization effects were studied by repeating the trainings with three different initializations.

Concerning the initialization effects the MLPs with a *logsig* output neuron still show more variation than the ones with a *linear* output neuron. However, compared to the original data (Sec. 5.3.2) they are reduced. More importantly, a considerable improvement can be seen in the achievable $\mathrm{SNR}_{\mathrm{out}}$ for all tested initializations and output neurons (see Fig. 5.9).



(a) *logsig* (b) *linear*

Figure 5.9: Efficiency (solid lines, left y axis) and $\mathrm{SNR}_{\mathrm{out}}$ (dashed lines, right y axis) depending on the cut value for different MLPs with $N = 50$ which have been trained with decorrelated data with a *logsig* (a) and a *linear* (b) output neuron.

Compared to the MLPs trained with the original (non-decorrelated) data, the SNR gain that can be reached at an efficiency of $r = 0.99$ is enhanced by a factor of about 1.5–1.7 for both kinds of output neurons (see Fig. 5.10b). Considering that decorrelating data is a computationally rather cheap operation, these results show that data decorrelation can be a powerful tool to enhance the classification capabilities of MLP classifiers. This can in turn be used to reduce the size of the hidden layer needed to accomplish a given performance.

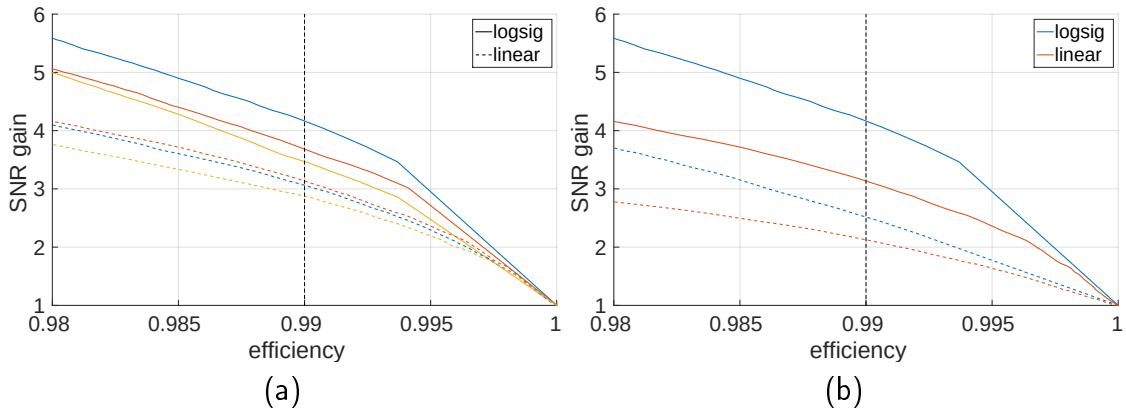


Figure 5.10: (a) SNR gain vs. efficiency for different initializations for MLPs with $N = 50$ trained with decorrelated data and *logsig* (solid lines) and *linear* (dashed lines) output neurons. (b) Comparison of the best performing MLPs from (a) (solid lines) and Fig. 5.8 (dashed lines)

5.4 Boosted Decision Tree Classifiers

To investigate the performance of BDTs two different boosting algorithms (Sec. 4.3.2) and parameters like depth of the DTs or number of splits in the DTs are assessed in two different approaches. BDTs employing *AdaBoost* are trained and evaluated with MATLAB which offers an easy to use implementation [72], whereas *Stochastic Gradient Boosting* is tested with the so called *FastBDT* implementation [19] that is part of the BASF2 externals. It provides an interface to `ROOT::TMVA` [73] but can also be used in a standalone version. Effects of different numbers of decision splits are examined with *AdaBoost* whereas effects of different tree depths are investigated with the FastBDT . If not stated otherwise, BDT always refers to the *AdaBoost* version, while FastBDT refers to the implementation of SGB in BASF2 in this section.

5.4.1 Number of Decision Splits

Due to long training times the number of decision splits S that are tested are limited to $S \in \{10, 20, 50\}$, and the learning rate is fixed to $\beta = 1$ (Sec. 4.3.2). Besides the number of decision splits in one tree, the number of trained trees in the ensemble N_B determines the performance of a BDT. The MATLAB implementation allows to choose the number of boosting steps to use for the evaluation after the training. Thus, it does not have to be chosen before training. However, to keep training times at an acceptable level, the maximum number is fixed to $N_B = 2000$ for the following tests.

To check at which number of boosting steps overtraining starts, the misclassification rate of the BDT evaluated on the test set is plotted against N_B. The misclassification rate is defined as the ratio of the number of samples that are misclassified to the number of samples that are classified correctly. The number of boosting steps where the misclassification rate starts to increase marks the onset of overfitting. This procedure differs from the *Validation Stop* (Sec. 5.3) approach only in the chosen validation set.

The misclassification rate monotonously decreases for all tested S, both for BDTs trained with original and decorrelated data implying that no trained BDTs is overfitting. However, plateaus of increasing length appear with increasing numbers of boosting steps (see Fig. 5.11). This indicates that further improvements require increasing numbers of additional boosting steps. BDTs with higher numbers of decision splits perform better and reach an "acceptable" level at lower N_B. Positive effects on the performance due to decorrelating the input data are observed in accordance to the findings for MLPs (Sec. 5.3.3).

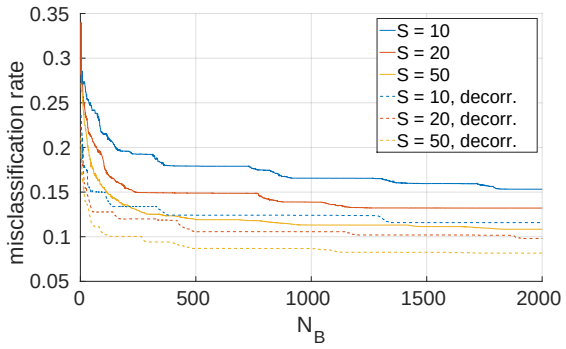
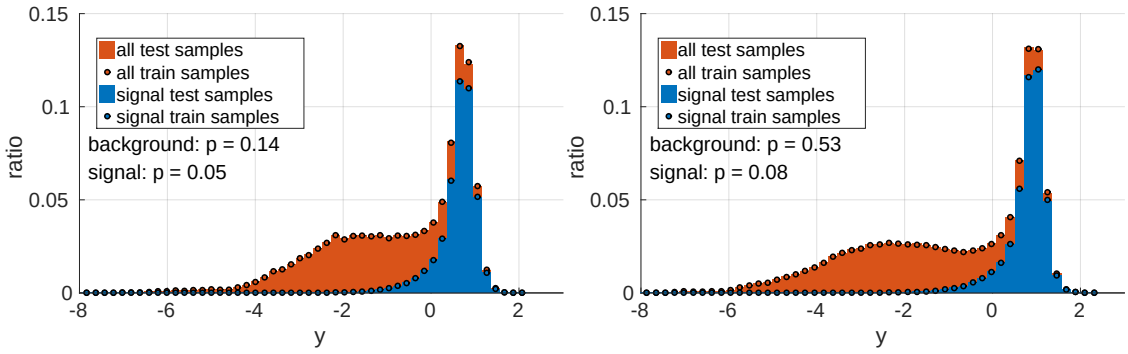


Figure 5.11: Misclassification rate vs. number of boosting steps N_B for BDTs with trees with different numbers of decision splits S trained with *AdaBoost*.

A crosscheck to exclude overtraining is performed by comparing the output distributions of the BDTs for the training and test data sets via a KS test (see Fig. 5.12). The p-values for $N_B = 2000$ are substantially lower compared to the p-values obtained for the output distributions for MLP classifiers (see Fig. 5.4), however overtraining can still be excluded at a 5 % significance level.

The output distribution of signal samples has its mode around $y \approx 1$. It features a very long tail reaching far into the negative numbers. The background output distribution is rather broad, featuring a tail into the negative numbers as well, but also reaching into the signal region quite prominently (see Fig. 5.12). Nevertheless a separation of signal and background maintaining a high efficiency is possible. As the tail of the signal distribution contains only a small fraction of all signal samples discarding a large fraction of background samples is possible.

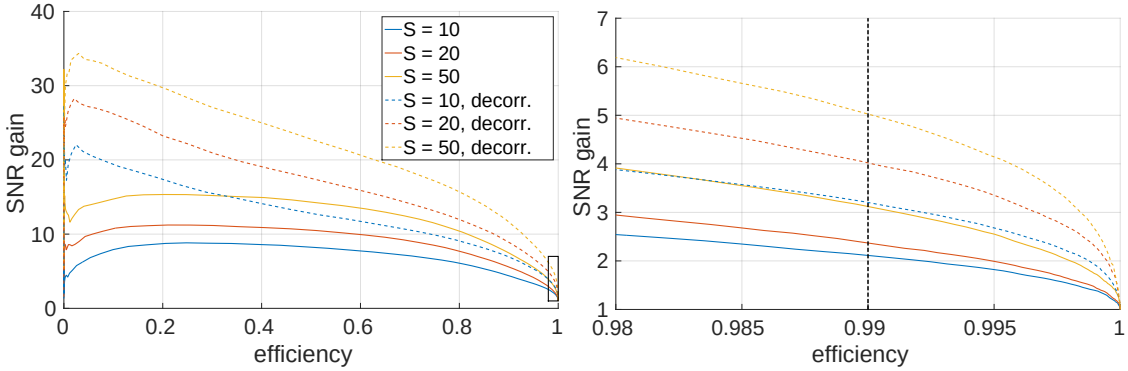
The SNR gain at $r = 0.99$ that can be achieved with BDTs trained with original data spans from ≈ 2.1 for $S = 10$ to ≈ 3.1 for $S = 50$. Training with decor-



(a) $S = 20, N_B = 2000$ (b) $S = 50, N_B = 2000$

Figure 5.12: Output of a BDT trained with *AdaBoost* on original data with different numbers of decision splits S and 2000 boosting steps. The output distributions for the training and the test data set are compared via a KS test.

related data yields a SNR gain between ≈ 3.1 for $S = 10$ and ≈ 5 for $S = 50$ (see Fig. 5.13b). Hence, the additional gain achieved by decorrelating is comparable to the MLP classifiers (Sec. 5.3.3). In contrast to the latter the maximum SNR gain is considerably larger for decorrelated data for BDTs although it is only achievable at very low efficiencies (see Fig. 5.13a).



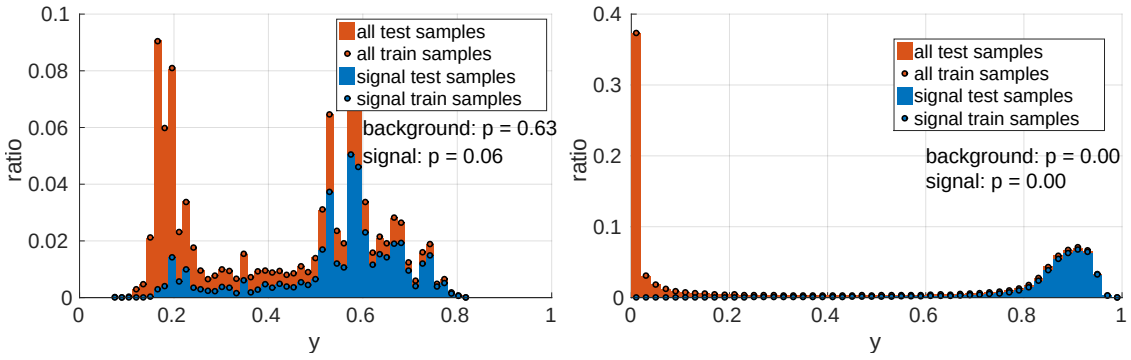
(a) (b)

Figure 5.13: SNR gain vs. efficiency for BDTs with different numbers of decision splits S and $N_B = 2000$ boosting steps. (a) full efficiency range and (b) high efficiency range (rectangle in (a)).

5.4.2 Number of Trees and Tree Depth in FastBDTs

The implementation of the FastBDT works with a fixed number of trees. Thus, the number of boosting steps that are tested are $N_B \in \{100, 500, 1000, 2000\}$. The range of tree depths that is investigated is $D \in \{2, 3, 4, 5, 6\}$, and the shrinkage parameter is fixed to $\nu = 0.1$. The fraction of the subset that is drawn randomly from the training set for the training of each tree is set to 0.5. Unlike for MLP classifiers these random effects are negligible. To check if a FastBDT classifier is overfitting, the output distributions of training and test set are compared via a KS test.

The output of a FastBDT is confined to the interval $y \in [0, 1]$. The separation of signal and background output distribution gets better with larger D and N_B. The mode of the background distribution tends towards $y = 0$ and its tail gets narrower with increasing D and N_B, whereas the signal distribution forms its maximum somewhere around $y \approx 0.75-0.9$, showing a tail to smaller numbers with decreasing width with increasing D and N_B (see Fig. 5.14). Decorrelating the input data does not significantly change these distributions, however, it helps to achieve a better separation for smaller D and N_B already.



(a) $D = 2, N_B = 100$ original data (b) $D = 6, N_B = 2000$ decorrelatd data

Figure 5.14: Output distributions of different FastBDT classifiers. A FastBDT with a small number of very shallow trees shows no real clustering of signal or background (a).

Checking the p values of KS tests comparing the output distributions of the training and the test data sets reveals that almost all FastBDT classifiers with $D \geq 4$ and $N_B \geq 500$ are overtrained, regardless of the preprocessing of the input data (see Fig. 5.15). Checking the high efficiency range reveals the effects of overtraining on the performance of FastBDT classifiers. Although the achievable SNR gain at $r = 0.99$ increases with increasing D and N_B for the training and the test data set, the difference between the two set gets larger (see Fig. 5.16).

Nevertheless the performance of the best FastBDT classifiers exceeds the performance of the previously discussed BDTs trained with *AdaBoost* (Sec. 5.4.1) and

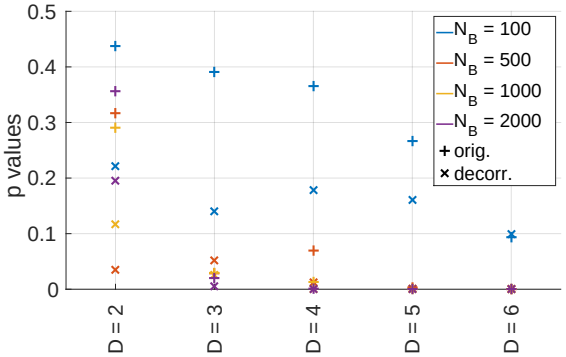


Figure 5.15: p values of KS test comparing the output distributions for the training and the test data set for different parameters of FastBDT classifiers. '+' trained on original data, '×' trained on decorrelated data.

MLPs (Sec. 5.3). The SNR gain that can be achieved by FastBDTs when trained with decorrelated input data reaches up to approx. 7.4 for $D = 6$ and $N_B = 2000$ (see Fig. 5.16b).

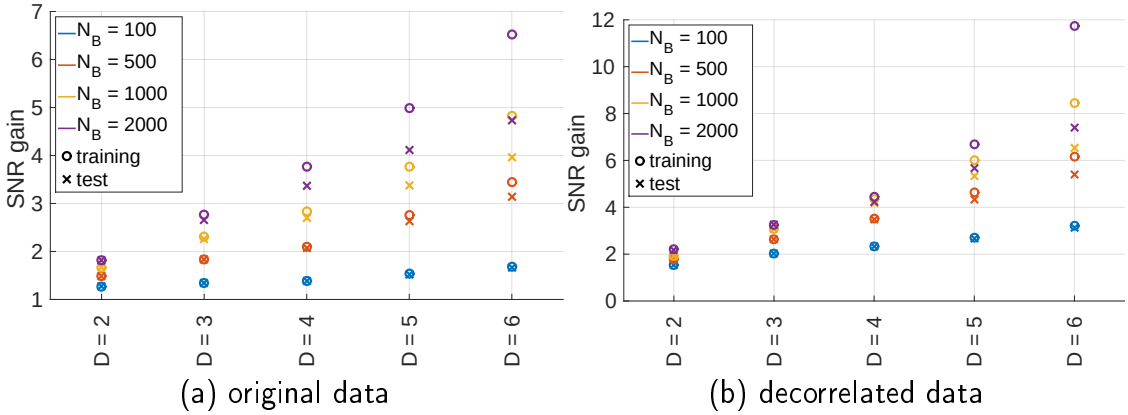


Figure 5.16: Comparison of achievable SNR gain at an efficiency of $r = 0.99$ for different parameters of FastBDT classifiers for the training '○' and the test '×' data set. The classification cuts to determine $r = 0.99$ have been determined on each set independently.

This overtraining scenario has some major implications on the extended training process. To ensure a given efficiency the classification cut can not be determined from the output distribution of the training set. While determining these cuts from the test data set imposes no practical problems, the original purpose of the test set is compromised as it has to be used in the training process.

5.5 Comparison of Classifiers

Apart from the classification performance, evaluation time is a crucial aspect in online track finding. Thus, the latter has to be considered as well to choose an appropriate classifier. To allow for an easier decision this section compiles the results of previously described classifiers (Sec. 5.3 and Sec. 5.4) into a global comparison and provides additional information on training and evaluation times for each category.

To reduce information to a digestible amount only the performance of certain classifiers of each category will be compared. These classifiers along with their specifics are:

- **MLP linear**: $N = 50$, linear output neuron
- **MLP logsig**: $N = 50$, logsig output neuron
- **BDT**: $S = 50$, $N_B = 2000$, *AdaBoost*
- **FastBDT**: $D = 6$, $N_B = 2000$, SGB

The comparison is done for both the original and the decorrelated data to illustrate the effects on the performance of decorrelating the input data. Initialization effects for MLPs are neglected however, and only the initialization leading to the best performance on the test set is chosen for the comparison.

5.5.1 Classification Performance

Directly comparing the different classifiers shows that the FastBDT classifier achieves the highest SNR gain at $r = 0.99$ for training with both decorrelated and original input data. All classifiers profit from decorrelating the input data before training, achieving approx. $1.3 - 1.6$ larger SNR gains (see Tab. 5.2 and Fig. 5.17).

	original	decorrelated
MLP logsig	2.67	4.05
MLP linear	2.41	3.10
BDT	3.11	4.91
FastBDT	4.73	7.40

Table 5.2: SNR gains at $r = 0.99$ for the presented classifiers with different preprocessing of the input data.

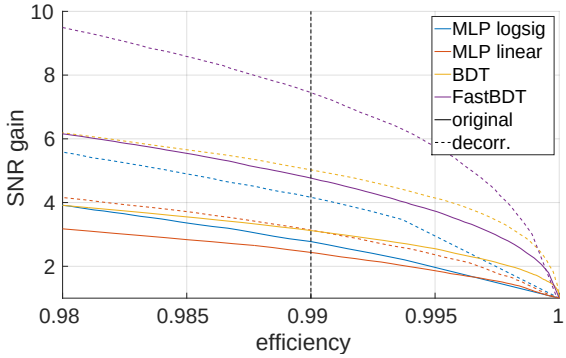


Figure 5.17: SNR gain vs. efficiency for different ML classifiers in the high efficiency range. Solid lines: classifier trained on original data, dashed lines: classifier trained on decorrelated data.

5.5.2 Evaluation and Training Times

Decorrelating the input data has no effect on computation times. Thus, evaluation times are only affected by the additional transformation that has to be applied to every input. Considering that it is only a relatively cheap multiplication of a constant matrix and a vector, no real restrictions are expected in online track finding. Hence, this transformation is neglected in this discussion. Training times are only affected indirectly by a possibly smaller number of necessary training or boosting steps.

Since only very few BDTs have been trained, not enough data points are present to make a reliable statement on training and evaluation times. Although more MLPs have been trained, training times are highly dependent on the random initialization. Hence, only approximate orders of magnitude and trends will be stated for these times. The MLP classifiers exhibit the lowest evaluation times and are faster by an approximate factor of 50 compared to the FastBDT classifiers (see Tab. 5.3). However, the times for MLPs were obtained from MATLAB and a direct comparison to an implementation in BASF2 is probably unfeasible. Nevertheless MLPs are the fastest classifiers, since their evaluation requires merely two matrix-vector multiplications. Such an operation can easily be parallelized.

classifier	time [μs/sample]	
	training	evaluation
MLP logsig, linear	$\sim (2-4)\cdot 10^3$	$\approx 2.1-2.4$
BDT	$\sim 10^4$	$\sim 10^3$
FastBDT	$\approx 500-600$	$\approx 100-120$

Table 5.3: Comparison of training and evaluation times per presented sample for the presented classifiers.

The number of trained FastBDTs is large enough to allow a statistical evaluation of training and evaluation times. Neglecting overhead, training times are about three times longer as evaluation times. The time τ a FastBDT needs to classify a sample depends on the number of boosting steps N_B and the tree depth D and can be calculated to (see Fig. 5.18):

$$\tau = 0.013 \cdot N_B \cdot D\,\mu\text{s}.$$

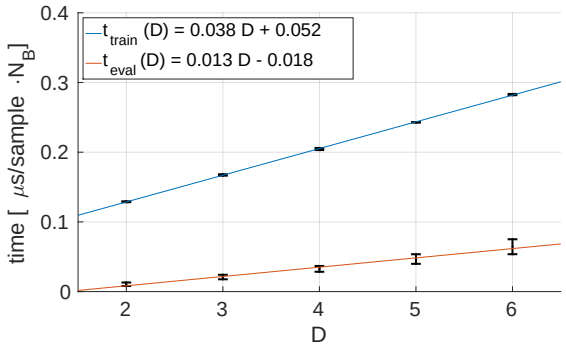


Figure 5.18: Training and evaluation times per sample and tree vs. tree depth for different parameters of a FastBDT with a linear fit to the data.

All of the above times have been taken by first collecting all samples and then presenting them to the classifiers. As only the presentation of the samples to the classifiers is timed, possible overhead is not considered. However, already this first estimate rules out BDTs for online track finding. The question whether FastBDTs are able to compensate their slower evaluation times by a better classification performance compared to MLPs remains open at this point.

5.6 Detailed Performance Analysis

The previous analysis in Sec. 5.3–5.5 compared the "overall" performance of the different classifiers only. While this allowed to easily compare different classifiers, a more detailed analysis has the prospect of revealing possible weak spots and of anticipating if the approach actually holds benefits ready compared to the current one.

All trained classifiers exhibit the same qualitative characteristics if the classification cut is determined to yield an overall efficiency of $r = 0.99$. The main difference is the ratio of rejected background. Thus, only the FastBDT with $D = 6$ and $N_B = 2000$, which shows the best overall performance (Sec. 5.5), will be used for presenting the following results.

5.6.1 Angle Dependent Performance

To determine the spherical coordinates (θ, ϕ) used in this discussion the innermost hit of a sample is used. While this is a reasonable choice to determine θ, it is somewhat arbitrary for ϕ. Nevertheless it provides an unambiguous definition.

Performance in bins of θ

The distribution of samples in θ shows a maximum at small θ values. This indicates that more track candidates are found in the forward direction of the detector, which is expected due to the asymmetry of the colliding beams (Sec. 2.1). However, most of these samples are background. Setting a classification cut such that the overall efficiency is $r = 0.99$, allows to discard the majority of the background while keeping almost all signal in the θ-acceptance range of the detector (see Fig. 5.19)

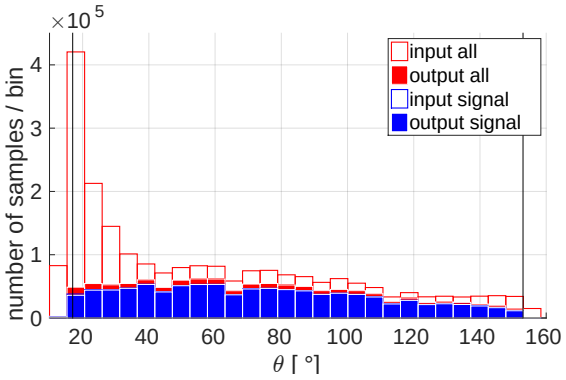


Figure 5.19: Distribution of classifier inputs and outputs in bins of θ for a classification cut that ensures an overall efficiency $r = 0.99$. The vertical black lines indicate the acceptance boundaries of the detector (Sec. 2.2).

Only at the very edges, close to the acceptance boundaries, the efficiency drops below 0.99. However, $r \geq 0.95$ can be achieved with this global classification cut (see Fig. 5.20b). The $\mathrm{SNR}_{\mathrm{in}}$ is small at the edges, but is almost constant with a value between 1.5 and 2 for $40° \lesssim \theta \lesssim 135°$. The same holds for the $\mathrm{SNR}_{\mathrm{out}}$ reaching values between approx. 5.8 and 7.5 in the same range of θ. Since there is more background to reject at the borders of the detector (see Fig. 5.19), the SNR gain reaches its largest values there, peaking at around 30 for $\theta \approx 18°$ (see Fig. 5.20b).

It is possible to set individual cuts for every bin to reach $r = 0.99$ in every bin. This results in lower $\mathrm{SNR}_{\mathrm{out}}$ at the edges, since a looser cut is required to reach the desired efficiency. For $40° \lesssim \theta \lesssim 135°$ higher $\mathrm{SNR}_{\mathrm{out}}$ can be achieved, since the cuts can be made stricter here (see Fig. 5.20a). While the SNR gain is not greatly affected by setting individual cuts for each bin in this middle range, it drops considerably at low values of θ (see Fig. 5.20b). Although the highest values are

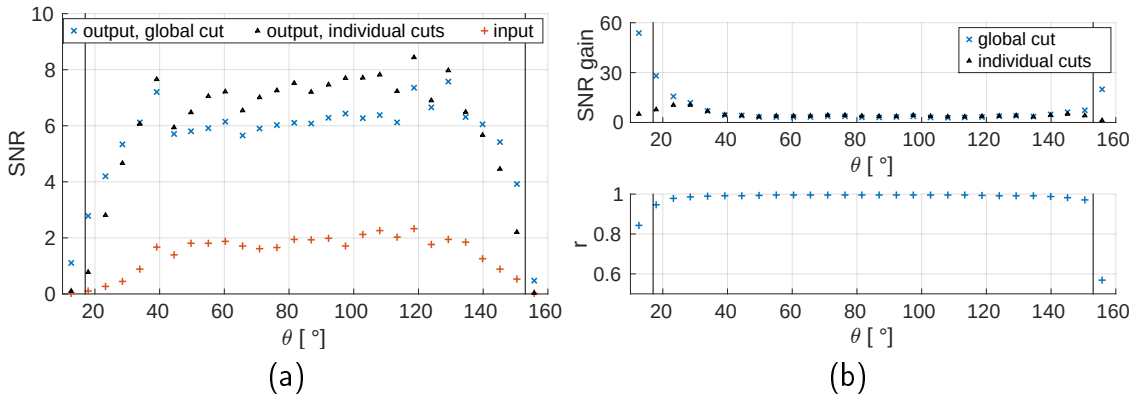


Figure 5.20: Comparison of SNR in input and output for a global classification cut and individual cuts for each bin (see text) (a). Comparison of achievable SNR gain for a global cut and individual cuts for each bin (b, top). Efficiency r for a global cut with overall efficiency $r = 0.99$ (b, bottom). The vertical black lines indicate the official detector boundaries (Sec. 2.2).

still reached there, the overall SNR gain drops from 7.4 with a global classification cut (see Tab. 5.2) to 5.9 with individual cuts in each bin. This degradation can be traced back to the high occupancy of the bins with small θ.

Performance in bins of ϕ

Naively a flat distribution of samples in ϕ would be expected. However, the distribution shows a broad peak around $\phi \approx 30°$ (see Fig. 5.21). The peak consists of background only and stems from the background simulation. This can be verified by using a particle gun with uniform angular distributions as simulation input. Comparing the distributions of simulations with and without added background reveals that the peak is not caused by wrongly tuned filters in the SectorMap, but actually by the added physics background.

The distribution of signal samples is almost flat, with dips at the overlapping parts of the detector. As all the samples that appear in the overlapping part of the detector are filtered out, this agrees with the expectation. Setting a global classification that ensures $r = 0.99$ results in an almost flat output distribution and a good background rejection in general (see Fig. 5.21). The efficiency is $r \geq 0.98$ for the whole range of ϕ with a tendency to smaller values for the ϕ-values with higher background ratios in the input (see Fig. 5.22b). The $\mathrm{SNR}_{\mathrm{out}}$ varies between approx. 4.5 and 6.5 and shows the same tendency to smaller values as the efficiency and the $\mathrm{SNR}_{\mathrm{in}}$ (see Fig. 5.22a). However, the SNR gain is increased in this ϕ range (see Fig. 5.22b), simply due to a lower $\mathrm{SNR}_{\mathrm{in}}$.

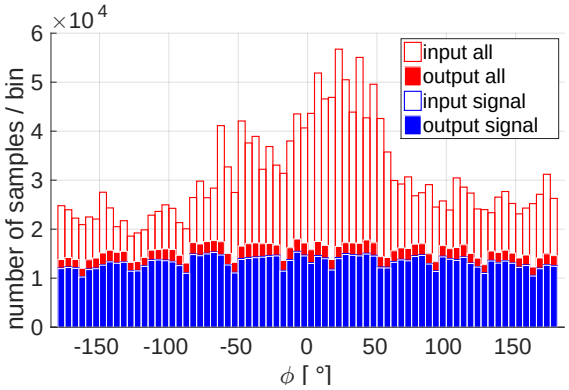


Figure 5.21: Distribution of classifier inputs and outputs in bins of ϕ for a classification cut that ensures overall $r = 0.99$.

Choosing individual cuts for each bin to have $r = 0.99$ in each bin has no significant effect. The $\mathrm{SNR}_{\mathrm{out}}$ and, as a consequence, the SNR gain, is increased in bins with $r > 0.99$ and decreased in bins with $r < 0.99$. However, since the cuts have to be adjusted only slightly, the overall SNR gain hardly changes (see Fig. 5.22).

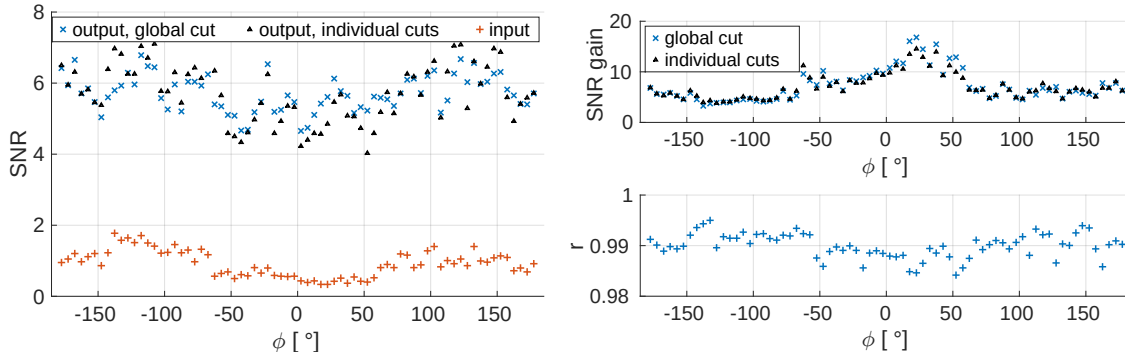


Figure 5.22: Comparison of SNR in input and output for a global classification cut and individual cuts for each bin (see text) (a). Comparison of achievable SNR gain for a global cut and individual cuts for each bin (b, top). Efficiency r for a global cut with overall efficiency $r = 0.99$ (b, bottom).

5.6.2 Momentum Dependent Performance

No MC information is available for background samples. Thus, only signal samples are used subsequently and only statements about the efficiency are possible.

As the VXD track finder is targeted at finding tracks over a wide momentum range, including low momenta down to 50 MeV, high efficiency is desired also for the latter. While setting individual cuts in different regions of the detector is the essence of the SectorMap approach, MC information dependent cuts are inadmissible. Hence, the classification cut is set to ensure $r = 0.99$ in the overall performance, and the efficiency depending on p and p_T is checked.

In the test set the lowest values are roughly 10 MeV/c for both p and p_T, and the highest values reach up to $p \approx 7\,\mathrm{GeV/c}$ and $p_T \approx 5.5\,\mathrm{GeV/c}$. The sample distributions have their maximum around $p_T \approx 400\,\mathrm{MeV/c}$ and $p \approx 500\,\mathrm{MeV/c}$ (see Fig. 5.23). The efficiency is stable at $r \geq 0.99$ in the high momentum range and drops below 0.99 at $p \approx 220\,\mathrm{MeV/c}$ and $p_T \approx 160\,\mathrm{MeV/c}$. However, it remains above 0.9 for $p \gtrsim 60\,\mathrm{MeV/c}$ and $p_T \gtrsim 45\,\mathrm{MeV/c}$ before dropping even further (see Fig. 5.23).

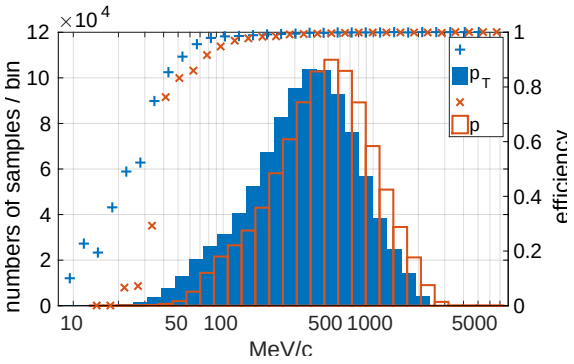


Figure 5.23: Distribution of signal samples (bars, left axis) and efficiency (right axis) depending on p and p_T.

Although only roughly 1–2% of all signal samples have $p < 60\,\mathrm{MeV/c}$ or $p_T < 45\,\mathrm{MeV/c}$, about 20 % are in a momentum range where $r < 0.99$. Setting a global classification cut that results in $r \geq 0.99$ in every bin of p and p_T results in an overall SNR gain of 6.2 and 5.7 respectively, compared to a value of 7.4 with an overall efficiency of $r = 0.99$ (see Tab. 5.2).

5.6.3 Charge and Particle Dependent Performance

From all particles produced in a collision and their decay products only few types actually reach the tracking volume before decaying; furthermore, only charged particles can be detected by the silicon detectors in the SVD. These particles are: electrons, positrons, muons, the charged pions and kaons as well as protons and anti-protons.

Multiple scattering and energy loss (Sec. 3.1) have the largest influence on how well a particle can be detected and tracked in the detector. Other properties like the

charge or the lifetime have further influence on this. A shorter lifetimes implies less traversed detector layers. Thus, less measurements are available for track finding and track fitting.

Apart from electrons, positrons and protons all particles have efficiencies $r \gtrsim 0.99$ if a global classification cut is set with overall $r = 0.99$. However, the efficiency never drops below 0.95 for any particle (see Fig. 5.24). The lower efficiencies of the e^- and the e^+ are related to their low mass and their likelihood of being secondary particles. The latter is also the source of the reduced efficiency for protons. The SectorMap is tuned to detect particles originating from the interaction point. Thus, secondary particles with decay vertices inside the detector volume naturally suffer from a lower detection efficiency.

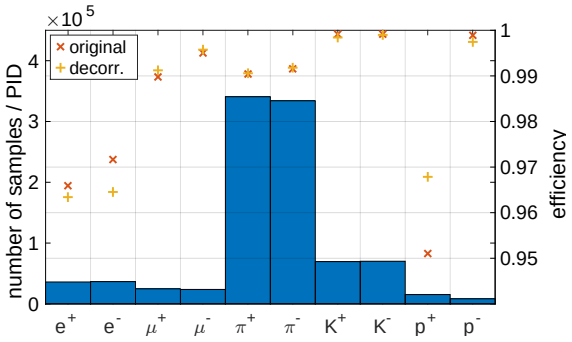


Figure 5.24: Efficiency ('×' and '+') reached with a classification cut that ensures an overall efficiency of $r = 0.99$ and number of samples for different particles that occurred in the SVD in the simulated events. p^- denotes the antiproton $\bar{p}$.

Combining all positively, resp. all negatively charged particles reveals that positively charged particles have a lower efficiency than negatively charged particles. The effect can be found independently of decorrelating the input data. The reason for the better performance for tracklets from negative particles can probably be linked to the detector performing differently for oppositely charged particles [74]. However, since the deviation is only marginal, choosing a slightly looser cut results in the desired efficiency of $r = 0.99$ without changing the achievable SNR gain significantly.

5.7 Towards a Combination of Approaches

Until now the SectorMap approach has only been used to reduce the amount of input data for the ML classifiers. However, possible benefits of further exploitation of the SectorMap are likely to emerge. Two possible approaches are investigated and compared to the previously described approach: 1) one global ML classifier is

used but the classification cut varies with the sector combination. 2) different ML classifiers are trained for different sector combinations.

As a first step both approaches are compared to a global ML classifier trained on the whole data set with a global classification cut. Since the full SectorMap information is not available for these standalone tests, the data set is split into three subsets according to table 5.1. The classification cuts for the first approach are determined such that $r = 0.99$ in every subset individually.

Compared to the global approach with only one global classification cut, individual cuts for the different subsets show only minor changes. The slight degradation of the output SNR compared to the global cut is due to the imbalanced number of samples in the subsets (see Tab. 5.1). While the largest subset, the *consecutive* set, surpasses $r = 0.99$ only slightly, the other two subsets, *overlapping* and *skipping*, reach only $r \approx 0.97$. The stricter cut that can be chosen on the consecutive set cannot compensate the worse background rejection that is necessary to increase the efficiency in the other two sets sufficiently. However, training different classifiers on the three different subsets significantly improves background rejection while ensuring the desired efficiency (see Fig. 5.25).

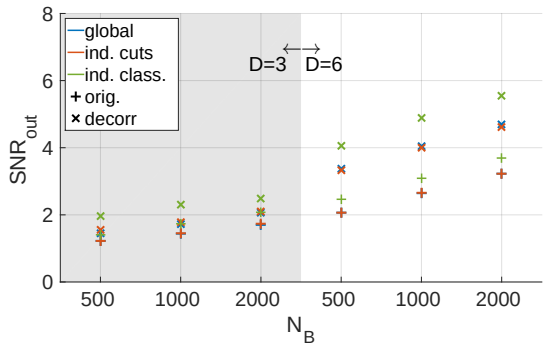


Figure 5.25: Comparison of SNR_{out} for different approaches (see text) utilizing more of the information available from the SectorMap. The used ML classifiers are different FastBDTs with different data preprocessing ('×' decorrelated, '+' original input data).

Although splitting the detector into only three regions is far from utilizing all available information, some conclusions still can be drawn. First of all, training different classifiers for different sector combinations significantly improves the performance. Defining different cuts for different sector combinations affects the performance only weakly. Both approaches share the prospect of having a finer control over the achieved efficiencies compared to a classifier with one global cut.

Determining individual cut values for the different sector combinations can be achieved rather easily and imposes no additional overhead to the training and data collection scheme. On the other hand, having different classifiers for different sector combinations requires the development of an appropriate scheme for training and

data collection. However. only an implementation into the SectorMap approach is able to reveal how serious the impact of these drawbacks actually is.

6 Conclusion and Outlook

In the thesis presented here the general feasibility of employing supervised machine learning techniques to track finding in the Belle II Silicon Vertex Detector have been investigated. Using machine learned classifiers to filter three-hit combinations in the SectorMap has shown some promising results.

Due to the prefiltering of the SectorMap the inputs to the classifiers are highly correlated. It has been shown that preprocessing the input data by applying a transformation to linearly decorrelate the inputs is able to improve the performance of the tested classifiers significantly. Given that this transformation is a computationally rather cheap operation and can in principle be even incorporated into a MLP classifier, the implementation of a decorrelation step seems mandatory to achieve the desired results.

Testing the angle dependent performance of the classifiers has not revealed any weak spots. The investigation of the momentum dependent performance shows that a high efficiency can be obtained even for low momenta. Moreover, the desired efficiency of $r = 0.99$ can be reached with only a slight degradation of the classification performance. Thus, it is reasonable to expect that the performance of the current VXD track finding algorithm can be improved for the low momentum range by using machine learning techniques.

To test how the SectorMap approach and the machine learned filters approach can be combined, the performance of one global classifier has been compared to two possible approaches that might be able to exploit the advantages of both: specific cuts and individual classifiers for different regions of the detector. Both approaches allow for a finer grained control over the efficiency throughout the detector compared to using only one global cut. While specific cuts can easily be implemented into the current track finder, they show only minor differences to the performance of the global classifier. However, using as few as three different individual classifiers improves the performance significantly. Thus, it might be possible to exchange the large number of simple filters that are currently in place by a small number of sophisticated machine-learned filters.

Nevertheless it has to be stated that these results were not obtained with an implementation into the current track finder and have thus to be treated with caution. First of all, it is not guaranteed that the way the samples for training and testing were obtained actually reflects the situation that is encountered in track finding. Second, it is not clear how the obtained results translate into the tracking performance. However, a direct comparison of the performance of the current track

finder and the machine-learned filters was not possible because of a major redesign of the former's implementation.

Thus, the logical next step is an implementation into the framework once the redesign is finished. Once this implementation is done it will be possible to draw conclusions on how the results of this thesis translate into tracking performance. Furthermore only an implementation is able to reveal how the time constraints posed by the requirement of on-line track finding influence the choice of the appropriate classifying technique.

A Appendix

A.1 Shortcomings of linear activation functions

Consider a network consisting of m input neurons (with an arbitrary activation function $S_{in}(z)$), n neurons in on hidden layer with linear activation functions $S(z) = z$, and an arbitrary number of output neurons with an arbitrary activation function $S_{out}(z)$.

Given an input vector $\vec{x} \in \mathbb{R}^m$ the outputs of the i-th input neuron $a_i = S_{in}(x_i)$, hence the outputs of the hidden neurons is

$$h_k = \sum_{i=1}^{m} w_{ik} a_i, \qquad k = 1...n, \tag{A.1}$$

where w_{ik} are the weights connecting the m input neurons with the n hidden layers. The output value of a neuron in the output layer o can now be expressed as

$$o = S_{out}(\sum_{k=1}^{n} w_{ko} h_k) = S_{out}(\sum_{k=1}^{n} w_{ko} \sum_{i=1}^{m} w_{ik} a_i). \tag{A.2}$$

The hidden linear neurons are now easily removed by connecting the inputs directly to the output o with the new weights

$$w_{io} = w_{ko} \sum_{k=1}^{n} w_{ik} \tag{A.3}$$

resulting in an output value of

$$o = S_{out}(\sum_{i=1}^{m} w_{io} a_i), \tag{A.4}$$

which is equivalent to the original network containing the hidden neurons.

Acknowledgments

The list of people that have contributed in helping me finish this thesis is long and I probably will not be able to get it complete here. However, I would like to particularly mention a few that played an important role.

First of all, I would like to thank ***Jochen Schieck***, ***Rudolf Frühwirth***, and ***Christoph Schwanda*** for offering me the opportunity to do my thesis on such an interesting project like the Belle II experiment. With their untiring support and guidance they helped me to complete this thesis.

I would like to emphasize the great support I experienced from ***Rudolf Frühwirth***. He left none of my numerous questions unanswered and with his calm nature guided me and my work through some patches of rough water that came along in the process.

Many thanks I would like to bring out for my office neighbor and person in charge for the implementation of the VXDTF, ***Jakob Lettenbichler***. Despite his own very full agenda, he patiently introduced me to BASF2 until I was able to walk on my own feet before he encountered with me in many fruitful discussions, providing me with new approaches to many problems.

I owe a thank you as well to all of the Institute of High Energy Physics of the Austrian Academy of Sciences. You made this year an inspiring journey and never have I regretted to have chosen the path of high energy physics.

Another thank you is appropriate for the tracking group of the Belle II collaboration for their kindness and swiftness in answering any software or tracking related questions.

Transitioning to a more personal level, I would like to thank all of my fellow students at the TU Wien. You have indisputably made the last six years of my physics studies an unforgettable time. For productive discussions about physics and life in general but also for the probably most unproductive but funniest lunch breaks, I would like to thank: ***Philipp Moser***, ***Lukas Semmelrock***, ***Wolfgang Moser***, ***Jakob Fellinger***, ***Matthias Müllner***, ***Andreas Renner*** and ***Willi Grosinger*** – just to name a few.

A special thanks to a member of this group I would like to transmit to ***Johannes Brandstetter***, for assisting me with great personal advice whenever I needed it.

I would like to thank as well ***Harald Triebnig***, ***Laurin Schwarzmann*** and especially my best friend ***Matthias Köb*** for balancing the scales of live towards things outside the physics world, for providing me with honest opinions and advice and for making my years of study in Vienna one of the best times of my life.

And finally, the ones with the possibly largest share towards my success so far, my family: Thank You ***Mama***, Thank You ***Däta***, Thank You ***Sarah***, Thank You ***Christoph***. For your everlasting support in every area of life, for being always prepared for a more or less sudden visit and in general just for being there whenever I need you.

Thank You!

Glossary

ANN Artificial Neural Networks

ARICH Aerogel Ring Imaging Cherenkov Detector, see Sec. 2.2.3

BASF Belle AnalysiS Framework

BASF2 Belle AnalysiS Framework 2, Software used at Belle II, see Sec. 2.3

barn unit of area used to measure cross sections, $1\,\mathrm{b} = 10^{-24}\,\mathrm{cm}^2$

BDT Boosted Decision Trees, see Sec. 4.3

BP Backpropagation, algorithm for training MLPs, see Sec. 4.2.4

BSM Physics Beyond the Standard Model, see NP

Belle Physics experiment at KEKB

Belle II Physics experiment at SuperKEKB

C++11 version of the C++ programming language, a general-purpose programming language widely spread in high energy physics

C Charge conjugation, exchanges particles with anti-particles

CA Cellular Automaton, see Sec. 2.4.2

CDC Central Drift Chamber, see Sec. 2.2.2

CDF Cumulative Distribution Function

CERN European Organization for Nuclear Research, Geneva

CKF Combinatorial Kalman Filter, a numerically more robust version of the KF

CKM matrix Cabibbo-Kobayashi-Maskawa matrix

CMS Center of Mass System

CP Combination of charge conjugation and subsequent parity transformation

CPT Combination of charge conjugation, parity transformation and time reversal. This is the only symmetry that is observed to be exact in the SM.

DEPFET DEpleted P-channel Field Effect Transistors

DSSD Double Sided Silicon Strip Detectors

DT Decision Trees, see Sec. 4.3.1

ECL Electromagnetic Calorimeter, see Sec. 2.2.4

EvtGen Library for event generation used for MC simulation of B-meson decays in BASF2 [20]

FFN Feedforward Neural Network

Geant4 Library for simulating the passage of particles through matter [21]

HER High Energy Ring

HLT High Level Trigger

HNN Hopfield Neural Network

instantaneous luminosity L is a number that measures a particle accelerators ability to produce a required number of interactions R. It connects the rate of events dR/dt with the cross section of a given process σ_p, $dR/dt = L \cdot \sigma_p$

integrated luminosity is the time integral of the instantaneous luminosity $\int Ldt$ and a measure for the amount of data collected by an experiment. It is usually measured in *inverse barn*. It can be used to calculate the expected number of measurements of a process given its cross section. If the process has a cross section $\sigma_p = 1\,\text{fb}$ and the integrated luminosity is $1\,\text{fb}^{-1}$ (on average) one event has been collected for this process

IP Interaction Point

IR Interaction Region

KEK *Kō Enerugī Kasokuki Kenkyū Kikō*, The High Energy Accelerator Research Organization. A national organization operating the largest particle physics laboratory in Japan.

KEKB Collider used in the Belle experiment, located at the KEK facility in Tsukuba, Japan

KF Kalman Filter, a linear and locally linear estimator, equivalent to to the global least squares method, see Sec. 2.4.3

KLM K-Long and Muon Detector, see Sec. 2.2.5

KS test Kolmogornov-Smirnov test, statistical test for deciding if two samples are drawn from the same distribution

LER Low Energy Ring

LHC Large Hadron Collider, particle accelerator at CERN

Luminosity Measure for the expected rate of events per time, see also instantaneous and integrated luminosity

MC Monte Carlo, a method for simulating random processes using pseudo-random numbers

MCParticle a simulation object in BASF2 that holds information from the MC simulation which can be used to calibrate and test different parts of the software

ML Machine Learning

MLP Multilayer Perceptrons, see Sec. 4.2.2

NP New Physics, term coined for subsuming different theories that try to explain phenomena that cannot be described by the SM

P Parity Transformation: $P : \vec{x} \to -\vec{x}$

PID Particle Identification or Particle Identity

PMT Photo Multiplier

PXD PiXel Detector, see Sec. 2.2.1

Particle Gun Software tool to generate particles with predefined properties at any given position in the detector simulation. Can be used to produce clean events with any desired properties.

Python general-purpose programming language, see http://www.python.org

QCD Quantum Chromodynamics, the theory describing the strong interaction

QED Quantum Electrodynamics, the relativistic quantum field theory of electrodynamics

QI Quality Index

ROI Region(s) of Interest

ROOT Library for data analysis heavily used in particle physics, developed at CERN [22]

RPC Resistive Plate Chamber

SCG Scaled Conjugate Gradient, algorithm for solving systems of linear equations numerically

SGB Stochastic Gradient Boosting, algorithm for building an ensemble of DTs, see Sec. 4.3.2

SM Standard Model of Particle Physics

SNR Signal-to-Noise Ratio

SR Synchrotron Radiaton

SVD Silicon Vertex Detector, see Sec. 2.2.1

SuperKEKB major upgrade of the KEKB B-Factory, collider used in the Belle II experiment

T Time Reversal, $T : t \to -t$

TC Track Candidate

TOP Time of Flight Propagation Counter, see Sec. 2.2.3

tracklet Part of a particle track containing only a small number of hits

TRISTAN Particle accelerator experiment, predecessor of KEKB

vertex Origin of a particle trajectory

VXD VerteX Detector, see Sec. 2.2.1

VXDTF VXD Track Finder, see Sec. 2.4

Bibliography

[1] A. D. Sakharov. *Violation of CP Invariance, c Asymmetry, and Baryon Asymmetry of the Universe.* Pisma Zh. Eksp. Teor. Fiz. **5**, 32 (1967).

[2] K. Abe *et al. Measurement of time dependent CP violating asymmetries in B0 → phi K0(s), K+ K- K0(s), and eta-prime K0(s) decays.* Phys. Rev. Lett. **91**, 261602 (2003).

[3] K. Abe *et al. Observation of Large CP Violation in the Neutral B Meson System.* Phys. Rev. Lett. **87**, 091802 (2001).

[4] L. Deng and X. Li. *Machine learning paradigms for speech recognition: An overview.* IEEE Transactions on Audio, Speech, and Language Processing **21**, 1060 (2013).

[5] C. M. Bishop. *Pattern recognition and machine learning.* Springer (2006).

[6] G. Zhang, B. E. Patuwo, and M. Y. Hu. *Forecasting with artificial neural networks: The state of the art.* International Journal of Forecasting **14**, 35 (1998).

[7] M. Gardner and S. Dorling. *Artificial neural networks (the multilayer perceptron)-a review of applications in the atmospheric sciences.* Atmospheric Environment **32**, 2627 (1998).

[8] B. P. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor. *Boosted decision trees as an alternative to artificial neural networks for particle identification.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **543**, 577 (2005).

[9] X. Chen and L. Xia. *Search for Flavor Changing Neutral Current in $t \to Hc, H \to \tau\tau$ Decay at the LHC.* arXiv:1509.08149 **[hep-ph]** (2015).

[10] C. Schwanda. *SuperKEKB machine and Belle II detector status.* Nuclear Physics B - Proceedings Supplements **209**, 70 (2010). Proceedings of the Third Workshop on Theory, Phenomenology and Experiments in Heavy Flavour Physics.

[11] J. P. Wiechczynski. *The Belle II experiment at the SuperKEKB collider.* In *EPS HEP 2015, Vienna* (2015).

[12] T. Abe, I. Adachi, K. Adamczyk, S. Ahn, H. Aihara, K. Akai, M. Aloi, L. Andricek, K. Aoki, Y. Arai, and et al. *Belle II Technical Design Report.* ArXiv e-prints (2010).

[13] A. Moll. *The Software Framework of the Belle II Experiment.* Journal of Physics: Conference Series **331**, 032024 (2011).

[14] A. Abashian *et al. The Belle detector.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **479**, 117 (2002). Detectors for Asymmetric B-factories.

[15] S. Kurokawa and E. Kikutani. *Overview of the KEKB accelerators.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **499**, 1 (2003). KEK-B: The KEK B-factory.

[16] *online* (retrieved: Sep 2015). http://belle2.desy.de/sites2009/site_belle2/content/e103206/e103207/SuperKEKB:BelleII.jpg.

[17] C. Pulvermacher. *dE/dx particle identification and pixel detector data reduction for the Belle II experiment.* Master's thesis, KIT (2012). KIT, Diplomarbeit, 2012.

[18] KEK High energy accelerator research organization. *New electronics tested for Belle II central drift chamber.* http://www2.kek.jp/proffice/archives/feature/2010/pdf/BelleIICDCDesign.pdf (2010).

[19] T. Keck. *The Full Event Interpretation for Belle II.* Ms, Karlsruher Institut für Technologie (KIT) (2014). Karlsruher Institut für Technologie (KIT), Masterarbeit, 2014.

[20] A. Ryd, D. Lange, N. Kuznetsova, S. Versille, M. Rotondo, D. P. Kirkby, F. K. Wuerthwein, and A. Ishikawa. *EvtGen: A Monte Carlo Generator for B-Physics* (2005).

[21] S. Agostinelli *et al. GEANT4: A Simulation toolkit.* Nucl. Instrum. Meth. **A506**, 250 (2003).

[22] I. Antcheva *et al. ROOT - A C++ framework for petabyte data storage, statistical analysis and visualization.* Computer Physics Communications **180**, 2499 (2009).

[23] J. Lettenbichler. *Pattern recognition in the Silicon Vertex Detector of the Belle II experiment.* Masters thesis, University of Vienna, Austria (2012).

[24] A. Strandlie and R. Frühwirth. *Track and vertex reconstruction: From classical to adaptive methods.* Review of Modern Physics **82** (2010).

[25] R. Frühwirth. *Application of Kalman filtering to track and vertex fitting.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **262**, 444 (1987).

[26] R. Frühwirth. *Selection of optimal subsets of tracks with a feed-back neural network.* Computer Physics Communications **78**, 23 (1993).

[27] R. Frühwirth, R. Glattauer, J. Lettenbichler, W. Mitaroff, and M. Nadler. *Track finding in silicon trackers with a small number of layers.* Nucl. Instrum. Meth. **A732**, 95 (2013).

[28] T. Toffoli and N. Margolus. *Cellular automata machines: a new environment for modeling.* MIT press (1987).

[29] W. N. Cottingham and G. D. A. *An Introduction to the Standard Model of Particle Physics.* Cambridge University Press, Cambridge, UK, 2 edn. (2007).

[30] C. Burgess and G. Moore. *The standard model: A primer.* Cambridge University Press (2006).

[31] K. A. Olive *et al. Review of Particle Physics.* Chin. Phys. **C38**, 090001 (2014).

[32] S. Chatrchyan *et al. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC.* Phys. Lett. **B716**, 30 (2012).

[33] G. Aad *et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC.* Phys. Lett. **B716**, 1 (2012).

[34] J. Ellis and T. You. *Updated Global Analysis of Higgs Couplings.* JHEP **06**, 103 (2013).

[35] *online* (retrieved: Sep. 2015) https://sciencenode.org/spotlight/go-particle-quest-first-cern-hackfest.php (2012).

[36] M. Kobayashi and T. Maskawa. *CP Violation in the Renormalizable Theory of Weak Interaction.* Prog. Theor. Phys. **49**, 652 (1973).

[37] A. J. Bevan, B. Golob, T. Mannel, S. Prell, B. D. Yabsley, H. Aihara, F. Anulli, N. Arnaud, T. Aushev, M. Beneke, and et al. *The Physics of the B Factories.* European Physical Journal C **74**, 3026 (2014).

[38] T. Aushev *et al. Physics at Super B Factory.* arXiv:1002.5012 **[hep-ex]** (2010).

[39] S. Sugihara. *Background Estimation at SuperKEKB by machine study* (2010). Belle II internal note.

[40] L. Piilonen. *Touschek Background in the Barrel KLM* (2011). Belle II internal note.

[41] E. Nedelkovska. *Estimation of the two-photon QED background at Belle II* (2011). Belle II internal note.

[42] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edn. (2003).

[43] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge (1998).

[44] D. W. Patterson. *Artificial Neural Networks: Theory and Applications.* Prentice Hall (1996).

[45] R. Callan. *The Essence of Neural Networks.* Prentice Hall Europe (1999).

[46] L. Breiman, J. H. Friedman, C. J. Stone, and R. Olsen. *Classification and Regression Trees (Wadsworth Statistics/Probability).* Chapman and Hall/CRC, 1 edn. (1984).

[47] S. Kotsiantis. *Supervised Machine Learning: A Review of Classification Techniques.* Informatica **31**, 249 (2007).

[48] R. Caruana and A. Niculescu-Mizil. *An Empirical Comparison of Supervised Learning Algorithms.* In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 161–168. ACM, New York, NY, USA (2006).

[49] R. Lippmann. *An introduction to computing with neural nets.* ASSP Magazine, IEEE **4**, 4 (1987).

[50] R. Lippmann. *Pattern classification using neural networks.* Communications Magazine, IEEE **27**, 47 (1989).

[51] J. H. Friedman. *Stochastic gradient boosting.* Computational Statistics & Data Analysis **38**, 367 (2002). Nonlinear Methods and Data Mining.

[52] W. McCulloch and W. Pitts. *A logical calculus of the ideas immanent in nervous activity.* The bulletin of mathematical biophysics **5**, 115 (1943).

[53] D. O. Hebb. *The Organization of behavior: A neuropsychologichal approach.* Wiley, New York (1949).

[54] M. Minsky and P. Seymour. *Perceptrons.* MIT press (1969).

[55] G. Cybenko. *Approximation by superpositions of a sigmoidal function.* Mathematics of Control, Signals and Systems **2**, 303 (1989).

[56] K. Hornik, M. Stinchcombe, and H. White. *Multilayer feedforward networks are universal approximators.* Neural Networks **2**, 359 (1989).

[57] G.-B. Huang, Y.-Q. Chen, and H. Babri. *Classification ability of single hidden layer feedforward neural networks.* Neural Networks, IEEE Transactions on **11**, 799 (2000).

[58] K. Hornik. *Approximation capabilities of multilayer feedforward networks.* Neural Networks **4**, 251 (1991).

[59] D. E. Rumelhart, G. E. Hinton, and W. R.J. *Learning Internal Representations by Error Propagation.* In D. E. Rumelhart and J. McClelland (editors), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. MIT Press, Cambridge, MA (1986).

[60] M. F. Møller. *A scaled conjugate gradient algorithm for fast supervised learning.* Neural Networks **6**, 525 (1993).

[61] S. K. Murthy. *Automatic construction of decision trees from data: A multi-disciplinary survey.* Data mining and knowledge discovery **2**, 345 (1998).

[62] L. Hyafil and R. L. Rivest. *Constructing optimal binary decision trees is NP-complete.* Information Processing Letters **5**, 15 (1976).

[63] R. Schapire. *The strength of weak learnability.* Machine Learning **5**, 197 (1990).

[64] Y. Freund, R. E. Schapire *et al. Experiments with a new boosting algorithm.* In *ICML*, vol. 96, pp. 148–156 (1996).

[65] R. E. Schapire and Y. Singer. *Improved Boosting Algorithms Using Confidence-rated Predictions.* In *Machine Learning*, pp. 80–91 (1999).

[66] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* New York: Springer (2009).

[67] Y. Dodge. *The Oxford dictionary of statistical terms.* Oxford University Press, Oxford New York (2003).

[68] S. Neubauer. *Search for $B \to K^{(*)}\nu\bar{\nu}$ Decays Using a New Probabilistic Full Reconstruction Method.* Ph.D. thesis, Karlsruher Institut für Technologie (KIT) (2011).

[69] F. J. Massey Jr. *The Kolmogorov-Smirnov test for goodness of fit.* Journal of the American statistical Association **46**, 68 (1951).

[70] R. Frühwirth. *Private Conversation* (2015).

[71] *MATLAB and Neural Network Toolbox Release 2015a.* Natick, Massachusetts, United States, The MathWorks, Inc.

[72] *MATLAB and Statistics and Machine Learning Toolbox Release 2015a.* Natick, Massachusetts, United States, The MathWorks, Inc.

[73] A. Hoecker *et al.* *TMVA - Toolkit for Multivariate Data Analysis.* arXiv:physics/0703039 (2007).

[74] M. Heck. *Private Conversation* (2015).