technische universität
dortmund

Bachelorarbeit

Analyse von IceCube-Daten und Vergleich von Voting-Mechanismen für Random Forest

Kai Brügge
Dezember 2011

Gutachter:
Prof. Dr. Katharina Morik
Dipl. Inform. Marco Stolpe

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl VIII
http://www-ai.cs.uni-dortmund.de

In Kooperation mit der Fakultät für Physik
Lehrstuhl E5b Astroteilchenphysik
http://www.e5.physik.uni-dortmund.de

Inhaltsverzeichnis

1. Einleitung

Die genaue Beobachtung und Analyse von Neutrinos spielt eine große Rolle in vielen Bereichen der Physik und Astronomie. Das IceCube Neutrino-Teleskop am Südpol sammelt seit Baubeginn 2004 beständig Daten. Die Datenmengen, die IceCube produziert, werden mit verschiedenen Methoden analysiert und reduziert. Durch kosmische Strahlung, die ständig auf die Erdoberfläche trifft, entsteht ein starkes Rauschen in den Daten. Ziel der Datenanalyse ist die bestmögliche Entfernung des Rauschanteils aus den Daten. Dazu werden schon direkt in den Sensoren von IceCube Datenverarbeitungsschritte durchgeführt, welche Teile des Rauschens entfernen. Den Rohdaten aus dem Detektor werden durch verschiedene Rekonstruktionsmethoden Merkmale zugeordnet. Am Ende der Datenverarbeitungskette stehen maschinelle Lernverfahren wie Random Forest. Bei Random Forest handelt es sich um ein 2001 von Leo Breiman veröffentlichtes überwachtes Lernverfahren, welches aus einem Ensemble von Entscheidungsbäumen besteht.[16] Nach Durchführung aller Verarbeitungsschritte sollen Daten entstehen, die möglichst frei von Rauschen sind und gleichzeitig möglichst viel vom gesuchten Signal beinhalten. Nur dann können physikalisch relevante und signifikante Aussagen getroffen werden. Im Rahmen dieser Arbeit sollte das Random Forest-Verfahren durch erweiterte Abstimmungsmethoden verbessert werden. Dazu wurde eine von Robnik-Ŝikonja [36] und Tsymbal[40] vorgeschlagene Methode implementiert und auf exemplarischen IceCube Daten getestet. Eine alternative Methode, die auf Clustering zurückgreift, wird außerdem getestet. Darüber hinaus wird eine von Breiman vorgeschlagene Merkmalsgewichtung durch Random Forest implementiert und empirisch mit anderen Verfahren verglichen.

Im nächsten Abschnitt wird der grundlegende Aufbau von IceCube erläutert. Anschliessend werden physikalische Grundlagen erklärt, die eine Motivation für das Experiment und die Analyse liefern sollen. In Abschnitt 4 wird beschrieben welche Verarbeitungsschritte die Daten durchlaufen bis sie mit Lernverfahren analysiert werden können. Darauf werden in Abschnitt 5 Random Forest und Erweiterungen beschrieben. In Kapitel 8 wird empirisch überprüft, ob die Gewichtung durch Random Forest bessere Ergebnisse erzielt als bisher verwendete Methoden. Abschnitt 9 soll statistisch nachweisen, ob die erweiterten Abstimmungsmethoden für Random Forest tatsächlich eine bessere Klassifikation auf IceCube Daten erlauben. Zuvor werden in Teil 7 die für den Vergleich benötigten Verfahren beschrieben.

2. Aufbau von IceCube

Das IceCube Neutrino-Teleskop am geographischen Südpol soll Physikern und Astronomen dabei helfen, das Elementarteilchen namens Neutrino zu beobachten. Das Teleskop wurde im Dezember 2010 in der Antarktis fertiggestellt und schließt etwa $1\,\mathrm{km}^3$ antarktisches Eis ein. Direkt neben der Amundsen-Scott-Station für Polarforschung wurden 86 Löcher gebohrt, in denen digitale Lichtsensoren an Kabeln, sogenannten Strings, in eine Tiefe von bis zu $2,45\,\mathrm{km}$ herabgelassen wurden. Seit Baubeginn im Jahr 2004 wurden insgesamt 5160 dieser Lichtsensoren (DOM, Digital Optical Module) im Eis platziert. Die Strings werden an der Oberfläche im IceCube Lab (ICL) zusammengeführt. Das ICL dient der Speicherung und teilweisen Verarbeitung der Sensordaten. Der schematische Aufbau von IceCube ist in Abbildung 1 dargestellt.

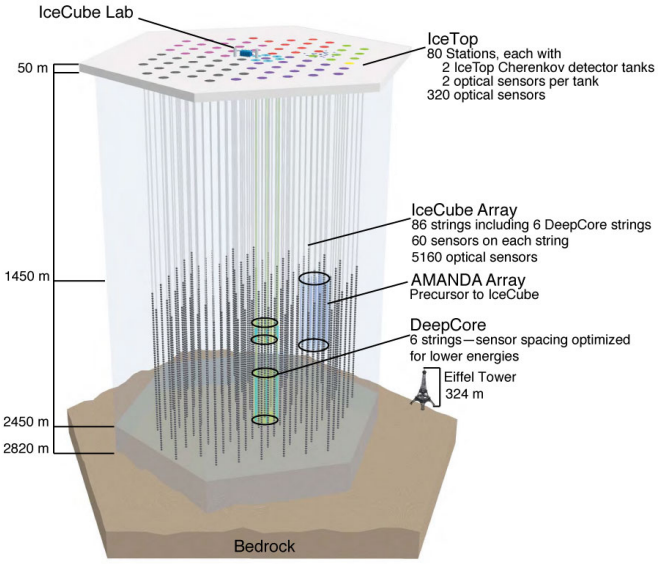

Abbildung 1: Schematischer Aufbau des IceCube Detektors, inklusive DeepCore Erweiterung, in der finalen 86-String Konfiguration.[5]

Durch das große Volumen, welches der Detektor mit dem Eis einschließt, wird eine effektive Beobachtung von hoch-energetischen Neutrinos von verschiedenen kosmologischen Quellen möglich. [26] IceCube kann mit seinen Lichtsensoren die Nebenprodukte erkennen, die bei der Interaktion eines Neutrinos mit einem Wassermolekül entstehen. Die physikalischen Hintergründe werden im folgenden Kapitel kurz erläutert.

3. Physikalische Hintergründe

Die Existenz von Neutrinos wurde 1930 von Wolfgang Pauli vorhergesagt um das kontinuierliche Energiespektrum beim Beta-Zerfall zu erklären. Der einfache Beta-Zerfall ist ein radioaktiver Prozess bei dem ein Neutron zerfällt und Betastrahlung in Form von Elektronen abgibt. Pauli sagte voraus, dass neben dem Elektron noch ein weiteres Teilchen beteiligt sein müsse. Zuerst stand auch Pauli selbst seiner Idee skeptisch gegenüber. Das vorausgesagte Teilchen besitzt keine elektrische Ladung und nur eine geringe Masse, was einen experimentellen Nachweis schwierig macht. Im Jahr 1934 formulierte Enrico Fermi seine Theorie des Beta-Zerfalls und taufte das Teilchen auf den Namen Neutrino. Der eigentliche experimentelle Beweis für die Existenz des Neutrinos gelang Clyde L. Cowan und Frederick Reines erst im Jahre 1956 durch das Project Poltergeist [34]. Reines erhielt für seine Entdeckung den Nobelpreis für Physik im Jahr 1995.

Neutrinos sind, genau wie Elektronen, sogenannte Leptonen. Das heißt sie interagieren nicht, wie etwa Quarks, über die starke Wechselwirkung, sondern nur über die schwache Wechselwirkung, die Gravitation und die elektromagnetische Kraft. Das Besondere an Neutrinos ist, dass sie im Gegensatz zum Elektron keinerlei elektrische Ladung tragen. Wie alle Fermionen kommen auch Neutrinos in drei Varianten, sogenannten Flavours, vor: das Elektron Neutrino ν_e , das Myon Neutrino ν_μ und das Tau Neutrino ν_τ. Die Massen der verschiedenen Neutrinos sind noch nicht genau bestimmt worden und sind Gegenstand aktueller Forschungen.[1] Wie erwähnt tragen Neutrinos keine elektrische Ladung und unterliegen nur den Einflüssen der schwachen Wechselwirkung und der Gravitation. Aufgrund ihrer sehr geringen Masse können Neutrinos weite Strecken durch den Raum zurücklegen ohne durch Gravitation oder elektromagnetische Felder abgelenkt zu werden. Aus der Richtung eines Neutrinos lässt sich deshalb, im Gegensatz zu anderen Teilchen, auf ihren räumlichen Ursprung schließen. Genau diese Eigenschaft macht sie so interessant für astronomische Betrachtungen. Die Beobachtung von hochenergetischen Neutrinos aus kosmologischen Quellen ist eine der Hauptmotivationen für den Bau von Neutrino-Teleskopen wie IceCube.

3.1. Quellen von Neutrinos

Neutrinos unterschiedlicher Energien entstehen an vielen Orten im Universum. Sie haben Einfluss auf die Messungen von IceCube und die Daten, die IceCube produziert. Auf der Erde entstehen Neutrinos vor allem durch den natürlich β-Zerfall der Materie. Diese Neutrinos sind für IceCube allerdings nicht von Interesse. Viel wichtiger für die Analyse und die Funktion des Detektors sind die Neutrinos, die außerhalb unseres Planeten entstehen, wie zum Beispiel in Supernovae, durch den Fusionsprozess in der Sonne oder durch schwarze Löcher in den Zentren von Galaxien. Neutrinos unterschiedlichster Energien und Herkunft treffen auf den Detektor. Die verschiedenen Entstehungsorte von Neutrinos werden im folgenden kurz erläutert.

[1] Siehe auch das KATRIN Experiment am Karlsruher Institut für Technologie [6]

3.1.1. Atmosphärische Neutrinos

Im Jahr 1912 entdeckte Viktor Hess bei einem Ballonflug die kosmische Strahlung. Der Begriff *Strahlung* ist allerdings irreführend, da es sich um einen Teilchenstrom handelt. Kosmische Strahlung besteht zu 98% aus geladenen Atomkernen, wie Protonen (Wasserstoff) oder Alphateilchen (Helium). Auch schwerere Elemente kommen in der Strahlung vor, allerdings wesentlich seltener. Die Strahlung erreicht Energien bis zu $10^{20}eV$. Die Entstehungsprozesse, besonders bei Teilchen höherer Energien ab $10^{18}eV$, sind noch nicht bekannt. Vermutlich aber haben diese Teilchen ihren Ursprung außerhalb unserer Galaxie. Zumindest ist kein Objekt in der Milchstrasse bekannt, welches diese Energien erzeugen könnte. In Abbildung 2 ist der Fluss der Kosmischen Strahlung zu sehen. Deutlich zu erkennen ist, dass die Häufigkeit der Teilchen mit steigender Energie rasch abnimmt. Das seltene Auftreten von Teilchen hoher Energien macht die Untersuchung schwierig. Besonders über die Herkunft der Teilchen mit mehr als $10^{20}eV$ ist bisher wenig bekannt.

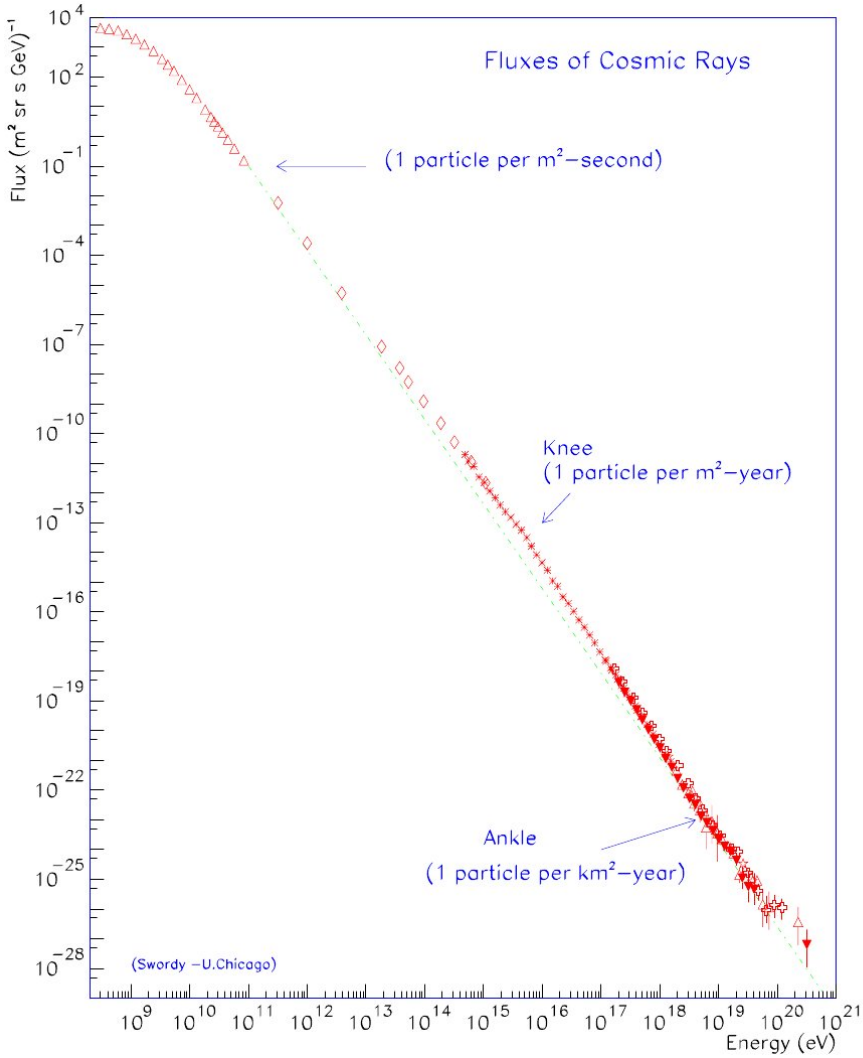

Abbildung 2: Fluss kosmischer Strahlen in Abhängigkeit der Teilchenenergie.[30]

Kosmische Strahlung trifft ununterbrochen auf die Atmosphäre und das Magnetfeld der Erde. Ein Teil der Strahlung wird vom Magnetfeld abgelenkt und erreicht den Erdboden nicht. Ein bekannter Nebeneffekt der Interaktion zwischen dem Magnetfeld und den geladenen Teilchen ist das Polarlicht, welches manchmal auch in unseren Breitengraden beobachtet werden kann. Energiereiche Teilchen können das Magnetfeld allerdings durchdringen und treffen auf Teilchen in der Atmosphäre. Bei der Reaktion entstehen Teilchenschauer die aus mehreren Millionen Sekundärteilchen bestehen können. Die meisten davon erreichen die Erdoberfläche allerdings nicht, da

sie zu kurzlebig sind und zerstrahlen oder durch weitere Interaktionen mit der Atmosphäre verloren gehen. Die Teilchenschauer bestehen aber auch aus Myonen und Neutrinos. Beide Teilchen können den Erdboden und damit IceCube erreichen [30, vgl. Abbschnitt 3.4]. Die Untersuchung der atmosphärischen Neutrinos ist hilfreich bei der Kalibrierung des Detektors. Darüber hinaus erlaubt sie genauere Untersuchungen der Neutrino-Oszillation [25].

Entscheidend bei dem Prozess ist, dass die Myonen aus den Teilchenschauern den Detektor erreichen. Diese erzeugen ein Hintergrundrauschen im Detektor, welches das eigentliche Neutrino Signal überlagert. Das Herrausfiltern dieser eigentlich uninteressanten Myonen aus den aufgezeichneten Daten ist eine der größten Herausforderungen bei der Datenanalyse. Mehr dazu folgt in Teil 4.1.

3.1.2. Hochenergetische Kosmische Strahlung

Neben den geladenen Teilchen der kosmischen Strahlung erreichen uns auch energiereiche Neutrinos und Photonen aus kosmologischen Quellen. Im Gegensatz zu den geladenen Teilchen erlauben die Neutrinos aber einen Rückschluss auf die Quelle. Dadurch, dass sie nicht von Magnetfeldern abgelenkt werden, fliegen die Neutrinos auf einer fast geraden Bahn durch den Raum. Dadurch können eventuell die Entstehungsprozesse für hochenergetische kosmische Strahlen und Neutrinos verstanden werden. Ein Kandidat für die Herkunft Kosmischer Strahlung sind Active Galactic Nuclei (AGN). AGN sind Zentren von Galaxien, die eine große Menge von Energie abstrahlen. Sie gehören zu den hellsten Objekten am Himmel. Die Energie kommt vermutlich von einem supermassiven schwarzen Loch, welches sich im Zentrum der Galaxie befindet. Eine weitere mögliche Quelle für kosmische Strahlen sind Gamma Ray burst (GRBs). GRBs sind kurze extrem helle Pulse von Gamma-Strahlen. Sie können zwischen einigen Millisekunden bis zu einigen Minuten dauern. Ihre genaue Herkunft ist unbekannt, doch wird vermutet, dass unter anderem bei Kollisionen von Neutronensternen oder dem Kollaps eines Sterns zu einem schwarzen Loch entstehen. Wie schon in 3.1.1 beschrieben, nimmt der Fluss der Strahlen mit zunehmender Energie ab. Im oberen Energiebereich treffen also nur noch wenige Teilchen pro Fläche auf die Erde. Mit seiner Ausdehnung von über $1\,\mathrm{km}^3$ kann IceCube genug Teilchen aus hohen Energiebereichen aufzeichnen, um statistisch signifikante Ergebnisse zu erhalten [26].

3.1.3. Neutrinos aus Supernovae

Wird eine Supernova kritisch, werden enorme Mengen von Energie frei. Neben sichtbaren Licht und Strahlung wird eine großer Teil der Energie in Form von Neutrinos frei. [2] Das erste mal konnte dies im Jahr 1987 beobachtet werden. Verschiedene Neutrino-Detektoren konnten 3 Stunden bevor das sichtbare Licht der Supernove SN1987A uns erreichte, etwa 20 Neutrino-Interaktionen aufzeichnen, die über dem üblichen Hintergundrauschen lagen. Selbst wenn die Neutrinos, die bei einer Supernova enstehen, nur Energien im Bereich von wenigen MeV haben, wird IceCube mithilfe der DeepCore-Erweiterung einen Anstieg von Neutrinos feststellen können, bevor ein Stern in unserer

[2] Nach aktuellen Modellen etwa 99% der Energie.[24, Seite. 1]

Galaxie kritisch wird.[24] IceCube bildet zusammen mit anderen Neutrino-Detektoren das Supernova Early Warning System (SNEWS). Wenn das SNEWS Netzwerk einen Anstieg von Neutrinos in einem bestimmten Energiebereich feststellt, wird ein Alarm ausgesendet. Optische Teleskope, auch von Amateur-Astronomen, können dann in Richtung der Neutrinoquelle gerichtet werden und können vielleicht einen Stern beobachten während er zur Supernova wird. Zurzeit sind vier Neutrino-Detektoren Teil des SNEWS: LargeVolumeDetector, Super-Kamiokande, IceCube und Borexino.

3.1.4. Solare Neutrinos

Solare Neutrinos waren lange Zeit ein großes Rätsel für die Neutrinoforschung. Beim Kernfusionsprozess in der Sonne entstehen Elektron-Neutrinos. Die Menge der produzierten Neutrinos lässt sich vorhersagen. Das Homestake-Experiment maß den Flux der Neutrinos aus der Sonne, konnte allerdings in den 1968 veröffentlichten Ergebnissen nur etwa ein Drittel der erwarteten Neutrinos beobachteten. Erst 2001 konnte die Kollaboration um das Sudbury Neutrino Observatory (SNO) bestätigen, dass Neutrino-Oszillationen für die Messergebnisse verantwortlich waren. Auch diese Neutrinos können von IceCube beobachtet werden und helfen, genauere Messungen der Oszillationen zu machen. [13]

3.1.5. Neutrinos aufgrund von WIMPs

Weakly Interacting Massive Particles (WIMPs) sind mögliche Kandidaten für dunkle Materie. Dunkle Materie wird vorausgesagt um die Effekte von Gravitation auf sichtbare Materie im Universum zu erklären. Existieren WIMPs, dann sammeln sie sich in der Sonne an und erzeugen Neutrinos, deren Energie höher ist als die der anderen solaren Neutrinos. IceCube wäre in der Lage, diese Neutrinos anhand ihrer Richtung und Energieverteilung zu erkennen.

3.2. Funktionsweise von IceCube

IceCube versucht, mit Messungen im antarktischen Eis die Richtung, Häufigkeit und Energie der Neutrinos zu bestimmen, die den Detektor durchqueren. Wie oben beschrieben, wechselwirken Neutrinos nicht elektromagnetisch und können Materie durchdringen. Die direkte Messung von Neutrinos ist also nicht möglich. Trotz der im Allgemeinem geringen Wechselwirkungswahrscheinlichkeit, kann ein Neutrino mit Materie interagieren.[3] Aufgrund der großen Anzahl von Neutrinos, die ständig auf die Erde treffen, wird es auch im Eis um den IceCube-Detektor zu Reaktionen kommen. Bei der Reaktion wird ein Myon frei, welches sich in die selbe Richtung wie das ursprüngliche Neutrino bewegt. Das Myon bewegt sich mit $v_{\nu_\mu} > c_{ice}$ durch das Eis. Die Geschwindigkeit des entstandenen Myons liegt demnach über der, der Lichtgeschwindigkeit im Eis, wobei c_{ice} natürlich kleiner als die Vakuumlichtgeschwindigkeit c_0 ist.[4]

[3] Diese Interaktion findet über die schwache Wechselwirkung statt.

[4] Die Ergebnisse des OPERA Experiments, welche im Herbst 2011 veröffentlicht wurden, haben auf diese Tatsache keinen Einfluss, selbst *wenn* sie stimmen sollten.

Durch die erhöhte Geschwindigkeit des Myons kommt es zum sogenannten Cherenkov-Effekt. Bewegt sich ein geladenes Teilchen durch das Eis, richten sich die Dipole in Richtung des Teilchens aus und erzeugen eine elektromagnetische Welle. Bewegt sich das Teilchen schneller als die Lichtgeschwindigkeit des umgebenden Mediums, kommt es zu einer konstruktiven Inferenz der Wellen und es entsteht Licht. Ähnlich wie beim Überschallflug in der Luft, breitet sich das Licht kegelförmig hinter dem Myon aus. Cherenkov-Licht hat Wellenlängen im sichtbaren Bereich und kann mit bloßem Auge erkannt werden. In Abbildung 3 ist Cherenkov-Licht zu sehen welches durch schnelle Elektronen in Wasser eines Kernreaktors entsteht. Dieses Licht können die DOMs im Eis registrieren und somit indirekt Neutrinos Messen.

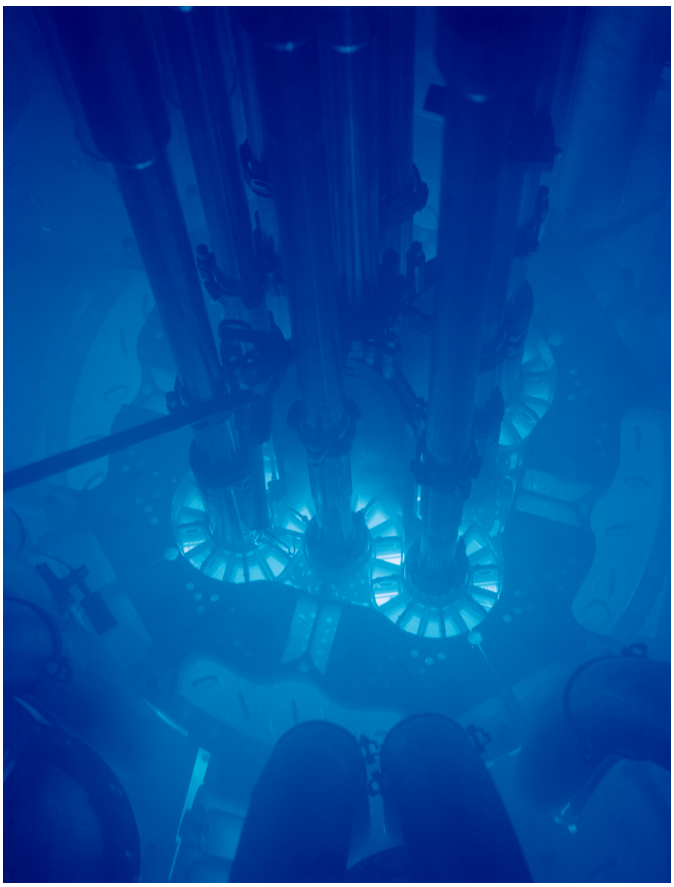

Abbildung 3: Das blaue Cherenkov-Licht in diesem Kühlbecken wird durch schnelle Elektronen erzeugt. Der Reaktor steht im Idaho National Laboratory. [2]

3.3. Das Digital Optical Module

Die DOMs bilden das eigentliche Herzstück des Detektors. Vor Umwelteinflüssen, wie Wasser und hohem Druck, wird der DOM durch eine Glasummantelung geschützt. Im Inneren besteht ein DOM aus dem eigentlichen Sensor, dem Photomultipler (PMT), einem eingebetteten System für Signalberechnungen, dem DOM Mainboard und Modulen zur Spannungsversorgung. Jeder DOM ist über Kupferkabel mit dem ICL an der Oberfläche verbunden. Das ICL kann über die Kabel mit jedem DOM im Eis kommunizieren. Die Software auf den DOMs kann so von zentraler Stelle aktualisiert, überwacht und kalibriert werden. So gelangen auch die eigentlichen Daten aus den Sensoren an die Oberfläche. Für Kalibrierungszwecke und für die Triggerfunktionen (siehe Abbschnitt 4.1) können die DOMs außerdem mit ihren unmittelbaren Nachbarn auf einem String kommunizieren. Der Photomultipler wandelt Lichtsignale in elektrische Signale um. Erzeugt ein Teilchen im Eis Cherenkov-Licht, wird im PMT eine Spannungskurve erzeugt, deren Form und Amplitude von der Intensität und Dauer des Lichtpulses abhängt. Dieses analoge Signal wird digitalisiert und kann dann, je nach Trigger-Bedingung (siehe Abschnitt 4.1), im internen Speicher des DOMs gepuffert oder verworfen werden. Um ein Signal auch einem Myon beziehungsweise Neutrino zuzuordnen, muss jeder DOM seine Daten mit einem Zeitstempel versehen. Die Uhren im gesamten Detektor müssen mit einer Genauigkeit von 5 ns arbeiten; deshalb werden die DOM internen Uhren etwa alle 1.5 Sekunden von der Oberfläche aus mit einer GPS Uhr synchronisiert.[26, Abschnitt H] Die Architektur der DOMs ist so flexibel wie möglich gehalten. Die Software im DOM kann komplett überschrieben werden und erlaubt so Anpassungen an Sensoreigenschaften oder Datenverarbeitungsschritten, die schon im DOM stattfinden. So können Veränderungen am DOM vorgenommen werden, obwohl dieser tief im Eis liegt. Die Zuverlässigkeit der DOMs war schon beim Design von hoher Bedeutung; so sind von den insgesamt 5120 vergrabenen DOMs nur 6% vollständig unbrauchbar [26]. In Abbildung 4 ist ein DOM am Kabelstrang zu sehen.

Abbildung 4: Der letzte DOM, der in das Eis herabgelassen wurde. Der PMT befindet sich an der Unterseite der Kugel. [5]

4. Datenanalyse

Die Analyse soll nun aus der Helligkeit, räumlichen Verteilung und Dauer der Lichtblitze im Eis, Informationen über die Energie und Herkunft eines Neutrinos gewinnen. Die Schwierigkeit dabei besteht darin, dass das eigentliche Signal vom Hintergrundrauschen getrennt werden muss. Die Anzahl an atmosphärischen Myonen überwiegt stark gegenüber den Myonen, die aus Neutrino-Interaktionen entstanden sind. Analog zum Fluss kosmischer Strahlen (Abbildung 2) schwindet die Häufigkeit der Neutrinos aber mit zunehmender Energie. Das Rauschen der Atmosphärischen Myonen muss also möglichst gut gefiltert werden, ohne dass dabei Informationen über die gesuchten Neutrinos verloren geht.

Die großen Datenmengen, die IceCube produziert, werden durch diverse Filter und Trigger reduziert und machen einen teilweisen Datentransfer vom Südpol via Satellit möglich. Damit die Daten analysiert werden können, wird sowohl das Rauschen als auch das Signal mit eigenen Programmen simuliert. Zur Simulation von IceCube Daten kommt das Programm CORSIKA [4] zum Einsatz. Trotz Filtermethoden haben die Daten einen hohen Rauschanteil, der mit maschinellen Lernverfahren soweit wie möglich eliminiert werden soll. Von der IceCube Gruppe der Physik-Fakultät der Technischen Universität Dortmund wird RapidMiner [1] als Framework zur Analyse benutzt. Dabei kommt unter anderem Random Forest als Lernverfahren bei der Datenanalyse zum Einsatz [37]. Verwendet wird dabei die Implementierung aus dem WEKA Paket [10], welche durch die WEKA-Extension in RapidMiner eingebunden wird.

4.1. Trigger

Ziel der am Südpol angewendeten Trigger und Filter ist die Reduzierung des Rauschens in den Daten, ohne dass dabei große Teile des tatsächlich erwünschten Signals verloren gehen. Neben dem Rauschen durch atmosphärische Myonen entstehen falsche Signale auch direkt in den Sensoren der DOMs. Die PMTs reagieren nicht nur auf das gesuchte Cherenkov-Licht, sondern unter anderem auch auf die natürliche Radioaktivität aus der Umgebung und der Glasummantelung des DOMs. Übersteigt die Intensität des Signals, welches der PMT registriert, einen Schwellenwert, wird die Spannungkurve aus dem PMT im DOM gespeichert. In diesem Fall spricht man von einem Hit (auch Puls). Das Datenpaket zu einem Hit muss vor allem eine Beschreibung der Spannungskurve, einen Zeitstempel und die DOM-Id enthalten, damit man die Spur eines Myons im Detektor rekonstruieren kann. Für die Digitalisierung der Spannungskurven sind, wie in Abschnitt 3.3 beschrieben, zwei Analog-Digitalwandler zuständig. Das Feature Extraction Modul (NFE) berechnet aus den Spannungskurven die Ankunftszeiten der einzelnen Cherenkov-Photonen und weitere Merkmale, die Rückschlüsse auf die Energie des Myons zulassen. Die in IceCube benutzten PMTs produzieren durch das sensorisch bedingte Rauschen eine Hit-Rate von etwa 300 Hz. [26] Da dieses Rauschen aber zum größten Teil uninteressant für weitere Analysen ist, gibt es Triggerkriterien, die einschränken, wann die Hits tatsächlich in die Analyse mit einbezogen werden. Die Hard Local Coincidence (HLC) berücksichtigt die Informationen aus einem DOM nur dann, wenn innerhalb eines Zeitfensters von 1 μsec die nächsten oder übernächsten

DOMs auf dem selben String auch einen Hit registrieren. Die HLC-Hit-Rate beträgt etwa $3 - 15\,\mathrm{Hz}$ je nach Tiefe des optischen Moduls. [26] Bei der Soft Local Coincindece (SLC) handelt es sich nicht, wie der Name vermuten lässt, um eine lokale oder zeitliche Einschränkung. Die SLC betrachtet alle Hits, speichert sie aber in einem reduzierten Format, welches nicht die gesamte Spannungskurve enthält, sondern nur deren Amplitudenmaximum.Diese Kriterien werden unter dem Begriff In-Ice Trigger zusammengefasst und werden schon innerhalb der DOMs überprüft.

Will man die Spur die ein Neutrino durch den Detektor genommen hat rekonstruieren, muss man die Daten aller DOMs in dem Zeitfenster in dem das Teilchen im Detektor war, zusammenfassen. Dieses Datum wird dann als Event bezeichnet. Vereinzelte HLC- oder gar SLC-Hits sind kein ausreichendes Kriterium um ein Event zu klassifizieren. Deshalb gibt es weitere Online-Trigger, die im IceCube Lab an der Oberfläche überprüft werden. Wenn in einem Zeitfenster von $5\,\mu\mathrm{sec}$ insgesamt 8 oder mehr DOMs im Detektor einen HLC-Hit registriert haben, wird ein Event erstellt. Dabei ist die Position der DOMs unerheblich. Dieser Trigger wird Simple Multiplicity Trigger (PMT8) genannt. Außerdem wird ein Event registriert, wenn 5 von 7 benachbarten DOMs auf einem String innerhalb von $1.5\,\mu\mathrm{sec}$ einen HLC-Hit produzieren. Beide Trigger werden manchmal auch als Time Window Cleaning (TWC) bezeichnet. Das Event enthält dann die Informationen aller DOMs in einem Zeitfenster von $10\,\mu\mathrm{sec}$ um den Zeitpunkt des Triggers. Sollten sich zwei Events zeitlich überschneiden, werden sie zu einem Event zusammengefasst.[22, vgl. Abschnitt 7] Abbildung 5 zeigt ein Event, wie es von IceCube aufgezeichnet wird.

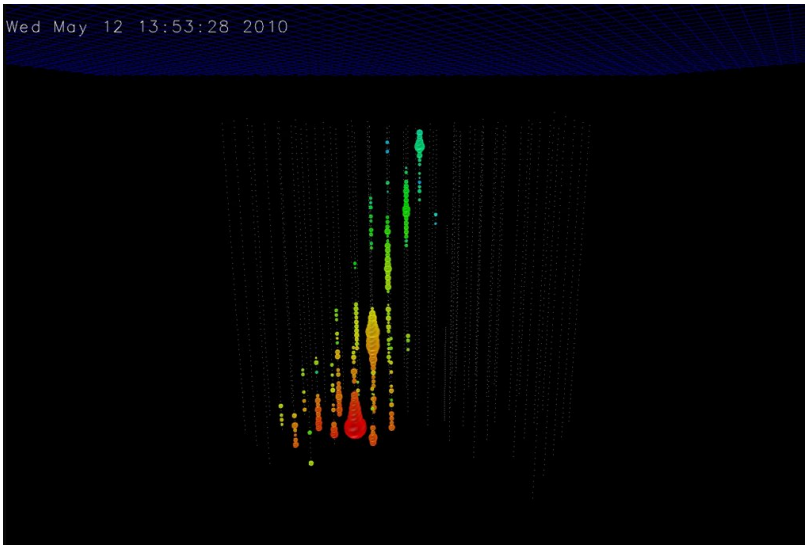

Abbildung 5: Ein visualisiertes Event, wie es von IceCube gesehen wird. Hier die Simulation eines aufsteigenden Myons in der 56-String-Konfiguration von IceCube. Die Farben Kodieren den Zeitpunkt an dem ein Hit registriert wurde.Die Größe der Punkte steht für die aufgezeichnete Helligkeit. [5]

Aus einem Event lässt sich die Richtung eines Neutrinos rekonstruieren. Außerdem lässt die Intensität und Verteilung des Lichts Rückschlüsse auf Neutrinoflavour und Energie zu.

Der komplette Datensatz von Events wird im ICL auf Bändern gesichert, die einmal im Jahr nach Wisconsin geflogen werden. In den Events befinden sich auch all die, die von atmosphärischen Myonen ausgelöst wurden und demnach für die Analyse der Neutrinos keine Bedeutung haben. Auch ist es aufgrund der beschränkten Bandbreite von Satellitenverbindungen zum Südpol, nur möglich, einen Teil der Daten täglich per Satellit zu übertragen. Deshalb werden am Südpol Filter eingesetzt, die die Größe und Qualität der Daten verbessern sollen.

4.2. Filter

Filter wählen die Events aus, die für bestimmte physikalische Analysen interessant sein könnten. Zu den ausgewählten Events gehören vor allem die, denen eine hohe Energie zugeordnet werden kann (EHE Filter) und Events die voraussichtlich mit Myonen korrespondieren, die den Detektor von unten nach oben durchquert haben (Muon Filter). Im Anhang A sind exemplarisch die Filter und der jeweilige Datendurchsatz für die Saison 2010 dargestellt. Die grundsätzliche Idee zum Filtern der Myonen mit atmosphärischen Ursprung ist, sich nur auf Events aus einer bestimmten Richtung zu beschränken. Wie in Abschnitt 3 erwähnt, können Neutrinos, anders als Myonen, weit in Materie eindringen, ohne mit dieser zu interagieren. Abbildung 6 zeigt die Weglänge

eines Neutrinos in Abhängigkeit von seiner Energie. Zu Erkennen ist, dass bei Energien unter 10^{14} eV die Erde transparent für Neutrinos ist.

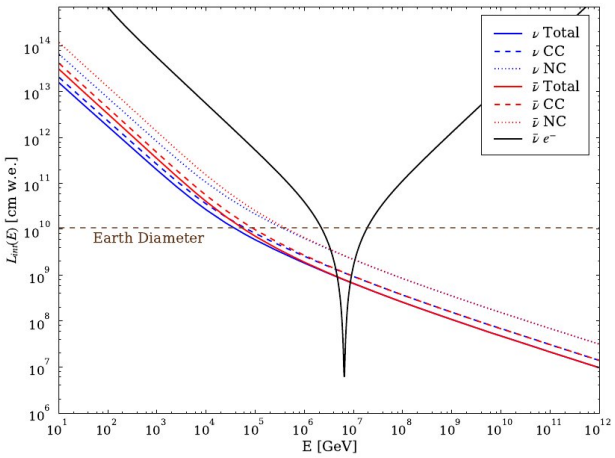

Abbildung 6: Der Plot zeigt die durchschnittliche Weglänge eines Neutrinos in der Erde. Gezeigt sind die Weglängen verschiedener Interaktionen. Bei etwa $10^5 GeV$ ist die erwartete Weglänge kürzer als der Erddurchmesser. [42, Kapitel 3.1]

Neutrinos können also den gesamten Erdball durchqueren und dann im Eis Myonen erzeugen, die den Detektor von unten nach oben durchqueren (upgoing). Die Filter versuchen, diese Events mit den limitierten Rechnerkapazitäten am Südpol zu erkennen.

Der MuonFilter beispielsweise geht dabei wie folgt vor. In einer ersten Rekonstruktion wird das LineFit verfahren benutzt. Dafür wird vereinfacht angenommen, dass sich das Cherenkov-Licht entlang einer eindimensionalen Linie durch den Detektor ausbreitet. Angenommen, der Startpunkt des Myons sei r und die Geschwindigkeit v, dann erreicht das Licht den DOM r_i nach der Zeit t_i. Also:

$$r_i = r + v \cdot t_i$$

Die Parameter r und v sind zu bestimmen. Nimmt man an, die Richtungen, aus der die Myonen kommen, seien gleichverteilt, kann man die gesamte Verteilung als χ^2 Verteilung ausdrücken.

$$\chi^2 = \sum_i (r_i - r - v \cdot t_i)$$

Minimiert man diesen Term ergeben sich für v und r folgende Werte

$$r = \langle r_i \rangle - v \cdot \langle t_i \rangle \,.$$

$$v = \frac{\langle r_i \cdot t_i \rangle - \langle t_i \rangle \cdot \langle r_i \rangle}{\langle t_i^2 \rangle - \langle t_i \rangle^2}$$

Dabei sind $\langle r_i \rangle$ und $\langle t_i \rangle$ jeweils die Durchschnittswerte. Vergleiche dazu auch[20] und [7] Mit den Ergebnissen der LineFit Methode lässt sich der Winkel zum Zenith einer Myonenspur auf etwa 10° genau bestimmen [35]. Alle Events mit einem Zenithwinkel von über 70° werden anschließend mit Likelihood Methoden untersucht, wodurch sich die Winkelgenauigkeit auf bis zu 0.7° verbessert [8]. LineFit kann die zu überprüfende Anzahl an Events für die Likelihood Methoden reduzieren und somit Rechnerresourcen schonen. Durch verschiedenen Rekonstruktionsmethoden und Filter der unterschiedlichen Arbeitsgruppen können einem IceCube Event mehrere tausend Merkmale zugeordnet werden. Im weiteren Verlauf der Arbeit wird ein Datensatz benutzt, der Merkmale aus LineFit und diversen Likelihood-Verfahren besitzt.

Nachdem die Daten der Filter vom Südpol übertragen wurden, können sie mit genaueren Methoden untersucht werden, um die Merkmale für Richtung und Energie etc. eines Events mit höherer Genauigkeit zu bestimmen. Auch nach der Anwendung all dieser Filtermethoden überwiegt die Anzahl der Events von absteigenden atmosphärischen Myonen gegenüber den eigentlich gesuchten. Das Rauschen in den Daten sind also fälschlicherweise als upgoing identifizierte Events. Falsche Events entstehen vor allem aus zwei Gründen: Myonen die sich nur im Randbereich des Detektors bewegen, können nur schlecht rekonstruiert werden; und es können sich mehrere Myonen gleichzeitig an verschiedenen Orten im Detektor aufhalten. Abblidung 7 zeigt ein Event mit zwei, etwa zeitgleichen, Lichtblitzen im Detektor, welches von LineFit fälschlicherweise als upgoing klassifiziert wird.

Abbildung 7: Zwei Lichtblitze die vom LineFit-Algorithmus (rote Linie) fälschlicherweise als ein aufsteigendes Teilchen erkannt werden. [5]

Das Rauschen könnte zwar durch strengere Kriterien reduziert werden, allerdings würden dabei auch Teile der eigentlich gesuchten Events verloren gehen. Das Signal-zu-Rausch Verhältnis der Daten beträgt nach allen Filtern immer noch $1.46 \cdot 10^{-3}$. [37]

4.3. Simulation

Unterschiedliche Neutrinointeraktionen ergeben unterschiedliche Events in IceCube. Neutrinos verschiedener Herkunft und Energien bilden unterschiedliche Muster im Detektor. Die gesamte Hardware des Detektors und die verschiedenen physikalischen Vorgänge, bei denen Neutrinos entstehen, müssen simuliert werden. Dadurch kann die Effektivität verschiedener Filter und Analysemethoden überprüft werden. Außerdem macht es den Einsatz von überwachten Lernverfahren zur Datenanalyse möglich. Das Programm CORSIKA simuliert Luftschauer, die durch kosmische Strahlen entstehen. Die Reaktion der kosmischen Teilchen mit unserer Atmosphäre kann in CORSIKA nach verschiedenen theoretischen Modellen nachgestellt werden. Trifft ein Teilchen der kos-

mischen Strahlung auf die Atmosphäre, interagiert es vor allem über die starke Wechselwirkung mit den Atomkernen in der Luft. Dabei entstehen viele Sekundärteilchen, die wiederum mit Teilchen reagieren und neue Teilchen erzeugen. Der komplette Weg der Primär- und Sekundärteilchen bis zur Erdoberfläche kann von CORSIKA simuliert werden.[4] Atmosphärische Myonen, die in den Detektor eindringen, können so simuliert werden. Neutrinos werden durch NuGen (Neutrino Generator) erzeugt und der Weg bis zu einer Interaktion simuliert. Im Eis wird der Weg der Myonen von dem Programm Muon-Monte-Carlo (MMC) simuliert. Die Ausbreitung des Lichts im Eis übernimmt Photonics, welches auch die verschiedenen Staubschichten im antarktischen Eis berücksichtigt. [9] Die Simulation der PMT's und der dazugehörigen Elektronik in den DOMs wird von romeo und DOMsimulator übernommen. Zusammen können diese Programme eine Menge von Events erzeugen wie sie von IceCube gesehen werden würden.

4.4. Überwachte Lernverfahren

Wie bereits erwähnt, sind die Daten nach allen Vorverarbeitungsschritten immer noch mit starken Rauschanteilen behaftet. Überwachte Lernverfahren versuchen, Punkte aus dem Datensatz anhand ihrer Merkmale in verschiedene Klassen einzuteilen. Ein Datensatz G der Größe n besteht aus Vektoren $\vec{g}_i$. Die Vektoren enthalten je Komponente einen Wert für eines der Merkmale $a_1, \ldots, a_d$, haben also die Form $\vec{g}_i = (g_{i,a_1}, \ldots, g_{i,a_d})$. Überwachte Lernverfahren benötigen eine Trainingsmenge T, bei der die Klassenzugehörigkeiten der einzelnen Punkte bekannt sind. Die Trainingsmenge T hat also Paare der Form $(\vec{x}_i, c)$ wobei c das Label einer Klasse aus der Menge der Klassen C_T ist. Ziel des Lernverfahrens ist es, aus den Trainingsdaten ein Modell zu erstellen, welches in der Lage, ist Datenpunkten $\vec{g}_i$ eine Klasse zuzuordnen. Im Folgenden werden Punkte aus dem Merkmalsraum aus Gründen der Übersichtlichkeit nicht mehr als Vektoren notiert. Endprodukt des Lernverfahrens ist demnach eine Funktion $f\colon G \to C$. f wird im folgenden als Klassifizierer oder Lerner bezeichnet. Ziel ist die Erstellung eines Modells aus den Trainingsdaten, welches möglichst gut auf andere Daten aus dem selben Problembereich anwendbar ist. Dazu muss die erwartete Abweichung zwischen der Voraussage des Modells und der tatsächlichen Klasse geschätzt werden.[27, Kapitel 7] Im Fall binärer Klassifikation kann die Abweichung D , für ein Modell L und einen Punkt x mit zugehöriger Klasse c_x, ausgedrückt werden durch

$$D(x, C_x, L) = \begin{cases} 1 & \text{wenn } L(x) = c \\ 0 & \text{wenn } L(x) = c_x \end{cases}$$

So lässt sich der erwartete Wert von D über den Trainingsdaten einfach über den Durchschnitt bestimmen.

$$D_T(T, C, L) = \frac{1}{|T|} \sum_{t \in T} D(t, C_t, L)$$

Um zu bestimmen wie L auch auf anderen Daten klassifiziert, muss D für Mengen mit unbekannten Klassenzugehörigkeiten geschätzt werden. Dies kann über Verfahren wie

der k-fachen Kreuzvalidierung geschehen. Dabei wird die Trainingsmenge in k Teilmengen T_k unterteilt und der Lerner L nur mit den Punkten aus T_k erstellt. Dann wird Abweichung D_{T_k} auf der Menge $T \setminus T_k$ gemessen. Der geschätzte Wert für D ist dann das Mittel aller D_{T_k}. Um einen Klassifizierer zu erstellen, der möglichst gut generalisiert, reicht es im Allgemeinen nicht, den Trainingsfehler D_T zu minimieren. Mit steigender Komplexität von L kann der Fehler immer reduziert werden. Im trivialen Fall merkt sich das Modell einfach zu jeder Merkmalskombination die zugehörige Klasse. Abbildung 8, zeigt wie es dabei zu einer Überanpassung des Modells kommen kann, der Klassifizierer L also nicht mehr gut auf andere Daten generalisiert.

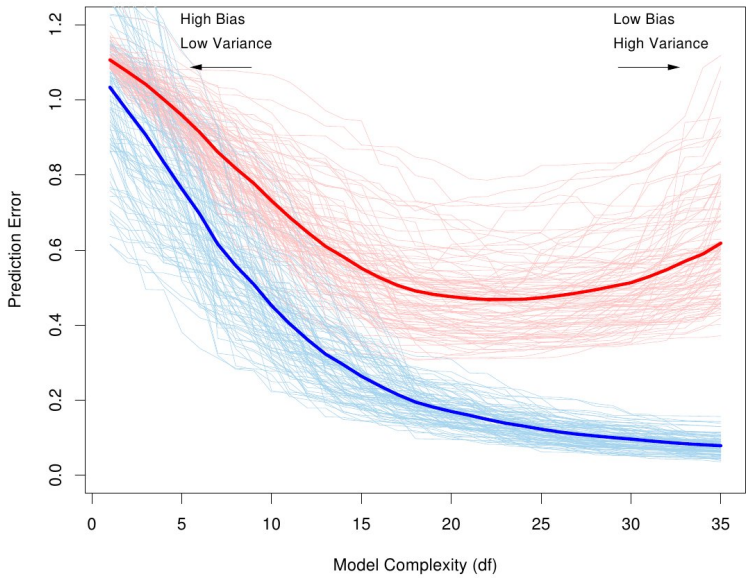

Abbildung 8: In Blau der Trainingsfehler für verschiedene Teilmengen der Trainingsdaten. Fett ist dabei das Mittel. In Rot eingezeichnet ist die Abweichung D auf Daten die nicht für die Erstellung des Modells genutzt wurden. Die horizontale Achse beschreibt dabei die allgemeine Komplexität des Modells. Entnommen aus [27, Seite 220]

Im Fall von IceCube, geht es darum aus den simulierten Daten ein Modell zu erstellen, welches in der Lage ist, die Events aus dem Detektor in Hintergrund- und Signalevents einzuteilen. Das Verfahren das im Folgenden auf die IceCube Daten angewandt wird heißt Random Forest und wird in Abschnitt 5 erklärt.

4.5. RapidMiner und Weka

Die Daten aus IceCube werden an der Icecube Arbeitsgruppe der TU Dortmund zum Teil mit dem Programm RapidMiner verarbeitet. RapidMiner ist ein plattformunabhängiges Open-Source Framework zur Datenanalyse. In dem Java Programm kann der Benutzer per Drag&Drop verschiedene Operatoren in die Datenverabeitungskette einfügen. Das Programm bietet verschiedene Lernverfahren und Datenmanipulationswerkzeuge, die durch ein Plugin/Extension System noch erweitert werden können. WEKA ist ein ebenfalls in Java geschriebenes Programm, welches an der University of Waikato entwickelt wird. Es bietet wie RapidMiner viele Methoden zur Verarbeitung und Analyse verschiedenster Daten an. Durch die WEKA-Extension sind fast alle Algorithmen aus WEKA auch in RapidMiner verfügbar. Der Random Forest Algorithmus aus dem WEKA-Paket wurde bereits erfolgreich zur Analyse von IceCube Daten benutzt [37]. Seit Version 3.7.0 hat WEKA eine Implementierung des Random Forest Lernverfahrens, welches parallel auf mehreren Prozessorkernen arbeitet. Im Rahmen dieser Arbeit wurde unter anderem die WEKA-Extension aktualisiert, so dass unter anderem auch die parallele Variante von Random Forest in RapidMiner nutzbar wird. Details folgen in Abschnitt 6.

5. Random Forest

Random Forest ist ein Verfahren für das überwachte Lernen, welches 2001 von Breiman vorgeschlagen wurde. Das Verfahren erstellt dabei ein Ensemble von Entscheidungsbäumen (Random Tree) nach dem Prinzip des Baggings, welches 1994 ebenfalls von Breiman veröffentlicht wurde. Die Motivation für die Benutzung von Random Forest ist die effiziente Parallelisierbarkeit und die guten Ergebnisse, die Random Forests häufig im Vergleich zu anderen Verfahren erzielen.[27] Random Forests wurden außerdem bereits erfolgreich auf simulierte IceCube-Daten angewandt. [37]

5.1. Bagging

Bagging ist ein Prinzip zum Aufbau eines Ensembles von Klassifizierern. Das Ensemble L fasst die Entscheidungen von k Lernern f_i zusammen, um eine Klassifizierung zu erzeugen. Jeder einzelne Klassifizierer des Ensembles wird dabei mit einer anderen Trainingsmenge erstellt. Welche Klasse einem Punkt $g \in G$ zugewiesen wird, wird durch eine Mehrheitsabstimmung der einzelnen f_i bestimmt. Gibt die Mehrheit der f_i die Klasse c zurück, so ist auch das Ergebnis des gesamten Ensembles $L(g) = c$. Bagging lässt sich auch zum Lösen von Regressionsaufgaben verwenden. In diesem Fall gibt es keine diskreten Klassen, sondern ein kontinuierliches numerisches Label, welches den Punkten aus G zugewiesen wird. Das arithmetische Mittel aller $f_i(g)$ ist dann das Ergebnis des Ensembles.

Die Trainingsmenge T der Größe n wird in k Mengen $T_1 \dots T_k$ der Größe n eingeteilt. Die Teilmengen T_i sind also genauso groß wie T. Dies geschieht durch zufälliges Ziehen mit Zurücklegen (Bootstrapping). Eine Teilmenge T_i enthält folglich im Allgemeinen nicht alle Elemente aus T, dafür aber einige doppelt.[5] Wie viele doppelte Elemente im Mittel in den T_i enthalten sind, lässt sich wie folgt abschätzen.

Die Wahrscheinlichkeit, dass ein festes Element aus T zufällig gezogen wird, beträgt $\frac{1}{n}$. Die Gegenwahrscheinlichkeit, dass ein Element nicht gezogen wird ist $1 - \frac{1}{n}$. Die Menge, aus der man das Element zieht, ist immer gleich groß, da die Elemente nach der Ziehung wieder zurückgelegt werden. Die Wahrscheinlichkeit, dass nach n-maligem Ziehen ein Element nicht gezogen wurde, beträgt demnach $(1 - \frac{1}{n})^n$. Für große n gilt nun

$$\lim_{n\to\infty} (1 - \frac{1}{n})^n = \frac{1}{e}$$

Wegen $\frac{1}{e} \approx 0.37$ kommen 37% aller Elemente in einer Teilmenge doppelt vor. Dementsprechend sind im Allgemeinen 37% der Elemente aus T nicht in T_i enthalten.[11, S.1]

Die Menge $T \setminus T_i$ wird Out-Of-Bag Menge zum Lerner f_i, OOB_{f_i} genannt. Dadurch lässt sich der sogenannte Out-Of-Bag Error E_{OOB} berechnen. Um E_{OOB} zu berechnen, wird zu jedem f_i aus L die Menge OOB_{f_i} gebildet. Dann werden die $t \in OOB_{f_i}$ von f_i klassifiziert und die Fehlerrate für diesen Lerner $E_{OOB_{f_i}}$ ermittelt. Die Fehlerrate gibt den Anteil an Punkten an, denen die falsche Klasse zugeordnet wurde. Das geschieht

[5] Obwohl eine Menge formal gesehen keine doppelten Elemente enthält, werde ich die T_i als Mengen bezeichnen.

für alle f_i aus L. Das Mittel der Fehlerrate über alle f_i ist dann der Out-Of-Bag-FehlerE_{OOB}. Breiman beschreibt den Out-Of-Bag-Fehler als schnelle und akkurate Methode zur Fehlerabschätzung. Im Gegensatz zu anderen Testmethoden wie etwa der Kreuzvalidierung wird weder eine zusätzliche Testmenge noch der Aufbau weiterer Lerner benötigt. Vergleiche von Fehlerabschätzungsmethoden gibt es viele z.B. von Bylander et al. (2002) [17] und Kohavi et al. (1995) [29]. Im Laufe dieser Arbeit wird vor allem 10-fache Kreuzvalidierung zur Fehlerabschätzung benutzt.

Ensemble Methoden wie Bagging oder Boosting können die Klassfikationsgenauigkeit eines Lerners häufig verbessern. Laut Breiman funktioniert das im Fall von Bagging besonders gut wenn die Lerner f_i eher schwach beziehungsweise instabil sind. Das heißt, dass kleine Änderungen der Trainingsmenge zu einer großen Änderung der Eigenschaften von f_i führen können [15, vgl. Abbschnitt 1]. Genau das trifft auf die Entscheidungsbäume zu, die in Random Forest zum Einsatz kommen.

5.2. Random Tree

Kern des Random Forest sind die Entscheidungsbäume, die das Ensemble bilden. Ein Entscheidungsbaum besteht aus Knoten und Kanten mit speziellen Eigenschaften. Jeder Knoten, der kein Blattknoten ist, ist mit einem Merkmal aus dem Datensatz beschriftet. Im Fall von diskreten Merkmalen gibt es für jeden Attributwert eine Kante zu einem weiteren Knoten. Soll ein Datenpunkt g klassifiziert werden, fängt man in der Wurzel an und folgt den Kanten des Baums entsprechend der Werte von g bis man in einem Blatt angekommen ist. In dem Blatt steht dann zu welcher Klasse g gehört. Die Entstehung des Baum geschieht in Top-Down Reihenfolge. Beginnend bei der Wurzel wird dazu in jedem Knoten ein Attribut A ausgewählt und dessen Werte an die Kanten unter dem Knoten notiert. Ein Knoten teilt die Vektoren t_i aus der Trainingsmenge entsprechend ihres Wertes für das Attribut A in Teilmengen T_p ein. Für jede Menge T_p wird demnach ein neuer Knoten erzeugt, indem wieder ein neues Attribut ausgewählt werden muss. Abgebrochen wird diese Rekursion, entweder wenn die maximale Tiefe des Baums erreicht ist oder die aktuelle Menge T_p zu klein wird. Die maximale Tiefe des Baums und die minimale Größe der Trainingsmenge in einem Blatt sind frei wählbare Parameter.

Welche Merkmale in einem Blatt des Baumes stehen, wird durch ein Auswahlkriterium bestimmt. Das hier verwendete Kriterium zur Auswahl des Attributes heißt Information Gain und basiert auf Entropie der Attribute. Information Gain ist definiert als die Differenz des Informationsgehaltes einer Menge vor und nach Auswählen eines Attributs.

$$gain(a) = H(T_p) - \sum_{values(a)} P(value(a) = v) H(T_{p,value(a)=v})$$

Dabei bezeichnet $H(T_p)$ die Entropie der Menge T_p und $T_{p,value(a)=v}$ die Teilmenge von Elementen aus T_p deren Wert für das Attribut a gleich v ist. Allgemein ist die Entropie einer Zufallsvariable X definiert als

$$H(X) = -\sum_{x \in X} P(X = x) \log P(X = x)$$

Der Term $-\log P(X = x)$ ist der Informationsgehalt $Inf(x)$ des Ereignisses x. Man kann die Entropie als Erwartungswert der Information definieren.

$$H(X) = E(Inf(X))$$

Im Kontext der Klassifizierung entspricht das Ereignis x der Zugehörigkeit eines Punktes aus T_p zu einer Klasse c.

$$H(T_p) = E(Inf(T_p))$$

$$H(T_p) = -\sum_{c \in C_{T_p}} P(C = c) \log P(C = c)$$

wobei die $P(C = c)$ die Wahrscheinlichkeit des Auftretens der Klasse c in den Vektoren aus T_p ist. Das Attribut A, bei dem $gain(A)$ maximal ist, wird für den aktuellen Knoten ausgewählt. [14, Seite 117]

Weil es im Fall von numerischen Attributwerten nicht sinnvoll ist einen Knoten für jeden in T vorkommenden Wert zu erzeugen, muss ein Grenzwert gefunden werden, anhand dessen die Trainingsmenge unterteilt wird. Dazu wird die Menge aller Werte des Attributs a_{num} , alle $v_{a_{num}}$, anhand ihrer Werte sortiert. Alle Mittelpunkte m_a zwischen zwei aufeinander folgenden Werten sind mögliche Grenzwerte für eine Unterteilung.

$$gain(A) = H(T_p) - \big(\quad H(T_{p,value(a)<m_a}) + H(T_{p,value(a)\geq m_a}) \quad \big) \tag{1}$$

Information Gain wird für jeden Mittelpunkt und jedes Attribut berechnet und dasjenige bei dem $gain(A)$ maximal ist wird gewählt. Fayyad et. al. zeigten 1992, dass es genügt, bei der Berechnung des Gains nur die Mittelpunkte zwischen Werten zu betrachten deren Vektoren zu unterschiedlichen Klassen gehören.[21] In der hier genutzten Implementierung wird diese Verbesserung nicht verwendet (siehe Kapitel 6). Anders als bei den üblichen TDIDT(Top-Down Induction of Decision Trees) Verfahren, wie ID3 und C4.5, verwendet Random Tree bei der Auswahl des besten Attributes nur eine zufällige Teilmenge der gesamten Attributmenge. Die Größe dieser Teilmenge kann über einen Parameter gesteuert werden. Häufige Werte sind $\log(d)$ oder $\sqrt{(}d)$. Diese Besonderheit sorgt für die notwendige Instabilität des Verfahrens, welche Bagging benötigt.

5.3. Merkmalsgewichtung durch Random Forest

Bei der Merkmalsgewichtung versucht man den d Attributen eines Datensatzes Gewichte zuzuordnen, die beschreiben, welchen Einfluss sie auf die Klassifizierung eines Punktes haben. Breiman beschreibt eine Möglichkeit, Merkmalsgewichte durch Random Forests zu bestimmen. Die Berechnung passiert, wenn alle Entscheidungsbäume erstellt wurden. Die Out-Of-Bag Trainingsbeispiele zu jedem Baum OOB_{f_i} werden anschließend klassifiziert und die absolute Anzahl an korrekt klassifizierten Beispielen gespeichert. Dann werden erneut alle Out-Of-Bag-Daten klassifiziert, wobei zuvor die

Trainingsbeispiele verändert. Bei jedem Trainingsbeispiel, das klassifiziert wird, wird der Wert eines Attributs a geändert. Dieser wird ersetzt durch den Wert des Attributs a eines anderen zufällig ausgewählten Punktes aus der Menge T. Anders ausgedrückt wird $x_{i,a}$ auf $x_{j,a}$ gesetzt für ein zufälliges $1 \leq j \leq n$. [6] Für das Attribut a wird jetzt die Differenz zwischen der absoluten Anzahl an korrekt klassifizierten Beispiel vor und nach dem Ändern der Attributwerte gespeichert. Diese Prozedur wird für jedes Attribut $a_i \in \{a_1, ...a_d\}$ durchgeführt. Anders gesagt wird also der Wert des Attribut a künstlich verrauscht und gemessen, wie sich die Klassifizierung ändert. Ist die Differenz zwischen neuem und alten Wert niedrig oder gar negativ, hat das Attribut keinen großen Einfluss auf das Ergebnis der Klassifizierung und bekommt ein niedriges Gewicht zugeordnet. Ist die Differenz groß, wird dem Attribut entsprechend ein hohes Gewicht zugeordnet. Breiman bezeichnet diese Größe auch als Feature Importance, also Merkmalsrelevanz. Ein Vergleich dieser Merkmalsgewichtung mit anderen Verfahren folgt in Abschnitt 8.

5.4. Abstimmung mit Intrinsic Proximity

Zur Klassifizierung eines Datenpunktes wird ein einfaches Mehrheitsvotum der einzelnen Random Trees benutzt, aus denen ein Random Forest besteht. Im Allgemeinen kann ein Lerner auf verschiedenen Gruppen von Trainingspunkten aber verschiedene Klassifikationsgenaugkeiten aufweisen. Speziell bei RandomTrees kann dieser Fall auftreten. Zufälligkeit entsteht im Random Forest zum einem durch die zufällige Unterteilung der Trainingsmenge und zum anderen durch die zufällige Auswahl von Attributen bei der Generierung der Entscheidungsbäume. Dadurch können die einzelnen Random Trees sehr unterschiedliche Genauigkeiten bei der Klassifizierung aufweisen [15; 40]. Möchte man das Vertrauen in das Votum über einen Punkt x eines Klassifizierers erhöhen, ist es sinnvoll seine Ergebnisse bezüglich ähnlicher Trainingsbeispiele zu messen. Hier setzen die verschiedenen Ideen zur Verbesserung von Bagging und speziell Random Forest an. Insbesondere wird hier eine Methode betrachtet, die Robnik-Ŝikonja in seinem Artikel „Improving Random Forests“beschreibt. Darin schlägt er vor, die Bäume bei der Abstimmung so zu Gewichten, dass Bäume mit einer höheren Genauigkeit mehr Einfluss auf das Ergebnis der Abstimmung haben. Die Gewichte der Bäume werden für jeden zu klassifizierenden Punkt neu berechnet. Ähnliche Ansätze verfolgt auch Tsymbal et al. in „Dynamic Integration with Random Forests“[40] Für die Messung der Genauigkeit der Bäume bei der Klassifizierung eines Punktes x werden die k Punkte aus der Menge der Trainingsbeispiele, die x am ähnlichsten sind, gesucht. Ŝikonja nutzt dafür ein von Breiman vorgeschlagenes Ähnlichkeitsmaß [vgl. 36, S. 8] Zwei Punkte gelten demnach als ähnlich, wenn sie von einen Baum im selben Blattknoten klassifiziert werden. Dieses Ähnlichkeitsmaß, im Folgenden Intrinsic Proximity, beschreibt also die Ähnlichkeit zweier Punkte zueinander in Abhängigkeit von der Struktur der Random Trees in einem Random Forest. Intrinsic Proximity ist allerdings kein Distanzmaß und beschreibt demzufolge keine Metrik. Insbesondere können zwei Punkte nach der Intrinsic Proximity getrennt sein, auch wenn sie nach

[6] $i = j$ wird hier nicht explizit ausgeschlossen

euklidischer Distanz nah beieinander liegen. Punkte, die auf unterschiedlichen Seiten der Klassifikationsgrenze liegen, sind nach diesem Maß verschieden [40, Abschnitt 3.2].

Bei der Klassifizierung eines Punktes x wird nun wie folgt vorgegangen. Es werden die k ähnlichsten Punkte N_x zu x aus der Trainingsmenge T gesucht. Der Punkt x wird von jedem Baum im Ensemble einmal klassifiziert, um die Ähnlichkeiten zu berechnen. Landet x in einem Blattknoten, wird der Ähnlichkeitswert $\text{sim}(x, x_i)$ zwischen x und allen Punkten x_i die das Blatt bilden, um eins erhöht. Als nächstes wird die margin Funktion berechnet. Für ein Ensemble von Klassifizierern ist die Funktion definiert als

$$\text{margin}(x) = P(f_i(x) = c) - \max P(f_i(x) = c_f) \quad \forall\colon c_f \neq c \qquad \forall\colon f_i \tag{2}$$

Dabei bezeichnet c die korrekte Klasse zum Punkt x. Margin ist die Differenz zwischen dem Anteil richtiger Prädiktionen und dem maximalen Anteil von falschen Prädiktionen für eine Klasse.[7] Die margin Funktion für einen Baum f_i und nur einen Punkt x ist definiert als

$$\text{margin}(f_i, x) = \begin{cases} 1 & \text{wenn } f_i(x) = c \\ -1 & \text{wenn } f_i(x) = c_f \quad c_f \neq c \end{cases}$$

Das folgt direkt aus der Definition für margin oben. Wie oben ist c wieder die korrekte Klasse der Instanz x und c_f die falsche. Ŝikonja benutzt den durchschnittlichen Wert für $\text{margin}(f_i, x)$ über alle $x \in NO_{x,f_i}$) für das Gewicht $w_{f_i}(x)$ des Baumes f_i.

$$w_{f_i}(x) = \frac{\sum_{x_i \in NO_{x,f_i}} \text{margin}(x, f_i)}{|NO_{x,f_i}|} \tag{3}$$

NO_{x,f_i} bezeichnet den Schnitt von N_x und OOB_{f_i} und umfasst somit alle k ähnlichsten Punkte zu x aus der Trainingsmenge die in der Out-Of-Bag Menge von Baum f_i liegen. Für den Fall, dass $N_x \cap OOB_{f_i} = \emptyset$ ist, wird w_{f_i} als 1 definiert. Informell bekommen die Bäume bei der Abstimmung eine hohes Gewicht, die auf der Menge N_x die besten Ergebnisse erzielen. Ŝikonja konnte durch diese Anpassungen eine Verbesserung der Klassfikationsgenauigkeit auf manchen Datensätzen erreichen.

Tsymbal benutzt den Wert für margin nicht direkt, sondern berücksichtigt auch die jeweilige Intrinsic Proximity der Datenpunkte für die Gewichtung. Das Gewicht eines Baumes f_i bei der Abstimmung über einen Punkt x ist nach Tsymbals Methode

$$w_{f_i}(x) = \frac{\sum_{x_i \in NO_{x,f_i}} \left(\text{sim}(x, x_i)^3 \cdot \text{margin}(f_i, x_i)\right)}{\sum_{x_i \in NO_{x,f_i}} \text{sim}(x, x_i)^3} \tag{4}$$

In dieser Arbeit wird die Variante von Tsymbal verwendet. In Abbschnitt 9 wird das Verfahren empirisch mit der normalen Merheitsabstimmung und dem Clustered Voting (folgender Abschnitt) verglichen.

Theoretisch lässt sich die Intrinsic Proximity auf ein weicheres Abstandsmaß verallgemeinern. Zwei Punkte, die Beispielsweise in benachbarten Blättern eines RandomTrees klassifiziert werden, sind zwar nach Intrinsic Proximity nicht Ähnlich, die

[7] In den Worten von Breiman: „The margin measures the extent to which the average number of votes [...] for the right class exceeds the average vote for any other class."

Attributwerte der beiden Punkte können dabei aber gleichartig sein. Man könnte die Abstandsberechnung der Intrinsic Proximity so erweitern, dass je nach Abstand der Blätter auch Ähnlichkeitswerte > 0 für Punkte entstehen, die nicht im selben Blatt des Entscheidungsbaums liegen. Dieses Verfahren birgt einige Nachteile. Bei benachbarten Blättern muss beachtet werden, dass die Punkte in den Blättern dann nicht unbedingt die selbe Klassenzugehörigkeit besitzen. Außerdem ist die Laufzeit der gewichteten Abstimmung nach Intrinsic Proximity bereits wesentlich höher als bei der einfachen Merheitsabstimmung. Die Erweiterung der Abstandsberechnung würde diese Laufzeit noch weiter erhöhen. Eine etwas genauere Betrachtung der Laufzeiten folgt in Abschnitt 6. Im Folgenden wird eine alternative Methode zur Abstimmung mittels Intrinsic Proximity vorgestellt.

5.5. Abstimmung durch Clustering

Die Grundidee der gewichteten Abstimmung nach Tsymbal und Ŝikonja ist das Gewicht eines Baumes anhand der k ähnlichsten Punkte aus der Trainingsmenge zu bestimmen. Bei der oben beschriebenen Methode muss diese Menge für jeden zu klassifizierenden Punkt x neu berechnet werden. Die Idee ist, die Nachbarschaften für jeden Punkt im Vorhinein durch Clustering zu bestimmen. Ein Clustering-Algorithmus teilt eine Punktmenge anhand eines Ähnlichkeitsmaß in Teilmengen (Cluster) verschiedener Größe ein. Der optimale Clustering-Algorithmus erzeugt Cluster, so dass die paarweise Ähnlichkeit zwischen Punkten innerhalb eines Clusters geringer ist, als die zweier Punkte aus verschiedenen Clustern.[27, Seite 507] Im folgenden wird das Ähnlichkeitsmaß einfach über die euklidische Norm definiert. Zwei Punkte sind demnach Ähnlich wenn sie nah beieinander liegen. Hier wird das bekannte k-means Verfahren benutzt welches im folgenden kurz erklärt wird. Das k-means Verfahren teilt die Punktmenge T iterativ in k Teilmengen $M_1, \ldots, M_k$ auf. Dafür werden im ersten Schritt k Punkte, genannt Zentroide, initialisiert. Sei Z_i so ein Zentroid. In Schritt zwei werden alle Punkte, die Z_i am Ähnlichsten sind, der Menge M_i hinzugefügt. In Schritt drei werden neue Zentroide für jedes M_i bestimmt. Die neuen Z_i' werden auf den Schwerpunkt der Menge M_i gesetzt. Im Fall euklidischer Distanz ist demnach

$$Z_i' = \frac{1}{|Z_i|} \sum_{x \in M_i} x.$$

Dann wird $Z_i = Z_i', M_i = \emptyset$ gesetzt und der erste Iterationsschritt ist beendet. Die Schritte zwei und drei werden nun so oft wiederholt, bis sich die Mengen M_i in einer Iteration nicht mehr ändern.[27, Kapitel 13.2.1] Die Qualität der Cluster hängt stark von den initialen Zentroiden ab. Eine von David Arthur und Sergei Vassilvitskii vorgeschlagene Verbesserung namens k-means++ verbessert die Wahl von Startzentroiden.[12] Dabei wird wieder Iterativ vorgegangen. Zuerst wird wieder ein Startpunkt Z_0 zufällig aus der Menge gewählt. Sei $D_z(x)$ die Distanz zwischen einem Punkt x und seinem nächsten, schon gewählten, Zentroid. Wähle anschließend x als neuen Zentroid mit Wahrscheinlichkeit

$$P(x) = \frac{D_c(x)^2}{\sum_{x \in T} D_c(x)^2}.$$

Das bedeutet, dass die Wahrscheinlichkeit für x mit zunehmender Entfernung zu bisherigen Zentroiden ansteigt. Dadurch wird es weniger Wahrscheinlich, dass zwei initiale Zentroide in einem Gebiet liegen, welches, nach optimaler Lösung, zum selben Cluster gehört. Abbildung 9 soll den Unterschied der Verfahren veranschaulichen.

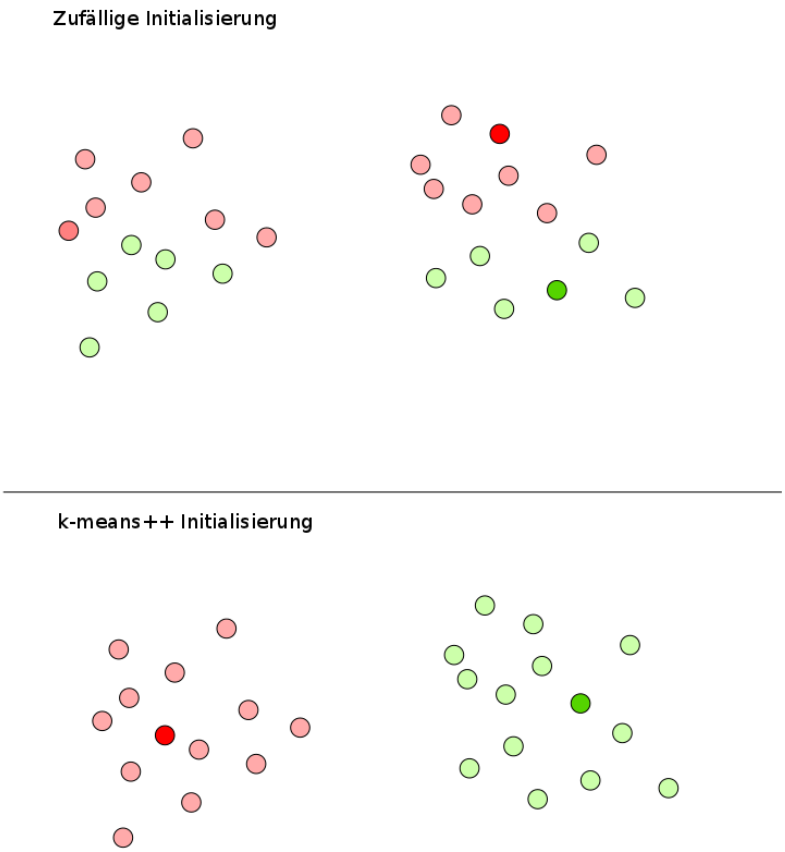

Abbildung 9: Zwei Cluster die durch k-means gebildet Werden. Oben nach Auswahl zufälliger Startzentroide. Unten nach k-means++ Verfahren.

Da hier nur binäre Klassifikationsprobleme betrachtet werden, wird im folgenden k , die Anzahl der Cluster, auf zwei gesetzt. Um nun die Entscheidungsbäume für eine Abstimmung im Random Forest zu gewichten wird, analog zur Gewichtung mit Intrinsic Proximity, wieder eine Menge N_x mit zu x ähnlichen Punkten gebildet. Ähnlich gelten diesmal alle Punkte, die im selben Cluster liegen. Die Menge N_x entsteht nun durch zufälliges Ziehen mit zurücklegen von Punkten, die sich im selben Cluster befinden

wie x. Die Größe von N_x kann mit dem selben Parameter angepasst werden wie bei der Intrinsic Proximity. Anschließend wird wie bei Ŝikonja das Baumgewicht über die Gleichung 3 bestimmt.

5.6. Fehlergewichtete Abstimmung

Wie in Abschnitt 5.1 gesehen, kann der Out-Of-Bag-Fehler E_{OOB} im Ensemble berechnet werden. Der Fehler soll eine Abschätzung des eigentlichen Klassifizierungsfehlers sein. Während der Berechnung von E_{OOB} kann, für jeden Baum f_i die Differenz zwischen $|OOB_{f_i}|$, der Größe der Out-Of-Bag Menge, und der Anzahl der korrekt klassifizierten Punkte aus OOB_{f_i} gespeichert werden. Formal geschrieben ist die Differenz

$$d_{f_i} = |OOB_{f_i}| - \sum_{x \in OOB_{f_i}} \mathrm{I}(f_i(x) = c_x).$$

wobei I die Indikatorfunktion und c_x die tatsächliche Klasse des Punktes x ist. Bei der fehlergewichteten Abstimmung wird das Baumgewicht auf den quadrierten Kehrwert der Differenz

$$w_{f_i} = \frac{1}{d_{f_i}^2}$$

gesetzt. Die Gewichte werden also nicht für jeden Punkt neu berechnet, sondern nur einmal durch die gesamte Trainingsmenge bestimmt.

6. Implementierungsdetails

Die Erweiterung des Random Forest Algorithmus geschieht direkt im WEKA Quellcode. Die WEKA-Extension bindet WEKA in RapidMiner ein. Sie besteht aus Adapterklassen die direkt auf die Methoden von WEKA zugreifen. Dazu wird einfach eine als .jar Datei vorkompilierte Version von WEKA dem Klassenpfad hinzugefügt. Verwendet wird die Version 3.7.3 von WEKA. Das gewichtete Abstimmungsverfahren nach Ŝikonja [36] und die Merkmalsgewichtung wurden implementiert, wie in Abschnitt 5.4 und 5.3 beschrieben.

6.1. Parallelität

Seit Version 3.7.0 unterstützt WEKA unter anderem die parallele Ausführung von einigen Ensemble Methoden wie Bagging. Diese Metalerner beziehungsweise Ensembleobjekte verwalten eine Menge von einzelnen Klassifizierern und aggregieren deren Ergebnisse. Wenn das Ensemble parallel ausgeführt werden soll, werden die einzelnen Basislerner in einem ThreadPoolExecutor gesammelt. In der Trainingsphase werden die Trainingsdaten an die Basisklassifizierer verteilt und der Trainingsprozess in jedem Klassifizierer gestartet. Der ThreadPoolExecutor kümmert sich dabei um die Verteilung der Aufgaben auf freie Threads. Die Verwaltung durch den ThreadPoolExecutor hat einige Vorteile. Würde man jedem Basisklassifizierer einen eigenen Thread zuweisen wäre der Overhead von Speicher- und Verwaltungsaufwand durch das Betriebssystem wesentlich größer. Durch den ThreadPool können die Objekte im Speicher gehalten werden ohne dabei jedes mal über Systemaufrufe Threads zu zerstören oder zu starten. Durch die Flexibiltät des Java ThreadPools lassen sich die Ensembles auch auf Maschinen oder Rechenclustern mit einer großen Anzahl an Prozessoren ausführen.

6.2. Gewichtete Abstimmung mit Intrinsic Proximity

Für die Berechnung der gewichteten Abstimmung wurde der Quellcode für das Bagging angepasst. Die Gewichte der einzelnen Bäume bei der Abstimmung werden wie in Formel 4 berechnet. Das erstellen der Liste l_{sim}, also der Liste mit den k ähnlichsten Punkten zu einem Punkt x bedarf etwas Arbeit.

Die Punkte der Trainingsmenge werden mit IDs von $1 \dots |T|$ versehen. Die Nummern entsprechen den Positionen in der Liste l_{sim}, welche bei der Erstellung des Random Forests angelegt wird. Zuvor wurde schon bei der Erstellung der einzelnen Random Trees in jedem Blattknoten gespeichert, welche IDs aus den Trainingsdaten das Blatt bilden. Wird der Punkt x von einem Baum f_i klassifiziert, steigt der Punkt solange den Baum herab, bis er in einem Blattknoten angelangt ist. In l_{sim} werden nun alle Werte um eins inkrementiert, deren IDs beziehungsweise Listenpositionen im Blatt vermerkt sind. Wurde x von allen Bäumen klassifiziert, enthält l_{sim} die Ähnlichkeitswerte zwischen dem Punkt x und allen Punkten der Trainingsmenge. Die Elemente aus den Trainingsdaten mit den k größten Werten in l_{sim} bilden die Menge N_x. Nach Ausführung hat man jetzt alles was man braucht um $w_{f_i}(x)$ (Gleichung 4) zu berechnen.

Die minimale Anzahl an Punkten, die ein Blatt bilden, kann durch einen Parameter festgelegt werden (siehe auch Abschnitt 5.2). Wird dieser als konstant angenommen, benötigt die Berechnung der Ähnlichkeiten zusätzlich $\mathcal{O}(|L| \cdot b + |T|)$ Speicherplatz, wenn b die durchschnittliche Anzahl der Blätter in einem Baum bezeichnet und $|T|$ die Länge der Liste l_{sim} ist.Um die k größten Werte in l_{sim} zu finden, wird die Liste zunächst absteigend sortiert und dann werden die ersten k Positionen ausgewählt. Das heißt, dass für die Klassifizierung eines Punktes zusätzlich $\mathcal{O}(|T| \log |T|)$ Zeit verbraucht wird. Die Benutzung der Trainingsmenge während der Klassifizierung neuer Punkte, bedeutet zusätzlichen Speicherplatzbedarf. Da die Trainingsdaten jetzt Teil des Modells sind und nach der Trainingsphase nicht mehr verworfen werden können.

6.3. Gewichtete Abstimmung durch Clustering

Der hierbei verwendete Algorithmus für das k-means-Clustering kommt aus WEKA. Diese Implementierung erlaubt es auch, die Initialisierung nach dem k-means++-Verfahren zu nutzen. Der Cluster wird einfach in die Klasse für den Random Forest eingebunden. Die Parameter für das k-means-Verfahren können über den Operator eingestellt werden. Die Cluster werden vor Trainingsphase gebildet, so dass sich die Laufzeit für die Bildung des Klassifizierunsmodell erhöht. In der Praxis überwiegt die benötigte Zeit für das Training der Entscheidungsbäume jedoch deutlich gegenüber dem k-means-Clustering. Die Cluster müssen im Modell gespeichert werden, wodurch mindestens linear viel Speicherplatz zusätzlich benötigt wird. Bei der Klassifizierung eines neuen Punktes x wird zunächst der dazugehörige Cluster bestimmt. Dann werden zufällig k Punkte aus dem Cluster zu l_{sim} hinzugefügt und danach die Baumgewichte über Gleichung 3 bestimmt. Gegenüber der Abstimmung durch Intrinsic Proximity muss die Nachbarschaft also nur einmal berechnet werden. Auch bei dieser Methode muss die Trainingsmenge weiter im Modell gespeichert bleiben.

6.4. Fehlergewichtete Abstimmung

Die Werte für d_{f_i} werden direkt nach der Erstellung des Random Forest durch die Trainingsdaten berechnet. Das Array, welches die d_{f_i} hält, benötigt zusätzlich $\mathcal{O}(L)$ Platz. Die Trainingsdaten müssen allerdings nicht mehr gespeichert werden. Die Berechnung der Gewichte geschieht auch nur einmal und nicht für jeden zu klassifizierenden Punkt aufs Neue. Die Laufzeit gegenüber der Merheitsabstimmung ändert sich kaum weil die d_{f_i} zusammen mit dem Out-Of-Bag-Fehler berechnet werden.

Diagramm 10 zeigt, wie sich die Laufzeit der Abstimmungsmethoden in der Praxis bemerkbar macht.

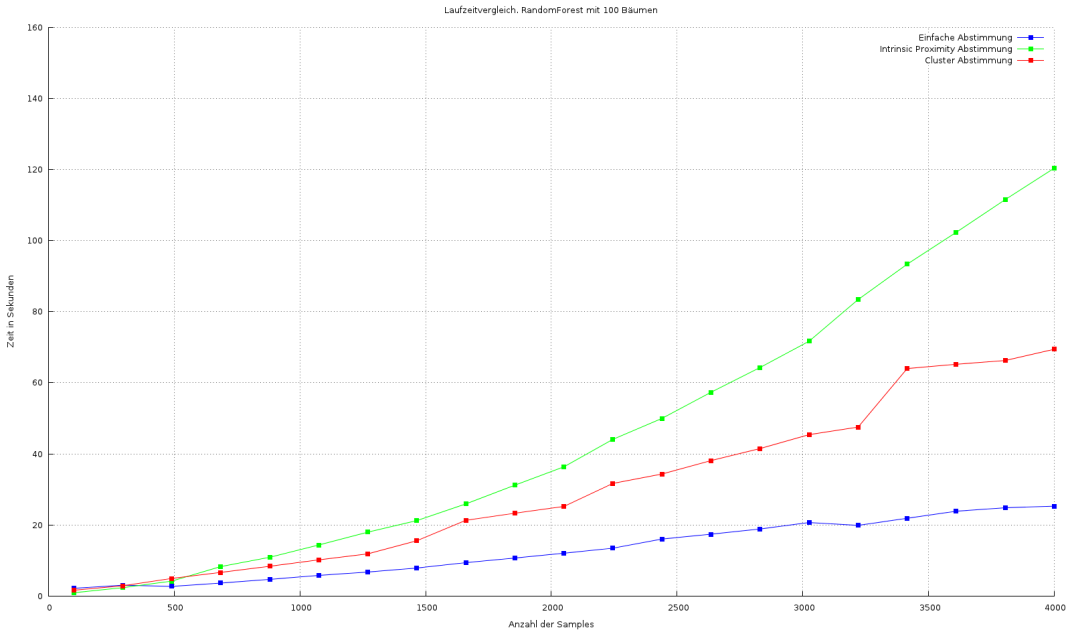

Abbildung 10: Laufzeit eines Random Forest, mit 100 Bäumen, mit einfacher Merheitsabstimmung (blau), Cluster Abstimmung (rot) und Intrinsic Proximity Abstimmung (grün). Die fehlergewichtete Abstimmung wurde vernachlässigt da kein Unterschied zu erkennen war.

6.5. Merkmalsgewichtung

Die Merkmalsgewichtung wird ausgeführt, nachdem der Random Forest erstellt wurde. Wie in Abschnitt 5.3 beschrieben, wird einfach jeder Punkt aus den Trainingsdaten erneut klassifiziert, wobei vorher der Wert eines Attributes verändert wurde. Für jedes Attribut wird einmal komplett über die Trainingsmenge iteriert. Jedes einzelne Trainingsbeispiel wird zweimal vom Ensemble klassifiziert, einmal vor Änderung und einmal nach der Änderung des Attributwertes. Die Differenz zwischen den Klassifikationen wird für jedes Attribut in einer Liste gespeichert, die anschließend anhand des Maximums normalisiert wird. Ist die Größe des Ensembles konstant, benötigt die Berechnung der Merkmalsgewichte demzufolge $\mathcal{O}(|T| \cdot |A|)$ Zeit. Auch für diese Berechnung wurde nur der Code für Bagging angepasst. Es können demzufolge auch andere Basislerner als Random Tree benutzt werden, um Merkmalsgewichte zu bestimmen. Die Klasse für Random Forest wurde nur erweitert, um zusätzliche Parameter für den Benutzer auf der graphischen Oberfläche darzustellen.

7. Verfahren für den Vergleich

Im nächsten Abbschnitt sollen die implementierten Erweiterungen von Random Forest empirisch verglichen werden. Besonders bei der Merkmalsgewichtung kommen dabei viele zusätzliche Verfahren zum Einsatz, die im Folgenden erläutert werden. In praktischen Anwendungen sind Daten häufig hochdimensional. Das heißt, dass viele Merkmale nötig sind, um einen Datenpunkt eindeutig zu beschreiben. Im Kontext des maschinellen Lernens soll ein fester Anteil des gesamten Volumens der möglichen Datenpunkte durch Trainingsbeispiele beziehungsweise Beobachtungen beschrieben werden. Die Anzahl der dafür benötigten Beobachtungen steigt dabei mit zunehmender Dimension exponentiell an. Eine Analyse dieser hochdimensionaler Daten mit automatischen Analysemethoden ist ohne Dimensionsreduzierung deshalb häufig nicht effizient möglich. Das Problem ist auch unter dem Begriff „Fluch der Dimensionen“bekannt, welcher von Richard Bellmann eingeführt wurde. [8] Ein Beispiel für hochdimensionale Daten sind die Ergebnisse aus Untersuchungen mit DNA-Microarrays, die einige tausend Dimensionen haben können.[19, vgl. S. 1] Die Merkmalsgewichtung ordnet den Merkmalen eines Datensatzes Gewichte zu, die den Einfluss des Merkmals auf die Klassifizierung darstellen. Merkmale, die keinen oder einen gar negativen Einfluss auf die Klassifizierung haben, sollen auf diese Weise erkannt werden. Die Dimensionsreduzierung erfolgt dann über die explizite Auswahl einer Teilmenge von Merkmalen. Eine automatische Klassifizierung der Daten soll auch nach der Merkmalsselektion genau so gut oder sogar besser als vorher funktionieren. Später wird die Gewichtung durch Random Forest mit anderen Verfahren experimentell verglichen. Dazu werden Merkmale Selektiert und anschließend, unter Berücksichtigung dieser Merkmale, die Klassifikationsgüte gemessen. Dabei kommt neben der Klassifikation durch Random Forest auch die bekannte Naive Bayes Methode zum Einsatz. Als vergleichende Merkmalsselektionen werden, neben Information Gain Ration und der SVM-Gewichtung, auch drei Verfahren aus der Feature Selection Extension von RapidMiner getestet. [38] Diese sind mRMR, SAM und Shrunken-Centroid, die alle im folgenden kurz erläutert werden. Auch die Gewichtung durch LASSO/LARS[9] wurde getestet, brachte allerdings keine Ergebnisse. Der Grund dafür wurde nicht untersucht. Als nächstes wird die Naive Bayes Klassifikation erklärt.

7.1. Naive Bayes Klassifikation

Um die Performanz der Merkmalsgewichtungen zu testen, werden die Testdaten neben SVM und Random Forest auch mit dem Naive Bayes Klassifizierer getestet. Die Idee hinter dieser Methode basiert auf der Annahme, dass alle Variablen unabhängig voneinander sind. In „Machine Learning “von T. Mitchell [31] wird das Verfahren wie folgt erklärt. Die Wahrscheinlichkeit für die Beobachtung einer Klasse C gegeben der

[8] Donoho [19] liefert einige interessante Erklärungsansätze dazu.
[9] LARS oder Least Angle Regression produziert auch LASSO Lösungen

Merkmalswerte A_i kann durch das Theorem von Bayes geschrieben werden als:

$$P(C \mid A_1 \dots A_m) = \frac{P(C) \cdot P(A_1 \dots A_m \mid C)}{\sum_{c \in C} P(C = c) P(A_1 \dots A_m \mid C = c)}$$

Da nach Annahme alle A_i unabhängig sind, gilt

$$P(A_1 \dots A_m \mid C = c) = \prod_i P(A_i \mid C = c)$$

Die Wahrscheinlichkeiten der $P(A_i \mid C = c)$ und $P(C = c)$ lassen sich aus den Trainingsdaten schätzen. Soll die Instanz mit den Merkmalen $A_1 \dots A_m$ klassifiziert werden, muss nur das c gefunden werden, das $P(C \mid A_1 \dots A_m)$ maximal werden lässt. Im Fall binärer Klassifikation lässt sich 7.1 auch anders ausdrücken.

$$P(C \mid A_1 \dots A_m) = \frac{P(C) \cdot \prod_i P(A_i \mid C = c)}{P(C) P(A_1 \dots A_m \mid C) + P(\bar{C}) P(A_1 \dots A_m \mid \bar{C})}$$

Der Nenner kann weggelassen werden, da er unabhängig von C ist. Die Funktion

$$P(C \mid A_1 \dots A_m) = P(C) \cdot \prod_i P(A_i \mid C = c)$$

bleibt also übrig. Obwohl die Annahme, dass die Merkmale komplett unabhängig voneinander sind, in den meisten Fällen falsch ist, erreicht der Bayes-Klassifikator in der Praxis besonders im Bereich des Information Retrievals gute Ergebnisse. [32].

7.2. Merkmalsgewichtung durch SVM

Support Vector Machines (SVM) sind eines der bekanntesten und meist verwendeten maschinellen Lernverfahren. Das Verfahren versucht eine Ebene im Merkmalsraum zu finden, die die Punkte verschiedener Klassen möglichst gut voneinander trennt. Das heißt, dass der Abstand zwischen den Punkten und der Trennebene maximiert werden soll. Die Ebene kann definiert werden durch einen Vektor $\vec{\beta}$ und eine Ebenengleichung in hessescher Normalform $\vec{\beta}\vec{x} + \beta_0 = 0$. Bei einem binären Klassifizierungsproblem können die Klassenzugehörigkeiten mit 1 oder -1 beschrieben werden. Hat man eine Ebene die die Klassen optimal trennt, so ergibt sich die Klasse eine Punktes x_g zu $\text{sign}(x_g\vec{\beta} + \beta_0)$.[27, Kapitel 4.5] Ein Vorteil der Support Vector Machines besteht darin, dass das Problem mithilfe von Lagrange-Multiplikatoren als konvexes Optimierungsproblem dargestellt werden kann, welches sich relativ effizient lösen lässt. Die libSVM Implementierung, die auch in RapidMiner zur Verfügung steht, nutzt beispielsweise die SMO (Sequential Minimal Optimization) Methode, die im Jahr 1998 von John Platt erdacht wurde, um dieses spezielle Optimierungsproblem für SVMs zu lösen. [33; 18] Die Lage der Trennebene wird nicht durch alle Punkte des Datensatzes beeinflusst, sondern nur von denen, die der Trennebene am nächsten liegen. Diese sogenannten Stützvektoren (support vectors) geben der Methode ihren Namen. Trennebenen können nur gefunden werden, wenn die zu Grunde liegenden Daten linear

separierbar sind, was aber in der Praxis häufig nicht der Fall ist. Abhilfe schafft der sogenannte Kernel-Trick mit dem die Punkte in einen Raum mit höherer Dimension transformiert werden können. In diesem neuen Merkmalsraum können die Daten dann eventuell durch eine Ebene getrennt werden. Durch die Transformation in den neuen Merkmalsraum ist es also möglich, dass die Daten im eigentlichen Raum durch eine nicht lineare oder sogar nicht zusammenhängende Funktion getrennt werden. Zur Merkmalsgewichtung kann allerdings nur die lineare Variante genutzt werden. Die Gewichte der Merkmale ergeben sich direkt durch die Koeffizienten der berechneten Trennebene.

7.3. Merkmalsgewichtung durch Information Gain Ration

Wie schon in Abschnitt 5.2 gesehen, ist der Information Gain als die Differenz zwischen der Entropie der gesamten Trainingsmenge und der Entropie der Trainingsmenge, gegeben ein Attribut a, definiert.

$$gain(a) = H(T_p) - \sum_{values(a)} P(value(a) = v)H(T_{p,value(a)=v})$$

Für jedes Attribute wird $gain(a)$ berechnet und so das Merkmalsgewicht gebildet. Die Daten aus den IceCube Simulationen sind numerisch. Das heißt, auch hier wird wieder $gain(a)$ für jeden möglichen Zwischenwert berechnet, wie bereits in Abschnitt 5.2 beschrieben. Information Gain bevorzugt Attribute, die mehrere Werte besitzen gegenüber Attributen mit wenigen Werten. Information Gain Ratio gleicht diesen Bias aus, indem durch die Entropie des betrachteten Attributs geteilt wird.[14, Seite 118]

$$gainratio(a) = \frac{gain(a)}{H(a)}, \quad H(a) = -\sum_{values(a)=v} P(value(a) = v) \log P(value(a) = v)$$

Im Folgenden wird Information Gain Ratio nur noch als Information Gain oder noch kürzer IG bezeichnet.

7.4. Merkmalsgewichtung durch SAM

Wie zuvor erwähnt, können die Daten aus Experimenten mit DNA-Microarrays zehntausende von Merkmale besitzen. In dem Paper „Significance analysis of microarrays applied to the ionizing radiation response“[41] wird eine Methode vorgeschlagen um Aussagen über die statistische Signifikanz von Expressionsleveln in DNA-Microarrays zu treffen. Für jedes Gen i wird ein Gewicht wie folgt berechnet:

$$d_i = \frac{r_i}{s_i + s_0}$$

Dabei ist s_i eine normierte Standardabweichung und s_0 ein frei wählbarer Programmparameter. Die Zähler r_i ist die Differenz der mittleren Genexpression für die verschiedenen Phänotypen. Auf eine binäre Klassifizierung reduziert, sind die Gene als Merkmale zu verstehen und die Phänotypen als die Klassen des Klassifizierungsproblems.

7.5. Merkmalselektion durch nearest shrunken centroid

Eigentlich handelt es sich dabei um ein Klassifizierungsverfahren, welches ursprünglich zur Analyse von DNA-Microarraydaten entwickelt wurde.[39] Zu jeder Klasse des Datensatzes werden die Zentroiden der Daten gebildet. Der Zentroid $z_C$für eine Klasse C ist der Schwerpunkt aller Daten, die zu dieser Klasse gehören.

$$\mathbf{z_C} = \frac{1}{|T_C|} \sum_{x \in T_C} \begin{pmatrix} x_{a_1} \\ x_{a_2} \\ \vdots \\ x_{a_d} \end{pmatrix}$$

Dabei sind $x \in T_C$ die Punkte, aus den Trainingsdaten T, die zur Klasse C gehören. Betrachtet man nur eine Komponente von z_C, so erhält man den Zentroiden für ein Attribut $z_{C,a}$. Die Differenz zwischen $z_{C,a}$ und dem Zentroid, des Attributes, für alle Klassen z_a wird normiert durch die Standardabweichung von a innerhalb der Klasse C.

$$d_{a,C} = \frac{z_{a,C} - z_a}{m_C \cdot (\sigma_a + \sigma_0)}$$

Die Größe σ_a ist die Standardabweichung und σ_0 eine wählbare Konstante. Bis hierhin ist das Verfahren sehr ähnlich zu SAM. Ist dieser Wert für $d_{a,C}$ einmal berechnet, werden neue Zentroiden berechnet. Stellt man 7.5 um, so erhält man

$$z_{a,C} = z_a + m_C(\sigma_a + \sigma_0) \cdot d_{a,C}$$

Das Gewicht $d_{a,C}$ wird dann durch einen konstanten Wert Δ *geschrumpft* und dadurch neue Zentroiden $z'_{a,C}$ für jede Klasse berechnet. Die Anzahl an Klassen, für die $z'_{a,C} \neq z_a$ ist, bilden das Gewicht des Attributes a. Im Fall binärer Klassifikation entspricht dies einer direkten Merkmalselektion. Die Anzahl der selektierten Merkmale ist dabei nur von Δ abhängig. Im Folgenden wird das Verfahren mit PAM abgekürzt.

7.6. Merkmalselektion durch mRMR

Diese Methode wählt Merkmale anhand der gegenseitigen Information (Mutual Information) zwischen den Merkmalen und der Klasse aus. Das verwendete Kriterium nennt sich „minimum Redundancy Maximum Relevance“(mRMR). Ziel ist es, die Merkmale zu finden, die die höchste Relevanz bezüglich der Klasse haben und gleichzeitig möglichst wenig redundant sind. Beides lässt sich über die Mutual-Information ausdrücken.

$$mutualInformation(A, B) = H(A) - H(A \mid B)$$

Der Ausdruck $H(A \mid B)$ ist die bedingte Entropie zwischen den Zufallsvariablen A und B und ist definiert als

$$H(A \mid B) = \sum_{b \in B} P(B = b) \cdot H(A \mid B = b),$$

wobei $H(A \mid B = b)$ dann wieder die bekannte Definition für Entropie ist (siehe 5.2).

$$H(A \mid B = b) = -\sum_{a \in A} P(A = a \mid B = b) \log P(A = a \mid B = b)$$

Die gegenseitige Information $mI(a, c)$ zwischen einem einzelnen Attribut a und einer Klasse c ist dementsprechend definiert als

$$\begin{aligned} mI(a, c) &= H(a) - H(a \mid c) \\ &= p(a) \cdot \log p(a) \quad - \quad p(a \mid b) \log p(a \mid b) \\ &= Inf(a) - Inf(a \mid b). \end{aligned}$$

Dies wird jetzt benutzt um die Relevanz einer Attributmenge A und einer Klasse c zu erfassen und ist nach Peng et Al. [32] definiert als

$$L(A, c) = \frac{1}{|A|} \sum_{a \in A} mI(a, c).$$

Die Redundanz der Merkmale in einer Attributmenge a untereinander wird mit

$$R(A) = \frac{1}{|A|^2} \sum_{a_i, a_k \in A} mI(a_i, a_k)$$

berechnet. Der mRMR Algorithmus versucht näherungsweise das Optimum für eines der folgenden Kriterien zu finden

$$\begin{aligned} &\max(L(A, c) - R(A)), \\ &\max\Big(\frac{L(A, c)}{R(A)}\Big). \end{aligned}$$

Erkennbar ist, dass die Relevanz von A eher groß und die Redundanz eher klein sein muss, um einen möglichst guten Wert für das Optimalitätskriterium zu erreichen. In dieser Arbeit wird nur das zweite Kriterium verwendet. Im nächsten Kapitel werden die Gewichtungsverfahren empirisch miteinander verglichen. mRMR dient dabei als Vergleich für die Merkmalselektion anhand der Gewichte.

8. Vergleich von Merkmalsgewichtungen

Die empirischen Vergleiche werden mit insgesamt 5 Datensätzen mit RapidMiner durchgeführt. Um eine Vergleichbarkeit zu den IceCube Monte-Carlo Daten zu schaffen, werden nur binäre Klassifikationsprobleme mit reelwertigen Attributen betrachtet. So kommen neben den IceCube-Daten noch Datensätze aus dem UCI Data Repository hinzu [23].

Ionosphere:
Daten aus Radaruntersuchung der Ionosphäre. Binäre Klassifikation anhand von 34 reelwertigen Variablen. Insgesamt 351 Instanzen, davon 225 mit positiver Klasse und 126 mit negativer Klasse.

Sonar:
Sonarreflektionen von Steinen und Metall bilden insgesamt 60 reelwertige Merkmale. Von den 208 Instanzen gehören 111 zu Metall und 97 zu Gestein.

BreastCancerDiagnostic (BCD):
Bilder von Gewebeproben, aus denen 32 Merkmale der untersuchten Zellen gewonnen wurden. Der Datensatz enthält 569 Beispiele, davon 212 mit malignösen und 357 mit gutartigen Zellen.

Magic:
Daten des Magic Cherenkov Teleskops. Ziel ist die Klassifizierung der Bilder in hadronischen Hintergrund und Signalen von Gammastrahlen. Die Daten wurden wie bei IceCube durch das CORSIKA Programm erzeugt. Von den 19020 Beispielen sind 12332 Gamma Events und 6688 Hintegrund. Die Bilder werden durch 10 Reelwertige Merkmale beschrieben.

Ice-21 Von CORSIKA simulierte Daten für die 56 String Konfiguration von IceCube. Auf die Daten wurde bereits eine Merkmalsauswahl durch das mRMR Verfahren angewendet[37, vgl.]. Der Datensatz hat 21 Variablen und insgesamt 16659 Beispiele, davon 8332 Positiv und 8327 Negativ.

Ice-475 Von CORSIKA simulierte Daten. Dieser Datensatz enthält 475 Merkmale die aus verschiedenen Rekosntruktionsmethoden gewonnen wurden.Insgesamt besteht des Datensatz aus 60 000 Punkten. 156 der Attribute haben mindestens einen fehlenden Wert.

8.1. Stabilität der Random Forest Merkmalsgewichtung

Zunächst soll die Stabilität der Zahlenwerte der Methoden bei unterschiedlichen Eingaben verglichen werden. Dazu wird der Datensatz durch “stratified Sampling„(stratifizierte Zufallsstichprobe) in Teilmengen kleinerer Größe aufgeteilt. Die Verteilung der Klassenzugehörigkeiten des Datensatzes ändert sich dabei nicht. Auf jeder Teilmenge werden Merkmalsgewichte durch die Verfahren berechnet. Es werden insgesamt 10 Teilmengen gebildet, auf denen die Merkmalsgewichte berechnet werden. Die

Korrelation zwischen allen Teilmengen[10] wird ausgerechnet und Mittelwert und Standardabweichung gebildet. Die Korrelation wird durch Pearsons Korrelationskoeffizient bestimmt. Um auch den Rang der Gewichte zu berücksichtigen, wird zusätzlich der Rangkorrelationskoeffizent nach Spearman gebildet. [28, vgl. Kapitel 2] Die Größe der Teilmengen betrug jeweils 5000. Die verschiedenen Gewichtungmethoden wurden auf den selben Teilmengen durchgeführt. Die Parameter in den folgenden Tests sind immer die Gleichen. Der Random Forest besteht aus 100 Bäumen und die Anzahl der berücksichtigten Merkmalen pro Knoten beträgt $\lfloor log(|A|+1) \rfloor$. Der s_0-Wert von SAM wurde auf dem Standardwert von 0.1 belassen, wie auch der Kostenfaktor C für die Schlupfvariablen der SVM. Ein kurzer Test mit anderen Werten ergab keinen maßgeblichen Unterschied. Die Ergebnisse sind in Tabelle 1 und 2 zu sehen.

Magic

	Pearsons r	σ	Spearmans r	σ
SAM	0.9970	0.0020	0.9784	0.0209
RF	0.9909	0.0032	0.9766	0.0170
SVM	0.9784	0.0199	0.9189	0.0486
IG	0.9313	0.0443	0.8954	0.0542

Tabelle 1: Durchschnittliche Korrelationen mit Standardabweichungen verschiedener Teilmengen des Magic Datensatzes.

Ice-21

	Pearsons r	σ	Spearmans r	σ
SAM	0.9934	0.0056	0.9803	0.0147
RF	0.9743	0.0121	0.9770	0.0099
SVM	0.9447	0.0267	0.9329	0.0292
IG	0.9622	0.0203	0.8357	0.0606

Tabelle 2: Durchschnittliche Korrelationen mit Standardabweichungen verschiedener Teilmengen des Ice-21 Datensatzes.

Der Spearman-Koeffizient von Information Gain Ratio liegt bei beiden Datensätzen an letzter Stelle mit einem Wert < 0.9 . Die restlichen Werte liegen dafür deutlich über 0.9 und deuten auf eine hohe Stabilität der Verfahren auf diesen Datensätzen hin.

Um die Stabilität der direkten Selektion durch Gewichte zu testen, wird der Jaccard Index benutzt. Dieser ist definiert als $J(M_1, M_2) = \frac{|M_1 \cap M_2|}{|M_1 \cup M_2|}$ für zwei Mengen von Merkmalen M_1 und M_2. Getestet werden Merkmalsgewichtungen durch Random Forest, SAM und Information Gain Ratio auf dem Ice-475 Datensatz. Es werden jeweils die 20 Merkmale mit den höchsten absoluten Gewichten ausgewählt. Die Gewichtung durch SVM kann an dieser Stelle nicht sinnvoll eingesetzt werden, weil das Verfahren alle Attribute mit fehlenden Werten komplett verwirft. Etwa 98% der Punkte im Datensatz enthalten mindestens einen fehlenden Wert. Als Alternative kommt das mRMR

[10] Insgesamt entstehen so 55 Werte für die Korrelation.

Verfahren zum Einsatz, mit dem ebenfalls 20 Merkmale selektiert werden. Die Selektion durch PAM kann hierbei nicht durchgeführt werden, da die Zahl der selektierten Merkmale von Δ abhängt und für jede Teilmenge neu berechnet werden müsste. Die Berechnung des durchschnittlichen Jaccard-Indexes erfolgt analog zum obigen Fall. Es werden 10 Teilmengen gleicher Größe gebildet und der Jaccard-Index berechnet.

Ice-475		
	Jaccard	σ
RF	0.4126	0.0770
SAM	0.6985	0.1878
IG	0.6157	0.1366
mRMR	0.7865	0.0970

Tabelle 3: Jaccard Index verschiedener Teilmengen des Ice-475 Datensatzes.

Das mRMR Verfahren scheint bei dieser Auswahl am stabilsten zu sein. Die Standardabweichung des Jaccard-Indexes ist bei SAM und IG mit 0.1878 respektive 0.1366 deutlich höher als bei Random Forest und mRMR. Das deutet darauf hin, dass IG und SAM auf verschiedene Paaren von Teilmengen zu sehr unterschiedlichen Selektionen führen und eher instabil sind. Mit einem Wert von 0.4126 ist auch Random Forest eher instabil, auch wenn die Standardabweichung dort geringer ist. Ein Jaccard-Index von weniger als 0.5 bedeutet schließlich, dass mehr als die Hälfte der Merkmale sich unterscheiden.

8.2. Vergleich der Merkmalsgewichte von Random Forest

Als nächstes sollen die Verfahren untereinander verglichen werden. Dafür wurden die Gewichte auf den kompletten Datensätzen berechnet und per Korrelationskoeffizient miteinander verglichen. Dabei dient wieder Pearsons und Spearmans Koeffizient als Maß der Ähnlichkeit zwischen den Werten. In Tabelle 4 ist die Korrelation der Gewichte zwischen Random Forest und dem jeweiligen Verfahren dargestellt.

Starke Korrelation zu Random Forest ist bei keinem der Verfahren erkennbar. Auf dem Ice-21 Datensatz sind SVM und Random Forest schwach korreliert. Änderung des C Parameters der SVM ergaben dabei Änderungen, die jedoch keinen eindeutigen Trend zeigten. In diesem Fall wurde C auf dem Standardwert belassen. Die Spearman Korrelation ist hier vermutlich eine bessere Abschätzung als die von Pearson, da auf dem geringen Stichprobenumfang von 21 beziehungsweise 10 Merkmalen keine Normalverteilung angenommen werden sollte.

8.3. Qualität der Random Forest Merkmalsgewichtung

Um eine Aussage über die Qualität der Gewichtung zu treffen, werden Merkmale anhand ihrer Gewichte selektiert und die Klassifikationsgenaugkeit gemessen. Zur Klassifikation wird Naive Bayes und Random Forest genutzt. Gemessen werden die Größen

Ice-21

	Pearson	p	Spearman	p
IG	0.020		0.142	
SVM	0.432	0.05	0.623	0.003
SAM	0.232		0.121	

MAGIC

	Pearson	p	Spearman	p
IG	-0.365		-0.438	
SVM	0.635	0.048	0.503	0.138
SAM	0.570		0.479	

Tabelle 4: Spearman- und Pearson-Korrelation der Merkmalsgewichte zwischen Random Forest und den anderen Verfahren. Der p-Wert ergibt sich als zweisetige Teststatistik mit vollständiger Unkorreliertheit als Nullhypothese.

Accuracy[11], Recall[12] und AUC[13] mittels 10-facher Kreuzvalidierung. Selektiert werden 20 Merkmale des Ice-475 Datensatzes. Die Merkmalsgewichte der SVM wurden auf dem selben Datensatz allerdings ohne fehlende Attributwerte berechnet. Der C Parameter wurde durch Parameteroptimierung mit dem Naive Bayes Klassifizierer auf 1.0 bestimmt. Der s_0 Parameter der SAM Gewichtung hatte keinen ausschlaggebenden Einfluss und wurde auf dem Standardwert von 0.1 gelassen. Der Δ-Parameter des PAM Verfahrens wurde so gewählt, dass 20 Merkmale selektiert wurden. In diesem Falls war $\Delta = 11.8$, da die Klassifikationsgenaugkeiten bei allen Verfahren auf den gleichen Daten gemessen wurde. Dafür wurde einmal eine zufällige Teilmenge von Ice-475 mit 10 000 Elementen gezogen. Die Ergebnisse der Klassifikation durch Naive Bayes sind in Tabelle 5 zu sehen.

Information Gain, Random Forest und mRMR liegen bei allen drei Kriterien weit vor den restlichen Verfahren. Information Gain hat bei Accuracy und Recall die besten Werte. Ein t-Test zeigt, dass die Werte auch signifikant voneinander abweichen. Der Versuch wurde mit den selben Einstellungen wiederholt. Diesmal wurde mit einem Random Forest aus 100 Bäumen klassifiziert. Die Ergebnisse sind in Tabelle 6 zu sehen.

Diesmal hat die Selektion durch SVM die besten höchsten Werte bei allen drei Messungen. Dahinter schließt sich die Random Forest Selektion mit sehr ähnlichen Zahlenwerten an. Der p-Wert für einen zweiseitigen t-Test zwischen SVM und Random Forest beträgt 0.0031. Der Unterschied ist demnach laut t-Test stark signifikant. Abschließend wurde die optimale Anzahl von selektierten Merkmalen für mRMR, In-

[11] Accuracy $\widehat{=}$ (True positives + True negatives)/Gesamtanzahl
[12] Recall $\widehat{=}$ True positives/(True positives + False negatives)
[13] *Area under Curve* der *Reciever Operating Characteristic*

	Accuracy	σ	**Recall**	σ	**AUC**	σ
IG	**0.7963**	0.0127	**0.6586**	0.0177	0.876	0.016
SAM	0.7545	0.0110	0.5702	0.0191	0.848	0.016
PAM	0.7277	0.0085	0.5166	0.0156	0.829	0.006
SVM	0.7179	0.0066	0.5132	0.0143	0.839	0.009
RF	0.7728	0.0093	0.6082	0.0200	0.877	0.008
mRMR	0.7833	0.0097	0.6370	0.0186	**0.882**	0.007
	0.7510	0.0118	0.5844	0.0191	0.849	0.015

Tabelle 5: Accuracy, Recall und AUC mit verschiedenen Merkmalen. Klassifiziert wurde mit Naive Bayes. Die Ergebnisse der letzten Zeile wurden ohne jegliche Merkmalselektion berechnet.

	Accuracy	σ	**Recall**	σ	**AUC**	σ
IG	0.8803	0.0128	0.8540	0.0169	0.945	0.008
SAM	0.8146	0.0074	0.7838	0.0184	0.883	0.008
PAM	0.8539	0.0074	0.8430	0.0148	0.928	0.006
SVM	**0.9276**	0.0105	**0.9106**	0.0138	**0.974**	0.002
RF	0.9126	0.0091	0.8934	0.0141	0.966	0.004
mRMR	0.8675	0.0115	0.8454	0.0158	0.933	0.007
	0.9373	0.0102	0.9138	0.0166	0.981	0.004

Tabelle 6: Accuracy, Recall und AUC mit verschiedenen Merkmalen. Klassifiziert mit einem Random Forest aus 100 Bäumen. In der letzten Zeile wurde keine Selektion durchgeführt.

formation Gain, Random Forest und SVM bestimmt. Auf dem kompletten Ice-475 Datensatz wurde Accuracy, Recall und AUC mittels 10-Facher Kreuzvalidierung und dem Naive Bayes-Klassifizierer gemessen. Für mRMR wurden 7 Merkmale bestimmt, für Information Gain 39, für SVM 9 und für Random Forest 25. Auf diesen Merkmalen wurde mit einem Random Forest nochmals der komplette Ice475-Datensatz klassifiziert und die Genauigkeit mit 10-facher Kreuzvalidierung bestimmt. Die Diagramme befinden sich in Anhang B.

Ice-475						
	Accuracy	σ	Recall	σ	AUC	σ
SVM (9)	0.8933	0.0050	0.8626	0.0091	0.9528	0.0030
RF (25)	**0.9378**	0.0029	**0.9218**	0.0039	**0.9811**	0.0010
IG (39)	0.9067	0.0038	0.8884	0.0062	0.9637	0.0020
mRMR (7)	0.8601	0.0035	0.8330	0.0044	0.9253	0.0032

Tabelle 7: Messung von Accuracy, Recall und AUC. Die Merkmalsanzahl wurde zuvor für das jeweilige Verfahren bestimmt.

Die Werte für Accuracy und Recall bei Random Forest liegen mit 93,78% und 92,18% deutlich über den Werten der anderen Selektionen. Ein höherer Wert für Accuracy und Recall bedeutet, dass die Daten nach Anwendung des Lernverfahrens weniger Rauschen enthalten und weniger Neutrinos verloren gehen. Laut t-Test weichen die Werte hoch signifikant voneinander ab. Der Nachteil der Random Forest-Selektion liegt in der vergleichsweise hohen Laufzeit (siehe Abbschnitt 6.5). Da die Gewichtung allerdings nur einmal pro Datensatz durchgeführt werden muss, halte ich den Aufwand jedoch für gerechtfertigt.

8.4. Selektierte Merkmale der IceCube-Daten

In folgender Tabelle sind die Top 20 der von Random Forest selektierten Merkmale der IceDaten zu sehen. Myonen atmosphärischen Ursprungs lassen sich von Myonen aus Neutrinointeraktionen durch ihre Energie unterscheiden. Die selektierten Merkmale sollten also vor allem mit der Helligkeit und Streuung eines Events zusammenhängen. Die NDirA/B/C-Parameter der verschiedenen Rekonstruktionsmethoden beschreiben die Anzahl der direkten Hits innerhalb eines Zeitfensters. Ein direkter Hit ist in diesem Fall definiert als ein Hit, welcher in der Nähe der berechneten Neutrinospur liegt. Das LDirE Attribut hängt mit der Länge der aufgezeichneten Myonespur zusammen. Ein energiereiches Neutrino führ zu einer Myonenspur die mehr direkte Hits erzeugt und eine längere Strecke umfasst.

Selektierte Merkmale der Ice-475 Daten:

MPEFitATWD LDirE
MPEFitMuEATWD LDirE
MPEFitMuEATWD NDirB
MPEFitMuE LDirE
MPEFitMuE NDirB
MPEFitPhotorecEnergyATWD LDirE
MPEFit LDirE
MPEFit NDirB
MPEFit NDirC
NStringAll
SPEFit8Bayesian NDirD
SPEFit8Bayesian NLate
SPEFit8Noisey LDirE
SPEFit8 LDirE
SPEFit8 NDirA
SPEFit8 NDirB
SPEFit8 NDirC
SPEFitSingle LDirE
SPEFitSingle NDirA
SPEFitSingle NDirB
SPEFitSingle NDirC

Tabelle 8: Von Random Forest selektierte Merkmale des Ice-475 Datensatzes.

9. Vergleiche der Abstimmungsverfahren

Im Folgenden soll das Verhalten der implementierten Abstimmungsverfahren genauer untersucht werden. Dazu wird zunächst die Parameterabhängigkeit der Abstimmung empirisch getestet. Die Merheitsabstimmung wird im Folgenden häufig durch MV, die Abstimmung nach Intrinsic Proximity durch IntPV,fehlergewichtete Abstimmung durch errV, die Abstimmung mit k-means Clustering durch CV und mit k-means++ durch CV++ abgekürzt.

9.1. Parameterstabilität der Abstimmungsverfahren

Zunächst wird die Anzahl der Entscheidungsbäume im Random Forest verändert. In jedem Schritt wird dabei die Klassifikationsgenaugkeit gemessen. Getestet wird mit einer Teilmenge des Ice-21-Datensatzes. Die Ergebnisse für die verschiedenen Abstimmungsverfahren sind in Abbildung 11 dargestellt.

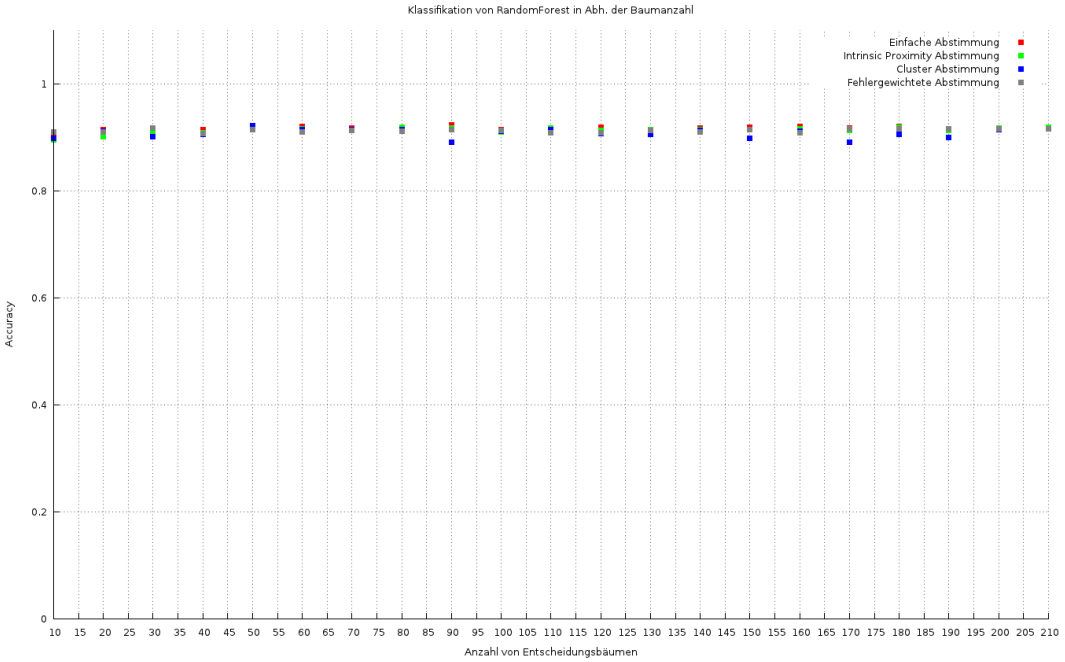

Abbildung 11: Klassifikationsgenaugkeit (Accuracy) in Abhängigkeit der Baumanzahl.

Die Klassifikationsgenaugkeit bleibt bei allen Verfahren stabil. Die Anzahl der verwendeten Bäume wird im Folgenden auf 100 belassen. Wie in Kapitel 5.4 gesehen, bestimmt die Größe der Nachbarschaft N_x, wie viele Punkte bei der Berechnung der Baumgewichte genutzt werden. Getestet wurde dies wie zuvor auf den Ice-21 Daten. Der Parameter N_x wurde von 5 bis 35 durchlaufen und die Genauigkeit mit 10-facher

Kreuzvalidierung gemessen. Die anderen Parameter wurden dabei auf Standardwerten belassen. Die Ergebnisse sind in Diagramm 12 zu sehen.

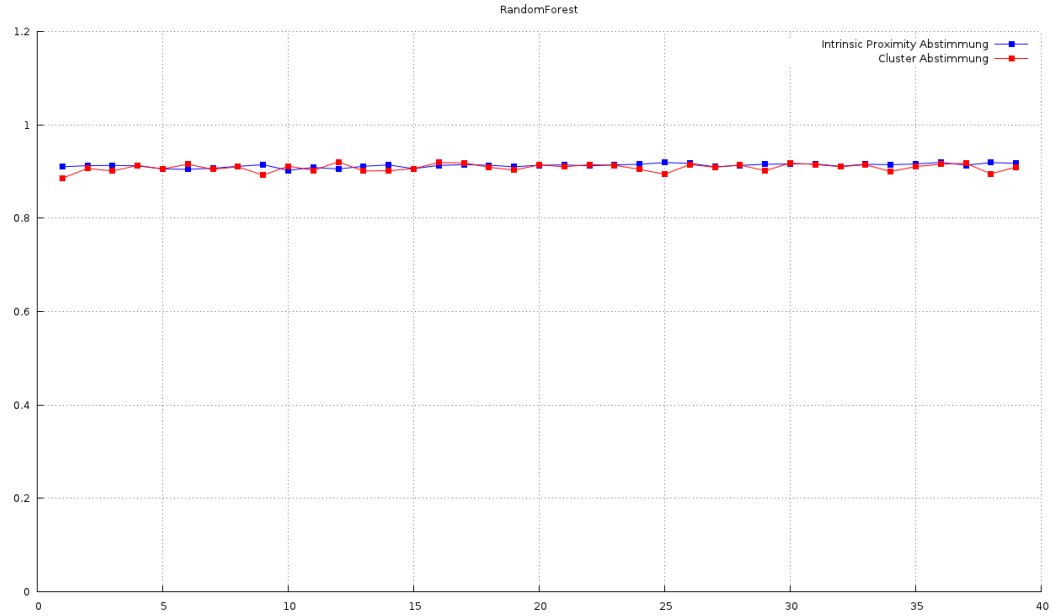

Abbildung 12: Klassifikationsgenaugkeit (Accuracy) in Abhängigkeit der Nachbarschaftsgröße

Dieser Parameter hat ebenfalls keinen starken Einfluss auf die Klassifizierung. Neben der Nachbarschaftsgröße kann auch die minimale Anzahl an Punkten, die ein Blatt bilden (minNum), Einfluss auf die Messung der Intrinsic Proximity haben. Auch dies wird wie oben mit 10-facher Kreuzvalidierung für verschiedene Parameterwerte bestimmt, wie in Diagramm 13 zu sehen ist.

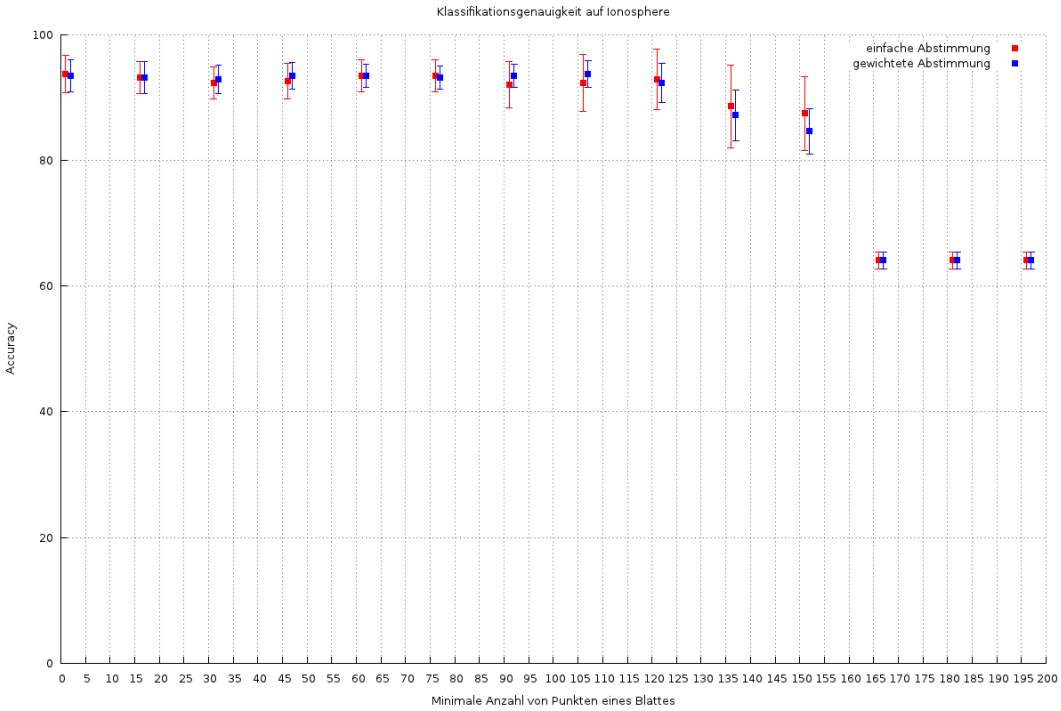

Abbildung 13: Klassifikationsgenaugkeit (Accuracy) in Abhängigkeit von der minimalen Anzahl von Punkten die ein Blatt bilden.

Sobald ein bestimmter Wert überschritten ist, findet keine Veränderung der Klassifizierung mehr statt. Ist der Wert von minNum größer als die Hälfte der Anzahl von Punkten im Datensatz, so bestehen die Entscheidungsbäume nur noch aus einer Wurzel mit zwei Blättern. Weitere Änderungen an dem Parameterwert verändern die Struktur des Baumes nicht mehr. Zuletzt wird noch die Anzahl der betrachteten Merkmale pro Blatt variiert. Dieser Parameter beeinflusst alle Abstimmungsmethoden gleichermaßen und wird nur für die Merheitsabstimmung überprüft. Wie in Diagramm 14 zu sehen, spielt aber auch dieser Parameter keine Rolle für die folgenden Tests.

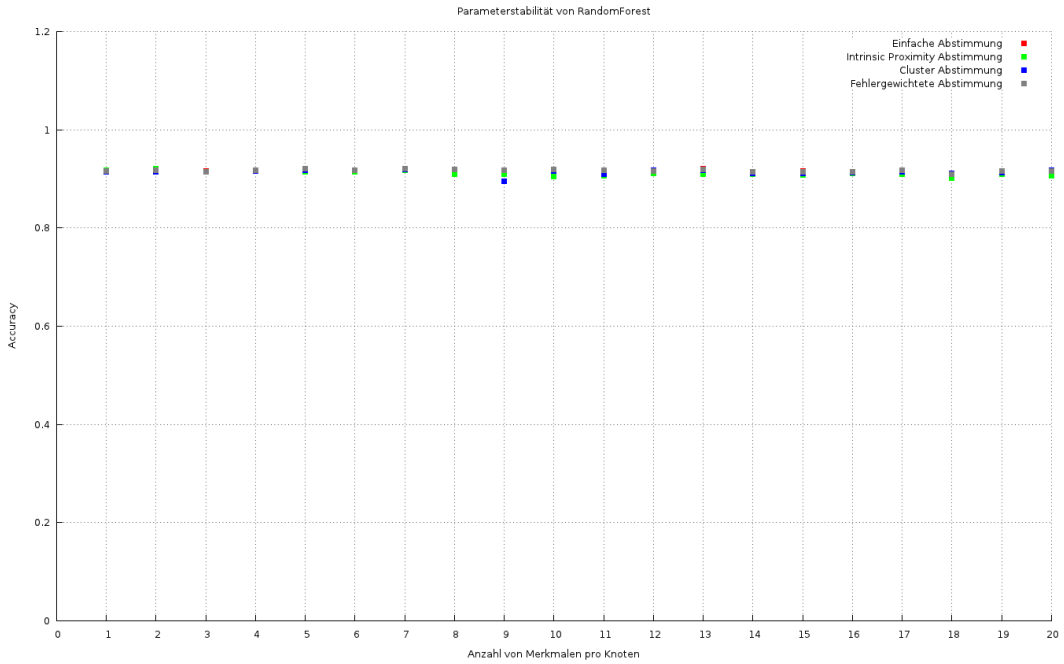

Abbildung 14: Klassifikationsgenaugkeit (Accuracy) in Abhängigkeit von der Anzahl der betrachteten Merkmale in einem Knoten des Random Trees.

Zusammengefasst ist also keiner der betrachteten Parameter kritisch für die Benutzung der verschiedenen Abstimmungsmethoden. In den folgenden Tests werden die Standardwerte benutzt. Die Größe der Nachbarschaft beträgt 30, die Anzahl der Merkmale pro Blatt $\log(d)$ und minNum 1.

9.2. Klassifikationsgenaugkeit mit gewichteter Abstimmung

Die Klassifikationsgenaugkeit wird anhand der bereits beschriebenen Datensätzen Ice-21, Sonar und Ionosphere mit 10-facher Kreuzvalidierung getestet. Der Random Forest besteht wieder aus 100 Bäumen, alle anderen Parameter bleiben ebenfalls unverändert. Die Ergebnisse sind in Tabelle 9 dargestellt.

Auf dem Sonar-Datensatz sind die deutlichsten Unterschiede in den Zahlenwerten erkennbar. Die Werte für Recall und AUC unterscheiden sich nur geringfügig. Mit einer Accuracy von 85,57% bei IntPV und 82,69% bei der Merheitsabstimmung scheint die Abstimmung mittels Intrinsic Proximity besser zu sein, allerdings ist das Ergebnis aufgrund der großen Standardabweichung, zumindest laut zweisetigem t-Test, nicht signifikant. Auf den anderen Daten sind die Unterschiede noch geringer. Hier ist ebenfalls keine Signifikanz zwischen den Zahlenwerten gegeben. Die Initialisierung von Zentroiden durch das k-means++-Verfahren macht auf diesen Daten keinen Unterschied gegenüber der normalen k-means-Variante.

Ice-21

	Accuracy	σ	Recall	σ	AUC	σ
MV	0.9284	0.0078	0.9474	0.0092	0.9763	0.0036
IntPV	0.9239	0.0071	0.9414	0.0085	0.9631	0.0039
CV	0.9275	0.0077	0.9473	0.0082	0.9762	0.0036
CV++	0.9289	0.0053	0.9485	0.0110	0.9762	0.0029
errV	0.9293	0.0053	0.9500	0.0103	0.9765	0.0029

Sonar

	Accuracy	σ	Recall	σ	AUC	σ
MV	0.8269	0.0681	0.8917	0.1198	0.9471	0.0352
IntPV	0.8557	0.0840	0.9098	0.0857	0.9456	0.0501
CV	0.7974	0.0920	0.8462	0.1143	0.8731	0.0772
CV++	0.8279	0.0678	0.8189	0.1361	0.9157	0.0558
errV	0.8474	0.0944	0.9091	0.0958	0.9536	0.0384

Ionosphere

	Accuracy	σ	Recall	σ	AUC	σ
MV	0.9372	0.0422	0.8712	0.1107	0.9814	0.0172
IntPV	0.9401	0.0343	0.8635	0.1031	0.9841	0.0190
CV	0.9401	0.0368	0.8712	0.1107	0.9747	0.0247
CV++	0.9401	0.0368	0.8712	0.1107	0.9751	0.0245
errV	0.9372	0.0422	0.8712	0.1107	0.9826	0.0165

BCD

	Accuracy	σ	Recall	σ	AUC	σ
MV	0.9684	0.0272	0.9776	0.0256	0.9912	0.0168
IntPV	0.9719	0.0222	0.9804	0.0188	0.9916	0.0154
CV	0.9684	0.0296	0.9832	0.0269	0.9911	0.0168
CV++	0.9684	0.0296	0.9832	0.0269	0.9911	0.0168
errV	0.9684	0.0272	0.9804	0.0264	0.9914	0.0168

Tabelle 9: Messung von Accuracy, Recall, AUC und Standardabweichungen auf verschiedenen Datensätzen.

Nach Tests durch Tsymbal [40] und Ŝikonja [36] wird durch den Einsatz von IntPV die Klassifikationsgenaugkeit auf manchen Datensätzen signifikant besser. Tsymbal testete die Accuracy des Random Forest durch 30-fache Kreuzvalidierung, mit jeweils 70 % Trainings- und 30% Testdaten. Dort wurden auch Daten mit nominalen Attributen getestet. Auf den hier getesteten Daten wurde durch andere Kreuzvalidierungen keine Erhöhung der Signifikanz festgestellt. Um eine Vergleichbarkeit mit den IceCube-Daten sicherzustellen, wurden hier allerdings nur Daten mit numerischen Merkmalen getestet.

10. Fazit und Ausblick

Die Analyse der IceCube-Daten birgt großes Potential für viele Bereiche der Teilchenphysik und Astronomie. Die Daten, die die Sensoren am Südpol sammeln, bestehen zum größten Teil aus Rauschen. Über viele verschiedene Filterstufen wird versucht, das Rauschen zu entfernen und dabei möglichst viel vom eigentlichen Signal zu behalten. Gerade Neutrinos mit hohen Energien treten nur selten auf. Zugehörige Daten dürfen deshalb nicht durch die Filter verworfen werden. Viele verschiedene Methoden generieren aus den Rohdaten des Detektors Merkmale, die zu physikalischen Parametern wie Richtung, Geschwindigkeit oder Energie eines Teilchen korrespondieren sollen. Programme wie CORSIKA werden genutzt um das Verhalten des Detektors zu simulieren. Maschinelle Lernverfahren werden auf die merkmalsbehafteten Daten angewandt, um das Rauschen der IceCube-Daten weiter zu reduzieren. Hier wurde das Random Forest-Verfahren auf simulierte IceCube-Daten angewandt. Es wurde eine Erweiterung von Random Forest implementiert, die von Ŝikonja und Tsymbal vorgeschlagen wurde. Nach [36] und [40] verbessert sich die Klassifikation durch Random Forest auf vielen Datensätzen. Für die in dieser Arbeit getesteten Daten konnte das jedoch nicht bestätigt werden. Die Unterschiede der Klassifikationsgenauigkeiten waren nicht signifikant. Gleiches gilt für die zweite Abstimmungsmethode, die auf k-means-Clustering beruht. Auch diese Methode wurde implementiert und empirisch getestet. Die Laufzeit der beiden erweiterten Abstimmungsmethoden ist erheblich höher als die der normalen Abstimmung. Eine Benutzung dieser lohnt sich demnach speziell auf großen Datenmengen nicht. Außerdem wurde eine Merkmalsgewichtungsmethode für Random Forest getestet und mit anderen Verfahren empirisch verglichen. Alle Implementierungen wurden in der WEKA-Umgebung entwickelt und per aktualisierter WEKA-Extension in das Analyseframework RapidMiner eingebunden. Dadurch können jetzt auch alle Ensemblemethoden aus WEKA in RapidMiner auf Multi-Core-Prozessoren genutzt werden. Durch eine Selektion von Merkmalen auf den simulierten IceCube-Daten konnte die Klassifikationgenauigkeit deutlich erhöht werden. Hier könnten weitere Analysen folgen um festzustellen, ob sich durch die Methode auch reale Daten aus dem Detektor besser klassifizieren lassen. Wie in Abschnitt 5.3 gezeigt, werden Attribute im Entscheidungsbaum nach dem Information Gain ausgewählt. Andere Kriterien, die besser für numerische Merkmale geeignet sind, könnten in zukünftigen Analysen getestet werden.

A. Tabellen

Tabellenverzeichnis

Name in Filter	**Requested BW** (GB per day)	**Actual BW used** (GB/day)	**Rate of events** (Hz)
MuonFilter_10	13.1	15.6	36.3
CascadeFilter_10	6	8.9	27.3
EHEFilter_10	4.3	1.7	1.7
IceTopSTA3_10	1.7	2.9	8.2
IceTopSTA3_InIceSMT_10	2.3	2	3.6
IceTopSTA8_10	0.4	0.8	1.6
IceTopSTA8_InIceSMT_10	0.7	0.2	0.6
InIceSMT_IceTopCoincidence_10	0.4	0.4	1.1
SlowParticle_10	1	0.7	0.9
GalacticCenter_10	22	19.4	53.4
MoonFilter_10	2.2(avg/28 day)	2.2/day avg	≈7.0
SunFilter_10	8.0 (max rate)	8.0 (max rate)	23 Hz max rate)
LowUpFilter_10	2.6	3.8	18.5
LowEnergyContainedFilter_10	1.1	3	12.1
DeepCoreFilter_10	6	6.2	18.2
FilterMinBias_10	–	1.3	3.8
PhysicsMinBiasTrigger_10	–	0.2	1.1
I3DST_10	4.8	4.5	–
Non-Filter TDRSS Requests	3.5	3.5	–
Total	72 GB/day 80 GB/day MAX	75.4 GB/day 83.4 GB/day	157 Hz

Tabelle 10: Bandbreiten der im Jahr 2010 benutzen Filter (kopiert aus [3]).

B. Abbildungen

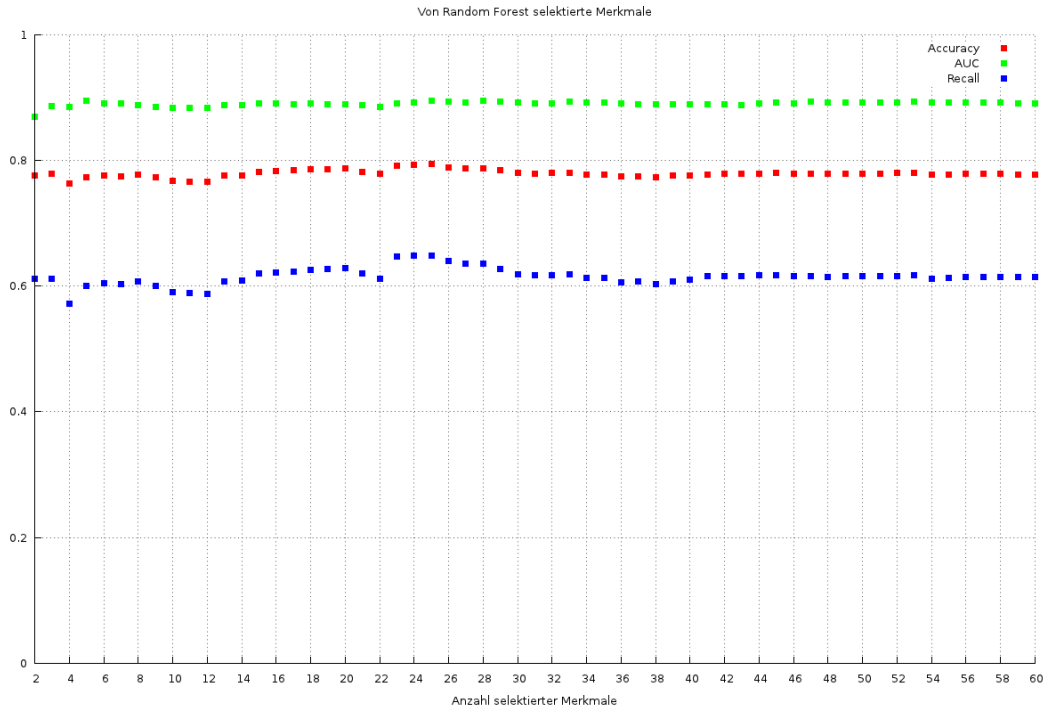

Abbildung 15: Klassifikationsgenaugkeit (Accuracy, Recall und AUC) in Abhängigkeit der Merkmalsanzahl. Selektiert durch Random Forest und klassifiziert durch NaiveBayes.

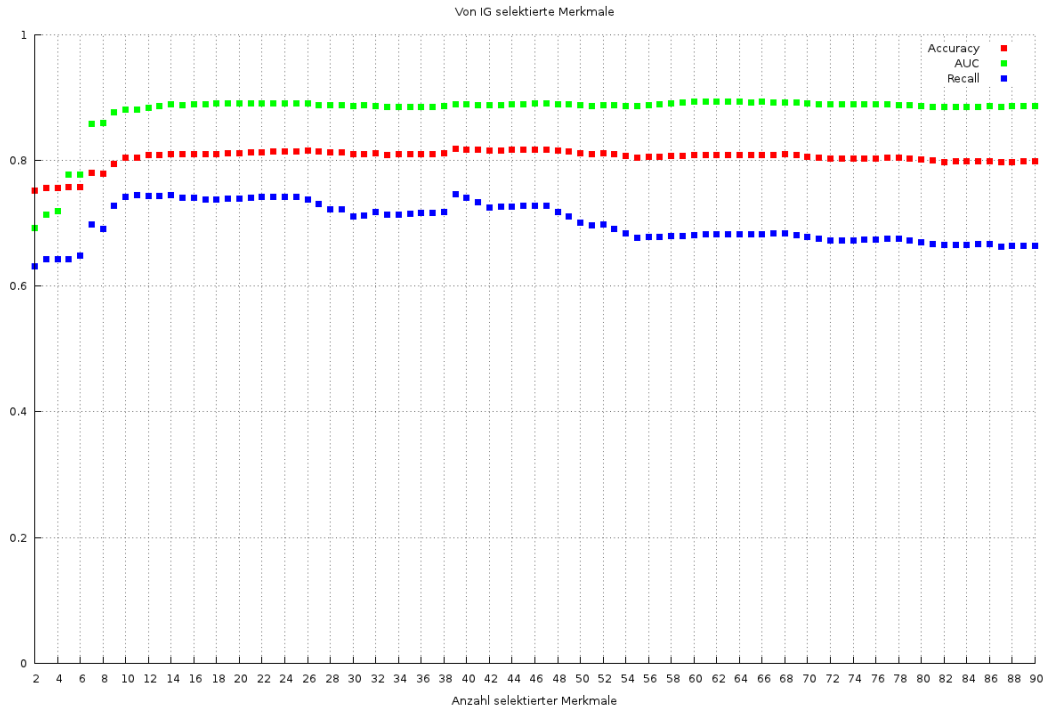

Abbildung 16: Klassifikationsgenaugkeit (Accuracy, Recall und AUC) in Abhängigkeit der Merkmalsanzahl. Selektiert durch Information Gain Ratio und klassifiziert durch NaiveBayes.

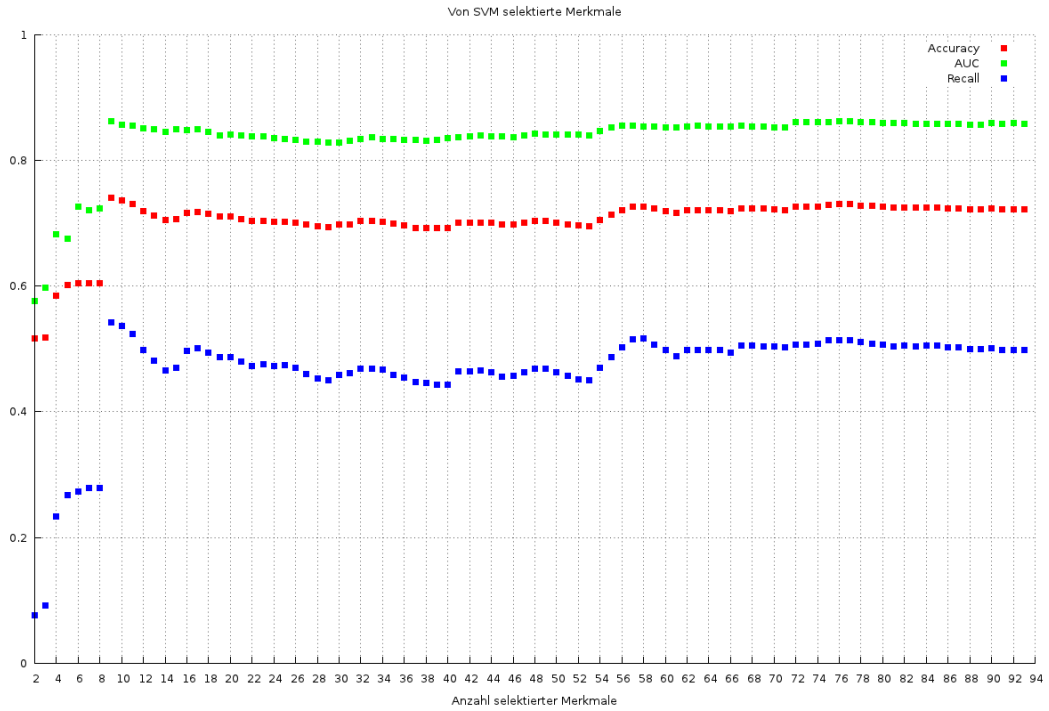

Abbildung 17: Klassifikationsgenaugkeit (Accuracy, Recall und AUC) in Abhängigkeit der Merkmalsanzahl. Selektiert durch SVM und klassifiziert durch NaiveBayes.

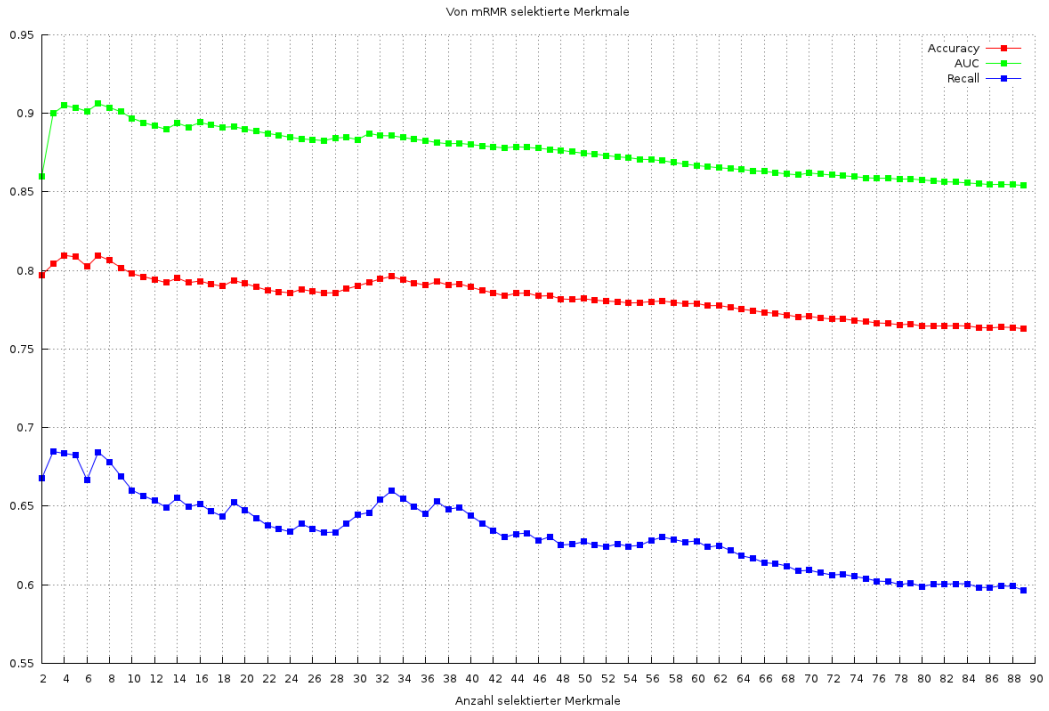

Abbildung 18: Klassifikationsgenaugkeit (Accuracy, Recall und AUC) in Abhängigkeit der Merkmalsanzahl. Selektiert durch mRMR und klassifiziert durch NaiveBayes.

Abbildungsverzeichnis

C. Abkürzungsverzeichniss

ICL	IceCube Lab. Teil der Amundsen-Scott Polarforschungsstation
mRMR	minimum Redundancy, maximum Relevance
IG	Information Gain Ratio
RF	Random Forest
SVM	Suppot Vector Machine
AUC	Area under Curve
MC	Monte-Carlo
PAM	shrunken centroids classification. Methode zur Merkmalsgewichtung
SAM	Significance Analysis of Microarrays. Merkmalsgewichtung
HLC	Hard Local Coincidence
SLC	Soft Local Coincidence
DOM	Digital Optical Module
IntPV	Abstimmung nach Intrinsic Proximity
CV	Abstimmung nach k-means Clustering
CV++	Abstimmung nach k-means++ Clustering
errV	Abstimmung mit Fehlergewichtung
MV	Einfache Merheitsabstimmung
TDIDT	Top-Down Induction of Decision Trees
C4.5, ID3	Verfahren zum Erstellen eines Entscheidungsbaums
TDIDT	Top-Down Induction of Decision Trees
PMT	Photomultiplier Tube. Der Lichtsensor im DOM
WIMP	Weakly Interacting Massive Particles. Hypothetisches Teilchen

Anmerkungen

Die Daten wurden unter anderem mit der SciPy Python Library (`http://www.scipy.org/`) und eigens dafür angefertigten Skripten ausgewertet. Geschrieben und gesetzt wurde der Text in Kile, dem KDE Latex-Editor (`http://kile.sourceforge.net/`). Programiert wurde mithile der Eclipse IDE (`http://www.eclipse.org/`). Tabellen wurden durch LibreOffice erstellt (`http://www.libreoffice.org/`). Das implementierte Programm kann über `https://sourceforge.net/projects/m-core-wekaext/` bezogen werden.

Danken möchte ich allen, die mich beim Schreiben moralisch und tatkräftig unterstützt haben. Insbesondere den wunderbaren Korrekturlesern Debby, Linda und Alexey. Für die kompetente Beantwortung all meiner Fragen, danke ich Marco Stolpe. Dank gilt auch dem gesamten Lehrstuhl 8, nicht zuletzt wegen des leckeren Frühstücks. Dank geht an Frau Prof. Dr. Morik die mich auf diesen interessanten Themenbereich aufmerksam gemacht hat. Für die Versorgung mit Informationen und Daten rund um IceCube und der hervorragenden Beantwortung aller physikalischen Fragen, danke ich Tim Ruhe und auch den anderen Mitarbeitern des E5b Lehrstuhls.

Literatur

[1] Rapidminer. URL http://rapid-i.com. Datamining Application.

[2] New idaho national lab collaboration tackles nuclear fuel recycling science. URL http://www.anl.gov/Media_Center/News/2009/news090925.html. 2009, Presseerklärung des Argonne National Labratory.

[3] Wiki seite zu den online filtern in 2010. URL http://wiki.icecube.wisc.edu/index.php/TFT_2010_Season_Planning. TFT Filter Planing 2010.

[4] Corsika. URL http://www-ik.fzk.de/corsika/. Air Shower Simulation Program.

[5] Official website of the icecube neutrino observatory. URL http://icecube.wisc.edu/.

[6] Katrin. URL http://www-ik.fzk.de/~katrin/index.html. Karlsruhe Tritium Neutrino Experiment zur genaueren Bestimmung von Neutrino Massen.

[7] Linefit online documentation. URL http://software.icecube.wisc.edu/ICEREC-V03-03-03/doxygen/linefit/index.html. LineFit Software Module.

[8] Muonfilter for 2011 description in the icecube wiki. URL http://wiki.icecube.wisc.edu/index.php/2011_Online_Muon_Filter_Proposal. MuonFilter 2011 Proposal.

[9] Wiki seite zur simulation. URL http://wiki.icecube.wisc.edu/index.php/Simulation_Documentation_Wiki. Simulation Documentation Wiki.

[10] Weka. URL http://www.cs.waikato.ac.nz/ml/weka/. Java Tool for Machin Learning.

[11] Matthew Wiener Andy Liaw. Classification and regression by randomforest. 2002.

[12] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[13] J. Bahcall. Solving the mystery of the missing neutrinos. *Arxiv preprint physics/0406040*, 2004.

[14] C. Beierle and G. Kern-Isberner. *Methoden wissensbasierter Systeme: Grundlagen-Algorithmen-Anwendungen*. Vieweg+ Teubner, 2008. ISBN 9783834895172. URL http://www.ub.tu-dortmund.de/katalog/titel/1232123.

[15] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[16] Leo Breiman and E. Schapire. Random forests. pages 5–32, 2001.

[17] T. Bylander. Estimating generalization error on two-class datasets using out-of-bag estimates. *Machine Learning*, 48(1):287–297, 2002.

[18] C. Chih-Chung and L. Chih-Jen. Libsvm: a library for support vector machines, 2001. URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[19] D. L. Donoho. High-dimensional data analysis: the curses and blessings of dimensionality. In *American Mathematical Society Conf. Math Challenges of the 21st Century*. 2000.

[20] J. Ahrens et Al. Muon track reconstruction and data selection techniques in amanda. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 524(1-3):169 – 194, 2004.

[21] U.M. Fayyad and K.B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine learning*, 8(1):87–102, 1992.

[22] Spencer R. Klein for the IceCube Collaboration. Icecube: A cubic kilometer radiation detector. *Nuclear Science, IEEE Transactions*, 56(3):1141 – 1147, 2009.

[23] A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

[24] F. Halzen and J.P. Rodrigues. Detection of supernova explosions with icecube. *Classical and Quantum Gravity*, 27:194003, 2010.

[25] Francis Halzen. Icecube science. *J.Phys.Conf.Ser.*, 2009. URL `http://arxiv.org/abs/0901.4722`.

[26] Francis Halzen and Spencer R. Klein. Icecube: An instrument for neutrino astronomy. *Rev.Sci.Instrum.*, 81:081101, 2010. URL `http://arxiv.org/abs/1007.1247`.

[27] Trevor. Hastie, Robert. Tibshirani, and JH (Jerome H.) Friedman. *The elements of statistical learning*. Springer, 2009.

[28] A. Kalousis, J. Prados, and M. Hilario. Stability of feature selection algorithms. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[29] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, pages 1137–1145. Citeseer, 1995.

[30] Prof. Dr. Hermann Kolanoski. Einfuhrung in die astroteilchenphysik, 2009. URL `www-zeuthen.desy.de/~kolanosk/astro0910/skripte/astro.pdf`. Skript zur Vorlesung gehalten and der Humboldt-Universität Berlin.

[31] Tom M. Mitchell. *Machine Learning*. Draft edition.

[32] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, pages 1226–1238, 2005.

[33] J. Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods-Support Vector Learning*, 208:98–112, 1999.

[34] F. Reines and C. Cowan. The reines-cowan experiments: Detecting the poltergeist. *Los Alamos Science*, (25):4–6, 1997.

[35] Elisa Bernardini Robert Lauer. Suche nach neutrino punktquellen bei energien im bereich von tev und eev, 2009. Praesentation auf der DPG Früjahrstagung 2009.

[36] Marko Robnik-Sikonja. Improving random forests. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *ECML*, volume 3201 of *Lecture Notes in Computer Science*, pages 359–370. Springer, 2004. ISBN 3-540-23105-6.

[37] Tim Ruhe, Katharina Morik, and Benjamin Schowe. Data mining on ice. In *Procs. of the Workshop on Astrostatistics and Data Mining in Large Astronomical Databases*, 2011.

[38] Benjamin Schowe et al. Feature selection extension for rapidminer. URL `http://sourceforge.net/projects/rm-featselext/`.

[39] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567, 2002.

[40] Alexey Tsymbal, Mykola Pechenizkiy, and Padraig Cunningham. Dynamic integration with random forests. In Tobias Scheffer and Myra Spiliopoulou, editors, *ECML*, volume 4212 of *Lecture Notes in Computer Science*, pages 801–808. Springer, 2006. URL `http://dblp.uni-trier.de/db/conf/ecml/ecml2006.html`.

[41] V.G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences*, 98(9):5116, 2001.

[42] Bernhard Voigt. *Sensitivity of the IceCube Detector for Ultra-High Energy Electron-Neutrino Events*. PhD thesis, Humboldt-Universität zu Berlin, 2008.