Studiengang Informatik

Diplomarbeit

Auswirkungen von Sensoreneigenschaften auf die Angriffserkennung mittels Sensorfusion

Malte Humann

Bremen, den 24. März 2014

Erstgutachter: Dr. Karsten Sohr
Zweitgutachter: Prof. Dr. Michael Lawo
Betreuer: Carsten Elfers

Humann, Malte
humi@informatik.uni-bremen.de
Auswirkungen von Sensoreneigenschaften auf die Angriffserkennung mittels Sensorfusion
Diplomarbeit, Studiengang Informatik
Universität Bremen, März 2014

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbständig ohne die Hilfe Dritter verfasst habe. Ich habe keine anderen als die explizit angegebenen Quellen und Hilfsmittel verwendet. Sämtliche wissentlich verwendeten Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden stets als solche gekennzeichnet.

Bremen, den 24. März 2014

Malte Humann

Danksagung

Mein Dank gilt Dr. Karsten Sohr, der nicht nur die Beurteilung der Arbeit als Erstgutachter übernommen hat, sondern auch den Fortgang der Arbeit durch seine persönliche Betreuung immer wieder vorangetrieben hat.

Weiterhin bedanke ich mich bei Prof. Dr. Michael Lawo dafür, dass er sich als Zweitgutachter zur Beurteilung meiner Arbeit zur Verfügung gestellt hat.

Besonderer Dank gilt Carsten Elfers, der es trotz anderer Verpflichtungen immer geschaft hat, sich die Zeit zu nehmen, den Fortschritt der Arbeit durch wertvolle Ratschläge und Hinweise, sowie ausführliche inhaltliche Diskussionen maßgeblich zu unterstützen.

Darüber hinaus danke ich Christoph Greulich und Arne Humann, die weite Teile der Arbeit auf Grammatik- und Rechtschreibfehler geprüft haben.

Inhaltsverzeichnis

1 Einleitung

Im Zeitalter des Internets eröffnen sich, durch die steigende Vernetzung und die damit verbundenen Anwendungsmöglichkeiten, mehr und mehr Angriffspunkte auf Computersysteme [PP07]. Auch wenn es früher für Systemadministratoren möglich war, Angriffe durch das manuelle Überwachen von Benutzeraktivitäten auf einem System zu erkennen, so ist dieses Vorgehen bei der heutigen Menge an anfallenden Daten nicht mehr praktikabel [KV02]. Um dennoch Angriffe erkennen zu können, werden sogenannte Intrusion Detection Systeme (IDS) eingesetzt, die Computernetze und auch Computer selbst auf entsprechende Aktivitäten hin überwachen und verdächtige Ereignisse zeitnah melden [Con02]. Solange ein IDS nicht unfehlbar ist, erkennt es unter Umständen bestimmte Angriffsarten besser als andere oder stuft gewisse normale Aktivitäten als bedrohlich ein. Unter der Annahme, dass unterschiedliche IDS nicht die exakt selben Fehler machen, bietet es sich an, eine Gruppe aus IDS einzusetzen und die Entscheidungen der einzelnen Systeme zu einer gemeinsamen Entscheidung zu kombinieren [TB09]. Um einzelne IDS zu testen und zu vergleichen, wurde oft auf den Datensatz des KDD'99 Cup[1] zurückgegriffen, obwohl er inzwischen nicht mehr zeitgemäß ist und auch anderweitig kritisiert wurde [CGM+09]. Da die Fusionsmethoden, die verwendet werden, um die Entscheidungen mehrerer IDS zu kombinieren, aber primär auf eben diesen Entscheidungen aufbauen, kann ein Vergleich auch auf Grundlage solcher IDS-Ausgaben durchgeführt werden. Das hat den Vorteil, dass keine Testdaten für die IDS selbst benötigt werden, die sonst beispielsweise durch das Aufzeichnen von simulierten Angriffen in einem Testnetz gesammelt werden müssten. Stattdessen können direkt die abstrakten Entscheidungen der IDS verwendet werden, was das Erstellen eines Testdatensatzes erheblich vereinfacht.

Im Rahmen dieser Arbeit wird eine Simulationsumgebung entwickelt, die entsprechend ihrer Konfiguration in der Lage ist, beliebige solcher Datensätze zu generieren. Diese Daten werden direkt an die zu untersuchende Fusionsmethode weitergeleitet, und die Ergebnisse werden zur späteren Auswertung in einer Datenbank gespeichert. Mit Hilfe dieser Daten wird anschließend untersucht, wie sich die Eigenschaften, die Anzahl und die Zusammenstellung der Sensoren auf

[1] Online verfügbar unter `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`. [Stand: 18.03.2014]

die unterschiedlichen Fusionsmethoden auswirken. Der Vorteil einer Simulation gegenüber einer rein analytischen Betrachtung von Fusionsmethoden liegt dabei darin, dass so auch Verfahren untersucht werden können, die sich nur sehr schwer mathematisch beschreiben lassen, weil sie beispielsweise komplexe Algorithmen zum Lernen von Trainingsdaten anwenden. Die Untersuchung soll dabei helfen herauszufinden, welche Fusionsmethode für ein vorgegebenes Szenario am besten geeignet ist bzw. wie mit den zur Verfügung stehenden Mitteln ein möglichst effektives IDS-Ensemble aufgebaut werden kann. Beispielsweise könnte es in einem Fall besser sein, weniger, dafür leistungsfähigere IDS einzusetzen, während in einem anderen Fall der Vorteil darin liegen könnte, möglichst viele IDS an unterschiedlichen Stellen in einem Netz zu platzieren, um so die Varianz zu erhöhen. In dieser Arbeit wird primär evaluiert, wie sich die Größe eines IDS-Ensembles auf die Erkennungsraten verschiedener Fusionsmethoden auswirkt. Weiterhin wird untersucht, wie die Fusionsmethoden auf den Ausfall eines der IDS reagieren und ob es Vorteile bringt, IDS einzusetzen, die sehr stark auf das Erkennen nur einer bestimmten Angriffsart spezialisiert sind. Für Fusionsmethoden, die auf eine Trainingsphase angewiesen sind, werden zusätzlich die Auswirkungen des Über- und Unterschätzen der IDS-Erkennungsraten im Training analysiert.

Als Einführung in die Thematik werden in Kapitel 2 zunächst die unterschiedlichen Arten von IDS vorgestellt. Anschließend wird ein Überblick über Fusionsmethoden im Bereich der Angriffserkennung gegeben und abschließend werden mögliche Bewertungsmetriken für den Vergleich von IDS und auch den Ergebnissen der IDS-Ensembles eingeführt. Kapitel 3 widmet sich der hier entwickelten Simulationsumgebung. Nachdem im ersten Teil von Kapitel 4 die Simulationsumgebung validiert worden ist, wird im zweiten Teil untersucht, wie sich die Sensoren auf die Fusionsmethoden auswirken. In Kapitel 5 werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick auf mögliche Erweiterungen der Simulationsumgebung sowie Anregungen für weitere Untersuchungen gegeben.

2 Sensorfusion in der IDS Domäne

Hall und Llinas [HL97] unterscheiden drei Ebenen, auf denen Sensordaten[1] kombiniert werden können. Zunächst ist es möglich, die Rohdaten der Sensoren direkt zusammenzuführen, sofern die Daten es zulassen. Sensordatenfusion auf dieser Ebene wird als *data level fusion* bezeichnet. Ist dies nicht möglich, müssen die Daten aufbereitet werden, was zu *feature level* bzw. *decision level fusion* führt. Für die *feature level fusion* werden aus den Rohdaten Merkmale extrahiert, welche dann mittels Fusion zu einem neuen Merkmalsvektor kombiniert und anschließend weiterverarbeitet werden können. Wenn die Sensoren bereits selbst eine Entscheidung auf Grundlage der Daten getroffen haben und auf Basis dieser einzelnen Entscheidungen eine gemeinsame Entscheidung gebildet werden soll, handelt es sich um *decision level fusion*.

Für die Sensorfusion auf der höchsten Ebene gibt es in der Angriffserkennung zwei große Anwendungsgebiete. Zum einen die Kombination von Klassifikatoren und zum anderen die Alarmkorrelation [CGM+09]. Das Ziel der Alarmkorrelation ist es, einzelne Meldungen zu einem Gesamtbild zusammenzufassen, um so die Anzahl der Ereignisse, die vom Sicherheitspersonal überprüft werden müssen, zu reduzieren [VVKK04]. Die Kombination von Klassifikatoren hingegen befasst sich damit, die Ergebnisse mehrerer IDS[2] auszuwerten und auf eine einzige Entscheidung zusammenzuführen.

Diese Arbeit beschäftigt sich in Bezug auf Sensorfusion mit der Kombination von Klassifikatoren bzw. IDS. Dazu werden in Abschnitt 2.1 zunächst die unterschiedlichen Arten von IDS mit ihren Vor- und Nachteilen vorgestellt. Anschließend wird in Abschnitt 2.2 auf die hier betrachteten Fusionsmethoden eingegangen und abschließend wird in Abschnitt 2.3 beschrieben, wie die Ergebnisse der Sensorfusion untereinander und auch mit den einzelnen IDS selbst verglichen werden können.

[1] In diesem Zusammenhang wird ein IDS selbst auch als Sensor verstanden.

[2] Ein IDS ist in gewisser Weise ein Klassifikator, da es versucht die beobachteten Ereignisse einer Klasse (Angriff, Nicht-Angriff oder auch unterschiedliche Arten von Angriffen) zuzuordnen.

2.1 Sensortypen und Eigenschaften

Um einen Überblick über die verschiedenen Arten von IDS und ihren unterschiedlichen Eigenschaften zu geben, wird primär auf die entsprechenden Übersichten von Kizza [Kiz13, Kapitel 13], Stallings und Brown [SB12, Kapitel 8], Axelsson [Axe00], Sundaram [Sun96] und eine im Auftrag des Bundesamt für Sicherheit in der Informationstechnik (BSI) durchgeführte Studie zu diesem Thema [Con02] zurückgegriffen.

Demnach hat ein IDS die Aufgabe, Computernetze oder auch Computer selbst auf Angriffe hin zu überwachen und entsprechende Aktivitäten zu melden. Ein Angriff kann in diesem Fall unterschiedliche Formen haben und reicht von dem Versuch, von außen illegaler Weise Zugriff auf das Zielsystem zu erlangen, über *Denial of Service* (DoS) Attacken bis hin zu Insidern, die ihre Zugriffsrechte missbrauchen [MHL94]. Die Grundidee, auf der IDS aufbauen, ist, dass sich das Vorgehen eines Angreifers in irgendeiner Form von dem Verhalten eines normalen Nutzers unterscheidet und so erkannt werden kann [MHL94]. Dazu gibt es zwei unterschiedliche Herangehensweisen. Eine Möglichkeit ist es, zu versuchen, über vorab definierte Signaturen Angriffe zu erkennen, die auf die entsprechenden Muster passen. Die zweite Methode, die zur Anwendung kommt, ist die Anomalieerkennung, bei der festgelegt wird, was normale Aktivitäten sind, um davon abweichende Handlungen als Angriffe einstufen zu können. Zusätzlich können IDS noch anhand ihres Einsatzgebietes unterschieden werden. IDS, die ein Netz überwachen, werden als netzbasierte IDS (NIDS) bezeichnet, entsprechend werden IDS, die Computer direkt beobachten, hostbasierte IDS (HIDS) genannt.

2.1.1 Anomalieerkennung

Für die Anomalieerkennung wird davon ausgegangen, dass sich Angriffe durch anormales Verhalten erkennen lassen. Entsprechend ist es nötig, dass dem IDS bekannt ist, was das erwartete Normalverhalten ist. Dafür eignen sich zum einen allgemeine Grenzwerte, die die zulässige Häufigkeit bestimmter Ereignisse festlegen, zum anderen Profile, die vorgeben, wie sich bestimmte Anwender, Anwendergruppen oder auch Programme und System-Ressourcen verhalten bzw. genutzt werden. Die Schwierigkeit besteht im Festlegen solcher Grenzwerte und Profile, da davon ausgegangen werden kann, dass es zu Überschneidungen zwischen Normalverhalten und potentiellen Angriffen kommen kann. Solche Überschneidungen können dazu führen, dass legitime, aber unübliche Aktionen als Bedrohung gewertet werden oder, was gefährlicher ist, dass böswillige Aktivitäten fälschlicherweise als normal eingestuft werden. Obwohl die Anomalieer-

kennung das Problem, dass die Fehlalarmrate sehr hoch sein kann, mit sich bringt, hat sie auch ihre Vorteile: Es ist beispielsweise nicht notwendig, dass Sicherheitslücken vorab bekannt sind, da das IDS mit dem normalen Verhalten trainiert wird, was gleichzeitig dazu führt, dass auch bisher unbekannte Angriffe erkannt werden können.

2.1.2 Signaturerkennung

Die Grundlage für die Signaturerkennung bildet die Annahme, dass jeder Angriff einem bestimmten Muster folgt, also eine Art Signatur hinterlässt, anhand der er und auch leichte Abwandlungen erkannt werden können. Die Signaturen können dabei unterschiedliche Formen haben und von einfachem *Pattern Matching* in Daten bis hin zu Verhaltensmustern (z.B. die Anzahl an Login-Fehlversuchen innerhalb eines bestimmten Zeitraumes) reichen. Das Problem dabei ist, dass die Signaturen vorab bekannt sein müssen und es somit nicht möglich ist, unbekannte Angriffe zu erkennen, für die noch keine Regeln erstellt wurden. Außerdem muss beim Erstellen der Signaturen darauf geachtet werden, dass sie zwar auf möglichst alle Variationen eines Angriffs passen, aber gleichzeitig nicht zu allgemein sind, da dies die Fehlalarmrate erhöhen würde. Der Vorteil der Signaturerkennung liegt zum Teil darin, dass durch die Regeln das Vorgehen leicht verständlich ist und üblicher Weise wenig Fehlalarme gemeldet werden.

2.1.3 Netzbasierte Intrusion Detection Systeme

Netzbasierte IDS (NIDS) überwachen den Datenverkehr in einem Netz auf verdächtige Aktivitäten. Dabei kann ein NIDS dazu eingesetzt werden, sowohl den kompletten Datenverkehr eines Netzes oder auch nur den an den Host, auf dem das System läuft, gerichteten Verkehr zu überwachen. In der Regel wird das NIDS allerdings auf einem eigens dafür vorgesehen Rechner betrieben, um andere Anwendungen nicht zu stören. Somit ist es möglich, dass ein Netz durch nur einen einzigen Rechner überwacht werden kann. Da zur Überwachung Einblick in den gesamten Datenverkehr vorliegt, ist es auch möglich, Angriffe zu erkennen, die mehrere Systeme als Ziel haben. Ein weiterer Vorteil ist, dass es für einen Angreifer schwerer wird, seine Spuren zu verwischen, da neben dem eigentlichen Zielrechner auch Zugriff auf das NIDS erlangt werden müsste, welches wiederum noch schwerer zu erreichen sein kann als das ursprüngliche Ziel. Doch es gibt auch limitierende Faktoren für die Überwachung. Unter anderem stellen Netze, die durch einen Switch verbunden sind, eine Hürde dar, weil sie den sichtbaren Bereich des NIDS einschränken. Auch hohe Datenaufkommen können ein Problem darstellen, wenn der Sensor sie nicht mehr verarbeiten kann. Ein weiteres Problem stellt der Mangel an hostspezifischem Wis-

sen dar, ohne das beispielsweise verschlüsselte Daten nicht überprüft werden können oder nicht abgeschätzt werden kann, wie sich bestimmte Paketsequenzen auf den Host auswirken.

2.1.4 Hostbasierte Intrusion Detection Systeme

Hostbasierte IDS (HIDS) überwachen nur den einen Rechner, auf dem sie betrieben werden, auf verdächtigen Aktivitäten. Da sie direkt auf dem Host arbeiten, sind sie in der Lage, Angriffe auf Anwendungs- oder Betriebssystemebene zu erkennen, indem unter anderem die Zugriffe auf Dateien und Programme überwacht oder auch die Prüfsummen wichtiger Betriebssystemdateien regelmäßig geprüft werden. Dazu zählt beispielsweise auch die Rechteüberschreitung von Benutzern, was nicht zwingender Weise auf einen Angriff von außen hinweisen muss, sondern durchaus auch von einem Insider ausgehen könnte, welcher durch ein NIDS nicht erkannt würde. Ein weiterer Vorteil gegenüber NIDS ist, dass ein HIDS mit verschlüsseltem Datenverkehr umgehen kann und auch die tatsächliche Reaktion des Systems beobachten kann. Die Nähe zum Hostsystem birgt allerdings auch das Risiko, dass im Falle eines gelungenen Angriffs das HIDS selbst manipuliert werden könnte. Entsprechend ist es notwendig, dass ein Eingriff in Echtzeit erkannt wird, bevor die Daten auf denen das HIDS arbeitet oder das System selbst manipuliert werden können [DDW99]. Zu weiteren Nachteilen zählen, dass der Host durch den Betrieb belastet wird, auf jedem zu überwachenden Rechner ein HIDS installiert werden muss und das eingeschränkte Sichtfeld bezüglich des Netzes.

2.2 Kombination von Klassifikatoren

Um Klassifikatoren[3] zu kombinieren, können unterschiedliche *Machine-Learning*- bzw. Sensorfusionsmethoden angwandt werden. Abbildung 2.1 skizziert grob, wie ein entsprechendes System aufgebaut sein kann. Dabei trifft jedes IDS für sich eine Entscheidung, die anschließend zu einer einzigen Entscheidung zusammengeführt werden. Der folgende Absatz gibt einen kleinen Überblick über Fusionsansätze im Bereich der Angriffserkennung. Auf die genannten Ansätze wird in den anschließenden Abschnitten genauer eingegangen.

In ihrer Arbeit kombinieren Han und Cho [HC03] mehrere HIDS-Methoden mittels eines regelbasierten Systems. Giacinto et al. [GRD03] vergleichen mehrere Fusionsansätze, indem sie drei Sensoren auf jeweils unterschiedliche Merkmale hin trainieren und anschließend zusammenführen. In den Experimenten wurden Abstimmungsverfahren, ein *naive Bayes*-basierter

[3] bzw. IDS oder Sensoren; die Begriffe werden hier synonym verwendet

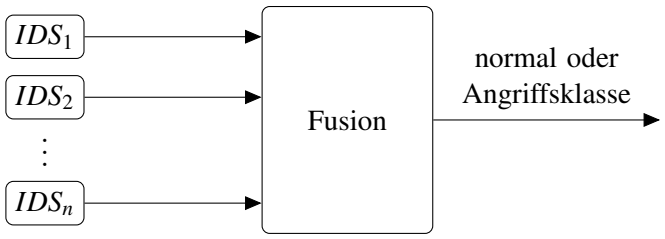

Abbildung 2.1: Mögliches Konzept für die Fusion von IDS.[4]

Ansatz und zwei Methoden, die aus den gegebenen Klassifikatoren den besten für die jeweilige Situation auswählen, *decision templates* und *dynamic classifier selection*, getestet. Mit einer auf Mehrheitsentscheidungen basierten Fusion zeigen Mukkamala et al. [MSA05] durch Experimente, dass ein Ensemble aus drei Klassifikatoren, die jeweils unterschiedliche Verfahren anwenden, den einzelnen Klassifikatoren überlegen ist. Gu et al. [GCL08] beschreiben einen *likelihood ratio test*-Ansatz mit einem Kostenmaß und vergleichen ihn mit verschiedenen Abstimmungsverfahren. Da verschiedene Sensoren unterschiedlich glaubwürdig sein können oder auch zum Teil gar nicht in der Lage sind, gewisse Aktivitäten zu überwachen bzw. zu erkennen, benutzen Yu und Frincke [YF05] einen gewichteten Dempster-Shafer-Theorie (DST) Ansatz. Wang et al. [WYWZ04] verwenden ebenfalls DST, um HIDS und NIDS zu kombinieren. Für ihre *data-dependent decision fusion* Architektur bestimmten Thomas und Balakrishnan [TB08] mit Hilfe von künstlichen neuronalen Netzen (KNN) mehrere Gewichtungen für jedes einzelne IDS in Abhängigkeit der eingehenden Daten. Somit erhält jedes IDS nicht nur eine einzelne Gewichtung, sondern mehrere, aus denen die passende entsprechend der vorliegenden Daten im Fusionsschritt gewählt wird. Für die Fusion selbst ist kein Verfahren vorgegeben, aber Thomas [Tho09] verwendet eine angepasste Variante der DST. Siaterlis und Maglaris [SM04] nutzen ebenfalls DST und kombinieren mehrere Sensoren zur DoS-Erkennung. Dabei haben sie sich gegen Kalman-Filter und KNN entschieden, da diese beiden Ansätze mehr Wissen über das System benötigen. Für ihr hybrides IDS kombinieren Depren et al. [DTAC05] ein Anomalieerkennungs- und ein Signaturerkennungssystem unter der Verwendung eines regelbasierten Ansatzes. Aydın et al. [AZC09] erstellen ebenfalls ein hybrides IDS, allerdings integrieren sie die zwei Anomalieerkennungsmethoden PHAD [MC01] und NETAD [Mah03] direkt in das Signaturerkennungssystem Snort[5] [Roe99].

[4] Die Grafik orientiert sich an Abbildung 1 aus [TB09] und Abbildung 2 aus [GRD03].
[5] `http://www.snort.org` [Stand: 18.03.2014]

Im Folgenden werden die genannten Methoden näher erläutert. Im Vordergrund stehen dabei Abstimmungsverfahren und die *naive Bayes*-basierte Fusion (NBF), da sich diese beiden Ansätze durch ihre relativ unkomplizierten mathematischen Grundlagen gut für die spätere Validierung der Simulationsumgebung eignen. Im Zusammenhang mit NBF wird weiterhin kurz auf Grundlagen der Wahrscheinlichkeitstheorie eingegangen, die ebenfalls relevant für die Validierung sind.

2.2.1 Regelbasierte Systeme

Regelbasierte Systeme verwenden feste *if-then*-Regeln, um Entscheidungen zu treffen. Im Falle der Fusion von IDS könnten einfache Regeln beispielsweise wie folgt aussehen [DTAC05].

WENN die Anomalieerkennung einen Angriff meldet
UND die Signaturerkennung einen Angriff meldet
DANN liegt der Angriff vor, den die Signaturerkennung gemeldet hat,

WENN die Anomalieerkennung keinen Angriff meldet
UND die Signaturerkennung einen Angriff meldet
DANN liegt der Angriff vor, den die Signaturerkennung gemeldet hat,

WENN die Anomalieerkennung einen Angriff meldet
UND die Signaturerkennung keinen Angriff meldet
DANN liegt ein unbekannter Angriff vor.

Der Vorteil solcher Regeln ist, dass sie für den Anwender leicht verständlich sind. Der Nachteil ist allerdings, dass sie vorher definiert (oder trainiert) werden müssen.

2.2.2 Abstimmungsverfahren

Eine recht intuitive Vorgehensweise, um die Entscheidungen mehrerer Klassifikatoren zusammenzuführen, sind Mehrheitsentscheidungen. Kuncheva [Kun04, Seite 112 ff.] listet dazu die drei Varianten relative Mehrheit, einfache Mehrheit und Einstimmigkeit auf. Gu et al. [GCL08]

verwenden im Kontext der Angriffserkennung noch eine weitere Variante, bei der es ausreichend ist, wenn mindestens ein Sensor einen Angriff meldet. Diese vier Methoden werden in Grafik 2.2 veranschaulicht. Dabei stehen die Farben schwarz, grau und weiß für die unterschiedlichen Entscheidungen der Sensoren. In allen Beispielen fällt hier die Entscheidung auf „schwarz“.

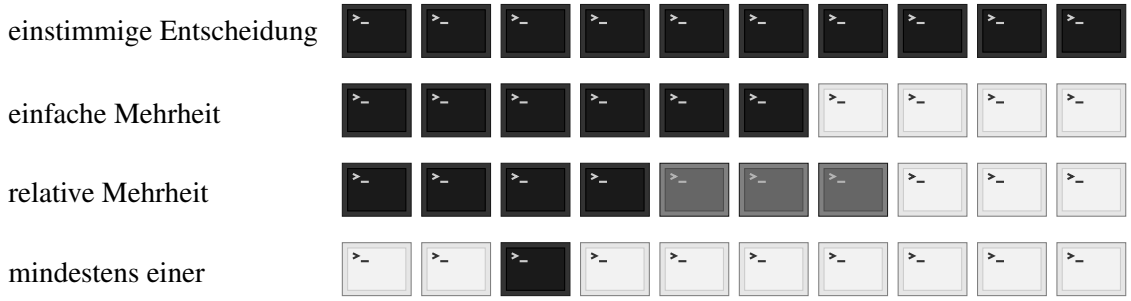

Abbildung 2.2: Beispiel für die vier Abstimmungsformen relative Mehrheit, einfache Mehrheit, einstimmige Entscheidung und mindestens einer. Die getroffene Entscheidung fällt in allen Fällen auf „schwarz“.[6]

Im Folgenden beschreiben die Variablen $d_i \in \{0,1\}$, $i = 1,\ldots,n$, ob ein Sensor D_i einen Angriff (1) oder keinen Angriff (0) meldet. Damit lässt sich eine einstimmige Entscheidung als

$$AND(d_1,\ldots,d_n) = \begin{cases} \text{Angriff}, & \text{wenn } \sum_{i=1}^{n} d_i = n, \\ \text{kein Angriff} & \text{sonst}, \end{cases}$$

beschreiben.

Entsprechend lässt sich die Regel „mindestens einer“, bei der es ausreichend ist, wenn mindestens ein Sensor einen Angriff meldet, als

$$OR(d_1,\ldots,d_n) = \begin{cases} \text{Angriff}, & \text{wenn } \sum_{i=1}^{n} d_i \geq 1, \\ \text{kein Angriff} & \text{sonst}, \end{cases}$$

angeben.

Da die hier betrachteten Sensoren nur die zwei Fälle Angriff und Nicht-Angriff unterscheiden, fallen relative Mehrheit und einfache Mehrheit zu einen Fall zusammen:

[6] Die Grafik wurde in Anlehnung an Abbildung 4.1 aus [Kun04, Seite 113] erstellt.

$$MAJ(d_1,\ldots,d_n) = \begin{cases} \text{Angriff}, & \text{wenn } \sum_{i=1}^{n} d_i \geq \frac{n}{2}, \\ \text{kein Angriff}, & \text{sonst.} \end{cases}$$

Falls die Anzahl der Sensoren n gerade ist, besteht die Möglichkeit, dass es keine einfache Mehrheit gibt. In diesem Fall entscheidet sich *MAJ* dafür einen Angriff zu melden, anstatt einen potentiellen Angriff unerkannt zu lassen. Aber auch die umgekehrte Herransgehensweise wäre möglich und bei Stimmengleichheit könnte kein Angriff gemeldet werden.

Wenn die verwendeten Sensoren unterschiedlich gute Ergebnisse erzielen, ist es sinnvoll, sie entsprechend ihrer Leistung zu gewichten [Kun04, Seite 123]. Dazu kann jedem Sensor D_i ein Koeffizient w_i zugeteilt werden. Sind die Koeffizienten so gewählt, dass

$$\sum_{i=1}^{n} w_i = 1$$

gilt und wird d_i anstatt durch 0 und 1 durch -1 und 1 repräsentiert [GCL08], d.h. $d_i \in \{-1,1\}$, lässt sich die entsprechende Entscheidungsregel folgendermaßen beschreiben.

$$wMAJ(d_1,\ldots,d_n) = \begin{cases} \text{Angriff}, & \text{wenn } \sum_{i=1}^{n} w_i d_i \geq 0, \\ \text{kein Angriff} & \text{sonst.} \end{cases}$$

Im Falle von Stimmengleichheit wird wieder zugunsten des Angriffs entschieden.

Beispiel 2.1 Gegeben seien drei Sensoren D_1, D_2 und D_3 mit einer Gewichtung von $w_1 = w_2 = 0,3$ und $w_3 = 0,4$. Angenommen D_1 und D_3 melden einen Angriff und D_2 hat keinen Angriff erkannt. Dann ergibt sich für die Summe

$$\begin{aligned} \sum_{i=1}^{3} w_i d_i &= 0,3 \cdot 1 + 0,3 \cdot (-1) + 0,4 \cdot 1 \\ &= 0,4. \end{aligned}$$

Entsprechend ist die Entscheidung von $wMAJ(1,(-1),1) =$ Angriff, es wird also angenommen, dass ein Angriff vorliegt.

2.2.3 Naive Bayes-basierte Fusion

Die *naive Bayes*-basierte Fusion (NBF) ist im Gegensatz zu Abstimmungsverfahren ein probabilistischer Ansatz, der berechnet, wie wahrscheinlich ein Angriff (bzw. kein Angriff) unter den zur Verfügung stehenden Sensorenaussagen ist.

Zappi et al. [ZSF+07] und Altınçay [Alt05] verstehen NBF im Sinne eines *naive Bayes*-Klassifikators, der als Eingabe die Ergebnisse anderer Klassifikatoren erhält. Sudano [Sud03] hingegen schränkt den Begriff zusätzlich ein und geht für NBF davon aus, dass das Auftreten der möglichen Klassen gleichverteilt ist. Diese Annahme vereinfacht zwar die Berechnung, würde im Falle der Angriffserkennung aber bedeuten, dass davon ausgegangen wird, dass die A-priori-Wahrscheinlichkeit eines Angriffs genau so hoch ist, wie die, dass kein Angriff stattfindet. Um sich nicht vorab einer solchen Einschränkung zu unterwerfen, wird der Begriff hier wie von Zappi et al. und Altınçay verwendet.

2.2.3.1 Wahrscheinlichkeit

Um die grundlegenden Konzepte zu klären, wird auf die entsprechenden Einführungen von Pearl [Pea94, Seite 29 ff.] und Russell und Norvig [RN04, Seite 570 ff.] zurückgegriffen.

Um mit unsicherem Wissen umgehen zu können, wird einer Aussage A (die wahr oder falsch sein kann) ein Glaubensgrad $P(A) = p$ zugeordnet, der angibt, wie wahrscheinlich es ist, dass A wahr ist. Dabei folgt $P(A)$ den drei grundlegenden Axiomen der Wahrscheinlichkeitstheorie von Kolmogorow [Pea94, Seite 30]:

1. $0 \leq P(A) \leq 1$,
2. $P(wahr) = 1$,
3. $P(A \vee B) = P(A) + P(B)$, wenn A und B disjunkt sind.

Wenn neben A keine weiteren Aussagen vorliegen oder berücksichtigt werden sollen, wird $P(A)$ als unbedingte oder A-priori-Wahrscheinlichkeit bezeichnet. Entsprechend gibt es auch die bedingte oder A-posteriori-Wahrscheinlichkeit $P(A \mid B)$, bei der eine Aussage A unter der Annahme, dass eine weitere Aussage B gilt, betrachtet wird. Ein Beispiel dafür ist „die Wahrscheinlichkeit, dass tatsächlich ein Angriff vorliegt, wenn alle Sensoren einen Angriff melden“.

Um die bedingte Wahrscheinlichkeit bestimmen zu können, kann auf die unbedingte Wahrscheinlichkeit zurückgegriffen werden [Pea94, Seite 31]:

$$P(A \mid B) = \frac{P(A,B)}{P(B)}, \tag{2.1}$$

wobei $P(A,B)$ eine Kurzform für $P(A \wedge B)$ ist und $P(B) > 0$ gelten muss.

2.2.3.2 Naive Bayes

Für die Fusion ist die bedingte Wahrscheinlichkeit, ob ein (oder kein) Angriff vorliegt, gegeben die aktuell vorliegenden Sensorenauswertungen, von Interesse, um darauf basierend eine Entscheidung zu treffen. Diese bedingte Wahrscheinlichkeit, $P(A \mid B)$, kann mit Hilfe von Gleichung 2.1 berechnet werden. Um bei der Bestimmung des Zählers $P(A,B)$ nicht auf die vollständige gemeinsame Verteilung der Hypothese, ob ein oder kein Angriff vorliegt, und aller Sensorenaussagen angewiesen zu sein, kann ausgenutzt werden, dass die Gleichung auch für $P(B \mid A)$ aufgestellt werden kann.

$$P(B \mid A) = \frac{P(A,B)}{P(A)}$$

Wenn diese Gleichung nun nach $P(A,B)$ umgestellt wird

$$P(A,B) = P(B \mid A)P(A) \tag{2.2}$$

wird sie als Produktregel bezeichnet [RN04, Seite 579] und kann so anschließend für den Ausdruck $P(A,B)$ in Gleichung 2.1 eingesetzt werden.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Die so erhaltene Gleichung wird als Bayessche Regel[7] bezeichnet [RN04, Seite 590]. In dieser Form lassen sich die drei Wahrscheinlichkeiten auf der rechten Seite der Gleichung (und jeweils deren Negation), mit Hilfe von Trainingsdaten oder über Expertenwissen bestimmen. Dabei entspricht $P(A)$ der Wahrscheinlichkeit, dass ein Angriff überhaupt auftritt, $P(B \mid A)$ der Wahrscheinlichkeit, dass der Sensor einen Angriff erkennt, wenn er vorliegt und $P(B)$ der Wahr-

[7] auch Bayessches Gesetz oder Bayessches Theorem genannt

scheinlichkeit, dass der Sensor sich grundsätzlich für einen Angriff entscheidet. Wenn allerdings mehr als nur ein Sensor berücksichtigt werden sollten, steigt die Komplexität der Gleichung schnell an, da sich B zu $B_1, \ldots, B_n$ entwickelt, womit sich 2^n verschiedenen Möglichkeiten für die Sensorenbelegung (wenn die Sensoren nur zwischen Angriff und keinem Angriff unterscheiden) ergeben [RN04, Seite 592]:

$$P(A \mid B_1, \ldots, B_n) = \frac{P(B_1, \ldots, B_n \mid A)P(A)}{P(B_1, \ldots, B_n)}. \tag{2.3}$$

Eine mögliche Vereinfachung ist die Annahme der bedingten Unabhängigkeit der Sensoren bzgl. A, d.h. dass das Ergebnis eines Sensors nur davon abhängig ist, ob ein Angriff vorliegt und nicht von den Ergebnissen der anderen Sensoren beeinflusst wird. Allgemein lässt sich die bedingte Unabhängigkeit zwischen zwei Variablen X und Y bzgl. einer dritten Variablen Z durch folgende Gleichung ausdrücken [RN04, Seite 593]:

$$P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z).$$

Da diese Annahme oft fälschlicherweise auch auf eigentlich voneinander abhängige Variablen angewandt wird, wird sie als *naive Bayes* bezeichnet [RN04, Seite 594]. Zwar geben Gu et al. [GCL08] an, dass es durchaus möglich ist, die Abhängigkeiten zwischen den Sensoren zu modellieren, aber sie weisen auch darauf hin, dass die Annahme der bedingten Unabhängigkeit bereits eine ausreichende Näherung für die Praxis liefert. Mit Hilfe dieser (naiven) Annahme lässt sich der Zähler aus Gleichung 2.3 nun zu

$$\begin{aligned} P(B_1, \ldots, B_n \mid A)P(A) &= P(B_1 \mid A)P(B_2 \mid A) \ldots P(B_n \mid A)P(A) \\ &= P(A)\prod_{i=1}^{n} P(B_i \mid A) \end{aligned}$$

vereinfachen. Als Formel um $P(A \mid B_1, \ldots, B_n)$ zu bestimmen ergibt sich also

$$P(A \mid B_1, \ldots, B_n) = \frac{P(A)\prod_{i=1}^{n} P(B_i \mid A)}{P(B_1, \ldots, B_n)}.$$

Der Nenner dient dabei der Normalisierung und bleibt konstant, solange sich die Bedingungen $B_1, \ldots, B_n$ nicht ändern. Beispielsweise wenn für eine vorgegebene Sensorenbelegung die Wahrscheinlichkeiten für verschiedene Angriffsklassen gesucht sind. Russell und Norvig [RN04, Sei-

te 586] verwenden in diesem Fall die Normalisierungskonstante α als Kurzform für den Normalisierungsfaktor $\frac{1}{P(B_1,\ldots,B_n)}$.

$$P(A \mid B_1,\ldots,B_n) = \alpha P(A)\prod_{i=1}^{n} P(B_i \mid A) \tag{2.4}$$

2.2.3.3 Entscheidungsregeln

Der nächste Schritt, da nun die Wahrscheinlichkeit eines Angriffs auf Grundlage der Sensorenentscheidungen bestimmt werden kann, ist, basierend auf diesen Daten zu entscheiden, ob tatsächlich ein Angriff vorliegt. Um solch eine Entscheidung zu treffen, stehen verschiedene Entscheidungsregeln zur Auswahl, auf die die NBF hin trainiert werden kann. Eine verbreitete Variante ist die *maximum a posteriori* (MAP) Methode, bei der für die vorliegende Evidenz E die Hypothese H_i mit der höchsten A-posteriori-Wahrscheinlichkeit $P(H_i \mid E)$ gewählt wird [RN04, Seite 870].

$$\begin{aligned} NBF_{MAP}(E_1,\ldots,E_n) &= \arg\max_i \alpha P(H_i)\prod_{j=1}^{n} P(E_j \mid H_i) \\ &= \arg\max_i P(H_i)\prod_{j=1}^{n} P(E_j \mid H_i) \end{aligned}$$

Da die Normalisierungskonstante α keinen Einfluss darauf hat, für welche Belegung die Wahrscheinlichkeit maximal ist, kann sie hier weggelassen werden.

Ein Problem, das MAP mit sich bringt, ist, dass die A-priori-Wahrscheinlichkeit, dass ein Angriff überhaupt stattfindet im Verhältnis zu der Wahrscheinlichkeit, dass kein Angriff erfolgt, sehr gering sein kann. Das kann dazu führen, dass viele Angriffe fälschlicherweise als Nicht-Angriff eingestuft werden, da nur versucht wird, die A-posteriori-Wahrscheinlichkeit zu maximieren und keine Rücksicht auf mögliche Fehlklassifizierungen genommen wird [Mar09, Seite 170 f.]. Eine Vereinfachung der MAP-Methode, die dieses Problem „umgeht“, ist anzunehmen, dass die A-priori-Wahrscheinlichkeiten $P(H_i)$ gleichverteilt sind, was dazu führt, dass dieser Term ebenfalls weggelassen werden kann. Die so entstandene Entscheidungsregel wird *maximum likelihood* (ML) Methode genannt [RN04, Seite 871].

$$NBF_{ML}(E_1,\ldots,E_n) = \arg\max_i \prod_{j=1}^{n} P(E_j \mid H_i)$$

Weitere Entscheidungsregeln sind unter anderem die *recalibrated likelihood*, bei der anstatt die A-priori-Wahrscheinlichkeit $P(H_i)$ ganz wegzulassen, sie durch eine Gewichtung w_i für jede Hypothese H_i ersetzt wird, um so die Wahrscheinlichkeitsverteilung „nachzujustieren“ [Fla12, Seite 275] und auch Ansätze, die die Kosten für bestimmte Entscheidungen modellieren und berücksichtigen. Im Fall von *minimum risk* wird dabei eine *loss matrix* angelegt, in der die Kosten für eine Fehlklassifikation notiert werden. Entsprechend dieser Matrix wird versucht, das geringste Risiko bei der Entscheidung zu finden, anstatt die A-posteriori-Wahrscheinlichkeit zu maximieren [Mar09, Seite 171].

Unter der Einschränkung, dass H_i nur zwei Werte annehmen kann, besteht auch die Möglichkeit, das Verhältnis der beiden Fälle zu betrachten [Fla12, Seite 28]. Für die *posterior odds* (PO) genannte Methode wird dazu der Quotient der beiden A-posteriori-Wahrscheinlichkeiten gebildet.

$$\frac{P(H \mid E_1, \ldots, E_n)}{P(\neg H \mid E_1, \ldots, E_n)} = \frac{P(H) \prod_{i=1}^{n} P(E_i \mid H)}{P(\neg H) \prod_{i=1}^{n} P(E_i \mid \neg H)}$$

Ist die Verteilung der A-priori-Wahrscheinlichkeiten nicht von Interesse oder gleichverteilt, kann, wie bei der Vereinfachung von MAP zu ML, der Faktor $P(H)$ weggelassen werden. Der so entstandene Quotient wird als *likelihood ratio* und die daraus resultierende Entscheidungsregel (bzw. Test welche der beiden Hypothesen zutrifft) als *likelihood ratio test* (LRT) bezeichnet.

$$\frac{\prod_{i=1}^{n} P(E_i \mid H)}{\prod_{i=1}^{n} P(E_i \mid \neg H)}$$

Als Ergebnis wird die Hypothese mit dem größeren Wert gewählt, d.h. wenn der Quotient größer als 1 ist, H und wenn der Quotient kleiner als 1 ist, $\neg H$. Falls beide Werte gleich sind, kann der Gleichstand je nach Anwendung beliebig aufgelöst werden. In diesem Fall wird zugunsten des Zählers entschieden, also ein Angriff gemeldet. Damit ergeben sich für die PO-Methode und den LRT die folgenden Entscheidungsregeln.

$$NBF_{PO}(E_1, \ldots E_n) = \begin{cases} H, & \text{wenn } \frac{P(H) \prod_{i=1}^{n} P(E_i|H)}{P(\neg H) \prod_{i=1}^{n} P(E_i|\neg H)} \geq 1, \\ \neg H & \text{sonst.} \end{cases} \tag{2.5}$$

$$NBF_{LRT}(E_1, \ldots E_n) = \begin{cases} H, & \text{wenn } \frac{\prod_{i=1}^{n} P(E_i|H)}{\prod_{i=1}^{n} P(E_i|\neg H)} \geq 1, \\ \neg H & \text{sonst.} \end{cases} \tag{2.6}$$

Es ist auch möglich, einen anderen Vergleichswert zu wählen, der ggf. bessere Entscheidungen erlaubt. Gu et al. [GCL08] verwenden beispielsweise für ihren LRT-basierten Ansatz einen Wert, der auf einem Kostenmaß für Fehlklassifizierungen aufbaut.

Beispiel 2.2 Gegeben seien drei IDS, von denen aktuell zwei einen Angriff melden, $E_1 = E_2 =$ Angriff, den der dritte Sensor nicht bestätigt, $E_3 =$ Normal. Mit Hilfe von NBF soll nun entschieden werden, ob es sich um einen Angriff handelt (H_a) oder nicht (H_n). Dazu ist weiterhin die Wahrscheinlichkeit, dass ein Angriff überhaupt stattfindet mit $P(H_a) = 0,05$ vorgegeben. Entsprechend gilt $P(H_n) = P(\neg H_a) = 0,95$, d.h. in 95% der Fälle handelt es sich um normale Aktivitäten. Die Erkennungsraten der einzelnen IDS sind gleich und mit $P(\text{Angriff} \mid H_a) = 0,9$ und $P(\text{Normal} \mid H_n) = 0,9$ spezifiziert.

Um die MAP-Regel anwenden zu können, müssen zunächst, unter der Verwendung von Gleichung 2.4, die Werte für die unterschiedlichen Hypothesen (in diesem Fall Angriff und Normal) bestimmt werden.

$$\begin{aligned}
P(H_a \mid E_1, \ldots, E_3) &= \alpha P(H_a) P(\text{Angriff} \mid H_a) P(\text{Angriff} \mid H_a) P(\text{Normal} \mid H_a) \\
&= \alpha P(H_a) P(\text{Angriff} \mid H_a) P(\text{Angriff} \mid H_a)(1 - P(\text{Angriff} \mid H_a)) \\
&= \alpha \cdot 0,05 \cdot 0,9 \cdot 0,9 \cdot (1 - 0,9) = 0,00405\alpha \\
P(H_n \mid E_1, \ldots, E_3) &= \alpha P(H_n) P(\text{Angriff} \mid H_n) P(\text{Angriff} \mid H_n) P(\text{Normal} \mid H_n) \\
&= \alpha P(H_n)(1 - P(\text{Normal} \mid H_n))(1 - P(\text{Normal} \mid H_n)) P(\text{Normal} \mid H_n) \\
&= \alpha \cdot 0,95 \cdot (1 - 0,9) \cdot (1 - 0,9) \cdot 0,9 = 0,00855\alpha
\end{aligned}$$

Da der Wert für Normal ($0,00855\alpha$) größer als der Wert für Angriff ($0,00405\alpha$) ist, ergibt sich $NBF_{MAP}(\text{Angriff}, \text{Angriff}, \text{Normal}) = \text{Normal}$.

Für ML kann ein Teil der Rechnung wiederverwendet werden, da die entsprechenden A-priori-Wahrscheinlichkeiten nicht berücksichtigt werden müssen.

$$\begin{aligned}
\prod_{i=1}^{3} P(E_i \mid H_a) &= P(\text{Angriff} \mid H_a) P(\text{Angriff} \mid H_a) P(\text{Normal} \mid H_a) \\
&= 0,9 \cdot 0,9 \cdot (1 - 0,9) = 0,081 \\
\prod_{i=1}^{3} P(E_i \mid H_n) &= P(\text{Angriff} \mid H_n) P(\text{Angriff} \mid H_n) P(\text{Normal} \mid H_n) \\
&= (1 - 0,9) \cdot (1 - 0,9) \cdot 0,9 = 0,009
\end{aligned}$$

Ohne die A-priori-Wahrscheinlichkeit fällt die Entscheidung auf $NBF_{ML}(\text{Angriff}, \text{Angriff}, \text{Normal})$ = Angriff, da in diesem Fall der Wert für Angriff $(0,081\alpha)$ größer als der Wert für Normal $(0,009\alpha)$ ist.

Da in diesem Beispiel nur zwischen den beiden Klassen H_a und H_n unterschieden wird, können auch die PO- und *likelihood ratio*-Methoden angewandt werden. Die PO-Regel setzt auf dem Verhältnis der A-posteriori-Wahrscheinlichkeiten auf, die bereits für MAP berechnet wurden.

$$\frac{P(H_a)\prod_{i=1}^{3} P(E_i \mid H_a)}{P(H_n)\prod_{i=1}^{3} P(E_i \mid H_n)} = \frac{0,00405}{0,00855} \approx 0,47368$$

Da der Quotient mit 0,47368 nicht größer oder gleich 1 ist, ist das Ergebnis $NBF_{PO}(\text{Angriff}, \text{Angriff}, \text{Normal}) = \text{Normal}$.

Für den LRT können die Werte von ML wiederverwendet werden, indem sie ins Verhältnis gesetzt werden.

$$\frac{\prod_{i=1}^{3} P(E_i \mid H_a)}{\prod_{i=1}^{3} P(E_i \mid H_n)} = \frac{0,081}{0,009} = 9$$

Die Entscheidung fällt hier auf $NBF_{LRT}(\text{Angriff}, \text{Angriff}, \text{Normal}) = \text{Angriff}$, da der Quotient mit 9 größer als 1 ist.

Zu beachten ist, dass nicht bekannt ist, ob nun tatsächlich ein Angriff vorliegt oder nicht. Es wurde nur bestimmt, welche Entscheidung die jeweilige NBF-Variante in dem gegebenen Szenario treffen würde. Weiterhin zeigt dieses Beispiel gut, dass obwohl die Erkennungsraten bei jeweils 90% liegen, die A-priori-Wahrscheinlichkeit eines Angriffs von nur 5% bereits einen Einfluss auf die Entscheidung nimmt. Die beiden Varianten, die diese Wahrscheinlichkeit berücksichtigen, MAP und PO, entscheiden sich jeweils gegen einen Angriff, da es mit 5% eher unwahrscheinlich scheint, dass es sich wirklich um einen Angriff handelt. Die beiden anderen Methoden, ML und LRT, hingegen würden in diesem Beispiel einen Angriff melden.

2.2.4 Dempster-Shafer-Theorie

Die Dempster-Shafer-Theorie (DST) oder Evidenztheorie, ist eine Weiterentwicklung der Arbeit von Arthur P. Dempster durch Glenn Shafer [Sha76]. Die Theorie zeichnet sich dadurch aus, dass sie zwischen Unsicherheit und Unwissen unterscheidet [RN04, Seite 645]. Dazu wird, anders, als es bei dem bayesschen Ansatz der Fall ist, statt der Wahrscheinlichkeit, dass eine Aussage zutrifft, die Wahrscheinlichkeit, dass die vorliegenden Daten die Aussage unterstützen, bestimmt. Dabei wird zum einen der Glaubensgrad, alle Evidenz, die für die Aussage spricht,

und zum anderen die Plausibilität, alle Evidenz, die nicht gegen die Aussage steht, gebildet [Kle12, Seite 185 f.]. Zusammen ergeben diese beiden Werte ein Glaubensintervall, wie in Abbildung 2.3 dargestellt. Kombiniert werden können die Aussagen mehrerer Sensoren mit der

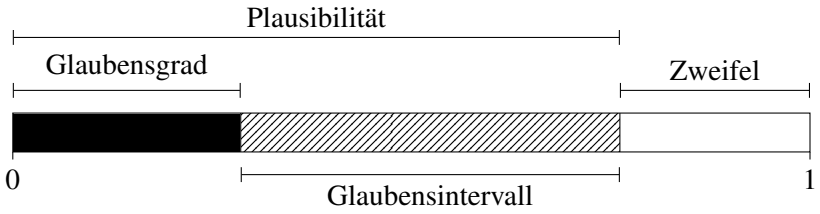

Abbildung 2.3: Der schwarze Bereich stellt den Glaubensgrad dar. Zusammen mit dem schraffierten Bereich bildet er die Plausibilität. Der verbleibende weiße Bereich gibt den Zweifel an der Richtigkeit der Aussage an. Der schraffierte Abschnitt alleine gibt das Glaubensintervall an, in dem die Wahrscheinlichkeit, dass die Aussage zutrifft, liegt.[8]

Kombinationsregel von Dempster (*Dempster's rule of combination*). Da allerdings Konflikte zwischen widersprüchlichen Evidenzen aufgelöst werden, indem sie verworfen bzw. anteilig auf die restlichen Fälle verteilt werden, kann es zu unintuitiven Ergebnissen kommen [Kle12, Seite 183], wie das folgende Beispiel zeigt.

Beispiel 2.3 Gegeben seien drei Sensoren, von denen zwei einen Angriff melden. Alle drei Sensoren sind sich in ihrer Entscheidung zu 100% sicher. Da die Kombinationsregel von Dempster kommutativ und assoziativ ist [SF02], können die drei Sensoren in beliebiger Reihenfolge kombiniert werden. Wenn die beiden Sensoren, die einen Angriff erkannt haben, kombiniert werden, ist das Ergebnis ein Glaubensgrad von 1 für einen Angriff. Sobald allerdings der dritte Sensor, der keinen Angriff erkannt hat, mit aufgenommen wird, ergibt sich ein Widerspruch. Um diesen Konflikt aufzulösen werden die widersprüchlichen Kombinationen verworfen. Da sich aber alle drei Sensoren ihrer Entscheidung zu 100% sicher sind, bleibt neben den widersprüchlichen Kombinationen nichts mehr übrig, was dazu führt, dass es kein Ergebnis gibt bzw. der Glaubensgrad für einen Angriff auf 0 sinkt und der Glaubensgrad gegen einen Angriff ebenfalls 0 beträgt. Wenn die Sensoren sich allerdings nur zu 99% sicher wären, wäre das Ergebnis ein Glaubensgrad von $0,99$ für einen Angriff und $0,01$ gegen einen Angriff.

Da das Ergebnis der Kombination wieder Glaubensintervalle für die unterschiedlichen Klassen liefert, muss, ähnlich wie bei NBF, anschließend noch eine Entscheidungsfindung durchgeführt werden.

[8] Die Grafik orientiert sich an Abbildung 6.2 aus [Kle12, Seite 187].

2.2.5 Künstliche neuronale Netze

Nach Russell und Norvig [RN04, Seite 896 ff.] besteht ein künstliches neuronales Netz (KNN) aus mehreren Knoten (oder Einheiten), die durch gerichtete Kanten miteinander verbunden sind. Jede Einheit beinhaltet eine Aktivierungsfunktion, die die Werte der eingehenden Kanten verarbeitet und so ein entsprechendes Ergebnis als Ausgabe der Einheit bestimmt. Diese Ergebnisse werden über die ausgehenden Kanten an die folgenden Einheiten weitergereicht, die ihrerseits ebenfalls Ausgaben erzeugen und weiterleiten, bis eine Ausgabeschicht erreicht ist, die keine ausgehenden Kanten besitzt. Entsprechend gibt es auch eine Eingabeschicht, die keine eingehenden Kanten verwendet und dafür verantwortlich ist, die Eingabedaten in das KNN zu übertragen. Neben diesen beiden Schichten können noch beliebig verborgene Einheiten eingebaut werden, die aber eben nicht nach außen sichtbar sind. Um Einfluss auf die einzelnen Einheiten nehmen zu können, ist jede Kante mit einer Gewichtung versehen. Diese Gewichtungen werden mit Hilfe von Trainingsdaten nach und nach justiert bis das Netz das gewünschte Ergebnis erzielt. Ein einfaches KNN mit vier Eingabeknoten, einer verborgenen Schicht und einer Ausgabeeinheit ist in Abbildung 2.4 dargestellt. Da das dargestellte Netz keine Zyklen enthält, wird es auch als Netz ohne Rückkopplung oder Feedforward-Netz bezeichnet. Im Gegensatz dazu wird ein Netz das Zyklen beinhaltet Netz mit Rückkopplung oder Recurrent-Netz genannt.

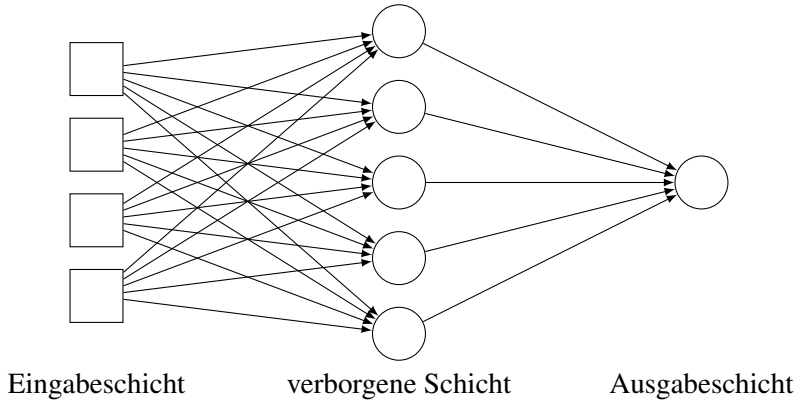

Abbildung 2.4: Beispiel eines KNN mit einer versteckten Schicht.

2.2.6 Decision Templates

Decision Templates [KBD01] vergleichen die aktuellen Sensorenausgaben mit vorher trainierten Schablonen (*templates*) und wählen die ähnlichste aus, um so die Klasse zu bestimmen. Dazu

wird zunächst für jede Klasse ein *decision template* trainiert. Die verwendeten Klassifikatoren D_i liefern zu jeder Eingabe x einen Ausgabevektor $D_i(x)$, der für jede der c Klassen eine Bewertung enthält, wie sicher sich der Klassifikator ist, dass es sich um die entsprechende Klasse handelt. Die einzelnen Ausgaben zusammen ergeben eine Matrix, wobei jede Zeile einem (transponierten) Ausgabevektor entspricht. Jede Zelle in der Matrix enthält somit einen Wert, der angibt, für wie wahrscheinlich der jeweilige Klassifikator die entsprechende Klasse bei den vorliegenden Daten hält. Eine solche Matrix ist in Abbildung 2.5 dargestellt. Um schließlich mehrere Senso-

$$DP(x) = \begin{pmatrix} d_{1,1}(x) & \dots & d_{1,j}(x) & \dots & d_{1,c}(x) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{i,1}(x) & \dots & d_{i,j}(x) & \dots & d_{i,c}(x) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{L,1}(x) & \dots & d_{L,j}(x) & \dots & d_{L,c}(x) \end{pmatrix}$$

Ausgabe von Klassifikator $D_i(x)$

Unterstützung von Klassifikator $D_1, \dots, D_L$ für Klasse j

Abbildung 2.5: Der Aufbau eines *decision profiles* bzw. *decision templates*, wie in [KBD01] Formel 3 vorgestellt. Dabei steht L für die Anzahl der Klassifikatoren und c für die Anzahl der Klassen.

ren zu kombinieren, wird aus den aktuellen Sensorenausgaben ein *decision profile* erstellt, das ebenfalls eine solche Matrix ist. Zu diesem Profil wird dann das ähnlichste *decision template* ermittelt und die Klasse mit der dieses *template* trainiert wurde als Entscheidung gewählt.

2.2.7 Dynamic Classifier Selection

Anstatt Klassifikatoren zu kombinieren, wählt die *Dynamic Classifier Selection* [GRF00] den für die aktuelle Sensorenbelegung am besten geeignetsten Klassifikator aus und übernimmt seine Entscheidung. Um diesen Klassifikator zu finden, wird ein Teil der Trainingsdaten abgetrennt und statt zum Training zur späteren Validierung der Klassifikatoren verwendet. Wenn ein Ereignis vorliegt, werden mit Hilfe eines k-Nearest-Neighbors-Algorithmus die k ähnlichsten Fälle aus den Validierungsdaten gewählt und verglichen, wieviele dieser k Fälle die einzelnen Klassifikatoren richtig erkennen. Der Klassifikator, der auf diesem Auszug aus den Validierungsdaten

die höchste Erkennungsrate liefert, wird ausgewählt, um das eigentliche Ereignis zu klassifizieren.

2.3 Bewertungsmetriken

Um die Qualität der Sensorfusionsergebnisse und auch der IDS selbst vergleichen zu können, stehen unterschiedliche Bewertungskriterien zur Verfügung. Lazarevic et al. [LKS05] haben dazu die drei Punkte Vorhersagekraft (*prediction performance*), Reaktionszeit (*time performance*) und Ausfalltoleranz (*fault tolerance*) als besonders relevant eingestuft. Um eine gute Vorhersagekraft zu erreichen, sollte ein System in der Lage sein, Angriffe zu erkennen, ohne dabei Fehlalarme zu produzieren. Die Reaktionszeit umfasst sowohl die Zeit, die benötigt wird, um die vorliegenden Daten auszuwerten, als auch die Zeit, die es braucht, einen gefundenen Angriff zu melden. Das dritte Kriterium, die Ausfalltoleranz, bezieht sich auf die Art und Weise, wie ein IDS mit Angriffen, die auf das IDS selbst gerichtet sind, umgehen kann, wie beispielsweise DoS und *buffer overflow* Angriffe, aber auch das künstliche Erzeugen einer großen Anzahl von Fehlalarmen. Die Reaktionszeit und die Ausfalltoleranz beziehen sich auf ein IDS als Ganzes, inklusive der softwareseitigen Umsetzung und auch der Hardware. Da in dieser Arbeit aber die Sensorfusionsmethoden im Vordergrund stehen, wird die Vorhersagekraft als Bewertungskriterium verwendet.

Als Basis dafür dient eine Wahrheitsmatrix (auch Konfusionsmatrix) wie in Tabelle 2.1 [Faw06].

		tatsächliche Ereignisse		
		Angriff	Normal	
IDS	Angriff	richtig positive (RP)	falsch positive (FP)	$RP+FP$
	Normal	falsch negative (FN)	richtig negative (RN)	$FN+RN$
		$RP+FN$	$FP+RN$	$RP+FN+FP+RN$

Tabelle 2.1: Wahrheitsmatrix

Dabei bezeichnet RP die Anzahl der korrekt erkannten Angriffe und RN die Zahl der richtig eingestuften normalen Ereignisse. FP sind die fälschlicherweise als Angriff gemeldeten normalen Ereignisse, also Fehlalarme, und FN ist die Zahl der Angriffe, die nicht als solche erkannt wurden und so vom IDS unentdeckt bleiben. Aus diesen Werten, die beispielsweise durch Experimente ermittelt werden können, lassen sich unterschiedliche Kennzahlen berechnen. Eini-

ge dieser Kennzahlen werden im Folgenden, auf Basis der einführenden Übersicht von Flach [Fla12, Seite 53 ff., 346 f.], näher beschrieben.

Die Korrektklassifikationsrate (auch *accuracy*) repräsentiert das Verhältnis der richtig klassifizierten Instanzen zur Gesamtmenge und kann wie folgt berechnet werden.

$$\text{Korrektklassifikationsrate} = \frac{RP + RN}{RP + FN + RN + FP}$$

Entsprechend lässt sich auch die Falschklassifikationsrate (auch *error rate*) als Gegenstück bestimmen.

$$\begin{aligned}\text{Falschklassifikationsrate} &= \frac{FP + FN}{RP + FN + RN + FP} \\ &= 1 - \text{Korrektklassifikationsrate}\end{aligned}$$

Da diese beiden Metriken sowohl die RP als auch die RN in einem Wert zusammenfassen, kann es zu Problemen kommen, wenn die betrachteten Klassen ungleich verteilt sind. In solch einem Fall kann es ausreichend sein, wenn die häufiger vertretene Klasse sehr gut erkannt wird, da die andere Klasse kaum ins Gewicht fällt. Der Effekt, wenn die Anzahl der einen Klasse, hier keine Angriffe, eine wesentlich höhere Häufigkeit hat, wird in Beispiel 2.4 gezeigt.

Beispiel 2.4 Gegeben sind zwei IDS, die beide eine Korrektklassifikationsrate von 90% erzielen. Allerdings sind bei den Testdaten für das erste IDS (Tabelle 2.2a) die Anzahl der Angriffe und Nicht-Angriffe gleich, während bei den Daten, mit denen das zweite IDS (Tabelle 2.2b) getestet wurde, die Anzahl der Angriffe deutlich geringer ist. Obwohl das zweite System jedes

		tatsächliche Ereignisse		
		Angriff	Normal	
IDS_1	Angriff	45	5	50
	Normal	5	45	50
		50	50	100

(a) Wahrheitsmatrix für IDS_1

		tatsächliche Ereignisse		
		Angriff	Normal	
IDS_2	Angriff	0	0	0
	Normal	10	90	100
		10	90	100

(b) Wahrheitsmatrix für IDS_2

Tabelle 2.2: Zwei verschiedene IDS, die auf unterschiedlichen Testdaten (gleichverteilt für IDS_1 und mit wesentlich weniger Angriffen als Nicht-Angriffen für IDS_2) jeweils eine Korrektklassifikationsrate von 90% erreichen.

Ereignis als normal einstuft und so niemals einen Angriff melden würde, erreicht es ebenfalls eine Korrektklassifikationsrate von 90%.

Da Angriffe in der Regel weniger häufig als normaler Datenverkehr sind, ist die Korrektklassifikationsrate in diesem Fall als Kennzahl ungeeignet. Der positive Vorhersagewert (auch *precision* oder *positive predictive value (PPV)*) verwendet ebenfalls Werte aus beiden Spalten der Wahrheitsmatrix, allerdings berücksichtigt er nur die erste Zeile und vermischt somit nicht die RP mit den RN. Er gibt den Anteil der als korrekt positiv klassifizierten Instanzen an allen als positiv klassifizierten Instanzen an, d.h.

$$\text{positiver Vorhersagewert} = \frac{RP}{RP+FP}.$$

Umgekehrt gibt der negative Vorhersagewert (auch *negative predictive value (NPV)*) das Verhältnis zwischen richtigerweise als negativ eingestuften Ereignissen und allen als negativ erkannten Ereignissen an.

$$\text{negativer Vorhersagewert} = \frac{RN}{RN+FN}$$

Eine Kennzahl, die sich nur auf die tatsächlichen Angriffe bezieht, ist die Sensitivität (auch Richtig-Positiv-Rate oder *recall*), die den Anteil der richtig erkannten Angriffe im Verhältnis zu den tatsächlichen Angriffen angibt.

$$\text{Sensitivität} = \frac{RP}{RP+FN} \tag{2.7}$$

Das Gegenstück dazu ist die Falsch-Negativ-Rate, die den Anteil der fälschlicherweise als normal deklarierten Angriffe angibt.

$$\begin{aligned}\text{Falsch-Negativ-Rate} &= \frac{FN}{RP+FN} \\ &= 1 - \text{Sensitivität}\end{aligned}$$

Entsprechend gibt es für Nicht-Angriffe die Spezifität (auch Richtig-Negativ-Rate), die die richtig erkannten normalen Ereignisse anteilig an allen normalen Ereignissen zeigt.

$$\text{Richtig-Negativ-Rate} = \frac{RN}{RN+FP}$$

Die passende Umkehrung dazu bildet die Falsch-Positiv-Rate (auch Fehlalarmrate), die sich auf den Anteil der als Angriff eingestuften Nicht-Angriffe, also Fehlalarme, bezieht.

$$\begin{aligned}\text{Falsch-Positiv-Rate} &= \frac{FP}{RN+FP} \\ &= 1 - \text{Richtig-Negativ-Rate}\end{aligned} \tag{2.8}$$

Beispiel 2.5 Wenn für die beiden IDS aus Beispiel 2.4 die Sensitivität und die Spezifität berechnet werden, ist ein deutlicher Unterschied zwischen den Systemen zu erkennen. Für das erste IDS ergibt sich eine Sensitivität von $0,9$ und eine Spezifität von ebenfalls $0,9$. Währenddessen erreicht das zweite System zwar eine Spezifität von 1, allerdings nur eine Sensitivität von 0, was zeigt, dass dieses IDS nicht zum Erkennen von Angriffen geeignet ist.

Eine weitere Metrik ist das F-Maß, das den positiven Vorhersagewert und die Sensitivität über ihr harmonisches Mittel zu einem Wert zusammenfasst.

$$F = \frac{2 \cdot \text{positiver Vorhersagewert} \cdot \text{Sensitivität}}{\text{positiven Vorhersagewert} + \text{Sensitivität}}$$

Das F-Maß ist gegenüber den RN unabhängig, da es nur auf Basis der RP, FP und FN bestimmt wird. Daher ist es gut für Bereiche geeignet, in denen die Anzahl der negativen Instanzen deutlich größer ist, als die der positiven Instanzen [Fla12, Seite 347]. Eine andere Bewertungsmetrik, die die zwei Kennzahlen Sensitivität und Spezifität vereint, sind sogenannte *Receiver Operating Characteristic* (ROC) Diagramme, die beide Werte zusammen darstellen. Dabei wird die Richtig-Positiv-Rate als eine Funktion der Fehlalarmrate (der Umkehrung der Spezifität) abgebildet [Axe99]. Auf der Grundlage der Einführung in die Analyse mittels der ROC im Bereich des *Machine Learning*s von Fawcett [Faw06], werden in den nachfolgenden Absätzen einige hier relevante Fakten wiedergegeben.

Die beiden Punkte $(0;0)$ und $(1;1)$ sind die Extremfälle bei denen ein Klassifikator jeweils alle Daten als negativ bzw. alle Daten als positiv einstuft. Im ersten Fall werden so zwar keine Fehlalarme ausgelöst, aber auch keine Angriffe erkannt. Umgekehrt wird im zweiten Fall alles als Angriff eingestuft, weshalb zwar alle Angriffe erkannt werden, aber auf Kosten einer maximal hohen Fehlalarmrate. Der Punkt $(0;1)$ entspricht einer perfekten Klassifikation, es werden alle Angriffe erkannt und dabei nicht ein Fehlalarm ausgelöst. Punkte die auf der Diagonalen liegen repräsentieren IDS die versuchen die richtige Klasse zufällig zu erraten. Wenn bei jedem Ereignis mit einem fairen Münzwurf zufällig entschieden würde, ob es sich um einen Angriff oder nicht handelt, würde die Hälfte der Angriffe erkannt werden, aber auch die Hälfte der Nicht-Angriffe würde als Fehlalarme durchgehen, was dem Punkt $(0,5;0,5)$ entspräche. Wird

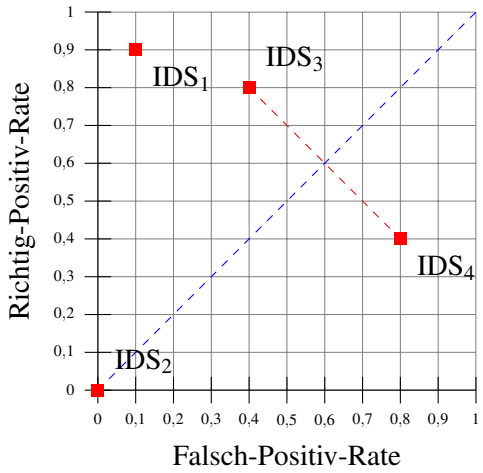

Abbildung 2.6: Ein ROC Diagramm, dass IDS_1 und IDS_2 aus den vorherigen Beispielen und zwei weitere IDS, die praktisch die selben Raten haben (wenn IDS_4 negiert würde), zeigt.

die Wahrscheinlichkeit, mit der das System auf einen Angriff tippt erhöht oder verringert, so ändert sich auch die Fehlalarmrate proportional und der Punkt wird nur auf der Diagonalen verschoben. Eine weitere Eigenschaft der Diagonalen ist, dass Punkte die unterhalb ihr liegen, und zunächst schlechter als ein zufälliger Klassifikator scheinen, an ihr gespiegelt werden können, indem statt der ursprünglichen Entscheidung immer das Gegenteil gewählt wird. Ein Beispiel dafür sind IDS_3 und IDS_4 in Abbildung 2.6, die, wenn die Entscheidungen von IDS_4 negiert werden, beide auf den Punkt $(0,4; 0,8)$ fallen. Als Faustregel kann also davon ausgegangen werden, dass ein Punkt, der weit im linken oberen Bereich liegt, ein gutes IDS repräsentiert.

Neben Klassifikatoren, die eine konkrete Klasse als Ausgabe liefern und sich so durch eine einzige Wahrheitsmatrix darstellen lassen, gibt es auch Ansätze, die jeder Klasse einen Wert zuordnen, der angibt, wie sicher sich der Klassifikator ist, dass es sich bei der aktuellen Instanz um diese Klasse handelt [Faw06]. Durch die Vorgabe eines Schwellenwerts, der die entsprechende Entscheidung angibt, kann daraus wieder ein diskretes Ergebnis abgeleitet werden. Beispielsweise könnte bei einem *naive Bayes*-Klassifikator (siehe Abschnitt 2.2.3) anstelle der MAP-Entscheidungsregel festgelegt werden, dass erst ab einer Schwelle von 50%, 60% oder 70% ein Angriff als solcher gemeldet wird. Jeder dieser Schwellenwerte bringt dann eine eigene Wahrheitsmatrix mit sich und entsprechend auch jeweils einen zusätzlichen Eintrag im ROC-Diagramm. Mit einem sogenannten *scoring* oder *ranking* Klassifikator ist es also möglich, bei einem IDS den Kompromiss zwischen Richtig- und Falsch-Positiv-Rate so zu konfigurieren,

wie es die Situtation erfordert. Wenn beispielsweise ein automatisches System Gegenmaßnahmen einleitet, kann es, je nach Art der Maßnahmen, akzeptabel sein, wenn auch auf Fehlalarme reagiert wird, solange möglichst viele Angriffe abgedeckt werden. Auf der anderen Seite könnte eine Gegenmaßnahme aber auch so verheerend sein, dass das System sehr sicher sein muss, dass es sich wirklich um einen Angriff handelt und ggf. einige unerkannte Angriffe in Kauf nehmen muss, um die Fehlalarmrate möglichst gering zu halten. Wie Abbildung 2.7 zeigt, können die

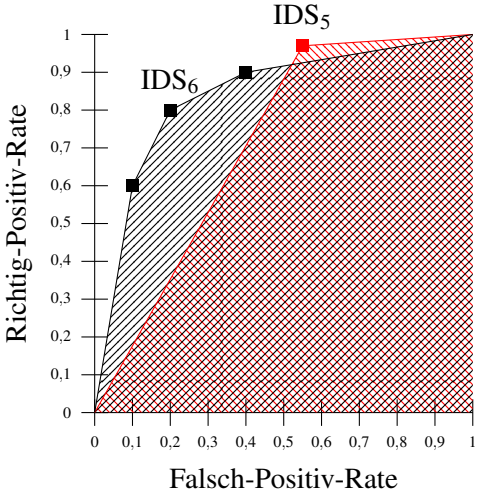

Abbildung 2.7: Ein ROC Diagramm, das zwei IDS mit den dazugehörigen ROC Kurven und AUC zeigt. IDS_5 ist ein diskreter Klassifikator, während IDS_6 mehrere Schwellenwerte unterstützt.

unterschiedlichen Punkte auch zu einer ROC-Kurve verbunden werden, die an den Enden die Punkte $(0;0)$ und $(1;1)$ berührt. Neben diesen konkreten Punkten ist es auch möglich Werte zu verwenden, die zwar auf der Kurve (bzw. der Geraden, die zwei Punkte miteinander verbindet) liegen, aber keiner bestimmten Konfiguration des IDS direkt entsprechen. Dazu kann ausgenutzt werden, dass der neue Punkt auf der Verbindungsgeraden der beiden bekannten Punkte liegt und diese in zwei Hälften teilt. Auf dieser Basis kann bestimmt werden, welcher Klassifikator bzw. welche Konfiguration welchen Einfluss auf den neuen Punkt haben muss. Entsprechend oft wird mal der eine, mal der andere Klassifikator verwendet, um so den gewünschten Punkt zu simulieren [Faw06].

Beispiel 2.6 Abbildung 2.8 skizziert, wie mit diesem Ansatz aus den zwei Klassifikatoren IDS_7 und IDS_8 ein neuer Klassifikator IDS_9 gebildet werden kann. Dazu wird in $\frac{a}{a+b}$ Fällen die Entscheidung von IDS_7 gewählt und in $\frac{b}{a+b}$ Fällen die Entscheidung von IDS_8.

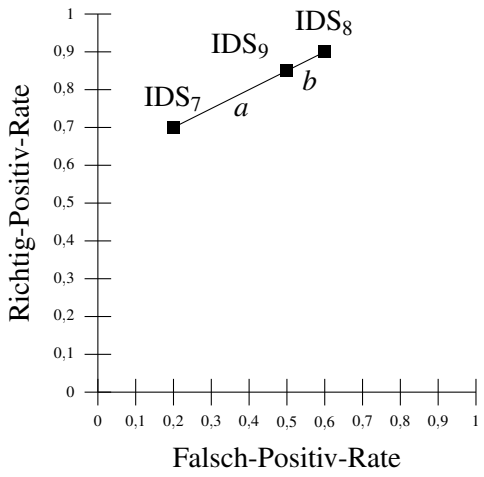

Abbildung 2.8: Wenn im Verhältnis $a : b$ immer je eine Entscheidungen von IDS_7 oder IDS_8 ausgewählt wird, ergibt sich IDS_9.

Auch wenn zwei IDS mit Hilfe von ROC-Kurven verglichen werden können, wäre es praktisch, jeweils nur einen einzigen Wert zu haben. Dazu lässt sich die Fläche unter der ROC-Kurve (AUC)[9] verwenden. Der Wert der AUC kann zwischen 0 und 1 liegen. Da ein IDS, das zufällig rät, bereits eine Fläche von $0,5$ hat, sollten brauchbare Werte zwischen $0,5$ und 1 liegen. Es ist durchaus möglich, dass ein Klassifikator, der eine größere AUC hat, in bestimmten Regionen schlechter abschneidet als ein Klassifikator, der eine geringere Fläche aufweist, wie in Abbildung 2.7 gezeigt. In der Praxis liefert die AUC aber gute Ergebnisse, wenn Klassifikatoren verglichen werden sollen [Faw06]. Ein weiteres Problem beim Vergleichen von IDS mittels AUC ist, dass es in der Praxis nicht relevant ist, wie gut das System mit allen möglichen Konfigurationen abschneidet, sondern lediglich, wie gut es mit der besten Konfiguration in der aktuellen Umgebung arbeitet [GFD+06].

Ein grundsätzliches Problem, dass bei der Bewertung von IDS berücksichtigt werden muss, ist, dass der Anteil an normalen Ereignissen wesentlich höher ist, als tatsächliche Angriffe. Axelsson [Axe99] macht deutlich, dass eine hohe Sensitivität alleine nicht ausreicht und die Spezifität der eigentlich limitierende Faktor ist. Auch wenn eine Sensitivität von 100% erreicht würde, muss dennoch eine sehr hohe Spezifität gegeben sein, um die Zahl der Fehlalarme niedrig genug zu halten, wenn die Meldungen manuell ausgewertet werden sollen. In seinem Beispiel schlägt Axelsson eine Fehlalarmrate von höchstens $1 \cdot 10^{-5}$ vor.

[9] Der Begriff AUC leitet sich aus der englischen Bezeichnung *area under the (ROC) curve* ab.

3 Simulationsumgebung

Kolonko [Kol08, Seite 1] beschreibt eine Simulation als „vereinfachtes Nachbilden einer komplexen Situation oder eines komplexen Systems, um Berechnungen oder Untersuchungen vorzunehmen“. Das Verwenden einer Simulation bietet sich besonders in Fällen an, in denen das reale System nicht oder nur schwer zu untersuchen ist. Um die Leistungsfähigkeit eines IDS zu untersuchen wäre es beispielsweise möglich, Angriffe auf ein laufendes Produktivsystem durchzuführen, allerdings würden erfolgreiche Angriffe entsprechend das echte System beschädigen. Aus diesem Grund werden im Bereich der Angriffserkennung Simulationen häufig zur Untersuchung von IDS eingesetzt. Dabei wird meist versucht, den grundlegenden Datenverkehr bzw. die Aktivitäten eines Anwenders nachzubilden. Das beinhaltet sowohl den meist größeren Anteil an normalen Aktivitäten, auch als *background traffic* oder Rauschen bezeichnet, aber auch Angriffe. Puketza et al. [PZC+96] verwenden Skripte, um Benutzereingaben auf einer Kommandozeile zu simulieren. Um auch Benutzereingaben durch eine graphische Oberfläche generieren zu können, kann der Ansatz von Garg et al. [GVUK06] verwendet werden. Kayacık und Zincir-Heywood [KZH05] simulieren das Verhalten von Anwendern auf Basis eines Modells, das normales Verhalten vorgibt. Anstatt solches Rauschen nur zu simulieren, schlagen Wan und Yang [WY01] die Möglichkeit vor, echten Datenverkehr zu verwenden. Dieser kann entweder extra innerhalb des Testnetzes produziert werden oder auch im echten Netz aufgenommen und in das Testnetz eingespielt werden. Oft wird zum Testen und Vergleichen von IDS auf den Datensatz des KDD'99 Cup[1] zurückgegriffen. Dieser Datensatz wurde künstlich erzeugt und enthält drei Wochen an Trainingsdaten und zwei Wochen an Testdaten inklusive Rauschen und, im Falle der Testdaten, auch Angriffe, die nicht in den Trainingsdaten vorkommen [LHF+00]. Anstatt künstlichen Datenverkehr zu erzeugen, simulieren Garg et al. [GUCK03] verschiedene Sensorenausgaben innerhalb eines Hosts, die an ein entsprechendes IDS weitergeleitet werden.

In der hier entwickelten Simulationsumgebung werden direkt die Ausgaben der Sensoren simuliert und anschließend einer Sensorfusionsmethode zur Entscheidungsfindung vorgelegt. Das Grundkonzept der Simulation wird im nächsten Abschnitt vorgestellt. In Abschnitt 3.2 wird er-

[1] Online verfügbar unter `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`. [Stand: 18.03.2014]

klärt, wie die abstrakten Sensorenausgaben generiert werden und in Abschnitt 3.3 wird auf die Konfigurationsmöglichkeit der Sensoren bzw. der Simulation selbst eingegangen. Abschließend wird in Abschnitt 3.4 die Implementierung der Simulationsumgebung beschrieben.

3.1 Die Simulation im Überblick

Die im Rahmen dieser Arbeit entwickelte Simulation bietet eine Plattform, mit der Fusionsmethoden in unterschiedlichen Sensorenanordnungen untersucht werden können. Da der Fokus auf den Fusionsmethoden liegt, werden direkt abstrakte Sensoren bzw. deren Ausgaben simuliert und keine wirklichen Angriffe, die erst durch die Sensoren verarbeitet werden müssten. Die Ergebnisse der simulierten Sensoren werden anschließend an die Fusionsmethode übergeben, die ihrerseits auf Basis dieser Daten ein Ergebnis liefert. Diese Schritte werden über den Verlauf der Simulation vielfach wiederholt, bis abschließend die gesammelten Ergebnisse in einer Datenbank für die spätere Auswertung gesichert werden.

Die Ausgabe der Sensoren und Fusionsmethoden ist dabei so ausgelegt, dass sie als Ergebnis nur die eine Klasse liefern, die der Entscheidung des jeweiligen Systems entspricht. Im einfachsten Fall beschränkt sich das Ergebnis auf eine der beiden Klassen Angriff und Nicht-Angriff. Um Fusionsmethoden zu untersuchen, die mit mehreren Angriffsklassen umgehen können, werden weiterhin die vier Klassen des KDD'99 Datensatzes, *DoS*, *R2L*, *U2R* und *probing*, unterstützt. Das bedeutet ebenfalls, dass keine unbekannten Angriffe direkt als Klasse modelliert, sondern höchstens über Manipulation der Erkennungsraten nachgebildet werden können.

Da die Fusionsmethoden ausschließlich über die Sensorenausgaben mit der Simulation verbunden sind, kann über die Konfiguration der Sensoreneigenschaften Einfluss auf die Simulation genommen werden. Dabei können die Erkennungsraten der einzelnen Sensoren für jede Klasse angepasst und auch die Korrelation der Sensoren untereinander angegeben werden. Diese Werte sind für einen Simulationslauf fest durch die Konfiguration vorgegeben und können nicht innerhalb der Simulation beeinflusst werden.

Die Fusionsmethoden hingegen haben die Möglichkeit, zunächst eine Trainingsphase zu durchlaufen, in der, neben den simulierten Sensorenausgaben, auch die eigentlich zu meldende Klasse zur Verfügung steht. Weiterhin ist in der Trainingsphase der komplette Trainingsdatensatz einsehbar, während in der Simulationsphase die Fälle immer einzeln vorgelegt werden. Somit ist es der Fusionsmethode freigestellt, wie sie die Trainingsdaten nutzt. Um im Training unterschiedliche Daten als in der späteren Simulation verwenden zu können, sind alle Parameter, d.h. die

Sensoreneigenschaften und der Inhalt der Datensätze, jeweils für die Simulation und die Trainingsphase konfigurierbar. Für die Datensätze selbst kann nur die Anzahl der zu beinhaltenden Klassen konfiguriert werden, die Reihenfolge wird zufällig gewählt.

3.2 Simulieren von Sensorenausgaben

Um möglichst viele verschiedene Situationen untersuchen zu können, werden die Sensorenausgaben künstlich generiert. Neben dem geringeren Aufwand, da weder echte Daten als Eingabe für die Sensoren benötigt werden, noch die Sensoren selbst betrieben werden müssen, können so auch Sensoren simuliert werden, die in der Praxis nicht vorhanden sind. Dabei macht es für die Simulation selbst keinen Unterschied, ob das vorgegebene Verhalten der Sensoren auf Expertenwissen basiert, durch Versuche ermittelte Erkennungsraten verwendet oder fiktive Daten angegeben werden. Allerdings sollte bei der anschließenden Auswertung berücksichtigt werden, auf welcher Grundlage diese Ergebnisse zu Stande gekommen sind. Die Sensoren selbst werden primär über ihre Erkennungsraten definiert, die zusammen mit Zufallszahlen dazu verwendet werden, entsprechende Sensorenentscheidungen zu simulieren.

3.2.1 Pseudozufallszahlen

Damit Experimente, ggf. mit geänderten Parametern, wiederholt werden können, werden keine echten, sondern sogenannte Pseudozufallszahlen verwendet. Dieser Begriff bezeichnet deterministische Folgen von Zahlen, die den Eindruck erwecken, zufällig zu sein [Knu02, Seite 4]. Knuth [Knu02, Kapitel 3.2] nennt, neben anderen, den linearen Kongruenzgenerator, als eine Methode, um gleichverteilte reelle Pseudozufallszahlen zu erzeugen. Diese Methode wird auch in der Java-Klasse `java.util.Random` für die Generierung von Pseudozufallszahlen verwendet [Jav], die wiederum als Grundlage für die Zufallszahlen der Simulation verwendet wird. Um die generierten Folgen in späteren Experimenten reproduzieren zu können, wird der initiale Startwert des Generators (auch *seed* genannt) als Parameter angegeben.

3.2.2 Sensorenausgaben generieren

Auf Basis solcher Zufallszahlen können Sensoren mit vorgegebenen Erkennungsraten simuliert werden. Dafür wird, wann immer ein Sensor eine Entscheidung treffen muss, eine neue Zufallszahl generiert und auf eine entsprechende Entscheidung abgebildet. Doch bevor dies geschieht,

muss die Simulation zunächst entscheiden, welche Angriffsklasse den Sensoren vorgelegt werden soll. Die gewünschte Verteilung der Klassen ist über die Konfiguration vorgegeben und wird ebenfalls mit Hilfe von Zufallszahlen realisiert. Im Unterschied zu den Sensoren ist die Verteilung durch die Anzahl der zu simulierenden Instanzen jeder Klasse beschrieben und nicht durch einen prozentualen Anteil. Das bedeutet, dass sobald die maximale Anzahl an Instanzen einer bestimmten Klasse erreicht wurde, diese nicht mehr als Ergebnis ausgewählt werden kann. Trotzdem kann ein Ansatz verwendet werden, der dem Erzeugen von bedingt unabhängigen Sensorenausgaben stark ähnelt, weshalb das initiale Auswählen einer Klasse nicht weiter im Detail eingegangen wird.

3.2.2.1 Bedingt unabhängige Sensoren

Nachdem die Simulation für die aktuell zu simulierende Instanz eine Klasse ausgewählt hat, kann für jeden Sensor die entsprechende Verteilung für diese Klasse aus der Konfiguration ermittelt werden. Da die Wahrscheinlichkeitsverteilung der Sensorenausgabe insgesamt 100% ergibt, kann sie ohne großen Aufwand auf das Intervall $[0;1)$ abgebildet werden. Dazu wird jeder Klasse ein Teilintervall zugeordnet, dessen Größe der Wahrscheinlichkeit entspricht, dass der Sensor diese Klasse als Ergebnis wählt. (Die Reihenfolge spielt dabei keine Rolle und kann beliebig gewählt werden.) Ein Beispiel dazu ist in Abbildung 3.1 gegeben. Dabei ist zu beachten,

Abbildung 3.1: Die Erkennungsraten eines fiktiven Sensors für die Klassen $normal = 0,1$, $DoS = 0,7$, $R2L = 0,05$, $U2R = 0,05$ und $probing = 0,1$, für den Fall, dass die Klasse *DoS* vorliegt, werden auf das Intervall $[0;1)$ abgebildet. Für den Zufallswert $0,18947707788171864$ wird die Klasse *DoS* gewählt.

dass die einzelnen Intervalle alle rechtsoffen sind. Das hat den Grund, dass bei der Auswahl einer Klasse durch eine Zufallszahl die Intervallgrenzen eindeutig einer Klasse zugeordnet werden müssen. Obwohl die 1 so nicht Teil des Intervalls ist, stellt dies kein Problem dar. Da die von Java erzeugten Zufallszahlen in $[0;1)$ liegen und annäherend gleichverteilt sind [Jav], kann davon ausgegangen werden, dass jeweils 50% der generieren Zufallswerte in $[0;0,5)$ bzw. $[0.5;1)$ liegen. Entsprechend fallen $x\%$ der generierten Zufallszahlen in ein Intervall $[a;b)$ der Größe

$x = b - a$. Nachdem eine Zufallszahl generiert worden ist, wird das Intervall gewählt, in dem die Zahl liegt und so eine Klasse bzw. das Ergebnis des Sensors bestimmt (siehe Abbildung 3.1).

Knuth [Knu02, Seite 120] beschreibt diesen Ansatz formal im Zusammenhang mit dem zufälligen Auswählen von Elementen aus einer endlichen Menge, wobei die einzelnen Elemente mit unterschiedlich gewichteten Wahrscheinlichkeiten ausgewählt werden sollen. Dabei ist X eine Zufallsvariable und p_i gibt die Wahrscheinlichkeit an, mit der X den Wert (bzw. hier die Klasse) x_i annimmt. Um nun einen zufälligen Wert U aus einer stetigen Gleichverteilung zwischen 0 und 1 auf die k Elemente aus X abzubilden, wird folgende Formel angegeben

$$X = \begin{cases} x_1, & \text{wenn } 0 \leq U < p_1, \\ x_2, & \text{wenn } p_1 \leq U < p_1 + p_2, \\ \vdots & \\ x_k, & \text{wenn } p_1 + p_2 + \cdots + p_{k-1} \leq U < 1 (= p_1 + p_2 + \cdots + p_k). \end{cases}$$

Da die verwendete Java-Funktion annähernd gleichverteilte Zufallszahlen liefert und für jeden Sensor eine eigene Zufallszahl generiert wird, werden so bedingt unabhängige Sensorenausgaben erzeugt.

3.2.2.2 Korrelierte Sensoren

Da in der Praxis die Sensoren nicht zwingend bedingt unabhängig sind, soll die Simulationsumgebung auch in der Lage sein, korrelierte Sensoren zu modellieren. Als Grundlage dafür kann in der Konfiguration eine Korrelationsmatrix angegeben werden, die beschreibt, wie die einzelnen Sensoren miteinander zusammenhängen. Scheuer und Stoller [SS62] beschreiben eine Methode, die auf Basis einer solchen Matrix C aus einem Zufallsvektor X einen korrelierten Zufallsvektor Y erzeugt. Dazu wird zunächst die untere Dreiecksmatrix L der Korrelationsmatrix $C = LL^T$ berechnet. Anschließend kann der korrelierte Zufallsvektor Y über die Formel $Y = LX$ bestimmt werden. Zur Bestimmung einer unteren Dreiecksmatrix wird hier der Cholesky–Banachiewicz Algorithmus [Loc93, Seite 248] verwendet.

Der benötigte Zufallsvektor ergibt sich aus den Zufallszahlen der einzelnen Sensoren, die für die aktuelle Instanz generiert wurden. Allerdings ist eine Anforderung an den Zufallsvektor, dass die einzelnen Variablen standardnormalverteilt sein müssen. Entsprechend wird für diesen Ansatz eine Java-Funktion verwendet, die annähernd standardnormalverteilte Zufallswerte

liefert. Die Einträge des Ergebnisvektors können anschließend wieder den einzelnen Sensoren zugeordnet werden.

Aus diesen korrelierten Zufallszahlen muss abschließend nun eine Entscheidung für jeden Sensor ermittelt werden. Um den Intervall-Ansatz der bedingt unabhängigen Sensorenausgaben (Abschnitt 3.2.2.1) wiederverwenden zu können, müssen diese Zufallszahlen entsprechend umgerechnet werden, da sie normal- und nicht gleichverteilt sind. Dazu kann die Verteilungsfunktion $F(x)$ verwendet werden, die die Wahrscheinlichkeit angibt, dass eine Zufallsvariable X einen Wert kleiner oder gleich x annimmt. Somit lässt sich jede dieser Zufallszahlen auf einen entsprechenden Wert zwischen 0 und 1 abbilden, der wiederum einem Intervall bzw. einer Klasse entspricht. Da sich die Verteilungsfunktion der Normalverteilung nicht direkt bestimmen lässt, wird üblicher Weise auf eine Hilfstabelle mit vorberechneten Wertepaaren zurückgegriffen.

Beispiel 3.1 Für eine Simulation mit drei Sensoren ist durch die Konfiguration folgende Korrelationsmatrix vorgegeben

$$C = \begin{pmatrix} 1 & 0,8 & 0,6 \\ 0,8 & 1 & 0,9 \\ 0,6 & 0,9 & 1 \end{pmatrix}.$$

Als untere Dreiecksmatrix nach der Cholesky Zerlegung ergibt sich daraus (gerundet)

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0,8 & 0,600 & 0 \\ 0,6 & 0,700 & 0,387 \end{pmatrix}.$$

Für die aktuelle Instanz wurden die folgenden Zufallszahlen für die drei Sensoren generiert

$$X = \begin{pmatrix} 1,0054995295180076 \\ 0,6873754979465944 \\ -0,05040924886235369 \end{pmatrix}.$$

Multipliziert mit der unteren Dreiecksmatrix ergibt sich ein Vektor mit entsprechend korrelierten Werten.

$$Y = \begin{pmatrix} 1 & 0 & 0 \\ 0,8 & 0,600 & 0 \\ 0,6 & 0,700 & 0,387 \end{pmatrix} \cdot \begin{pmatrix} 1,0054995295180076 \\ 0,6873754979465944 \\ -0,05040924886235369 \end{pmatrix} = \begin{pmatrix} 1,0054995295180076 \\ 1,2168249223823626 \\ 1,0649391481395485 \end{pmatrix}$$

Als finaler Schritt müssen die neuen Werte noch auf jeweils eine Entscheidung abgebildet werden. Dazu wird mit der Verteilungsfunktion die Wahrscheinlichkeit bestimmt, dass ein Sensor einen Wert animmt, der kleiner oder gleich dem nun korrelierten Wert ist. Für den ersten Sensor ist also die Wahrscheinlichkeit gesucht, mit der er einen Wert $\leq 1,0054995295180076$ animmt. Anschaulich bedeutet das, dass der Flächeninhalt unter der Dichtefunktion der Standardnormalverteilung im Intervall $[-\infty;x]$ gesucht ist. Für den ersten Sensor ergibt sich $\approx 0,84$, wie in Abbildung 3.2 gezeigt. Dieser Wert kann nun analog zu Abbildung 3.1 auf eine Klasse bzw.

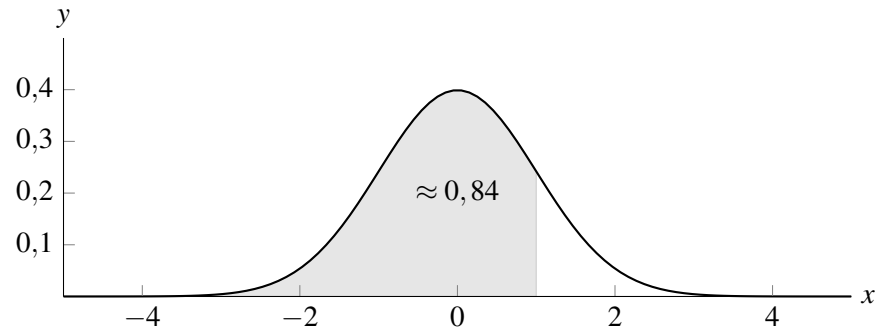

Abbildung 3.2: Dichtefunktion der Standardnormalverteilung. Der Flächeninhalt des Intervalls $[-\infty;1]$ ist hervorgehoben.

Entscheidung abgebildet werden. Mit den anderen beiden Sensoren wird entsprechend verfahren.

3.3 Konfiguration der Simulation

Die wichtigsten Einflussfaktoren für die Simulation bilden die Sensoren und die Aufteilung der zu simulierenden Klassen. Für die einzelnen Klassen wird jeweils die Anzahl der zu simulierenden Instanzen angegeben (`simulatedTrafficAmounts`). Die Erkennungsraten der Sensoren hingegen werden als Wahrscheinlichkeitsverteilungen angegeben; eine detailliertere Beschreibung der Konfigurationsmöglichkeit der Sensoren ist in Abschnitt 3.3.1 zu finden. Die entsprechenden Werte können zusätzlich auch für die Trainingsphase angegeben werden, sofern im Training andere Einstellung gelten sollten. Neben diesen direkten Einflussfaktoren können auch die Startwerte für die einzelnen Zufallszahlengeneratoren fest vorgegeben werden, um bestimmte Szenarios wiederholen zu können. Ausgelassene Startwerte werden automatisch aus dem primären Startwert generiert (`primarySeed`). Die Fusionsmethode bzw. der Name der ent-

sprechenden Java-Klasse muss ebenfalls angegeben werden (`fusionMethod`). Optional können für die Fusionsmethode extra Parameter angegeben werden, um die Methode selbst zu konfigurieren (`fusionConfiguration`). Damit die Ergebnisse gesichert und ausgewertet werden können, muss eine Datenbankanbindung konfiguriert werden (`database`). Die benötigten Daten können sich von Datenbank zu Datenbank unterscheiden. Weiterhin können Metainformationen, wie eine Bezeichnung (`label`) oder eine kurze Beschreibung des Szenarios (`description`), angegeben werden, um den Simulationslauf später für die Auswertung identifizieren zu können. Es ist auch möglich, mehrere Simulationsläufe auf einmal zu konfigurieren (`numberOfRuns`). Dabei werden, mit Ausnahme der Startwerte für die Zufallszahlengenerierung, alle Einstellungen beibehalten. Die Startwerte werden automatisch so für jeden weiteren Durchlauf bestimmt, dass sie reproduzierbar sind. Das bedeutet unter anderem, dass es möglich ist, eine bereits durchgeführte Versuchsreihe zu wiederholen oder zusätzliche Läufe anzuhängen.

Als Format für die Konfiguration wurde JSON[2] gewählt, da es für Menschen leicht zu lesen und schreiben ist und sich alle relevanten Einstellungsmöglichkeiten abbilden lassen. Eine Beispielkonfiguration ist in Listing 3.1 zu finden.

3.3.1 Konfiguration der Sensoren

Der wichtigste Aspekt der Sensorenkonfiguration sind die Erkennungsraten. Diese werden als Wahrscheinlichkeitsverteilungen für jeden Sensor separat angegeben und zusätzlich für jede Klasse unterschieden. Die Unterscheidung nach Klassen ist wichtig, wenn KDD'99 Klassen verwendet werden, da die Wahrscheinlichkeitsverteilungen in den meisten Fällen nicht gleich sind, wie Tabelle 3.1 zeigt. Eine Zeile dieser Tabelle gibt die gewünschten Erkennungsraten für

	Normal	U2R	R2L	DoS	probing
Normal	0,9	0,01	0,01	0,04	0,04
U2R	0,05	0,9	0,02	0,02	0,01
R2L	0,05	0	0,95	0	0
DoS	0,1	0	0	0,8	0,1
probing	0,1	0	0	0,1	0,8

Tabelle 3.1: Beispielkonfiguration der Erkennungsraten eines Sensors für die KDD'99 Klassen.

einen Sensor vor, wenn die entsprechende Klasse simuliert wird. Wenn in der Simulation nur die zwei Klassen Angriff und Nicht-Angriff untersucht werden, würde es ausreichen, jeweils

[2] JavaScript Object Notation, `http://www.json.org` [Stand: 18.03.2014]

nur einen Wert pro Zeile anzugeben. Da die Summe einer Zeile 1 bzw. 100% ergeben muss, ließe sich der zweite Wert so immer berechnen. Damit die Konfiguration einheitlich ist, werden aber auch diese Werte vollständig angegeben, wie in Tabelle 3.2 gezeigt. Wie diese Daten in

	Normal	Angriff
Normal	0,9	0,1
Angriff	0,2	0,8

Tabelle 3.2: Beispielkonfiguration der Erkennungsraten eines Sensors für die Klassen Angriff und Nicht-Angriff.

der Konfiguration repräsentiert werden, ist in der Beispielkonfiguration in Listing 3.1 zu sehen, wobei ein Eintrag immer einer Zeile der Tabelle entspricht.

Neben den Erkennungsraten beinhaltet die Konfiguration der Sensoren auch die Art, wie die Sensorenausgaben generiert werden. Dabei wird zwischen bedingt unabhängigen und korrelierten Sensoren unterschieden und ein entsprechender Zufallszahlengenerator ausgewählt. Im Falle von korrelierten Sensoren, muss zusätzlich noch eine Korrelationsmatrix angegeben werden, die beschreibt, wie genau sich die Sensoren gegenseitig beeinflussen. Die einzelnen Einträge der Korrelationsmatrix sind Korrelationskoeffizienten, die jeweils angeben, ob ein Sensorenpaar positiv (ein Wert größer als 0), negativ (ein Wert kleiner als 0) oder garnicht (bei einem Wert von 0) zusammenhängt.

3.4 Implementierung der Simulationsumgebung

Den Kern der Simulationsumgebung bilden die drei Komponenten Simulation, Datenhaltung und Auswertung. Die Simulationskomponente ist für die Simulation selbst verantwortlich. Das bedeutet, diese Komponente übernimmt die Zufallszahlengenerierung, das Erzeugen von Sensorenausgaben für die Trainings- und Simulationsdatensätze und die Ausführung der Sensorfusion. Die Simulationsergebnisse werden in Form von Wahrheitsmatrizen in einer Datenbank gesichert, um anschließend der Auswertungskomponente zur Verfügung zu stehen. Somit dient die Datenhaltung als Bindeglied zwischen den anderen beiden Komponenten. Die Auswertungskomponente ist letztlich dafür verantwortlich, die gesammelten Daten mit Hilfe von Metriken zur manuellen Analyse aufzubereiten. Eine weitere wichtige Komponente ist die Benutzerschnittstelle. Die Simulation (und die Auswertung) ist über die Kommandozeile zu bedienen, wobei alle wichtigen Einstellungen über die entsprechenden Konfigurationsdateien vorgenommen wer-

```
{
  "label" : "Majority Vote Example",
  "description" : "An example scenario including three sensors.",
  "primarySeed" : 123456789,
  "fusionMethod" : "da.simulation.core.Voting",
  "fusionConfiguration" : { "type" : "MAJ" },
  "eventGenerator" : "da.simulation.core.CorrelatedEventGenerator",
  "database" : { "type" : "sqlite", "filename" : "basedb.sqlite" },
  "simulatedTrafficAmounts" : { "normal" : 1000000, "attack" : 10000 },
  "numberOfRuns" : 2,
  "correlationMatrix" : {
    "sensor1" : {
      "sensor2" : 0.8,
      "sensor3" : 0.6
    },
    "sensor2" : {
      "sensor1" : 0.8,
      "sensor3" : 0.9
    },
    "sensor3" : {
      "sensor1" : 0.6,
      "sensor2" : 0.9
    }
  },
  "sensors" : [
    {
      "label" : "sensor1",
      "simulationDetectionRates" : {
        "normal" : { "normal" : 0.7, "attack" : 0.3 },
        "attack" : { "normal" : 0.1, "attack" : 0.9 }
      }
    }, {
      "label" : "sensor2",
      "simulationDetectionRates" : {
        "normal" : { "normal" : 0.8, "attack" : 0.2 },
        "attack" : { "normal" : 0.2, "attack" : 0.8 }
      }
    }, {
      "label" : "sensor3",
      "simulationDetectionRates" : {
        "normal" : { "normal" : 0.9, "attack" : 0.1 },
        "attack" : { "normal" : 0.3, "attack" : 0.7 }
      }
    }
  ]
}
```

Listing 3.1: Beispielkonfiguration für ein Abstimmungsverfahren, dass die Ausgaben von drei korrelierten Sensoren kombiniert.

den (siehe Abschnitt 3.3). Eine abstrakte Übersicht über die einzelnen Komponenten und deren Zusammenhang ist in Abbildung 3.3 gegeben.

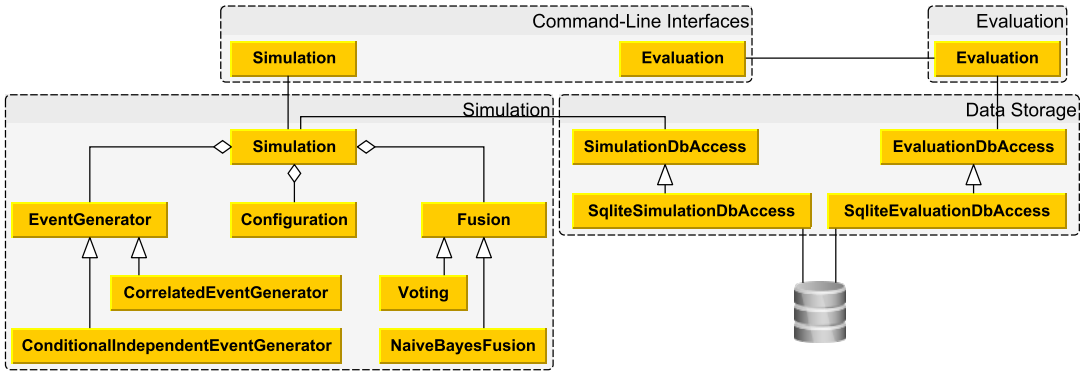

Abbildung 3.3: Diagramm der wichtigsten Klassen der Simulationsumgebung.

Um Plattformunabhängigkeit zu gewährleisten, wurde die Simulationsumgebung in Java umgesetzt. Dabei wurden neben der Standardbibliothek von Java 7 zwei weitere Bibliotheken verwendet. Der Zugriff auf SQLite-Datenbanken wird durch *sqlite4java*[3] ermöglicht und *Jackson*[4] unterstützt das Verarbeiten der Konfigurationsdateien im JSON Format. SQLite wurde als Datenbank gewählt, da so der zusätzliche Aufwand für das Installieren bzw. Einrichten eines Datenbankserver für den Anwender wegfällt. Der Hauptgrund JSON für die Konfigurationsdateien zu verwenden ist, dass es für Menschen leicht zu lesen und schreiben ist.

3.4.1 Fusionsmethoden

Um das Hinzufügen neuer Fusionsmethoden möglichst einfach zu gestalten, werden die entsprechenden Klassen via Reflexion über ihren Namen instanziiert Dafür ist es ausreichend, die Fusionsmethoden in den Java `classpath` Parameter mit aufzunehmen, anstatt das komplette Programm neu zu kompilieren. Die einzige Einschränkung besteht darin, dass alle Fusionsmethoden zunächst von der abstrakten Klasse `da.simulation.core.Fusion` abgeleitet werden müssen, damit die Simulation über eine einheitliche Schnittstelle mit den Fusionsmethoden interagieren kann.

Die Simulationsumgebung beinhaltet bereits Abstimmungsverfahren und *naive Bayes*-basierte Fusionsmethoden, die jeweils zwischen den beiden Klassen Angriff und Nicht-Angriff unter-

[3] `http://code.google.com/p/sqlite4java/` [Stand: 18.03.2014], Apache License 2.0
[4] `https://github.com/FasterXML/jackson` [Stand: 18.03.2014], Apache License 2.0

scheiden. Beide Methoden unterstützen verschiedene Entscheidungsregeln, die über die Konfiguration vorgegeben werden können. In den folgenden beiden Abschnitten, 3.4.1.1 und 3.4.1.2 wird auf die Implementierung dieser beiden Fusionsmethoden eingegangen.

3.4.1.1 Abstimmungsverfahren

Es werden die drei Varianten *einstimmige Entscheidung* (AND), *mindestens einer* (OR) und *einfache Mehrheit* (MAJ) implementiert, wobei die letzte Variante dasselbe Ergebnis wie die *relative Mehrheit* liefert, da nur zwischen zwei Klassen unterschieden wird. Da diese Abstimmungsverfahren kein Training benötigen, kann direkt mit der Entscheidungsfindung auf Grundlage der Sensorenausgaben begonnen werden.

Da AND die Anforderung hat, dass alle Sensoren einen Angriff melden, ist es ausreichend zu überprüfen, ob mindestens ein Sensor keinen Angriff gemeldet hat:

```
if (sensorOutput.contains(AttackType.NORMAL)) {
  return AttackType.NORMAL;
} else {
  return AttackType.ATTACK;
}
```

Derselbe Fall gilt umgekehrt für OR. Hier kann eine Entscheidung getroffen werden, sobald klar ist, dass mindestens ein Sensor einen Angriff erkannt hat:

```
if (sensorOutput.contains(AttackType.ATTACK)) {
  return AttackType.ATTACK;
} else {
  return AttackType.NORMAL;
}
```

Für die Implementierung von MAJ wurde der MJRTY Algorithmus von Boyer und Moore [BM91] verwendet. Als Vorausetzung für diesen Algorithmus muss allerdings bekannt sein, oder zumindest angenommen werden, dass eine relative Mehrheit vorhanden ist. Ansonsten kann von dem Ergebnis nicht auf eine eindeutige Aussage geschlossen werden. Da im Falle von nur zwei Klassen aber entweder eine Mehrheit existiert oder Gleichstand herrscht, der separat behandelt werden kann, ist diese Vorausetzung erfüllt.

Um eine Entscheidung zu treffen, wird einmalig über die Ausgaben der Sensoren iteriert. Nebenbei wird ein Zähler mitgeführt, der angibt, mit wievielen „Stimmen“ die aktuelle Ent-

scheidung gegenüber der anderen führt. Wenn der Zähler auf 0 steht, wird die aktuelle Klasse als Kandidat gewählt und der Zähler auf 1 gesetzt. Ansonsten wird, wenn die aktuelle Klasse mit dem Kandidaten übereinstimmt, der Wert um 1 erhöht oder, wenn die Klasse nicht passt, der Wert um 1 reduziert. Am Ende wird eine Entscheidung entsprechend des Kandidaten getroffen, es sei denn der Zähler steht auf 0, was einen Gleichstand zwischen den beiden Klassen bedeutet. In diesem Fall wird ein Angriff gemeldet. Der Algorithmus wurde wie folgt implementiert:

```
AttackType candidate = null;
int count = 0;

for (AttackType attackType : sensorOutput) {
  if (count == 0) {
    candidate = attackType;
    ++count;
  } else if (attackType == candidate) {
    ++count;
  } else {
    --count;
  }
}

if (count == 0) {
  candidate = AttackType.ATTACK;
}

return candidate;
```

3.4.1.2 Naive Bayes-basierte Fusion

Die Implementierung der *naive Bayes*-basierten Fusion ist in die zwei Bereiche Training und Simulation unterteilt. In der Trainingsphase wird zunächst gesammelt, welche Klassen wie oft auftreten und wie die einzelnen Sensoren beim Erkennen der Trainingsdaten abschneiden. Dazu wird über alle verfügbaren Trainingsdaten iteriert und die entsprechenden Werte mitgezählt bzw. in Wahrheitsmatrizen eingetragen. Aus diesen Daten lassen sich später die benötigten empirischen Wahrscheinlichkeiten für die Entscheidungsfindung berechnen.

Wie schon bei den Abstimmungsverfahren, werden nur die zwei Klassen Angriff und Nicht-Angriff berücksichtigt, weshalb die beiden auf Quotient basierenden Entscheidungsregeln, *posterior odds* (PO) und *likelihood ratio test* (LRT), umgesetzt worden.

Da PO auf den Werten von LRT aufbaut, wird, unabhängig von der gewählten Variante, zunächst der Quotient für LRT berechnet. Dazu wird für jeden Sensor D_i, $i = 1, \ldots, n$ das Verhältnis $\frac{P(d_i|\text{Angriff})}{P(d_i|\text{Normal})}$ bestimmt, wobei d_i die aktuelle Ausgabe des Sensors ist. Schritt für Schritt wird so das Produkt

$$\prod_{i=1}^{n} \frac{P(d_i \mid \text{Angriff})}{P(d_i \mid \text{Normal})} = \frac{\prod_{i=1}^{n} P(d_i \mid \text{Angriff})}{\prod_{i=1}^{n} P(d_i \mid \text{Normal})}$$

berechnet. Da im Training die absolute Häufigkeit der einzelnen Ereignisse ermittelt wurde, muss diese in eine relative Häufigkeit umgerechnet werden, die wiederum der für die Rechnung benötigten Wahrscheinlichkeit entspricht. Für den Fall des Zählers ergibt sich die Wahrscheinlichkeit

$$P(d_i = \text{Angriff} \mid \text{Angriff}) = \frac{RP}{RP+FN} \quad \text{bzw.} \quad P(d_i = \text{Normal} \mid \text{Angriff}) = \frac{FN}{RP+FN},$$

abhängig davon, ob der Sensor gerade einen Angriff meldet oder nicht. Die gesuchte Wahrscheinlichkeit entspricht also der Sensitivität bzw. der Falsch-Negativ-Rate. Der Wert des Nenners ergibt sich entsprechend aus der Falsch-Positiv-Rate bzw. der Richtig-Negativ-Rate:

$$P(d_i = \text{Angriff} \mid \text{Normal}) = \frac{FP}{RN+FP} \quad \text{bzw.} \quad P(d_i = \text{Normal} \mid \text{Normal}) = \frac{RN}{RN+FP}.$$

Um den entsprechenden Codeauszug verständlicher zu gestalten, werden einige Funktionsaufrufe durch Pseudocode ersetzt:

```
double ratio = 1;
for (int i = 0; i < sensorOutput.size(); ++i) {
  ConfusionMatrix cm = Wahrheitsmatrix des aktuellen Sensors;
  // Zähler
  int correct = abhängig von der Sensorenausgabe RP oder FN
                aus der Wahrheitsmatrix lesen;
  int actual  = Anzahl der Angriffe (RP + FN) aus der
                Wahrheitsmatrix lesen;
  ratio *= (double) correct / actual;
```

```
    // Nenner
    correct = abhängig von der Sensorenausgabe RN oder FP
              aus der Wahrheitsmatrix lesen;
    actual  = Anzahl der Nicht-Angriffe (RN + FP) aus der
              Wahrheitsmatrix lesen;
    ratio /= (double) correct / actual;
}
```

Für LRT kann nun entschieden werden, ob es sich um einen Angriff (`ratio` ≥ 1) oder keinen Angriff (`ratio` < 1) handelt. Im Fall von PO muss noch das Verhältnis der A-priori-Wahrscheinlichkeiten $P(\text{Angriff})$ und $P(\text{Normal})$ berücksichtigt werden. Die entsprechenden absoluten Häufigkeiten, a für die Anzahl der Angriffe und n für die Anzahl der Nicht-Angriffe, wurden in der Trainingsphase gezählt. Damit ergibt sich für den Quotienten

$$\frac{P(\text{Angriff})}{P(\text{Normal})} = \frac{\frac{a}{a+n}}{\frac{n}{a+n}} = \frac{a}{a+n} \cdot \frac{a+n}{n} = \frac{a}{n}.$$

Für die Implementierung bedeutet das, dass der `ratio` Wert noch ein letztes Mal multipliziert werden muss:

```
ratio *= (double) (Anzahl der Angriffe)
                / (Anzahl der Nicht-Angriffe);
```

Abschließend kann auch PO eine Entscheidung treffen:

```
result = ratio >= 1 ? AttackType.ATTACK : AttackType.NORMAL;
```

Da es nur eine feste Anzahl an möglichen Kombinationen der Sensorenausgaben gibt, wird ein Cache verwendet, um bereits getroffene Entscheidungen nicht immer wieder neu berechnen zu müssen.

3.4.2 Auswertungskomponente

Die Auswertungskomponente arbeitet auf Basis der Wahrheitsmatrizen, die für jeden Simulationslauf in der Datenbank abgelegt werden. Damit ist es möglich, alle Bewertungsmetriken, die in Abschnitt 2.3 vorgestellt werden, zu bestimmen, da sie letzten Endes auf den Daten einer Wahrheitsmatrix aufbauen. Für den ausgewählten Simulationslauf werden die Korrektklassifikationsrate (`acc`), der positive Vorhersagewert (`PPV`), die Sensitivität (`recall`), die Falsch-Positiv-Rate (`FPR`), das F-Maß (`F-measure`) und die AUC (`AUC`) bestimmt sowie die zugrundeliegende

Wahrheitsmatrix selbst ausgegeben. Zu beachten ist, dass die ROC-Kurve auf deren Basis der AUC-Wert bestimmt wird, nur einen einzelnen Messpunkt neben $(0;0)$ und $(1;1)$ enthält, da ein Simulationslauf nur genau eine Richtig-Positiv- und eine Falsch-Positiv-Rate liefert (vgl. IDS_5 aus Abbildung 2.7 auf Seite 26). Es ist ebenfalls möglich, mehrere Simulationsläufe gemeinsam auswerten zu lassen. In diesem Fall werden die Metriken für jeden Lauf einzelnen berechnet und anschließend der Mittelwert mit der Standardabweichung ausgegeben. In den meisten Fällen ist es allerdings ausreichend einen Simulationslauf mit einer ausreichend großen Anzahl an Instanzen durchzuführen, da sich die Simulation durch die Zufallszahlengenerierung den vorgegebenen Erkennungsraten der Sensoren annähert. Zusätzlich zu der Auswertung der Fusionsmethode werden die Metriken ebenfalls für die einzelnen Sensoren berechnet, um sie mit den Fusionsergebnissen vergleichen zu können. Diese Daten können weiterhin dafür genutzt werden, um zu überprüfen, wie nah die einzelnen Sensoren an den konfigurierten Erkennungsraten liegen und ob die Anzahl der zu simulierenden Instanzen erhöht werden sollte.

Eine Beispielausgabe der Auswertung mehrerer Simulationsläufe mit drei Sensoren ist in Listing 3.2 auf der nächsten Seite gezeigt. Um den Auszug der Ausgabe kompakt zu halten, werden hier nur die Wahrheitsmatrizen des letzten Simulationslaufs gezeigt. Die eigentliche Ausgabe enthält für jeden einzelnen Lauf einen kompletten Block mit den Matrizen der simulierten Sensoren und der Fusion. In einem Block wird als erstes die Fusionsmethode und anschließend die einzelnen Sensoren, in der Reihenfolge in der sie in der Konfiguration definiert wurden, aufgeführt. Zu jeder Wahrheitsmatrix werden die genannten Metriken berechnet und ausgegeben. Den letzten Teil der Ausgabe bilden die gemittelten Werte der Metriken über alle Simulationsläufe, in derselben Reihenfolge in der auch die einzelnen Blöcke strukturiert sind. Die Ausgabe der Wahrheitsmatrizen ist so ausgelegt, dass die Reihen jeweils die tatsächliche Klasse angeben und die Zeilen die Entscheidung des Sensors bzw. der Fusionsmethode. Zeile 42 gibt beispielsweise an, dass die Fusionsmethode 8046 Angriffe richtig erkannt und 1783 Fälle fälschlicherweise als Angriff eingestuft hat. Zusammen mit den Ergebnisse aus Zeile 43 ergibt sich dann unter anderem eine Richtig-Positiv-Rate (`recall`) von $8046/(8046+1954) = 0,8046$ für diesen Simulationslauf.

```
41          ATTACK  NORMAL
42  ATTACK  [8046,  1783]
43  NORMAL  [1954,  8217]
44  acc=0,813150    PPV=0,818598    recall=0,804600 FPR=0,178300    F-measure=0,811539
              AUC=0,813150
45
46          ATTACK  NORMAL
47  ATTACK  [8969,  2990]
48  NORMAL  [1031,  7010]
49  acc=0,798950    PPV=0,749979    recall=0,896900 FPR=0,299000    F-measure=0,816886
              AUC=0,798950
50
51          ATTACK  NORMAL
52  ATTACK  [7994,  1958]
53  NORMAL  [2006,  8042]
54  acc=0,801800    PPV=0,803256    recall=0,799400 FPR=0,195800    F-measure=0,801323
              AUC=0,801800
55
56          ATTACK  NORMAL
57  ATTACK  [6927,  987]
58  NORMAL  [3073,  9013]
59  acc=0,797000    PPV=0,875284    recall=0,692700 FPR=0,098700    F-measure=0,773362
              AUC=0,797000
60
61  acc=0,816250 ± 0,004384  PPV=0,820962 ± 0,003344  recall=0,808900 ± 0,006081       FPR
        =0,176400 ± 0,002687  F-measure=0,814885 ± 0,004733    AUC=0,816250 ± 0,004384
62  acc=0,802450 ± 0,004950  PPV=0,752288 ± 0,003266  recall=0,901850 ± 0,007000       FPR
        =0,296950 ± 0,002899  F-measure=0,820307 ± 0,004837    AUC=0,802450 ± 0,004950
63  acc=0,803675 ± 0,002652  PPV=0,804083 ± 0,001170  recall=0,803000 ± 0,005091       FPR
        =0,195650 ± 0,000212  F-measure=0,803539 ± 0,003133    AUC=0,803675 ± 0,002652
64  acc=0,801050 ± 0,005728  PPV=0,877428 ± 0,003032  recall=0,699850 ± 0,010112       FPR
        =0,097750 ± 0,001344  F-measure=0,778631 ± 0,007452    AUC=0,801050 ± 0,005728
```

Listing 3.2: Beispielausgabe der Auswertung mehrerer Simulationsläufe mit drei Sensoren.

4 Evaluation

Die Evaluation besteht aus zwei Teilen. Im ersten Teil, Abschnitt 4.1, wird die Simulationsumgebung aus Kapitel 3 validiert. Dazu wird zunächst vorgestellt, wie die erwarteten Ergebnisse der implementierten Fusionsmethoden rechnerisch bestimmt werden können. Anschließend werden unterschiedliche Testfälle simuliert und mit den berechneten Werten verglichen. Nachdem sichergestellt ist, dass die Implementierung der Simulationsumgebung ordnungsgemäß funktioniert, wird im zweiten Teil, Abschnitt 4.2, untersucht, wie sich die Eigenschaften der Sensoren auf die einzelnen Fusionsmethoden auswirken.

4.1 Validierung der Simulationsumgebung

Im Rahmen der Validierung wird angenommen, dass alle Sensoren bedingt unabhängig voneinander sind. Weiterhin werden nur die zwei Klassen Angriff und Nicht-Angriff untersucht. Da für die Simulation die Erkennungsraten der Sensoren und die Anzahl der zu simulierenden Angriffe und Nicht-Angriffe (sowohl für Training als auch für die Simulation selbst) konfiguriert werden müssen, bilden diese Angaben die Grundlage der Validierung.

Für die Validierung selbst ist es ausreichend, die Richtig-Positiv- und Falsch-Positiv-Raten zu berechnen und zu vergleichen. Da die Anzahl der simulierten Angriffe und Nicht-Angriffe bekannt ist, lassen sich mit diesen Werten alle Einträge der Wahrheitsmatrix bestimmen. Dazu werden die bekannten Werte in Gleichung 2.7 bzw. Gleichung 2.8 eingesetzt, um so die Anzahl der richtig positiv (RP) bzw. falsch positiv (FP) erkannten Instanzen zu erhalten:

$$RP = \text{Richtig-Positiv-Rate} \cdot \text{Anzahl der tatsächlichen Angriffe},$$
$$FP = \text{Falsch-Positiv-Rate} \cdot \text{Anzahl der tatsächlichen Nicht-Angriffe}.$$

Die zwei verbleibenden Werte der Wahrheitsmatrix, die falsch negativen (FN) und richtig negativen (RN), lassen sich nach Tabelle 2.1 aus der tatsächlichen Anzahl an Angriffen bzw. Nicht-Angriffen und den RP bzw. FP berechnen:

$$FN = \text{Anzahl der tatsächlichen Angriffe} - RP,$$
$$RN = \text{Anzahl der tatsächlichen Nicht-Angriffe} - FP.$$

Der einzige Fall, in dem die Anzahl der Angriffe und Nicht-Angriffe berücksichtigt werden muss, ist das Training der *posterior odds* (PO) Methode. Da die Werte allerdings nur das Ergebnis des Trainings beeinflussen, ändert das nichts daran, dass für die Auswertung die Richtig-Positiv- und Falsch-Positiv-Raten der Fusionsmethoden ausreichend sind.

Für das restliche Kapitel wird auf folgende Notation zurückgegriffen. Bei bedingten Wahrscheinlichkeiten $P(A \mid B)$, $A, B \in \{\text{Angriff}, \text{Normal}\}$ bezieht sich die Bedingung B immer darauf, ob tatsächlich ein Angriff vorliegt. A gibt jeweils die Entscheidung eines Sensors oder der Fusion an. Die Anzahl der verwendeten Sensoren wird als n bezeichnet. Die Entscheidung eines Sensors D_i, $i = 1, \ldots, n$ wird als $d_i \in \{\text{Angriff}, \text{Normal}\}$ angegeben. Wenn d_i eine konkrete Belegung x annimmt, wird für $P(d_i = x \mid B)$ abkürzend $P_i(x \mid B)$ geschrieben.

In den folgenden beiden Abschnitten, 4.1.1 und 4.1.2 wird zunächst gezeigt, wie sich die Richtig-Positiv- und Falsch-Positiv-Raten der verwendeten Fusionsmethoden rechnerisch bestimmen lassen. Anschließend werden in Abschnitt 4.1.3 die Ergebnisse der Simulation mit den erwarteten Werten verglichen, um zu überprüfen, ob die Simulation korrekt funktioniert.

4.1.1 Erkennungsraten der Abstimmungsverfahren

Da die Abstimmungsverfahren kein Training verwenden, sind die gesuchten Erkennungsraten nur von den Sensorenausgaben zur Ausführungszeit (bzw. der Simulationsphase) abhängig. Die beiden Varianten *einstimmge Entscheidung* und *mindestens einer* sind verhältnismäßig einfach zu berechnen, da es jeweils nur einen Fall gibt, der das gesuchte Ergebnis erzielt (bei *mindestens einer* kann der umgekehrte Fall, dass kein Sensor einen Angriff meldet, betrachtet werden). Bei der *einfachen Mehrheit* muss im Grunde jede mögliche Sensorenbelegung, bei der mindestens die Hälfte einen Angriff meldet, berücksichtigt werden, was die Berechnung gegenüber den anderen beiden Varianten etwas aufwändiger macht.

4.1.1.1 Einstimmige Entscheidung

Diese Variante meldet einen Angriff, wenn alle Sensoren einen Angriff erkannt haben. Die Wahrscheinlichkeit, dass dies passiert, in Abhängigkeit davon, ob tatsächlich ein Angriff vorliegt, lässt sich durch das Produkt der Erkennungsraten der einzelnen Sensoren berechnen:

$$P_{AND}(\text{Angriff} \mid \text{Angriff}) = \prod_{i=1}^{n} P_i(\text{Angriff} \mid \text{Angriff}),$$
$$P_{AND}(\text{Angriff} \mid \text{Normal}) = \prod_{i=1}^{n} P_i(\text{Angriff} \mid \text{Normal}).$$

Diese beiden Wahrscheinlichkeiten entsprechen der Richtig-Positiv- bzw. Falsch-Positiv-Rate.

Beispiel 4.1 Gegeben seien $n = 5$ Sensoren mit denselben Erkennungsraten. Ein Angriff wird zu 80% als solcher erkannt und ein Nicht-Angriff zu 90%. Für die Sensoren gilt also $P_i(\text{Angriff} \mid \text{Angriff}) = 0,8$ und $P_i(\text{Normal} \mid \text{Normal}) = 0,9$ bzw. $P_i(\text{Angriff} \mid \text{Normal}) = 1 - 0,9 = 0,1$. Damit ergeben sich die folgenden Richtig-Positiv- und Falsch-Positiv-Raten für die Fusion durch einstimmge Entscheidung:

$$\begin{aligned} P_{AND}(\text{Angriff} \mid \text{Angriff}) &= 0,8 \cdot 0,8 \cdot 0,8 \cdot 0,8 \cdot 0,8 \\ &= 0,32768, \\ P_{AND}(\text{Angriff} \mid \text{Normal}) &= 0,1 \cdot 0,1 \cdot 0,1 \cdot 0,1 \cdot 0,1 \\ &= 0,00001. \end{aligned}$$

4.1.1.2 Mindestens Einer

Die Wahrscheinlichkeit, dass mindestens ein Sensor einen Angriff meldet, kann mit Hilfe der Gegenwahrscheinlichkeit, dass kein Sensor einen Angriff meldet, ausgedrückt werden:

$$P_{OR}(\text{Angriff} \mid \text{Angriff}) = 1 - \prod_{i=1}^{n} P_i(\text{Normal} \mid \text{Angriff}),$$
$$P_{OR}(\text{Angriff} \mid \text{Normal}) = 1 - \prod_{i=1}^{n} P_i(\text{Normal} \mid \text{Normal}).$$

Die Ergebnisse entsprechen der Richtig-Positiv- bzw. Falsch-Positiv-Rate.

Beispiel 4.2 Es seien dieselben fünf Sensoren aus Beispiel 4.1 gegeben. Für die Richtig-Positiv- und Falsch-Positiv-Rate der Fusion ergibt sich in diesem Fall:

$$\begin{aligned} P_{OR}(\text{Angriff} \mid \text{Angriff}) &= 1 - 0,2 \cdot 0,2 \cdot 0,2 \cdot 0,2 \cdot 0,2 \\ &= 0,99968, \\ P_{OR}(\text{Angriff} \mid \text{Normal}) &= 1 - 0,9 \cdot 0,9 \cdot 0,9 \cdot 0,9 \cdot 0,9 \\ &= 0,40951. \end{aligned}$$

4.1.1.3 Einfache Mehrheit

Da es für die einfache Mehrheit ausreichend ist, wenn mindestens die Hälfte der Sensoren einen Angriff melden, müssen deutlich mehr Fälle berücksichtigt werden als bei den beiden vorherigen Varianten. Jede Kombination von Sensoren, bei denen mehr als die Hälfte einen Angriff melden trägt zur Richtig-Positiv-Rate der Fusion bei. Bei fünf Sensoren wären das bereits alle 16 Kombinationen, bei denen drei, vier oder sogar alle fünf Sensoren einen Angriff melden. Sofern die Erkennungsraten aller Sensoren gleich sind, lassen sich die gesuchten Wahrscheinlichkeiten als Summe von Binomialverteilungen berechnen [Mar09, Seite 162]:

$$\begin{aligned} P_{MAJ}(\text{Angriff} \mid \text{Angriff}) &= \sum_{k=\lfloor n/2 \rfloor + 1}^{n} \binom{n}{k} P(\text{Angriff} \mid \text{Angriff})^k P(\text{Normal} \mid \text{Angriff})^{n-k}, \\ P_{MAJ}(\text{Angriff} \mid \text{Normal}) &= \sum_{k=\lfloor n/2 \rfloor + 1}^{n} \binom{n}{k} P(\text{Angriff} \mid \text{Normal})^k P(\text{Normal} \mid \text{Normal})^{n-k}. \end{aligned}$$

Dabei entspricht jeder Summand der Wahrscheinlichkeit, dass genau k Sensoren einen Angriff melden (in allen möglichen Kombinationen). Wenn die Erkennungsraten der Sensoren nicht gleich sind, müssen für jeden Summanden alle $\binom{n}{k}$ Fälle einzeln berechnet werden. Sei S_k die Menge aller möglichen Kombinationen von Sensorenbelegungen für die gilt, dass genau k Sensoren einen Angriff melden, d.h. $|S_k| = \binom{n}{k}$. Ein Element $s \in S_k$ dieser Menge ist eine konkrete Sensorenbelegung der Form $(d_1, \ldots, d_n)$. Mit Hilfe dieser Notation können die Summanden auf folgende Weise beschrieben werden:

$$\begin{aligned} P_{MAJ}(\text{Angriff} \mid \text{Angriff}) &= \sum_{k=\lfloor n/2 \rfloor + 1}^{n} \sum_{s \in S_k} \prod_{i=1}^{n} P(d_i \mid \text{Angriff}), \\ P_{MAJ}(\text{Angriff} \mid \text{Normal}) &= \sum_{k=\lfloor n/2 \rfloor + 1}^{n} \sum_{s \in S_k} \prod_{i=1}^{n} P(d_i \mid \text{Normal}). \end{aligned}$$

Sensorenbelegung	k	$\prod_{i=1}^{n} P(d_i \mid \text{Angriff})$	$\prod_{i=1}^{n} P(d_i \mid \text{Normal})$
(Angriff, Angriff, Angriff)	3	0,512	0,001
(Angriff, Angriff, Normal)	2	0,128	0,009
(Angriff, Normal, Angriff)	2	0,128	0,009
(Normal, Angriff, Angriff)	2	0,128	0,009

Tabelle 4.1: Die Wahrscheinlichkeiten, dass eine bestimmte Sensorenbelegung auftritt, wenn ein bzw. kein Angriff vorliegt.

Die so bestimmten Wahrscheinlichkeiten entsprechen den gesuchten Richtig-Positiv- bzw. Falsch-Positiv-Raten. Das folgende Beispiel zeigt die Berechnung für beide Varianten.

Beispiel 4.3 Gegeben seien $n = 3$ Sensoren, deren Erkennungsraten für Angriffe mit 80% und für Nicht-Angriffe mit 90% konfiguriert sind. Entsprechend gilt für die Sensoren $P_i(\text{Angriff} \mid \text{Angriff}) = 0,8$ und $P_i(\text{Normal} \mid \text{Normal}) = 0,9$. Als erster Schritt werden die Wahrscheinlichkeiten der einzelnen Sensorenbelegungen berechnet. Als Beispiel werden die Werte für die Belegung (Angriff, Angriff, Normal) bestimmt:

$$\begin{aligned} &P_1(\text{Angriff} \mid \text{Angriff}) \cdot P_2(\text{Angriff} \mid \text{Angriff}) \cdot P_3(\text{Normal} \mid \text{Angriff}) \\ &\quad = 0,8 \cdot 0,8 \cdot 0,2 = 0,128, \\ &P_1(\text{Angriff} \mid \text{Normal}) \cdot P_2(\text{Angriff} \mid \text{Normal}) \cdot P_3(\text{Normal} \mid \text{Normal}) \\ &\quad = 0,1 \cdot 0,1 \cdot 0,9 = 0,009. \end{aligned}$$

Die Ergebnisse sind in Tabelle 4.1 aufgeführt. Da sie nicht benötigt werden, wurden die Fälle für $k < \lfloor n/2 \rfloor + 1$ ausgelassen. Wie zu erkennen ist, sind die Werte in den drei Fällen für $k = 2$ gleich. Es wäre also ausreichend gewesen, nur einen Fall stellvertretend zu berechnen und diesen mit $\binom{3}{2} = 3$ zu multiplizieren. Unabhängig davon, wie die Werte zusammengezählt werden, ergibt sich für die gesuchten Richtig-Positiv- und Falsch-Positiv-Raten:

$$\begin{aligned} P_{MAJ}(\text{Angriff} \mid \text{Angriff}) &= 0,128 + 0,128 + 0,128 + 0,512 \\ &= 3 \cdot 0,128 + 0,512 \\ &= 0,896, \\ P_{MAJ}(\text{Angriff} \mid \text{Normal}) &= 0,009 + 0,009 + 0,009 + 0,001 \\ &= 3 \cdot 0,009 + 0,001 \\ &= 0,028. \end{aligned}$$

4.1.2 Erkennungsraten der naive Bayes-basierten Verfahren

Im Gegensatz zu den Abstimmungsverfahren benötigen die *naive Bayes*-basierten Fusionsmethoden (NBF-Methoden) eine Trainingsphase. Durch das Training wird für jede Belegung der Sensoren vorab bestimmt, welche Entscheidung zu treffen ist. Für die eigentliche Entscheidung im Einsatz (bzw. hier der Simulation) wird entsprechend der aktuellen Sensorenausgaben die trainierte Entscheidung ausgewählt.

Für die Berechnung der Erkennungsraten des *likelihood ratio test* (LRT) und der *posterior odds* (PO) Methode kann dasselbe Berechnungsschema angewandt werden. Der einzige Unterschied ist, dass PO die Wahrscheinlichkeit, dass ein Angriff überhaupt stattfindet, im Training berücksichtigt. Daher wird zuerst das Vorgehen für LRT beschrieben und anschließend die nötigen Erweiterungen für PO vorgestellt.

4.1.2.1 Likelihood Ratio Test

Um besser mit den trainierten Entscheidungen rechnen zu können, wird die Hilfsfunktion δ eingeführt:

$$\delta(d_1,\ldots,d_n) = \begin{cases} 1, & \text{wenn } \frac{\prod_{i=1}^{n} P(d_i|\text{Angriff})}{\prod_{i=1}^{n} P(d_i|\text{Normal})} \geq 1, \\ 0 & \text{sonst.} \end{cases}$$

Diese Funktion ergibt 1, wenn für die gegebene Sensorenbelegung $d_1,\ldots,d_n$ die Entscheidung Angriff trainiert wurde und 0 sonst. Die verwendete Entscheidungsregel entspricht dabei dem LRT (Gleichung 2.6). Um die Richtig-Positiv- bzw. Falsch-Positiv-Rate der Fusion bestimmen zu können, muss für jede mögliche Kombination von Sensorenausgaben berechnet werden, wie häufig sie auftritt und welche Entscheidung in diesem Fall getroffen wird. Sei S die Menge aller möglichen Kombinationen von Sensorenbelegungen $(d_1,\ldots,d_n)$ und $s \in S$ ein Element dieser Menge. Dann lässt sich die Richtig-Positive-Rate als

$$P_{LRT}(\text{Angriff} \mid \text{Angriff}) = \sum_{s\in S} \delta(s) \prod_{i=1}^{n} P(d_i \mid \text{Angriff})$$

und die Falsch-Positiv-Rate durch

$$P_{LRT}(\text{Angriff} \mid \text{Normal}) = \sum_{s\in S} \delta(s) \prod_{i=1}^{n} P(d_i \mid \text{Normal})$$

berechnen. Die Anwendung dieser Formel wird im folgenden Beispiel gezeigt.

Sensorenbelegung	$\prod_{i=1}^{n} P(d_i \mid \text{Angriff})$	$\prod_{i=1}^{n} P(d_i \mid \text{Normal})$	$\delta(d_1, d_2, d_3)$
(Angriff, Angriff, Angriff)	0,504	0,001	1 (Angriff)
(Angriff, Angriff, Normal)	0,056	0,009	1 (Angriff)
(Angriff, Normal, Angriff)	0,126	0,009	1 (Angriff)
(Angriff, Normal, Normal)	0,014	0,081	0 (Normal)
(Normal, Angriff, Angriff)	0,216	0,009	1 (Angriff)
(Normal, Angriff, Normal)	0,024	0,081	0 (Normal)
(Normal, Normal, Angriff)	0,054	0,081	0 (Normal)
(Normal, Normal, Normal)	0,006	0,729	0 (Normal)

Tabelle 4.2: Die Wahrscheinlichkeiten, auf deren Grundlage Entscheidungen trainiert wurden, und die entsprechenden Entscheidungen für die jeweilige Sensorenbelegungen.

Beispiel 4.4 Gegeben seien $n = 3$ Sensoren. Die Sensoren sind für das Training so konfiguriert, dass sie einen Angriff zu 70%, 80% und 90% erkennen. Normale Aktivitäten werden von allen drei Sensoren zu 90% als solche erkannt. Für die Simulation wird die Richtig-Negativ-Rate der Sensoren auf 95% angehoben, was den Effekt simulieren soll, dass sie in der Praxis bessere Raten erreichen, als auf den Trainingsdaten. Die Sensitivität bleibt bei 70%, 80% bzw. 90%.

Als erster Schritt werden die trainierten Entscheidungen bestimmt. Dazu wird für jede der acht möglichen Sensorenbelegungen δ berechnet. Für den Fall, dass alle drei Sensoren einen Angriff melden, ergibt sich für den Zähler des *likelihood ratio*

$$\begin{aligned} &P_1(\text{Angriff} \mid \text{Angriff}) \cdot P_2(\text{Angriff} \mid \text{Angriff}) \cdot P_3(\text{Angriff} \mid \text{Angriff}) \\ &\quad = 0,7 \cdot 0,8 \cdot 0,9 = 0,504. \end{aligned}$$

Der Nenner beläuft sich auf

$$\begin{aligned} &P_1(\text{Angriff} \mid \text{Normal}) \cdot P_2(\text{Angriff} \mid \text{Normal}) \cdot P_3(\text{Angriff} \mid \text{Normal}) \\ &\quad = 0,1 \cdot 0,1 \cdot 0,1 = 0,001. \end{aligned}$$

Entsprechend ergibt sich für $\delta(\text{Angriff}, \text{Angriff}, \text{Angriff}) = 1$, da $\frac{0,504}{0,001} = 504 \geq 1$ ist. Die Ergebnisse für alle Belegungsmöglichkeiten sind in Tabelle 4.2 aufgelistet.

Der zweite Schritt ist, mit diesen Trainingsdaten die Richtig-Positiv- und die Falsch-Positiv-Rate der Fusion zu bestimmen. Dazu werden wieder alle Kombinationen von Sensorenausgaben benötigt. Falls die Sensoren mit denselben Erkennungsraten für Training und Simulation konfiguriert wären, könnten die Werte aus Tabelle 4.2 wiederverwendet werden. In diesem Bei-

Sensorenbelegung	$\prod_{i=1}^{n} P(d_i \mid \text{Angriff})$	$\prod_{i=1}^{n} P(d_i \mid \text{Normal})$
(Angriff, Angriff, Angriff)	0,504	0,000125
(Angriff, Angriff, Normal)	0,056	0,002375
(Angriff, Normal, Angriff)	0,126	0,002375
(Angriff, Normal, Normal)	0,014	0,045125
(Normal, Angriff, Angriff)	0,216	0,002375
(Normal, Angriff, Normal)	0,024	0,045125
(Normal, Normal, Angriff)	0,054	0,045125
(Normal, Normal, Normal)	0,006	0,857375

Tabelle 4.3: Die Wahrscheinlichkeiten, dass eine bestimmte Sensorenbelegung auftritt, wenn ein bzw. kein Angriff vorliegt.

spiel ist das für $\prod_{i=1}^{n} P(d_i \mid \text{Angriff})$ der Fall. Da die Richtig-Negativ-Rate für die Simulation höher ist, müssen alle $\prod_{i=1}^{n} P(d_i \mid \text{Normal})$ Kombinationen mit den entsprechenden Erkennungsraten neu berechnet werden. Das Ergebnis ist in Tabelle 4.3 zusammengefasst. Unabhängig davon, wie die Wahrscheinlichkeiten in der Simulationsphase ausfallen, wird die bereits trainierte Entscheidung für eine Sensorenbelegung gewählt. Beispielsweise müsste die Belegung (Normal, Normal, Angriff) aus Tabelle 4.3 im Training als Angriff eingestuft werden. Da dieser Fall im Training aber als Normal eingestuft wurde, wird auch Normal als Entscheidung angegeben, obwohl ein Angriff die höhere Wahrscheinlichkeit hätte. Um die Richtig-Positiv-Rate, $P_{LRT}(\text{Angriff} \mid \text{Angriff})$, zu erhalten, müssen die Wahrscheinlichkeiten $\prod_{i=1}^{n} P(d_i \mid \text{Angriff})$ addiert werden, für die $\delta = 1$ ist (die Fälle, in denen $\delta = 0$ ist, werden durch die Multiplikation mit δ zu 0 und können ausgelassen werden):

$$P_{LRT}(\text{Angriff} \mid \text{Angriff}) = 0,504 + 0,056 + 0,126 + 0,216 = 0,902.$$

Für die Falsch-Positiv-Rate ergibt sich entsprechend

$$P_{LRT}(\text{Angriff} \mid \text{Normal}) = 0,000125 + 0,002375 + 0,002375 + 0,002375 = 0,00725.$$

4.1.2.2 Posterior Odds

Das Vorgehen für PO ähnelt dem von LRT. Der einzige Unterschied ist, dass für δ die PO Entscheidungsregel, Gleichung 2.5, verwendet wird.

$$\delta(d_1,\ldots,d_n) = \begin{cases} 1, & \text{wenn } \frac{P(\text{Angriff})\prod_{i=1}^{n} P(d_i|\text{Angriff})}{P(\text{Normal})\prod_{i=1}^{n} P(d_i|\text{Normal})} \geq 1, \\ 0 & \text{sonst.} \end{cases}$$

Da im zweiten Schritt die A-priori-Wahrscheinlichkeiten $P(\text{Angriff})$ und $P(\text{Normal})$ keinen Einfluss mehr auf die trainierten Entscheidungen haben, können alle weiteren Berechnungen wie für LRT beschrieben durchgeführt werden.

4.1.3 Validierung der Simulationsumgebung

Für die Validierung wird jede der beschriebenen fünf Varianten mit denselben sechs Szenarien getestet. Je zwei Szenarien verwenden drei, sechs und neun Sensoren. Sie unterscheiden sich darin, dass im ersten Fall alle Sensoren mit denselben Erkennungsraten konfiguriert sind, während im zweiten Fall die Sensoren unterschiedliche Erkennungsraten haben und zusätzlich die Raten im Training von denen der Simulation abweichen. Eine Übersicht der verwendeten Szenarien ist in Tabelle 4.4 gegeben. Für jedes Szenario werden zehn Simulationsläufe durchgeführt und anschließend der Mittelwert der Richtig-Positiv- und Falsch-Positiv-Raten gebildet. Bei den NBF-Methoden werden für die Trainingsphase 100.000 Angriffe und 9.900.000 Nicht-Angriffe simuliert, was der A-priori-Wahrscheinlichkeit eines Angriffs von 1% entspricht. Da bei den verwendeten Fusionsmethoden das Verhältnis von Angriffen zu Nicht-Angriffen in der Simulationsphase keinen Einfluss auf die Entscheidungsfindung hat, werden für diese Phase jeweils 1.000.000 Angriffe und 1.000.000 Nicht-Angriffe simuliert.

Für jedes Szenario ist ein fester Startwert für die Zufallszahlengenerierung gesetzt, damit die Sensoren sich bei jeder Fusionsvariante gleich verhalten. Die Erkennungsraten, die die Sensoren in der Simulationsphase erzielen, sind in Tabelle 4.5 aufgeführt. Die simulierten Erkennungsraten liegen sehr nahe bei den konfigurierten Werten aus Tabelle 4.4 und stimmen, auf zwei Nachkommastellen gerundet, mit ihnen überein.

Die Ergebnisse der Fusionsmethoden aus der Simulation sind mit den berechneten Werten in Tabelle 4.6 und Tabelle 4.7 gegenübergestellt. Dabei wurden die berechneten Erkennungsraten auf sechs Nachkommastellen gerundet. Bei den Abstimmungsverfahren entsprechen die Ergebnisse der Simulation in fast allen Fällen bis auf mindestens drei Nachkommastellen den

	Sensor 1				Sensor 2				Sensor 3			
	Simulation		Training		Simulation		Training		Simulation		Training	
Fall	RPR	RNR	RPR	RNR	RPR	RNR	RPR	RNR	RPR	RNR	RPR	RNR
1	0,99	0,95	0,99	0,95	0,99	0,95	0,99	0,95	0,99	0,95	0,99	0,95
2	0,9	0,8	0,9	0,8	0,9	0,8	0,9	0,8	0,9	0,8	0,9	0,8
3	0,75	0,85	0,75	0,85	0,75	0,85	0,75	0,85	0,75	0,85	0,75	0,85
4	0,95	0,85	0,85	0,95	0,9	0,9	0,9	0,9	0,85	0,95	0,95	0,85
5	0,95	0,7	0,7	0,95	0,9	0,75	0,75	0,9	0,85	0,8	0,8	0,85
6	0,99	0,99	0,6	0,6	0,95	0,95	0,65	0,65	0,9	0,9	0,7	0,7

	Sensor 4				Sensor 5				Sensor 6			
	Simulation		Training		Simulation		Training		Simulation		Training	
Fall	RPR	RNR	RPR	RNR	RPR	RNR	RPR	RNR	RPR	RNR	RPR	RNR
2	0,9	0,8	0,9	0,8	0,9	0,8	0,9	0,8	0,9	0,8	0,9	0,8
3	0,75	0,85	0,75	0,85	0,75	0,85	0,75	0,85	0,75	0,85	0,75	0,85
5	0,8	0,85	0,85	0,8	0,75	0,9	0,9	0,75	0,7	0,95	0,95	0,7
6	0,85	0,85	0,75	0,75	0,8	0,8	0,8	0,8	0,75	0,75	0,85	0,85

	Sensor 7				Sensor 8				Sensor 9			
	Simulation		Training		Simulation		Training		Simulation		Training	
Fall	RPR	RNR	RPR	RNR	RPR	RNR	RPR	RNR	RPR	RNR	RPR	RNR
3	0,75	0,85	0,75	0,85	0,75	0,85	0,75	0,85	0,75	0,85	0,75	0,85
6	0,7	0,7	0,9	0,9	0,65	0,65	0,95	0,95	0,6	0,6	0,99	0,99

Tabelle 4.4: Die konfigurierten Richtig-Positiv- und Richtig-Negativ-Raten (RPR und RNR) der Sensoren der sechs Validierungsszenarien.

	Sensor 1		Sensor 2		Sensor 3	
Fall	Richtig-Positiv-Rate	Falsch-Positiv-Rate	Richtig-Positiv-Rate	Falsch-Positiv-Rate	Richtig-Positiv-Rate	Falsch-Positiv-Rate
1	$0,990004 \pm 0,000076$	$0,050043 \pm 0,000189$	$0,989999 \pm 0,000117$	$0,049951 \pm 0,000159$	$0,990009 \pm 0,000124$	$0,050032 \pm 0,000151$
2	$0,899962 \pm 0,000269$	$0,200013 \pm 0,000273$	$0,900163 \pm 0,000245$	$0,199807 \pm 0,000403$	$0,899960 \pm 0,000282$	$0,200062 \pm 0,000299$
3	$0,749924 \pm 0,000481$	$0,150037 \pm 0,000303$	$0,750151 \pm 0,000336$	$0,149965 \pm 0,000282$	$0,749882 \pm 0,000499$	$0,149980 \pm 0,000364$
4	$0,950025 \pm 0,000243$	$0,150135 \pm 0,000503$	$0,900044 \pm 0,000390$	$0,099972 \pm 0,000303$	$0,849960 \pm 0,000363$	$0,050117 \pm 0,000140$
5	$0,950061 \pm 0,000144$	$0,300001 \pm 0,000533$	$0,899955 \pm 0,000393$	$0,249977 \pm 0,000382$	$0,849948 \pm 0,000305$	$0,199819 \pm 0,000462$
6	$0,989989 \pm 0,000104$	$0,009965 \pm 0,000152$	$0,949981 \pm 0,000154$	$0,049944 \pm 0,000257$	$0,899911 \pm 0,000337$	$0,100035 \pm 0,000292$

	Sensor 4		Sensor 5		Sensor 6	
Fall	Richtig-Positiv-Rate	Falsch-Positiv-Rate	Richtig-Positiv-Rate	Falsch-Positiv-Rate	Richtig-Positiv-Rate	Falsch-Positiv-Rate
2	$0,900034 \pm 0,000414$	$0,199841 \pm 0,000499$	$0,900132 \pm 0,000287$	$0,200218 \pm 0,000243$	$0,900039 \pm 0,000336$	$0,199947 \pm 0,000289$
3	$0,749998 \pm 0,000455$	$0,149943 \pm 0,000354$	$0,750266 \pm 0,000367$	$0,149956 \pm 0,000255$	$0,750020 \pm 0,000289$	$0,149961 \pm 0,000410$
5	$0,799943 \pm 0,000521$	$0,149926 \pm 0,000441$	$0,749712 \pm 0,000509$	$0,100164 \pm 0,000226$	$0,700068 \pm 0,000374$	$0,049903 \pm 0,000069$
6	$0,850102 \pm 0,000326$	$0,150057 \pm 0,000258$	$0,799948 \pm 0,000484$	$0,199938 \pm 0,000558$	$0,750119 \pm 0,000369$	$0,250076 \pm 0,000498$

	Sensor 7		Sensor 8		Sensor 9	
Fall	Richtig-Positiv-Rate	Falsch-Positiv-Rate	Richtig-Positiv-Rate	Falsch-Positiv-Rate	Richtig-Positiv-Rate	Falsch-Positiv-Rate
3	$0,749958 \pm 0,000363$	$0,150093 \pm 0,000330$	$0,749929 \pm 0,000546$	$0,149924 \pm 0,000392$	$0,749903 \pm 0,000225$	$0,149961 \pm 0,000430$
6	$0,700019 \pm 0,000483$	$0,300104 \pm 0,000370$	$0,650064 \pm 0,000393$	$0,350162 \pm 0,000469$	$0,600078 \pm 0,000619$	$0,399942 \pm 0,000533$

Tabelle 4.5: Die Erkennungsraten der simulierten Sensoren, gemittelt über zehn Läufe.

	Einstimmige Entscheidung		Mindestens Einer		Einfache Mehrheit	
Fall	Richtig-Positiv-Rate	Falsch-Positiv-Rate	Richtig-Positiv-Rate	Falsch-Positiv-Rate	Richtig-Positiv-Rate	Falsch-Positiv-Rate
1	$0,970310 \pm 0,000152$ $0,970299$	$0,000126 \pm 0,000007$ $0,000125$	$0,999999 \pm 0,000001$ $0,999999$	$0,142602 \pm 0,000262$ $0,142625$	$0,999702 \pm 0,000018$ $0,999702$	$0,007298 \pm 0,000079$ $0,007250$
2	$0,531635 \pm 0,000476$ $0,531441$	$0,000062 \pm 0,000009$ $0,000064$	$0,999999 \pm 0,000001$ $0,999999$	$0,737671 \pm 0,000369$ $0,737856$	$0,998721 \pm 0,000035$ $0,998730$	$0,098977 \pm 0,000352$ $0,098880$
3	$0,075203 \pm 0,000185$ $0,075085$	$0,000000 \pm 0,000000$ $0,000000$	$0,999997 \pm 0,000002$ $0,999996$	$0,768269 \pm 0,000526$ $0,768383$	$0,951073 \pm 0,000298$ $0,951073$	$0,005657 \pm 0,000036$ $0,005629$
4	$0,726759 \pm 0,000610$ $0,726750$	$0,000748 \pm 0,000021$ $0,000750$	$0,999249 \pm 0,000020$ $0,999250$	$0,273505 \pm 0,000474$ $0,273250$	$0,974021 \pm 0,000151$ $0,974000$	$0,025971 \pm 0,000138$ $0,026000$
5	$0,305007 \pm 0,000565$ $0,305235$	$0,000011 \pm 0,000001$ $0,000011$	$0,999990 \pm 0,000004$ $0,999989$	$0,694673 \pm 0,000421$ $0,694765$	$0,991977 \pm 0,000098$ $0,991943$	$0,064604 \pm 0,000271$ $0,064620$
6	$0,117804 \pm 0,000419$ $0,117851$	$0,000000 \pm 0,000000$ $0,000000$	$1,000000 \pm 0,000000$ $1,000000$	$0,882223 \pm 0,000211$ $0,882149$	$0,986393 \pm 0,000150$ $0,986373$	$0,013607 \pm 0,000136$ $0,013627$

Tabelle 4.6: Die Simulationsergebnisse der Abstimmungsverfahren, gemittelt über zehn Läufe, im Vergleich mit den berechneten Erkennungsraten.

	Likelihood Ratio Test		Posterior Odds	
Fall	Richtig-Positiv-Rate	Falsch-Positiv-Rate	Richtig-Positiv-Rate	Falsch-Positiv-Rate
1	$0,999702 \pm 0,000018$ $0,999702$	$0,007298 \pm 0,000079$ $0,007250$	$0,970310 \pm 0,000152$ $0,970299$	$0,000126 \pm 0,000007$ $0,000125$
2	$0,984147 \pm 0,000174$ $0,984150$	$0,016994 \pm 0,000143$ $0,016960$	$0,885845 \pm 0,000338$ $0,885735$	$0,001609 \pm 0,000028$ $0,001600$
3	$0,989987 \pm 0,000174$ $0,990005$	$0,033958 \pm 0,000124$ $0,033932$	$0,834157 \pm 0,000260$ $0,834274$	$0,000628 \pm 0,000031$ $0,000634$
4	$0,974021 \pm 0,000151$ $0,974000$	$0,025971 \pm 0,000138$ $0,026000$	$0,726759 \pm 0,000610$ $0,726750$	$0,000748 \pm 0,000021$ $0,000750$
5	$0,968319 \pm 0,009844$ $0,954373$	$0,035208 \pm 0,009846$ $0,021200$	$0,717883 \pm 0,000535$ $0,717949$	$0,000491 \pm 0,000024$ $0,000495$
6	$0,860014 \pm 0,001894$ $0,860561$	$0,139421 \pm 0,000431$ $0,139439$	$0,642160 \pm 0,000539$ $0,641997$	$0,033483 \pm 0,000185$ $0,033489$

Tabelle 4.7: Die Simulationsergebnisse der NBF-Methoden, gemittelt über zehn Läufe, im Vergleich mit den berechneten Erkennungsraten.

berechneten Erkennungsraten. Bei den NBF-Methoden stimmen die meisten Erkennungsraten der Simulation mit den berechneten Werten überein, wenn auf drei Nachkommastellen gerundet wird. Eine große Ausnahme bildet Fall 5 des LRT-Szenarios aus Tabelle 4.7. In diesem Fall weichen die Mittelwerte der Erkennungsraten um fast $0,015$ von den erwarteten Werten ab. Die Ursache dafür ist, dass die Erkennungsraten der Sensoren nicht exakt den konfigurierten Werten entsprechen, sondern, bedingt durch die Zufallszahlengenerierung, sich den Raten nur annähern. Das kann dazu führen, dass besonders knappe Entscheidungen anders ausfallen als angenommen. Wenn der Quotient, auf dessen Grundlage der LRT eine Entscheidung trifft, nahe bei $1,0$ liegt, kann eine kleine Schwankung bereits dafür sorgen, dass der Wert über bzw. unter 1 fällt und die „falsche“ Entscheidung trainiert wird. Ein Indiz dafür, dass die Fusionsmethode solchen Schwankungen unterliegt, kann die Standardabweichung sein, die in diesem Fall mit fast $0,01$ im Verhältnis zu allen anderen simulierten Szenarien relativ hoch ist. Wenn die

Berechnung mit den Mittelwerten der simulierten Sensoren aus Trainings- und Simulationsphase durchgeführt wird, ergibt sich für die Richtig-Positiv-Rate ein Wert von $0,9641657035$ und für die Falsch-Positiv-Rate $0,031002723$. Diese beiden Erkennungsraten liegen deutlich näher an den Simulationsergebnissen und bestätigen somit, dass es sich nicht um einen Fehler in der Simulationsumgebung handelt.

Auch wenn diese Abweichung zunächst problematisch erscheint, ist sie keineswegs ein Argument gegen die Simulation. In der Praxis wird kein Sensor exakt die erwarteten Erkennungsraten liefern können, weshalb es durchaus hilfreich sein kann, solche Ergebnisse durch die Simulation zu erhalten. Da, bis auf einige wenige Ausnahmen, die Erkennungsraten der Simulation nah genug an den erwarteten Werten liegen, kann davon ausgegangen werden, dass die Implementierung der Fusionsmethoden und die Generierung der Sensorenausgaben sich wie erwartet verhält.

4.2 Experimente

In diesem Teil der Evaluation wird mit der Simulationsumgebung untersucht, wie sich die Eigenschaften der Sensoren auf die Fusionsmethoden auswirken. Dazu werden verschiedene Versuchsreihen durchgeführt, die jeweils mehrere Szenarien enthalten, um so die fünf Fusionsvarianten *einstimmge Entscheidung* (AND), *mindestens einer* (OR), *relative Mehrheit* (MAJ), *likelihood ratio test* (LRT) und *posterior odds* (PO) abzudecken. Weiterhin werden die Szenarien nach Anzahl der verwendeten Sensoren unterteilt. Dabei werden drei, sechs und neun Sensoren für die Fusionsmethoden und ein einzelner Sensor als Vergleichswert unterschieden. Jedes dieser Szenarien wird mit den Erkennungsraten von 50% bis 100% in 5% Schritten simuliert, zuzüglich dem Fall 99%, um einen zusätzlichen Wert nahe an 100% vorliegen zu haben. Entsprechend besteht jedes Szenario aus zwölf einzelnen Simulationsläufen. Auf Erkennungsraten unter 50% wird verzichtet, da es sich im Grunde um Erkennungsraten über 50% handelt, wenn genau die gegenteilige Entscheidung gewählt würde. Für jeden dieser Läufe werden 20.000 Angriffe und 1.980.000 Nicht-Angriffe generiert, um eine Angriffswahrscheinlichkeit von $0,01$ zu simulieren.

In den folgenden Abschnitten, 4.2.1 bis 4.2.4 werden die einzelnen Versuchsreihen mit ihren Ergebnissen vorgestellt. Eine vollständige Auflistung aller simulierten Szenarien und deren Ergebnisse sind auf der CD aus Anhang A zu finden. Anschließend werden in Abschnitt 4.2.5 die einzelnen Erkenntnisse zusammengefasst und diskutiert.

4.2.1 Anzahl der Sensoren

Mit dieser Versuchsreihe soll untersucht werden, wie sich die Anzahl der verwendeten Sensoren auf die Fusion auswirkt. Weiterhin dienen diese Ergebnisse als Vergleichswerte für die weiteren Versuchsreihen. Innerhalb eines Simulationslaufs werden die Erkennungsraten aller Sensoren einheitlich gewählt und über ein vollständiges Szenario von 50% bis 100% erhöht. Über die Szenarien hinweg ändert sich die Anzahl der verwendeten Sensoren und welche der Erkennungsraten variiert wird. Dabei werden drei Varianten von Erkennungsraten simuliert: Für den ersten Fall werden die Richtig-Positiv- und die Richtig-Negativ-Rate gleichermaßen erhöht. Im zweiten Fall wird die Richtig-Positiv-Rate fest auf 99% gesetzt und nur die Richtig-Negativ-Rate verändert. Beim dritten Fall ist es genau umgekehrt und die Richtig-Negativ-Rate wird festgesetzt, während die Richtig-Positiv-Rate von 50% bis 100% gesteigert wird. Zusätzlich werden die Szenarien für bedingt unabhängige und korrelierte Sensoren simuliert.

4.2.1.1 Bedingt unabhängige Sensoren

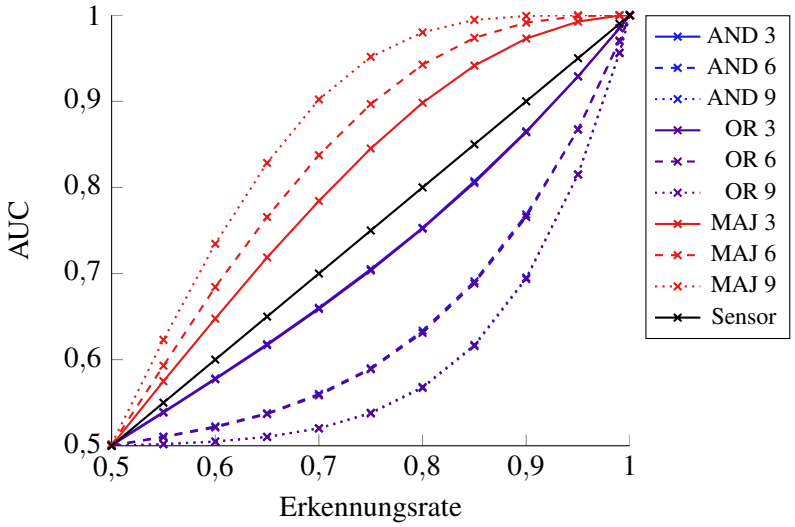

Abbildung 4.1: AUC-Werte der Abstimmungsverfahren mit je drei, sechs und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate. AND und OR überlagern sich.

Richtig-Positiv- und Richtig-Negativ-Rate gleich Die beiden Abbildungen 4.1 und 4.2 zeigen den Verlauf der AUC-Werte der einzelnen Szenarien bei steigenden Erkennungsraten. Dabei liegen sowohl AND und OR als auch MAJ und LRT jeweils auf denselben Kurven. Im Fall von

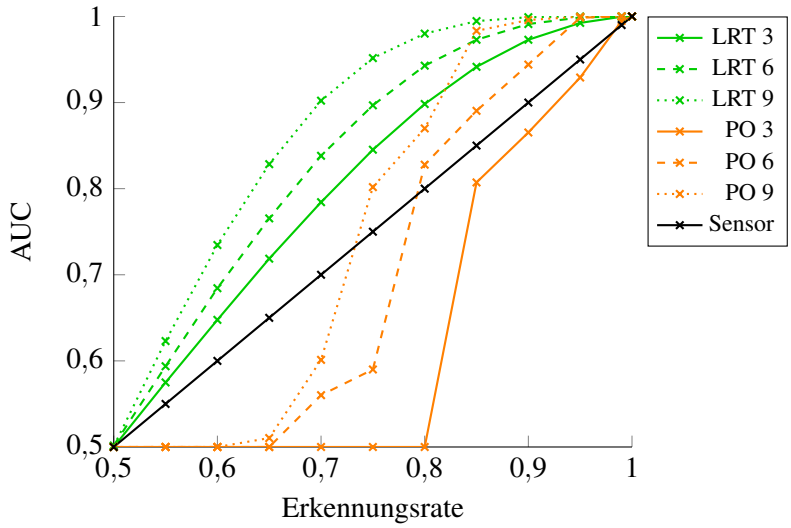

Abbildung 4.2: AUC-Werte der NBF-Verfahren mit je drei, sechs und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate.

AND und OR liegt das daran, dass wenn die Richtig-Positiv- und Richtig-Negativ-Raten der Sensoren gleich sind, die Richtig-Positiv-Rate von AND der Richtig-Negativ-Rate von OR entspricht und umgekehrt. Bei MAJ und LRT für drei und neun Sensoren sind die Raten, bis auf den Anfang, identisch, da in diesem Szenario LRT ab einem gewissen Punkt, wie MAJ, die Entscheidung der Mehrheit der Sensoren übernimmt. Bei sechs Sensoren entscheidet sich MAJ allerdings bei nur drei Angriffsmeldungen für einen Angriff und LRT nicht. Dass die Werte der AUC dennoch übereinstimmen liegt daran, dass LRT alle diese Fälle als Nicht-Angriffe wertet. Damit nimmt die Richtig-Positiv-Rate gegenüber MAJ um einen bestimmten Wert ab, dieser Wert wird bei MAJ allerdings zur Falsch-Positiv-Rate hinzugefügt, weshalb zumindest die AUC gleich ist. Bei den drei Fusionsmethoden MAJ, LRT und PO erreicht jeweils eine höhere Anzahl von Sensoren ein besseres Ergebnis. Im Gegensatz dazu verschlechtert sich bei AND und OR das Ergebnis bei steigender Sensorenanzahl.

Um zu untersuchen, wie sich die Fusionsmethoden bei einer geringen Zahl von Angriffen verhalten, wird in den Abbildungen 4.3 und 4.4 das F-Maß für eine Angriffswahrscheinlichkeit von 1% dargestellt. Obwohl es so aussieht, beginnen die Kurven, mit Ausnahme von PO, nicht mit einem Wert von 0, sondern liegen nur sehr nahe bei 0. Um einen Wert von 0 zu erreichen, darf kein einziger Angriff als solcher erkannt werden. Dieser Fall tritt hier nur bei PO ein, da diese Methode bei niedrigen Erkennungsraten tatsächlich immer Nicht-Angriff meldet. Weiter-

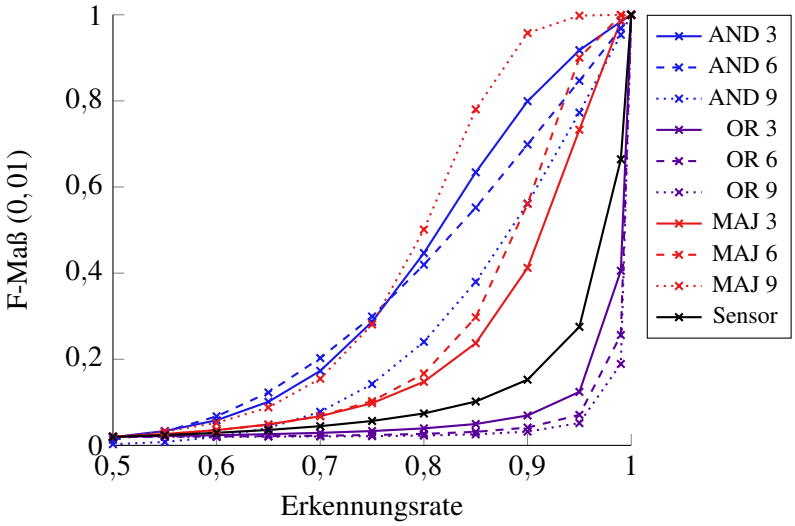

Abbildung 4.3: F-Maß Bewertungen für die Abstimmungsverfahren mit je drei, sechs und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate.

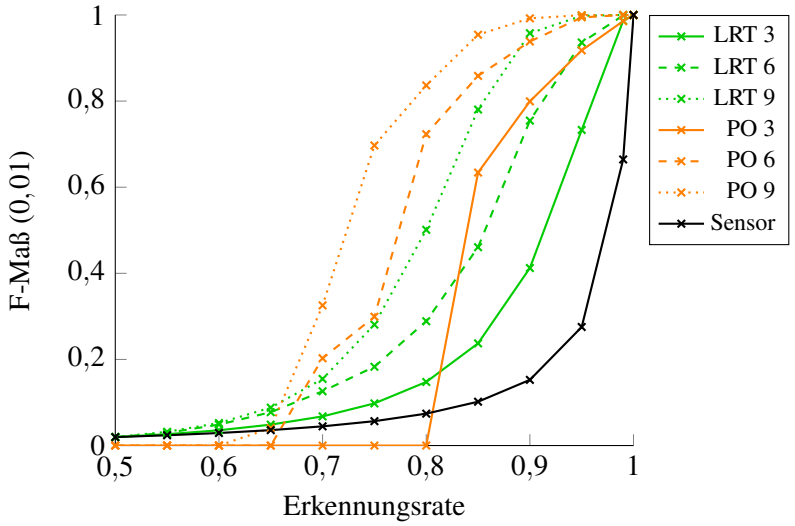

Abbildung 4.4: F-Maß Bewertungen für die NBF-Verfahren mit je drei, sechs und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate.

hin teilen sich MAJ und LRT bei drei und neun Sensoren, bis auf die ersten Werte, wieder eine Kurve, da sie dieselben Entscheidungen für jede Sensorenbelegung treffen. Bei sechs Sensoren liegt LRT beim F-Maß vor MAJ, da LRT ein Teil der Sensitivität für eine höhere Spezifität eintauscht, die bei einer größeren Anzahl von Nicht-Angriffen stärker ins Gewicht fällt. OR liegt als einzige Methode unter dem einzelnen Sensor. Für AND und OR gilt wieder, dass weniger Sensoren eine bessere Leistung aufweisen, wobei sich das im Fall von AND erst bemerkbar macht, wenn die Erkennungsraten auf 1 zugehen. Überraschend ist, dass bei nur drei Sensoren AND den anderen Methoden überlegen ist bzw. zum Teil mit PO gleich auf ist. Das ist auf die hohe Spezifität von AND zurückzuführen, die im F-Maß stärker berücksichtigt wird. Bei den anderen drei Fusionsmethoden erzielen mehr Sensoren auch beim F-Maß bessere Ergebnisse. Insgesamt scheint PO von den betrachteten Methoden die besten Ergebnisse zu liefern, sobald eine gewisse Erkennungsrate überschritten ist.

Bei den sehr steilen Anstiegen und Einbrüchen der NBF-Verfahren, wie beispielsweise in Abbildung 4.2 bei PO von $0,8$ nach $0,85$ und in Abbildung 4.8 an LRT im Bereich $0,85$ bis $0,9$ zu sehen, handelt es sich mathematisch gesehen eigentlich um Sprungstellen. Da der Verlauf der Graphen für die Untersuchungen ausreichend genau zu erkennen ist, wurde darauf verzichtet, mit der Simulation die exakten Sprungstellen zu bestimmten.

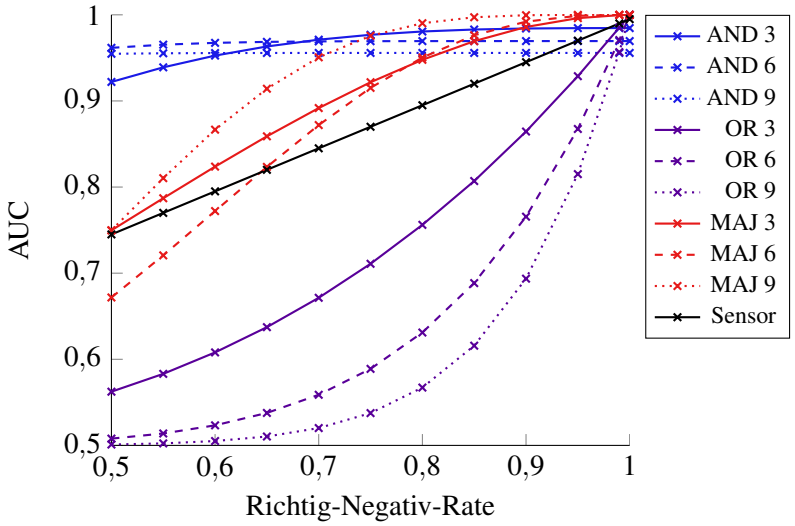

Abbildung 4.5: AUC-Werte der Abstimmungsverfahren mit je drei, sechs und neun Sensoren mit einer festen Richtig-Positiv-Rate von $0,99$.

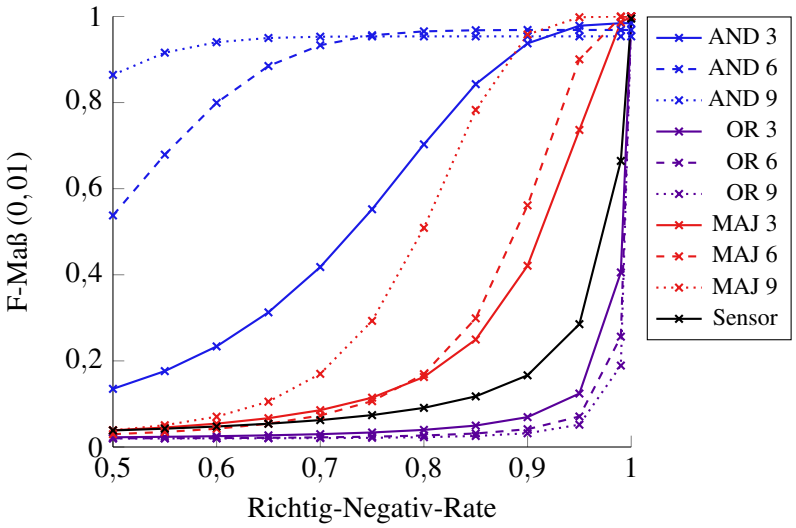

Abbildung 4.6: F-Maß Bewertungen für die Abstimmungsverfahren mit je drei, sechs und neun Sensoren mit einer festen Richtig-Positiv-Rate von 0,99.

Richtig-Positiv-Rate fest 99% In den Abbildungen 4.5 und 4.6 sind die Ergebnisse der Abstimmungsverfahren für eine fest gewählte Richtig-Positiv-Rate von 0,99 dargestellt. Im Fall von OR ist eindeutig zu erkennen, dass auch in dieser Variante weniger Sensoren bessere Ergebnisse liefern, wobei die Fusionsmethode, mit Ausnahme von einer Richtig-Negativ-Rate nahe bei 1, dem einzelnen Sensor unterlegen ist. Bei AND liefern zunächst neun bzw. sechs Sensoren die besseren Werte, aber mit steigender Richtig-Negativ-Rate ändert sich die Reihenfolge, sodass bei hohen Werten weniger Sensoren die besseren Bewertungen erhalten. Ähnlich verhält es sich bei MAJ, dort sind zwar neun Sensoren sechs Sensoren überlegen, aber bei niedriger Richtig-Negativ-Rate liefern durchaus auch drei Sensoren die besseren Ergebnisse. Doch bei steigender Spezifität fällt die drei-Sensoren-Variante zunächst hinter die neun-Sensoren- und anschließend hinter die sechs-Sensoren-Varianten zurück.

Die beiden Abbildungen 4.7 und 4.8 zeigen die Ergebnisse für die NBF-Methoden. Bei LRT und PO sind, mit wenigen Ausnahmen bei PO, mehr Sensoren einer geringeren Sensorenzahl überlegen. Wie in Abbildung 4.7 zu sehen, gibt es kleine Bereiche, in denen eine PO-Variante mit weniger Sensoren bessere Werte erzielt. Zwischen 0,55 und 0,65 sind die AUC-Werte von sechs Sensoren höher als die von neun Sensoren und im Bereich von 0,75 bis 0,8 ist die Bewertung von drei Sensoren besser als die von sechs Sensoren. Das hängt damit zusammen, dass PO und auch LRT im Extremfall nur einen Angriff erkennen, wenn alle Sensoren einen Angriff

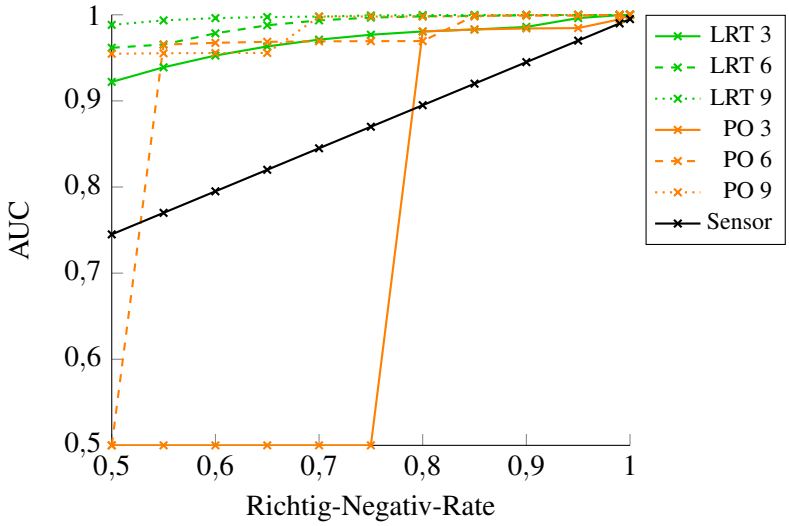

Abbildung 4.7: AUC-Werte der NBF-Verfahren mit je drei, sechs und neun Sensoren mit einer festen Richtig-Positiv-Rate von $0,99$.

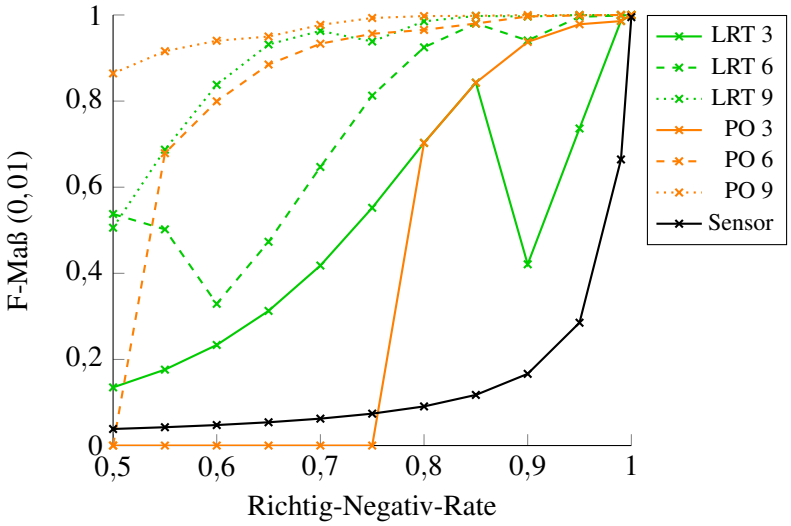

Abbildung 4.8: F-Maß Bewertungen für die NBF-Verfahren mit je drei, sechs und neun Sensoren mit einer festen Richtig-Positiv-Rate von $0,99$.

melden. (Dieser Fall entspricht den Werten von AND, weshalb sowohl LRT als auch PO sich in Teilstrecken mit AND überlagern.) Wenn Sensoren mit höheren Erkennungsraten verwendet werden, kann es passieren, dass die Anzahl der Sensorenbelegungen, die als Indikator für einen Angriff trainiert werden, ansteigt. Im Gegensatz zu den Abstimmungsverfahren, bei denen die Anzahl der Sensorenbelegungen, die für eine Angriffsmeldung nötig sind, unabhängig von der Erkennungsrate der Sensoren ist, kann es so zu Sprüngen kommen, wenn die Erkennungsraten der Sensoren angehoben wird. Da diese Sprünge nicht für jede Sensorenanzahl bei den gleichen Erkennungsraten auftritt, führt das dazu, dass teilweise eine Variante besser ist, bis die anderen ebenfalls ihren Sprung gemacht haben. Die Einbrüche bei den F-Maß-Werten von LRT in Abbildung 4.8 lassen sich auf ähnliche Weise erklären. An jeder Sprungstelle erhöht sich die Anzahl der Fälle von Sensorenbelegungen, die für einen Angriff sprechen. Entsprechend steigt die Sensitivität, allerdings auf Kosten der Spezifität, da die Anzahl der Fälle die gegen einen Angriff sprechen nun geringer ist. Ohne die Wahrscheinlichkeit eines Angriffs zu berücksichtigen verbessert sich so die Erkennungsrate, wie an den AUC-Werten in Abbildung 4.7 zu sehen ist. Offensichtlich wirkt sich diese Änderung aber zunächst sehr negativ aus, wenn die Rate der Angriffe in die Bewertung mit einbezogen wird, wie es beim F-Maß der Fall ist.

Richtig-Negativ-Rate fest 99% Die AUC-Kurven von AND und OR in Abbildung 4.9 sind gegenüber den Kurven aus Abbildung 4.5 genau vertauscht: AND hat nun die AUC-Bewertungen von OR aus dem vorherigen Abschnitt und umgekehrt. Wie bei den Szenarien mit gleicher Sensitivität und Spezifität, sind hier, aber über zwei Szenarien hinweg, die Erkennungsraten getauscht. Die Wahrscheinlichkeit, dass bei einer festen Richtig-Positiv-Rate von $0,99$ alle Sensoren einen Angriff melden, wenn ein Angriff vorliegt, entspricht genau der Wahrscheinlichkeit, dass bei einer festen Richtig-Negativ-Rate von $0,99$ alle Sensoren Normal melden, wenn kein Angriff vorliegt. Das bedeutet, dass die Sensitivität von AND aus der vorherigen Versuchsreihe genau der Spezifität von OR in diesen Simulationen entspricht und die Spezifität von AND nun der Sensitivität von OR. Trotz derselben AUC-Werte unterscheiden sich die beiden Methoden im F-Maß (Abbildung 4.10) deutlich von der jeweils anderen aus der vorherigen Versuchsreihe (Abbildung 4.6). Hier ist, bei steigender Richtig-Positiv-Rate, AND der OR Fusionsmethode wieder überlegen, da die besseren AUC-Werte für OR primär auf die hohe Sensitivität zurückzuführen sind. Sowohl für AND als auch für OR sind weniger Sensoren wieder besser, mit der Einschränkung, dass dies bei den AUC-Kurven von OR nur zutrifft, wenn sich die Richtig-Positiv-Rate 1 nähert.

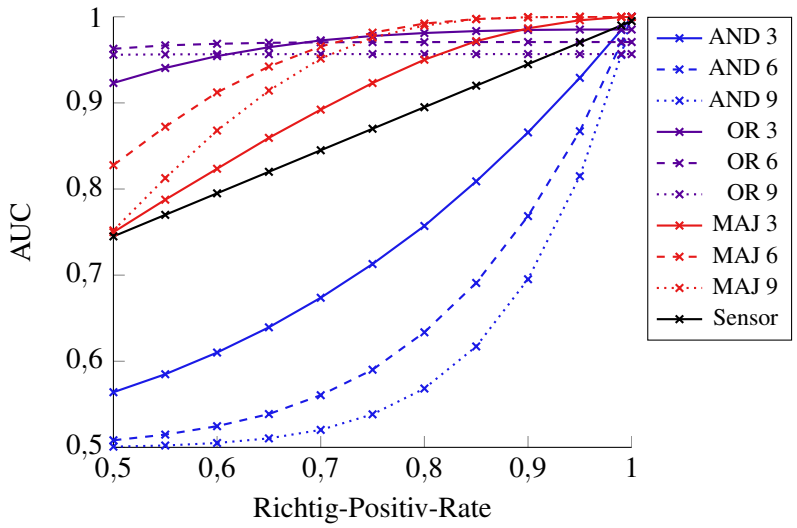

Abbildung 4.9: AUC-Werte der Abstimmungsverfahren mit je drei, sechs und neun Sensoren mit einer festen Richtig-Negativ-Rate von $0,99$.

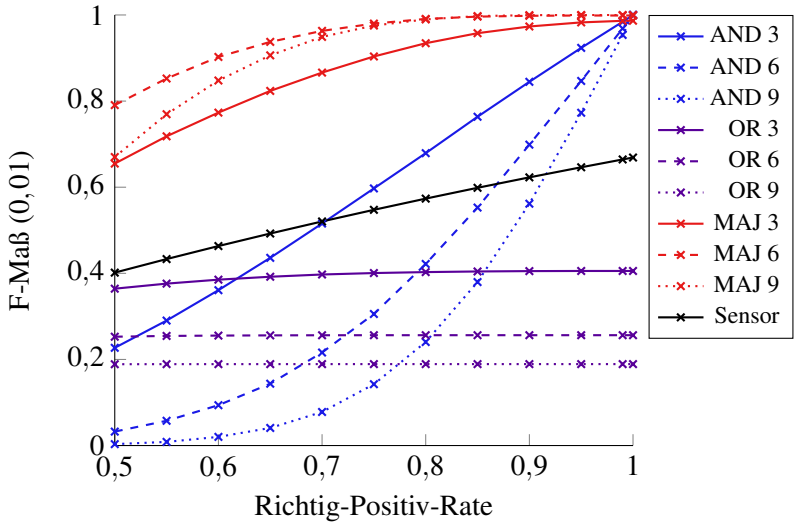

Abbildung 4.10: F-Maß Bewertungen für die Abstimmungsverfahren mit je drei, sechs und neun Sensoren mit einer festen Richtig-Negativ-Rate von $0,99$.

Auffallend ist, dass die F-Maß-Werte des einzelnen Sensors und der OR Fusionsmethode in Abbildung 4.10 weniger die Form einer Kurve haben, sondern nur eine vergleichsweise niedrige Steigung aufweisen bzw. fast eine waagerechte Gerade bilden. Der Grund dafür liegt in der festgesetzen Spezifität. Zum einen sind so die anfänglichen Werte höher als in den vorherigen Simulationen, weshalb der Graph relativ gerade beginnt. Zum anderen verhindert ein fester Wert von $0,99$, dass, im Gegensatz zu den vorherigen Fällen, ein F-Maß nahe an 1 erreicht wird, weshalb der Graph mit zunehmender Richtig-Positiv-Rate nur leicht ansteigt.

Im Fall von MAJ mit drei und neun Sensoren und auch dem einzelnen Sensor sind die AUC-Kurven in Abbildung 4.9 direkt mit denen aus Abbildung 4.5 identisch. Die sechs-Sensoren-Variante von MAJ liegt aber sichtbar über ihrem Gegenstück. Das liegt daran, dass es bei einer geraden Anzahl von Sensoren keine absolute Mehrheit gibt und hier ein Gleichstand als Angriff gewertet wird. Die MAJ-Kurven verhalten sich beim F-Maß (Abbildung 4.10) wie schon im AUC-Diagramm: Zunächst ist die sechs-Sensoren-Version den anderen beiden überlegen, aber bei zunehmender Sensitivität liegt die neun-Sensoren-Version, kaum sichtbar, vorne.

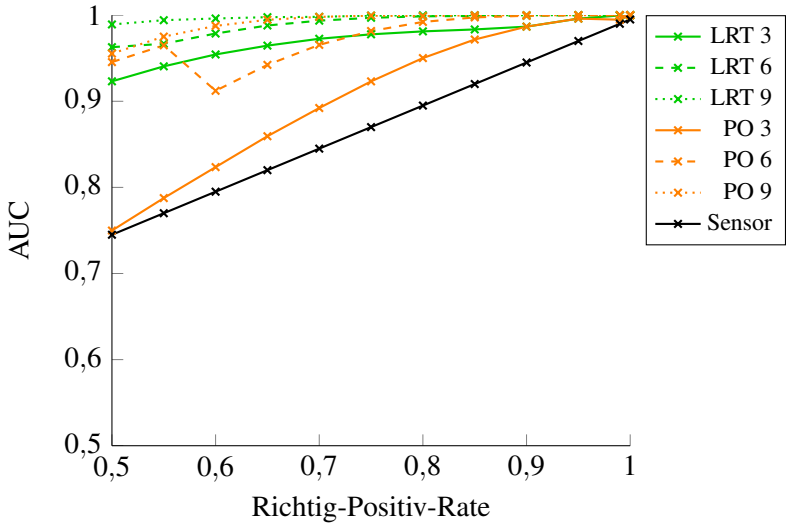

Abbildung 4.11: AUC-Werte der NBF-Verfahren mit je drei, sechs und neun Sensoren mit einer festen Richtig-Negativ-Rate von $0,99$.

Die Bewertungen der beiden NBF-Methoden sind in Abbildung 4.11 und Abbildung 4.12 gezeigt. Bedingt durch die Sprünge im F-Maß gilt für LRT nicht immer, dass mehr Sensoren überlegen sind, aber dennoch trifft dies für die meisten Fälle zu. Im Fall von PO erreicht hier die größere Anzahl von Sensoren bei beiden Bewertungsmetriken immer die besseren Bewertungen.

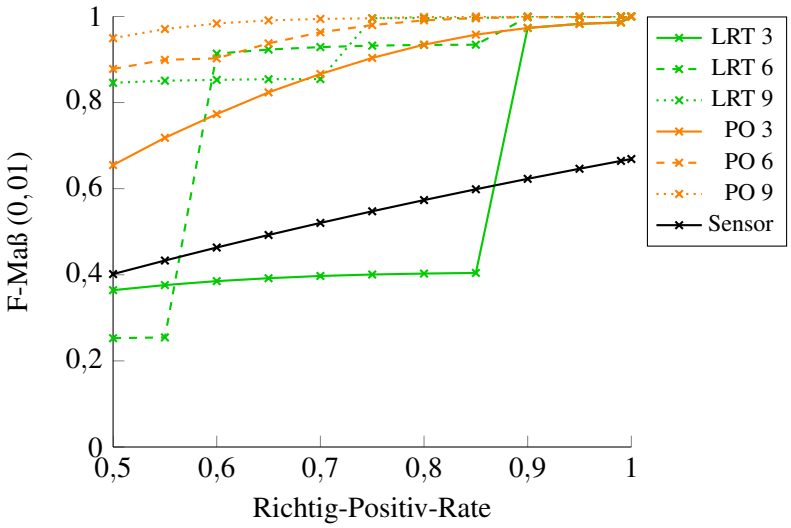

Abbildung 4.12: F-Maß Bewertungen für die NBF-Verfahren mit je drei, sechs und neun Sensoren mit einer festen Richtig-Negativ-Rate von 0,99.

4.2.1.2 Korrelierte Sensoren

Für diese Versuchsreihe werden positiv korrelierte Sensoren mit den Korrelationskoeffizienten 0,25, 0,5 und 0,75 simuliert. Nicht korrelierte Sensoren entsprechen den bereits im letzten Abschnitt simulierten bedingt unabhängigen Sensoren und vollständig korrelierte Sensoren entsprechen einem einzelnen Sensor, da sie alle immer dieselbe Aussage treffen. Negativ korrelierte Sensoren werden nicht betrachtet, da, wenn ein Sensor „immer“ das Gegenteil von einem anderen Sensor behauptet, die Vermutung nahe liegt, dass einer der beiden Sensoren fehlerhaft (konfiguriert) ist.

Abbildungen 4.13 bis 4.15 zeigen an einigen Beispielen, wie sich positiv korrelierte Sensoren auf die Fusionsmethoden auswirken können. Erwartungsgemäß nähern sich alle Graphen der Kurve des einzelnen Sensors an, wenn die Korrelation der Sensoren steigt. Für OR gilt, wie schon bei den bedingt unabhängigen Sensoren, dass eine geringere Anzahl an Sensoren die besseren Ergebnisse liefert. Bei den anderen vier Fusionsmethoden hingegen ändert sich teilweise die Reihenfolge beim F-Maß. Der Grund dafür ist, dass sich alle Kurven mit zunehmender Korrelation der Kurve des einzelnen Sensors angleichen. Da sich die ursprünglichen Kurven zum Teil stark in ihrem Verlauf unterscheiden, nähern sie sich auf unterschiedliche Weise der Kurve des einzelnen Sensors an. In Abbildung 4.14 ist zu erkennen, dass sich bei AND mit neun Sensoren bei steigenden Korrelationskoeffizienten zunächst die Kurve positiv auf der y-Achse

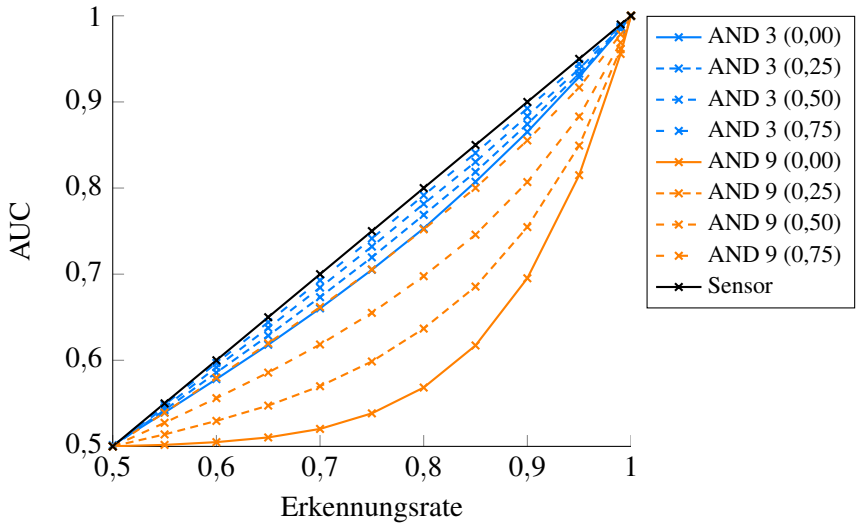

Abbildung 4.13: AUC-Werte für AND mit drei und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate bei unterschiedlichen Korrelationskoeffizienten.

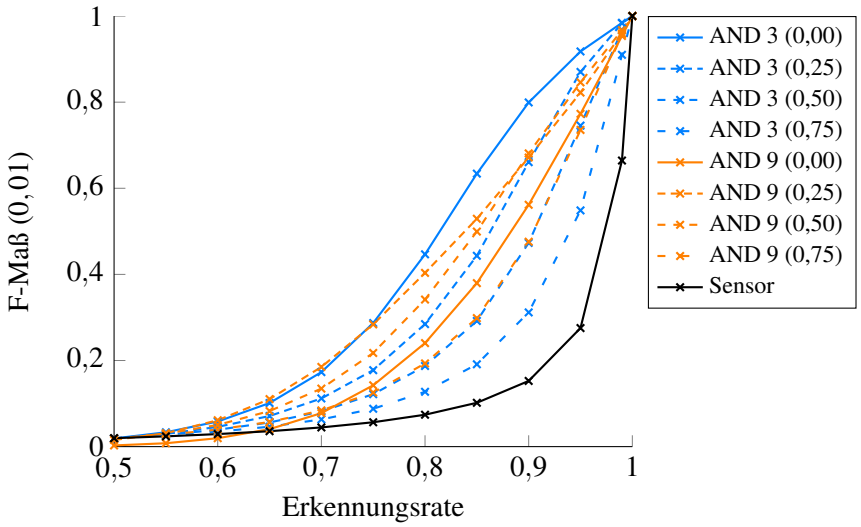

Abbildung 4.14: F-Maß Bewertungen für AND mit drei und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate bei unterschiedlichen Korrelationskoeffizienten.

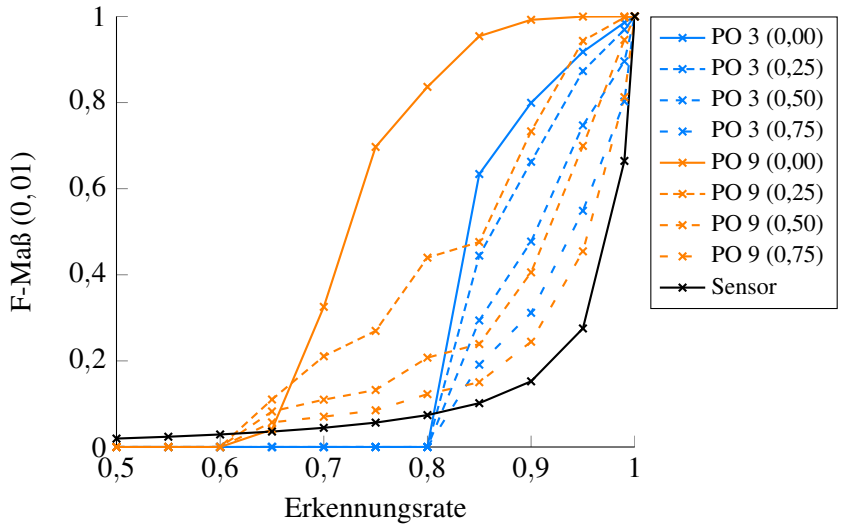

Abbildung 4.15: F-Maß Bewertungen für PO mit drei und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate bei unterschiedlichen Korrelationskoeffizienten.

verschiebt, um sich dem Wert des einzelnen Sensors bei $0,5$ zu anzugleichen. Anschließend senkt sich die Kurve ab, bis der Verlauf des einzelnen Sensors erreicht ist. Diese Unterschiede können dazu führen, dass sich für bestimmte Korrelationskoeffizienten die bevorzugte Anzahl an Sensoren einer Fusionsmethode ändert oder auch, dass mit den korrelierten Sensoren sogar bessere Bewertungen als mit unkorrelierten Sensoren erreicht werden. Beispielsweise zeigt Abbildung 4.14, dass für AND mit drei Sensoren die weniger stark korrelierten Sensoren überlegen sind. Bei neun Sensoren sind allerdings Sensoren mit einem Korrelationskoeffizienten von $0,25$ und $0,5$ sogar besser als bedingt unabhängige Sensoren. Bei zunehmender Korrelation scheint die neun Sensoren Variante auch bessere Ergebnisse als die drei Sensoren Version zu erreichen. Ein anderes Beispiel zeigt Abbildung 4.15. Dort sind für neun Sensoren im Bereich von $0,6$ bis $0,7$ ebenfalls die korrelierten Sensoren überlegen. In den übrigen Bereichen erreichen bei drei und neun Sensoren aber die weniger stark korrelierten Sensoren bessere Bewertungen. Auch wenn bei niedrigeren Korrelationskoeffizienten neun Sensoren überlegen sind, so fällt bei den Koeffizienten $0,5$ und $0,75$ die neun-Sensoren-Variante ab einer Erkennungsrate von etwa $0,85$ über weite Bereiche hinter die mit nur drei Sensoren zurück.

Insgesamt lässt sich schwer sagen, wann welche Anzahl von Sensoren die bessere Wahl ist. Aufgrund der durchgeführten Tests ist davon auszugehen, dass bei zunehmender Korrelation die

Reihenfolge gegenüber den bedingt unabhängigen Simulationen vertauscht ist und bei den Abstimmungsverfahren AND und OR mehr Sensoren und bei MAJ und den beiden NBF-Verfahren weniger Sensoren die besseren Ergebnisse liefern.

4.2.2 Abweichende Erkennungsraten im Training

Da diese Versuchsreihe eine Trainingsphase voraussetzt, werden nur die beiden NBF-Methoden untersucht. Die verwendeten Szenarien sind in zwei Gruppen unterteilt. Für die erste Gruppe werden die Erkennungsraten in der Simulationsphase gegenüber den Raten aus dem Training um $0,05$, $0,1$ und $0,15$ erhöht bzw. reduziert. Bei der zweiten Gruppe wird umgekehrt vorgegangen: Die Erkennungsraten im Training werden gegenüber der Simulationsphase um dieselben Werte erhöht bzw. verringert. Wobei für beide Gruppen die veränderten Erkennungsraten auf $1,0$ bzw. $0,5$ als untere Grenze limitiert wurden. Eine Erkennungsrate über 100% gibt es nicht und bei Werten unter 50% würden diese Verfahren jeweils die gegenteilige Entscheidung trainieren, was effektiv zu Erkennungsraten über 50% für den jeweiligen Sensor führen würde.

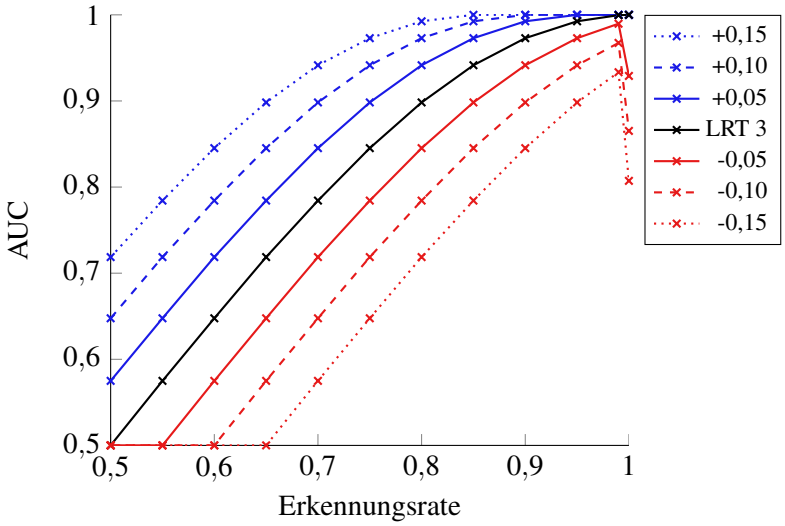

Abbildung 4.16: AUC-Werte für LRT mit drei Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate. Die Erkennungsraten sind in der Simulationsphase gegenüber der Trainingsphase um die angegebenen Werte erhöht bzw. verringert.

Nach Abbildung 4.16, gezeigt am Beispiel von LRT für drei Sensoren, wirkt es sich positiv aus, wenn die Sensoren in der Simulation bessere Erkennungsraten als im Training aufweisen, bzw. negativ, wenn die Erkennungsraten schlechter ausfallen. Der starke Rückgang in der Bewer-

tung bei einer Erkennungsrate von 1 ist darauf zurückzuführen, dass sich die Fusionsmethode durch das Training zu 100% auf die Richtigkeit der Sensorenausgaben verlassen, die für die in der Simulation schlechter ausfallenden Sensoren nicht mehr gegeben ist. Daraus lässt sich allerdings nicht direkt schließen, dass es besser wäre, die Sensoren im Training zu unterschätzen.

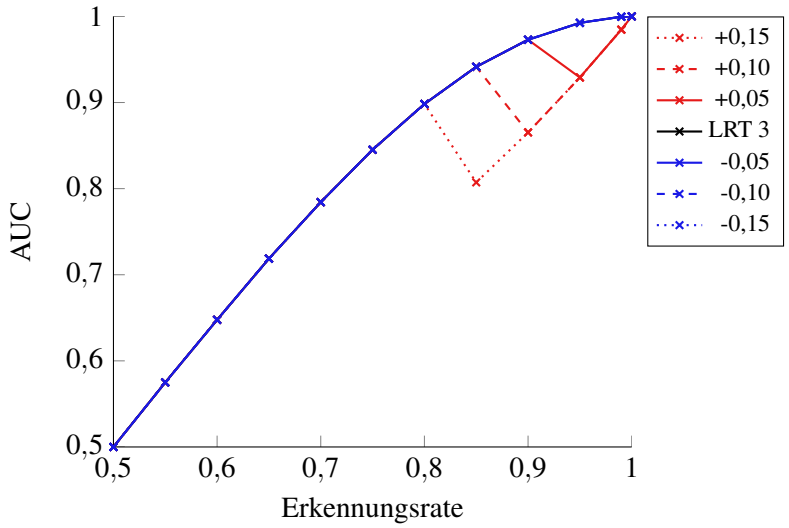

Abbildung 4.17: AUC-Werte für LRT mit drei Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate. Die Erkennungsraten sind in der Trainingsphase gegenüber der Simulationsphase um die angegebenen Werte erhöht bzw. verringert. Die Kurven der im Training unterschätzenden Szenarien überlagern sich mit der exakt trainierten Kurve.

Die Abbildungen 4.17 und 4.18 zeigen, wie es sich auswirken kann, wenn die Sensoren im Training bewusst unter- und überschätzt werden. In diesem Fall hat das Unterschätzen keinen Effekt: Die entsprechenden Kurven verlaufen exakt auf der genau trainierten Kurve. Das Überschätzen hingegen wirkt sich zumindest beim F-Maß sogar positiv aus und verbessert die Ergebnisse der Fusionsmethode. Wenn die überschätzten Erkennungsraten sehr hoch sind, geht die Richtig-Positiv-Rate stark zurück, weshalb die Anzahl der Normal-Meldungen steigt, was wiederum zu einer höheren Richtig-Negativ-Rate führt. Da die Richtig-Negativ-Rate beim F-Maß stärker ins Gewicht fällt, wirkt sich dieser „Fehler“ dort positiv aus. Die AUC-Werte berücksichtigen die Rate der Angriffe nicht, weshalb bei dieser Bewertung die Fälle mit überschätzten Erkennungsraten negativ ausfallen.

Sofern möglich, sollten die Fusionsmethoden mit Daten trainiert werden, die möglichst nahe an den zu erwartenden Daten liegen. Obwohl das bewusste Überschätzen von Erkennungsra-

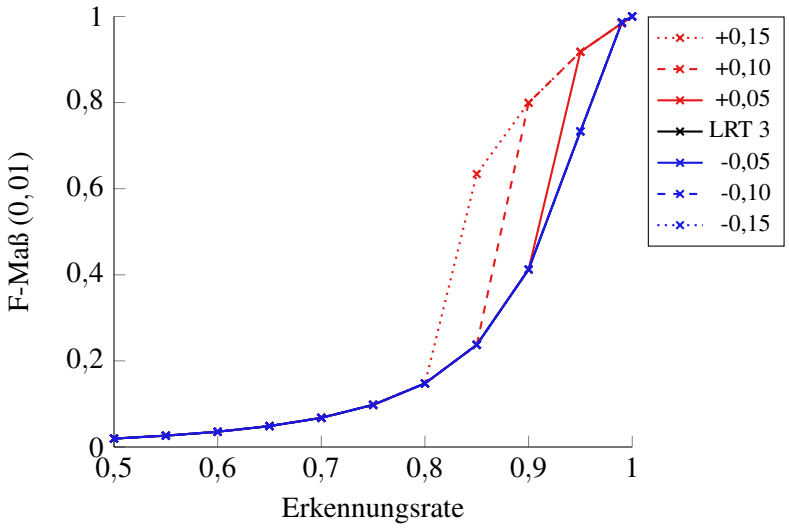

Abbildung 4.18: F-Maß Bewertungen für LRT mit drei Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate. Die Erkennungsraten sind in der Trainingsphase gegenüber der Simulationsphase um die angegebenen Werte erhöht bzw. verringert. Die Kurven der im Training unterschätzenden Szenarien überlagern sich mit der exakt trainierten Kurve.

ten sich positiv auswirken kann, sollten sie im Zweifelsfall eher unterschätzt werden. In den meisten der untersuchten Szenarien wird so die aus den Training erwartete Erkennungsrate der Fusionsmethode höchstens nach oben korrigiert. Es sollte auch darauf geachtet werden, dass im Training keine Sensoren Erkennungsraten von 100% erreichen, da, wenn diese Erkennungsraten später nicht zutreffend sind, die Erkennungsraten der Fusionsmethoden stark negativ beeinflusst werden können.

4.2.3 Ausfall von Sensoren

Für diesen Versuchsaufbau wird in der Simulationsphase die Richtig-Positiv- und die Richtig-Negativ-Rate eines Sensors auf je $0,5$ gesetzt, um einen defekten Sensor, der nur zufällig rät, zu simulieren. Im Training bleibt dieser Sensor unbeeinflusst.

Die Abbildungen 4.19 und 4.20 zeigen am Beispiel von AND bzw. PO, wie gut drei und neun Sensoren mit diesem Ausfall umgehen können. In beiden Fällen ändert sich die Rangordnung der Sensorenanzahl nicht, d.h. bei AND sind auch nach dem Ausfall weniger Sensoren besser, während bei PO mehr Sensoren die besseren Bewertungen erhalten. Ergänzend zeigen die Ab-

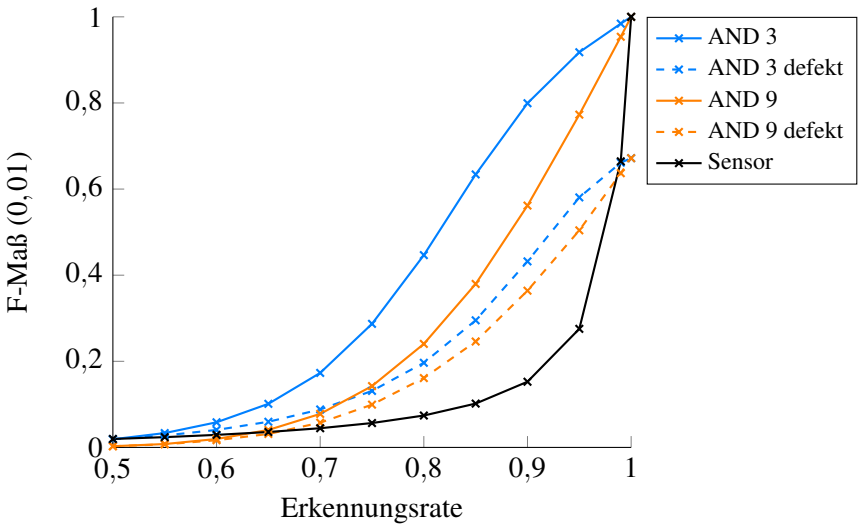

Abbildung 4.19: F-Maß Bewertungen für AND mit drei und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate. Jeweils einmal ohne und einmal mit einem ausgefallenen Sensor.

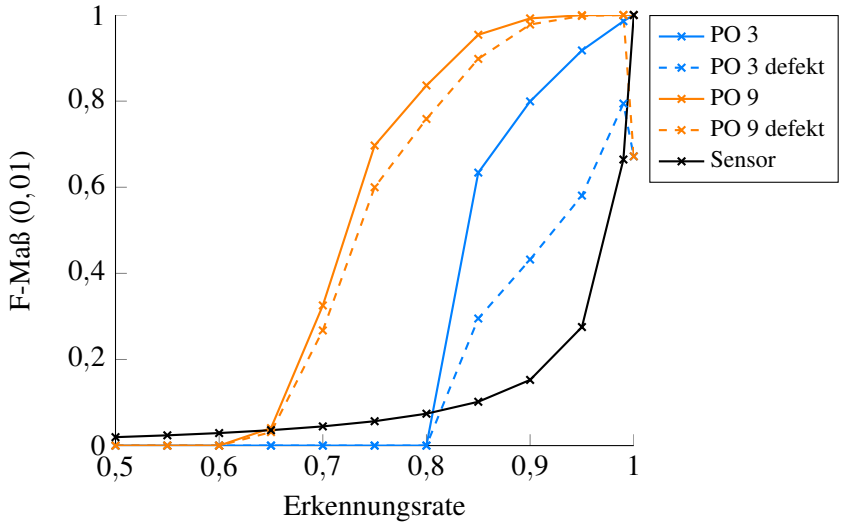

Abbildung 4.20: F-Maß Bewertungen für PO mit drei und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate. Jeweils einmal ohne und einmal mit einem ausgefallenen Sensor.

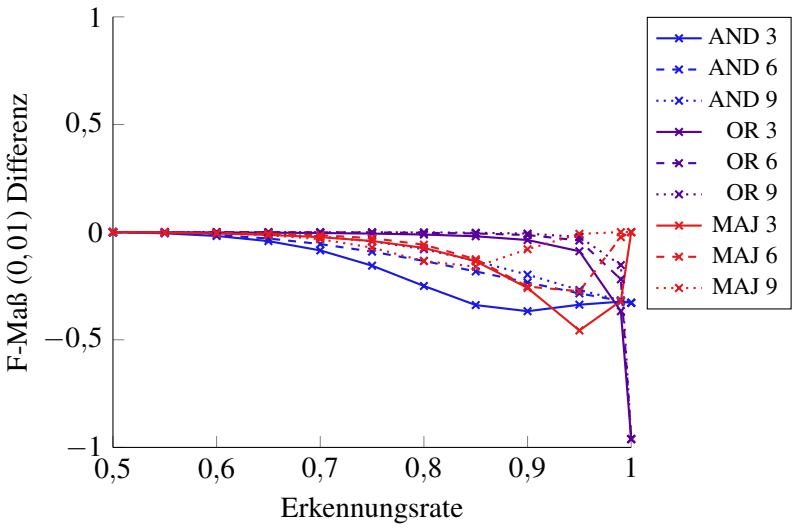

Abbildung 4.21: Durch den Ausfall eines Sensors ausgelöst Abweichungen in den F-Maß Bewertungen der Abstimmungsverfahren mit je drei, sechs und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate.

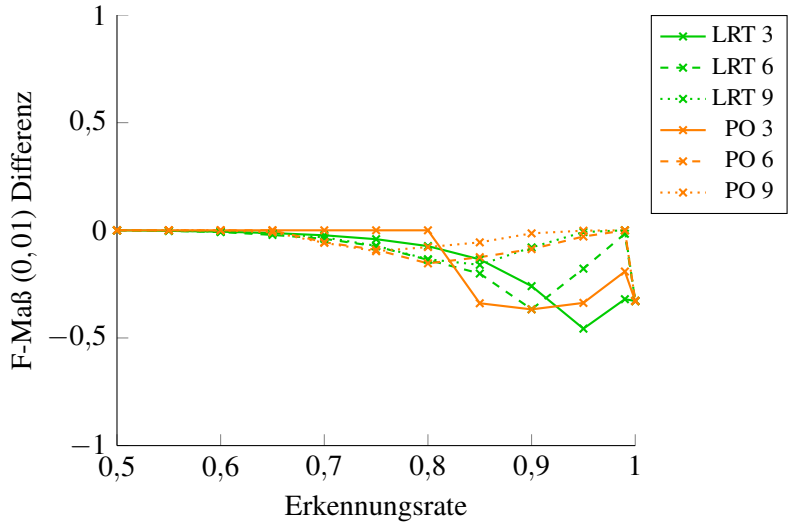

Abbildung 4.22: Durch den Ausfall eines Sensors ausgelöst Abweichungen in den F-Maß Bewertungen der NBF-Verfahren mit je drei, sechs und neun Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate.

bildungen 4.21 und 4.22 die Abweichungen von den ursprünglichen Bewertungen durch den Ausfall eines Sensors. Bei allen Verfahren, auch AND und OR, fängt ab einer Erkennungsrate von etwa $0,85$ eine höhere Anzahl von Sensoren den Ausfall besser auf. Vor diesem Punkt sind, je nach Fusionsmethode, kleinere Sensorengruppen weniger stark von einem Ausfall betroffen. In den meisten Fällen pendeln sich die Fusionsmethoden auf einem schlechteren Wert wieder ein und können den defekten Sensor zumindest zum Teil wieder auffangen. MAJ, LRT und PO schaffen es sogar an einigen Stellen den Sensor fast komplett wieder auszugleichen, wobei es nur MAJ bei sehr hohen Erkennungsraten gelingt, da die verbleibenden Sensoren irgendwann einen Punkt erreichen, an dem sie den defekten Sensor bei den meisten Fällen überstimmen können. LRT und PO brechen bei sehr hohen Erkennungsraten ein, weil sie sich vom Training her zu sehr auf die Aussage des nun defekten Sensors verlassen.

4.2.4 Spezialisierte Sensoren

In dieser Versuchsreihe werden Varianten mit sechs Sensoren untersucht, weil sich eine gerade Anzahl gut auf zwei spezialisierte Sensorenarten (à drei Sensoren) aufteilen lässt, ohne dass eine Art stärker vertreten ist. Ein spezialisierter Sensor ist dabei ein Sensor, bei dem entweder die Richtig-Positiv-Rate höher als die Richtig-Negativ-Rate ist, oder umgekehrt. Für diesen Versuch werden als Grundlage Sensoren verwendet, bei denen beide Erkennungsraten gleich sind. Von diesen Erkennungsraten wird, zu gleichen Teilen, eine erhöht und die andere verringert, um so spezialisierte Sensoren zu erhalten, die ebenfalls über die Erkennungsraten skaliert werden können. Wenn direkt die Sensoren aus den vorherigen Tests mit einer festen Rate von $0,99$ verwendet würden, wären diese von Anfang an deutlich überlegen, da die Vergleichssensoren erst bei einer Erkennungsrate von $0,5$ beginnen. Der Wert der addiert bzw. abgezogen wird verringert sich, wenn $0,5$ als untere bzw. $1,0$ als obere Grenze erreicht werden, damit die Grenzen nicht unter- bzw. überschritten werden.

Abbildung 4.23 zeigt die Ergebnisse dieser Versuche für die drei Abstimmungsverfahren. Für OR macht es nahezu keinen Unterschied ob spezialisierte Sensoren verwendet werden oder nicht. Bei AND hingegen liegen die spezialisierten Sensoren zwischenzeitlich hinter den Vergleichswerten, da die Hälfte der Sensoren eine geringere Richtig-Positiv-Rate aufweist und so die einstimmge Entscheidung negativ beeinflusst. MAJ profitiert von den spezialisierten Sensoren, da mindestens die Hälfte richtig liegen muss, damit die Entscheidung der Fusion stimmt und eben genau eine Hälfte auf das Erkennen von Angriffen und die andere Hälfte auf das Erkennen von Nicht-Angriffen spezialisiert ist.

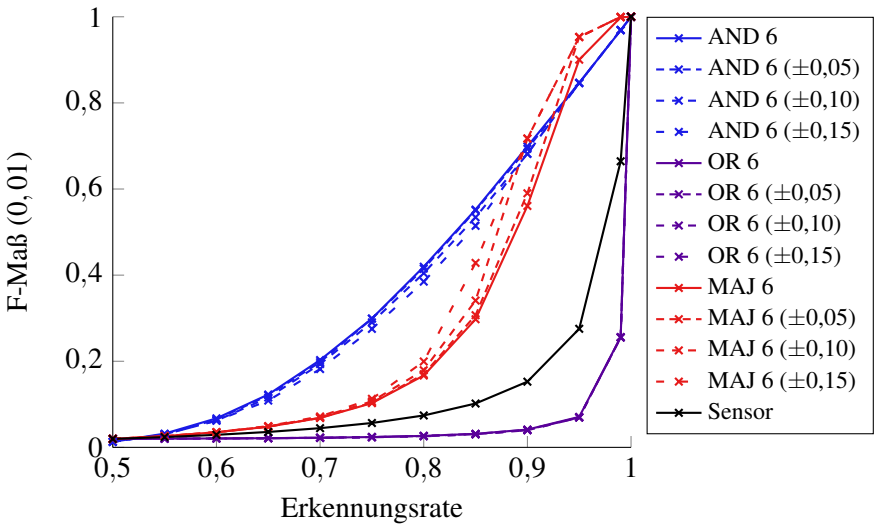

Abbildung 4.23: F-Maß Bewertungen der Abstimmungsverfahren mit sechs spezialisierten Sensoren. Als Vergleichswert dienen Kurven mit sechs unveränderten Sensoren.

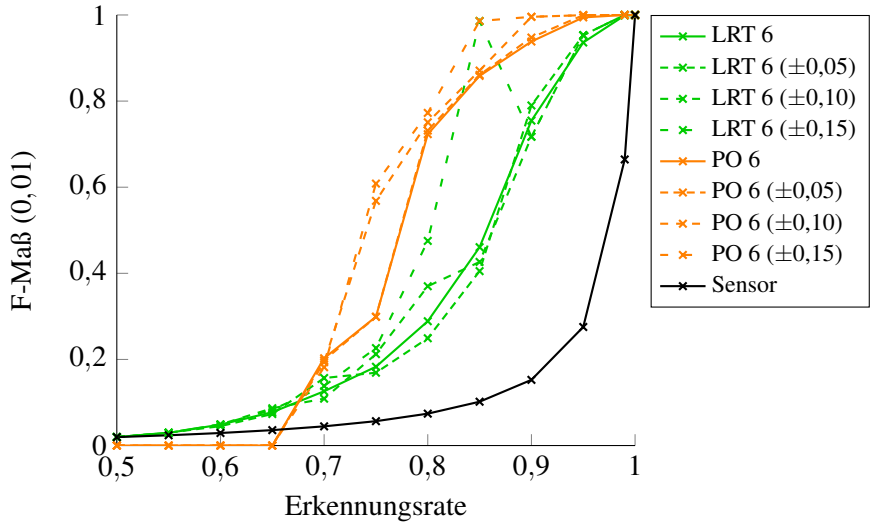

Abbildung 4.24: F-Maß Bewertungen der NBF-Verfahren mit sechs spezialisierten Sensoren. Als Vergleichswert dienen Kurven mit sechs unveränderten Sensoren.

Die F-Maß-Bewertungen der beiden NBF-Verfahren sind in Abbildung 4.24 dargestellt. PO profitiert erkennbar von den spezialisierten Sensoren. Die Kurven von LRT hingegen zeigen, dass die spezialisierten Sensoren in bestimmten Fällen eine deutliche Verbesserung in der Bewertung bewirken, aber teilweise auch die umgekehrte Auswirkung haben können. Die AUC-

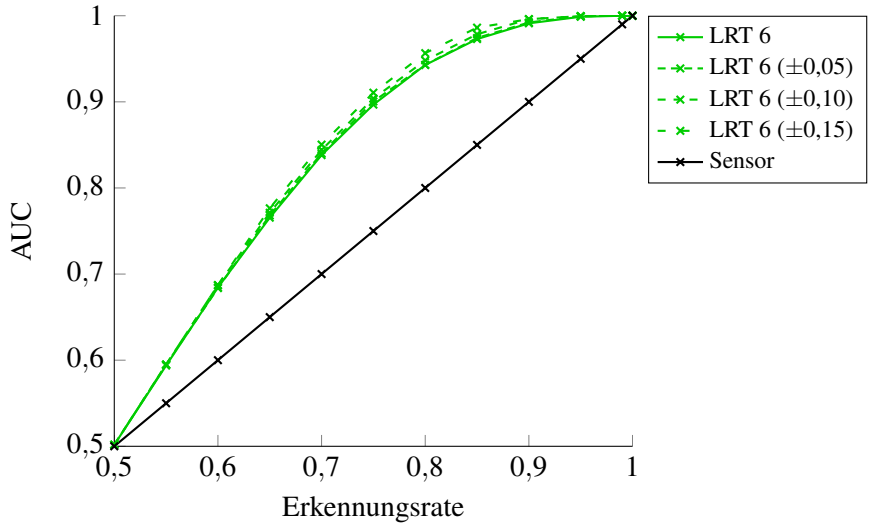

Abbildung 4.25: AUC-Werte für LRT mit sechs spezialisierten Sensoren. Als Vergleichswert dient eine Kurve mit sechs unveränderten Sensoren.

Kurve von LRT (Abbildung 4.25) zeigt im Vergleich dazu, dass spezialisierte Sensoren doch einen positiven Einfluss auf dieses Verfahren haben können, wenn in der Bewertung die Angriffsrate außer Acht gelassen wird. Da sich für LRT aus den Testfällen auf keine eindeutige Empfehlung schließen lässt, muss also je nach Anwendungsfall untersucht werden, ob der Einsatz von spezialisierten Sensoren eine Verbesserung bewirkt.

4.2.5 Zusammenfassung und Diskussion

Bis auf wenige Ausnahmen, lagen die Bewertungen von OR immer unter denen des einzelnen Sensors. Die Auswahl der Sensoren hat zwar Einfluss auf diese Fusionsmethode, ändert allerdings nichts daran, dass sie für die Angriffserkennung ungeeignet ist. AND liegt im F-Maß über dem einzelnen Sensor und ist in einigen Fällen sogar die beste Wahl unter den getesteten Fusionsmethoden. Sowohl AND als auch OR liefern bessere Ergebnisse, wenn die Anzahl der Sensoren niedrig gehalten wird. Dadurch sind sie in gewisser Weise limitiert, da die Anzahl nicht beliebig weit reduziert werden kann, um weitere Verbesserungen zu erzielen. Im Gegensatz dazu

profitieren MAJ, LRT und PO von einer steigenden Sensorenanzahl und liefern auch bei einer geringen Anzahl bereits bessere Ergebnisse als ein einzelner Sensor.

Der Grund dafür, dass AND und OR mit einer geringeren Anzahl von Sensoren bessere Bewertungen erzielen, ist, dass beide Methoden auf eine einstimmge Entscheidung zurückgreifen. AND benötigt eine Angriffsmeldung von allen Sensoren, um selbst einen Angriff zu melden, während bei OR alle Sensoren Normal melden müssen, damit die Fusionsmethode keinen Angriff meldet. Die anderen drei Fusionsmethoden sind in dieser Hinsicht flexibler und können bereits aus einem Teil der Sensorenausgaben auf eine entsprechende Entscheidung schließen. Bei sehr stark korrelierten Sensoren treffen diese Annahmen über die Auswirkung der Sensorenanzahl allerdings nicht mehr zu. So kann es beispielsweise bei einem Korrelationskoeffizienten von $0,75$ für AND besser sein neun statt drei Sensoren zu verwenden (siehe Abbildung 4.14). Aber auch bei bedingt unabhängigen Sensoren gibt es nicht immer eine Methode oder Sensorenanzahl, die allen anderen überlegen ist. Bei drei Sensoren mit gleicher Richtig-Positiv- und Richtig-Negativ-Rate ist nach F-Maß AND die beste Wahl (siehe Abbildung 4.3 bzw. 4.4), während in den meisten anderen Fällen PO die besseren Ergebnisse erzielt. Abbildung 4.12 zeigt, dass es bei LRT Fälle gibt, in denen sechs Sensoren bessere Bewertungen erreichen als neun. Entsprechend lässt sich keine allgemeine Aussage treffen, welches Verfahren und welche Anzahl an Sensoren verwendet werden sollte. Die einzige Lösung, um eine passende Auswahl zu treffen, ist, die Verfahren unter den Rahmenbedingungen der geplanten Einsatzumgebung zu vergleichen, wie von Gu et al. [GFD+06] für den Vergleich von IDS vorgeschlagen.

Der unbemerkte Ausfall eines Sensors wirkt sich auf alle untersuchten Fusionsmethoden negativ aus. Eine größere Anzahl von Sensoren ist besser in der Lage, die Auswirkungen eines Ausfalls auszugleichen und sich den ursprünglichen Erkennungsraten der Fusionsmethode wieder anzunähern. Welche Anzahl an Sensoren für welche Methode besser geeignet ist, änder sich dabei allerdings nicht. Zur Sicherheit sollte regelmäßig überprüft werden, ob alle Sensoren ordnungsgemäß funktionieren und nicht defekt sind oder sogar kompromittiert wurden.

Die drei Fusionsmethoden, die nicht auf einer einstimmigen Entscheidung aufbauen, MAJ, LRT und PO, können von spezialisierten Sensoren profitieren. AND und OR ziehen keinen Nutzen aus dieser Art von Sensorenzusammenstellung bzw. können dadurch sogar negativ beeinflusst werden. Abhängig von der zum Einsatz kommenden Fusionsmethode kann es also durchaus sinnvoll sein, eine Kombination aus Sensoren zu verwenden, die eher auf Angriffe und Nicht-Angriffe spezialisiert sind.

Für die beiden untersuchten NBF-Methoden hat sich herausgestellt, dass es besser ist, die Erkennungsraten der Sensoren im Training zu unter- als zu überschätzen. Allerdings werden die besten Ergebnisse erzielt, wenn die Trainingsdaten möglichst nahe an den zu erwartenden Daten liegen.

5 Zusammenfassung und Ausblick

Die Zielsetzung der Arbeit war es zu untersuchen, wie sich verschiedene Sensoreneigenschaften bzw. die Art der Zusammenstellung der Sensoren auf Fusionsmethoden in der IDS-Domäne auswirken. Dazu wurde die in Kapitel 3 beschriebene Simulationsumgebung entwickelt, mit deren Hilfe unterschiedliche Szenarien simuliert und bewertet werden können. In Kapitel 4 wurde die Simulationsumgebung zunächst validiert und anschließend Experimente durchgeführt und ausgewertet.

In Abschnitt 5.1 wird zunächst auf einige zentrale Punkte der Simulationsumgebung eingegangen und anschließend werden die Ergebnisse der Experimente zusammengefasst. Der darauffolgende Abschitt 5.2 liefert einen Ausblick auf mögliche Erweiterungen der Simulationsumgebung und gibt Anregungen für weitere Szenarien bzw. Fragestellungen.

5.1 Zusammenfassung

Die Grundlage zur Untersuchung der Fragestellung dieser Arbeit bildet die in Kapitel 3 entwickelte Simulationsumgebung. Da der Fokus auf den Fusionsmethoden und nicht den einzelnen Sensoren bzw. IDS selbst liegt, werden in der Simulation direkt die Entscheidungen der einzelnen Sensoren generiert und keine Daten, die erst durch die Sensoren bewertet werden müssen. Eine wichtige Rolle übernimmt dabei die Generierung von Pseudozufallszahlen. Für jedes Szenario wird durch eine Konfiguration vorgegeben, wie sich die einzelnen Sensoren in welchen Situationen verhalten sollen. Mit Hilfe von Pseudozufallszahlen werden entsprechend der Konfiguration die Entscheidungen der Sensoren simuliert. Dabei besteht die Möglichkeit, neben bedingt unabhängigen Sensoren auch korrelierte Sensoren zu verwenden. Die so erzeugten Sensorenausgaben werden an die eingestellte Fusionsmethode übergeben, die ihrerseits eine Entscheidung trifft. Anschließend werden die Ergebnisse in einer Datenbank für die spätere Auswertung gesichert. Bei den implementierten Fusionsmethoden wurde darauf geachtet, dass die mathematische Herleitung der Ergebnisse nicht zu komplex ist, da sie zur späteren Validierung der gesamten Simulationsumgebung verwendet werden. Neben den leicht verständlichen

Abstimmungsverfahren *einstimmige Entscheidung*, *mindestens einer* und *relative Mehrheit* (Abschnitt 2.2.2) wurden zwei *naive Bayes*-basierte Methoden, *likelihood ratio test* und *posterior odds* (Abschnitt 2.2.3), umgesetzt. Für Fusionsmethoden, die zunächst trainiert werden müssen, bevor sie Entscheidungen treffen können, wie beispielsweise die beiden *naive Bayes*-basierten Verfahren, stellt die Simulationsumgebung eine Trainingsphase vor der eigentlichen Simulationsphase bereit. Die Einstellungen für die Trainingsphase können unabhängig von der Simulationsphase konfiguriert werden, um Abweichungen untersuchen zu können.

Um mit der Simulationsumgebung die Auswirkungen von unterschiedlichen Sensoreneigenschaften zu untersuchen, wurden mehrere Versuchsreihen aufgestellt und mit variierenden Werten für alle betrachteten Fusionsmethoden simuliert. Für jedes dieser Szenarien wurden dabei die Erkennungsraten der Sensoren schrittweise von 50% bis auf 100% gesteigert, um einen Eindruck des Verlaufs der einzelnen Fusionsmethoden zu gewinnen. Als Bewertungskriterien wurde auf die AUC und das F-Maß zurückgegriffen (Abschnitt 2.3). Allerdings waren die Ergebnisse des F-Maßes ausdrucksstärker, da diese Metrik die Rate der Angriffe mit einbezieht. Während in einigen Fällen unterschiedliche Fusionsmethoden dieselben AUC-Werte erreichten, war bzgl. des F-Maßes doch meist eine Methode überlegen.

In dieser Arbeit wurde gezeigt, dass die am besten für eine Fusionsmethode geeignete Sensorenanzahl nicht nur von der verwendeten Methode selbst, sondern auch von der Korrelation der einzelnen Sensoren abhängig ist. So kann es bei sehr stark korrelierten Sensoren besser sein, weniger Sensoren einzusetzen, auch wenn bei bedingt unabhängigen Sensoren, eine möglichst große Anzahl an Sensoren die besseren Erkennungsraten erreicht. Die primäre Erkenntnis aus den Experimenten ist entsprechend, dass von den untersuchten Fusionsmethoden keine eindeutig überlegen ist. Abhängig von der Anzahl der verwendeten Sensoren und wie stark diese korreliert sind, eignen sich, je nach erreichter Erkennungsrate der Sensoren, einige Fusionsmethoden eher als andere. Entsprechend sollte zum Vergleich der Fusionsmethoden die Konfiguration so gewählt werden, dass sie möglichst mit dem beabsichtigten Einsatzgebiet übereinstimmt. So kann mit Hilfe der Simulation ermittelt werden, welcher Aufbau für genau diese Umgebung die besten Resultate liefern sollte.

Weiterhin hat sich ergeben, dass Fusionsmethoden, die nicht in irgendeiner Weise auf eine einstimmge Entscheidung der Sensoren angewiesen sind, von einer Zusammenstellung aus unterschiedlich spezialisierten Sensoren profitieren können. Für die beiden untersuchten *naive Bayes*-basierten Verfahren ist es ebenfalls von Vorteil, die Sensoren im Training nicht zu überschätzen. Von einem ausgefallenen Sensor sind alle Fusionsmethoden negativ betroffen. Eine

größere Anzahl von Sensoren kann einen Ausfall besser ausgleichen, erreicht aber nur in sehr wenigen Fällen die ursprünglichen Erkennungsraten. Entsprechend sollte ein defekter Sensor möglichst schnell bemerkt und wiederhergestellt werden.

5.2 Ausblick

Die Simulationsumgebung ist bereits in der Lage, statt allgemeinen Angriffen die KDD'99 Klassen zu simulieren. Allerdings können die implementierten Verfahren nur mit den zwei Klassen Angriff und Nicht-Angriff umgehen. Um die Möglichkeiten der generierten Sensorenausgaben vollständig auszunutzen, wäre ein nächster Schritt, diese Verfahren entsprechend zu erweitern oder zusätzliche Verfahren zu implementieren, die bereits mit diesen Klassen umgehen können.Weiterhin wäre es nützlich, wenn die Erkennungsraten der Sensoren nicht nur über den Erwartungswert alleine, sondern zusätzlich noch durch die Abweichung oder sogar die komplette Dichtefunktion konfiguriert werden könnten. So ließen sich beispielsweise Sensoren mit hohen Erkennungsraten, aber starken Abweichungen mit Sensoren, deren Erkennungsraten etwas niedriger, aber dafür stabiler sind, vergleichen.

Ein anderer Punkt, der noch erweitert werden sollte, ist die Auswertung. In dieser Version können für einen oder mehrere Simulationsläufe zusammen verschiedene Bewertungsmetriken berechnet werden. Wünschenswert wäre es, wenn zusätzlich mehrere dieser Auswertungen gruppiert und direkt als Graph oder einem für andere Funktionsplotter passendem Format ausgegeben werden könnten. In diesem Zusammenhang wäre auch der Umstieg von einem Kommandozeilenprogramm auf eine grafische Benutzeroberfläche sinnvoll.

Wenn die Implementierung entsprechend erweitert wurde, eröffnen die KDD'99 Klassen neue Möglichkeiten, die Sensoren zu konfigurieren und entsprechend neue Szenarien zu untersuchen. Dabei sind wahrscheinlich Versuchsreihen, die sich mit spezialisierten Sensoren auseinandersetzen, besonders interessant, da sich dort viele neue Kombinationsmöglichkeiten ergeben. Auch ohne diese Erweiterung lohnt es sich, weitere Fusionsmethoden zu untersuchen. Dabei bieten sich besonders Verfahren an, die mathematisch nur sehr schwer zu analysieren sind und daher stark von den Untersuchsungsmöglichkeiten der Simulation profitieren können.

Mit den bereits vorhandenen Fusionsmethoden lassen sich aber ebenfalls weitere Fragestellungen untersuchen. Beispielsweise könnte untersucht werden, ob es für die Anzahl der Sensoren eine Art obere Grenze gibt, bei der es kaum noch eine Verbesserung der Erkennungsraten gibt oder ob sich zuviele Sensoren sogar negativ auswirken können. Ein weiteres Beispiel wäre

der Vergleich von gleichartigen Sensoren mit einem Zusammenschluss aus Sensoren, die alle unterschiedliche Erkennungsraten haben, oder auch wie sich ein etwas schlechterer, dafür bedingt unabhängiger Sensor, auf einen Zusammenschluss aus ansonsten gleichartigen und stark korrelierten Sensoren auswirkt.

A CD-ROM

Die beiliegende CD-ROM enthält sowohl den Quellcode der in Kapitel 3 entwickelte Simulationsumgebung, als auch eine ausführbare Version. Details über die Verwendung der Software sind der Datei `readme.txt` zu entnehmen. Weiterhin befindet sich auf der CD-ROM eine digitale Version der vorliegenden Arbeit und eine vollständige Auflistung der in Kapitel 4 simulierten Szenarien inklusive der Ergebnisse. Zusätzlich sind noch einige nützliche Skripte beigefügt, die die Simulation von mehreren unterschiedlichen Szenarien in Folge erleichtern. Details zu dem Format, in dem die Szenarien mit ihren Ergebnissen vorliegen, und wie die Skripte zu benutzen sind, sind ebenfalls in der Datei `readme.txt` hinterlegt.

Literaturverzeichnis

[Alt05] Hakan Altınçay. *On Naive Bayesian Fusion of Dependent Classifiers*. *Pattern Recognition Letters*, 26(15):2463–2473, 2005.

[Axe99] Stefan Axelsson. *The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection*. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, Seiten 1–7, 1999.

[Axe00] Stefan Axelsson. *Intrusion Detection Systems: A Survey and Taxonomy*. Forschungsbericht 99–15, Department of Computer Engineering, 2000.

[AZC09] M. Ali Aydın, A. Halim Zaim und K. Gökhan Ceylan. *A Hybrid Intrusion Detection System Design for Computer Network Security*. *Computers & Electrical Engineering*, 35(3):517–526, 2009.

[BM91] Robert S. Boyer und J. Strother Moore. *MJRTY – A Fast Majority Vote Algorithm*. In Robert S. Boyer, Herausgeber, *Automated Reasoning*, Band 1 aus *Automated Reasoning Series*, Seiten 105–117. Springer, 1991.

[CGM+09] Igino Corona, Giorgio Giacinto, Claudio Mazzariello, Fabio Roli und Carlo Sansone. *Information Fusion for Computer Security: State of the Art and Open Issues*. *Information Fusion*, 10(4):274–284, 2009.

[Con02] ConSecur GmbH. *Leitfaden zur Einführung von Intrusion-Detection-Systemen*. Studie, BSI, Oktober 2002. URL: `https://www.bsi.bund.de/DE/Themen/Cyber-Sicherheit/Themen/Sicherheitskomponenten/IntrusionDetectionSystemeIDS/intrusiondetectionsystemeids_node.html` [Stand: 18.03.2014].

[DDW99] Hervé Debar, Marc Dacier und Andreas Wespi. *Towards a Taxonomy of Intrusion-Detection Systems*. *Computer Networks*, 31(8):805–822, April 1999.

[DTAC05] Ozgur Depren, Murat Topallar, Emin Anarim und M. Kemal Ciliz. *An Intelligent Intrusion Detection System (IDS) for Anomaly and Misuse Detection in Computer Networks*. *Expert Systems with Applications*, 29(4):713–722, November 2005.

[Faw06] Tom Fawcett. *An Introduction to ROC Analysis*. *Pattern Recognition Letters*, 27(8):861–874, Juni 2006.

[Fla12] Peter Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[GCL08] Guofei Gu, Alvaro A. Cárdenas und Wenke Lee. *Principled Reasoning and Practical Applications of Alert Fusion in Intrusion Detection Systems*. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, Seiten 136–147, 2008.

[GFD+06] Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee und Boris Skorić. *Measuring Intrusion Detection Capability: An Information-Theoretic Approach*. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, Seiten 90–101, 2006.

[GRD03] Giorgio Giacinto, Fabio Roli und Luca Didaci. *Fusion of Multiple Classifiers for Intrusion Detection in Computer Networks*. *Pattern Recognition Letters*, 24(12):1795–1803, August 2003.

[GRF00] G. Giacinto, F. Roli und G. Fumera. *Selection of image classifiers*. *Electronics Letters*, 36(5):420–422, March 2000.

[GUCK03] Ashish Garg, Shambhu Upadhyaya, Ramkumar Chinchani und Kevin Kwiat. *SIMS: A Modeling and Simulation Platform for Intrusion Monitoring/Detection Systems*. In *Proceedings of 2003 Summer Computer Simulation Conference*, 2003.

[GVUK06] A. Garg, S. Vidyaraman, S. Upadhyaya und K. Kwiat. *USim: A User Behavior Simulation Framework for Fraining and Testing IDSes in GUI based systems*. In *Proceedings of the 39th Annual Simulation Symposium*, April 2006.

[HC03] Sang-Jun Han und Sung-Bae Cho. *Detecting Intrusion with Rule-Based Integration of Multiple Models*. *Computers & Security*, 22(7):613–623, Oktober 2003.

[HL97] David L. Hall und James Llinas. *An Introduction to Multisensor Data Fusion*. *Proceedings of the IEEE*, 85(1):6–23, 1997.

[Jav] *API Spezifikation der Java-Klasse Random*. URL: `http://docs.oracle.com/javase/7/docs/api/java/util/Random.html` [Stand: 18.03.2014].

[KBD01] Ludmila I. Kuncheva, James C. Bezdek und Robert P.W. Duin. *Decision Templates for Multiple Classifier Fusion: An Experimental Comparison*. *Pattern Recognition*, 34(2):299–314, February 2001.

[Kiz13] Joseph Migga Kizza. *Guide to Computer Network Security*. Computer Communications and Networks. Springer, 2. Auflage, 2013.

[Kle12] Lawrence A. Klein. *Sensor and Data Fusion: A Tool for Information Assessment and Decision Making*. SPIE Press, 2. Auflage, 2012.

[Knu02] Donald Ervin Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, 3. Auflage, 2002.

[Kol08] Michael Kolonko. *Stochastische Simulation: Grundlagen, Algorithmen und Anwendungen*. Vieweg+Teubner, 1. Auflage, 2008.

[Kun04] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[KV02] Richard A. Kemmerer und Giovanni Vigna. *Intrusion Detection: A Brief History and Overview*. *Computer*, 35(4):27–30, April 2002.

[KZH05] H. Güneş Kayacık und Nur Zincir-Heywood. *Generating Representative Traffic for Intrusion Detection System Benchmarking*. In *Proceedings of the 3rd Annual Communication Networks and Services Research Conference*, Seiten 112–117, Mai 2005.

[LHF+00] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba und Kumar Das. *Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation*. In Hervé Debar, Ludovic Mé und S.Felix Wu, Herausgeber, *Recent Advances in Intrusion Detection*, Band 1907 aus *Lecture Notes in Computer Science*, Seiten 162–182. Springer, 2000.

[LKS05] Aleksandar Lazarevic, Vipin Kumar und Jaideep Srivastava. *Intrusion Detection: A Survey*. In Vipin Kumar, Jaideep Srivastava und Aleksandar Lazarevic, Herausgeber, *Managing Cyber Threats*, Band 5 aus *Massive Computing*, Seiten 19–78. Springer, 2005.

[Loc93] Franz Locher. *Numerische Mathematik für Informatiker*. Mathematik für Informatiker. Springer, 2. Auflage, 1993.

[Mah03] Matthew V. Mahoney. *Network Traffic Anomaly Detection Based on Packet Bytes*. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, Seiten 346–350, 2003.

[Mar09] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC Machine Learning & Pattern Recognition. CRC Press/Taylor & Francis, 2009.

[MC01] Matthew V. Mahoney und Philip K. Chan. *PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic*. Forschungsbericht CS-2001-04, Florida Institute of Technology, 2001.

[MHL94] Biswanath Mukherjee, L. Todd Heberlein und Karl N. Levitt. *Network Intrusion Detection*. *IEEE Network*, 8(3):26–41, Mai/Juni 1994.

[MSA05] Srinivas Mukkamala, Andrew H. Sung und Ajith Abraham. *Intrusion detection using an ensemble of intelligent paradigms*. *Journal of Network and Computer Applications*, 28(2):167–182, April 2005.

[Pea94] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. The Morgan Kaufmann Series in Representation and Reasoning. Morgan Kaufmann, 2. Auflage, 1994.

[PP07] Animesh Patcha und Jung-Min Park. *An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends*. *Computer Networks*, 51(12):3448–3470, 2007.

[PZC+96] Nicholas J. Puketza, Kui Zhang, Mandy Chung, Biswanath Mukherjee und Ronald A. Olsson. *A Methodology for Testing Intrusion Detection Systems*. *IEEE Transactions on Software Engineering*, 22(10):719–729, Oktober 1996.

[RN04] Stuart Russell und Peter Norvig. *Künstliche Intelligenz: Ein moderner Ansatz*. Pearson Studium, 2. Auflage, 2004.

[Roe99] Martin Roesch. *Snort – Lightweight Intrusion Detection for Networks*. In *Proceedings of the 13th USENIX Conference on System Administration*, Seiten 229–238, 1999.

[SB12] William Stallings und Lawrie Brown. *Computer Security: Principles and Practice*. Always Learning. Pearson Education, 2. Auflage, 2012.

[SF02] Kari Sentz und Scott Ferson. *Combination of Evidence in Dempster-Shafer Theory*. Forschungsbericht SAND2002-0835, Sandia National Laboratories, 2002.

[Sha76] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[SM04] Christos Siaterlis und Basil Maglaris. *Towards Multisensor Data Fusion for DoS Detection*. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, Seiten 439–446, 2004.

[SS62] Ernest M. Scheuer und David S. Stoller. *On the Generation of Normal Random Vectors*. *Technometrics*, 4(2):278–281, Mai 1962.

[Sud03] John J. Sudano. *Equivalence Between Belief Theories and Naive Bayesian Fusion for Systems with Independent Evidential Data: Part I, The Theory*. In *Proceedings of the Sixth International Conference of Information Fusion*, Band 2, Seiten 1239–1243, 2003.

[Sun96] Aurobindo Sundaram. *An Introduction to Intrusion Detection*. *Crossroads*, 2(4):3–7, März 1996.

[TB08] Ciza Thomas und Narayanaswamy Balakrishnan. *Advanced Sensor Fusion Technique for Enhanced Intrusion Detection*. In *IEEE International Conference on Intelligence and Security Informatics*, Seiten 173–178, 2008.

[TB09] Ciza Thomas und N. Balakrishnan. *Mathematical Analysis of Sensor Fusion for Intrusion Detection Systems*. In *First International Communication Systems and Networks and Workshops*, Seiten 1–10, 2009.

[Tho09] Ciza Thomas. *Performance Enhancement of Intrusion Detection Systems using Advances in Sensor Fusion*. Dissertation, Supercomputer Education and Research Centre Indian Institute of Science, April 2009.

[VVKK04] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel und Richard A. Kemmerer. *A Comprehensive Approach to Intrusion Detection Alert Correlation*. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, 2004.

[WY01] Tao Wan und Xue Dong Yang. *IntruDetector: A Software Platform for Testing Network Intrusion Detection Algorithms*. In *Proceedings 17th Annual Computer Security Applications Conference*, Seiten 3–11, Dezember 2001.

[WYWZ04] Yong Wang, Huihua Yang, Xingyu Wang und Ruixia Zhang. *Distributed Intrusion Detection System Based on Data Fusion Method*. In *Fifth World Congress on Intelligent Control and Automation*, Band 5, Seiten 4331–4334, 2004.

[YF05] Dong Yu und Deborah Frincke. *Alert Confidence Fusion in Intrusion Detection Systems with Extended Dempster-Shafer Theory*. In *Proceedings of the 43rd annual Southeast regional conference*, Seiten 142–147, 2005.

[ZSF+07] Piero Zappi, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini und Gerhard Tröster. *Activity Recognition from On-Body Sensors by Classifier Fusion: Sensor Scalability and Robustness*. In *3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, Seiten 281–286, 2007.