Künstliche neuronale Netze zur Missbrauchserkennung in Mobilfunknetzen auf Basis von Verbindungsdaten.

Verfasst von
Alois Geith
Studienkennzahl: J151
Matrikelnummer: 9450190

Diplomarbeit
am Institut für Informationswirtschaft
an der WIRTSCHAFTSUNIVERSITÄT WIEN
Studienrichtung: Betriebswirtschaft

Begutachter: Priv.Doz. Dr. Michael Hahsler

Wien, 24. Juli 2006

Künstliche neuronale Netze zur Missbrauchserkennung in Mobilfunknetzen auf Basis von Verbindungsdaten.

Stichworte: künstliches neuronales Netz, vorwärts verkettetes Netz, Multi Layer Perzeptron, selbst organisierende Karte, Mobilfunknetz, Missbrauch, Betrug, Benutzerprofil.

Keywords: artificial neural network, feed forward network, multi layer perceptron, self organizing map, mobile network, fraud, user profile.

Zusammenfassung

Durch Betrugsfälle entgehen der Telekommunikationsbranche jährlich Milliarden Euro an Gewinnen. Mittels künstlicher neuronaler Netze, ein Ansatz der das menschliche Gehirn zum Vorbild hat, können viele dieser Fälle frühzeitig erkannt und damit auch darauf reagiert werden. Anwendungen in diesem Bereich reichen vom Erlernen bekannter Betrugsmuster bis hin zur selbständigen Kategorisierung von Transaktion und der Erstellung eines Benutzerprofils um durch Vergleich von aktuellem und historischem Profil auf Betrug bzw. Anomalien im Verhalten zu schließen.

Die vorliegende Arbeit beschäftigt sich mit der Aufdeckung von Betrugsfällen in Mobilfunknetzen mit besonderem Augenmerk auf dem Einsatz von künstlichen neuronalen Netzen. Während der erste Teil Grundlagen wie die verschiedenen Betrugsarten und künstlichen neuronalen Netzen behandelt wird im zweiten Teil ein Prototyp einer Applikation zur Identifizierung von Missbrauchsfällen entwickelt. Dieses System baut dabei auf den folgenden beiden Ansätzen auf, einerseits auf einer selbst organisierenden Karte zur Analyse der Daten und Ermittlung von Referenzvektoren die Datencluster repräsentieren und andererseits auf einer Differentialanalyse bei der das Verhaltensmuster bezogen auf das aktuelle Anruferverhalten eines Benutzers ermittelt und mit einem historischen Muster verglichen wird.

Abstract

Because of fraud the telecommunication industry lose profits in the order of several billions of euros every year. With artificial neural networks, an approach that aims to model the human brain, many cases of fraud can be identified at an early stage. Applications in this area range from the learning of known patterns to the autonomous categorisation of transactions and the development of user profiles to conclude from comparing a current with a historical profile that there is an anomaly in behaviour.

This diploma thesis is about the identification of fraud in mobile networks with a special focus on the usage of artificial neural networks. While the first part of this thesis covers basics like the different cases of fraud and artificial neural networks the second part deals with the design and implementation of a prototype to identify fraud. The developed system uses a self organizing map to find clusters in the call data, resulting in a number of reference vectors, and the differential approach of user profiling which determines a behaviour pattern, based on the user’s current behaviour, to compare it with a historical pattern.

Kernpunkte für das Management

Dieser Beitrag beschäftigt sich mit Missbrauchsfällen in Mobilfunknetzen und deren Identifizierung durch künstliche neuronale Netze. Dabei wird zuerst im ersten Teil der Arbeit auf die notwendigen Grundlagen näher eingegangen um danach im zweiten Teil einen Prototyp zur Identifizierung dieser Fälle zu entwickeln.

Dabei ergaben sich die folgenden Erkenntnisse:

- Durch Betrugsfälle entgehen der Telekommunikationsbranche jährlich Milliarden Euro an Einnahmen.

- Es existieren sehr viele verschiedene Methoden des Missbrauchs und es werden mehr.

- Die meisten Missbrauchsfälle bzw. -methoden gehen mit einer Änderung des Verhaltensprofils des betroffenen Benutzers einher.

- Durch führen eines aktuellen und historischen Verhaltensprofils pro Benutzer und einem regelmäßigen Vergleich der beiden können Änderungen im Verhalten alarmiert werden.

- Künstliche neuronale Netze können in einem System zur Identifizierung von Betrugsfällen für eine Vielzahl an Aufgaben eingesetzt werden.

- Im Zusammenhang mit der Benutzerprofilmethode können selbst organisierende Karten eingesetzt werden um Referenzvektoren für Datencluster zu errechnen die als Basis für die Erstellung der Profile dienen.

- Die Qualität der gelieferten Resultate eines auf einem künstlichen neuronalen Netz beruhenden Systems welches Benutzerprofile berechnet ist von einer Vielzahl an Faktoren (Distanzfunktion, Skalierungsintervall, Trainingsdaten des neuronalen Netz …) abhängig.

- Vor dem produktiven Einsatz eines solchen Systems ist auf Grund der umfangreichen Anzahl an Einflussfaktoren ein hoher Aufwand an Kalibrierungsmaßnahmen notwendig.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

1 Einführung

Betrug zählt zu den bedeutendsten Umständen für entgangene Einnahmen in der Telekommunikationsbranche. Einem durchschnittlichen Netzbetreiber in West Europa entgehen gesamt ca. 8 Prozent seiner jährlichen Erträge, in der Asien-Pazifik Region und Ost-Europa 31 bzw. 18,5 Prozent, der Grossteil davon durch Betrug [Wiel04]. Ergebnisse aus Umfragen der CFCA (Communications Fraud Control Association) zeigen für 2003 einen weltweiten Schaden von ca. 35 - 40 Milliarden Dollar während es im Jahr 1999 noch 12 Milliarden waren [CFCA03].

Aktuelle Missbrauchsfälle werden bereits mit organisiertem Verbrechen in Verbindung gebracht. Mit der Unterstützung durch Hacker werden teilweise technisch komplizierte Verfahren eingesetzt um Netzbetreiber zu schädigen. Andererseits bedeutet dies nicht, dass Alltagsbenutzer nicht zu betrügerisches Verhalten neigen, wenn auch im kleineren Ausmaß. Die allgemeine Verfügbarkeit von Informationen über oder Anleitungen zum Missbrauch und die Mühelosigkeit mit der einige davon genutzt werden können erlaubt es auch Amateuren in Netzwerke, Kundenkonten oder Services einzudringen um sie unerlaubt zu nutzen oder zu manipulieren. Diese Faktoren führen immer öfters zu Verlusten (bzw. entgangenen Erträgen), Netzwerk Ausfallzeiten, Service Fehlfunktionen und im Weiteren auch zu einem schlechten Image des Providers. [IECoJ]

Mit zunehmender Verbreitung neuer Technologien wie z.B. „Voice over IP“ eröffnen sich auch vermehrt neue Möglichkeiten um Netzbetreiber zu schädigen. Diesem Trend gilt es entgegen zu wirken um Kunden und Umsätze zu schützen.

Systeme deren Zweck das Aufdecken von Missbräuchen ist operieren mit riesigen Datenmengen, z.B. rund 37 GByte pro Tag bei einem großen deutschen Mobilfunkanbieter was in etwa 30-40 Million Anrufen entspricht. Aus diesen Zahlen ist ersichtlich dass falls nur 0,1 % davon durch Missbrauch bzw. Betrug entstehen, es zu einem Einnahmeausfall für 3000 – 4000 Anrufen pro Tag kommt.

Wahrscheinlich wird es unmöglich sein alle Betrugsfälle total zu eliminieren bzw. ist dies wie Abbildung 1 zeigt auch nicht kosteneffektiv. Abbildung 1 stellt den Zusammenhang zwischen den Kosten die durch Betrug entstehen und denen der Bekämpfungsmaßnahmen dar. Mit steigendem Aufwand für die Bekämpfung steigen auch die Kosten, ab einem gewissen Punkt übersteigen diese den durch Betrug entstandenen Schaden [ShHB99]. D.h. weitere Maßnahmen sind ab diesem Punkt nicht mehr rentabel. Ein System zur Identifizierung von Missbrauchsfällen muss kosteneffektiv sein.

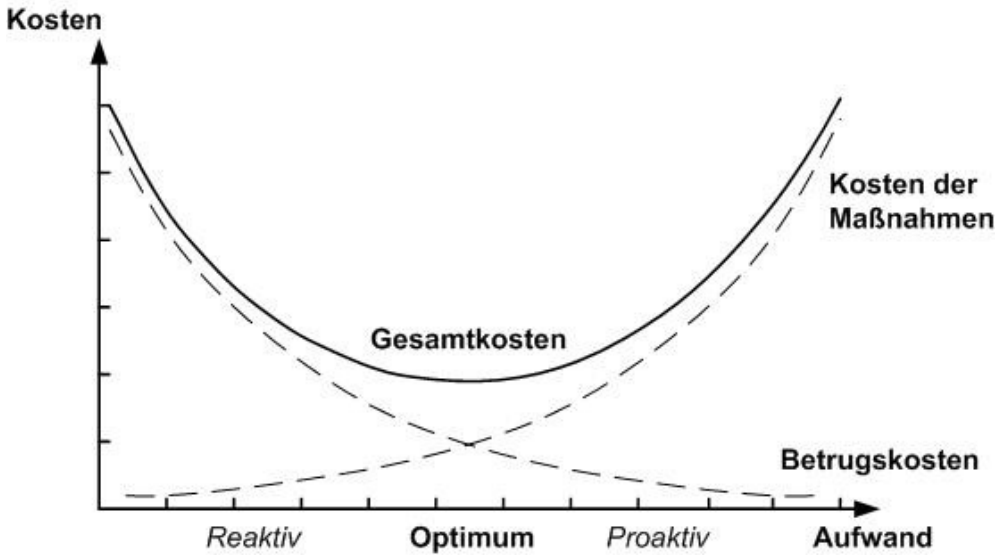

Abbildung 1: Kostenanalyse von Anti-Betrugs Maßnahmen [ShHB99]

Durch künstliche neuronale Netze, ein Ansatz der das menschliche Gehirn zum Vorbild hat, können viele der Missbrauchsfälle frühzeitig erkannt und damit auch darauf reagiert werden. Anwendungen in diesem Bereich reichen vom Erlernen bekannter Betrugsmuster bis hin zur selbständigen Kategorisierung von Transaktion und der Erstellung eines Benutzerprofils um durch Vergleich von aktuellem und historischem Profil auf Betrug bzw. Anomalien im Verhalten zu schließen.

Diese Arbeit wird sich vorwiegend mit der Anwendung von künstlichen neuronalen Netzen zur Identifizierung von Betrugsfällen befassen. Als Daten dienen Verrechnungsdaten (Call Detail Records), vom Netzwerkelement generierte Informationen über Gespräche, eines großen deutschen Mobilfunkbetreibers.

Einleitend werden in Kapitel 2 die häufigsten Methoden von Betrug im Telekommunikationsbereich angeführt und erläutert was man unter der jeweiligen Methode näher verstehen kann und ob man diese mit der hier behandelten Technologie der künstlichen neuronalen Netze aufspüren kann.

Kapitel 3 beschäftigt sich näher mit den Daten die von Netzwerkelementen für Anrufe generiert werden. Wie diese strukturiert sind, welche Informationen sie enthalten, wie sie gelesen werden können und auf welche Art diese bei großen Operatoren gesammelt werden.

Kapitel 4 widmet sich der Frage welche Methoden inklusive des hier behandelten Schwerpunkt Themas der künstlichen neuronalen Netze zur Verfügung stehen um Betrugsfälle zu identifizieren und zeigt Stärken und Schwächen der jeweiligen Ansätze auf.

Kapitel 5 behandelt den theoretischen Hauptteil der Arbeit, künstliche neuronale Netze, und geht dabei unter anderem auf Themen wie Lernalgorithmen und Netztopologien ein.

Kapitel 6 implementiert einen Prototyp einer Anwendung zur Identifizierung von Anomalien im Anruferverhalten basierend auf künstlichen neuronalen Netzen und der Benutzerprofil – Methode. Das System besteht aus einzelnen Komponenten die eine Art Workflow darstellen und damit das Betrugserkennungssystem durch Konfiguration erweiterbar macht (z.B. um

eine regelbasierte Vorstufe die bekannte Missbrauchsfälle erkennt). Dabei kommt die Programmiersprache JAVA zum Einsatz.

In Anhang 8 und Anhang 9 werden der Receiver Operating Characteristics – Ansatz, eine Methode zur Bewertung von Betrugserkennungssystemen, und genetische Algorithmen zur Lösung von Optimierungsproblemen kurz beschrieben

2 Betrugsarten

Betrüger passen auf der Suche nach einfacheren oder billigeren Wegen sich selbst zu bereichern ihre Methoden fortwährend an. Welche Variante schließlich zum Einsatz gelangt hängt zum großen Teil von der Komplexität der Netze, der Reife des Operators und der angebotenen Dienstleistungen ab. Beweggründe eines Betrügers variieren dabei stark und reichen vom Erhalten der freien Dienstleistungen bis zum illegalen Weiterverkauf, zu niedrigeren Preisen, des Services (Airtime, Ferngespräche, Inhalte, usw.).

Organisationen wie das „Forum for International Irregular Network Access" (FIINA) sowie führende Hersteller von so genannten „Fraud Management"-Systemen (FMS), sammeln kontinuierlich Informationen zu auftretenden Szenarien um Betreiber in der fortwährend Schlacht gegen Betrüger zu unterstützen – natürlich, wie im Falle der Hersteller von FMS, nicht immer frei von kommerziellen Interessen. Mit steigender Verbreitung von IP- und 3G-Diensten, kann auch eine steigende Anzahl an Betrugsmethoden und -varianten erwarten werden [Salp03].

Im Folgenden werden die geläufigsten Techniken und Angriffspunkte des Betrugs in der Telkommunikationsbranche vorgestellt, unter anderem auch die von Shawe-Taylor et al. [ShHB99] genannten Kategorien des Zeichnungsbetrugs, Manipulation von Nebenstellenanlagen- oder „Dial Through"-Betrugs, PRS Betrugs, Diebstahl des Endgeräts und Roaming-Betrugs. Da in Kapitel 7 ein Prototyp einer Applikation die auf der Profil-Methode basiert implementiert wird, gehen die folgenden Abschnitte jeweils zusätzlich auf die Möglichkeit einer Identifizierung durch diese Methode ein.

2.1 Zeichnungsbetrug („subscription fraud")

Zeichnungsbetrug ist die einfachste Art des Betrugs in der Telekommunikation. Betrüger registrieren sich mit falschen Daten für Dienstleistungen ohne der Absicht später für die entstandenen Kosten aufzukommen bzw. die Rechnungen zu begleichen.

Dieser Fall des Betrugs ähnelt stark dem einer uneinbringlichen Schuld, der wesentliche Unterschied besteht darin dass beim Zeichnungsbetrug von Anfang an nie die Absicht zur Bezahlung der offenen Beträge bestand.

In organisierten Fällen vermeiden Betrüger eine frühzeitige Erkennung indem sie über ausgedehnte Perioden Identitäten bilden, betreiben und ihre Rechnungen pünktlich begleichen um das Vertrauen des Providers zu gewinnen. Der Betrug tritt erst später nach Perioden hoher Netznutzung zu Tage. Üblicherweise wird dieser Ablauf danach mit neuen Registrierungsdaten wiederholt.

Diese Art des Betrugs kann mit erweiterten Benutzer Profilmethoden durch statistische Methoden und künstliche neuronale Netze erkannt werden. Profilkategorien die auf potentiell gefährliche Nutzer schließen lassen werden dabei angelegt und mit der Nutzung in den ersten Wochen verglichen bzw. kann eine signifikante Änderung des Verhaltens alarmiert werden. Problematisch ist jedoch der Fall das ein Betrüger ebenfalls ein „normales" Nutzungsprofil aufweißt und dadurch zum Zeitpunkt der Identifizierung (z.B. durch eine nicht bezahlte Rechnung) der Schaden bereits entstanden ist.

2.2 Roaming Betrug

Für Mobilfunkbetreiber spielt der Faktor „Roaming“ zusätzlich zum normalen Zeichnungsbetrug eine große Rolle. In diesem Fall registriert sich der Betrüger bei Operator A und nutzt danach das Gerät im Ausland bei Operator B z.B. für Mehrwertdienste. Dabei macht er sich die verzögerte Übertragung der Verrechnungsdaten von Operator B zu A, meist in Form von TAP (Transfer Account Procedure) - Dateien, zu Nutze und der Betrugsfall wird erst verspätet aufgedeckt.

Für diesen Fall gelten in Bezug auf die Erkennung durch künstliche neuronale Netze dieselben Fakten wobei wie bereits erwähnt vor allem die Zeitverzögerung der Synchronisierung der Verrechnungsdaten zwischen den Netzbetreibern ein Problem darstellt.

2.3 Partnerablehnung („partner repudiation“)

Ein Betrugsfall der üblicherweise in 3G Netzen auftritt. Ein Content-Partner erhält im Normalfall pro herunter geladene Datei eine Zahlung vom Netz-Operator. Es liegt also im Interesse dieses Partners die Gesamtanzahl der von seinen Servern herunter geladenen Dateien zu erhöhen. Auf Grund der Eigenschaften von Netzen ist es oftmals üblich die Zahlung bereits bei einer Übertragung von 95 % zu tätigen. Dies könnte den Partner dazu verleiten beispielsweise ein Skript zu installieren welches jede dritte Transaktion nach 96 % abbricht. Eine abgebrochene Aktion wird meist voll automatisch nochmals wiederholt was schließlich zu einer doppelten Bezahlung führt, einmal für die erste (abgebrochene) Transaktion und ein weiteres Mal für den erneuten Versuch.

Mittels eines Profils für das Verhalten der Content-Partner können atypisch hohe Abbruchsraten identifiziert werden bzw. mit Profilen von „normalen“ Partnern verglichen werden und eine signifikante Abweichung alarmiert werden. Als Daten zur Auswertung kommen spezielle „Call Detail Records“ zwischen Operator und Partner in Frage.

2.4 Prepaid Karten Betrug

Prepaid Karten wurden entwickelt um die Möglichkeiten zur Umgehung der Bezahlung von konsumierten Dienstleistungen (z.B. Sprache) einzuschränken und um dadurch in Folge Verluste durch Zahlungsausfälle und Zeichnungsbetrug zu verringern. Jedoch entwickelten sich auch bei diesem Ansatz schnell einige Varianten von Betrugstechniken. Betrüger nutzen beispielsweise Sicherheitslücken beim Wiederaufladen von Prepaid-Karten und sind dadurch in der Lage selbst gedruckte Ladebons oder illegal aufgeladene Karten zu verkaufen.

Diese Art des Betrugs kann, da sie sich nicht auf eine Änderung im Verhalten beziehen, nicht mittels eines Benutzerprofilansatzes erfasst werden.

2.5 Klonen von „Subscriber Identity Module“ (SIM) Karten

Klonen ist eine der häufigsten Arten des technischen Betrugs mit dem sich Mobilfunkbetreiber auseinandersetzen müssen. Identitätsdetails legitimer SIM-Karten werden abgefangen (durch Abhorchen des Funkverkehrs zu den Basisstationen, usw.) und dann in geklonte SIM-Karten programmiert. Bis der Betrug festgestellt wird, werden Gebühren die der Klon verursacht dem legitimen Teilnehmer in Rechnung gestellt. Abgesehen von verärgerten und möglicherweise in weiterer Folge auch zu anderen Operatoren wechselnden Kunden führt dies auch zu direkten Verlusten.

Diese Verluste sind umso höher je kostspieliger die Dienstleistungen sind die durch solche geklonten SIM-Karten genutzt werden, z.B. Roaming, M-Commerce oder Inhalte.

Im Allgemeinen geht mit einer geklonten SIM-Karte auch eine Änderung des Nutzungsprofils einher, damit ist auch eine Identifizierung mittels des Benutzerprofilansatzes möglich. Beispielsweise steigt die Anzahl der Anrufe plötzlich oder es werden so genannte Mehrwertdienste genutzt.

2.6 Calling Card Betrug

Calling Card Services bei denen eine Kombination von Konto-Nummer und PIN zur Identifizierung des Benutzers verwendet werden sind ein sehr attraktives Ziel für Betrüger. Durch das erlangen der Zugangsdaten des legitimen Benutzers und deren unerlaubten Nutzung wird der Betrug begangen. Die Motivation dahinter ist das Service entweder für sich selbst oder andere zu nutzen.

Wege um die notwendigen Informationen zu erlangen:

- „Eavesdropping“ (oder auch „shoulder–surfing“): Legitime Benutzer werden bei der Eingabe ihrer Daten entweder belauscht oder beobachtet um diese später für sich selbst zu nutzen bzw. um sich für den Benutzer auszugeben, ihn zu imitieren.

- „Dumpster diving“ (oder auch „skip-hopping“): Müll wird auf weggeworfene Gesprächsnachweise, Post des Anbieters, Rechnungen oder ähnliche Informationen durchsucht.

- Einfacher Diebstahl.

- Unerlaubtes Eindringen in ein System um die notwendigen Codes zu erhalten.

Calling Card Betrug wird oftmals in Verbindung mit Call Selling und PRS Betrug ausgeübt, um ein signifikantes Einkommen zu generieren. Diese Art des Missbrauchs kann vor allem wenn simultane Anrufe über die Karte erlaubt sind in kurzer Zeit zu sehr hohen Rechnungen bzw. Verlusten führen.

In diesem Fall kann meist ebenfalls eine Änderung im Benutzerverhalten festgestellt werden und dadurch auch durch den beschriebenen Ansatz zumindest ein Alarm der auf diese Änderung hinweist generiert werden.

2.7 Interner Betrug

Der Personenkreis der Betrüger beschränkt sich nicht nur auf Personen außerhalb der Organisation des Betreibers. Oftmals werden auch Angestellte des Betreibers, Händler, Hersteller oder Wiederverkäufer die ihr Insiderwissen über Prozesse bzw. Prozeduren nutzen um sich selbst zu bereichern oder um für jemanden anderen aktiv zu werden.

Unter diese Kategorie fallen z.B. Aktivitäten wie das Einrichten von Testverbindungen die keine Rechnungsdaten generieren oder die Installation von automatischen Wählern zu PRS (Premium Rate Services) Nummern.

Diese Interna werden auch oft an Kriminelle, die nicht direkt mit dem Netzbetreiber in Verbindung gebracht werden können und dadurch in der Lage sind die erlangten Informationen sicher zu nutzen, verkauft.

Diese Art des Missbrauchs kann nicht mittels eines Systems welches das Anruferverhalten analysiert und mit historischen Daten vergleicht nicht entdeckt werden.

2.8 PRS (Premium rate service)

Bei dieser Art des Betrugs kann es dazu kommen das die Person die das Service bezieht mit dem PRS Anbieter zusammenarbeitet um dessen Umsatz zu erhöhen. Der Service Anbieter hofft zumindest einen Teil dieses Umsatzes vom Netzbetreiber einfordern zu können. Dadurch bezahlt der Netzbetreiber für Services ohne dafür selbst Einnahmen aus Anrufgebühren verrechnen zu können. Selbst wenn der Betreiber nicht selber das PRS zugänglich macht, d.h. es ist nicht in seinem Netz, kann er in Mitleidenschaft gezogen werden da Anrufe zu diesem Dienst aus seinem Netz stammen können oder über sein Netz zu einem anderen Betreiber weitergeleitet werden (call transition, interconnect) und damit jedenfalls Ressourcen beanspruchen.

Alternativ kann der Betrüger PRS auch einfach nur zu seiner privaten Unterhaltung nutzen. In diesem Fall kann meist ebenfalls keine Gebühren verrechnet werden da entweder die Registrierung mit falschen Daten zustande kam (siehe auch Zeichnungsbetrug) oder der legitime Inhaber des Kontos dafür nicht haftbar gemacht werden kann.

PRS Nummern sind meist internationale Nummern oder bei anderen Netzbetreibern. Internationale PRS Nummern erschweren die Identifizierung zusätzlich da sie meist einem so genannten „long calling“-Muster gleichen.

In vielen dieser Fälle steht diese Art des Missbrauchs in engem Zusammenhang mit Zeichnungsbetrug. Aus diesem Grund gelten hier bezüglich der Entdeckung des Missbrauchs dieselben Aussagen wie beim Zeichnungsbetrug.

2.9 Teeing In / Clip-On Betrug

Diese Art umfasst die physische Verbindung zu einem existierenden Service um unerlaubt Gespräche zu führen oder diese Verbindung zu einer anderen Partei umzuleiten (z.B. einer PRS Nummer). Die Verbindung kann an jedem Punkt an dem das Kabel zugänglich ist hergestellt werden.

In diesem Fall kann ein Profilansatz nur indirekt eingesetzt werden, z.B. könnte ein Profil für die generelle Nutzung von PRS erstellt werden um eine signifikanten Abweichung zu alarmieren.

2.10 Dial Through & Umleitungs Betrug

Viele Nebenstellenanlagen beherrschen durch Nutzung des DISA (Direct Inward System Access) Dienstes das so genannte „Dial Through“ Feature. Dieses erlaubt es automatisiert Verbindungen aufzubauen welche dann dem Inhaber der Nebenstellenanlage verrechnet werden.

Ein Anrufer wählt eine Zugangsnummer, üblicherweise gebührenfrei, und wird dazu aufgefordert einen PIN anzugeben. Ist der PIN Code gültig ertönt das Wahlaufforderungssignal und der Anrufer ist damit in der Lage auf fremde Kosten beispielsweise zu PRS oder internationalen Nummern zu telefonieren.

Erlangt der Angreifer die für diesen Dienst erforderlichen Zugangsdaten (Nummer und PIN) kann auch er dieses Feature benutzen.

Anomalien können bis zu einem gewissen Grad durch den Einsatz eines Profils welches das Verhalten der Nebenstellenanlage erfasst ermittelt werden. Jedoch können beispielsweise einzelne unlautere Ferngespräche in der Menge „untergehen“. Mit der Profil Methode können nur signifikante Abweichungen festgestellt werden.

2.11 VoiceMail

Diese Art ist nicht so weit verbreitet wie die bisher genannten Varianten. Betrüger versuchen Zugang zu ungeschützten Nebenstellenanlagen VoiceMail Boxes zu erlangen um entweder einfach nur Nachrichten zu hinterlassen (beispielsweise für Komplizen in kriminelle Aktivitäten) oder um Zugang zur DISA Funktion (siehe Punkt 2.10) zu erlangen. Der eigentliche Benutzer kann durch eine Änderung des PINs aus seiner Mailbox ausgesperrt werden.

Diese Art des Missbrauchs kann durch die oftmals nur geringe Änderung im Verhalten nicht mittels eines profilorientierten Ansatzes erfasst werden.

2.12 Tarif Manipulation

Diese Art wird oft nicht als Betrug klassifiziert da es durch legitime aber vom Operator nicht beabsichtigte Nutzung von Eigenschaften eines Services Zustande kommt.

Als Beispiel für Tarif Manipulation kann folgendes angeführt werden:

- Ein Anrufer benutzt seine Calling-Card mit flat-rate (z.B. 12 Cent/Minute) um ein PRS zu nutzen (z.B. 40 Cent/Minute).

Diese Art der unbeabsichtigten Nutzung eines Services durch den Kunden kann mittels eines Benutzerprofilansatzes nur dann identifiziert werden wenn im Laufe der Analyse einer signifikanten Änderung des Benutzerverhaltens oder der Servicenutzung die „Lücke“ im Tarif entdeckt wird.

2.13 Pin Hacking

Jeder Dienst der mittels einer Kontonummer und einem PIN den Zugang erlaubt ist potentiell angreifbar. In Abhängigkeit von der PIN Politik (Länge, Anzahl der erlaubten Fehleingaben, keine Default PIN usw.) kann diese Bedrohung als hoch bis niedrig eingestuft werden.

Kommt es zu einer wesentlichen Änderung des Verhaltens kann durch den Einsatz der Profil-Methode ein Alarm generiert werden.

2.14 E/M-Commerce Betrug

„Mobile Commerce“ und „Electronic Commerce“ beschreiben den Prozess um online (electronic) oder mobil Transaktionen zu bearbeiten, z.B. Produkte oder Services online zu erwerben.

Dieser Bereich ist wegen seines verhältnismäßig hohen finanziellen Wertes für Betrüger besonders attraktiv. Die Methodologie zum Begehen von E/M-Commerce Betrug variiert stark und hängt vor allem auch vom Typus des Dienstes ab. Grundsätzlich kann diese Art aber auf wenige Kategorien herunter gebrochen werden:

- Missbräuchliche Nutzung einer Anmeldung: Ein sekundärer Angriff folgend auf Zeichnungsbetrug bei dem ein Konto genutzt wird um Zugang zu E/M-Services zu erlangen ohne der Absicht später für die entstandenen Rechnungen zu bezahlen oder den Weiterverkauf der auf diese Art illegal an sich gebrachten Waren.

- Technisch: Unterwandern der technischen Gegebenheiten bzw. Sicherheitsmechanismen um ohne Zahlung oder Registrierung Waren oder Inhalte zu erlangen.

- Zahlung: Missbräuchliche Nutzung der Zahlungsmethoden um nicht für Services zu bezahlen. Zum Beispiel die Nutzung einer falschen oder gestohlenen Kreditkarte oder das Bestreiten der erfolgten Lieferung

Je nach Kategorie gelten verschiedene Bedingungen um in der Lage zu sein den Missbrauch aufzuspüren. Siehe dazu die entsprechenden Vorkapitel.

2.15 Ungeschützte Bereiche in der Organisation

Betrüger machen sich Schwachstellen innerhalb der Organisation zu Nutze. Zum Beispiel sind neue Services die unter Zeitdruck ausgebracht und aktiviert wurden ohne sie vorher komplett zu testen oder die nicht vollständig von der Organisation unterstützt werden potentielle Angriffspunkte.

Zu den gefährdeten Bereichen gehören unter anderem [Salp03]:

- Customer relationship management (CRM), Vertrieb, Kundendienst, und Kreditabteilung: Betrüger nützen Schwachstellen in existierenden Prozessen um verschiedenste Varianten von Zeichnungsbetrug zu verüben.

- Netz und Technik: Betrüger nützen Schwachstellen in der bestehenden Netzwerk Hardware und Prozessen um verschiedene Arten des technischen Betrugs zu verüben. Technischer Betrug beinhaltet beispielsweise das Klonen von Mobiltelefonen, Calling Cards oder SIM Karten, das unerlaubte Eindringen in Nebenstellenanlagen usw. Während der technische Schutz der Netze sich fortgehend verbessert wird auch diese Art von Betrug immer schwieriger und kostspieliger und nimmt dem zu Folge ab. Betrüger suchen nach einfacheren und kosteneffektiveren Methoden.

- Business- und Operation- Support Systeme (BOSS), Personalwesen, IT und Buchhaltung: Betrüger nützen Schwachstellen im Sicherheitssystem von BOSS und IT Systemen (gewöhnlich durch die Nutzung des Know-hows von Insidern oder Ex-

Angestellten) um ihre Gebühren zu reduzieren und Rückzahlungen oder Zugangscodes zu erhalten. Da Konsolidierungsprozesse und Personalabbau aktuell im Wirtschaftsleben voranschreiten wird dieser Typus des Betrugs aufgrund des vermehrten „Angebots“ an Ex-Mitarbeitern voraussichtlich zunehmen.

Je nach Bereich bzw. Kategorie gelten verschiedene Bedingungen um in der Lage zu sein den Missbrauch aufzuspüren. Siehe dazu die entsprechenden Vorkapitel.

2.16 Social Engineering

Dies ist in diesem Zusammenhang die einfachste Art um zu betrügen. Das Opfer wird kontaktiert, gewöhnlich telefonisch, und überzeugt seine individuellen und geheimen Kontoinformationen preis zu geben.

Betrüger geben meist vor für die Telefongesellschaft zu arbeiten, mit einer Analyse eines Fehlers, aufgetreten am Anschluss des Opfers, beschäftigt zu sein und man die Zugangsdaten des Kunden bestätigt braucht. In manchen unverschämten Fällen kommt es sogar dazu das der Kriminelle vorgibt auf der Spur eines Betruges in Verbindung mit diesem Anschluss zu sein und falls der legitime Kunde ungewöhnliche Posten auf seiner Rechnung wahr nimmt sich keine Sorgen zu machen braucht da man diesem Fall bereits auf der Spur ist.

In diesem Fall kann eine signifikante Änderung im Nutzungsverhalten durch den Profilansatz ebenfalls aufgezeigt bzw. alarmiert werden.

Wie beschrieben kommt es in fast allen der oben angeführten Arten des Betrugs gleichzeitig mit dem Betrugsfall auch (durch den Missbrauch) zu einer signifikanten Änderung des Benutzerverhaltens. Durch die Beobachtung des Verhaltens, der Erstellung ein aktuelles Profil und dem Vergleich mit einem Langzeit-Profil, kann gegebenenfalls schnell auf Anomalien in der Nutzung geschlossen werden.

Weiters kann ein künstliches neuronales Netz auch zum Erlernen bereits bekannter Fälle eingesetzt werden. Bei dieser Kategorie ist es allerdings notwendig a priori die Methode identifiziert zu haben und Daten aus Echtfällen zum Training des Netzes zu sammeln. Damit können zwar in Zukunft bekannte Fälle schnell erkannt werden (wobei auch eine gewisse Unschärfe durch die Eigenschaft der Generalisierung der Netze genutzt wird) jedoch dauert es bis neue, innovative Methoden entdeckt werden was bereits zu entgangenen Einnahmen führt.

Während in Kapitel 4 auch auf diverse andere Methoden zur Erkennung der oben angesprochenen Fälle näher eingegangen wird beschäftigt sich diese Arbeit vor allem mit der Identifizierung von potentiell betroffenen Benutzern durch künstliche neuronale Netze die sich unter anderem zur Erstellung bzw. Pflege von Benutzerprofilen sehr gut eignen.

Eine Ausnahme dazu stellt der Zeichnungsbetrug dar bei dem noch kein Verhalten zum Vergleich vorliegt bzw. kein Langzeit-Profil erstellt werden konnte. In diesem Fall wird von einigen Herstellern auf einen Vergleich mit für bestimmte Benutzergruppen typischen Profilen oder die Vorhersage der Wahrscheinlichkeit eines Betrugsfalls bzw. Zahlungsausfalls an Hand eines Entscheidungsbaums zurückgegriffen.

3 Verrechnungs-Dateien und Datensätze

3.1 Allgemein

Netzwerkelemente speichern permanent Detail Datensätze („Call Detail Records“) aller Anrufe die sie abwickeln um sie für eine spätere Weiterverarbeitung zur Verfügung zu stellen. Inhalt dieser Datensätze sind z.B. Informationen über Gesprächsdauer, Ort des Gesprächs, Rufnummer des rufenden und gerufenen Teilnehmers usw.

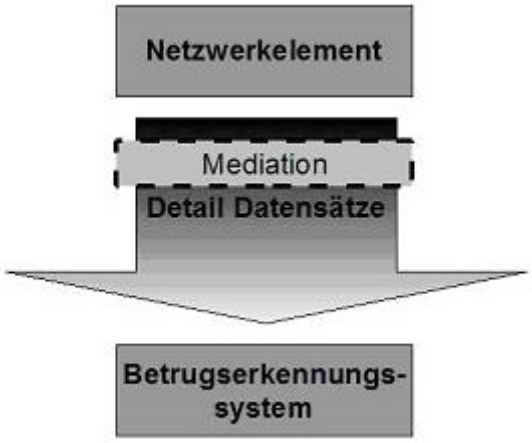

Abbildung 2: Detaildatensatztransfer

Wie in Abbildung 2 schematisch dargestellt transferieren Netzwerkelemente die von ihnen produzierten Detaildatensätze im so genannten push-Modus zu einem Mediation System. Dieses dekodiert das proprietäre Format der Daten, konvertiert sie und stellt sie dann beispielsweise in Form von binären, standardisierten Dateien zur Verfügung oder legt sie in einer Datenbank ab. Eine weitere Funktion von Mediation Systemen ist die „near realtime“ Weiterleitung von konvertierten Daten an registrierte Systeme was z.B. für den Prepaid-Dienst von besonderer Bedeutung ist.

Als Konsumenten dieser Daten kommen eine Vielzahl von Systemen in Frage darunter beispielsweise welche zur Betrugserkennung, Verrechung oder auch zur Auswertung der Netzqualität.

Struktur und Inhalte der von den Netzwerkelementen gelieferten Datensätze variieren von Hersteller zu Hersteller. Diese proprietären Formate werden dabei jeweils mehr oder weniger detailliert in der vom Hersteller mitgelieferten Dokumentation beschrieben, beispielsweise im Falle von Nortel im Dokument „*Bellcore Format Automatic Message Accounting – Reference Guide*“ [Nort00], und sind mit einer Vertrauensklausel belegt. Auf Grund dieser Tatsachen werden im Folgenden nur die üblichen allgemeinen Zusammenhänge beschrieben.

3.2 Datei – Struktur

Die Verrechungsdatei ist ein strukturiertes Objekt welches Detaildatensätze enthält. Zusätzlich zu diesen Daten enthält sie auch Informationen über die Struktur der Datei selbst und die Menge der beinhalteten gültigen Datensätze.

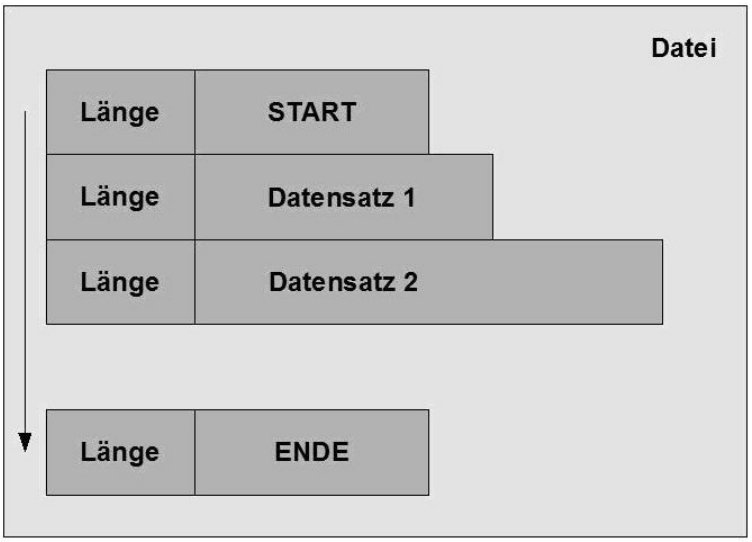

Abbildung 3: Verrechungsdatei

Im Wesentlichen besteht jede Verrechungsdatei, wie in Abbildung 3 vereinfacht dargestellt, aus einem Start-Datensatz, einer bestimmten oder unbestimmten Menge an Detaildatensätzen und einem Ende-Datensatz. In den meisten Fällen finden sich vor den eigentlichen Datensätzen noch ein oder zwei Byte die eine Aussage über die Länge des folgenden Datensatzes treffen.

In der Welt der Telkommunikation sind neben den herstellerabhängigen Formaten einige wenige standardisierte Dateiformate vorherrschend. Hier sollen kurz die 3 gängigsten Varianten genannten werden:

- BAF - Bellcore Automatic Message Accounting Format: Ein binäres Format mit Datensätzen von fix vordefinierter Anzahl an Datenfeldern und Feldtypen. Auf diesem Format basieren spezifische Formate der meisten großen Hersteller, wie z.B. Nortel.

- IPDR – IP Detail Records: Detail Daten über IP Services (z.B. VoIP) werden in XML oder XDR (External Data Representation [Srin95]) – Dateien geschrieben. Wobei die IPDR Organisation nicht nur Datei- und Datensatzformate sonder auch Transportprotokolle und -prozeduren vorgibt.

- TAP3 - Transfer Account Procedure: Ein binäres, tief strukturiertes, ASN1-BER oder -DER kodiertes Format welches oft zum Austausch von Roaming Daten zwischen Mobilnetzbetreibern verwendet wird.

3.3 Datensatz – Struktur

Abbildung 4 stellt die Struktur eines Verrechungsdatensatz schematisch dar. Jeder Datensatz besteht aus einem strukturierten Teil und zusätzlich aus Null oder mehreren Erweiterungen. Falls Erweiterungen verwendet werden wird abschließend ein Satz der das Ende der Erweiterungsliste signalisiert angehängt. Die Nutzung und Anzahl von Erweiterungen wird oft auch schon in den Kopfdaten angezeigt.

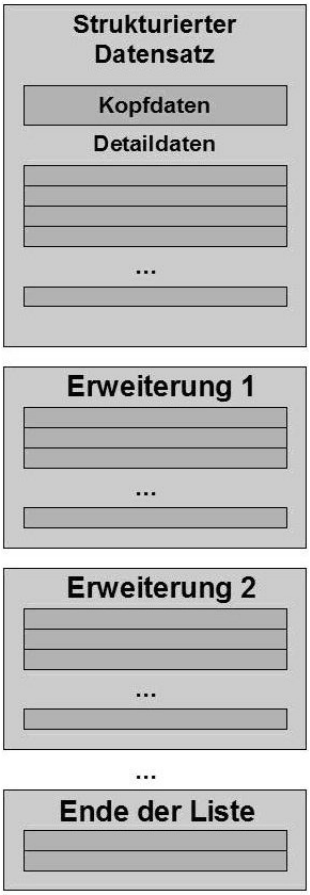

Abbildung 4: Verrechungsdatensatz

Der strukturierte Detaildatensatz besteht aus einer fixen Folge von Feldern. Dabei enthalten die Kopfdaten unter anderem ein Feld welches die Datenstruktur des Detaildatensatzes eindeutig identifiziert.

Der Inhalt der Felder wird in den meisten Fällen bzw. Formaten auf eine der folgenden Arten kodiert: BCD (binary encoded decimal), Hexadezimal oder (eher selten) ASCII.

Erweiterungen sind analog dem strukturierten Teil aus Feldern in fixer Reihenfolge und einem führenden Feld welches die Erweiterung eindeutig identifiziert aufgebaut. Sie werden genutzt um den Detaildatensatz um spezifische Informationen wie z.B. Standortinformationen zu erweitern. Während die Typen von zulässigen Erweiterungen für jeden Detaildatensatz genau vorgeben sind. Ist die Reihenfolge in der sie angehängt werden meist frei. Manche Erweiterungen können auch ein oder mehrmals in einem Verrechnungsdatensatz vorkommen.

Gesprächsdaten können zum Zwecke der Verarbeitungssicherheit auch in Teildatensätze, so genannte Interimtickets, aufgeteilt werden. Im Falle von Verbindung die lange andauern (z.B. 2 Stunden) entspräche der Verlust eines einzelnen Datensatzes einem nicht akzeptierbaren Einnahmeverlust auf Seite des Netzbetreibers. Falls aber für die Verbindung zyklisch, z.B. alle 10 Minuten, Informationen generiert werden dann entspricht der maximale Verlust nur dem einer 10 Minuten andauernden Verbindung.

3.4 Enthaltene Informationen

In den oben angesprochenen Dateien und Datensätze finden sich unter anderem Informationen zu folgenden Ereignissen wieder:

- Ausgehender/terminierender Anruf
- Roaming
- Transit: Falls eine Verbindung über das Netz eines Operators weitergeleitet wird, ohne seinen Ursprung oder Bestimmung in diesem Netz zu haben.
- Ausgehende/terminierende SMS
- Ortsabhängige Services: Personalisierte und ortsabhängige Services, z.B. ein Willkommensgruß am Flughafen mit Wetterinformationen.
- Ergänzende Services: Dazu zählen z.B. auch Rufumleitungen oder Konferenzschaltungen.

Je nach Ereignis enthalten die Datensätze eine vordefinierte Anzahl an Datenfeldern mit Informationen wie z.B.

- Rufende Nummer
- Gerufene Nummer
- Gesprächsdauer
- Referenznummer pro Verbindung: Diese dient vor allem dazu um verschiedene Eventtickets die während eines Gespräches generiert wurden später in Verbindung zu bringen (z.B. Interimtickets die während eines langen Telefonats entstehen)
- Datum des Verbindungsaufbaus
- Datum der Anrufbeantwortung
- Datum zu dem die Verbindung beendet wurde
- Grund des Verbindungsabbaus: Information ob ein Gespräch erfolgreich oder aus anderen Gründen (z.B. Netzprobleme) beendet wurde.
- Verwendetes Endgerät
- ID des bearbeitenden Netzwerkelements

3.5 Mediation

Laut Definition ist Mediation der Prozess der Erleichterung der Kommunikation und des Informationsaustausches zwischen zwei inkompatiblen Beteiligten. In der Welt der

Telekommunikation, entsprechen diese beiden inkompatiblen Beteiligten dem Kommunikationsnetz und der Infrastruktur der “Operations-Support” Systeme (OSS) (wie Gebührenverrechung, Betrugserkennung und Netzwerkmanagement-Systeme).

Dabei kann Inkompatibilität aus folgenden Gründen bestehen:

- Die beiden Seiten “verstehen” einander nicht (d.h., es werden unterschiedliche Protokolle und Datenformate verwendet)

- Die beiden Seiten möchten einander nicht verstehen (d.h., der Verkäufer des Netzelements oder der des OSS will die spezifischen Besonderheiten der jeweils anderen Schnittstelle nicht implementieren).

- Es sind mehrere Seiten beteiligt die jeweils andere Sprachen sprechen (d.h. verschiedene Netzwerkelement Hersteller, mehrere teils herstellerabhängige Protokolle und/oder einige OSS). Als ein Paradebeispiel dafür kann das Festnetz der Telekom Austria genannt werden welches zum einen Teil von der Firma Siemens und zum anderen Teil von der Firma Kapsch (Nortel Partner) ausgerüstet wird.

Diese Inkompatibilitäten verhindern, dass Netzbetreiber effektiv ihre Gespräche verrechnen, Services verwalten und Netzwerkmanagement ausführen. Mediation schlägt damit die Brücken zwischen diesen inkompatiblen Beteiligten.

Da Mediation-Systeme, entsprechend ihrer Natur, bereits nahe am Netz liegen, stellen sie eine ideale Plattform für die Bereitstellung der Daten zur Identifizierung von Betrugsfällen dar. Als „Sammler“ und „Verteiler“ der Netznutzungsdaten und Brücke zwischen dem Netz und dem OSS, ist es häufig auch das einzige System das Zugriff auf alle vom Netz generierten Nutzungsdaten hat. Da aber die meisten OSS nur an einer Teilmenge dieser Nutzungsdaten bzw. an Teilen der Information interessiert sind wird meistens vor der Übermittlung eine Filterung der Daten vorgenommen. Beispielsweise sind Verrechnungssysteme kaum an unbeantworteten Anrufen oder einem nicht erfolgreichen Verbindungsaufbau interessiert. Wohingegen Systeme die das Netz oder den Netzverkehr analysieren um daraus Schlüsse auf die Qualität des Netzes zu ziehen sehr wohl an diesen Daten interessiert sind.

In Verbindung mit Betrugserkennungssystemen stellt sich die Frage welche dieser Daten benötigt werden. Einerseits sollten, will man möglichst viele Fälle identifizieren (auch neue Methoden), so viele Dimensionen wie möglich in die Analyse mit einbezogen werden. Andererseits benötigt das Mehr an Daten auch signifikant mehr an Rechenleistung um zumindest „near realtime“ die Anomalien in der Nutzung zu erkennen. Welche Daten zur Analyse herangezogen werden wird in späteren Kapiteln näher behandelt.

Aus der Darlegung der bisherigen Fakten kann man erkennen dass ein Mediation System einer der essentiellen Komponenten eines Betrugserkennungssystems ist. Es muss alle benötigten Daten kontinuierlich sammeln und in einer geeigneten Form, welche niedrige Zugriffszeiten zulässt, zur Verfügung stellen.

Abschließend soll durch Abbildung 5 noch eine mögliche Ausprägung eines solchen Systems grob skizziert werden.

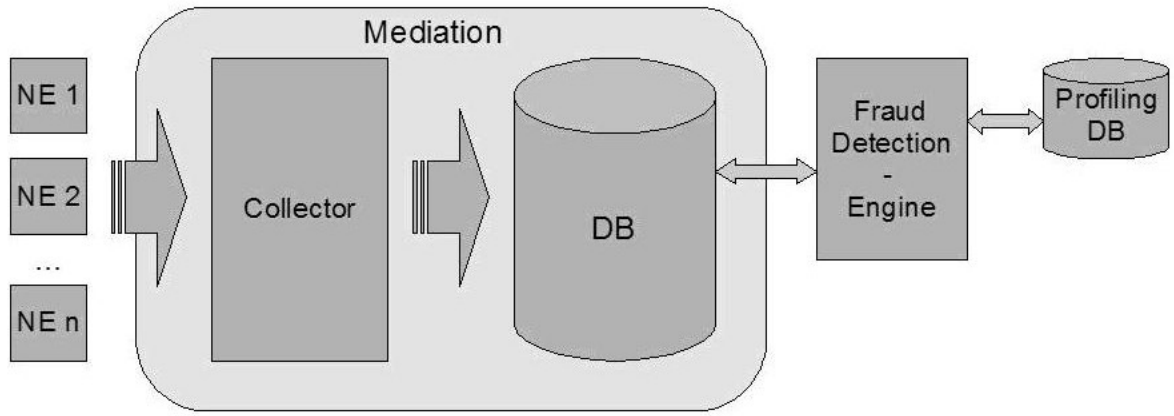

Abbildung 5: Mediation System

Netzwerkelemente (NE 1 … NE n) transferieren die von ihnen per Ereignis generierten Datensätze zu einem Collector. Diese Komponente sammelt die Daten von n Elementen, konvertiert sie in ein system- bzw. herstellerunabhängiges Format und schreibt das Ergebnis abschließend in eine Datenbank. Auf diese Datensammlung können nun alle Systeme die an Netznutzungsdaten interessiert sind, in Abbildung 5 ein Betrugserkennungssystem, zugreifen um die für sie relevanten Daten abzufragen.

4 Methoden zur Betrugserkennung

Im Folgenden werden die verschiedenen Methoden zur Identifizierung unterschiedlichster Betrugsszenarien beschrieben.

4.1 Benutzerprofil

4.1.1 Absolut

Dieser Ansatz der Betrugserkennung versucht durch die Identifikation von exzessiven Anruferverhalten auf Betrug zu schließen, z.B. wird ein Alarm generiert falls der Benutzer mehr als 2 Stunden pro Tag nach Süd Amerika telefoniert. Diese Methode wird absolute Analyse genannt da der Alarm auf Grund von Aktivitäten unabhängig vom jeweiligen Benutzer ausgelöst wird [MoVa97]. Das Hauptproblem dieses Ansatzes ist es, dass Benutzer mit intensiven Nutzungsverhalten, d.h. welche für die auch zweistündige Anrufe nach Süd Amerika ein normales Verhalten darstellen, ebenfalls Alarme auslösen.

Diese Methode eignet sich gut um Extrema von Aktivitäten zu erfassen die möglicherweise auf einen Betrugsfall hinweisen [BSCM97].

4.1.2 Differential

Um das Problem der absoluten Analyse zu umgehen wird bei der Differential-Analyse das Verhaltensmuster bezogen auf das aktuelle Anruferverhalten eines Benutzers ermittelt und mit einem historischen Muster verglichen.

Wenn die Vermittlungsstelle einen neuen Verrechungssatz erzeugt wird das Profil des rufenden Benutzers aktualisiert und an das Betrugserkennungssystem weiter geleitet. Dieses führt dann den Vergleich mit dem historischen Profil durch [MoVa97].

Die Abweichung der Profile kann durch verschiedene Verfahren quantifiziert werden, beispielsweise durch die Hellinger-Distanz, einem natürlichen Maß zum Vergleich zweier Wahrscheinlichkeitsverteilungen welches einen Wert zwischen 0 für vollkommene Übereinstimmung und 2 für Orthogonalität liefert [BuSh01]. Ist die Abweichung größer als ein erlaubter vordefinierter Wert wird ein Alarm ausgelöst. Gewöhnlich geht bei fast allen Methoden der missbräuchlichen Nutzung von Kommunikationsdiensten eine signifikante Abweichung des aktuellen vom historischen Profil einher.

Gegenüber dem absoluten Ansatz können unter andere folgende Vorteile angeführt werden:

- Ein Muster das für eine Kategorie von Benutzern als potentieller Indikator für Betrug angesehen werden kann, ist für eine andere akzeptabel. Im Falle der Differential-Analyse wird für jeden Benutzer oder jede Benutzergruppe ein eigenes Profil erzeugt welches das gewöhnliche Verhalten widerspiegelt. Dies ermöglicht eine Betrugserkennung auf persönlicher Ebene [BSCM97].

- Ein dem absoluten Ansatz folgendes Erkennungssystem würde eine Vielzahl von Fehlalarmen auslösen.

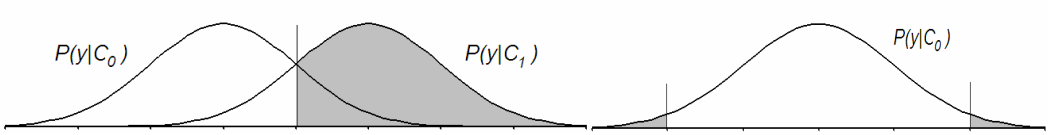

Abbildung 6: Absolut- und Differential-Analyse aus Wahrscheinlichkeitssicht [Holl00]

Aus Sicht der Wahrscheinlichkeiten stellt sich die Absolut- und die Differential-Analyse folgendermaßen dar, dass im absoluten Ansatz (linker Teil von Abbildung 6) sowohl für den Normalfall (C0) als auch für den Missbrauchsfall (C1) ein Modell formuliert werden muss während beim differentialen (rechter Teil von Abbildung 6) die Formulierung des Normalfalls und eines Schwellwerts (die gestrichelte Linien) ausreichend ist. Die grauen Bereiche in der Abbildung kennzeichnen den Missbrauchsbereich [Holl00].

4.2 Regeln

In Systemen die auf Regeln basieren werden bestimmte Parameter aus dem Profil, absolut oder differential, mit von Experten definierten Regeln bearbeitet. Regeln kann man sich vereinfacht folgendermaßen vorstellen: *IF {condition}, THEN {action}.*

Eine andere Methode, aus der Statistik, um eine Menge solcher Regeln aufzustellen ist die Nutzung von Algorithmen die nach Abarbeitung einer bestimmten Anzahl von Datensätzen dazu in der Lage sind automatisch Regeln zur Klassifizierung der Daten zu generieren. Beispiele für solche Algorithmen sind BAYES, FOIL, RIPPER, CART und C4.5 [BoHa02]

Im Gegensatz zu künstlichen neuronalen Netzen, die im folgenden näher beschrieben werden, ist ein auf Regeln basierendes System ein White-Box Ansatz, dadurch kann dem Betrugsanalytiker schnell und einfach der Grund für den Alarm geliefert werden [MLVV99].

4.3 Neuronale Netze

Die Vielzahl und die Heterogenität verschiedenster Betrugsszenarien erfordern den Einsatz intelligenter Methoden zur Erkennung. Ein System zur Identifizierung dieser Fälle des Missbrauchs muss flexibel genug sein um mit der Vielfalt von Betrug schritt halten zu können und sollte weiters auch über eine genügende Anpassungsfähig verfügen um die von Betrügern neu erdachten Methoden zu handhaben.

Diese Anforderungen führen direkt zum Einsatz von künstlichen neuronalen Netzen als mögliche Lösung. Neuronale Netze sind Systeme bestehend aus elementaren Entscheidungseinheiten die sich durch ihre Lernfähigkeit Bedingungen anpassen und damit die komplexesten Aufgaben lösen können. Ein Nachteil dieser Methode ist allerdings dass die Bewertung als Betrugsfall von außen nicht nachvollziehbar ist, da ein neuronales Netz einer Black-Box entspricht.

Es gibt mehrere Arten wie ein künstliches neuronales Netz lernen kann. Zu den Hauptgruppen des Lernens zählen z.B. überwachtes, unüberwachtes bzw. selbständiges und bestärkendes. Je nach Stärke der jeweiligen Gruppe ergeben sich daraus auch verschiedene Ansätze um Betrugsfälle zu identifizieren.

Als Beispiel für den Einsatz von unüberwachtem Lernen kann die selbständige Suche nach Strukturen (Clusteranalyse) in der Datenmenge angeführt werden. Ergebnis dieser Analyse ist eine optimale diskrete Abbildung einer kontinuierlichen Variablen bzw. eine bestimmte

Anzahl an Referenzvektoren die Clustern entsprechen. Im Zusammenhang mit Betrugserkennungssystemen kann diese Eigenschaft zur Klassifikation von Gesprächsdaten oder zur Berechnung von Benutzerprofilen genutzt werden.

4.4 Link Analyse

Die Link-Analyse bringt durch die Verknüpfung bzw. Verflechtung von Datensätzen und Sozialnetz Methoden bekannte Betrüger mit anderen Individuen in Verbindung. Beispielsweise fanden Sicherheitsbeauftragte heraus, dass in der Telekommunikation Betrüger selten isoliert von einander arbeiten oder dass falls eine SIM-Karte wegen Betrugs gesperrt wurde der Betrüger von einer anderen SIM-Karte meistens dieselben Nummern anwählt. Dadurch können Anrufe mit betrügerischen Anschlüssen in Verbindung gebracht werden. Ein ähnlicher Ansatz wird auch zum Aufspüren von Geldwäsche eingesetzt [BoHa02].

4.5 Visualisierung

Die Methode der Visualisierung [CEWB97] wurde entwickelt um sehr große Mengen an Datensätzen in der Telekommunikation bezüglich Betrugs zu Analysieren. Dieser Ansatz nutzt im Zusammenspiel mit der grafischen Darstellung von Daten durch einen Computer die Fähigkeit des Menschen zur Mustererkennung. Als Beispiel kann die Bewertung der Anzahl von Anrufen unter verschiedenen Benutzern in verschiedenen Regionen genannt werden. Ein mögliches Zukunftsszenario wäre es die vom Menschen erkannten Muster als Software zu implementieren [BoHa02].

4.6 Hybrider Ansatz

Im Zuge des ACTS ASPeCT (Advanced Securtiy for Personal Communications Technologies) Projekts wurde eine Kombination von vier Komponenten und drei Methoden vorgeschlagen um die jeweiligen Stärken der einzelnen Ansätze zu nutzen. Daten durchlaufen sequentiell verschiedenen Komponenten des Systems und werden im Laufe der Bearbeitung ergänzt, gefiltert und bewertet. [ShHB99]

Folgende Komponenten kommen zum Einsatz:

- Unüberwacht lernendes neuronales Netz: Sehr gut um neue Methoden aufzuspüren und hoher negativer Vorhersagewert, d.h. Anrufe die keinerlei Anomalien aufweisen werden sofort ausgefiltert und belasten damit nicht die Folgestufen.
    - Analyse der gerufenen Nummer: Klassifizierung des Anrufziels. Diese Information wird dem ursprünglichen Datensatz hinzugefügt.
    - Analyse der rufenden Nummer: Bewertung der Abweichung des Verhaltensmusters.
- Überwacht lernendes neuronales Netz: Erkennt auf effiziente Weise Benutzer deren Verhalten dem von bereits früher erkannten Betrugsfällen entspricht. Hoher positiver Vorhersagewert, d.h. der Operator kann sich darauf verlassen, dass wenn das Netz etwas als betrügerisch einstuft wirklich etwas nicht stimmt.

- Regel basierender Teil: Da dieser Teil als White-Box angesehen werden kann eignet er sich sehr gut zur Erklärung warum ein Fall alarmiert wurde.

5 Neuronale Netze

Als Vorbild für künstliche neuronale Netze dient das menschliche Nervensystem das aus einem Geflecht von untereinander verbundenen Nervenzellen, den Neuronen, besteht.

Vergleicht man das menschliche Gehirn mit digitalen Rechnern kann man beträchtliche Unterschiede in der Art der Informationsverarbeitung feststellen. Während digitale Rechner große Datenmengen fehlerlos abspeichern und Instruktionen mit großer numerischer Exaktheit meist sequentiell ausführen zeichnet sich das Gehirn durch eine hohe Parallelität und Fehlertoleranz aus. Vergleicht man die Schaltzeiten des digitalen Rechners mit jenen des Gehirns so schaltet ein Rechner im Bereich von Nanosekunden deutlich schneller gegenüber dem Gehirn mit Schaltzeiten im Bereich von Millisekunden [Pass04a]. Damit ist die lokale Verarbeitungsgeschwindigkeit des Gehirns vergleichsweise langsam. Allerdings wird dies durch den hohen Parallelisierungsgrad zum größten Teil wieder ausgeglichen. Diese Eigenschaft ist auch der Grund weswegen die menschliche Reaktionszeit bei einfachen wie bei komplexen Situationen nahezu gleich ist [Kreb00].

	Gehirn	**Rechner**
Anzahl der Verarbeitungselemente	ca. 10^{11} (Neuronen)	ca. 10^{9} (Transistoren)
Art der Informationsverarbeitung	Massiv parallel	im wesentlichen seriell (klassische Architektur - Von Neumann)
Art der Speicherung	Assoziativ	Adressbezogen
Schaltzeit eines Verarbeitungselements	ca. 1ms (10^{-3} s)	ca. 1ns (10^{-9} s)
Theoretisch mögliche Schaltvorgänge pro Sekunde	ca. 10^{13}	ca. 10^{18}
Tatsächliche Schaltvorgänge pro Sekunde	ca. 10^{12}	ca. 10^{10}

Tabelle 1: Vergleich Gehirn/Rechner [Zell94]

Im Folgenden wird zuerst das Grundprinzip des Nervensystems angesprochenen, das als Vorbild für die danach beschriebenen künstlichen neuronalen Netze dient. Im Abschnitt über künstliche neuronale Netze wird vor allem auf vorwärts verkettete Netze und selbst organisierende Karten eingegangen. Diese werden im Zusammenhang mit der Anwendung der Missbrauchserkennung hauptsächlich eingesetzt. Andere Klassen wie z.B. partiell rückgekoppelte Netz (Jordan Netz) werden nur kurz angesprochen.

5.1 Natürliche Neuronale Netze

5.1.1 Gliederung des Nervensystems

Das menschliche Nervensystem gliedert sich in zwei Teile, in den zentralen und den peripheren. Dabei umfasst der zentrale Teil Gehirn und Rückenmark. Der periphere Teil unterteilt sich weiter in eine motorische (willkürliche) und eine vegetative (unwillkürliche)

Komponente. Wobei die motorische für willentliche Muskelbewegungen und die vegetative für innere Aktivitäten unseres Körpers wie z. B. den Herzschlag verantwortlich ist.

Das periphere Nervensystem kann auch als „Rezeptions- und Ausführungsorgan des zentrale Nervensystems“ bezeichnet werden [AnFA04]. Bestehend aus Nerven die den ganzen Körper durchziehen werden Impulse entweder von der Peripherie zum zentralen Nervensystem oder vom zentralen Nervensystem in die Peripherie getragen.

5.1.2 Funktionsprinzip des Nervensystems

Das Funktionsprinzip des Nervensystems kann auf folgende stark vereinfachte weise beschrieben werden: Periphere Sinnesreize werden über den Rezeptor wahrgenommen und über eine Nervenfaser dem zentralen Nervensystem zugeleitet, das die ankommenden Impulse verarbeitet. Dabei bildet das zentrale Nervensystem zahlreiche Neuronenkreise (Hintereinanderschaltungen von Nervenzellen) aus. Diese Neuronenkreise enden schließlich an einer Nervenzelle, deren ableitende motorische Faser wieder vom zentralen Nervensystem weg führt, um in der Peripherie die Impulse des zentralen Nervensystems z.B. an eine Muskelzelle oder Drüse weitergibt.

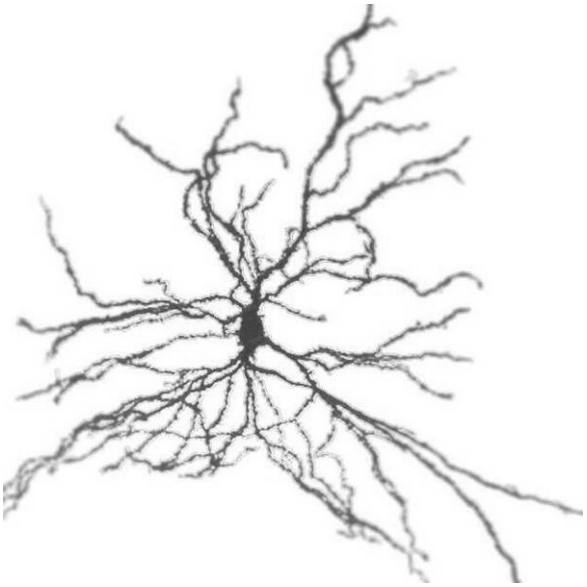

Abbildung 7: Gehirnzelle [Gähl06]

Abbildung 7 zeigt die Aufnahme einer eingefärbten, für das Gehirn typischen, pyramidenförmigen Nervenzelle.

5.1.3 Das Neuron

Die elementarsten Einheiten des Nervensystems sind Neuronen. Das zentrale Nervensystem besteht überwiegend aus diesen Neuronen. Sie ermöglichen durch ihre Aneinanderkettung und Vernetzung überhaupt erst die Funktion und Arbeitsweise des zentralen Nervensystems. Das gesamte menschliche Nervensystem enthält ca. 10^{11} Neuronen [Pass04b].

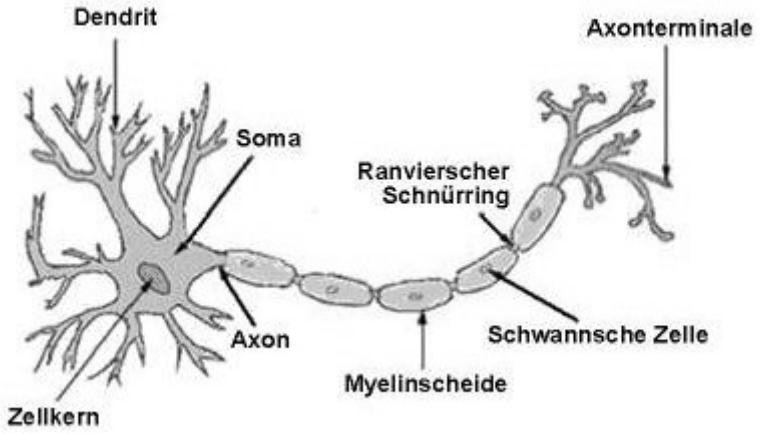

Abbildung 8: Schematischer Aufbau eines Neurons [Wiki06]

Ein Neuron hat die Fähigkeit, elektrische Signale weiterzuleiten. Wie in Abbildung 8 schematisch dargestellt besteht es aus einem Zellkörper, der den Zellkern enthält, und aus einem oder mehreren Fortsätzen. Diese Fortsätze (Nervenfasern) werden wiederum in Dendriten (für Erregungsempfang) und Axone (für Erregungsweitergabe) unterteilt. In der Regel besitze ein Neuron nur ein Axon, kann aber viele Dendriten haben.

Im Gegensatz zu den Dendriten ist das Ende eines Axons in der Regel baumartig verzweigt. Die Enden dieser Zweige bilden zusammen mit der Zellmembran der nachfolgenden Zelle und dem dazwischenlegenden Spalt die sog. Synapsen. An den Synapsen findet die Erregungsübertragung von einem Neuron zum nächsten statt. Dies geschieht dadurch, dass am Ende eines Zweiges ein elektrochemischer Vorgang ausgelöst wird bei dem eine chemische Substanz ausgeschüttet wird. Diese in den Spalt ausgeschüttete Substanz wird als Neurotransmitter bezeichnet und bewirkt eine Änderung des elektrischen Potenzials der Folgezelle [AnFA04].

Neurotransmitter können je nach Beschaffenheit des Rezeptors nicht nur erregend sondern auch hemmend wirken. Über Synapsen erhält ein Neuron von zahlreichen anderen Nervenzellen erregende oder hemmende Impulse gesendet. Die ankommenden erregenden Informationen werden summiert, hemmende Impulse abgezogen und die daraus resultierende Erregung weitergeleitet.

Synapsen die häufig genutzt werden arbeiten effektiver als andere, d.h. das die Stärke ihres Einflusses durch Lernen verändert wird [Pass04b].

5.2 Künstliche neuronale Netze

Künstliche Neuronale Netze, ein Teilbereich der künstlichen Intelligenz, versuchen die Arbeitsweise des menschlichen Gehirns zu imitieren. Genauso wie das zentrale Nervensystem durch die Zusammenschaltung von elementaren Entscheidungseinheiten (Neuronen) bestimmte Funktionen erlernen und erfüllen kann soll auch der Computer dazu in die Lage versetzt werden durch Lernvorgänge die Struktur eines neuronalen Netzes mit hemmenden und erregenden Impulsen nachzubilden um komplexe Aufgaben zu lösen.

Wie schon im Zusammenhang mit der Missbrauchserkennung beschrieben kann das künstliche neuronale Netz auf die Erfüllung spezieller Funktionen trainiert werden um

anschließend in der Lage zu sein, sowohl auf die Wiederholung der Trainingssituationen als auch auf neue Situationen angemessen zu reagieren. Diese Fähigkeit wird durch die Vernetzung von Neuronen und Gewichtung (hemmend bzw. erregend) ihrer Inputs erreicht. Der Output eines Neurons kann wiederum als Input für das nächste dienen, wodurch sich verschiedene unterschiedliche Netztopologien ergeben die sich zur Erfüllung verschiedener Aufgaben unterschiedlich gut eignen. Auf diverse Architekturen wird im Weiteren noch genauer eingegangen.

Der Einsatz von künstlichen neuronalen Netzen kann vor allem dann sinnvoll sein wenn keine effektive algorithmische Lösung bekannt und auch Versuche mittels regelbasierter Ansätze gescheitert sind [AnFA04].

5.2.1 Das Neuron

Wie bereits beschrieben versucht ein künstliches Neuron die Funktion des biologischen Vorbilds vereinfacht nachzubilden und kann wie in Abbildung 9 schematisch dargestellt werden.

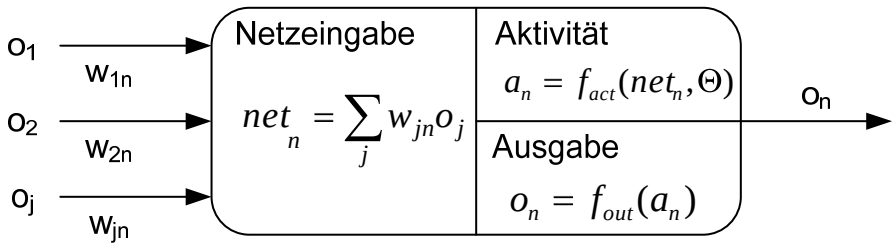

Abbildung 9: Schematische Darstellung eines künstlichen Neurons

Die wichtigsten Komponenten bzw. Funktionen eines Neurons:

- **Eingabeinformationen (o_j):** Impulse können von der Umgebung oder vom Output eines anderen künstlichen Neuron stammen. Typische Wertebereiche sind die reellen Zahlen, das Intervall [0, 1] oder die diskreten Werte {0, 1}. Statt {0,1} wird auch {-1,1} verwendet, dies hat den Vorteil, dass weitergehenden Verbindungen Werte ungleich 0 liefern wodurch oft bessere Ergebnisse aus der Lernphase zu erwarten sind.

- **Gewichtungen (w_{jn}):** Jeder Verbindung (oft auch als Synapse bezeichnet) in einem Neuronalen Netz ist eine reelle Zahl als Gewicht zugeordnet welches die Stärke (erregend oder hemmend) der Verbindung beschreibt. In diesen Verbindungsgewichten ist das Wissen des neuronalen Netzes gespeichert [Reif00], die Gewichtungen können also als Informationsspeicher eines neuronalen Netzes angesehen werden [Pass04b].

- **Propagierungsfunktion (net_n):** Eine Netzeingabe fasst die Informationen zusammen, die aus dem Netz in das Neuron einfließen. Als Funktion wird meist die einfache Summe der gewichteten Eingangssignale gebildet.

$$net = \sum_i w_i x_i$$

- **Aktivierungsfunktion (f_{act}):** Diese Funktion bestimmt aus der Netzeingabe (Ergebnis der Propagierungsfunktion) unter Berücksichtigung eines Schwellwertes θ_n (bias, threshold) den neuen Zustand des Neurons: $a_n = f_{act}(net_n, \theta_n)$

  Als Aktivierungsfunktion wird häufig eine der folgenden gewählt:

  - **Lineare Funktion:** $f(net) = net$

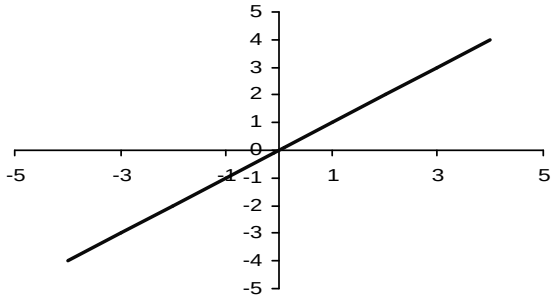

Abbildung 10: Lineare Aktivierungsfunktion

Wie in Abbildung 10 veranschaulicht wird durch diese Funktion die Netzeingabe 1:1 weitergegeben.

  - **Schwellwert Funktion:** $f(net) = \begin{cases} -1 : net < 0 \\ 1 : net \geq 0 \end{cases}$

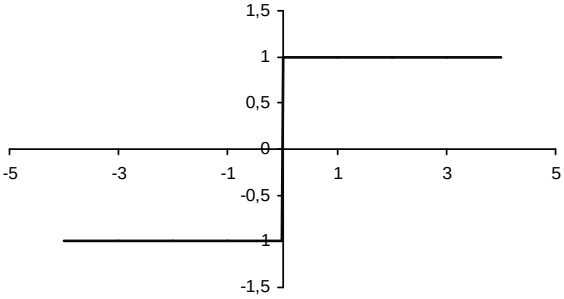

Abbildung 11: Schwellwertfunktion

Abbildung 11 stellt die Schwellwertfunktion grafisch dar. Durch diese Funktion wird die Netzeingabe folgendermaßen verarbeitet so dass die Funktion falls die Eingabe größer gleich 0 ist den Wert 1 liefert, sonst -1.

- **Logistische Funktion:** $f(net) = \frac{1}{1+e^{-net}}$

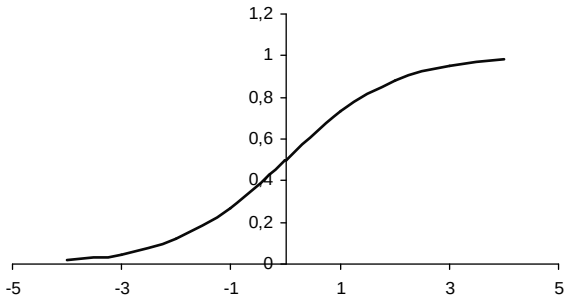

Abbildung 12: Logistische Funktion

Wie in Abbildung 12 dargestellt ist die diese Funktion eine nichtlineare die Ausgaben im Intervall]0, 1[liefert.

- **Tangens-Hyperbolikus:** $f(net) = \tanh(net)$

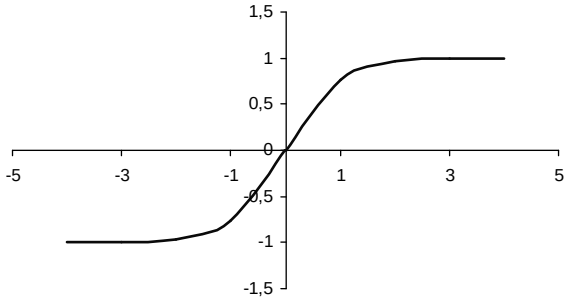

Abbildung 13: Tangens-Hyperbolikus

Abbildung 13 veranschaulicht das der Tangens-Hyperbolikus eine nichtlineare Funktion ist die Ausgaben im Intervall]-1, 1[liefert.

- **Schnelle Aktivierungsfunktion:** $f(net) = \frac{net}{1+|net|}$

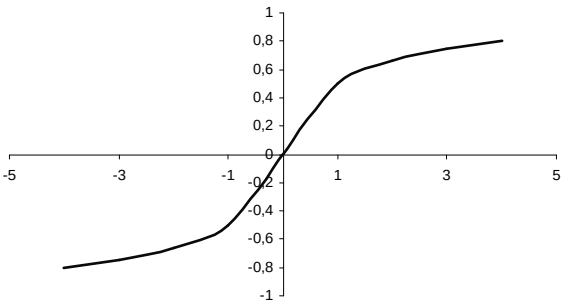

Abbildung 14: Schnelle Aktivierungsfunktion

Abbildung 14 stellt die schnelle Aktivierungsfunktion dar. Untersuchungen

ergaben, dass die Zeit zur Berechnung der Aktivierung und der ersten Ableitung bei der logistischen Funktion 2.3 Mal und beim Tangens-Hyperbolikus 4.3 Mal höher ist als bei der schnellen Aktivierungsfunktion [Weis98].

- **Ausgabefunktion (f_{out}):** Ermittelt aus der Aktivität a_n des Neurons den Ausgabewert o_n, der an das nachfolgende Neuronen weitergeleitet wird und dort als Bestandteil der Netzeingabe auftritt.
  Als Ausgabefunktion kommt häufig die Identität zum Einsatz, obwohl andere Funktionen nicht ausgeschlossen sind.

Da die Leistungsfähigkeit solcher Netze gewahrt werden soll werden Propagierungs-, Aktivierungs- und Ausgabefunktion sehr einfach gehalten und auf komplexe Funktionen bewusst verzichtet. Welche der angeführten Aktivierungsfunktionen schlussendlich zum Einsatz kommt hängt auch von der Anwendung ab, z.B. benötigt die Modellierung eines nichtlinearen Zusammenhangs eine nichtlineare und der später näher beschriebene Backpropagation Lernalgorithmus zum Training eines MLP eine differenzierbare Aktivierungsfunktion [Pass04b].

5.2.2 Netzwerktopologie

Zusätzlich zu den Eigenschaften der individuellen Neuronen, werden die Eigenschaften eines Neuronalen Netzes auch durch Netzwerktopologie und Lernalgorithmus bestimmt [LuSt97].

Unter der Topologie des Netzwerkes wird das Muster der Verbindungen zwischen den einzelnen künstlichen Neuronen verstanden. Daraus ergibt sich dass ein künstliches neuronales Netz aus einer Menge von Neuronen besteht, die durch gerichtete und gewichtete Verbindungen miteinander verknüpft sind. Durch die Menge dieser Verbindungen werden unterschiedliche Architekturen definiert [AnFA04].

Vorwärts Verkettetes Netz
Am Beispiel des in Abbildung 15 dargestellten Netz lassen sich die grundlegenden Strukturelemente eines vorwärts verketteten Netzes am einfachsten erläutern:

- **Eingabe-Neuron:** Ein Neuron, dass keine einkommenden Verbindungen besitzt (in Abbildung 15 Neuron 1 und 2).

- **Ausgabe-Neuron:** Ein Neuron, dass keine ausgehenden Verbindungen besitzt (in Abbildung 15 Neuron 7).

- **Verdeckte Neuronen oder verdeckte Schicht:** Neuronen, die weder zur Eingabe- noch zur Ausgabeschicht gehören, d.h. sie besitzen einen Eingang und einen Ausgang (in Abbildung 15 Neuron 3 bis 6).

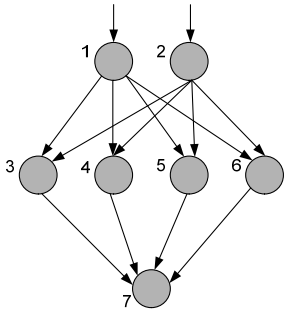

Abbildung 15: Vorwärts verkettetes Netz

Netz mit Rückkopplungen
Abbildung 16 stellt ein Netz mit Rückkopplung dar. Bei dieser Kategorie übermittelt das Ausgabe-Neuron (6) seinen Output an ein so genanntes Kontext-Neuron (7) dessen Output als zusätzlicher Input der verdeckten Schicht dient.

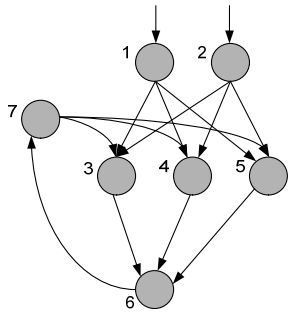

Abbildung 16: Netz mit Rückkopplungen

Vollvernetzte Schichten ohne direkte Rückkopplungen
Wie in Abbildung 17 dargestellt bilden die Neuronen 1 und 2 die Eingabeschicht und sind deshalb nicht rückgekoppelt. Alle anderen Neuronen sind gegenseitig verbunden.

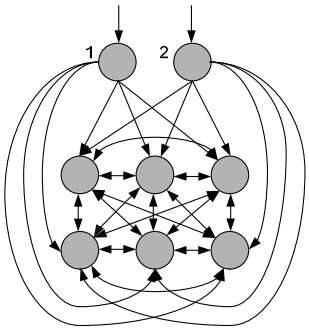

Abbildung 17: Vollvernetzte Schicht

Anmerkung: Häufig stellt die in der Literatur angegebenen Anzahl an Schichten die der Gewichtsschichten und nicht die der Neuronen dar [Pass04b].

5.2.3 Lernalgorithmen

Die verwendeten Lernverfahren bzw. der verwendete Lernalgorithmus legt fest auf welche Art das Netzwerk für seine zukünftigen Aufgaben trainiert wird. Grundlegend kann, im

Zusammenhang mit künstlichen neuronalen Netzen, unter folgenden Kategorien des Lernens unterschieden werden:

Überwachtes Lernen
Bei dieser Form des Lernens müssen die Eingabedaten a priori mit den zu erwarteten Ergebnissen versehen werden. Diese Daten werden dann zum Training des Netzes herangezogen. Anhand dieser festen Lernaufgabe werden die Gewichte der Verbindungen verändert. Diese Methode wird auch als Lernen mit Lehrer bezeichnen [Kreb00]. Wird der Gesamtfehler Null oder wird ein definierter Schwellwert unterschritten war der Lernprozess erfolgreich.

Bestärkendes Lernen
Mitunter kann für die Eingabe nicht gesagt werden, wie richtig oder wie falsch diese ist. Man kann nur feststellen, ob die Ausgabe richtig oder falsch ist. Das Netz muss mit diesen groben Informationen trainiert werden.

Unüberwachtes Lernen / Wettbewerbslernen
Aus biologischer Sicht am plausibelsten ist das unüberwachte Lernen [AnFA04]. Bei dieser Methode lässt man das Netz während der Lernphase selbständig nach Strukturen (Clusteranalyse) in der Datenmenge suchen. Man spricht in diesem Zusammenhang auch vom Lernen mit freier Lernaufgabe (dem Netz werden nur Eingabemuster präsentiert und keine gewünschten Ausgaben) [Pass04b]. Auf einen Eingabevektor darf jeweils nur ein Neuron reagieren. Dasjenige Neuron, das die höchste Aktivierung aufgrund eines Eingabevektors aufweißt, gibt 1 aus. Alle anderen 0. Die Neuronen stehen also untereinander im Wettbewerb. Aus diesem Grund spricht man auch von einer „winner-takes-all“ Aktivierungsfunktion [Kreb00]. Eine Art von künstlichem neuronalen Netz welches diese Methode anwendet sind die selbst organisierenden Karten die später näher behandelt werden. Weiters sei bereits hier angemerkt, dass bei diesem Verfahren nicht nur ein einziges Neuron gewinnen kann sondern über eine Nachbarschaftsfunktion auch Neuronen in lokaler Nähe (genauer die Gewichtsvektoren) beeinflusst werden können. Anwendungen hierfür sind z.B. Klassifikationsversuche, wobei Art und Umfang der Klasseneinteilung vorher nicht bekannt sind.

Im Weiteren wird in den speziellen Abschnitten zu den verschiedenen Netztopologien auf bestimmte Lernalgorithmen, wie z.B. der Backpropagation Algorithmus bei Multi Layer Perzeptrons, näher eingegangen.

5.3 Vorwärts verkettete neuronale Netze

Vorwärtsverkettete neuronale Netze basieren auf dem Prinzip mehrere Schichten von künstlichen Neuronen nacheinander zu verketten. Der minimale Aufbau dieser Netze besteht dabei aus einer Eingabe- und aus einer Ausgabe- Schicht. Dazwischen kommen in den meisten Anwendungen ein oder mehrere so genannte verdeckte Schichten zum Einsatz. Das bestimmende Merkmal dieser Netze ist, dass nur Neuronen verschiedener Schichten und alle nur in Richtung Ausgabeschicht miteinander verknüpft sind.

Im Folgenden werden einige Varianten dieses Prinzips vorgestellt.

5.3.1 Perzeptron

Das Perzeptron wurde Ende der 50er Jahre von Frank Rosenblatt ausgehend vom Vorbild des menschlichen Auges zur Simulation der Netzhaut entwickelt und ist die einfachste Form eines vorwärts verketteten neuronalen Netzes. Da es keine innere, verdeckte Schicht besitzt wird es auch oft als einstufiges vorwärts verkettetes Netz bezeichnet.

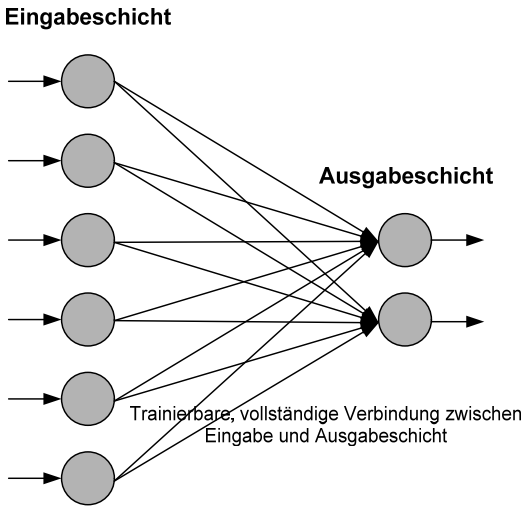

Abbildung 18: Perzeptron

Wie in Abbildung 18 dargestellt nimmt die Eingabeschicht die Information auf und sendet diese über trainierbare Verbindungen mit modifizierbaren Gewichten an die Ausgabeschicht.

Unter dem Begriff Perzeptron wird auch oft das binäre Modell des Perzeptrons verstanden, bei dem sowohl die Eingaben als auch die Aktivierungen der Neuronen nur die binären Werte 0 und 1 (bzw. -1 und 1) annehmen dürfen. Die Gewichte können beliebige reelle Werte annehmen.

In diesem Modell werden Neuronen der Eingabeschicht nicht näher spezifiziert. Es genügt, wenn man sie als binäre Eingaben betrachtet. Jedes einzelne Neuron summiert seine Inputs, die sich als Output der Vorgängerneuronen o_i multipliziert mit dem jeweiligen Gewicht w_{ij} ergeben und wendet auf diese Netzeingabe net_j eine binäre Schwellenwertfunktion an. Der Output o_j ist 1, falls die Netzeingabe größer oder gleich dem der Schwellenwert T_j ist, andernfalls ist die Ausgabe 0.

$$net_j = \sum_i o_i w_{ij}$$

$$o_j = a_j = \begin{cases} 1 : net_j \geq \theta_j \\ 0 \end{cases}$$

Ein Perzeptron kann nicht nur ein einzelnes Ausgabeneuron haben, sondern beliebig viele, ohne dass sich am Prinzip viel ändern würde.

Im folgendem werden die Begriffe der Repräsentierbarkeit, der Lernfähigkeit und die Eigenschaft der linearen Separierbarkeit kurz beschrieben da diese unter anderem für Aussagen über die Leistungsfähigkeit von neuronalen Netzen von Bedeutung sind.

Die **Repräsentierbarkeit** bezeichnet die Eigenschaft eines Netzes, eine gegebene Funktion mit realisieren zu können. Die Topologie des Netzes ist vorgegeben, aber alle Gewichte und Schwellenwerte dürfen optimal gewählt werden.

Die **Lernfähigkeit** ist die Eigenschaft eines Lernalgorithmus, ein Netzwerk eine repräsentierbare Funktion lernen zu lassen, d.h. die Gewichte und Schwellenwerte durch den Algorithmus korrekt zu bestimmen [GaReoJ].

Die Eigenschaft der **lineare Separierbarkeit** bringt zum Ausdruck, dass sich ein einfaches Perzeptron mit zwei Eingabewerten und einem einzigen Ausgabeneuron zwar zur Darstellung einfacher logischer Operatoren wie AND und OR eignet, aber nicht in der Lage ist den Operator XOR zu erlernen.

Diese Unterscheidung zwischen **Repräsentierbarkeit** und **Lernfähigkeit** ist aus historischen Gründen sehr wichtig, weil sie in der ersten Blütephase neuronaler Netze noch nicht bekannt war und daher ein berühmtes Theorem, das Perzeptron-Lern-Theorem (perceptron convergence theorem) von F. Rosenblatt über die Fähigkeit des Perzeptron-Lernalgorithmus, meist falsch verstanden wurde [GaReoJ].

Perzeptron-Lern-Theorem:
Der Lernalgorithmus des Perzeptrons konvergiert in endlicher Zeit, d.h. das Perzeptron kann in endlicher Zeit alles lernen, was es repräsentieren kann.

Die Anpassung (Training) an eine zu lernende Funktion geschieht über so genanntes überwachtes Lernen mit Hilfe der **Delta- oder Widrow-Hoff-Regel**, ein Verfahren zur Adaption der Gewichte innerhalb eines vorwärts verketteten Netzes mit nur einer Schicht. Die Grundidee ist dass die Gewichte nur soweit verändert werden wie sie auch zum Fehler beigetragen haben. Um dies Festzustellen benötigt man, wie bei überwachtem Lernen üblich, den erwarteten Output zum jeweiligen Input. Zusätzlich dazu bestimmt die Lernrate wie stark das Gewicht pro Durchgang verändert wird.

An folgendem Beispiel, einem Perzeptron zur Darstellung eines AND Operators, wird die Delta-Regel praktisch dargestellt.

<u>Angaben:</u>

- T = 0.5
- Input: $x_1 = (o_1, o_2) = (1,1)$.
- Abbildung 19 veranschaulicht das Beispiel.

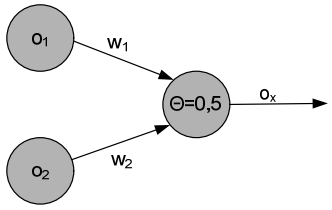

Abbildung 19: Perzeptron AND Beispiel [AnFA04]

Ist die Netzeingabe größer als der Schwellenwert so ist der Output 1 ansonsten 0.

$$net = w_1 o_1 + w_2 o_2 = 0 \cdot 1 + 0 \cdot 1 = 0$$

net … Netzeingabe
o_i … Output des Eingabeneurons i
w_i … Gewichtung der Verbindung zu Eingabeneuron i

Da 0 den Schwellenwert nicht übersteigt ist der Output auch 0. Daraus ergibt sich mit Hilfe des so genannten Teaching Output der Fehler folgendermaßen:

$$err_{x1} = t_{x1} - o_{x1} = 1 - 0 = 1$$

err_{x1} … Aktueller Fehler bei anlegen des Eingabemusters 1
t_{x1} … Teaching Output (gewünschter Output für das Eingabemuster 1)
o_{x1} … Aktueller Output des Ausgabeneurons bei anlegen des Eingabemusters 1

Mit diesem Fehler ist es nun möglich die Verbindungsgewichte die einen Beitrag zum Fehler geleistet haben zu korrigieren.

$$w_1' = w_1 + err_{x1} o_1 = 0 + 1 \cdot 1 = 1$$

w_1 … Aktuelles Gewicht der Verbindung 1
w_1' … Neues Gewicht der Verbindung 1

Analog wird auch das neue Gewicht für die zweite Verbindung berechnet. Beide Gewichte erhalten dadurch neu den Wert 1.

Werden noch weitere Muster wie (1,0) und danach (0,1) usw. benutzt zeigt sich dass diese Gewichte nie lernen werden, d.h. sie werden immer von einem Wert zum anderen springen und niemals in einem stabilen Zustand konvergieren. Aus diesem Grund benötigt man die Lernrate damit die Stärke der Modifizierung der Gewichte im Verhältnis zum Beitrag zum Fehler bestimmt werden kann.

Die Einführung dieses neuen Faktors führt schließlich zu folgender Formel für die Anpassung der Gewichte:

$$w_i' = w_i + \lambda (t_x - o_x) o_i$$

λ … Lernrate (kleiner als Schwellenwert)

Grundsätzlich können auch mehrschichtige Perzeptrons (engl. Multi Layer Perceptron, MLP) konstruiert werden. Diese werden im folgenden Kapitel der Backpropagation-Netze näher behandelt

5.3.2 Multi Layer Perzeptron / Backpropagation-Netz

In einem vorwärts gerichteten Netz befinden sich mehrere Schichten von künstlichen Neuronen deren Gewichte der Verbindungen variabel sind. Der Aufbau gestaltet sich folgendermaßen, dass zwischen einer Eingabe- und einer Ausgabe- Schicht eine oder mehrere innere oder verdeckte Schichten (siehe Abbildung 20) angeordnet sind. Dabei sind die

Neuronen schichtweise vollständig, nur vorwärts und nicht schichtübergreifend verbunden. Diese Netze kommen in der Praxis am häufigsten zum Einsatz [Pass04b].

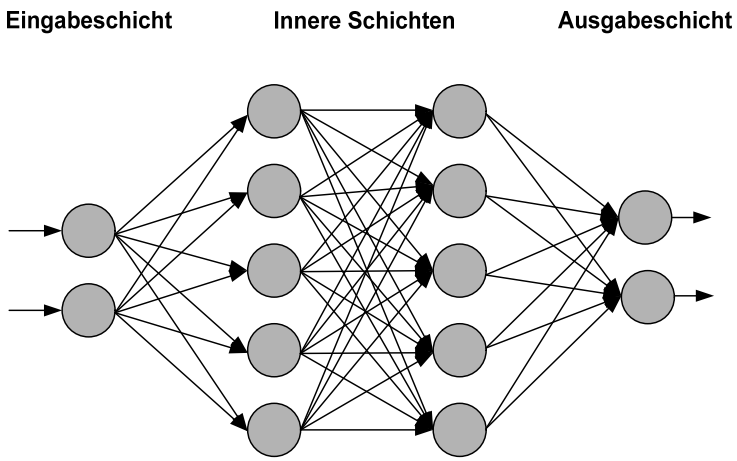

Abbildung 20: Multi Layer Perzeptron

Mit Hilfe von mathematischen Verfahren wie dem Backpropagation Algorithmus, einem Gradientenverfahren, wird es ermöglicht für innere Schichten an denen weder ein berechneter noch erwarteter Ausgabewert bestimmt werden kann die Gewichte zu modifizieren. Gewichtsmodifizierungen werden beginnend mit den Verbindungen zur Ausgabeschicht schichtweise rückwärts in Richtung Eingabeschicht vorgenommen. Dieses Verfahren basiert auf der ersten Ableitung einer Funktion und kann in die Kategorie des überwachten Lernens eingeordnet werden. Während im eindimensionalen Fall die erste Ableitung dem Anstieg in einem gegebenen Punkt entspricht zeigt sie für mehrdimensionale Funktionen in Richtung des stärksten Anstieges. Der Grundgedanke dahinter ist das Minimum einer Fehlerkurve zu finden da damit theoretisch auch der kleinste Fehler in diesem Punkt vorhanden ist. Dabei wird der Gesamtfehler nicht für ein bestimmtes Eingabemuster sondern in Abhängigkeit von den Gewichten betrachtet (Fehlerfläche).

Durch dieses Prinzip lässt sich eine Formel für die notwendigen Änderungen der Gewichte erstellen:

$$\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial w_{ij}}$$

η … Lernrate
E … Fehler

Das negative Vorzeichen ist nötig damit die Funktion in Richtung des Minimums anstatt in Richtung des Maximums zeigt.

Man kann grundsätzlich zwischen zwei Verfahren zur Korrektur der Gewichte unterscheiden dem Batch- und dem Online-Verfahren. Während das Batch-Verfahren die Änderung der Gewichte erst nach dem Bearbeiten der gesamten Lernaufgabe (einer Epoche) vornimmt modifiziert das Online-Verfahren die Gewichte jeweils nach jedem Trainingsmuster.

Da Fehler und Gewichte zusammenhängen, kann die Formel für die Fehlerkorrektur noch konkretisiert werden:

$$\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}}$$

Dabei definiert das Produkt der ersten beiden Faktoren das Fehlersignal.

$$\delta_j = -\frac{\partial E}{\partial net_j}$$

Der letzte Faktor kann noch folgendermaßen vereinfacht werden.

$$\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_i o_i w_{ij} = o_i$$

Nach diesen Umformungen ergibt sich dann für die Änderung der Gewichtungen folgende Formel

$$\Delta w_{ij} = \eta \cdot \delta_j \cdot o_i$$

Dieser Wert wird schließlich wie bei der Delta-Regel zu dem ursprünglichen Wert addiert.

Für Neuronen der Ausgabeschicht und der verdeckten Schichten unterscheidet sich die Ermittlung des Fehlersignals.

Für Neuronen der Ausgabe Schicht kann der Fehler direkt als Differenz zwischen Lerneingabe (Teaching Output) und berechnetem Output ermittelt werden. Dabei ist E der Fehler für ein Muster, t_j die Lerneingabe und o_j die Ausgabe von Neuron j bei diesem Muster. Das Quadrat verhindert dass sich negative und positive Abweichungen gegenseitig aufheben und der Faktor ½ bewirkt eine Vereinfachung der ersten Ableitung. Zur Bestimmung optimaler Gewichte spielt es keine Rolle, ob man den ganzen oder den halben Fehler minimiert. Die Minimierung des Quadrats des euklidischen Abstandes anstelle des euklidischen Abstandes wirkt sich ebenfalls nicht auf die Optimierung der Gewichte aus.

$$E = \frac{1}{2} \sum_j (t_j - o_j)^2$$

$$-\frac{\partial E}{\partial o_j} = -\frac{\partial}{\partial o_j} \left(\frac{1}{2} \sum (t_j - o_j)^2\right) = (t_j - o_j)$$

Zur Modifizierung der Gewichtungen der inneren Schichten wird nun der Backpropagation Algorithmus angewandt, der im Wesentlichen eine Übertragung des Fehlers in die umgekehrte (d.h. rückwärts) Richtung bewirkt. Dabei werden ausgehend von der Ausgabeschicht zurück bis zur ersten inneren Schicht schrittweise jede der Schichten (bzw. die Gewichtungen) modifiziert.

Für die logistische Funktion kann dadurch der Fehler eines inneren Neurons aus den Fehlersignalen der folgenden Neuronen ermittelt werden.

$$-\frac{\partial E}{\partial o_j} = -\sum_k \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial o_j} = \sum_k (\delta_k \cdot \frac{\partial}{\partial o_j} \sum_j o_j w_{jk}) = \sum_k \delta_k w_{jk}$$

Durch einsetzen der beiden eben hergeleiteten Formeln in

$$\delta_j = -\frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} = -\frac{\partial E}{\partial o_j} \cdot \frac{\partial a_j(net_j)}{\partial net_j} = -\frac{\partial E}{\partial o_j} \cdot a'_j(net_j)$$

ergeben sich nun auch 2 unterschiedliche Formeln, je nach Lage der Neuronenschicht, um das Fehlersignal und damit auch die Anpassung der Gewichte zu ermitteln.

Für die logistische (sigmoide) Funktion, die wegen ihrer relativ einfach zu ermittelten ersten Ableitung verwendet wird bedeutet dies konkret folgendes:

Funktion	$f(x) = \frac{1}{1+e^{-x}}$
Erste Ableitung	$f'(x) = \frac{d}{dx}(\frac{1}{1+e^{-x}}) = f(x) \cdot (1 - f(x))$
Fehlersignal Ausgabeschicht	$\delta_j = o_j(1-o_j)(t_j - o_j)$
Fehlersignal verdeckte Schicht	$\delta_j = o_j(1-o_j)\sum_k \delta_k w_{jk}$

Tabelle 2: Logistische Funktion – Backpropagation

5.3.3 Lernziele

Als Lernziele können die Approximation oder die Generalisierung genannt werden. Je nach gewünschtem Ziel ist ein anderes Vorgehen beim Training des Netzes von Vorteil.

- Approximation: Unter Approximation versteht man die Annäherung einer nichtlinearen Funktion an die Eingabemuster (Stützstellen), d.h. dass nach erfolgtem Training die Differenz zwischen gewünschtem und tatsächlichem Output möglichst gering sein sollte. Durch den Einsatz von beliebig vielen Neuronen in der verdeckten Schicht kann eine Funktion beliebig genau approximiert werden [ReMa99]

- Generalisierung: Unter Generalisierung versteht man die Eigenschaft dass die nichtlineare Funktion auch Daten zwischen den Stützstellen, d.h. bisher nicht bekannte Eingabemuster gut annähert. Nach dem Training soll die Differenz zwischen tatsächlichem und gewünschtem Output auch für bisher unbekannte Eingaben minimal sein. Um dies zu erreichen wird die Lernaufgabe meist in 2 Teilmengen aufgeteilt. Mit der einen wird trainiert, mit der anderen das Ergebnis des Trainings überprüft. Dieses Ziel ist schwieriger zu erreichen als das der Approximation an die Eingabedaten [Pass04b]

5.3.4 Netzgröße

Eine schwierige Aufgabe bei der Konstruktion eines Neuronalen Netzes ist die Anzahl der Neuronen der inneren Schichten zu bestimmen.

Im direkten Zusammenhang damit steht die Speicherkapazität welche durch die Anzahl der Verbindungen zwischen den Neuronen der inneren Schichten festgelegt ist.

Die formelle Definition der Speicherkapazität lautet wie folgend: *Die Speicherkapazität eines Netzes wird durch die Anzahl der Muster bestimmt, für die eine fehlerlose Zuordnung der zufälligen Eingabe-Muster zu den zufällig gesetzten Ausgabe-Muster möglich ist.* [AnFA04]

Ist diese zu groß d.h. ist die innere Schicht zu groß, kann sich das Netz jedes angewandte Trainingsmuster merken und lernt es gewissermaßen auswendig. Ein Indiz dafür ist ein Fehler von 0 während des Trainings. Damit ist das Netz nicht fähig auf neue Muster angemessen zu reagieren.

Im Falle des Lernziels der Generalisierungsfähigkeit soll genau diese Eigenschaft maximiert werden, d.h. der Generalisierungsfehler ist zu minimieren. Der Generalisierungsfehler ist damit zum einen von der Netzgröße und zum anderen von der Anzahl der Trainingsdurchläufe abhängig.

Ist das Lernziel die Approximation sind Multi Layer Perzeptron Netze mit nur einer verdeckten Schicht theoretisch dazu in der Lage eine stetige Funktion beliebig genau zu approximieren. Weiters kann gesagt werden, dass durch zwei inneren Schichten, mit insgesamt weniger verdeckten Neuronen, eine identische Güte bei der Approximation erreicht wird wie mit einer [Pass04b].

Die Größe der inneren Schicht kann aber auch nicht einfach sehr klein gehalten werden denn falls die Speicherkapazität, durch die ungenügend große Anzahl der Verbindungen, zu klein ist können überhaupt keine Muster gelernt werden.

Daraus ergibt sich, dass zwischen ausreichender Genauigkeit und einer Überanpassung ein Kompromiss gefunden werden muss, oder anders gesagt ein Kompromiss muss in der Größe des Netzes gefunden werden. Einerseits soll das Netz klein genug gewählt werden um die Generalisierungsfähigkeit zu erhalten, andererseits muss es aber auch in der Lage sein eine gewisse Anzahl von Mustern zu speichern.

Bezüglich der Anzahl der Neuronen in inneren Schichten kann keine allgemeine Regel angegeben werden, jedoch findet man in der Literatur verschiedene heuristische Methoden. Eine Möglichkeit um die Anzahl der Neuronen in inneren Schichten näherungsweise zu bestimmen ist das Training mit einem Muster das nur aus zufälligem Rauschen (white noise) besteht. Erst wenn der Trainingsfehler ungleich null ist wird das Training gestoppt da dies auf Abstrahierungsversuche des Netzes schließen lässt. Eine andere ist, dass die Anzahl der verdeckten Neuronen als Wurzel aus der Summe von Ein- und Ausgabeneuronen bestimmt wird und dazu noch einige addiert werden.

In der Praxis kommen hauptsächlich Netze mit nur einer verdeckten Schicht zum Einsatz, in manchen Fällen welche mit zwei. Netze mit drei oder mehr inneren Schichten kommen nur sehr selten zum Einsatz.

5.3.4.1 Strukturoptimierung - Entfernen von Verbindungen und Neuronen (Pruning)

Die Entfernung von Verbindung (ein destruktives Verfahren) bezweckt eine Optimierung und Vereinfachung des künstlichen neuronalen Netzes. Grundsätzlich wird hier zwischen statistischen und genetischen Verfahren unterschieden. Im Folgenden wird auf das statistische Verfahren eingegangen, genetische Algorithmen werden in Anhang 9 kurz angesprochen und werden eher in die Kategorie der hybriden Verfahren zur Optimierung von neuronalen Netzen eingeordnet.

Zeigt sich, dass Verbindungen während längerer Zeit einen Wert nahe Null behalten trägt diese Verbindung sehr wahrscheinlich keinen Anteil am Ergebnis und man kann sie entfernen. Um jedoch durch dieses Entfernen der Verbindungen keine Speicherkapazität zu verlieren wird für jede gelöschte Verbindung einer Ebene auf der gleichen Ebene zufällig eine neue hinzugefügt. Der Grad der Vernetzung bleibt dabei, vorausgesetzt das Netz ist nicht zu 100% vernetzt, konstant. Mit einem vollständig verbundenem Neuronalen Netz ist dieses Verfahren daher nicht anwendbar [AnFA04]. Es besteht die Möglichkeit, dass nach der Anwendung dieses Verfahrens einige Eingabeneuronen keine Verbindungen mehr besitzen. Diese Neuronen können ebenfalls aus dem Netz entfernt werden.

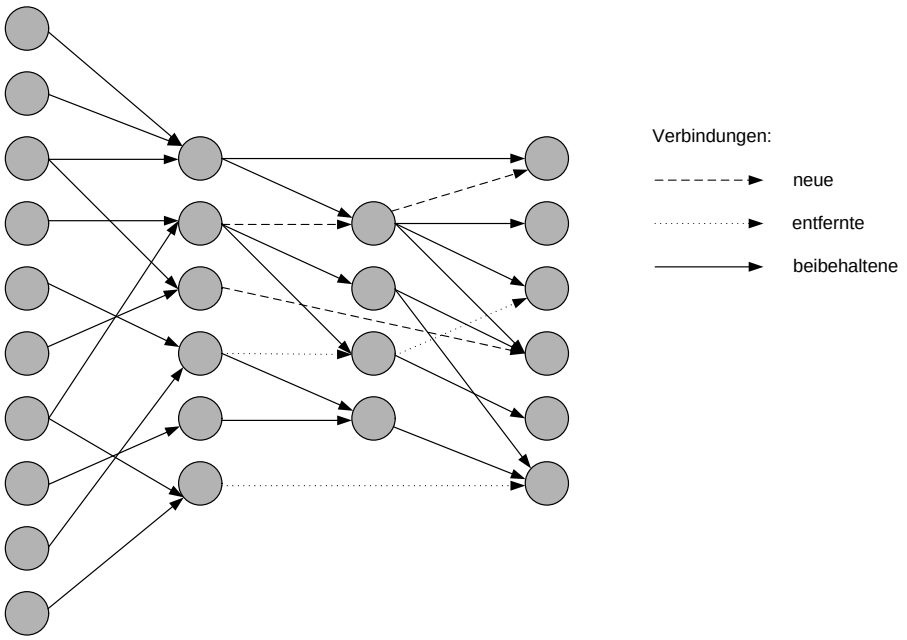

Abbildung 21: Optimieren der Verbindungen

Abbildung 21 stellt diesen Vorgang grafisch dar. Für entfernte Verbindungen, die für die Berechnung der Ausgabe nicht relevant sind, werden neue erstellt.

Durch diese Methode erreicht man das nur Verbindungen und Neuronen die auch zum Ergebnis beitragen erhalten bleiben, was die Effizienz des Netzes wesentlich erhöht. Die Generalisierungsfähigkeit wird dadurch jedoch kaum verbessert [Pass04b].

5.3.5 Qualität des trainierten Netzes

An ein Netz das in der Praxis eingesetzt wird werden Anforderungen wie Generalisierungsfähigkeit, Reproduzierbarkeit und Robustheit gestellt. Ein weiterer Faktor der sich auf die Güte des neuronalen Modells auswirkt ist die Wahl der Aktivierungsfunktion

welche von der Problemstellung abhängt [Pass04b]. Im folgenden Abschnitt werden Methoden vorgestellt wie man die Generalisierungsfähigkeit eines gegebenen oder zu optimierenden Netzes robust erlangen kann bzw. wie diese Fähigkeit überprüft werden kann.

5.3.5.1 Generalisierungsfähigkeit

Wie schon beschrieben wird mit dieser Fähigkeit das Verhalten des Netzes bei unbekannten Daten beschrieben.

Methoden zur Überprüfung der Generalisierungsfähigkeit:

- „hold out“ - Methode: Die Menge der Eingabedaten wird in eine Trainings- und eine Validierungsmenge aufgeteilt. Nachdem das Training abgeschlossen wurde wird mittels der Validierungsdaten die Generalisierungsfähigkeit überprüft.

- „k fold cross validation“ - Methode: Die Menge der Eingabedaten wird in k gleich große Mengen aufgeteilt. Jeder Teil wird als Validierungsmenge für ein Netz das mit den restlichen Daten trainiert wurde herangezogen. Aus den einzelnen Bewertungen wird der Mittelwert bestimmt. Häufig wird die Kreuzvalidierung zuerst zu der Optimierung der Netzstruktur eingesetzt, und nur auf Testdaten angewandt, und erst danach mit einem eigenen Satz von Validierungsdaten die Generalisierungsfähigkeit des Netz überprüft. Diese Methode ist vor allem dann zu empfehlen wenn wenige Eingabemuster zur Verfügung stehen [Pass04b].

- Weitere Methoden sind zum Beispiel:

    - Bootstrap: Aus der Menge der Eingabedaten werden mehrere (Bootstrap) Stichproben mit zurücklegen gezogen.

    - Jackknife: Aus der Menge der Eingabedaten werden mehrere Teilmengen ohne zurücklegen gezogen.

Wird das Netz zu lange trainiert kann es zu einer so genannten Überanpassung an die Trainingsdaten kommen. Während der Gesamtfehler für Trainingsmuster sinkt steigt er nach einer gewissen Anzahl an Epochen für unbekannte Muster wieder an (siehe Abbildung 22).

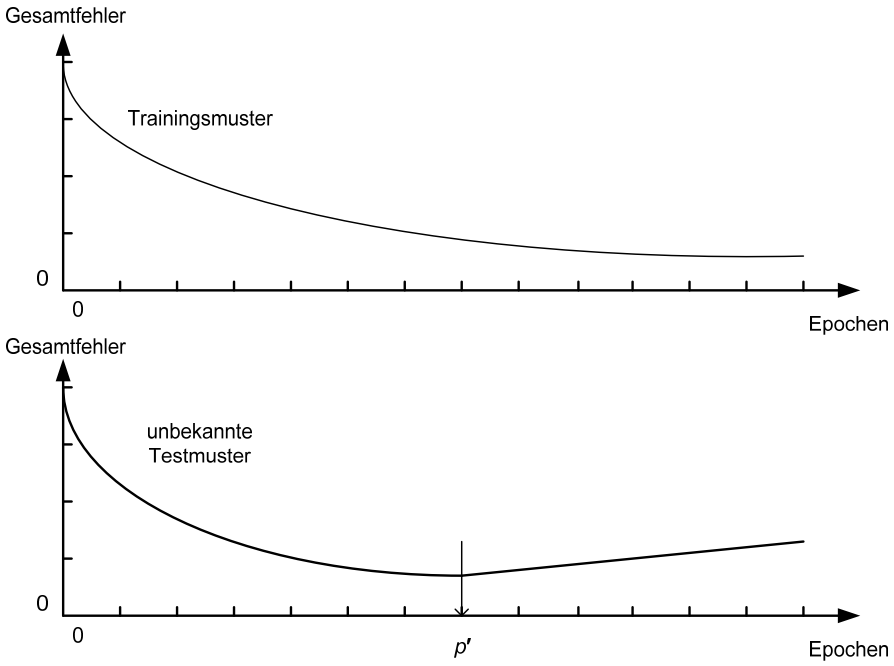

Abbildung 22: Überanpassung [Pass04b]

Methoden zur Vermeidung von Überanpassungen:

- Regularisierungs - Verfahren: Zum Ergebnis der Fehlerfunktion wird ein „penalty-term“ addiert.

- Verrauschte Daten: Trainingsdaten werden verfälscht indem die Datensätze mit weißem Rauschen (normalverteilte Störung; der Mittelwert ist 0, die Varianz ist konstant) überlagert werden.

- Informationsverdichtung: Die Kombination aus einem Netz mit vielen Gewichten und vielen signifikanten Input-Merkmalen erhöht die Wahrscheinlichkeit einer Überanpassung. Eine Reduzierung der Merkmale (Informationsverdichtung) wirkt sich dadurch positiv aus und ist die wichtigste Maßnahme zur Vermeidung von Überanpassungen [Pass04b].

- Optimierung der Netzstruktur: Wie bereits bei der Informationsverdichtung angesprochen hängt die Wahrscheinlichkeit einer Überanpassung von der Anzahl der Freiheitsgrade ab. Aus diesem Grund wirkt sich eine optimierte Netzstruktur, welche die Anzahl der Freiheitsgrade niedrig hält, ebenfalls positiv aus.

- Kombination von mehreren Netzausgaben: Mehrere neuronale Netze arbeiten an einem Problem. Dabei können die einzelnen Netze verschiede Teilaufgaben („mixture of experts“) oder aber auch die gleiche Aufgabe bearbeiten.

- Verbindung mit identischen Gewichten: Gruppen von Gewichten dürfen nur selbe Werte annehmen („weight sharing“), dies bewirkt gleichfalls eine Reduktion der Freiheitsgrade.

5.3.6 Probleme

Probleme ergeben sich bei der Backpropagation Methode, einem Gradientenverfahren, vor allem daraus dass es ein lokales Verfahren ist. Es hat keine Informationen über die Fehlerfläche insgesamt, sondern nur Kenntnis der lokalen Umgebung (des Gradienten bzw. bei Erweiterungen des Verfahrens einige zusätzliche vorher besuchte Stellen der Fehlerfläche).

Im Folgenden werden die wichtigsten Probleme angeführt und Lösungsansätze vorgestellt.

5.3.6.1 Symmetry Breaking

Das Problem des „symmetry breaking“ bezieht sich auf die Initialisierung der Gewichtungen vollständig vorwärts verketteter neuronaler Netze. Diese dürfen am Anfang keinesfalls gleich groß gewählt werden. Aus dem Grund, weil bei einer Initialisierung mit gleichen Werten (z.B. mit 0) das Backpropagation Lernverfahren nicht mehr dazu in der Lage ist in vorangehenden Schichten, ausgehend von der Ausgabeschicht, unterschiedliche Gewichtungen anzunehmen. Bezogen auf ein zweistufiges Netz bedeutet dies, dass in der ersten Schicht keine unterschiedlichen Gewichte mehr ausgebildet werden [GaReoJ].

Dieses Problem kann durch eine zufällige initiale Zuweisung von (kleinen) Gewichtungen gelöst werden. Die Folge daraus ist, dass alle Knoten eine Netzeingabe von ungefähr Null haben und die erste Ableitung der logistischen Funktion (vorausgesetzt eines Schwellwerts T = 0) am größten ist. Von diesem Ausgangszustand erfolgt die Anpassung am schnellsten. Eine weitere Möglichkeit wäre es die Initialwerte der Gewichtungen der Verbindungen der jeweiligen Neuronen von der Anzahl der Verbindungen abhängig zu machen so dass die Netzweingabe 1 beträgt wenn alle Inputs 1 betragen. In diesem Fall werden die Gewichtungen folgendermaßen ermittelt: $w_{ij} = \frac{1}{Anzahl\ der\ Eingabeverbindungen\ j}$

5.3.6.2 Lokale Minima der Fehlerfläche

Gradientenverfahren haben das Problem, dass sie anstatt das globale Minimum zu erreichen möglicherweise in einem lokalen Minimum der Fehlerfläche enden (siehe Abbildung 23 links oben). Ein Gradientenverfahren findet in diesem Fall also nur ein suboptimales Minimum. Bei neuronalen Netzen kann man davon ausgehen, dass mit steigender Dimension die Fehlerfläche immer zerklüfteter und damit auch die Wahrscheinlichkeit in einem lokalen anstatt dem globalen Minimum zu landen immer größer wird.

Zur Lösung dieses Problems gibt es wenige allgemeingültige Verfahren da es sehr stark von der Anwendung und von der Kodierung der Eingaben abhängt in welchem Ausmaß sich lokale Minima ausbilden. Andererseits hat die praktische Erfahrung gezeigt, dass Backpropagation bei einer genügend kleinen Schrittweite (Lernfaktor) η in sehr vielen Anwendungen ein Minimum findet, das gut genug am globalen Minimum liegt und für die Anwendung akzeptabel ist [GaReoJ]. Ein weiterer Faktor der das Verfahren in unterschiedlichen (lokalen) Minima konvergieren lässt ist die Initialisierung der Gewichte [Pass04b].

Weitere Verfahren um lokalen Minima zu entkommen und sich dem globalen zu nähern wären z.B. das im Anhang 9 beschriebene Verfahren der genetischen Algorithmen oder „Simulated Annealing“.

5.3.6.3 Flache Plateaus

Ein weiteres Problem von Gradientenverfahren sind flache Plateaus (siehe Abbildung 23 rechts oben). Die Größe der Änderung der Gewichte ist vom Betrag des Gradienten abhängig, daraus ergibt sich entweder eine Verlangsamung der Annäherung zum Optimum für ein Plateau mit geringer Steigung bzw. ein Stillstand für ein vollständig ebenes (Gradient ist der Nullvektor). Problematisch ist in diesem Fall vor allem dass man keinen Unterschied zwischen einem vollständig ebenem Plateau und einem Minimum feststellen kann, in beiden Fällen ist der Gradient der Nullvektor.

Zur Lösung dieses Problems existieren einige einfache Verfahren, z.B. durch Einführung eines Momentum Terms, mit denen modifizierte Varianten des Backpropagation-Verfahrens in der Lage sind ebene Stellen zu überwinden.

5.3.6.4 Oszillationen in steilen Schluchten

Auch steile Schluchten der Fehlerfläche können für das Lernverfahren problematisch sein und dazu führen dass es oszilliert. Dieser Fall kann eintreten wenn die Änderung der Gewichte dazu führt dass auf die gegenüberliegende Seite gesprungen wird. Ist auf dieser Seite der Schlucht der Anstieg genauso groß wird wieder zurückgesprungen (siehe Abbildung 23 links unten). Das Verfahren oszilliert.

Genauso wie im Fall der Flachen Plateaus kann die Einführung eines Momentum Terms für Abhilfe sorgen und die Oszillationen in steilen Schluchten dämpfen oder gar eliminieren.

5.3.6.5 Verlassen guter Minima

In bestimmten Fällen kann es auch dazu kommen dass selbst wenn das globale Minimum bereits fast erreicht wurde nochmals daraus ausgebrochen wird und das Verfahren in einem lokalen suboptimalen Minimum konvergiert. Beispielsweise kann es in engen Tälern der Fehlerfläche dazu kommen dass der Betrag des Gradienten so groß ist dass aus diesem Tal heraus gesprungen wird und danach eine Annäherung Richtung einem lokalen Minimum erfolgt.

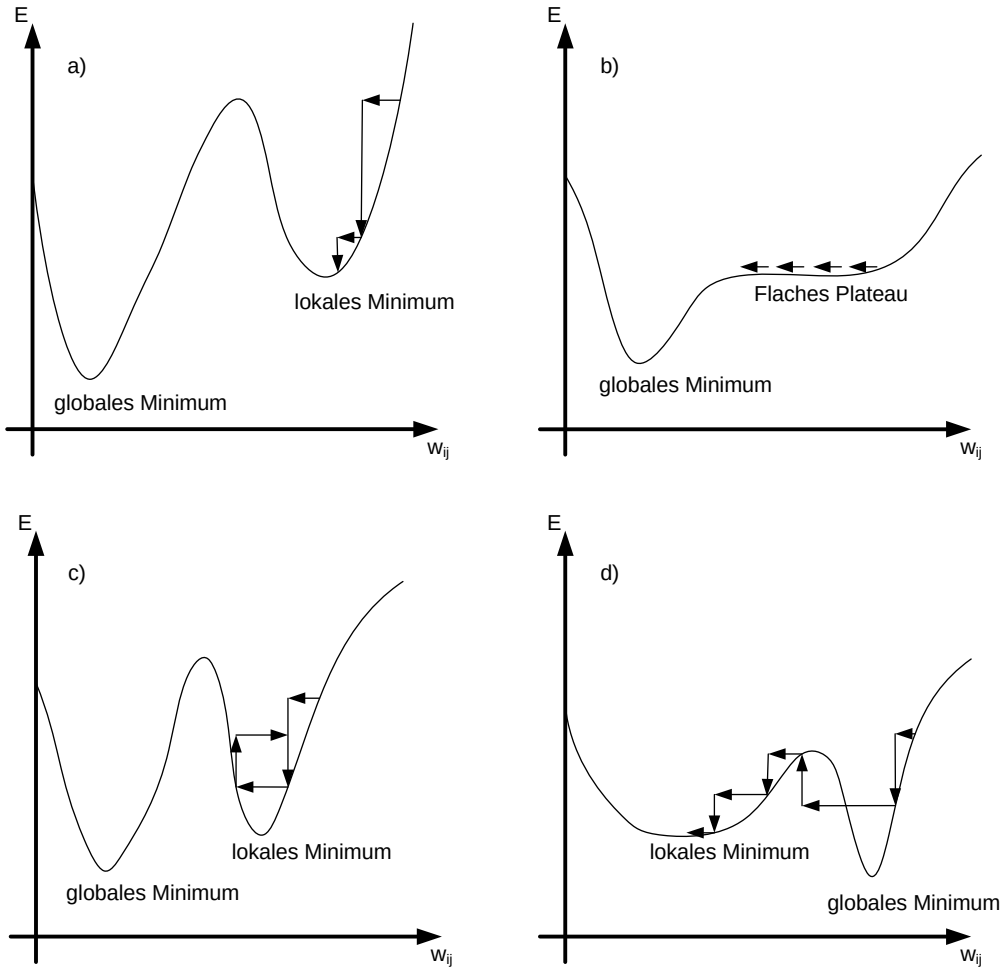

Abbildung 23: Probleme bei Gradientenverfahren [GaReoJ]

Abbildung 23 zeigt nochmals die oben angeführten Probleme in grafischer Form: a) lokales Minimum einer Fehlerfläche, b) Fehlerfläche mit weitem flachen Plateau, c) Oszillationen in steilen Schluchten, d) Verlassen guter Minima.

5.3.7 Alternative Lernalgorithmen

Alternativ zum Backpropagation Algorithmus können auch anderer Verfahren wie z.B. Backpropagation mit Momentum, Quickprop, Resilient Propagation (RPROP) und QRPROP dazu eingesetzte werden die Suchrichtung in der Fehlerfläche bzw. die Schrittweite zu bestimmen.

- Backpropagation mit Momentum: Durch die Berücksichtigung der Gewichtsanpassung der vorherigen Epoche soll das Training auf flachen Plateaus oder in engen Schluchten beschleunigt werden. Zu der aktuellen Änderung der Gewichtung wird die mit einem Momentum gewichtet Änderung der vorhergehenden addiert.

- Quickprop: Durch eine Approximation der Fehlerfläche mittels nach oben geöffneten Parabeln und direkten Schritten zu deren Scheitelpunkten soll eine Beschleunigung des Verfahrens erzielt werden.

Abbildung 24 veranschaulicht nochmals den Quickprop Algorithmus.

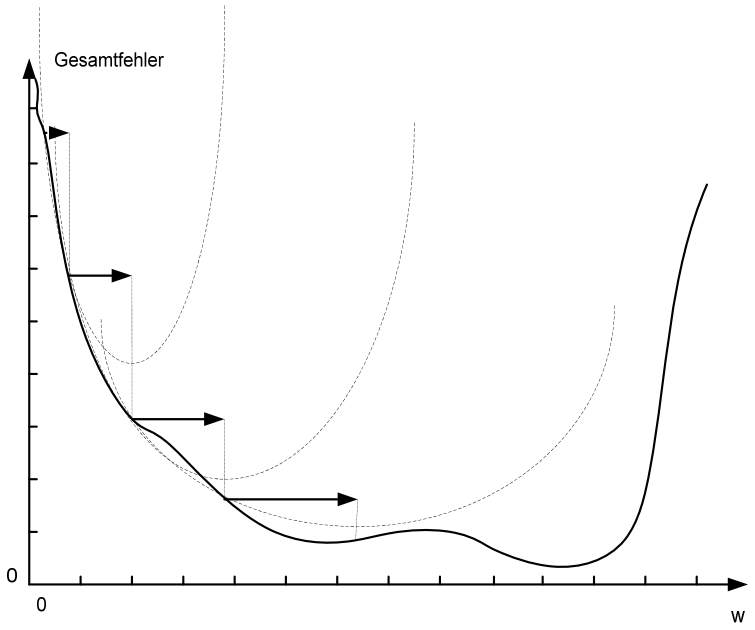

Abbildung 24: Quickprop Verfahren [Pass04b]

- Resilient Propagation (RPROP): Bei diesem Verfahren ist die Änderung der Gewichte nicht mehr von der Größe der partiellen Ableitung abhängig sondern von dessen Vorzeichen. Solange sich das Vorzeichen der partiellen Ableitung nicht ändert (es wurde kein Minimum übersprungen) wird die Schrittweite um einen fixen Faktor vergrößert. Ändert sich das Vorzeichen so erfolgt analog ein Schritt in die umgekehrte Richtung. Schrittweite und Faktor sind unabhängig von der Höhe des Werts der partiellen Ableitung.

  Abbildung 25 stellt das Vorgehen des Resilient Propagation Algorithmus grafisch dar.

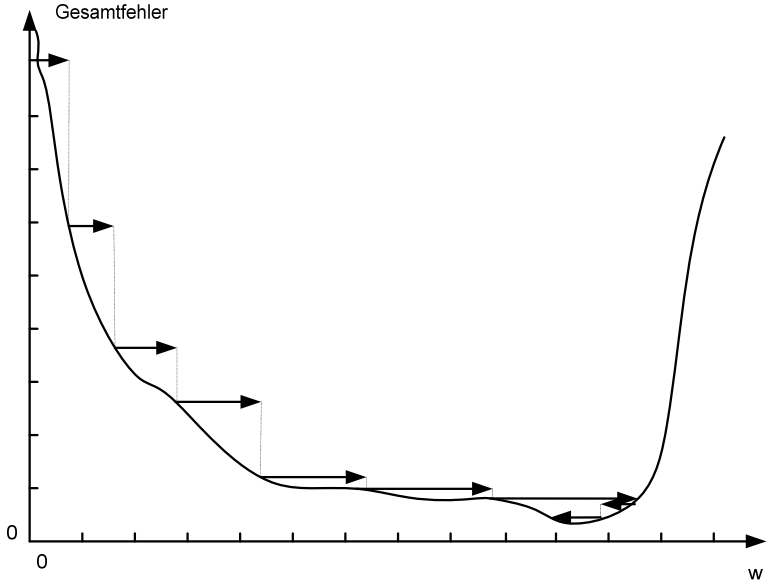

Abbildung 25: Resilient Propagation (RPROP) [Pass04b]

- QRPROP: Dieses Verfahren ist eine Kombination aus Quickprop und RPROP und funktioniert folgendermaßen, dass falls sich das Vorzeichen der partiellen Ableitung in zwei aufeinander folgenden Schritten nicht ändert RPROP zum Einsatz kommt anderenfalls Quickprop.

Keiner der oben angeführten Algorithmen kann das Erreichen eines globalen Minimums garantieren. Meistens ist das erreichen eines hinreichend guten lokalen Minimum für die Anwendung ausreichend [Pass04b].

5.3.8 Typische Anwendungen

Der Einsatzbereich vorwärts gerichteter Netze variiert stark und umfasst unter anderem folgenden Gebiete:

- Klassifizierung und Mustererkennung: Beispielsweise zur Einordnung von Kunden oder in der Qualitätskontrolle.

- Steuerung: Beispielsweise zur Steuerung von Bein-, Hand- oder anderen Prothesen.

- Prognose: Mittels neuronaler Netze und ihrer Eigenschaft zur Generalisierung werden z.B. Prognosen für Börsenwerte oder auch für den Produktabsatz erstellt.

5.4 Partiell rückgekoppelte Netze

Die oben beschriebenen vorwärts verketteten Netze sind in der Lage für eine bestimmte Eingabe stets die gleiche Ausgabe zu erzeugen. Für diese Art von Netzen ist es nicht möglich Kontext Informationen zu berücksichtigen [AnFA04]. Für bestimmte Aufgaben ist es jedoch notwendig eine Art von „Erinnerungsvermögen“ der vorher entstandenen Ausgabe zu besitzen.

Partiell rückgekoppelte Netze erfüllen genau diese Anforderung und beziehen die zuvor generierte Ausgabe in die aktuelle mit ein. Die Ausgabe einer neuen Eingabe ist von der vorhergehenden abhängig. Die beiden nächsten Abschnitte beschreiben Typen von solchen Netzen, das Jordan- und das Elman-Netz. Da diese Art von Netzen im praktischen Teil dieser Arbeit nicht angewandt wird beschränkt sich die Beschreibung auf eine kompakte Übersicht der wesentlichsten Eigenschaften.

5.4.1 Jordan-Netze

Das „Erinnerungsvermögen“ dieser Netze besteht aus genauso vielen speziellen Neuronen, so genannte Kontext-Neuronen, wie Ausgabe Neuronen. Einerseits besitzt jedes Neuron der Ausgabeschicht eine fixe (bezogen auf die Gewichtung von 1), nicht trainierbare Verbindung zum jeweiligen Kontext-Neuron. Andererseits sind Kontext-Neuronen mit den Neuronen der inneren Schicht durch veränderbare/trainierbare Gewichtungen voll verbunden. Weiters kann die Stärke des Erinnerungsvermögens, der Einfluss der vorherigen Eingabe, durch eine direkte Rückkopplung gesteuert werden. Abbildung 26 stellt den Aufbau dieser Netz-Kategorie grafisch dar.

Da die einzigen rückwärts gerichteten Verbindungen im Netz die zwischen Ausgabe- und Kontext-Neuronen ist, deren Gewichte fix mit 1 vorgegeben sind bleiben nur vorwärts gerichtete Verbindungen mit veränderbaren Gewichten. Aus diesem Grund kann der bereits

beim Multi Layer Perzeptron beschriebene Backpropagation-Algorithmus zum Training eingesetzt werden.

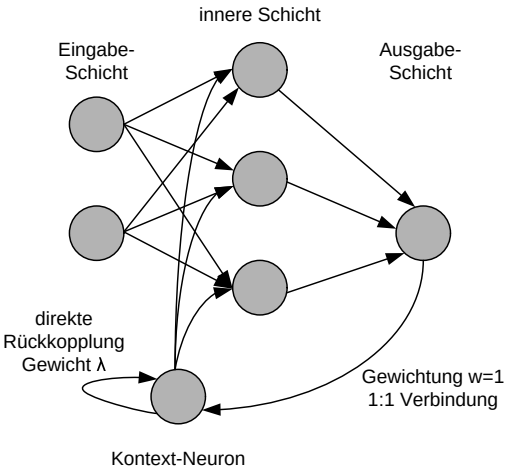

Abbildung 26: Jordan-Netz

5.4.2 Elman-Netze

Jordan-Netze sind wegen der durch die Anzahl der Ausgabe-Neuronen vorgegebenen Anzahl an Kontext-Neuronen in der Rückkopplung eingeschränkt [AnFA04]. Elan-Netze versuchen diese Einschränkung durch eine Erweiterung zu umgehen. Die Rückkopplung wird bei diesem Typus von neuronalen Netz durch die innere Schicht gesteuert, jedem inneren Neuron ist ein Kontext-Neuron zugeordnet (siehe Abbildung 27). Da die Anzahl der Neuronen der inneren Schicht und damit auch die der Kontext-Neuronen flexibler als die der Ausgabeschicht ist (meist fix vorgegeben) gestaltet sich auch die Rückkopplung flexibler [AnFA04].

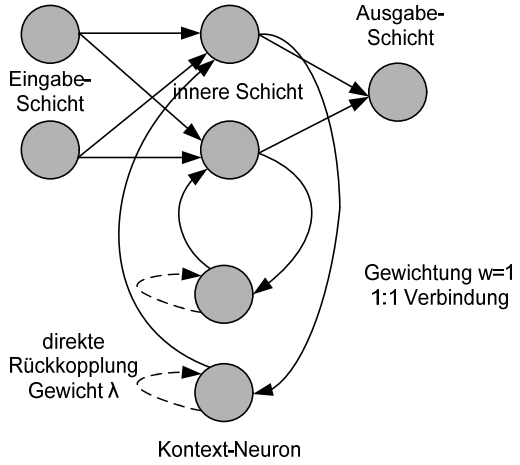

Abbildung 27: Elman-Netz

5.5 Selbst organisierende Karten

Eine weitere Art von künstlichem neuronalen Netz ist die der selbst organisierenden Karten, Self-Organizing Maps, Self-Organizing Feature Maps (SOFM) oder Kohonennetze. Sie wurden erstmals 1982 von Teuvo Kohonen vorgestellt und unterscheiden sich von den zuvor genannten Netzen vor allem in der Art und Weise wie sie lernen. Bei SOFM handelt es sich

um ein künstliches neuronales Netz mit einer Eingabe- und einer Ausgabeschicht, bei dem Wettbewerbs-Lernen (unüberwachtes) angewandt wird [Kreb00].

Ihr Funktionsprinzip ist dadurch motiviert, dass viele Strukturen im Gehirn eine lineare oder planare Topologie aufweisen (gefaltet in eine gewundene Form um in den Schädel zu passen) obwohl Signale, z.B. visuelle Reize, multidimensional sind. Dadurch stellte sich die Frage, wie diese multidimensionalen Eindrücke durch planare Strukturen verarbeitet werden. Untersuchungen zeigten, dass die Eingangssignale so abgebildet werden, dass ähnliche Reize nebeneinander liegen. Beispielsweise liegt die Region für die Hand nahe der Region für den Arm oder den Fingern (siehe auch Abbildung 28). Der Raum der angelegten Reize wird kartiert. In diesem Zusammenhang wird auch von einer topologieerhaltenden Repräsentation gesprochen [Pass04a] welche eine der wesentlichsten Eigenschaften dieser Netze darstellt.

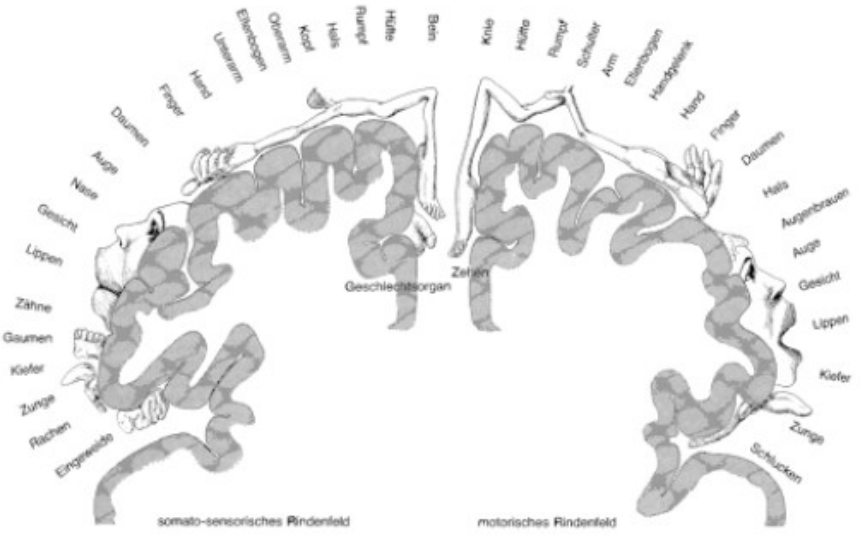

Abbildung 28: Topologieerhaltung im menschlichen Gehirn [Pass04a][Weiß04]

Eine der Hauptanwendungen selbst organisierender Karten ist als Methode zur explorativen Daten Analyse bzw. die Visualisierung von höher dimensionalen Daten. SOFMs können erlernen Datencluster zu erkennen die man in weiterer Folge klassifizieren kann. Eine weitere Anwendung dieser Cluster Technik ist das Erkennen von Neuheiten (bisher nicht aufgetretene Kombinationen an Attributen). Im Gegensatz zur der zuvor genannten Methode wird im Falle von neuen Daten (Daten außerhalb der bisherigen Cluster) die Klassifizierung scheitern was zu dem Schluss führt dass diese Daten neu oder außergewöhnlich sind.

Der Aufbau einer SOFM besteht aus 2 Schichten: der Eingabe- und der Ausgabe(/Topologische)-Schicht (gewöhnlich 2 dimensional). Man kann sich die Ausgabeschicht auch als ein 2 dimensionales Netz vorstellen welches verformt wird um in einem N dimensionalen Raum (gegeben durch die Daten) zu passen und so viel wie möglich der originalen Struktur zu erhalten. Mit der Abbildung von mehrdimensionalen Daten auf eine 2 dimensionale Ebene ist immer ein Detail-Verlust verbunden, trotzdem kann diese Art der Datenanalyse durch Visualisierung nützliche Erkenntnisse liefern [Stat06].

Im Folgenden wird beschrieben wie diese Erkenntnisse durch künstliche neuronale Netze, selbst lernende Karten, im speziellen umgesetzt werden.

5.5.1 Struktur

Die Struktur dieser Netze oder besser gesagt Karten ist von der bisher bekannten des mehrstufigen Perzeptrons verschieden. Sie besteht aus einer Eingabeschicht und einer Kartenschicht (auch Kohonenschicht genannt), in welcher Neuronen zweidimensional und meistens quadratisch angeordnet sind. In manchen Fällen wird auch eine drei- oder mehrdimensionale Anordnung verwendet [Weiß04]. Die Neuronen der beiden Schichten sind vollständig, wie in Abbildung 29 dargestellt, miteinander verbunden, was bedeutet dass jedes Neuron der Eingabeschicht mit jedem der Kartenschicht verbunden ist ohne jedoch irgendwelche Rückkopplungen vorzuweisen. Nur benachbarte Neuronen, welche in nächster Umgebung, können ebenfalls aktiviert und welche die weiter entfernt sind gehemmt werden. Durch das Training der Karte und der daraus resultierenden Anpassung des Netzes kristallisieren sich so genannte Erregungszentren heraus, ein Neuron der Kartenschicht das für alle Eingabe-Muster die einander ähnlich sind aktiv ist. Dieses Neuron kann als Repräsentant einer Klasse von Daten angesehen werden und wird oft als Gewinner-Neuron bezeichnet [AnFA04]. Benachbarte Regionen bzw. Erregungsbereiche entsprechen ähnlichen Signalen. Aus diesem Grund spricht man bei einer solchen Topologie auch von einer topologischen Merkmalskarte der Eingangssignale.

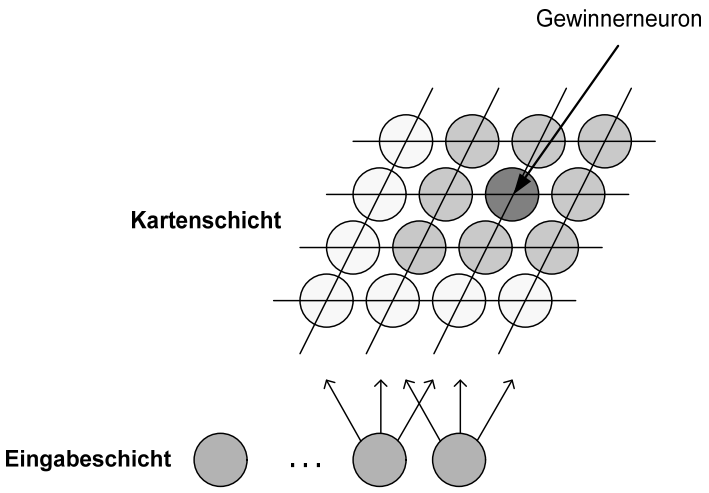

Abbildung 29: Selbst organisierende Karten

Weiters können mehrere selbst organisierende Karten hierarchisch angeordnet werden. In diesem Fall dient der Output einer oder mehrerer Karten als Input für eine Karte der nächsten Stufe.

Anstatt der Anordnung der Ausgabeneuronen als 2 dimensionale Gitter kann auch ein 3 oder 1 dimensionales gewählt werden. In der Kategorie der 2 dimensionalen Anordnungen sind weiters auch hexagonale, dreieckige oder kreisförmige Anordnungen möglich. In seltenen Fällen wird eine 2 dimensionale Karte auf einen dreidimensionalen Raum projiziert, z.B. eine Anordnung auf der Oberfläche einer Kugel [Pass04a].

5.5.2 Lernalgorithmus

Bei Kohonen Netzen wird für jedes Eingabemuster die Ähnlichkeit zu jedem Gewichtsvektor bewertet und ein Siegerneuron bestimmt, welches auch als Prototyp bezeichnet wird [Weiß04]. Andere Bezeichnungen für Gewichtsvektoren die in der Literatur auftauchen sind

Codeword- oder Referenzvektoren. Die Menge aller Codewörter wird als Codebook bezeichnet.

Der Lernalgorithmus der bei dieser Art von künstlichen neuronalen Netzen eingesetzt wird fällt in die Klasse der unüberwachten und beruht auf Wettbewerb. Wettbewerb bedeutet in diesem Zusammenhang dass für ein Eingabe-Muster jeweils nur ein einziges Neuron aktiviert wird („Winner takes all", Siegerneuron) und unüberwacht dass man für ein bestimmtes Eingangs-Muster a priori keine Ausgabe zuweißt sondern das Netz „entscheiden" lässt. Bei jedem Lernschritt werden nicht nur die Gewichte des Siegerneurons verändert, sondern auch die seiner Nachbarn wodurch eine ganze Region auf ähnliche Eingabe-Muster spezialisiert wird. Anders ausgedrückt bildet das Training durch diesen Algorithmus einen Anpassungsprozeß in dem die Neuronen des neuronalen Netzes allmählich empfindlicher für verschiedene Inputkategorien oder Mengen von Datensätzen einer spezifischen Domäne eines Eingaberaums werden [Kask97].

Die Karte tendiert durch das Training dazu gegen eine stationäre Verteilung zu konvergieren welche die Eigenschaften der Wahrscheinlichkeitsdichte reflektiert [Holl00].

Der Algorithmus lässt sich durch folgende Schritte beschreiben:

1. Aus der Menge aller Eingabe-Muster wird zufällig eines ausgewählt und an das Netz angelegt.

2. Ein Siegerneuron *k* wird ermittelt. Sieger ist jenes Neuron dessen Gewichtsvektor *w* dem Eingabevektor *e* am ähnlichsten ist (die Berechnung erfolgt durch die Propagierungsfunktion des Ausgabeneurons). Ähnlichkeit bedeutet in diesem Zusammenhang räumliche Nähe. Diese kann Beispielsweise durch die euklidische Distanz, einem Maß für die Entfernung zwischen zwei Vektoren, zwischen der Eingabe und allen anderen Neuronengewichten ermittelt werden.

$$n_k = \min_i \|w_i - e\| = \min_i \sqrt{\sum_{n=1}^{j} \left(w_{ij} - e_j\right)^2}$$

Falls 2 Gewichtsvektoren die gleiche euklidische Distanz zum Eingabevektor aufweisen wird einer zufällig ausgewählt und als Sieger erklärt.
Eine weiter Möglichkeit zur Bestimmung des Siegerneurons stellt das Skalarprodukt dar [Weiß04] [Pass04a].

3. Anschließend werden die Gewichte aller Neuronen i verändert die sich in einer durch den Radius r gegebenen Nachbarschaft zum Siegerneuron k befinden. Neuronen in der Nachbarschaft müssen aufgrund der gewünschten Erhaltung der Topologie, die benachbarte Anordnung ähnlicher Eingabemuster, Berücksichtigung finden [Weiß04].

Die Modifizierung der Gewichtungen erfolgt durch folgende Regel:

$$w_i \leftarrow w_i + \lambda h(j,i)(e - w_i)$$

$h(j,i)$ … Nachbarschaftsfunktion
λ … Lernrate

Falls der Abstand zwischen den Neuronen i und j größer als der Radius r ist, dann ist $h(j,i) = 0$ und die Gewichte werden nicht verändert. Mit der Lernrate λ kann der Lernvorgang parametrisiert werden.

Das bedeutet, dass nicht diejenigen Neuronen deren Gewichtsvektor dem des Siegerneurons ähneln lernen sondern jene die (auf der Karte) nahe liegen [Pass04a].

Die Nachbarschaft kann abgesehen vom Radius auch über andere Methoden bestimmt werden. Beispiele dafür sind Nachbarschaftsfunktionen wie die Gauß'sche Glockenkurve (Abbildung 30) oder auch die Mexican-Hat-Funktion (Abbildung 31).

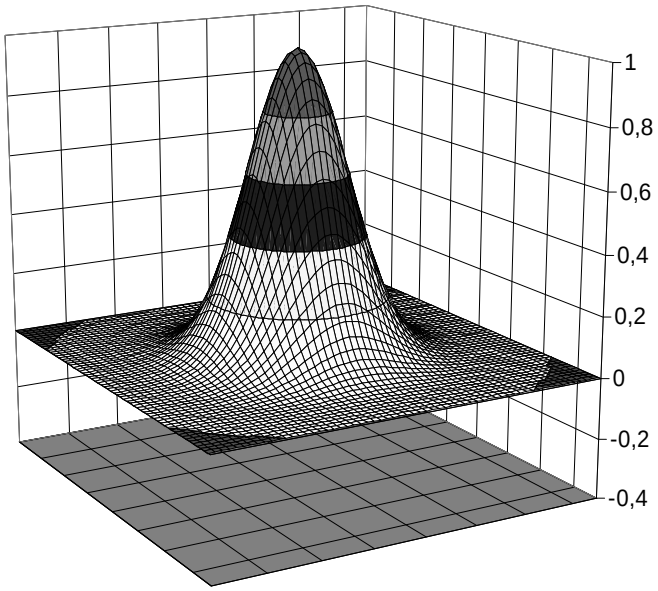

Abbildung 30: Gauß'schen Glockenkurve

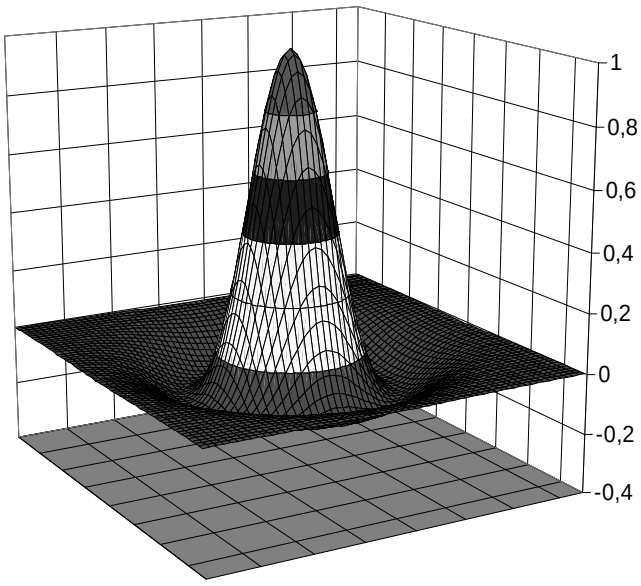

Abbildung 31: Mexican-Hat - Funktion

Hinweis: Die Vermutung das eine Abstoßung von Neuronen deren Gewichtsvektor einen größeren Abstand vom Siegerneuron haben das Training beschleunigt liegt nahe. Jedoch hat sich in der Praxis gezeigt, dass dies zu einer Explosion der Karte führen kann, das Training divergiert. Daher ist bei der Nutzung der Mexican-Hat - Funktion mit teilweise negativen Wertebereichen Vorsicht geboten [Pass04a] bzw. wird in der Praxis oftmals auf diese Abstoßung der Gewichte verzichtet [Weiß04]

4. Falls mehr Lernschritte vorgesehen sind, wird nach einer monoton fallenden Funktion die Lernrate λ und der Radius r (der Distanzfunktion) reduziert und wieder bei Schritt 1 angefangen.

   Als monoton fallende Funktionen kommen z.B. die folgenden in Frage [Pass04a]:
   a. lineare Distanzfunktion
   b. exponentielle Distanzfunktion
   c. Wurzel-Distanzfunktion

Durch den beschriebenen Algorithmus werden nun die Gewichte den Reizen entsprechend angepasst womit der Eingaberaum approximiert wird. Diese Anpassung kann im Online- oder Batch- Verfahren erfolgen. Während beim Online Verfahren die Anpassung jeweils nach jeder einzelnen Eingabe erfolgt wird diese beim Batch Verfahren während der Epoche akkumuliert und danach angewandt.

5.5.3 Qualität des trainierten Netzes

Nachdem das Netz trainiert wurde ist es wichtig zu wissen in wie weit eine Anpassung an die Trainingsdaten erfolgte. In diesem Zusammenhang kann man auch von der Qualität der Karte gesprochen werden, welche gewöhnlich anhand der Kartierungsgenauigkeit und der Topologieerhaltung gemessen wird [HoHa99].

Um die Qualität der Anpassung verschiedener Karten zu vergleichen kann beispielsweise der Quantisierungsfehler auf Basis der Trainingsdaten herangezogen werden. Soll hingegen die Qualität der Generalisierungsfähigkeit verglichen bzw. ermittelt werden muss der Fehler auf Basis neuer Testmuster (nicht zum Training der Karte verwendete) berechnet werden [Pass04a].

5.5.3.1 Kartierungsgenauigkeit

Die Kartierungsgenauigkeit gibt an wie „genau“ die Neuronen auf Inputdaten reagieren, beispielsweise wäre im Fall, dass der ermittelte Referenzvektor des Siegerneurons exakt dem angelegten Inputvektor entspricht der Fehler der Genauigkeit gleich 0. Im Normalfall wird dieser Fall allerdings nicht eintreten da die Anzahl der Inputvektoren die der Neuronen übersteigt. Aus diesem Grund ist der Fehler immer ungleich 0 [HoHa99].

Ein gebräuchliches Maß, das die Genauigkeit der Karte misst ist der durchschnittliche Quantisierungsfehler über die gesamte Menge an Inputvektoren.

$$\varepsilon_q = \frac{1}{N}\sum_{i=1}^{N}\left\|e_i - w_c\right\|$$

e_i … Eingabevektor

w_c … Referenzvektor des Siegerneurons

Während der Quantisierungsfehler eine Aussage über die Güte der Karte trifft kann durch den Abstand zum Siegerneuron auch eine über die Abbildung der Ähnlichkeitsbeziehungen (der Topologie) getroffen werden [Pass04a].

$$\varepsilon_{winnerdist} = \frac{1}{L}\sum_{k=1}^{L}\left\|h_j(k) - h_i(k)\right\|$$

$h_j(k)$ … Koordinatenvektor des Siegerneurons j für die Eingabe k

$h_i(k)$ … Koordinatenvektor des Ausgabeneurons i dessen Gewichtsvektor der Eingabe k am zweitähnlichsten ist.

L … Anzahl der Eingabemuster

5.5.3.2 Topologieerhaltung

Die Topologieerhaltung gibt an wie gut die selbst organisierende Karte die Topologie der Inputdaten erhält. Im Gegensatz zur Kartierungsgenauigkeit betrachtet diese die Struktur der Karte. Für ein gefaltetes Netz ist der topografische Fehler, auch wenn die Kartierungsgenauigkeit gut ist, hoch [HoHa99].

Um den topographischen Fehler der Karte zu erfassen kann folgende Methode angewandt werden:

$$E_t = \frac{1}{N}\sum_{k=1}^{N} u(e_k)$$

$$u(e_k) = \begin{cases} 1 : \textit{falls das Siegerneuron und das zweitbeste nebeneinander liegen} \\ 0 \end{cases}$$

5.5.4 Visualisierung

Eine selbst organisierende Karte kann auf verschiedene Arten visualisiert werden. Zum Beispiel können bei einer zweidimensionalen Kartenschicht die Neuronen je nach Häufigkeit der Aktivierung verschieden eingefärbt werden um Erregungszentren hervorzuheben. Die Wahl der jeweiligen Methode ist von den zu analysierenden Eigenschaften abhängig. In Fällen in denen Strukturen erkannt werden sollen reicht die Aussage über die Anzahl der Referenzvektoren die zu einem Neuron „gehören“ (Daten Histogramm) nicht aus, weil die hochdimensionalen Distanzen verzerrt dargestellt werden. Um diesen Abstand zwischen den Daten Clustern besser bzw. exakter darzustellen wird auf die U-Matrix zurückgegriffen.

Über die Jahre haben sich durch verschiedene Erfordernisse auch mehrere Darstellungen der Karte entwickelt von denen aber keine als den anderen als überlegen bezeichnet werden kann, jede Methode hat ihre Vor- und Nachteile [HoHa99].

Im Folgenden wird auf die U-Matrix, das Daten Histogramm und die Darstellung der Approximation des Eingaberaums einer Karte mit 2 Eingabeneuronen genauer eingegangen.

5.5.4.1 U-Matrix

Die U-Matrix (unified distance matrix) stellt den Abstand zwischen den Neuronen dar und ist vielleicht die populärste Methode zur Visualisierung [HoHa99]. Der Abstand zwischen den Referenzvektoren benachbarter Neuronen wird berechnet und danach z.B. in verschiedenen Farben oder bei einer 3-dimensionalen Darstellung durch Täler und Höhen dargestellt. Beispielsweise kann dabei eine helle Einfärbung auf die Nähe von Clustern deuten während eine dunklere auf einen größeren Abstand hinweist.

Die Visualisierung der angepassten Karte durch die U-Matrix bietet einen einfachen Weg um einen Einblick in die Verteilung der Daten zu bekommen [Holl96]

5.5.4.2 Daten Histogramm

Ein Daten Histogramm zeigt wie Eingabevektoren durch die Karte verschiedenen Clustern zugeordnet werden, die Häufigkeitsverteilung. Anders gesagt wie viele der Eingabevektoren zu einem Cluster, definiert durch Neuronen, zugeordnet wurden. Das Histogramm kann durch Eingabe eines Daten Sets in ein bereits trainiertes Netz erstellt werden. Die Anzahl der „hits“ kann danach auf verschiedenste Art und Weise dargestellt werden. Als Beispiele können eine Einfärbung der Neuronen (z.B. durch Graustufen) oder die Höhe, je nach Anzahl der Treffer, von Balken pro Neuron angeführt werden.

Ein Beispiel dafür, ein Histogramm einer Karte mit 12x8 Neuronen, ist in Abbildung 32 dargestellt.

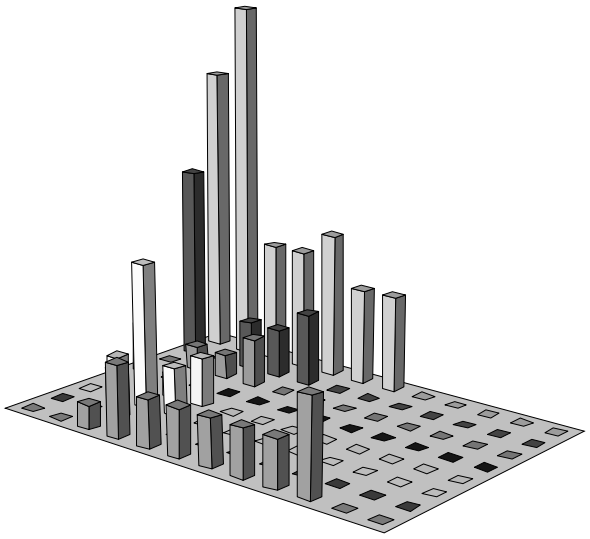

Abbildung 32: 3D Histogramm einer 12x8 Karte

5.5.4.3 Visualisierung der Gewichte

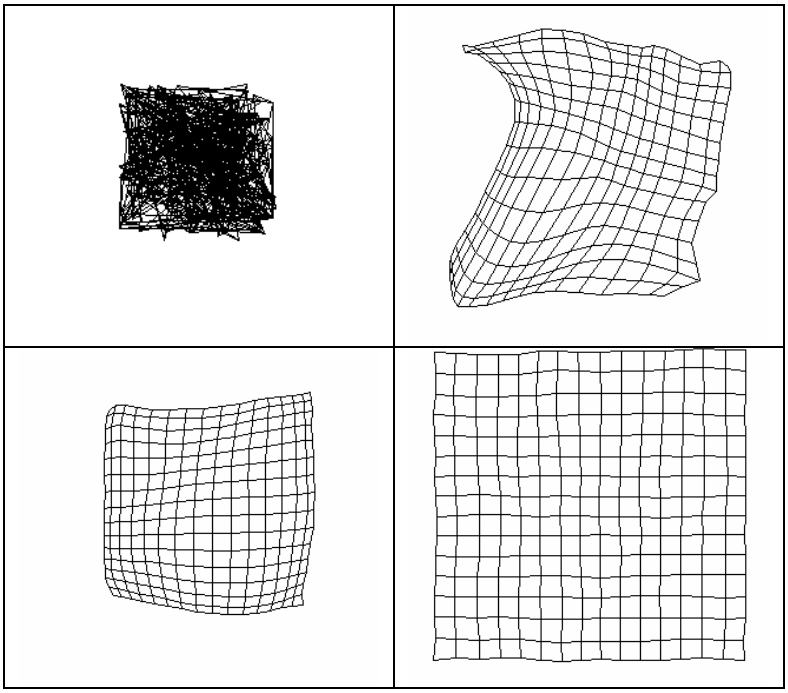

Abbildung 33: Approximation des Eingaberaums bei fortschreitendem Lernprozess

Abbildung 33 zeigt vier Momentaufnahmen eines selbst organisierenden Netzes mit 2 Eingabeneuronen und einer Ausgabeschicht von 16x16 Kartenneuronen während der Lernphase. Die Eingabemuster entsprechen auf Basis einer Gleichverteilung zufällig gewählten Koordinaten innerhalb eines Einheitsquadrats. Dabei entspricht die X-Achse dem Gewicht des jeweiligen Kartenneurons zum Eingabeneuron 1 und die Y-Achse zum Eingabeneuron 2. Das zuerst zusammengefaltete Netzwerk breitet sich mit voranschreitendem Training über den Eingaberaum aus.

Die Approximation findet durch die in Kapitel 5.5.2 genannten Lernschritte statt:

1. Aus einem 2-dimensionalen Eingaberaum wird ein Punkt innerhalb des Quadrats ausgewählt.

2. Das Neuron, dessen ebenfalls 2-dimensionaler Gewichtsvektor dem Eingabevektor am nächsten ist, ist der Gewinner.

3. Seine Gewichte werden so angepasst dass die Distanz zwischen den beiden Vektoren noch verkleinert wird. Da das Gewinner-Neuron auch die Gewichte seiner Nachbarn verändern kann, wird eine ganze Region an den Eingabepunkt herangezogen. Dadurch breitet sich das Netzwerk über den kompletten Eingaberaum aus.

5.5.5 Mögliche Probleme

Während gleichverteilte Daten ein gleichmäßiges Netz ohne Defekte (Abbildung 34) erwarten lassen ist bei Daten die sich in einem Bereich des Eingaberaums häufen eine entsprechend höhere Auflösung der Karte in einer Region der trainierten Karte zu erwarten (Abbildung 35).

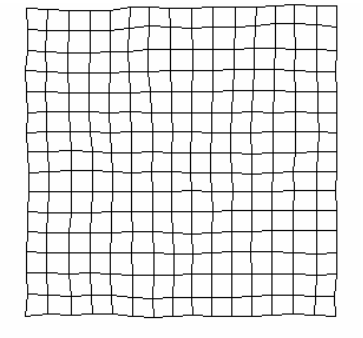

Abbildung 34: Karte ohne Defekte

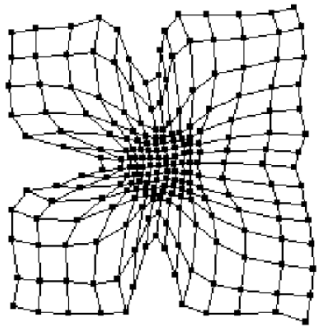

Abbildung 35: Karte mit einem Bereich höherer Auflösung [Pass04a]

Bei ungünstiger Wahl der Parameter die das Netz bestimmen (z.B. Distanzfunktion oder Lernratenfunktion) kann es zu verschiedenen Problemen kommen.

Dabei können unter anderen die beiden folgenden genannt werden:

- Topologischer Defekt: Werden Distanz- und Lernratenfunktion schlecht gewählt kann es zu einem so genannten topologischen Defekt kommen bei dem sich die Karte nicht vollständig entfalten kann (obwohl mit gleichverteilte Daten trainiert wurde). Nimmt die Lernrate zu schnell ab werden die jeweiligen Referenzvektoren zu schnell fixiert was in Folge in manchen Fällen zu diesem Problem führen kann.

  Abbildung 36 zeigt ein Beispiel für eine Karte mit topologischem Defekt.

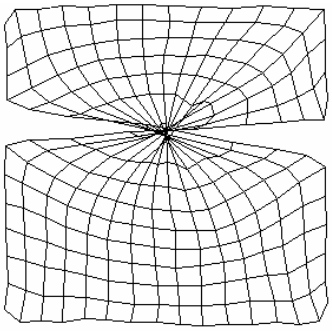

Abbildung 36: Karte mit topologischem Defekt

- Randeffekt: Dieser Effekt bezieht sich auf eine höhere Auflösung des Netzes am Rand der trainierten Karte. Der Grund dafür ist, dass Neuronen am Rand weniger Nachbarn aufweisen und damit einem kleineren Einfluss der Nachbarschaftsfunktion unterliegen [Pass04a]. Die Folge davon ist dass diese Neuronen nicht stark genug nach außen gezogen werden.

  Abbildung 37 zeigt ein Beispiel dieses Effekts.

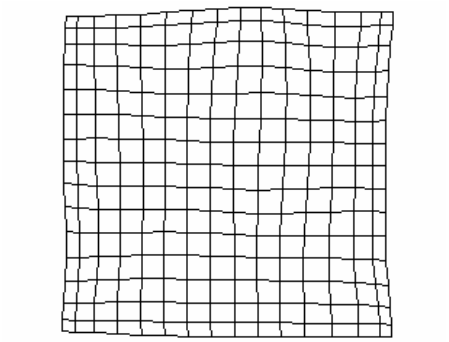

Abbildung 37: Karte mit Randeffekt

5.5.6 Netzgröße

Einerseits ist die Anzahl der Neuronen auf der Input Seite durch die Dimension der Eingabedaten vorgegeben, andererseits ist die Größe der Karte von den gewünschten Eigenschaften abhängig. Eine höhere Anzahl von Neuronen in der Kartenschicht führt zu einer höheren Auflösung während eine Glättung (der Zuordnung Eingabevektor/Karten-Neuron) unabhängig von der Anzahl durch den Radius in der Nachbarschaftsfunktion, der die Menge der aktivierten Neuronen bestimmt, gesteuert werden kann [Kask97]. Weiters spielt die gewünschte „Ausdünnung" der Daten bei dieser Aufgabe eine Rolle. Sollen die Daten vor allem generalisiert werden so sollte die Anzahl entsprechend der Anzahl der erwartenden Cluster gewählt werden. Steht im Gegensatz dazu aber die Approximation der Eingabevektoren im Vordergrund sollten wesentlich mehr Ausgabeneuronen als Eingabevektoren für das Training zur Verfügung stehen dimensioniert werden. Bei einer im Vergleich geringen Anzahl an Ausgabeneuronen werden mehrere Eingabevektoren durch einen Referenzvektor approximiert [Kreb00].

Die „Growing Cell Structure", eine Art von neuronalen Netz die der selbst organisierenden Karte verwandt ist, umgeht das Problem des Auffindens der optimalen Kartengröße indem sie durch selbständiges hinzufügen und entfernen von Neuronen während des Trainings die Struktur anpasst [Weiß04].

5.5.7 Typische Anwendungen

Selbst organisierende Karten können in den verschiedensten Anwendungen in denen große Mengen an Daten klassifiziert werden sollen zum Einsatz kommen.

Im Folgenden werden einige typische Anwendungen aufgezählt:

- Rundreiseproblem [Weiß04][Pass04a]: Zwischen n Städten soll eine möglichst kurze Route gefunden werden wobei keine Stadt mehrmals besucht werden darf.
- Mustererkennung: z.B. Erkennen von Buchstaben
- Sprachanalyse
- Klassifikation von Daten in der Medizin oder der Industrie
- Datenkompression

- Dimensionsreduktion komplexer Daten
- Identifizierung von Clustern und die Zuordnung neuer unbekannter Eingaben

6 Profile Monitor – Prototyp

In diesem Abschnitt der Arbeit wird ein Prototyp entwickelt der auf Basis der Differential-Analyse (einer Profil Methode) Anomalien im Nutzungsverhalten aufzeigen soll.

Beginnend mit einer Beschreibung der Anforderungen, einer ersten Architektur und der Analyse der wesentlichen Komponenten des Systems wird danach zu Design und Implementierung übergegangen. Abschließend wird das Ansprechverhalten des Systems an Hand einiger Testfälle analysiert und das Ergebnis diskutiert.

6.1 Beschreibung & Anforderungen

6.1.1 Beschreibung

Das zu erstellende System soll auf Kundenbasis eine Identifizierung bzw. Signalisierung von signifikanten Veränderungen des Verhaltens beim Gebrauch von Telekommunikations-Services ermöglichen. Um dies zu erreichen wird der differentiale Ansatz der Benutzerprofilmethode gewählt. Obwohl man diese Art von Analyse auch auf vielen anderen Gebieten anwenden kann (z.B. im Kreditkartenbereich) wird bei dieser Entwicklung besonderes Augenmerk auf den Einsatz im Mobilfunk zur Aufdeckung von Missbrauchsfällen gelegt. Dabei dienen Daten von so genannten „mobile originated calls“, die auch zur Verrechnung herangezogen werden und vom jeweiligen Netzwerkelement bereitgestellt werden, als Input. Output ist der gegebenenfalls generierte Alarm.

Die Methode des differentialen Ansatzes der Benutzerprofilmethode wurde im Wesentlichen aus den folgenden Gründen gewählt:

- Ein Missbrauchsfall geht meist mit einer Änderung im Benutzerverhalten einher.

- Diese Methode basiert nicht auf dem Nachvollziehen von technischen Gegebenheiten. Aus diesem Grund können auch Missbrauchsfälle die auf bisher unbekannte Techniken beruhen erkannt werden.

- Ein künstliches neuronales Netz wie die selbst organisierende Karte benötigt a priori keine als missbräuchlich gekennzeichneten Datensätze zum Training. Zur Strukturierung der Daten (dem erstellen von Prototypen) sind Daten aus dem Normalbetrieb des Funknetzes ausreichend.

- Eine absolute Methode würde nicht zwischen den verschiedenen Kategorien der Nutzung unterscheiden. Beispielsweise ist ein einzelner Schwellwert für die Anzahl der Gespräche nicht für alle Kunden anwendbar. Gerade die „besten“ Kunden telefonieren oft und lange und sollten durch häufige Fehlalarme auf Grund eines zu niedrigen Schwellwertes (der aber möglicherweise für 80 % aller anderen Kunden zutreffend ist) nicht verärgert werden. Eine Methode die für jeden Benutzer ein eigenes Profil konstruiert und aktuell hält um dieses mit einem historischen zu vergleichen ermöglicht eine Betrugserkennung auf persönlicher Ebene [BSCM97].

6.1.2 Anforderungen

Folgende Anforderungen soll das System zur Erkennung von Änderungen im Nutzungsverhalten erfüllen.

- Die Applikation soll plattformunabhängig sein.
- Eine Kommando Zeilen Applikation (kein GUI) ist ausreichend.
- Das System soll einen vereinfachten, erweiterbaren Workflow darstellen.
- Die Konfiguration dieses Workflows soll über eine XML Datei erfolgen.
- Als Input für das System sollen die gleichen Gesprächsdaten die auch einem Verrechnungssystem zur Verfügung stehen herangezogen werden.
- Gesprächsdaten können entweder aus einer Datei (z.B. „comma separated values“) oder aus einer Datenbank gelesen werden.
- Der erste Prototyp des Systems soll auf einen einmaligen Durchlauf einer gegebenen Menge an Daten ausgelegt sein (in späteren Phasen wäre eine Event gesteuerte Eingangsstufe denkbar die einen dauerhaften Serverbetrieb erlaubt).
- Das Training des künstlichen neuronalen Teils soll gesondert erfolgen und das Ergebnis dauerhaft gespeichert werden.
- Entdeckte Änderungen im Nutzungsverhalten sollen per Email alarmiert werden und/oder in einer Log Datei festgehalten werden.
- Der Prototyp der Applikation soll dazu in der Lage sein Benutzerprofile von 100.000 Kunden zu bearbeiten.

6.2 Analyse

Nach Auswertung der Anforderungen konnten drei Haupt-Usecases identifiziert werden. „ANN Training“, „Fraud Detection“ und „Alarming“ auf die im nächsten Abschnitt näher eingegangen wird. Anforderungen wie z.B. die Konfiguration des Systems durch eine XML Datei werden nicht im Detail beschrieben da sie keine wesentlichen Benutzeranforderungen darstellen.

Neben den bereits genannten Anforderungen an die Funktionalität der Software ist das Hauptaugenmerk vor allem auf den Workflow und die Differentialanalyse zu legen.

Als Workflow ist eine einfache Abfolge von Aktivitäten, eine Sequenz, bei der jede Aktivität (Komponente) gesondert konfiguriert werden kann ausreichend. Jede der Komponenten ist grundsätzlich dazu in der Lage einen Alarm zu generieren. Eine Möglichkeit diese Sequenz abzubilden bietet das Spring Framework [Spri06], ein auf JAVA basierendes Framework. Weiters erhält man durch den Einsatz dieser API auch einige nützliche Hilfsklassen, die z.B. die Kommunikation mit einer Datenbank erleichtern, mitgeliefert.

Die Berechnung und Aktualisierung der Benutzerprofile basiert auf einer einführenden Phase in der eine selbst organisierende Karte trainiert wird. Gefundene Strukturen können dabei als Prototypen angesehen werden welche Kategorien bzw. Cluster in den Daten entsprechen. Genauer gesagt ergeben sich diese Prototypen/Kategorien durch den zu einem Outputneuron gehörenden Gewichtsvektor. Ein JAVA basierendes Framework das den Aspekt der künstlichen neuronalen Netze abdeckt ist Joone (Java Object Oriented Neural Engine) [MaJo06]. Neben vorwärts verketteten Netzen können damit auch selbst organisierende Karten implementiert werden.

Zur Alarmierung und als Log-Subsystem kommt Log4j [ASF06] des Apache Jakarta Projekts zum Einsatz. Dieses System bietet neben der Funktionalität zur Erstellung von Log-Nachrichten (bzw. Alarme) in einer Datei auch die Möglichkeit diese per Email oder SNMP (Simple Network Management Protocol) zu versenden.

In den nächsten Abschnitten wird nach der näheren Beschreibung der drei Haupt-Usecases auf den Workflow, die Differentialanalyse (Erstellung der Benutzerprofile), die den wesentlichen Aspekt dieses Projekts darstellt, die Inputdaten und die Art der Alarmierung im Detail eingegangen.

6.2.1 Usecase Diagramm

Während der Anforderungsanalyse konnten wie bereits erwähnt die drei in Abbildung 38 dargestellten Haupt-Usecases identifiziert werden. Jeder dieser Usecases wird im Folgenden in einem gesonderten Unterabschnitt beschrieben.

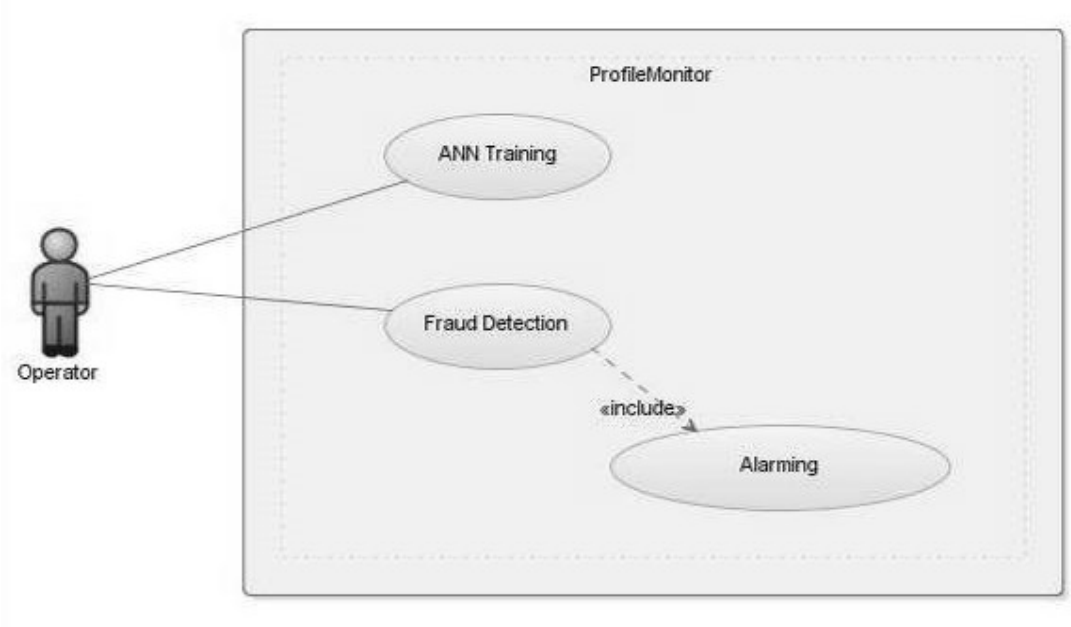

Abbildung 38: ProfileMonitor – Usecase Diagramm

6.2.1.1 Usecase - ANN Training

Usecase Name: ANN Training
Akteure: Operator
Vorbedingungen:

- Trainingsdaten mit MOC (Mobile Originated Call) Tickets stehen in einer CSV (Comma Separated Values) Datei oder in einer Datenbank zur Verfügung.

- Die Konfigurationsdatei des Workflows die unter anderem auch die Parameter der selbst organisierenden Karte enthält ist erstellt.

Nachbedingungen:

Auslöser:

- Der Operator möchte Prototypen (Kategorien bzw. Cluster in den Daten) ermitteln.

Ablaufbeschreibung:

1. Datenquelle öffnen.
2. Datensatz einlesen.
3. Anpassen der Gewichte des künstlichen neuronales Netzes.
4. (weiter mit Schritt 2.) solange bis alle Datensätze eingelesen wurden.
5. Ergebnis (die Gewichtsvektoren bzw. die Prototypen) speichern.

Fehlsituationen:

- Keine Inputdaten bzw. die Datenquelle existiert nicht.
- Fehlerhafte Datensätze.
- Die Konfigurationsdatei für den Workflow existiert nicht.
- Die Komponente „Feature Extraction“ ist fehlerhaft.
- Das Ergebnis kann nicht abgespeichert werden.

Instanzen:

- Ein Operator möchte aus den Verrechnungsdaten die ein Netzwerkelement generiert neue Prototypen erstellen welche die aktuelle Situation widerspiegeln. Diese sollen danach in der Fraud Detection Applikation Profile Monitor zur Generierung von Benutzerprofilen verwendet werden. Dazu konfiguriert er den Workflow und gibt im Zuge dessen Parameter wie das Persistenzverzeichnis in dem das Ergebnis gespeichert wird oder die Anzahl der gewünschten Prototypen an und startet danach die Applikation mit dem Parametern *Datenquelle* und *-t*.

Ergebnisse:

- Die Prototypen werden im konfigurierten Verzeichnis in einer Datei abgelegt.

Autor: Geith Alois

Ursprung: Diplomarbeit am Institut für Informationswirtschaft

6.2.1.2 Usecase – Fraud Detection

Usecase Name: Fraud Detection

Akteure: Operator

Vorbedingungen:

- Das künstliche neuronale Netz wurde trainiert und es liegen abgespeicherte Prototypen vor.
- Die Konfigurationsdatei des Workflows ist erstellt.
- Der Schwellwert für den Abstand zwischen aktuellem und historischem Profil wurde gewählt.
- Daten zur Analyse liegen vor.

Nachbedingungen:

Auslöser:

- Der Operator möchte eine Menge von Gesprächsdatensätzen analysieren bzw. auf signifikante Änderungen im Gesprächsverhalten untersuchen.

Ablaufbeschreibung:

1. Datenquelle öffnen.
2. Datensatz einlesen.
3. Durchlaufen der „Feature Extraction“ Komponente.
4. Anpassen des aktuellen und des historischen Profils.
5. Bewertung des Abstands zwischen den beiden Profilen.

6. Generieren eines Alarms falls der Schwellwert für den Abstand überschritten wurde.

Fehlsituationen:

- Keine Inputdaten bzw. die Datenquelle existiert nicht.
- Fehlerhafte Datensätze.
- Die Konfigurationsdatei für den Workflow existiert nicht.
- Die Komponente „Feature Extraction“ ist fehlerhaft.
- Der Alarm kann nicht generiert werden.
  - Kein Netz bei einer Alarmierung per Email oder SNMP
  - Zu wenig Harddiskspeicher bei einer Speicherung in Dateien

Instanzen:

- Ein Operator möchte die Verrechnungsdaten der vorangegangenen Periode auf signifikante Änderungen im Gesprächsverhalten einzelner Benutzer (bzw. auf Missbrauchsfälle) untersuchen. Prototypen welche die Struktur der Daten im Netz widerspiegeln wurden durch ihn bereits am Tag davor generiert. Nun konfiguriert er den Schwellwert für den Abstand zwischen aktuellem und historischem Profil durch dessen überschreiten ein Alarm ausgelöst und über alle angegebenen Kanäle versendet werden soll. Nach erfolgter Konfiguration stellt er eine CSV Datei mit Inputdaten zur Verfügung und startet die Applikation.

Ergebnisse:

- Alarme werden gegebenenfalls generiert.

Autor: Geith Alois

Ursprung: Diplomarbeit am Institut für Informationswirtschaft

6.2.1.3 Usecase - Alarming

Usecase Name: Alarming

Akteure:

Vorbedingungen:

- Eine signifikante Änderung im Nutzungsverhalten eines Kunden wurde festgestellt.

Nachbedingungen:

Auslöser:

- Eine signifikante Änderung im Nutzungsverhalten eines Kunden wurde festgestellt.

Ablaufbeschreibung:

1. Übernahme des Alarms.
2. Überprüfen ob die Profile des Benutzers bereits auf genügend Datensätzen beruhen.
3. Falls ja
   1. Verteilen des Alarms an alle Alarmkanäle.
4. Falls nicht
   1. Erstellen einer Alarmnachricht mit niedriger Priorität in einer Log-Datei.

Fehlsituationen:

- Der Alarm kann nicht ausgelöst werden.

Instanzen:

- Die Alarm Komponente erhält eine Nachricht die versendet werden soll. Sowohl das aktuelle als auch das historische Profil des Kunden für den dieser Alarm ausgelöst wurde beruhen bereits auf einer hohen Anzahl von Gesprächsdatensätzen. Aus diesem Grund wird ein Alarm an alle konfigurierten Kanäle, im speziellen Fall Email und Datei, gesendet.

Ergebnisse:

- Ein Alarm wurde bewertet und verteilt.

Autor: Geith Alois

Ursprung: Diplomarbeit am Institut für Informationswirtschaft

6.2.2 Workflow-Architektur

Die Modellierung eines Workflows ist ein Thema an dem bereits seit vielen Jahren gearbeitet wird. Im Dokument „Workflow Patterns“ [AHKB03] wird dazu eine Menge an Design-Patterns klassifiziert durch dessen Hilfe alltägliche Workflow Szenarien modelliert werden können. Der einfachste unter diesen Mustern ist die Sequenz. Sie besteht aus einer Menge von Aktivitäten die sequentiell ausgeführt werden, eine Aktivität im Prozess wird erst nach Abschluss einer anderen gestartet.

Umgesetzt auf die Applikation Profile-Monitor, für die der Einsatz dieses trivialen Musters ausreichend ist, kann der Ablauf des Systems in UML (Unified Modelling Language) [OMG06] Notation als Kollaborationsdiagramm, wie in Abbildung 39, dargestellt werden.

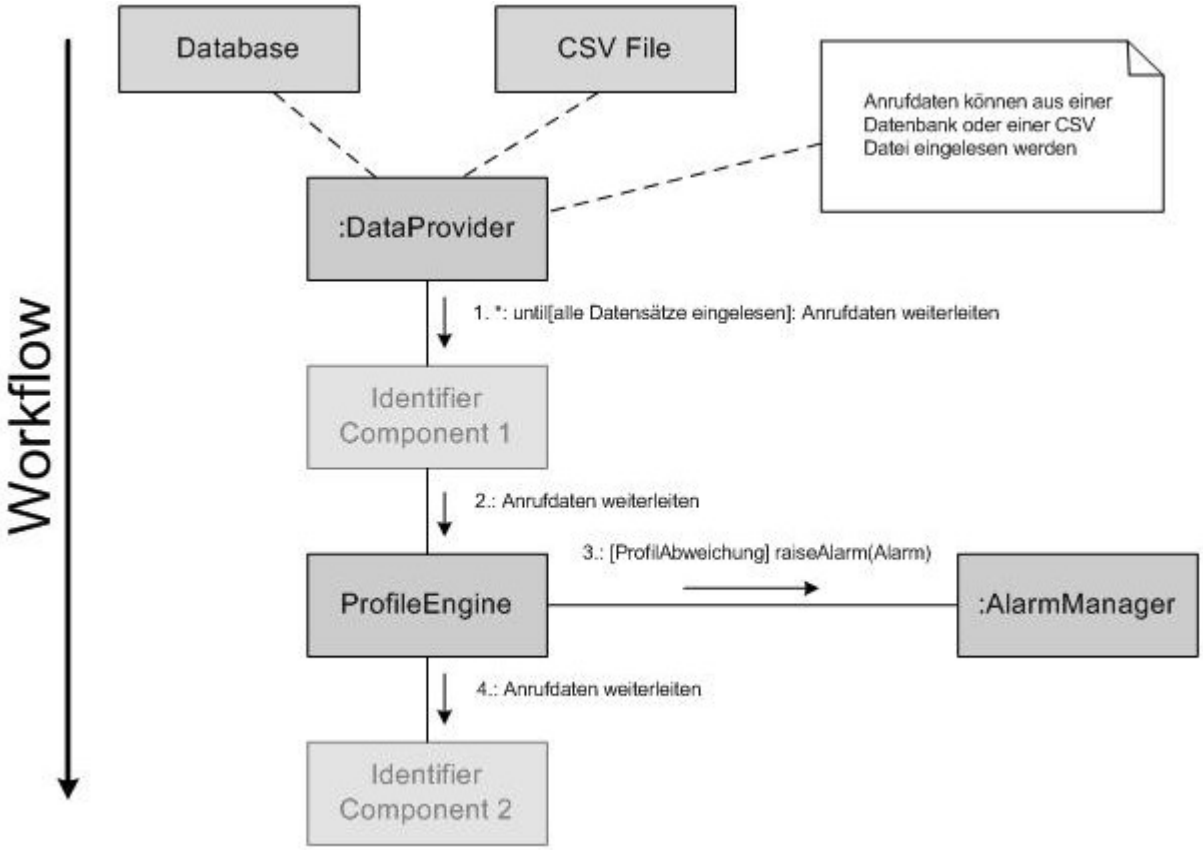

Abbildung 39: Schematischer Workflow Profile-Monitor

Der Prozess (die Sequenz) setzt sich dabei aus den folgenden Komponenten zusammen:

1. DataProvider: Gesprächsdatensätze werden aus einer Datenquelle gelesen, in einem Objekt verpackt und an die folgende Komponente weitergegeben.

2. ProfileEngine: Diese Komponente stellt ebenfalls eine Sequenz von Komponenten dar und hat die Verantwortung des eigentlichen Profiling (unter anderem die Erstellung der Benutzerprofile). Als Input dienen Objekte wie z.B. Gesprächsdatensätze die übergeben wurden.

In Abbildung 40 ist der Ablauf innerhalb der Komponente ProfileEngine als Aktivitätsdiagramm dargestellt:

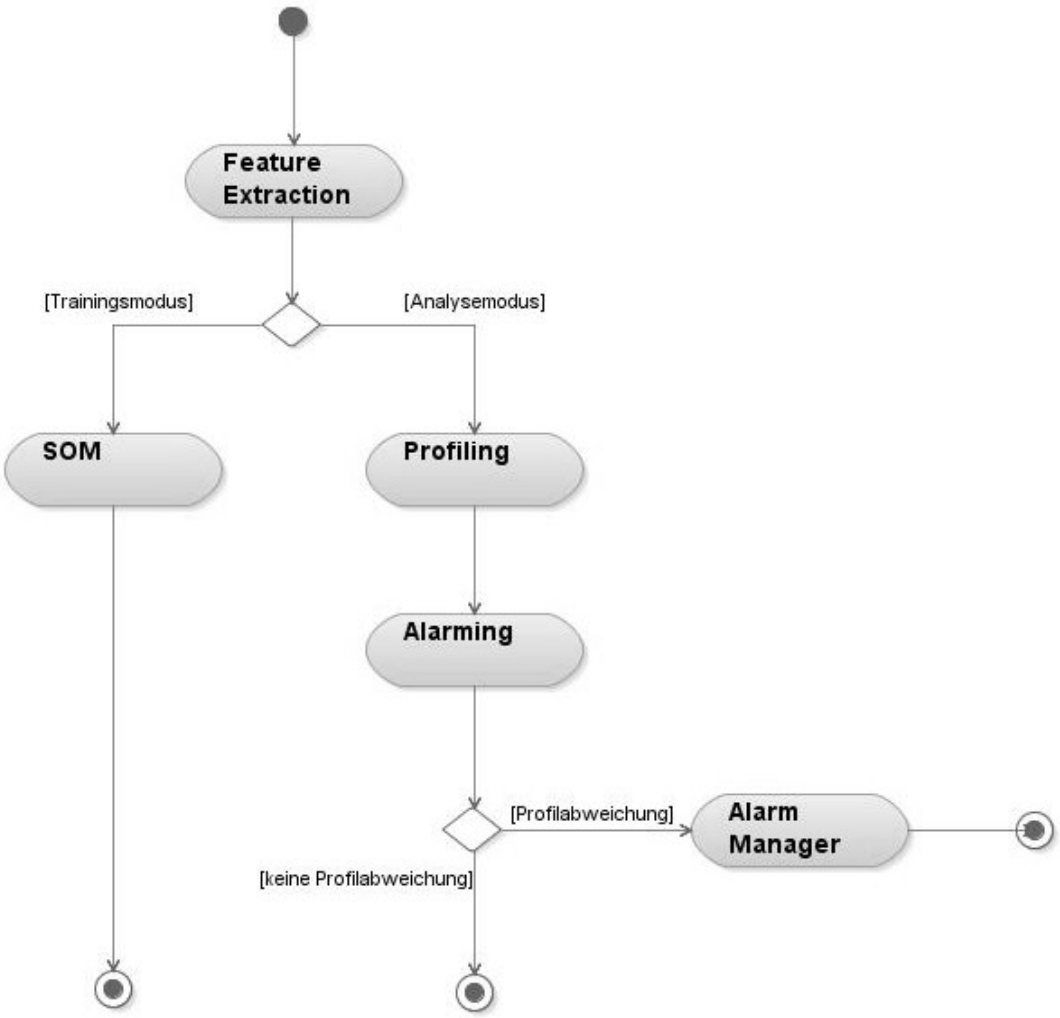

Abbildung 40: Aktivitätsdiagramm der ProfileEngine

Aufgaben der einzelnen Aktivitäten:

- Feature Extraction: Erweitert die Gesprächsdaten um zusätzliche errechnete Informationen, wie beispielsweise die durchschnittliche Anzahl an Ferngesprächen innerhalb der jeweils letzten 20 Anrufe.

- SOM (Self Organizing Map): Um Referenzvektoren für Datencluster zu erstellen.

- Profiling: Errechnet einen Featurevektor für den neuen Gesprächsdatensatz und hält sowohl das aktuelle als auch das historische Benutzerprofil aktuell.

- Alarming: Berechnet die Abweichung zwischen aktuellem und historischen Profil und löst bei überschreiten eines konfigurierbaren Schwellwerts einen Alarm aus.

Da die selbst organisierende Karte, wie später noch begründet wird, jeweils nur während eines Initialisierungsdurchlaufs zum Einsatz kommt kann diese durch eine Schalter- bzw. Entscheidungskomponente umgangen werden.

Die Komponente AlarmManager stellt einen gesonderten Teil des Systems dar dessen Aufgabe der Empfang von Alarm-Nachrichten aus dem Analyseprozess und die Verteilung auf verschiedene Kanäle ist.

Die beiden Komponenten Identifier 1 und 3 sollen nochmals veranschaulichen dass es sich um einen erweiterbaren sequentiellen Prozess handelt.

Um nun das System durch einen erweiterbaren Workflow, einer Sequenz von Aktionen und Entscheidungen, modellieren zu können wird auf das „Spring Application Framework" zurückgegriffen. Während man in anderen Frameworks wie z.B. J2EE zusätzlich zur eigentlichen Business Logik viele Methoden und Interfaces betreffend Lifecycle implementieren muss ermöglicht Spring dem Entwickler sich weitestgehend auf diese zu konzentrieren. Neben dieser Eigenschaft die durch einen „light weight container“ ermöglicht wird bietet es auch zusätzliche Funktionalitäten die es auf einfache Art und Weise erlauben die Applikation zu Konfigurieren, auf Datenbanken zuzugreifen oder Web User Interfaces zu erstellen.

Das dem Spring Application Framework zu Grunde liegende Prinzip nennt sich „Inversion of Control“. Vereinfacht kann dieses Prinzip durch folgende Punkte beschrieben werden:

- Die Funktionalität des Systems wird in “kleine” handhabbare Stücke zerteilt.
- Diese Stücke werden durch einfache „Java Beans“ repräsentiert.
- Der Entwickler muss sich keine Gedanken über den Lifecycle dieser Beans machen (Instanzierung, Setzen von Abhängigkeiten usw.)
- Stattdessen übernimmt der Spring Container, basierend auf einer XML Konfigurationsdatei, diese Aufgabe.

Das „Spring Application Framework“ (siehe Abbildung 41) setzt sich aus mehreren unabhängigen Sub-Frameworks zusammen:

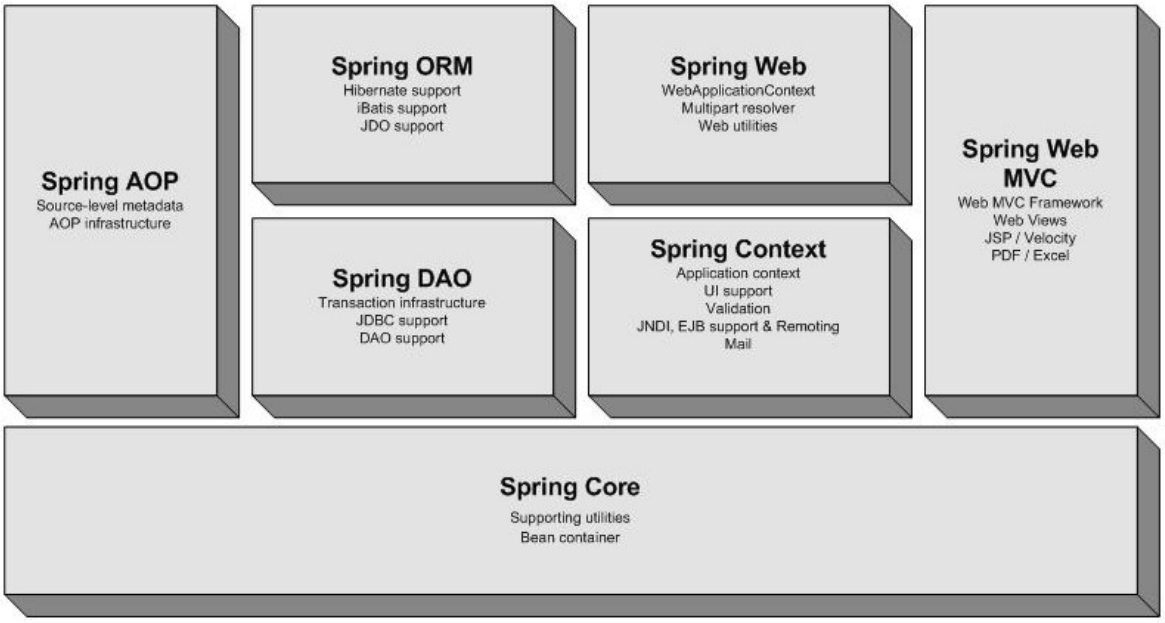

Abbildung 41: Überblick über das Spring Framework [Spri06]

- Spring Core: Dieses Paket stellt die grundlegende Funktionalität, auf dem alle anderen Pakete aufbauen, zur Verfügung. Wesentliches Konzept dahinter ist die Anwendung des „Factory-Patterns” welcher es dem Entwickler ermöglicht die Konfiguration und Spezifikation von Abhängigkeiten von der eigentlichen Programmlogik zu trennen.

- Spring Context: Stellt eine Methode zur Verfügung um Zugriff auf Beans in einer einheitlichen Art und Weise zu erhalten und erweitert „Spring Core“ um Funktionalitäten in den Bereichen Ressourcenverwaltung, Eventpropagation und applikationsspezifischen Kontexten.

- Spring DAO (DAO = data access object): Stellt vereinfacht beschrieben eine spezielle JDBC Abstraktionsschicht zur Verfügung welche den Umgang mit Datenbanken erleichtert. Weiters werden Mittel zum programmatischen und deklarativen Transaktionsmanagement zur Verfüg gestellt.

- Spring ORM: Dieses Paket enthält Integrationsschichten für verschiedene populäre objekt-relationale Mapping APIs (JDO, Hibernate). Durch die Nutzung dieser Schicht können z.B. auch die Transaktionsfeatures von Spring in Kombination mit diesen Mappern genutzt werden.

- Spring AOP: Integriert aspektorientierte Programmierung in das Framework, genauer in den Container.

- Spring Web: Dieses Paket stellt Funktionalitäten für die Webentwicklung, wie z.B. einem weborientierten Applikationskontext, bereit. Der Einsatz dieses Pakets ermöglicht beispielsweise auch die Integration mit Struts (einem populären

Framework des Apache Jakarta Projekts welches die Entwicklung von Webapplikationen auf Basis des Model-View-Controller (MVC) Patterns erlaubt).

- Spring Web MVC: Stellt eine MVC Implementierung für Webapplikationen bereit.

Im Zuge der Entwicklung der Applikation Profile-Monitor wird vor allem von den Paketen „Spring Core“ und „Spring Context“ gebrauch gemacht. Diese Pakete erlauben es, wie bereits beschrieben, das System durch eine XML Datei zu konfigurieren und zusammenzustellen wodurch der Aufbau des Workflows ermöglicht wird.

Weiters wird das Paket „Spring DAO“ eingesetzt, welches den Zugriff auf eine Datenbank in der Gesprächsdaten gespeichert wurden erleichtert.

6.2.3 Benutzerprofil

Da durch fixe Kriterien nur bekannte Betrugsfälle identifiziert werden können und die meisten Fälle auch mit einer Änderung im Anruferverhalten einhergehen wird der Weg der Erkennung über Benutzerprofile gewählt. Bei diesem Ansatz ist darauf zu achten das ein „default“ Profil nicht für die Bewertung jedes Benutzer geeignet ist. Beispielsweise können 10 Auslandsgespräche pro Woche für einen Kunden das Normalverhalten darstellen während es für einen anderen einen möglichen Missbrauchsfall signalisiert. Ein einziges Referenzprofil für alle Benutzer könnte in Kombination mit einem niedrigen Schwellwert für Abweichungen davon dazu führen, dass gerade umsatzstarke Kunden durch eine hohe Anzahl an Fehlalarmen verärgert werden. Die Folge dieser Erkenntnis ist, dass für jeden Benutzer ein aktuelles und ein eigenes historisches Profil (als Referenz) erstellt und gepflegt werden muss. Der Ansatz der diesen Forderungen nachkommt ist der differentiale Ansatz. Weiters sollte zwischen den Daten die in das aktuelle Profil und jenen die in das historische mit einfließen eine bestimmte Anzahl an Gesprächen liegen.

Die grundsätzliche Implementierung der Profile und die Ermittlung der Abweichung zwischen dem aktuellen und dem historischen erfolgt in Anlehnung an die Arbeit *„An Unsupervised Neural Network Approach to Profiling the Behavior of Mobile Phone Users for Use in Fraud Detection“* von Burge und Shawe-Taylor [BuSh01].

Künstliche neuronale Netz werden in diesem Zusammenhang zum Prototyping eingesetzt. Darunter versteht man ein Verfahren zur optimalen diskreten Abbildung einer kontinuierlichen Variablen. Für jeden Cluster/Prototyp wird im Zuge des Trainings ein Gewichtsvektor (Codeword, Referenzvektor) ermittelt der stellvertretend für eine Anhäufung von Daten ist. Diese Aufgabe wird in der Applikation Profil-Monitor durch eine selbst organisierende Karte übernommen.

Die einzelnen Einträge im Profil ergeben sich durch die gewichtete Integration eines aus den normalisierten euklidischen Distanzen zu den N Prototypen bestehenden Vektors. Der Vektor entspricht dabei der Wahrscheinlichkeitsverteilung in Bezug auf die Zuordnung zu den Prototypen.

Vereinfacht können die zur Erstellung der Profile und gegebenenfalls Auslösung von Alarmen notwendigen Schritte folgendermaßen beschrieben werden.

1. Ermittlung der Prototypen (Referenzvektoren).

2. Codierung der neuen Verrechnungsdatensätze mit Hilfe der Prototypen.

3. Integration des codierten Datensatzes in das aktuelle Profil.

4. Integration des codierten Datensatzes in das historische Profil.

5. Speichern der neu berechneten Profile

6. Vergleich von aktuellem und historischem Profil um gegebenenfalls einen Alarm auszulösen.

Schritt 1 wird im Workflow durch die Komponente SOM erfüllt, Schritt 2 bis 5 durch die Komponente Profiling und Schritt 6 durch die Komponente Alarming.

Im Folgenden wird auf jeden dieser Schritte näher eingegangen.

Ermittlung der Prototypen (Referenzvektoren).
Zur Ermittlung der die Datencluster repräsentierenden Referenzvektoren wird ein bestimmter Typus von künstlichem neuronalen Netz eingesetzt, die selbst organisierende Karte. Implementiert wird diese Karte auf Basis der Programmiersprache JAVA und der Bibliotheken von JOONE (Java Object Oriented Neural Engine).

JOONE ist ein frei verfügbares Framework zur Erstellung, Training und Test von künstlichen neuronalen Netzen. Netze die aufsetzenden auf diesem Framework entwickelt wurden können lokal gebaut, danach in einer verteilte Umgebung trainiert werden um schlussendlich auf den verschiedensten Gerät zum Einsatz zu kommen.

Einige Hauptmerkmale von JOONE

- Graphische Umgebung zur Konstruktion von neuronalen Netzen
- Unüberwachtes Lernen:
    - Selbstorganisierende Karten (“Winner Takes All”- oder Gauß- Output-Karten)
    - Hauptbestandteils Analyse (Principal Component Analysis)
- Überwachtes Lernen:
    - Vorwärtsverkettete Netze (Feed Forward neural networks)
    - Rekursive Netze (Elman, Jordan ...)
    - Zeitverzögerte Netze (Time Delay neural networks)
    - Lernalgorithmen:
        - Standard Back-prop (Gradient Descent, on-line und batch)
        - Resilient Back-prop (RPROP)

- Gemischte Netze durch einen Modularen Aufbau

- Mechanismen zur Datenvorverarbeitung

In der Applikation Profile-Monitor werden davon vor allem die Möglichkeiten zur Konstruktion von selbst organisierenden Karten genutzt. Ob ein „Winner Takes All“ oder ein Gaussian-Layer im Prototypen zum Einsatz kommt wird erst im Zuge der Entwicklung des Prototypen entschieden.

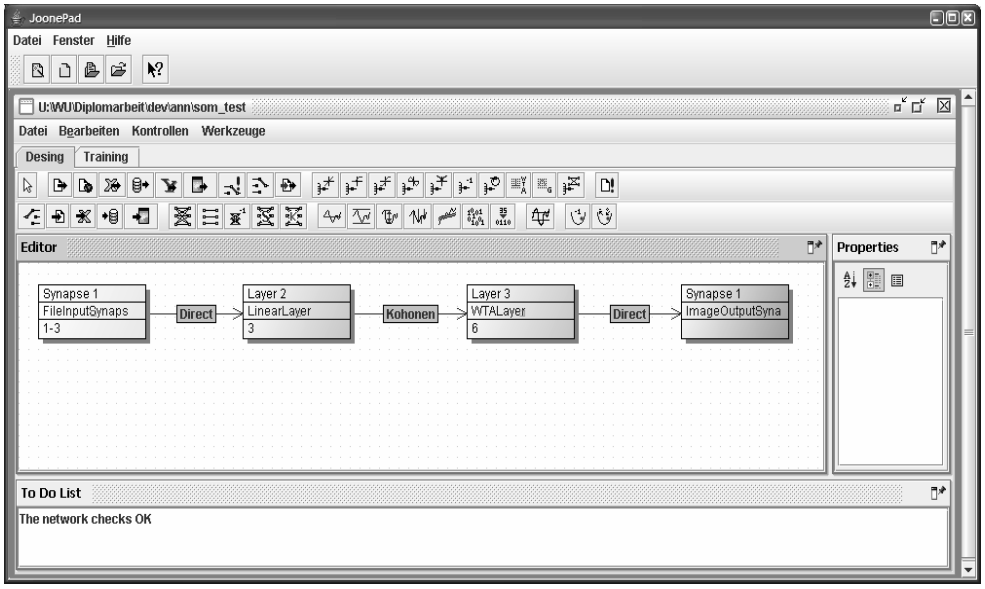

Abbildung 42: JOONE Modellierung einer selbst organisierende Karte [MaJo06]

Abbildung 42 zeigt das grafische Userinterface welches bei JOONE zur Modellierung von künstlichen neuronalen Netzen eingesetzt werden kann. Im konkreten Fall ein Beispiel für eine selbst organisierende Karte die ihren Input aus einer Datei bezieht und mittels so genannter Kohonen Synapsen mit einem „Winner Takes All“-Layer verbunden ist.

Während des Trainings der Karte tendiert das Netz dazu gegen eine stationäre Verteilung zu konvergieren und damit eine optimale diskrete Abbildung eines kontinuierlichen Inputs zu erzeugen. Die sich ergebenden Gewichtsvektoren entsprechen den gesuchten Referenzvektoren der gefundenen Datencluster. In der Terminologie der “Vector Quantization” spricht man in diesem Zusammenhang auch von der Erstellung des “Codebooks” welches so genannte “Codewords” beinhaltet. Die durch einen Referenzvektor „abgedeckte“ Region von Daten wird in der Literatur auch als Voronoi Region bezeichnet [JuMH05].

Ergebnis dieses Vorgangs sind N Referenzvektoren/Prototypen welche die gleiche Dimensionalität wie der Inputvektor aufweisen. Die Anzahl dieser Vektoren steht im direkten Zusammenhang mit der Anzahl der Outputneuronen der Karte, jedem Outputneuron ist ein Gewichtsvektor (Referenzvektor) zugeordnet. Da im Weiteren nur noch diese durch die Applikation genutzt werden ist eine permanente Integration der Karte in den Workflow nicht notwendig und eine dauerhafte Speicherung der in einem Trainingsdurchlauf ermittelten Referenzvektoren ausreichend. Der Workflow wird durch einen Parameter in einen Trainings- oder einen Analysemodus versetzt. Referenzvektoren werden in einer lokalen Datei oder einer Tabelle in der Datenbank abgelegt.

Codierung der neuen Verrechnungsdatensätze mit Hilfe der Prototypen.
Mit Hilfe der Prototypen können neue Inputvektoren kodiert werden. Bei diesem Vorgang wird ein Featurevektor errechnet der sich aus der zu jeden Prototypen normierten euklidischen Distanz zusammensetzt.

Die Komponenten des Featurevektors können mittels folgender Formel bestimmt werden:

$$v_j = \frac{\exp(\left\| X_{N+1} - Q_j \right\|)}{\sum_{j=1}^{K} \exp(\left\| X_{N+1} - Q_j \right\|)}$$

v_j … Komponente mit Index j des Featurevektors
X … Inputvektor.
Q_j … Prototyp j von K

Mittels dieser Kodierung gehen Abweichungen vom Prototyp als absolut Betrag in die Berechnung ein. Weiters werden hohe Werte mittels des durch die Exponential-Funktion bestehenden nichtlinearen Zusammenhangs mehr „bestraft“. Während eine Abweichung von 1 einen Funktionswert von 2,718 (e) zur Folge hätte, würde eine Abweichung von 3 mit 20,086 in die weitere Berechnung einfließen.

Abbildung 43 veranschaulicht die Funktion e^x.

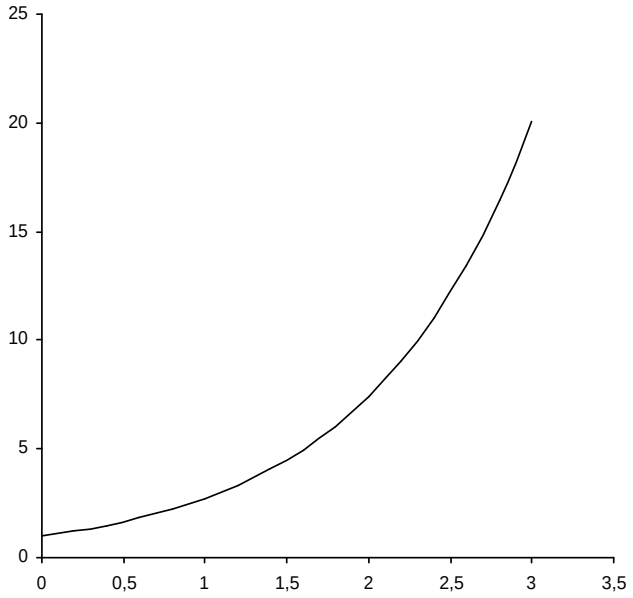

Abbildung 43: e^x im Intervall]0; 3[

Um andererseits keine zu hohen Strafterme für Abweichungen zu erhalten werden Inputwerte (sowohl für die Bestimmung neuer Prototypen als auch bei der Berechnung der Profile) auf das Intervall]0; 10[übertragen. Ein weiterer Vorteil dieser nicht linearen Funktion ist, dass falls alle Abweichungen vom Prototypen 0 ergeben keine Division durch 0 erfolgt.

Durch die Normierung auf die Gesamtabweichung ergibt die Summe aller Komponenten des Featurevektors 1 und kann dadurch als Wahrscheinlichkeitsverteilung angesehen werden [BuSh01].

$$\sum_{j=1}^{K} v_j = 1$$

Jede Komponente des Featurevektors entspricht somit der normierten euklidischen Distanz des Inputvektors zu einem der Prototypen.

Integration des codierten Datensatzes in das aktuelle und historische Profil.
Um den Featurevektor in das aktuelle und historische Profil, welche durch zwei verschiedene Zeitabschnitten gekennzeichnet sind, zu integrieren wird dieser jeweils mit einem eigenen Dämpfungsfaktor berücksichtigt.

Die Anpassung des historischen Profils soll erst nach einer Verzögerung von B Aktualisierungen des aktuellen erfolgen. Dadurch muss sichergestellt werden, dass der Einfluss jedes Verrechnungsdatensatzes auf das historische Profil nur so groß ist als ob jede Änderung des aktuellen Profils mit dem eigenen Dämpfungsfaktor berücksichtigt worden wäre. Um diese Wirkung zu erzielen wird für jedes Ticket ein eigener Faktor angewandt.

$$\alpha_{t+1} = \frac{\beta}{1 - \alpha_t + \beta}$$

α … Dämpfungsfaktor angewandt auf das aktuelle Profil
β … Dämpfungsfaktor angewandt auf das historische Profil

Dies führt bei $0 \le \beta \le \alpha \le 1$ zu Anpassungsfaktoren im Intervall $]\beta, \alpha_0[$ und zu einer iterationsabhängigen Formel für das aktuelle Profil.

Zeitpunkt	Alpha
0	0,80000
1	0,50000
2	0,28571
3	0,21875
4	0,20382
5	0,20077
6	0,20015
7	0,20003
8	0,20001
9	0,20000
10	0,20000
11	0,20000
12	0,20000
13	0,20000
14	0,20000

Tabelle 3: Dämpfungsfaktorbeispiel für $\beta = 0{,}2$ und $\alpha_0 = 0{,}8$

Mittels des Dämpfungsfaktors und des folgenden Zusammenhangs wird das aktuelle Profil nach jedem Datensatz aktualisiert.

$$C_i^{t+1} = \alpha_{t+1} C_i^t + (1 - \alpha_{t+1}) v_i$$

C_i … Komponente mit dem Index i des aktuellen Profils

Auf Grund der Formel und der Forderung nach einer Wahrscheinlichkeitsverteilung gilt auch hier, dass die Summierung der Komponenten des Profils 1 ergibt.

$$\sum_{i=1}^{K} C_i = 1 .$$

Die Anpassung des historischen Profils erfolgt nach B Aktualisierung des aktuellen Profils auf folgende Art und Weise.

$$H_i = \beta H_i + (1 - \beta) C_i$$
$$\sum_{i=1}^{K} H_i = 1$$

H_i … Komponente mit dem Index i des historischen Profils

Speichern der neu berechneten Profile
Sowohl das aktuelle als auch das historische Profil werden dauerhaft in einer lokalen Datei oder einer Datenbank abgelegt. Eine Aktualisierung erfolgt aus Performancegründen nur alle B Verrechnungsdatensätze.

Vergleich von aktuellem und historischem Profil um gegebenenfalls einen Alarm auszulösen.
Um das Ausmaß der Abweichung zwischen den beiden Profilen zu bestimmen wird dafür als Maß die Hellinger Distanz herangezogen.

$$d = \sum_{i=1}^{K} (\sqrt{C_i} - \sqrt{H_i})^2$$

Sie ist ein natürliches Maß um zwei Wahrscheinlichkeitsverteilungen zu vergleichen und nimmt immer einen Wert zwischen 0 für vollkommene Übereinstimmung und 2 für Orthogonalität an [BuSh01].

Diese Distanz wird nach jeder Änderung im aktuellen Profil, nach jedem neuen Verrechnungsdatensatz, errechnet. Überschreitet sie eine gewisse vorgegebene Schwelle wird ein Alarm generiert. Die Feinabstimmung dieser Schwelle muss in Zusammenhang mit der „Receiver Operating Characteristics“ (siehe Anhang 8) gesehen werden und wird im Verlauf der Entwicklung manuell durch verschiedene Versuche durchgeführt.

Bei diesem Schritt ist weiters anzumerken, dass Alarme nach ihrer Distanz, dem Ausmaß der Änderung des Verhaltens, priorisiert und in einer danach geordneten Liste abgelegt werden können. Falls nur die obersten Einträge der Liste bearbeitet werden kann eine niedrigere Schwelle konfiguriert werden.

6.2.4 Input Daten

Als Trainingsdaten für das neuronale Netz werden Echtdaten eines großen deutschen Mobilfunkanbieters für eine Stadt wie München (oder einem Teil davon) herangezogen. Diese wurden bereits aus den binären Verrechnungsdateien extrahiert und Teildatensätze zu einzelnen Gesprächen aggregiert. Da in den ursprünglichen Daten eine Fülle von zusätzlichen Informationen die für diesen ersten Prototypen nicht von Relevanz sind enthalten sind (z.B. SMS Informationen) wurden diese gefiltert. Als Input für das System verbleiben nach der Filterung nur Daten von so genannten „mobile originated calls“, die in Form von CSV-Dateien oder in einer Datenbank zur Verfügung stehen.

Fiktives Beispiel für Inputdaten (ohne IMEI):

```
REFERENCE_TIME,IMSI,CONFORMED_CALLING_NUMBER,CONFORMED_CALLED_NUMBER,DURATI
ON,DEST_ID
20050823223546,262073971099999,491799099999,491797899999,2458,86994
20050823223841,262073960399999,491795199999,491712299999,201,86995
20050823224907,262073930299999,491797799999,4916095499999,574,86995
20050823225903,262073951499999,4917620999999,491762499999,94,86994
…
```

Felder der Inputdaten:

1. IMSI - International Mobile Subscriber Identity: Diese Identifikationsnummer dient als eindeutige internationale Kennung eines Mobilfunkteilnehmers und ist vor allem im Zusammenhang mit Roaming von Bedeutung. Mittels dieser Nummer können Verrechnungsdatensätze die für dieses System als Input herangezogen werden eindeutig zugeordnet werden.

   Der Schlüssel zum jeweiligen Benutzerprofil erfolgt ebenfalls über diese Kennung.

   Sie setzt sich aus mehreren Teilen zusammen:
   a. MCC Mobile Country Code,
   b. MNC Mobile Network Code,
   c. MSIN Mobile Subscriber Identification Number

   Beispielsweise:

MCC	MNC	HLR	SN
232	01	62	12345678

   232 steht für Österreich, 01 für das Netz der Mobilkom Austria, 62 für die logische HLR-Adresse.

2. IMEI - International Mobile Equipment Identity: Entspricht der Seriennummer des Geräts (nicht der SIM-Karte)

   Sie besteht aus den folgenden Feldern:
   a. TAC Type Approval Code (6 Zeichen)
   b. FAC Final Assembly Code (2 Zeichen)
   c. SNR Serial Number (6 Zeichen)

d. SP Spare (1 Zeichen)

Ursprünglich sollte diese Nummer im Zusammenhang mit einem EIR - Equipment Identity Register verwendet werden welches aus mehreren Listen besteht. Darunter eine mittels der gestohlenen Endgeräten jedes Service verweigert werden sollte.

Beispielsweise kann an Hand des TAC folgende Aussage getroffen werden:
Bei einem TAC von 448904 handelt es sich um ein Endgerät des Herstellers Nokia vom Typ 7110.

3. A-Teilnehmernummer: Die Rufnummer des rufenden Teilnehmers im Format der MSISDN (Mobile Subscriber Integrated Service Digital Networknumber). Eine MSISDN besteht aus 14 bis 15 Ziffern und setzt sich aus den folgenden Teilen zusammen:
    a. CC Country Code
    b. NDC National Destination Code
    c. SN Subscriber Number
        i. HLR-Nummer
        ii. Individuelle Teilnehmer Nummer

   Beispielsweise kann die Nummer 43 664 62 8xxxx folgend aufgeschlüsselt werden. 43 entspricht dem Country Code von Österreich, 664 dem National Destination Code der Mobilkom Austria und 628xxxx der Kundennummer (genauer dem Home Location Register und der Teilnehmer Nummer).

4. B-Teilnehmernummer: Die Rufnummer des angerufenen Teilnehmers, ebenfalls im MSISDN Format

5. Dauer: Die Dauer des Gesprächs in Sekunden. Teildatensätze (partial records) der Verrechnungsdatei wurden bereits zu einem gesamten Gesprächsdatensatz aggregiert.

6. Zeit: Zeitpunkt zu dem das Gespräch geführt wurde.

Für die angestrebte Identifizierung von Missbrauchsfällen (bzw. zur Generierung des Benutzerprofils) werden davon die folgenden herangezogen.

- IMSI als eindeutiger Schlüssel für das Benutzerprofil.

- Dauer als die Gesprächsdauer.

Weiters werden die folgenden Eingaben für das künstliche neuronale Netz aus der „B-Nummer“ und der „Zeit“ abgeleitet:

- B-Nummer um abgeleitete quantitative Aussagen über verschiedene Rufziele zu tätigen (die A-Nummer dient hier als Referenz für den Anrufer)

    - Durchschnittliche Dauer von Rufen zu
        - internationalen Nummern
        - nationalen Nummern
        - lokalen Nummern

        - Mehrwertdiensten (0900 Nummern, 0190 Nummern wurden mit 31. Dezember 2005 in Deutschland eingestellt)

    innerhalb der jeweils letzten 20 Anrufe.

- Zeit um abgeleitete quantitative Aussagen über Rufe zu verschiedenen Tageszeiten zu tätigen

    - Durchschnittliche Dauer von Anrufen innerhalb der jeweils letzen 20 Anrufe in der Zeit zwischen
        - 06:00 und 09:00.
        - 09:00 und 18:00.
        - 18:00 und 22:00.
        - 22:00 und 06:00.

Durchschnittswerte können nicht durch das Moving Average Plugin von JOONE ermittelt werden da ein Bezug zum Benutzer hergestellt werden muss, d.h. der Durchschnitt darf nicht einfach mittels aller aufeinander folgenden Datensätze ermittelt werden sondern muss pro IMSI (Benutzer) berechnet werden.

Die Unterscheidung zwischen Gesprächen zu internationalen, nationalen oder Mehrwertdiensten erfolgt durch das festgelegte Format der MSISDN und dem Vergleich zwischen A- und B-Teilnehmernummer.

Beispielsweise gilt ein Anruf bei dem zwischen Anrufer und Angerufenen

- Country- und Nationale Destination Codes übereinstimmen als lokales Gespräch (im Falle von Mobilfunknetzen bedeutet dies ein Anruf innerhalb desselben Mobilfunknetzes)

- Country-Codes nicht übereinstimmen als international und jene bei denen

- Country-Code aber nicht Nationale Destination Code übereinstimmen als national.

In Verrechnungsdateien finden sich zu Gesprächen auch Informationen zur Mobilität des Kunden während des Anrufs wie die Zelle in der das Gespräch initiiert wurde und die in der es beendet wurde (jeweils repräsentiert durch die Local Area Identity welche einen Zellenverbund international eindeutig kennzeichnet). Daraus könnten in einer späteren Entwicklungsphase weitere Informationen für das Profil eines Kunden gewonnen werden. Beispielsweise die durchschnittliche Anzahl an verschiedenen Start-Zellenverbunden usw.

Zur näheren Analyse der Applikation Profile-Monitor werden anfangs künstlich generierte Daten herangezogen, anders gesagt es wird ein Kunde simuliert. Dazu werden verschiedene CSV Dateien manuell erstellt. Inhalt dieser Dateien ist eine fixe Anzahl an Gesprächen die einem vorgegebenen Profil in Bezug auf Rufziele (international, national, lokal, „premium rate services“ usw.), Gesprächsdauer und Tageszeit (morgens, abends, tagsüber und am Wochenende) folgen. Durch diese Maßnahme können Änderungen im Verhalten des Benutzers einfacher und kontrollierter dargestellt werden.

Die zur Verfügung stehenden Daten eines deutschen Mobilfunkbetreibers die zum Training der selbst organisierenden Karte herangezogen werden entsprechen den oben beschriebenen Inhalten und können weiters folgend charakterisiert werden:

Anzahl der Datensätze	170200
Anzahl der MSCs	2
Standort der MSCs	München
Zeitpunkt	Keine komplette Datenbasis zwischen 23. August 2005 22:35:46 und 22. Mai 2006 16:21:46
Gesprächsdauer	Maximal 7202 Sekunden Minimal 0 Sekunden
Anzahl von erfassten Kunden (IMSI)	80054
Anzahl an Gesprächen pro Kunden	Maximal 172 Minimal 1

Tabelle 4: Inputdaten

6.2.4.1 Skalierung

Die Skalierung von quantitativen Daten hat Einfluss auf die Orientierung der Karte, obwohl bei selbst organisierenden Karten eine Skalierung nicht notwendig ist wird diese manchmal empfohlen [Pass04a][Koho01]. Typisch ist dabei eine lineare Skalierung [Stat06].

Eine solche Skalierung bzw. Normierung kann beispielsweise mittels der folgenden Formel erreicht werden:

$$y_{neu} = \frac{y - \min}{\max - \min} \cdot \left(\max_{neu} - \min_{neu}\right) + \min_{neu}$$

y	der aktuelle Wert der Variable
y_{neu}	der neue, normierte Wert der Variable
$\max - \min$	der aktuelle Wertebereich
$\max_{neu} - \min_{neu}$	der neue Wertebereich (z.B. -1 bis +1)

Nachteil dieser Methode ist, dass sich der Wertebereich (min bis max) unbekannter Daten jederzeit ändern kann.

Falls das Inputfeature eine diskrete Zufallsvariable mit positiver Varianz ist kann alternativ die statistische Methode der linearen Transformation durchgeführt werden.

$$y_{neu} = \frac{y - \mu_y}{\sqrt{\sigma_y^2}}$$

Die resultierende Variable ist durch diese Transformation eine so genannte standardisierte Zufallsvariable mit dem Erwartungswert $\mu_{y_{neu}} = 0$ und der Varianz $\sigma^2_{y_{neu}} = 1$.

Bei der Entwicklung des Prototyps der Applikation wird von bekannten Daten ausgegangen (Maximal- und Minimalwerte sind bekannt). Die Skalierung erfolgt auf ein Intervall]0, +10[mittels der zuerst beschriebenen Methode.

Aufgrund des Aufbaus der Applikation durch einen Workflow bei dem die Komponente die ein künstliches neuronales Netz mittels der Bibliotheken von JOONE implementiert nur in der Trainingsphase (Bestimmung der Referenzvektoren) zum Einsatz kommt kann die Skalierung nicht durch das NormalizerPlugin von JOONE durchgeführt werden. Um sinnvolle Vergleiche zu den Referenzvektoren ziehen zu können ist in der Analysephase eine Skalierung auf das Intervall]0, +10[ebenfalls notwendig.

6.2.5 Alarmierung

Jede der Komponenten des Workflows kann Alarme auslösen in dem es Aufträge an eine zentrale Komponente sendet. Ein Auftrag besteht dabei im Wesentlichen aus der ID des Benutzers (IMEI) für den der Alarm ausgelöst werden soll, ein Maß für die Priorität (eine Zahl zwischen 0 und 2) und einer Beschreibung.

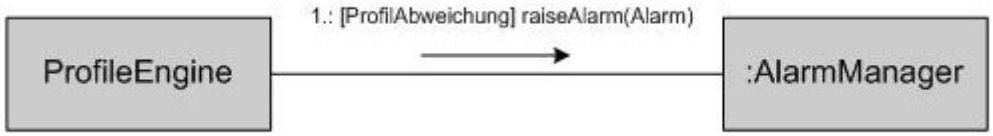

Abbildung 44: Alarmbeispiel

Die zentrale Komponente, in Abbildung 44 der Alarm Manager, ist für die Bewertung (Priorisierung), die Auflistung, und die Verteilung von Alarmen mittels verschiedener Alarm-Mechanismen verantwortlich.

Alarme können unter anderem mittels der folgenden Mechanismen verteilt werden:

- Protokollierung in einer Datei (Log)
- Protokollierung in der Konsole
- Versenden per SMTP/Email (Simple Mail Transfer Protocol)
- Versenden per SNMP (Simple Network Management Protocol)
- Eintragung in einer Datenbank mittels JDBC

Bei der Implementierung der Alarmierung kommt Log4j [ASF06] des Apache Jakarta Projekts zum Einsatz. Dieses System bietet neben der Funktionalität zur Erstellung von Log-Nachrichten in einer Datei auch die Möglichkeit Alarme per Email oder SNMP zu versenden.

Da sowohl für Email als auch für SNMP externe Systeme wie beispielsweise ein POP3 Server im Falle des Versands per Email zur Verfügung stehen müssen beschränkt sich der Prototyp der Applikation im ersten Schritt auf die Nutzung von Dateien und der Konsole. Weiters ist für den Einsatz von SNMP zusätzliche third-party Software notwendig. Eine Änderung des Mechanismus kann jederzeit durch Konfiguration erfolgen.

Datei	org.apache.log4j.RollingFileAppender
Konsole	org.apache.log4j.ConsoleAppender

Email	org.apache.log4j.net.SMTPAppender
SNMP	org.apache.log4j.ext.SNMPTrapAppender
Datenbank	org.apache.log4j.jdbcplus.JDBCAppender

Tabelle 5: Alarm Mechanismus/log4j Klassenzuordnung

6.3 Design & Implementierung

In diesem Abschnitt der Arbeit wird auf Design und Implementierung der einzelnen Komponenten des Systems näher eingegangen. Jede der Komponenten wird durch ein Klassendiagramm, grundlegende Sequenzdiagramme und eine Kurzbeschreibung der Klassen erklärt.

Eine ausführliche Beschreibung der Klassen mit ihren Methoden und Attributen kann der Java-Doc entnommen werden.

6.3.1 Workflow

Der Workflow bildet das Grundgerüst der Applikation und setzt sich aus den folgenden abstrakten Basisklassen zusammen.

- ProfileMonitor (nicht abstrakt): Liest den Konfigurationskontext und startet den Workflow durch Aufruf der ersten Workflow Komponente

- AbstractComponent: Entspricht einer grundlegenden Workflow Komponente und definiert dessen Eigenschaften, genauer die Fähigkeit aufgerufen zu werden.

- AbstractAction: Erweitert die Klasse AbtractComponent und entspricht einer Aktion. Jede Aktion kennt seinen Nachfolger und delegiert nach Durchlaufen der eigenen Logik die Steuerung auf ihn.

- AbstractRule: Erweitert ebenfalls die Klasse AbstractComponent und entspricht einer Entscheidung. Das Ergebnis der Entscheidung kann dabei entweder negativ oder positiv ausfallen und ruft dementsprechend verschiedene Nachfolger auf.

Anmerkungen:

- *Komponenten die in den Workflow integriert werden sollen müssen demnach entweder von AbstractAction oder AbstractRule erben.*

Klassendiagramm (Abbildung 45):

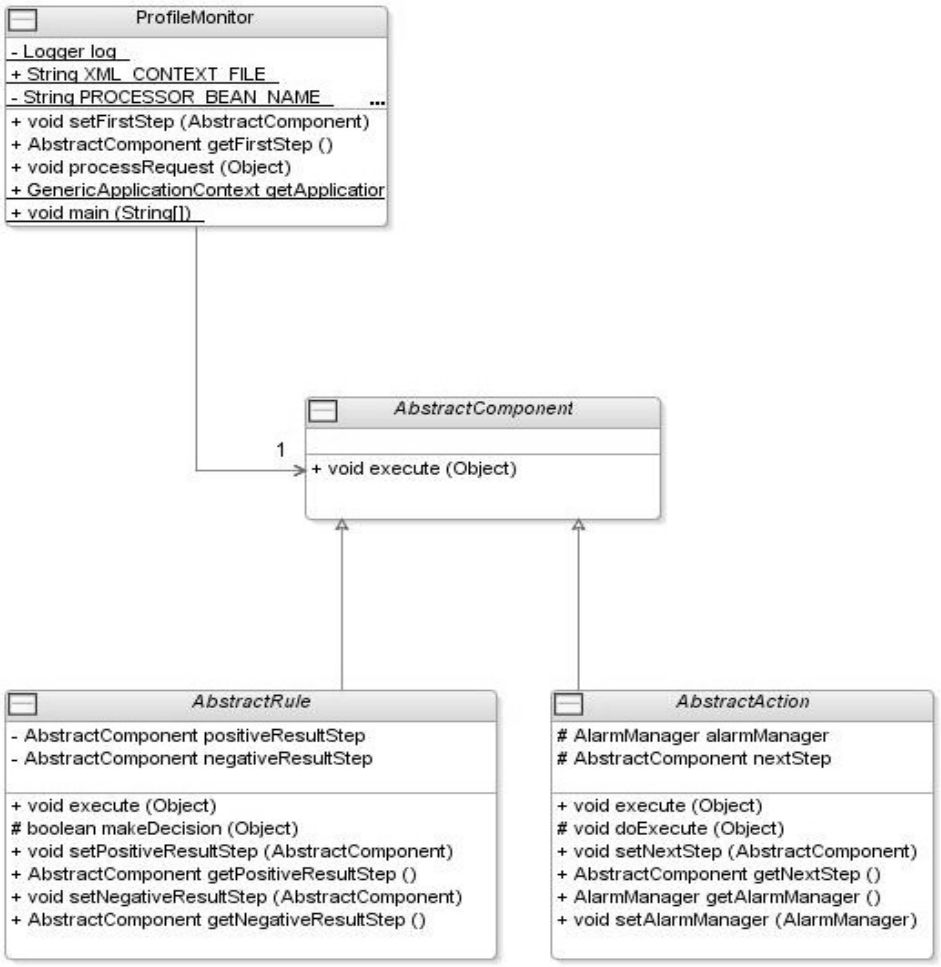

Abbildung 45: Klassendiagramm Workflow

Sequenzdiagramm (Abbildung 46):

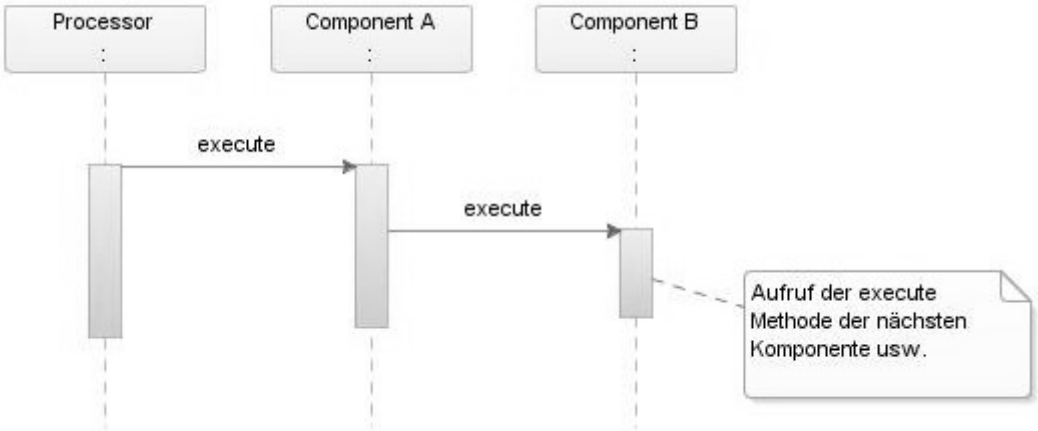

Abbildung 46: Sequenzdiagramm Workflow

Im Fall der Applikation ProfileMonitor entspricht die Klasse ProfileMonitor dem Processor der als ersten Schritt die Methode „execute“ der Aktion CSVFile (Component A) aufruft. Jede Aktion des Workflows erbt von der Klasse AbstractAction die von AbstractComponent (definiert die Methode „execute“) erbt und die nächste Aktion festlegt. Die nächste Aktion ist ebenfalls eine Instanz von AbstractComponent. Weiters wird in der Klasse AbstractAction die Methode „execute“ derart überschrieben, dass nach Abarbeitung der eigenen Logik die Methode „execute“ der nächsten Aktion aufgerufen wird. Durch dieses Konstrukt ist es möglich durch Aufruf der Methode „execute“ der ersten Komponente eine Kette von Aktionen und Entscheidungen auszulösen.

6.3.2 Workflow - Komponenten

6.3.2.1 DataProvider

Die DataProvider Komponenten dienen dazu von einer Datenquelle Datensätze zu lesen. Diese werden in Call Objekte umgewandelt und einzeln an die nächste Komponente des Workflows weitergegeben. Eine Besonderheit dieser Komponente ist, dass sie auf Grund eines Aufrufs den übrigen Workflow N mal auslöst. Beispielsweise wenn eine Datei gelesen wird und daraus N zu bearbeitende Datensätze resultieren.

Die beiden folgenden Klassen bzw. Interfaces stehen als DataProvider zur Verfügung:

- DataProvider: Marker Interface das eine Klasse als Lieferanten von Call Objekten markiert.

- Database: Liest Datensätze mittels eines DataSource Objekts aus einer Datenbank. Der Querystring ist frei konfigurierbar und die Position des Cursors wird zwischen zwei Durchläufen nicht gespeichert. Diese Eigenschaft hat zur Folge, dass durch einen nochmaligen Aufruf der Komponente wiederholt alle Datensätze, die durch den Querystring beschrieben werden, abgerufen werden.

- CSVFile: Liest Datensätze im CSV (Comma Separated Value) Format aus einer Datei.

Klassendiagramm (Abbildung 47):

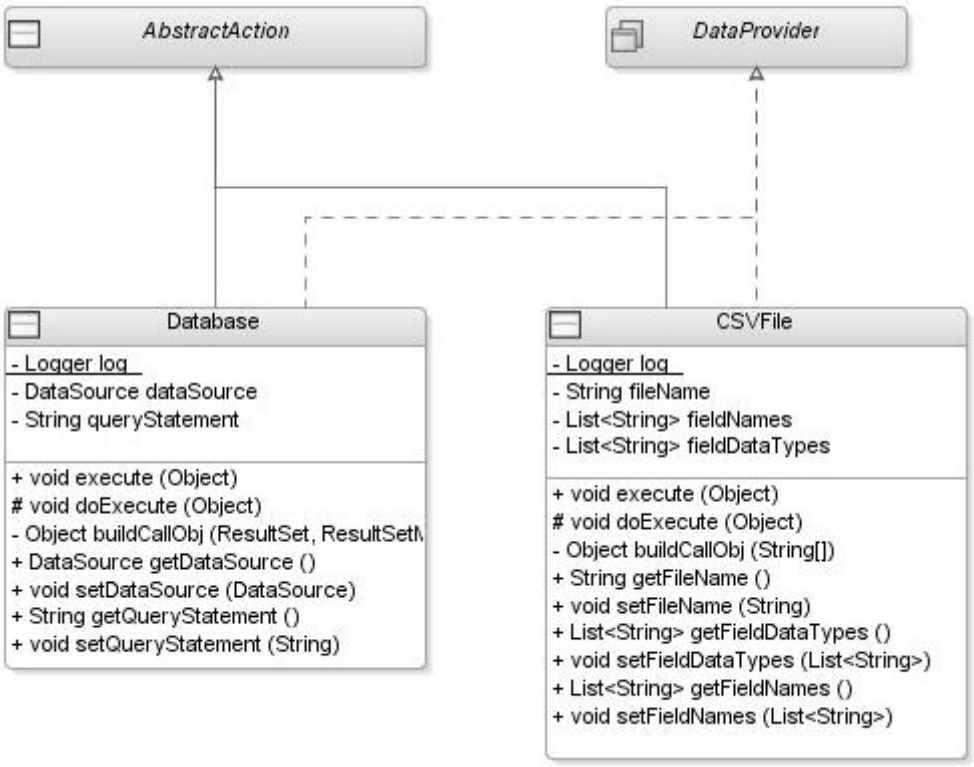

Abbildung 47: Klassendiagramm DataProvider

6.3.2.2 Feature Extraction

Die Komponente Feature Extraction dient zur Vorverarbeitung der Daten. Aufgaben die unter diesen Punkt fallen sind beispielsweise die Berechnung des gleitenden Durchschnittswerts der Gesprächsdauer oder die Skalierung der Daten. Da in der Trainingsphase der selbst organisierenden Karte und im Analysemodus die gleichen abgeleitenden Werte berechnet und skaliert werden erfolgen diese Berechnungen in einem Schritt vor der Entscheidungs-Komponente Training/Analyse. Weiters können die mit den JOONE Bibliotheken mitgelieferten Klassen zur Vorverarbeitung von Inputdaten nicht eingesetzt werden da eine Berechnung pro Kunde und nicht durchgehend auf den Datenstrom durchgeführt werden soll.

Der Aufbau dieser Komponente besteht dabei im Wesentlichen aus den folgenden Klassen:

- FeatureExtraction: Stellt die Schnittstelle, genauer die Aktion, zum Workflow dar und löst die Berechnung der Werte für den jeweiligen Kunden (identifiziert durch die IMSI) aus.

- IMSIDataRegister: Ein Objekt dieser Klasse enthält (bzw. erstellt) für jeden Kunden eine DataProcessorList (siehe nächster Punkt) und ruft die Methode „compute“ jedes in der Liste eingetragenen DataProcessors auf.

- DataProcessorList: Enthält die einem Kunden zugeordneten DataProcessor Objekte.

- DataProcessor: Interface welches eine flexible Erweiterung der Liste der Manipulationen und Berechnungen (DataProcessorList) die pro Datensatz ausgeführt werden erlaubt.

- MovingAverageDataProcessor: Berechnet den gleitenden Durchschnittswert für ein angegebenes Attribut des Call Objekts und speichert das Ergebnis als zusätzliches Attribut. Da ein Objekt dieser Klasse zwischen zwei Aufrufen seinen Status speichern muss wird für jeden Kunden eine eigene Instanz erstellt.

- Normalizer: Durch Aufruf der Methode „compute“ dieses DataProcessor erfolgt eine lineare Skalierung des Inputs auf ein konfigurierbares Intervall. Da sich Objekte dieser Klasse zwischen zwei Aufrufen keinerlei Status speichern, kann ein und dasselbe Objekt in mehrere DataProcessorList Objekte aufgenommen werden, genauer die Referenz auf dieses Objekt.

- CallDeterminationDataProcessor: Diese Klasse dient zur allgemeinen Kategorisierung des Anrufs an Hand einer im Datensatz zur Verfügung stehenden Kennung. Entspricht diese Kennung einem angegebenen Wert wird das Call Objekt um ein zusätzliches Attribut mit dem Wert 1 als Inhalt erweitert.

- SpecialCallDeterminationDataProcessor: Diese Klasse dient dazu um festzustellen ob das Gespräch zu einem bestimmten durch einen Operatorcode identifizierbaren Ziel geführt wurde. Beginnt die gerufene Nummer mit einem bestimmten Operatorcode, bestehend aus Country- und Network Destination Code, wird das Call Objekt um ein zusätzliches Attribut mit dem Wert 1 als Inhalt erweitert.

- TimeIntervallDeterminationDataProcessor: Bestimmt ob ein Anruf in einem bestimmten Tageszeitintervall getätigt wurde. Liegt die Referenzzeit des in einem Call Objekt gekapselten Anrufs im gewünschten Intervall, wird das Objekt um ein zusätzliches Attribut mit dem Wert 1 als Inhalt erweitert.

Anmerkungen:

– *Aus Gründen der Performance wird in den Klassen Call-, SpecialCall- und TimeIntervallDeterminationDataProcessor darauf verzichtet das Call Objekt um eine ein Attribut mit dem Wert 0 zu erweitern falls die jeweils überprüfte Bedingung nicht zutrifft. Da der Output dieser Processor Objekt nicht direkt als Input des neuronalen Netz dient stellt dieses Vorgehen kein Problem dar.*

– *Alle Berechnungen erfolgen mit „double“ Werten.*

Klassendiagramm (Abbildung 48):

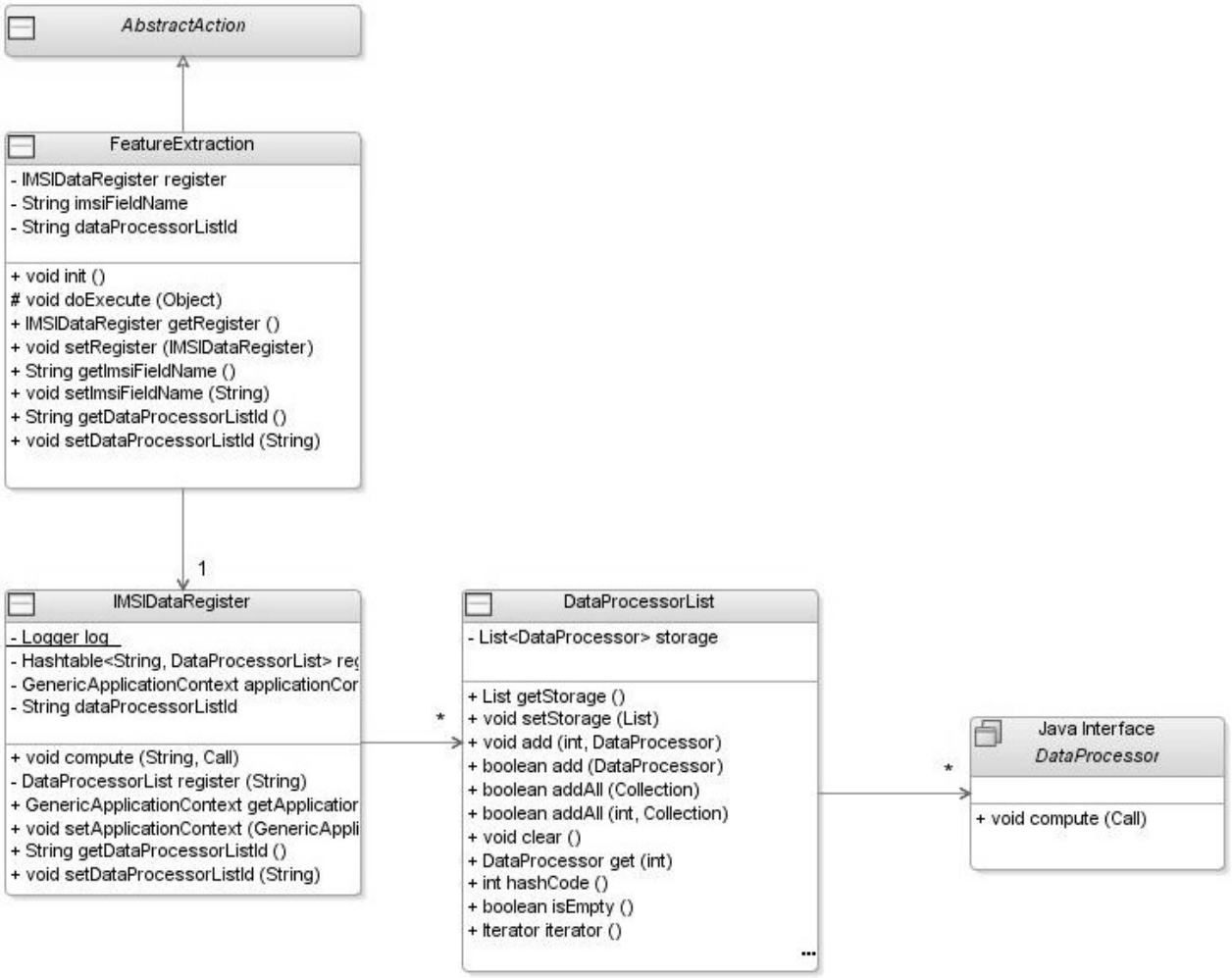

Abbildung 48: Klassendiagramm FeatureEctraction

Als DataProcessor stehen die im folgenden Klassendiagramm (Abbildung 49) angeführten zur Verfügung.

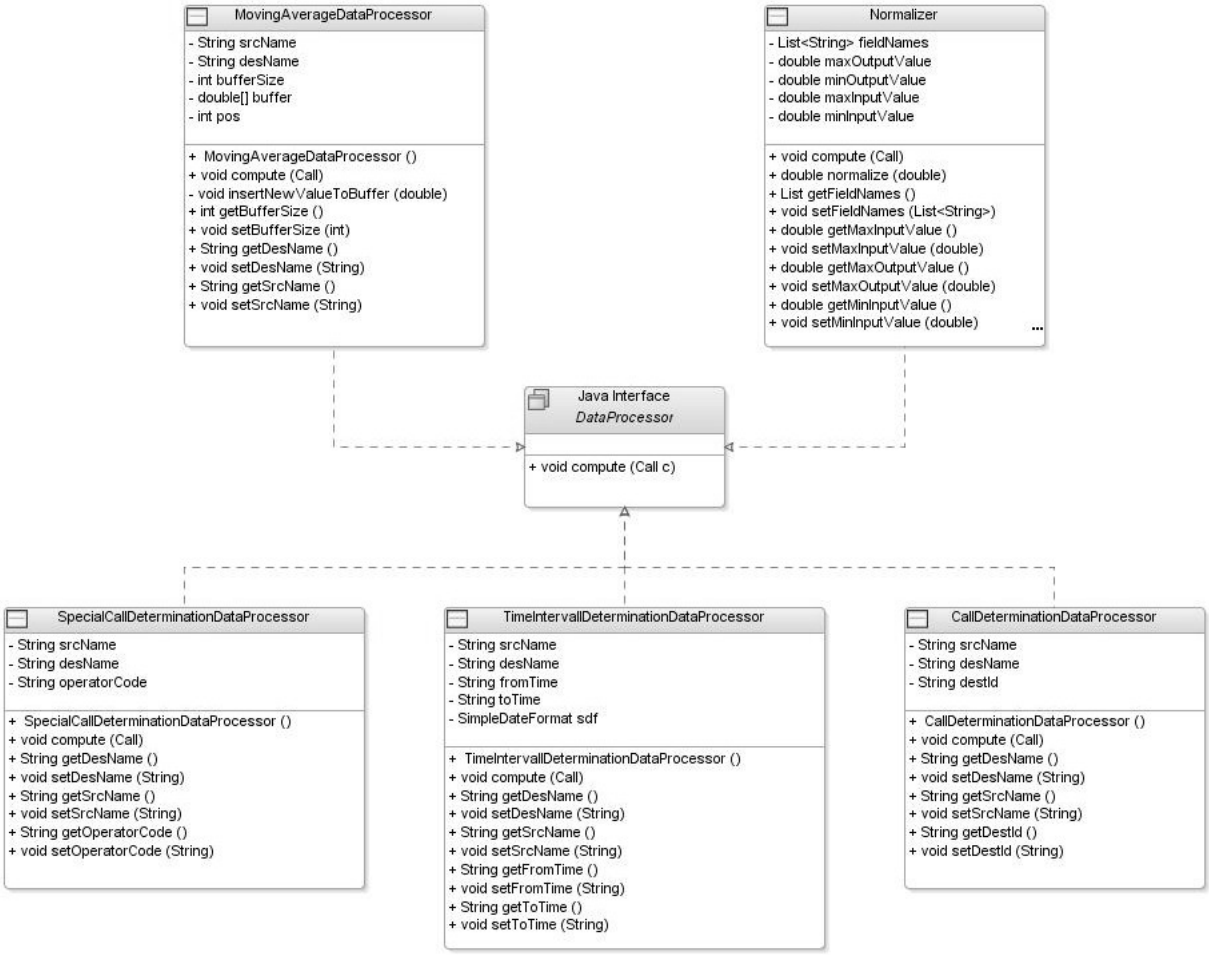

Abbildung 49: Klassendiagramm DataProcessor

Sequenzdiagramm (Abbildung 50):

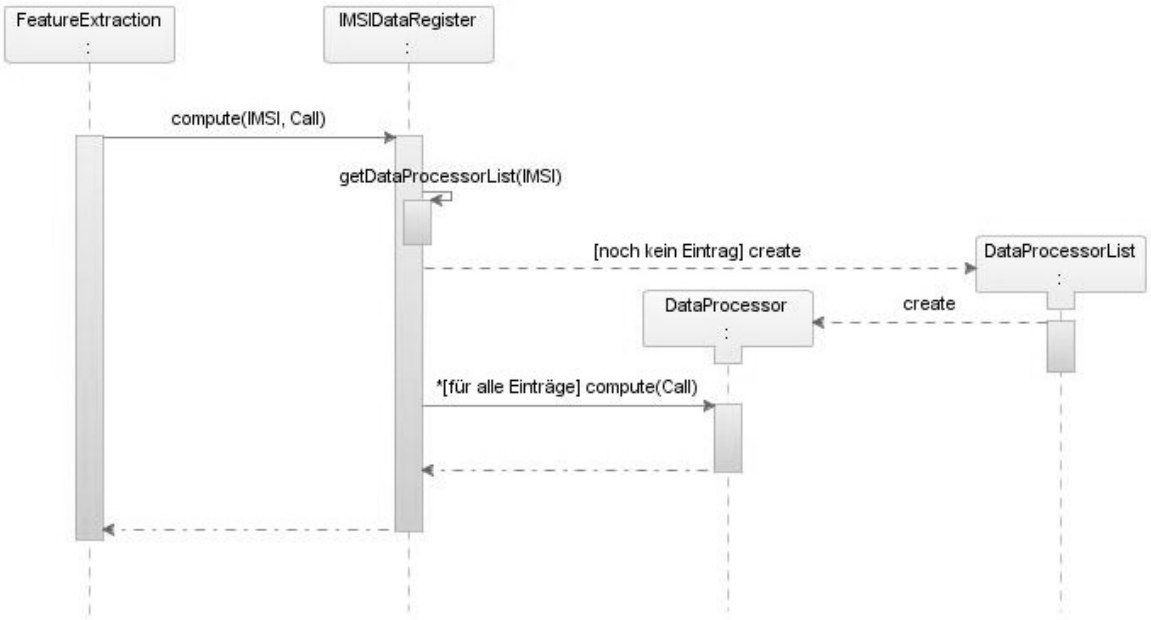

Abbildung 50: Sequenzdiagramm Feature Extraction

Beispiel für den Inhalt eines Call Objekts (zwecks Übersichtlichkeit formatiert) vor Durchlaufen der in DataProcessorList eingetragenen DataProcessor:

```
REFERENCE_TIME : Tue Aug 23 22:35:46 CEST 2005|
CONFORMED_CALLED_NUMBER : 491797899999|
DEST_ID : 86994|
DURATION : 58.0|
IMSI : 262073971099999|
CONFORMED_CALLING_NUMBER : 491799099999|
```

und danach:

```
AVERAGE_INTERNATIONAL_CALL : 0.0|
REFERENCE_TIME : Tue Aug 23 22:35:46 CEST 2005|
AVERAGE_NATIONAL_CALL : 0.0|
IMSI : 262073971099999|
CONFORMED_CALLING_NUMBER : 491799099999|
22-00_INTERVALL : 1.0|
INTERNAL_CALL : 1.0|
AVERAGE_00-06_INTERVALL : 0.0|
AVERAGE_22-00_INTERVALL : 0.03|
AVERAGE_09-18_INTERVALL : 0.0|
AVERAGE_06-09_INTERVALL : 0.0|
AVERAGE_18-22_INTERVALL : 0.0|
DEST_ID : 86994|
CONFORMED_CALLED_NUMBER : 491797899999|
DURATION : 0.14261884904086738|
AVERAGE_OTHER_MOBILE_CALL : 0.0|
AVERAGE_INTERNAL_CALL : 0.03|
AVERAGE_PREMIUM_CALL : 0.0|
```

6.3.2.3 Training Switch

Die Komponente Training Switch dient dazu um den Workflow zwischen Trainingsmodus bzw. Pfad (betreffend der selbst organisierenden Karte) und Analysemodus (indem nur noch die durch das Training bestimmten Referenzvektoren zum Einsatz kommen) umzuschalten. Die Festlegung auf einen der beiden Pfade erfolgt durch Angabe des Parameters ‚-t' für den Trainingsmodus beim Start der Applikation. Da dieser Schalter eine Entscheidung anhand der Existenz des Parameters darstellt erweitert er die Klasse AbstractRule.

Klassendiagramm (Abbildung 51):

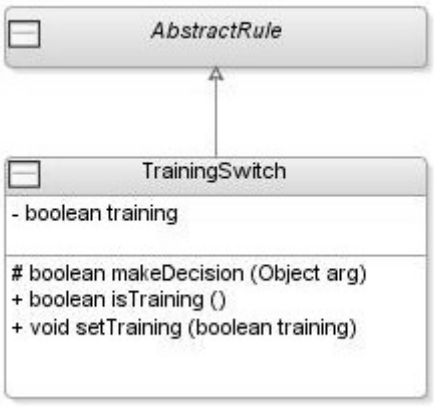

Abbildung 51: Klassendiagramm TrainingSwitch

6.3.2.4 SOM

Die SOM Komponente des Workflows stellt im Trainingsmodus die zentrale Komponente dar. Ihre Aufgabe besteht darin Struktur in den Daten zu erkennen und einzelnen Referenzvektoren zuzuordnen. Um das Ergebnis auch nach der Trainingsphase bzw. zwischen mehreren Durchläufen im Analysemodus zur Verfügung zu haben wird die Gewichtsmatrix dauerhaft gespeichert. Sobald die Referenzvektoren ermittelt und gesichert wurden können in der nächsten Phase Kunden Profile erstellt und aktualisiert werden. Herzstück dieser Komponente ist eine selbst organisierende Karte die unter Zuhilfenahme der Bibliotheken und des grafischen Modellierungstools von JOONE (Abbildung 52) erstellt wurde.

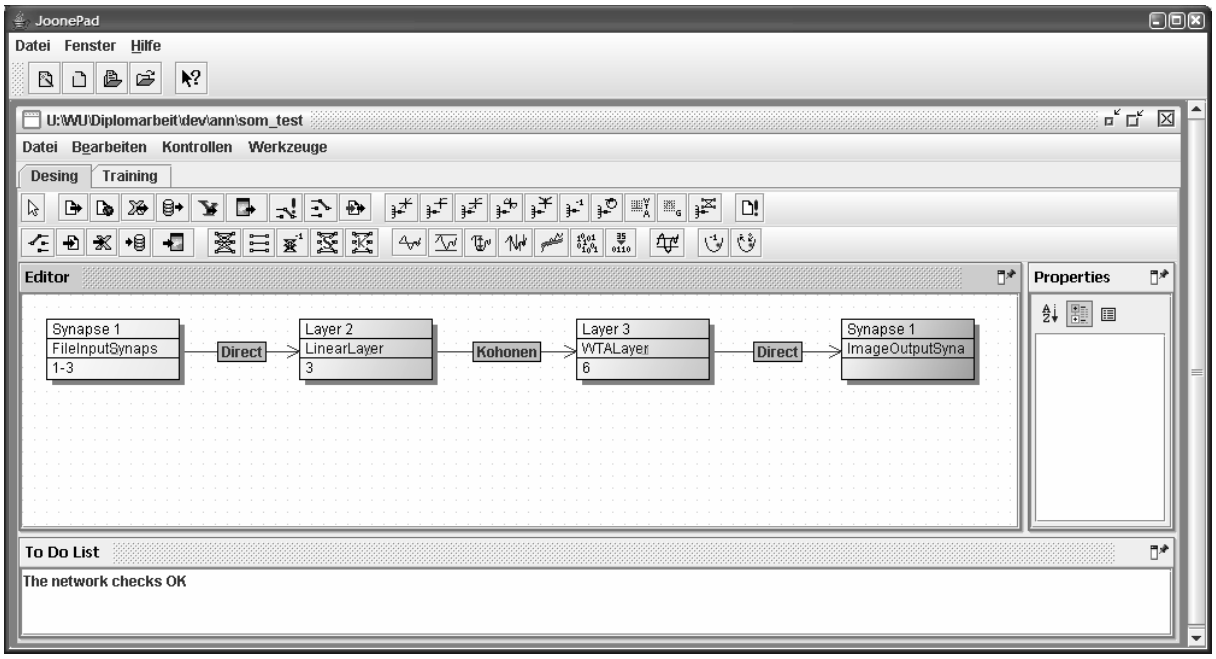

Abbildung 52: Grafische Erstellung des künstlichen neuronalen Netz

Kurzbeschreibung der Klassen:

- SOM: Stellt die Schnittstelle zum Workflow dar, übernimmt die durch die in der Komponente „Feature Extraction“ erweiterten Call Objekte, extrahiert die gewünschten Features und übergibt diese als Trainingsdaten an das künstliche neuronale Netz.

- CommSyncPoint: Diese Klasse entspricht einem zentralen Kommunikations- und Synchronisationspunkt zwischen Workflow und selbst organisierender Karte. Hauptgrund für diesen Adapter ist dass die Kontrolle des Datenflusses im Workflow und dem mittels JOONE implementierten neuronalen Netz zwei verschiedenen Konzepten folgt. Während eine Komponente des Workflows ein Call Objekt übernimmt, bearbeitet und weiterreicht geht JOONE davon aus, sich Daten je nach Bedarf selbst zu holen. Aus diesem Grund wurde ein Mechanismus benötigt der das daraus resultierende Problem löst.

  Um dieses Problem zu lösen wurden die durch JAVA zur Verfügung stehenden Synchronisationsmittel genutzt wobei sich das neuronale Netz einen Datensatz erst dann mittels der Methode „get“ holen kann wenn der Workflow einen mittels „put“

bereitgestellt hat. Weiters blockiert die „put“ Methode solange bis der Datensatz gelesen wurde und die Methode „get“ bis ein neuer Datensatz zur Verfügung gestellt wurde.

```
...

public void put(double[] obj) {
  synchronized (this) {
    this.obj = obj;

    notify();

    try { wait(); } catch (InterruptedException e) { e.printStackTrace(); }
  }
}

public double[] get() {
  synchronized (this) {
    if (obj == null)
      try { wait(); } catch (InterruptedException e) { e.printStackTrace(); }

    double[] myObj = obj;

    obj = null;

    notify();

    return myObj;
  }
}


...
```

Sequenzdiagramm (Abbildung 53):

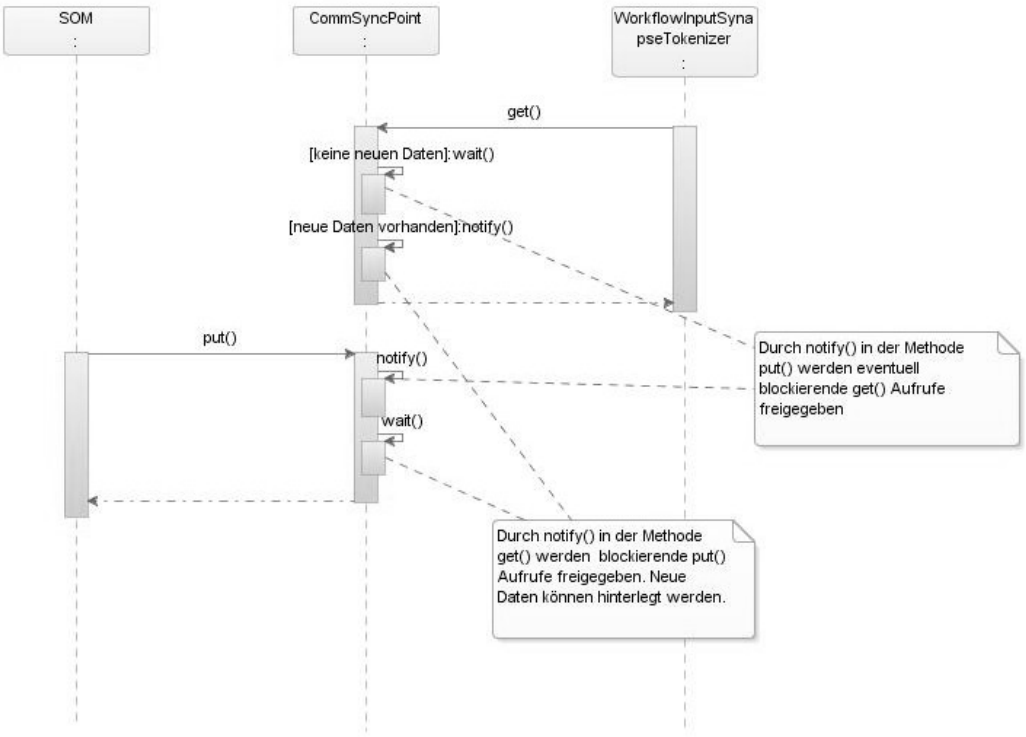

Abbildung 53: Sequenzdiagramm CommSyncPoint

- SomAnn: Repräsentiert das mittels JOONE implementierte künstliche neuronale Netz, die aus einer linearen Input-Schicht zur 1:1 Weitergabe der Input-Daten einer Kohonen-Schicht und einer WTA („Winner Takes All“) Schicht bestehende selbst organisierenden Karte.

  Eigenschaften der selbst organisierenden Karte:

Eigenschaft	***Wert***	
Anzahl der Inputneuronen	9	konfigurierbar
Anzahl der Kartenneuronen	18	konfigurierbar
Lernrate	0,7	konfigurierbar
Momentum	0,01	konfigurierbar
Epochen	10	konfigurierbar
Anzahl der Trainingsmuster	170000	konfigurierbar

Tabelle 6: Eigenschaft der selbst organisierenden Karte

  Die in Tabelle 6 angeführten Werte wurden empirisch bestimmt.

- WorkflowInputSynapse: Erweitert die Klasse org.joone.io.StreamInputSynapse und ermöglicht dem künstlichen neuronalen Netz Daten einzulesen.

- WorkflowInputSynapseTokenizer: Diese Klasse stellt die Verbindung zu dem oben beschrieben Kommunikations- und Synchronisationspunkt, dem eigentlichen Adapter zwischen Workflow und Netz, dar. Um diese Klasse auch im Zusammenhang mit der

Klasse org.joone.io.StreamInputSynapse zu nutzen wird das Interface org.joone.io.PatternTokenizer implementiert.

- SomProperties: Dient dazu um die Einstellungen der selbst organisierenden Karte zu Kapseln.

Klassendiagramm (Abbildung 54):

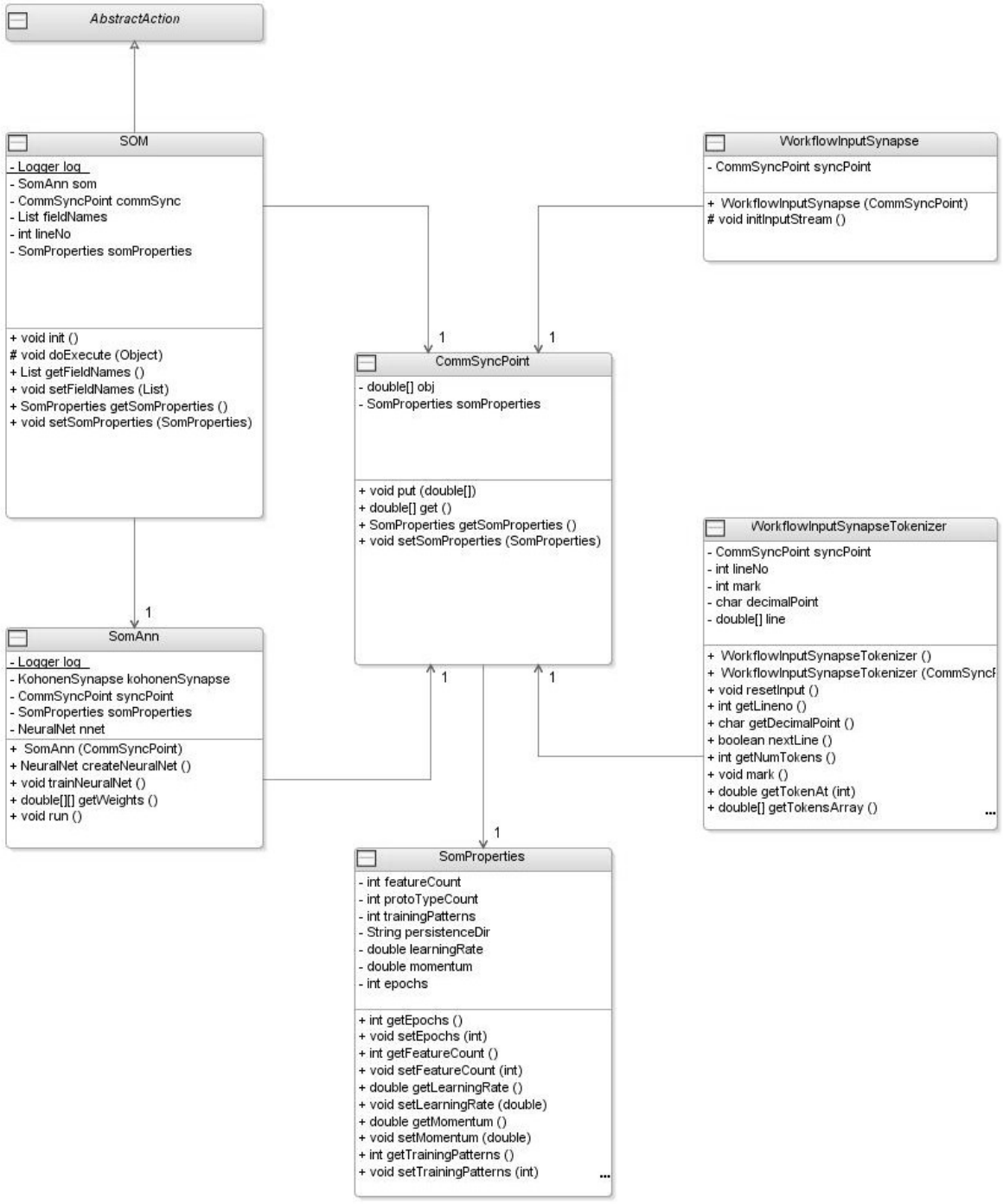

Abbildung 54: Klassendiagramm Self Organizing Map Komponente

6.3.2.5 Profiling & Alarming

Die Aufgabe dieser beiden Komponenten besteht in der Erstellung, Aktualisierung und Sicherung der Benutzerprofile und dem Vergleich zwischen aktuellem und historischem Profil und gegebenenfalls einer Alarmierung von Abnormalitäten.

Kurzbeschreibung der Klassen:

- Profiling: Erhält diese Workflow-Komponente (ersichtlich durch die Vererbungsstruktur im Zusammenhang mit der abstrakten Klasse AbstractAction) ein Call Objekt so werden zuerst die relevanten Inputdaten extrahiert, danach ein Featurevektor erstellt, der die Abweichung von den Prototypen kodiert, und das aktuelle Profil modifiziert. Nach einer konfigurierbaren Anzahl von Aktualisierungen des aktuellen Profils wird das historische Profil modifiziert.

- ProfileRegister: Diese Klasse dient dazu für jeden Kunden bzw. jede IMSI ein ProfileContainer Objekt zu halten. Da dieses Register sowohl von der Profiling als auch von der Alarming Komponente genutzt wird und bei einer hohen Anzahl an Benutzern einen hohen Speicherbedarf darstellt muss wie durch die Anwendung des Singleton Design-Pattern dafür gesorgt werden, dass von dieser Klasse jedenfalls nur ein Objekt existiert. Durch die Nutzung des Spring Application Frameworks muss dieser Designpattern nicht selbst umgesetzt werden sondern es wird standardmäßig nur eine Instanz pro konfigurierten Bean instanziert.

- ProfileContainer: Kapselt neben aktuellem und historischem Profil auch einen Zähler der zur Bestimmung des Änderungszeitpunkts des historischen Profils genutzt wird. Jedem Benutzer ist ein Container zugeordnet der wie oben beschrieben im ProfileRegister abgelegt ist.

- Alarming: Diese Workflow-Komponente berechnet nach jedem Call Objekt die Abweichung zwischen aktuellem und historischem Profil für den Anrufauslösenden Benutzer (identifiziert durch die IMSI).

Klassendiagramm (Abbildung 55):

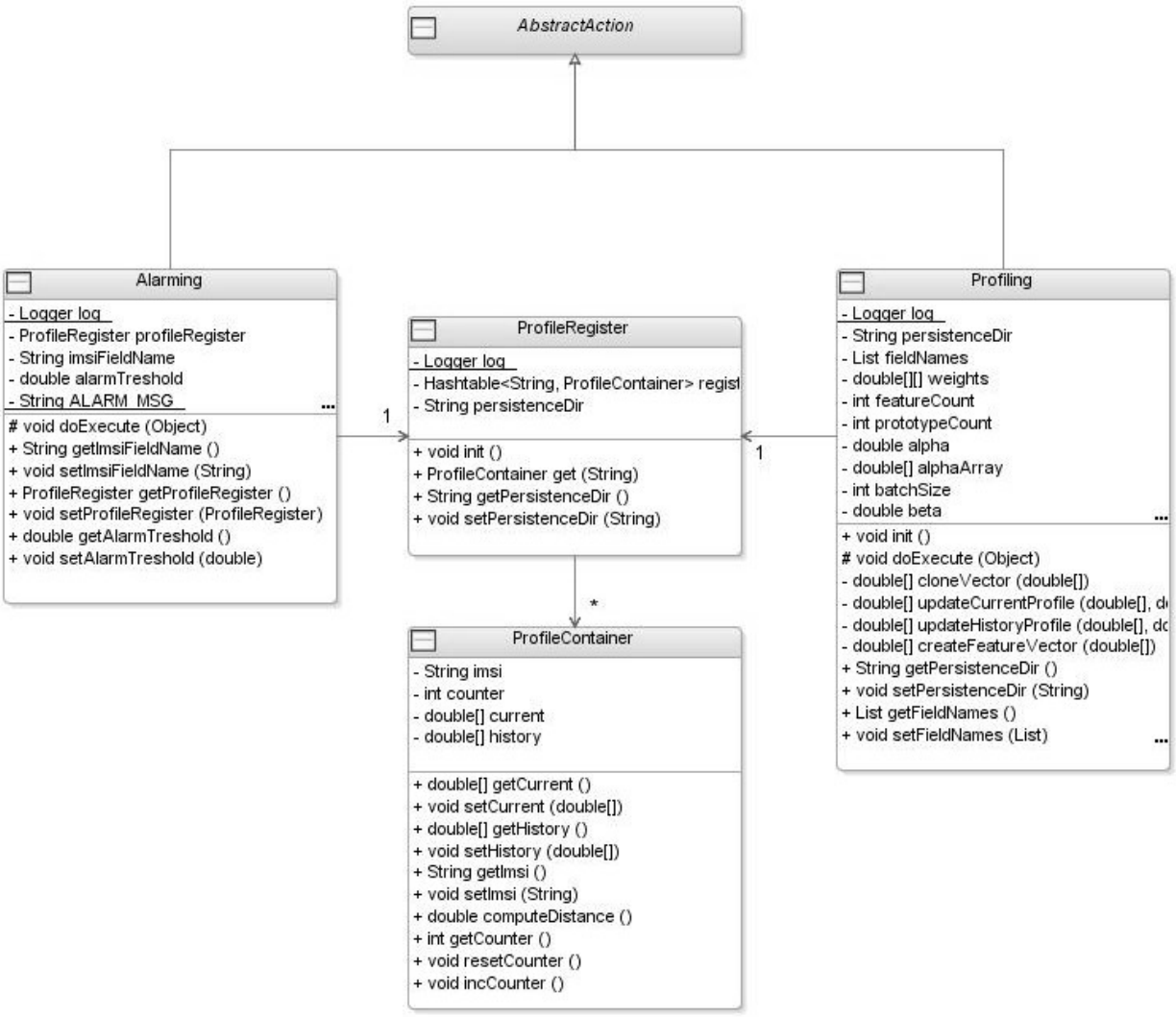

Abbildung 55: Klassendiagramm Profiling & Alarming

Sequenzdiagramm (Abbildung 56):

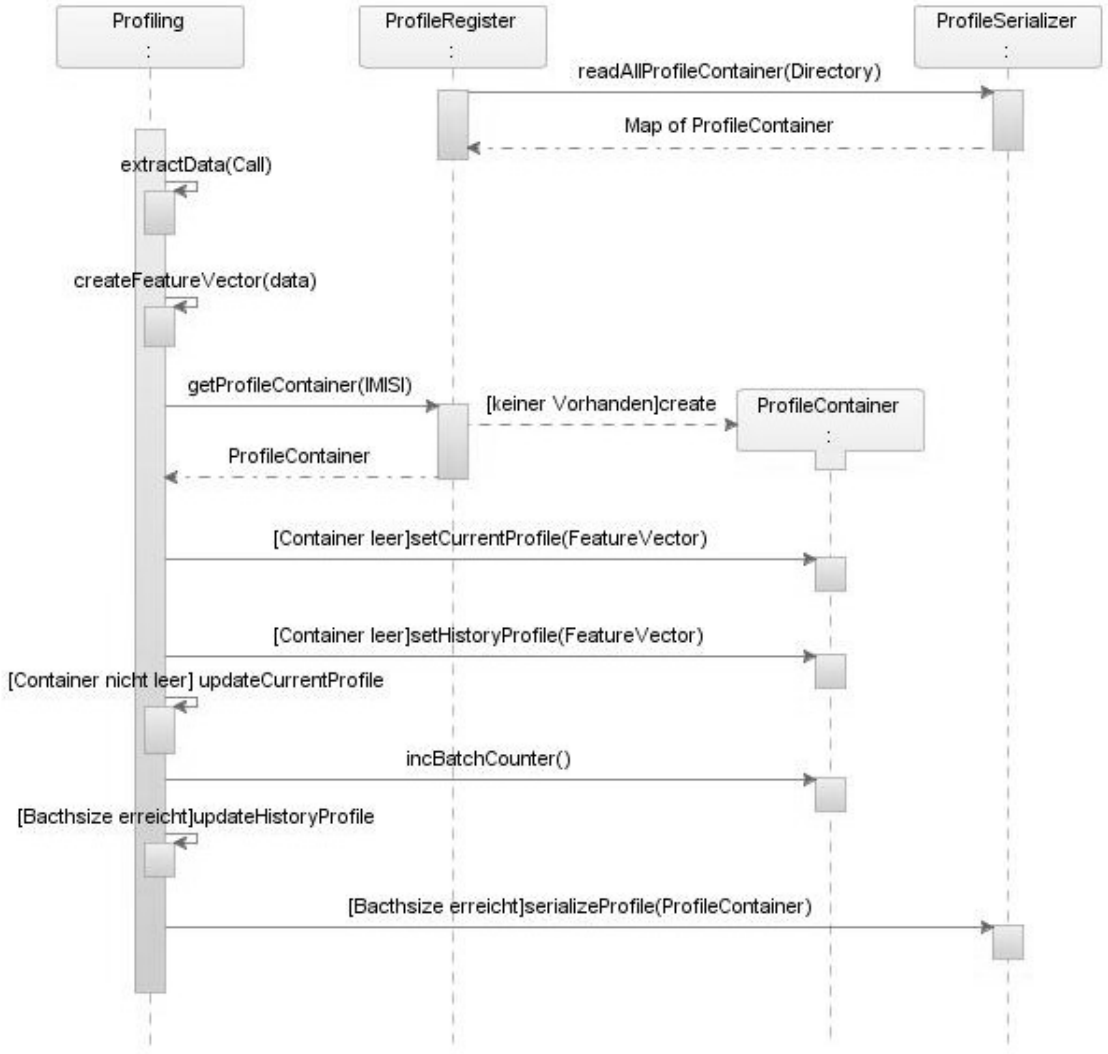

Abbildung 56: Sequenzdiagramm Profiling.

6.3.2.6 AlarmManager

Der AlarmManager ist eine vom Workflow unabhängige Komponente die jeder Klasse die von AbstractAction erbt als Attribut mitgegeben werden kann. Dadurch wird eine zentrale Stelle geschaffen die alle Alarme sammelt und diese an verschiedene Kanäle (Datei, Email …) verteilt. Um Alarme mittels verschiedener Mechanismen (konfigurierbar) zu verteilen wird Log4j des Apache Jakarta Projekts eingesetzt. Die Existenz von genau einem AlarmManager (falls nur einmal konfiguriert) wird durch das „Spring Application Framework“ sichergestellt welches standardmäßig nur eine einzige Instanz pro konfigurierten Bean erstellt (Singleton).

Um einen Alarm auszulösen bzw. dem AlarmManager zu melden muss die Aktion zuerst ein Alarm Objekt generieren welches die wichtigsten Eckdaten, wie z.B. Severity, Beschreibung und Datum eines Alarms kapselt.

Der Alarmmechanismus setzt sich aus den beiden folgenden Klassen zusammen:

- AlarmManager: Zentrale Stelle die Alarme empfängt, diese verwaltet und über verschiedene Mechanismen verteilt.

- Alarm: Stellt einen Alarm mit seinen wichtigsten Attributen dar.

Klassendiagramm (Abbildung 57):

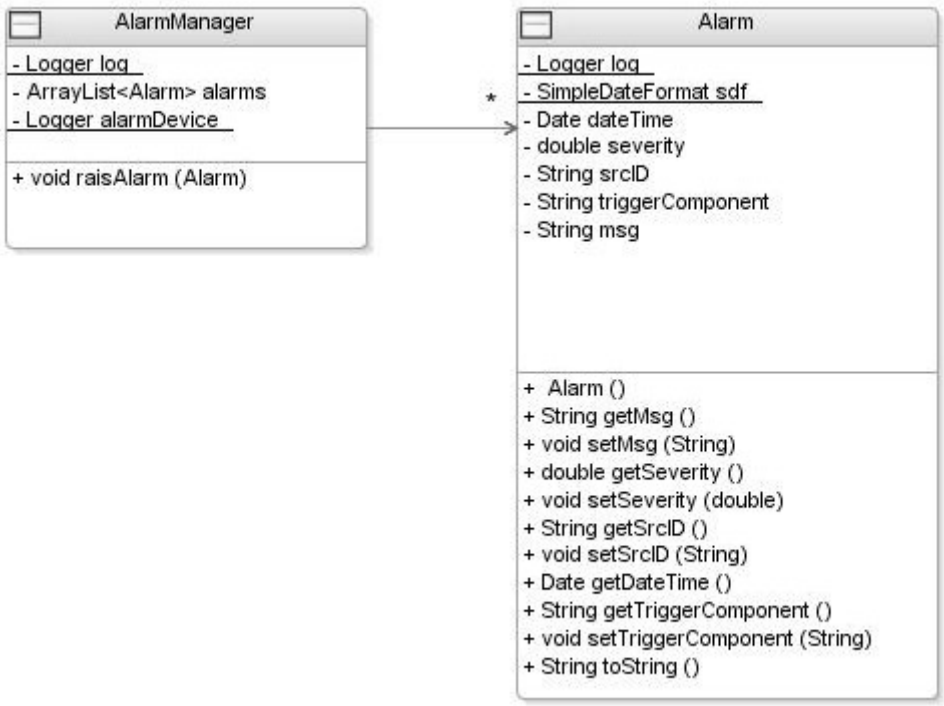

Abbildung 57: Klassendiagramm AlarmManager und Alarm

Sequenzdiagramm (Abbildung 58):

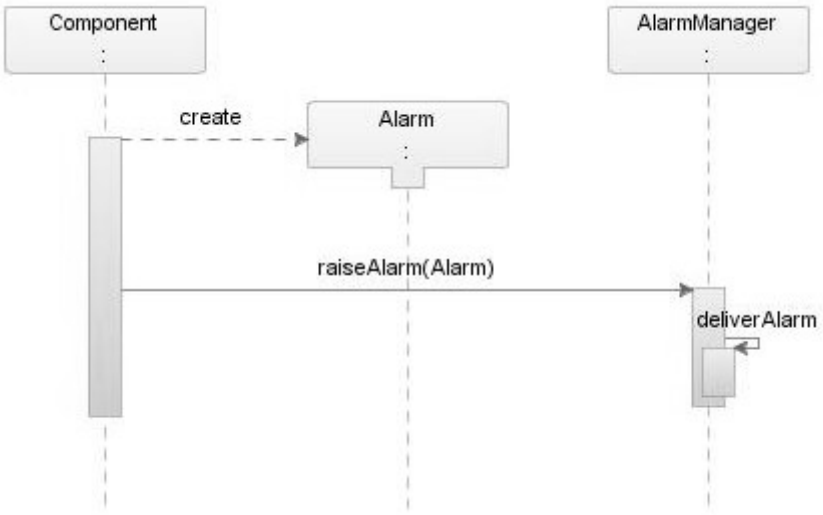

Abbildung 58: Sequenzdiagramm Alarming

6.3.2.7 Utility Klassen

Durch die Utility Klassen werden einige allgemeine, öfters verwendete Funktionen implementiert.

- MathExt: Erweitert die mathematischen Standardfunktionen von Java um Methoden
  - zur Berechnung der euklidischen Distanz,
  - zur Transposition einer Matrix und
  - der Berechnung der Hellinger Distanz.

- WeightSerializer: Klasse zur Serialisierung (schreiben/lesen) der Gewichts-Matrix der selbst organisierenden Karte. Das Verzeichnis kann mittels Methodenparameter angegeben werden und der Namen der im Persistenzverzeichnis angelegten Datei lautet *weights.tmp*.

- ProfileSerializer: Klasse zur Serialisierung (schreiben/lesen) von Profilcontainern (beinhalten das aktuelle und historische Profil eines Benutzers). Das Verzeichnis kann mittels Methodenparameter angegeben werden und der Namen der im Persistenzverzeichnis angelegten Dateien folgt dabei dem folgendem Format: *<IMSI>_profile.tmp*.

– BuildClasspath: Erstellt je nach Betriebssystem beim erstmaligen Start der Applikation eine ausführbare Datei welche an Hand eines Directorylistings die Umgebungsvariable CLASSPATH erweitert.

Klassendiagramm (Abbildung 59):

Abbildung 59: Klassendiagramm Utility Klassen

6.4 Installation/Konfiguration/Bedienung

6.4.1 Installation

Folgende Aufgaben müssen bei einer Installation der ProfileMonitor-Software auf einem Windows Betriebssystem durchgeführt werden.

Pre-Installationsaufgaben

1. Systemvoraussetzungen prüfen.
    a. Hardware: Minimalanforderungen Pentium 3; 300 MHz; 512MB RAM; 4MB an verfügbaren Harddisk-Speicher.
    b. Software: Java Virtual Machine JDK 5.0 Update 5

2. Stellen sie sicher das der Pfad zu der ausführbaren Datei *‚java.exe'* in ihrer *PATH* Umgebungsvariable eingetragen ist.

Installation

3. Entpacken sie die Datei *pm.zip* in einem Verzeichnis ihrer Wahl (dieses Verzeichnis wird im Folgenden als HOME Verzeichnis bezeichnet)

Post-Installationsaufgaben

4. Nehmen sie die Konfiguration des Systems durch bearbeiten der Datei *workflow.xml* im Verzeichnis *HOME\conf* vor.

6.4.2 Konfiguration

Für den Betrieb der Applikation ist das Vorliegen der im Folgenden beschriebenen Konfigurationsdateien notwendig.

workflow.xml

Diese Datei beinhaltet die Modellierung des Workflows in Form von XML Tags die in der „document type definition“ des Spring Application Frameworks definiert wurden. Jede Komponente des Workflows wird durch einen eigenen „bean“ Tag abgebildet.

Als Beispiel wird die Konfiguration einer einfachen Sequenz von 3 Aktionen angeführt. Für jede Komponente wird der Nachfolger als „property“ mit dem Namen „nextStep“ konfiguriert.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans SYSTEM "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
   <!–
      Importiert eine zusätzliche XML Datei welches der gleichen dtd folgt.
    -->
    <import resource="additional.xml"/>

    <!–
      Der Prozessor startet den Workflow in dem die erste Komponente des Workflows
      ausgeführt wird.
    -->
    <bean id="PROCESSOR" class="profiler.ProfileMonitor">
      <property name="firstStep" ref="COMPONENT_1"/>
    </bean>

    <!--
      **********************
      *** Workflow Anfang ***
      **********************
    -->

    <!–
      Die erste Komponente des Workflows mit Konfiguration ihres Nachfolgers (der
      Bean mit der ID ‚COMPONENT_2’.
      Eine Besonderheit stellt hier die Definition, durch das Attribut ‚init-
      methode’, einer Methode zur Initialisierung der Aktion auf. Dies hat zur
      Folge, dass im Beispiel nach setzen aller Properties die Methode init() des
      Objekts aufgerufen wird.
    -->
    <bean id="COMPONENT_1" class="profiler.component.dummy.DummyComponent"
       init-method="init">
      <property name="nextStep" ref="COMPONENT_2"/>
    </bean>

    <bean id="COMPONENT_2" class="profiler.component.dummy.DummyComponent" >
      <property name="nextStep" ref="COMPONENT_3"/>
    </bean>

    <bean id="COMPONENT_3" class="profiler.component.dummy.DummyComponent" />
</beans>
```

additional.xml

Enthält die Konfiguration die zusätzlich zum Workflow notwendig ist, beispielsweise die Konfiguration einer DataSource die eine Verbindung zu einer Datenbank darstellt.

log4j.properties

Enthält die Konfiguration des Log-Subsystems. Durch Änderungen in dieser Datei kann beispielsweise der Log-Level (DEBUG, INFO, WARN, ERROR und FATAL) oder das Ziel der Log-Nachrichten auf eine Datei geändert werden.

Beispiel:

```
# Der Rootlogger wird so konfiguriert, dass
# der Level ERROR ist und
# als Appender der ConsoleAppender verwendet wird.
log4j.rootCategory=ERROR, C

# C ist ein Appender der auf die Console schreibt.
log4j.appender.C=org.apache.log4j.ConsoleAppender
log4j.appender.C.layout=org.apache.log4j.PatternLayout
log4j.appender.C.layout.ConversionPattern=%p %d %c - %m%n

# RF ist ein Appender der in eine Datei schreibt.
#log4j.appender.RF=org.apache.log4j.RollingFileAppender
#log4j.appender.RF.File=example.log
#log4j.appender.RF.MaxFileSize=5120KB
#log4j.appender.RF.MaxBackupIndex=4
#log4j.appender.RF.layout=org.apache.log4j.PatternLayout
#log4j.appender.RF.layout.ConversionPattern=%p %d - %m%n

log4j.category.profiler=DEBUG
log4j.category.ALARM=INFO
```

6.4.3 Start/Stop

Nach der erfolgten Installation befindet sich im *HOME* Verzeichnis ein Unterverzeichnis Namens ‚*bin*' in dem sich ein Skript befindet um die Software im gewünschten Modus zu starten. Im Folgenden wird der Start der Applikation im gewünschten Modus auf Grund je eines Beispiels beschrieben.

Beispiel:

1. *Erstellen der Prototypen*
   Start der Software im Trainingsmodus: Navigieren sie dazu in einer Eingabeaufforderung in das Verzeichnis *HOME\bin* und führen sie die Datei *start.bat* mit den Parametern *Quelldatei* und *–t* aus. Sind alle Systemvorrausetzungen (siehe Kapitel Installation) erfüllt werden neue Prototypen erstellt und im Persistenzverzeichnis in der Datei *weights.tmp* abgespeichert.

   ```
   d:\profiler\> start.bat anruf_daten.csv -t
   ```

2. *Bewerten des Benutzerverhaltens*
   Start der Software im Analysemodus: Navigieren sie dazu in einer Eingabeaufforderung in das Verzeichnis *HOME\bin* und führen sie die Datei *start.bat* mit den Parameter *Quelldatei* aus. Sind alle Systemvorrausetzungen (siehe Kapitel Installation) erfüllt werden für die in der angegebenen Quelldatei enthaltenen Benutzer Profile erstellt, im Persistenzverzeichnis gesichert und für jeden Datensatz (Anruf) eine Bewertung der Änderung des Verhaltens durchgeführt.

   ```
   d:\profiler\> start.bat anruf_daten.csv
   ```

6.4.4 Log-Nachrichten

Der Status in dem sich die Applikation aktuell befindet ist aus den Log-Nachrichten in der Eingabeaufforderung ersichtlich.

Beispiel Analysemodus:

```
INFO 2006-07-17 12:27:00,976 profiler.util.WeightsSerializer - Weights loaded from
U:\WU\Diplomarbeit\dev\data\weights.tmp

DEBUG 2006-07-17 12:27:01,008 profiler.component.profiling.Profiling - Loaded transposed
weights [[1.0231015652670052, 1.1150762478619494, 1.011889624177964, 1.3297558524245507,
8.730152492516991, 1.0, 1.297305169051625, 1.2973174238081073, 8.449927564344673],
[1.0016858973294818, 1.0178441013327164, 1.2525325466552886, 1.1433600027442616,
1.0000003491463987, 8.445753635128115, 1.1109085489358026, 3.221124573405408,
6.329954097168703], [1.0008634310869393, 1.0407533490368461, 1.3456280619551202,
1.143812955791143, 4.12731837308266, 1.0991408138556904, 1.0011917165944773,
1.0343158739622709, 4.680392614127865], [1.0035625690954835, 1.0135341377917246,
4.994454862729971, 2.622537078672578, 1.1056944461291422, 1.0324507303464086,
1.0034794044997288, 1.007039272643158, 6.7341779278746765], [1.013901063018681,
1.0051747308360537, 1.4371512694487352, 1.0010773309379581, 1.1456620568138995,
1.000000100884154, 1.0, 1.0000230911818804, 1.5838545686532228], [1.000815561524594,
1.1935483803995475, 3.2406023088161087, 3.44807912048266, 4.259458835288521,
1.0002507423184832, 4.159191419853083, 1.0000000005668765, 4.567981670082967],
[1.000008469159433, 1.0000464589456066, 1.0118843250064833, 6.985342777511051,
1.4025517815360187, 1.0000000007424854, 1.000000150584125, 1.0035492222730176,
6.581981377428495], [1.00439289992419, 1.1906557065466907, 1.5751812678525248,
4.196190951921797, 4.173901162310823, 1.0003730679957004, 1.032263076646742,
1.0000800020490144, 7.902824959632461]]

INFO 2006-07-17 12:27:01,008 profiler.component.dataprovider.CSVFile - Processing file
U:/WU/Diplomarbeit/dev/data/test/moc_data_232ALOIS1_TEST1.txt

DEBUG 2006-07-17 12:27:01,086 profiler.component.profiling.Profiling - FeatureVector
[0.7125851673445538, 0.14006723679774213, 0.0019644555859974462, 0.019378810320760703,
5.260917649861827E-5, 0.0132046052680617, 0.056454034852275424, 0.05629308065411022]

DEBUG 2006-07-17 12:27:01,086 profiler.component.profiling.Alarming - Distance (232ALOIS1) >>
0.0010436303307551611

…

DEBUG 2006-07-17 12:27:01,133 profiler.component.profiling.Profiling - FeatureVector
[0.7658688024652103, 0.006413514179973033, 0.006685355042966976, 0.03200162129915438,
0.002952017940465857, 0.02366910158839104, 0.09235321507567926, 0.07005637240815911]

DEBUG 2006-07-17 12:27:01,133 profiler.component.profiling.Alarming - Distance (232ALOIS1) >>
0.12657289025538337

DEBUG 2006-07-17 12:27:01,164 profiler.component.profiling.Profiling - FeatureVector
[0.7048146990028308, 0.0010520541875708729, 0.01365828884855023, 0.039206257527181695,
0.014209797369595292, 0.044370760669178604, 0.11369335722501654, 0.06899478517007605]

DEBUG 2006-07-17 12:27:01,164 profiler.component.profiling.Alarming - Distance (232ALOIS1) >>
0.1597579718544151

INFO 2006-07-17 12:27:01,164 ALARM - Date: 20060717 12:27:01|Severity:
0.1597579718544151|Trigger: Profile Monitor Application - Alarming|Source ID:
232ALOIS1|Message: Profile abnormality

DEBUG 2006-07-17 12:27:01,164 profiler.component.profiling.Profiling - FeatureVector
[0.6812064738222356, 8.381214378982337E-4, 0.016894866591250765, 0.044675965816668654,
0.017601399179055146, 0.041719067929359616, 0.12148313171224483, 0.07558097351128726]

DEBUG 2006-07-17 12:27:01,180 profiler.component.profiling.Alarming - Distance (232ALOIS1) >>
0.170193413844897

INFO 2006-07-17 12:27:01,180 ALARM - Date: 20060717 12:27:01|Severity:
0.170193413844897|Trigger: Profile Monitor Application - Alarming|Source ID:
232ALOIS1|Message: Profile abnormality

DEBUG 2006-07-17 12:27:01,180 profiler.component.profiling.Profiling - FeatureVector
[0.6568796607879306, 4.857189399727916E-4, 0.01961426441940214, 0.049012191830699346,
0.020551961890877393, 0.04595582860954121, 0.1268820075782761, 0.08061836594330031]
```

```
DEBUG 2006-07-17 12:27:01,180 profiler.component.profiling.Alarming - Distance (232ALOIS1) >>
0.1825846612020018

INFO 2006-07-17 12:27:01,180 ALARM - Date: 20060717 12:27:01|Severity:
0.1825846612020018|Trigger: Profile Monitor Application - Alarming|Source ID:
232ALOIS1|Message: Profile abnormality

…

DEBUG 2006-07-17 12:22:22,083 profiler.component.profiling.Profiling - FeatureVector
[0.5938240236481501, 4.054570699258198E-4, 0.02901861793707461, 0.06006237099952089,
0.03624752520376452, 0.05033503298341516, 0.1407349978991154, 0.08937197425903355]

DEBUG 2006-07-17 12:22:22,083 profiler.component.profiling.Alarming - Distance (232ALOIS1) >>
5.769347297413687E-4

DEBUG 2006-07-17 12:22:22,083 profiler.component.profiling.Profiling - FeatureVector
[0.5938099602585076, 4.0563011638947605E-4, 0.029020769806956576, 0.060065684153879424,
0.03625023720916343, 0.05033592500132302, 0.14073958175451354, 0.08937221169926697]

INFO 2006-07-17 12:22:22,099 profiler.util.ProfileSerializer - ProfileContainer serialized to
U:\WU\Diplomarbeit\dev\data\232ALOIS1_profile.tmp

DEBUG 2006-07-17 12:22:22,099 profiler.component.profiling.Alarming - Distance (232ALOIS1) >>
6.597389670958518E-5

DEBUG 2006-07-17 12:22:22,130 profiler.component.profiling.Profiling - FeatureVector
[0.5940904636362767, 4.023077538804047E-4, 0.02897884076811719, 0.06000218253094996,
0.03619972797420542, 0.05031423550598951, 0.14065104037260873, 0.08936120145797206]

INFO 2006-07-17 12:22:22,224 profiler.util.ProfileSerializer - ProfileContainer serialized to
U:\WU\Diplomarbeit\dev\data\232ALOIS1_profile.tmp

DEBUG 2006-07-17 12:22:22,224 profiler.component.profiling.Alarming - Distance (232ALOIS1) >>
9.807196072346672E-6

INFO 2006-07-17 12:22:22,224 profiler.component.dataprovider.CSVFile - Processing of file
U:/WU/Diplomarbeit/dev/data/test/moc_data_232ALOIS1_TEST1.txt done. (1000 records)
```

Beispiel Trainingsmodus:

```
DEBUG 2006-07-17 12:07:31,540 profiler.component.profiling.Profiling - Loaded transposed
weights null

INFO 2006-07-17 12:07:31,540 profiler.component.dataprovider.CSVFile - Processing file
U:/WU/Diplomarbeit/dev/data/moc_data_echt.txt

DEBUG 2006-07-17 12:07:31,790 profiler.component.som.joone.SomAnn - Before training [[-
0.04648472571137924, -0.11862028274148076, -0.04632041044166807, 0.04202899084316991, -
0.004079507887864153, -0.022462444590914787, 0.1553869206633436, -0.04257071809021537],
[0.058998516556640734, -0.19153695815363858, 0.10430814140034894, 0.007333623377417553,
0.05048959948120574, 0.07472774460291598, -0.19194518304693875, 0.17086302520835112],
[0.06812132901435042, 0.07991341543158959, 0.001779963309012006, -6.664677303295785E-4,
0.10230827136414972, 1.1387664422088384E-4, -0.054161347159708134, -0.04324177213924352],
[0.04907689940993437, -0.09614910701731293, -0.12352215520596982, -0.1166965276227478, -
0.0041255113431536705, 0.007816471261913593, 0.13412831264854708, 0.06808842715067998],
[0.007373840157042233, 0.021432412808186252, -0.010425717197249107, -0.03345474360837247, -
0.1346612753311317, 0.06329590299307042, -0.053519849427128646, -0.19111467697332687],
[0.006068160463407468, -0.07484185015262557, -0.1805439186486702, 0.02004495999445105,
0.05089068712902228, -0.18583883740161733, -0.02995205956746183, 0.10781248869404697],
[0.03963296557117499, 0.12163586386049358, -0.07126014848406101, 0.07466623169180736,
0.15394011154064785, -0.15315707831046238, 0.03926092950869409, -0.12981856522099525],
[0.15677840678781452, -0.15338422095043758, 0.1321907600017183, -0.07530779153248801, -
0.023596060424251913, 0.08694335439821199, 0.16806146463900334, -0.008777465088587733], [-
0.17246579517918104, -0.05985156819683128, 0.08585712633679643, -0.02847133494378634, -
0.01058759753846314, 0.10107517140647715, 0.15101469599424533, -0.08657660641171061]]

INFO 2006-07-17 12:10:53,115 profiler.component.som.SOM - Feeding of self organizing map done.
(170200 records)

INFO 2006-07-17 12:10:53,115 profiler.component.dataprovider.CSVFile - Processing of file
U:/WU/Diplomarbeit/dev/data/moc_data_echt.txt done. (170200 records)

DEBUG 2006-07-17 12:12:18,774 profiler.component.som.joone.SomAnn - After training
[[1.0231015652670052, 1.0016858973294818, 1.0008634310869393, 1.0035625690954835,
1.013901063018681, 1.000815561524594, 1.000008469159433, 1.00439289992419],
[1.1150762478619494, 1.0178441013327164, 1.0407533490368461, 1.0135341377917246,
```

```
1.0051747308360537, 1.1935483803995475, 1.0000464589456066, 1.1906557065466907],
[1.011889624177964, 1.2525325466552886, 1.3456280619551202, 4.994454862729971,
1.4371512694487352, 3.2406023088161087, 1.0118843250064833, 1.5751812678525248],
[1.3297558524245507, 1.1433600027442616, 1.143812955791143, 2.622537078672578,
1.0010773309379581, 3.44807912048266, 6.985342777511051, 4.196190951921797],
[8.730152492516991, 1.0000003491463987, 4.12731837308266, 1.1056944461291422,
1.1456620568138995, 4.259458835288521, 1.4025517815360187, 4.173901162310823], [1.0,
8.445753635128115, 1.0991408138556904, 1.0324507303464086, 1.000000100884154,
1.0002507423184832, 1.0000000007424854, 1.0003730679957004], [1.297305169051625,
1.1109085489358026, 1.0011917165944773, 1.0034794044997288, 1.0, 4.159191419853083,
1.000000150584125, 1.032263076646742], [1.2973174238081073, 3.221124573405408,
1.0343158739622709, 1.007039272643158, 1.0000230911818804, 1.0000000005668765,
1.0035492222730176, 1.0000800020490144], [8.449927564344673, 6.329954097168703,
4.680392614127865, 6.7341779278746765, 1.5838545686532228, 4.567981670082967,
6.581981377428495, 7.902824959632461]]

INFO 2006-07-17 12:12:18,837 profiler.util.WeightsSerializer - Weights serialized to
U:\WU\Diplomarbeit\dev\data\weights.tmp
```

Hinweis: Durch Änderungen in der Datei *HOME\conf\log4j.properties* kann diese Ausgabe z.B. auch in eine Datei erfolgen.

6.5 Analyse & Erkenntnisse

In diesem Abschnitt werden neben der Beschreibung des Vorgehens während der Analyse des Systems auch dadurch gewonnene Erkenntnisse und Ideen für Erweiterungen die bei einem produktiven Einsatz des Systems sinnvoll sind genannt.

6.5.1 Ziel der Analyse

Vorderrangiges Ziel der Analyse war die Anpassung der Alarmierungsschwelle, die einen Kompromiss zwischen Fehlalarmen und einer hohen Rate an Identifizierungen darstellt. Ein System das zu empfindlich ist würde zu viele Fehlalarme erzeugen.

6.5.2 Analyseumgebung

Zur Herstellung der Analyseumgebung wurde zuerst die selbst organisierende Karte mit folgenden Parametern konfiguriert:

Anzahl der Inputneuronen	9
Anzahl der Kartenneuronen	18
Lernrate	0,7
Momentum	0,1
Epochen	10
Anzahl der Trainingsmuster	170200

Tabelle 7: Testkonfiguration der selbst organisierenden Karte

und danach ein Testbenutzer der im Normalfall folgende Eckdaten aufweist simuliert:

IMSI	232ALOIS1
A Nummer	439991234567
Zeitpunkt	10 % zwischen 6 und 9 Uhr 50 % zwischen 9 und 18 Uhr 40 % zwischen 18 und 22 Uhr
Ziel	10 % international 10 % nationale Festnetze 30 % andere Mobilfunknetze 50 % intern
Dauer	30 % zwischen 28 und 35 Sekunden 40 % zwischen 110 und 125 Sekunden 20 % zwischen 290 und 305 Sekunden 10 % zwischen 895 und 905 Sekunden

Tabelle 8: Eckdaten Testbenutzer

6.5.3 Vorgehen

Das Vorgehen während der Analyse folgte weitestgehend dem folgenden Schema:

1. Aktuelle Sicherung des ProfileContainer löschen.

2. Aufruf der Applikation mit einer Datei die Anruferdaten enthält welche das Normalverhalten des Benutzers abbilden um das aktuelle und ein historisches Profil zu erstellen.

3. Aufruf der Applikation mit einer Datei die Anruferdaten enthält die in einem oder mehreren Punkten signifikant vom Normalverhalten abweichen.

4. Erfassen der Ausgabe der Distanzen (an Hand dessen Höhe die Bewertung erfolgt ob ein Alarm ausgelöst wird)

Nach Herstellung der Umgebung und des Trainings der Karte wurden mittels des oben beschriebenen Vorgehens Tests betreffend Änderungen der Charakteristik einer oder mehrer Inputmerkmalen durchgeführt. Beispielsweise wurde bei einem Durchlauf die Verteilung der Anrufsziele des Benutzers so verändert, dass 70 % der Anrufe an ein internationales Ziel gingen und die restlichen 30% zu je 10 % an das nationale Festnetze, andere Mobilfunknetze und interne Ziele verteilt wurden.

6.5.4 Erkenntnisse

Während bzw. durch die Analyse des laufenden Systems ergaben sich die folgenden Erkenntnisse:

– Zu viele bzw. nicht relevante Kartenneuronen/Prototypen erhöhen den Bedarf an Rechenleistung. Die optimale Anzahl an Neuronen wurde durch einen Vergleich der Gewichtsmatrix vor und nach dem Training im Versuch bestimmt. Kartenneuronen die nach dem Training für ihre Gewichte zu den Inputneuronen die gleichen zufälligen Initialisierungswerte wie davor aufwiesen konnten entfernt werden. Dieses Phänomen kann dadurch erklärt werden, dass diese Neuronen bei der Anwendung des

Trainingsalgorithmus nie als Sieger hervor gingen („Winner Takes All“ – Prinzip).

– Enthält die Gewichtsmatrix in einer Zeile (entspricht einem Inputfeature) für jeden Prototypen den Wert 0 so kann daraus geschlossen werden, dass dieses Inputfeature keinen Beitrag zum Ergebnis leistet oder in den Daten nicht vorkommt. Als Beispiel kann hier der Fall genannt werden dass in den ursprünglichen Daten zum Training der selbst organisierenden Karte keine Anrufe zu Premium Nummern (0900 Nummer) zu finden waren.

– Die Manipulation von nur einer Inputkomponente wird durch die bestehende Konfiguration zwar erkannt führt aber nur zu sehr kleinen Abweichungen zwischen aktuellem und historischem Profil.

– Ein zu hoher Betafaktor (Faktor mit dem nach einer konfigurierten Anzahl an Gesprächsdatensätzen das historische Profil angepasst wird) in Kombination mit einer niedrigen Batchgröße führt erwartungsgemäß dazu, dass nach erreichen der Batchgröße und Anpassung des historischen Profils sich die Abweichung um ca. eine zehner Potenz niedriger darstellt.

– Die Funktion welche die Abweichung zu den Prototypen bestimmt – im Prototyp der Applikation $\exp(\|X_{N+1} - Q_j\|)$ - hat einen hohen Einfluss auf das Ergebnis. Sie bestimmt unter anderem ob hohe Abweichungen härter bestraft werden als niedrige. Weiters wird durch diese Funktion das Vorzeichen der Abweichung vernachlässigt.

– Das Skalierungsintervall hat Einfluss auf die Anzahl der gefundenen Prototypen bzw. auf die Anzahl der Kartenneuronen deren Gewichtsvektor in der Trainingsphase angepasst wird. Wird das Intervall zu klein gewählt so wird unter Umständen nur ein einzelner Prototyp gefunden.

6.5.5 Sinnvolle Erweiterungen

Für den andauernden Betrieb im produktiven Netz eines Mobilfunkbetreibers muss dieser Prototyp um verschiedene Eigenschaften erweitert werden. Einige Anregungen dazu sind im Folgenden aufgelistet:

- Aktuell werden Benutzerprofile auf der Harddisk abgespeichert und während des Betriebs eine Kopie davon (von allen) im Hauptspeicher gehalten. Weiters befindet sich für jeden Kunden eine eigene Instanz der DataProcessorList und der darin enthaltenen eigenen MovingAverageDataProcessor im Speicher. Für einen mittleren Mobilfunkbetreiber mit einer Kundenanzahl von 2 Million ergibt dies einen sehr hohen Hauptspeicherbedarf. Um dieses „Problem“ zu umgehen könnte die Strategie verfolgt werden nur eine bestimmte Anzahl an Profilen, die der gerade aktiven Teilnehmer, im Speicher zu halten. Falls sich ein Profil nicht im Cache befindet so wird es nachgeladen und im Cache abgelegt. Das am längsten inaktive Profil wird zurück auf die Platte gesichert und aus dem Hauptspeicher entfernt.

- Profile und Alarme könnten wie in der Analyse vorgeschlagen in einer Datenbank abgelegt werden, was den Vorteil einer zentralen Datenverwaltung hätte. Auf diesen zentralen Datenbestand könnten auch externe Systeme zugreifen.

- Ein andauernder Betrieb legt die Erweiterung der Applikation zu einer Serversoftware nahe. Der Server kann Datensätze durchgehend verarbeiten. Datensätze können dabei als Events angesehen werden die eine Verarbeitung auslösen. Jeder Gesprächsdatensatz der durch ein externes System erstellt wurde wird an den ProfileMonitor Server gesendet (z.B. über JMS oder bei Performanceproblemen durch ein angepasstes Protokoll) und löst dort eine Kette von Aktionen aus die den Datensatz bewerten.

- In den als Informationsquelle dienenden Verrechnungsdateien finden sich auch Angaben wie z.B. über die Mobilität des Kunden während des Anrufs oder die Nutzung von SMS wieder. Daraus können weitere Informationen für das Profil eines Kunden gewonnen werden.

6.6 Mögliche Anwendungen in anderen Bereichen

Das Einsatzgebiet von künstlichen neuronalen Netzen im Zusammenhang mit der Erstellung von Benutzerprofilen und deren differential Analyse, bei der ein Muster bezogen auf das aktuelle Verhalten ermittelt und mit einem historischen verglichen wird, ist weit.

Verschiedene Einsatzgebiete sind zum Beispiel:

- Käuferverhalten: An Hand von Benutzerprofilen kann das Verhalten auf Marketingaktionen erfasst werden. Im Falle einer Änderung kann eine Nachricht, welche das Ausmaß der Reaktion des Kunden signalisiert, generiert werden. Eine weitere Anwendung eines Benutzerprofils und deren Kategorisierung kann auch die gezielte Einschaltung von Werbungen sein.

- Bewegungsanalysen von Patienten die unter Beobachtung stehen: Weicht das Bewegungsmuster, erfasst durch ein Profil, signifikant vom Normalmuster ab so kann ein Notfallalarm generiert werden.

- Anomalien in einem TCP/IP Netz: Durch den Einsatz einer selbst organisierenden Karte und der Vektor Quantifizierung kann ein Profil für das normale Nutzungsverhalten erstellt und signifikante Abweichungen davon alarmiert werden. Vergleiche dazu [JuMH05].

- Anomalien in Besucherströmen („crowd control“): Durch die Erfassung von Eckdaten die einen Besucherstrom charakterisieren und der Erstellung eines Profils kann auch hier eine Abweichung vom „normalen“ Muster alarmiert werden. Je nach Ausmaß der Abweichung können als Reaktion z.B. nur eine erhöhte Aufmerksamkeit oder ein sofortiges Eingreifen gewählt werden.

- Verhaltensprofile an Hand von Email-Konten.

- Missbrauchserkennung im Kreditkartenbereich.

7 Schlusswort

Betrugsfälle im Mobilfunknetzen werden mit den verschiedensten Methoden identifiziert. Eine darunter die auch einen Schwerpunkt dieser Arbeit darstellt ist der Einsatz künstlicher neuronaler Netze. So weit das Feld der künstlichen neuronalen Netze ist, so weit ist auch deren Einsatzgebiet in kommerziellen Produkten die Betrugsfälle aufzuspüren versprechen. Wie in den ersten Abschnitten der Arbeit dargelegt gehen die meisten Missbrauchsfälle auch mit einer Änderung im Anrufverhalten einher. Aus diesem Grund wurde für den im praktischen Teil der Arbeit entwickelte Prototyp der Weg über die Erstellung und Überwachung von Benutzerprofilen gewählt, genauer der differentiale Ansatz bei dem ein aktuelles mit einem historischen Profil verglichen wird. Sinnvoll erschien in diesem Zusammenhang der Einsatz eines künstlichen neuronalen Netzes zur Bestimmung der Referenzvektoren, die bei der Berechnung der Benutzerprofile genutzt werden, da dadurch eine Konzentration auf diejenige Regionen in der Datenmenge erfolgt in denen tatsächlich eine Häufung an Daten auftritt. Als neuronales Netz welches diese Aufgabe erfüllen kann wurde die selbst organisierende Karte gewählt.

Während der Analyse des implementierten Systems stellte sich heraus, dass die Qualität des Ergebnisses (das Verhältnis von Fehlalarmen zu wirklichen Missbrauchsfällen) stark von der Wahl einiger Parameter und Funktionen abhängig ist. Beispielsweise die Distanzfunktion mit der die Abweichung eines durch einen Vektor repräsentierten Anrufs zu den Referenzvektoren bestimmt wird oder das Intervall auf das Inputdaten abgebildet werden.

Ein weiterer Einflussfaktor auf die Güte des Systems ist die Beschaffenheit der Inputdaten mit denen das künstliche neuronale Netz trainiert wird um aussagekräftige Prototypen (Referenzvektoren) zu erhalten. Da das Benutzerverhalten über den Tages- und Wochenverlauf stark variiert ist es für das Training wichtig eine Menge von Anrufdaten zur Verfügung zu haben die mindestens eine Woche vollständig abdeckt. Weiters ist im Falle von einer zufälligen Auswahl einzelner Datensätze zur Reduktion der Trainingsdatenmenge darauf zu achten dass Attribute die in das Profil aufgenommen werden sollen in diesen Daten auch repräsentativ enthalten sind, beispielsweise Anrufe zu PRS (Premium Rate Services).

Zusammenfassend kann gesagt werden dass der Prototyp der implementierten Applikation die gewünschten Anforderungen erfüllt und Abweichungen im Benutzerverhalten aufzeigt. Jedoch ist um gute Ergebnisse zu erzielen ein hoher Aufwand für das Tuning des Systems notwendig.

8 ANHANG - Receiver Operating Characteristics – ROC

Die wichtigste Eigenschaft eines Betrugserkennungssystems ist der Kompromiss zwischen der Anzahl identifizierter Fälle des Missbrauchs und der Anzahl von generierten Fehlalarmen. Abgesehen von den hohen Kosten die eine durch einen Alarm ausgelöste Analyse nach sich zieht sind Netzwerkbetreiber auch eher vorsichtig wenn es darum geht gute Kunden durch ständige Fehlalarme zu verärgern und diese möglicherweise in Folge an Mitbewerber zu verlieren. Fehleinschätzungen von 1 Prozent sind in diesem Fall inakzeptabel da, wie in der Einleitung erwähnt, die Menge der ausgewerteten Daten sehr groß ist und damit auch in absoluten Zahlen gemessen eine hohe Anzahl an Fehlalarmen generiert würde. Aus diesem Grund ist es wichtig die richtige Balance zwischen Fehlalarmen und Erkennung zu finden.

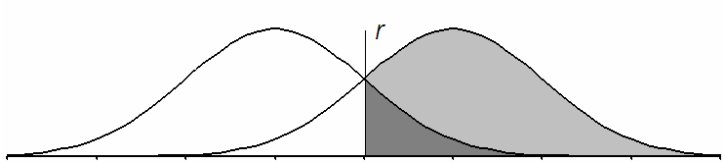

Abbildung 60: Wahrscheinlichkeit eines Fehlalarms [Holl00]

In Abbildung 60 graphisch durch die Dichtefunktionen der beiden Klassen Normalverhalten (die linke Funktion) und Missbrauch (die rechte Funktion) dargestellt ergibt sich folgendes Bild, wenn die Linie r einen Entscheidungspunkt darstellt so entspricht der hellgraue Bereich der Wahrscheinlichkeit den Missbrauch zu erkennen und der dunkelgraue der einen Fehlalarm zu generieren.

Eine Methode um dieses Verhältnis darzustellen ist die ROC welche auf einer Achse die Prozentzahl der korrekt identifizierten Betrügern (Empfindlichkeit) und auf der anderen die der Fehlalarme bei nicht Betrügern (Genauigkeit) für verschiedene Grenzwerte darstellt. Formal dargestellt entspricht die ROC einer Kurve von Punkten (u,v) wobei $p(x|w_1)$ und $p(x|w_2)$ den Wahrscheinlichkeitsdichten der Entscheidungsvariablen entspricht.

$$ROC = \left\{ (u,v) | u = \int_r^{\infty} p(x|w_1)dx; v = \int_r^{\infty} p(x|w_2)dx \right\}$$

Daraus kann der zu optimierende Performanceindex abgeleitet werden, die Fläche unter der Kurve.

Solche Ausgleichskurven ermöglichen dem Benutzer die Steuerung des Systems in Bezug auf die Betrugserkennungsrate im Verhältnis zu der Anzahl der Fehlalarme [MLVV99].

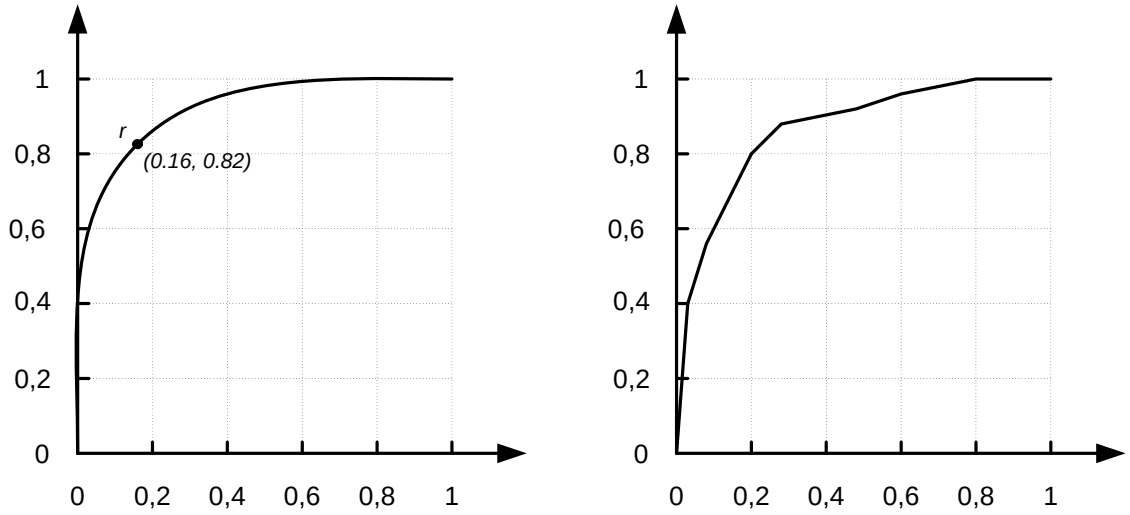

Abbildung 61: Beispiel einer ROC-Kurve [Holl00]

Abbildung 61 zeigt links die ROC Kurve für zwei Dichtefunktionen im Schnittpunkt r der einer Fehlalarmwahrscheinlichkeit von 0,16 und einer Erkennungswahrscheinlichkeit von 0,82 entspricht. Die rechte Kurve stellt ein Beispiel für eine empirisch geschätzte ROC Kurve dar.

9 ANHANG - Genetische Algorithmen

Eine andere Möglichkeit die Gewichte eines vorwärts verketteten Netzes zu Modifizieren ist der Einsatz von so genannten genetischen Algorithmen (GA), genau genommen eine stochastische Suchmethode [ReMa99] welche von John Holland in den 60er und 70er Jahren entwickelt wurde [Hola75]. Dieser Algorithmus ist nicht nur für das Training neuronaler Netze geeignet sondern wird generell zur Lösung vieler Optimierungsprobleme herangezogen. Da er in Bezug auf große nichtlineare Probleme mit mehreren lokalen Minima gut skaliert eignet er sich gut für die Optimierung der Gewichte in einem vorwärts verketteten neuronalen Netz [ReMa99].

Wie man dem Namen entnehmen kann leitet sich der GA von den evolutionären Mechanismen in der Natur ab und implementiert das Prinzip des Überlebens des Stärksten. In diesem Fall der besten bisher erzielten Lösung die sich an Hand einer Zielfunktion bestimmen lässt.

Bei den GA wird zuerst eine geeignete Form der Codierung zur Darstellung der Problem beschreibenden Parameter gesucht. Dies erfolgt analog zur Natur, in der die verschiedenen Genome durch Basenpaarsequenzen codiert sind, weshalb dieser Ansatz auch als genotypisch orientiert bezeichnet wird. Im Unterschied zur Codierung mit vier Basenpaaren in der Natur, bevorzugt man aus Gründen der effizienteren numerischen Behandlung bei GA binäre Codierungen. Nach der Wahl der Codierung findet eine Initialisierung der Anfangspopulation statt. Die Individuen der aktuellen Generation werden mittels einer Bewertungsfunktion, die die Güte des Individuums darstellt, bewertet. In einem zweiten Schritt wird aus diesem Wert die individuelle Fitness berechnet. Durch ein zufallsgesteuertes Auswahlverfahren, in das die Fitness der Individuen eingeht, werden Chromosomenvektoren zur Nachkommenerzeugung ausgewählt. Diese ausgewählten Chromosomenvektoren bilden durch Rekombination neue Nachkommen, wobei die Auswahl und die Rekombination durch ein Heiratsschema gesteuert werden. Anschließend wird aus den Nachkommen und den Elternchromosomen durch das Ersetzungsschema die neue Generation gebildet und der Vorgang, beginnend mit der Bewertung der Individuen der aktuellen Generation, bis zum Erreichen eines Abbruchkriteriums wiederholt durchlaufen [Wolk97].

Einer der Hauptvorteile dieser Methode ist, dass sie nur wenige problemspezifische Informationen benötigt, man benötigt nur eine Fitness-Funktion welche die individuelle Lösung bewertet und damit Abschätzung der Qualität der Lösung liefert. Als Nachteil ist die hohe Rechenzeit zur Bewertung großer Populationen und dem konvergieren hin zu einem Optimum anzuführen. Andere Methoden liefern oft schneller gleichwertige Lösungen.

GA können auf verschiedene Weise auf neuronale Netze angewandt werden, von der Bestimmung der Gewichte in einem vorgegeben Netz bis hin zur Wahl der kompletten Netzarchitektur: der Anzahl von Schichten und Neuronen, Verbindungen, Gewichtungen usw.

10 Literaturverzeichnis

[AHKB03]
Van der Aalst, W.M.P.; Hofstede, A.H.M.; Kiepuszewski, B.; Barros, A.P.: Workflow Patterns. http://is.tm.tue.nl/research/patterns/download/wfs-pat-2002.pdf, 2003-07, Abruf am 2006-06-17.

[AnFA04]
Anrig, Bernhard; Fornaiser, Patric; Aeschbacher, Daniel: Künstliche Neuronale Netze - Eine Einführung. HTI Biel, März 2004.

[ASF06]
Apache Software Foundation (Hrsg.): Logging Services Log4j. http://logging.apache.org/log4j, 2006, Abgerufen am 2006-07-17.

[BoHa02]
Bolton, R.J.; Hand, D.J.: Statistical fraud detection: A review. In: Statistical Science, 17(3) (2002), S. 235-249.

[BSCM97]
Burge, P.; Shawe-Taylor, J.; Cooke, C.; Moreau, Y.; Preneel, B.; Stoermann, C.: Fraud detection and management in mobile telecommunication networks. In: IEE Conference publication 437, London 1997, S. 91-96.

[BuSh01]
Burge, Peter; Shawe-Taylor, John: An Unsupervised Neural Network Approach to Profiling the Behavior of Mobile Phone Users for Use in Fraud Detection. In: Journal of Parallel and Distributed Computing 61 (2001), S. 915-925.

[CEWB97]
Cox, K. C.; Eick, S. G.; Wills, G. J.; Brachman, R. J.: Visual data mining: recognizing telephone calling fraud. In: Journal of Data Mining and Knowledge Discovery 1(2) (1997), S. 225-231.

[CFCA03]
CFCA (Hrsg.): Communications Fraud Control Association (CFCA) Announces Results Of Worldwide Telecom Fraud Survey. Survey Results Suggest Link Between Telecom Fraud And Terrorism. In: CFCA Press Release, März 2003. http://www.cfca.org/pressrelease/FraudLoss%20%20press%20release%203-03.doc, Abruf am 2005-09-30.

[Dodg05]
Dodge, Steve: Use Spring to create a simple workflow engine. http://www.javaworld.com/javaworld/jw-04-2005/jw-0411-spring.html, 2005-04-11, Abruf am 2006-06-17.

[EsHP06]
Estévez, P.A.; Held, C.M.; Perez, C.A.: Subscription Fraud Prevention in Telecommunications Using Fuzzy Rules and Neural Networks. In: Expert Systems with Applications Vol. 31 No. 2 (2006), S. 337-344.

[Gähl06]
Gähwiler, B.: Pyramidenförmige Gehirnzelle (Neuron). http://www2.unil.ch/edab/old/images/presse/neuron.jpg, Abruf am 2006-07-19.

[Garb05]
Garber, Mikhail: Use Spring built-in facilities to capture business logic. http://www.javaworld.com/javaworld/jw-04-2005/jw-0425-ruleengine.html, 2005-04-25, Abruf am 2006-06-17.

[GaReoJ]
Gauglitz, Guenter; Reichert, Manuela: Neuronale Netze - Eine Einführung. D-10587 Berlin, Franklinstraße 11, FIZ CHEMIE Berlin, Fachinformationszentrum Chemie GmbH.
http://www.vs-c.de/vsengine/tra/vsc/de/ch/13/trajektorien/nn_ein.tra.html, Abruf am 2005-10-05.

[Germ99]
Germano, Tom: Self Organizing Maps, 23 März 1999.
http://davis.wpi.edu/~matt/courses/soms/index.html, Abruf am 2005-10-11.

[HoHa99]
Hodju, Petri; Halme, Jokko: Neural Networks Information Homepage. http://koti.mbnet.fi/~phodju/nenet/index.html, 1999, Abruf am 2006-02-25.

[Hola75]
Holland, John: Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Systems. *The University Press of Michigan Press (Hrsg)*, 1975.

[Holl00]
Hollmén, Jaakko: User profiling and classification for fraud detection in mobile communications networks. Helsinki University of Technology, Department of Computer Science and Engineering, December 2000.

[Holl96]
Hollmén, Jaakko: Using the Self-Organizing Map.
http://www.cis.hut.fi/jhollmen/dippa/dippa.html, 1996-02-15, Abruf am 2006-03-28.

[IECoJ]
International Engineering Consortium (Hrsg.): Tutorial: Fraud Analysis in IP and Next-Generation Networks.
http://www.iec.org/online/tutorials/fraud_analysis/index.html, Abruf am 2005-09-30.

[JuMH05]
Jun, Zheng; Mingzeng, Hu; Hongli, Zhang: Usage behavior profiling for anomaly detection using vector quantization. In: Communication Systems and Applications 2005, Juli 2005, S. 107-112.

[Kask97]
Kaski, Samuel: Data Exploration Using Self-Organizing Maps. Doktorarbeit, Technische Universität Helsinki, Neural Networks Research Centre, 1997.

[Koho01]
Kohonen, Teuvo: Self-Organizing Maps. 3. Aufl., Springer, New York 2001.

[Kreb00]
Krebs, Susanne: Selbstorganisierende Karten. Seminararbeit zu „Intelligente Systeme im Finance“, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, September 2000.

[LuSt97]
Luger, Georg; Stubblefield, William: Artificial Intelligence. Structures and Strategies for complex problem solving; Addison Wesley Verlag Reading, Mass., 1997.

[MaJo06]
Marrone, Paolo; Joone Team: Java Object Oriented Neural Engine – JOONE. http://www.jooneworld.com/, 2006, Abgerufen am 2006-07-17.

[MLVV99]
Moreau, Y.; Lerouge, E.; Verrelst, H.; Vandewalle, J.; Stormann, C.; Burge, P.: A Hybrid System for Fraud Detection in Mobile Communications. In: Proc. of European Symposium on Artificial Neural Networks (1999), S. 447-454.

[MoVa97]
Moreau, Yves; Vandewalle, Joos: Detection of Mobile Phone Fraud using supervised Neural Networks: A first Prototype. In International Conference on Artificial Neural Networks Proceedings (ICANN'97), S. 1065–1070.

[Nort00]
Nortel Networks (Hrsg.): Bellcore Format Automatic Message Accounting – Reference Guide. Mai 2000.

[OMG06]
Object Management Group (Hrsg.): Unified Modeling Language – UML. http://www.uml.org/, 2006, Abgerufen am 2006-07-20.

[Pass04a]
Universität Passau: Technische Anwendungen von Selbstorganisierenden Karten. Universität Passau, Fakultät für Mathematik und Informatik. http://lrs2.fmi.uni-passau.de/online/SOM/SOM.pdf, 2004-07-15, Abruf am 2006-02-25.

[Pass04b]
Universität Passau: Technische Anwendungen von mehrlagigen Perzeptren. Universität Passau, Fakultät für Mathematik und Informatik. http://lrs2.fmi.uni-passau.de/online/MLP/MLP.pdf, 2004-07-20, Abruf am 2006-03-01.

[Reif00]
Reif, Gerald: Moderne Aspekte des Wissensverarbeitung - Ein interaktiver Lernbehelf für das Web Based Training. 8010 Graz, Technische Universität Graz, Jänner 2000.

[ReMa99]
Reed, Russel; Marks, Robert: Neural Smithing. Supervised Learning in Feedforward Artificial Neural Networks. *The MIT Press (Hrsg)*. 1999.

[Salp03]
Salpeter, Alice: A Holistic Approach Required to Survive the Ongoing Battle Against Telecommunication Fraud. *Forrester Research (Hrsg)*, Juli 2003.

[SestoJ]
Sester, Monika: Typifizierung mittels Kohonen Merkmalskarten. 30167 Hannover, Appelstraße 91, Universität Hannover.

[ShHB99]
Shawe-Taylor, John; Howkev, Keith; Burge, Peter: Detection of Fraud in Mobile Telecommunications. In: ACTS AC095, Projekt ASPeCT, Information Security Technical Report Vol. 4 (1999) 1, S. 16-28.

[Spri06]
Springframework (Hrsg.): Spring Application Framework. http://www.springframework.org/, Abgerufen am 2006-07-17.

[Srin95]
Srinivasan, R.: Request for Comments 1832 - XDR: External Data Representation Standard. Sun Microsystems, August 1995.

[Stat06]
Statsoft, Inc. (Hrsg.): Electronic Statistics Textbook. http://www.statsoft.com/textbook/stathome.html, 2006, Abruf am 2006-07-19.

[Weis98]
Weiss, Franz: Strukturfindung für Neuronale Time-Delay-Netze mit Hilfe verteilter genetischer Algorithmen. Diplomarbeit, 1998.

[Weiß04]
Weiß, Jochen: Selbstorganisierende Karten. Seminararbeit im Proseminar Soft Computing, Universität Ulm, Fakultät für Informatik, Mai 2004.

[Wiel04]
Wieland, K.: The last taboo? Revenue leakage continues to hamper the telecom industry. In: Telecommunications (International Edition) 38 (2004), S. 10-11. http://www.telecoms-mag.com/techzones/Article.asp?Id=0408i05, 2004-08-01, Abruf am 2005-09-30.

[Wiki06]
Wikipedia (Hrsg.): Nervenzelle. *http://de.wikipedia.org/wiki/Nervenzelle*, 2006-07-07, Abruf am 2006-07-19.

[Wolk97]
Wolkenhauer, Markus: Untersuchungen von evolutionären Algorithmen zum Training neuronaler Netze in der Sprachverarbeitung. Diplomarbeit, Johann Wolfgang Goethe-Universität Frankfurt am Main, Oktober 1997.

[Zell94]
Zell, Andreas: Simulation, Neuronaler Netze, Addison-Wesley Publishing, Bonn, Paris, Reading, 1994.