– Diplomarbeit –

Techniken des Data Mining zur Analyse von Telekommunikationsnetzwerken

Stefan Michaelis

Dezember 2000

Lehrstuhl für
Künstliche Intelligenz

Prof. Dr. K. Morik

Fachbereich Informatik
Universität Dortmund

Lehrstuhl für
Elektronische Systeme
und Vermittlungstechnik

Prof. Dr. R. Schehrer

Fachbereich Elektrotechnik
Universität Dortmund

We are drowning in information,
but starving for knowledge.
– John Naisbett

Zusammenfassung

Diese Diplomarbeit beschäftigt sich mit der Analyse von Zeitreihen aus dem Bereich der Lastkurven von Telekommunikationsnetzwerken. Betrachtet werden Verfahren des maschinellen Lernens, jeweils mit einer Einführung in die theoretischen Grundlagen und anschließender praktischer Anwendung. Abschließend erfolgt die Integration eines Lastprognosewerkzeugs in ein Simulations- und Managementsystem für Telekommunikationsnetzwerke.

Danksagung

Mein Dank geht an Katharina Morik und Rudolf Schehrer, ohne deren unbürokratische Kooperation und Unterstützung diese Diplomarbeit nicht stattgefunden hätte. Weiter danke ich Peter Brockhausen, der mir als geduldiger Ansprechpartner zu allen Fragen aus der künstlichen Intelligenz zur Verfügung stand.

Mein besonderer Dank geht an Walter Bindrim, nicht nur für seine ständige Hilfsbereitschaft, sondern vielmehr auch für die besonderen Eindrücke und Motivationen, die er mir im Verlauf dieser Diplomarbeit vermitteln konnte.

Letztendlich, allerdings nicht weniger bedeutend, geht mein Dank an Gerlinde Becker, Carsten Ernemann, David Rieger und Stefan Wesselmann, die gnadenlos für mehr Arbeit und einen (hoffentlich) besseren Text gesorgt haben.

Alle verbliebenen Fehler gehen voll zu Lasten des Autors.

Inhaltsverzeichnis

1 Einleitung **5**

2 Das System MEDIAN **7**

2.1 Verkehrstheorie . . . 7

2.1.1 Nachrichtenverkehr . . . 7

2.1.2 Leitungsdimensionierung . . . 8

2.1.3 Netzwerksysteme . . . 10

2.2 ATM – Asynchronous Transfer Mode . . . 11

2.3 Median . . . 14

2.3.1 Das Grundkonzept von Median . . . 14

2.3.2 Implementierung von Median . . . 18

3 Data Mining **20**

3.1 Begriffsklärung . . . 20

3.2 Zielsetzung . . . 21

3.3 Lernen . . . 22

3.4 Probleme . . . 25

3.5 Prozessmodell für das Data Mining . . . 26

4 Entscheidungsbäume **28**

4.1 Entscheidungsbäume – Decision Trees . . . 28

4.2 Konstruktion der Bäume . . . 30

4.3 Reduktion der Entscheidungsbäume . . . 33

4.4 Repräsentation der Daten . . . 35

4.5 Ergebnisse . . . 42

4.5.1 Verteilung der Last auf einer Route . . . 43

4.5.2 Verteilung der Last über drei Routen . . . 47

4.5.3 Diskrete Attribute . . . 48

4.5.4 Adaptive Zielklassengrößen . . . 49

4.5.5 Kürzere Taktung . . . 50

4.5.6 Kreuzvalidierung . . . 51

4.5.7 Logarithmische Attributauswahl . . . 52

4.5.8 Berücksichtigung der Steigung . . . 53

4.5.9 Auswertung entlang mehrerer Links . . . 54

4.5.10 Zusammenfassung . . . 55

5 Support Vector Machine 56
5.1 Messung des Risikos . . . 56
5.2 Die VC Dimension . . . 57
5.3 Support Vektoren . . . 59
5.4 Kostenfunktionen zur Regression . . . 62
5.5 Kernelfunktionen . . . 62
5.6 Ergebnisse . . . 64
5.6.1 Unterschiedliche Kapazitäten . . . 65
5.6.2 Verschiedene Vergangenheitsbereiche . . . 66
5.6.3 Vergleich der Kernel . . . 68
5.6.4 Logarithmische Attributauswahl . . . 69
5.6.5 Berücksichtigung der Steigung . . . 70
5.6.6 Kreuzvalidierung . . . 70
5.6.7 Zusammenfassung . . . 71

6 Vergleich der Verfahren 73
6.1 Tagesprognosen . . . 73
6.2 Angebotssteigerungen . . . 80
6.3 Einführung von Wochentagen . . . 83
6.4 Zusammenfassung . . . 84

7 Integration in MEDIAN 86
7.1 Parallel Virtual Machine . . . 86
7.2 Programmdokumentation . . . 90
7.3 Klassenübersicht . . . 95

8 Ausblick 99

A Installation und Betrieb 102
A.1 Stabilität . . . 102
A.2 Hilfsprogramme . . . 103

B Angebotskurven der Dienste 106

Literaturverzeichnis 107

Index 110

Kapitel 1

Einleitung

Im Zuge des wachsenden Telekommunikationsmarktes hat die Bedeutung der Dimensionierung und sinnvollen Auslastung von Netzwerken stetig zugenommen. Ungenutzte Kapazitäten bilden im Wettbewerb unter unzähligen Anbietern ein Potenzial, das nicht ungenutzt bleiben sollte. Im Gegenzug dürfen Reserven nicht vernachlässigt werden, die eine Reaktion auf Ausfälle und ungewöhnlich hohe Lasten im Netzwerk ermöglichen.

Zum Zweck der Untersuchung des Verhaltens von logischen und physikalischen Netzwerken entsteht am Lehrstuhl *Elektronische Systeme und Vermittlungstechnik* der Universität Dortmund das Softwarepaket *MEDIAN*[1].

Dieses ermöglicht anhand von Netzstruktur und Belastungskurven eine Simulation des Netzzustandes. Es besteht dabei die Möglichkeit, unterschiedliche Dienste mit ihren Merkmalen (konstante/variable Bitrate, Echtzeitanwendung...) parallel zu betrachten und Konfliktpunkte zu erkennen. Ziel ist es, auf hohe Lasten und Engpässe flexibel reagieren zu können und gleichzeitig die Auslastung des gesamten Netzes hoch zu halten. Die sich daraus ergebende Problematik liegt in der Komplexität der Berechnungen, der Menge der generierten Daten und der Bestimmung des Verhaltens des Systems.

An dieser Stelle setzt der zum maschinellen Lernen gehörende Bereich des *Data Mining (DM)* oder auch *Knowledge Discovery in Databases (KDD)* an. Diese Begriffe tragen dem steigenden Bedarf an Wissensentdeckung in großen und schwer handhabbaren Datenbanken Rechnung, die mit herkömmlichen Methoden nur schlecht zu bewältigen sind. Mit der zunehmenden Popularität des Data Mining wächst auch der Bedarf, einzelne Verfahren und Anwendungsgebiete nach Komplexität, Zeitbedarf und ähnlichen Kriterien einzuordnen. Ziel dieser Diplomarbeit ist es, unterschiedliche Verfahren voneinander abzugrenzen und auf ihre Einsetzbarkeit im Sinn der Vorhersage des Netzwerkverhaltens zu untersuchen. Im Vordergrund stehen dabei das Lernen, Erkennen von Mustern, Registrieren von Abweichungen und (nötigenfalls) der Anstoß eines erneuten Lernprozesses. Neben der Genauigkeit der Vorhersage ist ebenfalls die Geschwindigkeit dieses Zyklus von Belang, da Ergebnisse innerhalb eines Zeitraumes vorliegen müssen, der eine Konfiguration des Netzes auf der Basis von aktuellen Daten gewährleistet. Durch einen Kompromiss zwischen Geschwin-

[1] Management eines Dimensionierungswerkzeuges für ATM-Netzwerke.

digkeit und Genauigkeit kann es zu einem Trade-Off zwischen beiden Punkten kommen.

Diese Diplomarbeit ist in Kooperation zwischen dem Lehrstuhl für *Elektronische Systeme und Vermittlungstechnik (ESV)* des Fachbereichs Elektrotechnik und des Lehrstuhls für *Künstliche Intelligenz (KI)* des Fachbereichs Informatik der Universität Dortmund entstanden. Das zugrundeliegende System und das Hintergrundwissen zur Verkehrstheorie ist dem ESV entnommen, Werkzeuge und Gedanken des maschinellen Lernens stammen aus der KI.

Kapitel 2 stellt eine Einführung in das System Median und in grundlegende verkehrstheoretische Aspekte dar. An entsprechenden Stellen wird auf die beteiligten Diplomarbeiten verwiesen. In Kapitel 3 folgen die allgemeinen Konzepte des Data Mining. Mit der konkreten Realisierung der Verfahren und Algorithmen beschäftigen sich die Kapitel 4 und 5. Diese betrachten unterschiedliche Datentransformationen und Experimente zur Bestimmung optimaler Parametereinstellungen. In Kapitel 6 folgt der direkte Vergleich der Verfahren. Zusätzlich greift dieses Kapitel den an Median angegliederten Netzwerksimulator auf und stellt Besonderheiten und spezielle Transformationen der Netzwerkdaten dar. Die abschließende Integration eines Mining Moduls in Median beschreibt Kapitel 7.

In Anhang A werden Hinweise zu Betrieb und Installation der Data Mining-Module gegeben. Dieses Kapitel ist für den Fall gedacht, dass eine Neuinstallation notwendig geworden ist oder weitere Arbeiten an Median erfolgen sollen. Anhang B fasst die Median zugrundeliegenden Angebotskurven zusammen.

Kapitel 2

Das System MEDIAN

Als Reaktion auf die beschriebenen Veränderungen im Telekommunikationssektor ist das System *MEDIAN*[1] entstanden. Ziel der prototypisch implementierten Software ist es, die Realisierbarkeit eines dynamischen Managementsystems für ATM-Netzwerke zu zeigen. Dieses Kapitel gibt eine kurze Einführung in verkehrstheoretische Grundlagen und Begriffe, die Median zugrunde liegen, und stellt anschließend das realisierte Konzept mit seinen Modulen vor. Eine Übersicht über das Gesamtsystem lässt sich ebenfalls in [Bindrim00] finden.

2.1 Verkehrstheorie

In diesem Abschnitt werden Begriffe und Verfahren aus der Verkehrstheorie erläutert. Die dargestellten Gesetzmäßigkeiten bestimmen das Verhalten des Netzwerks und bilden die Grundlage für eine Simulation. Dabei orientiert sich der Text weitgehend an [Schehrer99].

2.1.1 Nachrichtenverkehr

Der Nachrichtenverkehr lässt sich als Zufallsprozess mit voneinander unabhängigen Variablen, den Teilnehmern, betrachten. Teilnehmer können in diesem Zusammenhang beliebige Quellen sein und werden nicht nur im herkömmlichen Sinn als Begriff für Telefonteilnehmer belegt. In den heutigen Netzwerken besteht häufig auf Ebene der Hardware eine Abstraktion von der Art der zu übertragenden Daten. Dies führt dazu, dass ein und dasselbe Netzwerk für mehrere Datenübertragungen parallel benutzt wird. Diese Anwendungen, wie z.B. Telefon, Übertragung von Daten im Sinne des Internets oder auch Unterhaltung (digitales Fernsehen…), werden allgemein als *Dienst* bezeichnet, ein Netzwerk, das mehrere Dienste zulässt, als *dienstintegrierend*.

Der Zufallsprozess des Nachrichtenverkehrs, der von den Diensten abhängt, wird durch drei Größen charakterisiert: Den *Ankunfts-*, den *Zustands-* und den *Endeprozess*. Der Zustandsprozess bezeichnet die zeitlich veränderliche Anzahl an Belegungen. Bestimmt wird er wesentlich durch die beiden anderen Prozes-

[1] Management eines Dimensionierungswerkzeuges für ATM-Netzwerke.

se, die jeweils Beginn und Ende eines *Rufes*, bzw. eines *Verbindungswunsches*, signalisieren.

Empirisch wurde nachgewiesen, dass für den Ankunftsprozess (die Verteilung der Anrufabstände) in der Praxis oft die folgende negativ-exponentielle Verteilungsfunktion beobachtet werden kann:

$$F_A(t) = P(T_A \leq t) = 1 - e^{-\lambda t}.$$

Dabei stellt λ die mittlere Anzahl von Anrufen dar, die während einer Zeiteinheit eintreffen, und wird deshalb als *Anrufrate* bezeichnet. Der Mittelwert, bzw. Erwartungswert $E(T_A)$, ergibt sich als Kehrwert der Anrufrate:

$$E(T_A) = 1/\lambda.$$

Für die Betrachtung des Endeprozesses kommt noch ein anderer Aspekt hinzu: die *Verbindungsdauer*. Die zeitliche Dauer einer Belegung, bzw. Verbindung, (englisch: holding time) hängt im Allgemeinen vom Zufall ab. Der Mittelwert der Belegungsdauer wird üblicherweise mit h, die Zufallsvariable entsprechend mit T_H bezeichnet. Die Verteilung der Belegungsdauern kann je nach Art des zugehörigen Dienstes verschieden sein. In Datennetzen kommt es häufig vor, dass die Größe der Datenpakete konstant ist; in anderen Fällen können die Belegungsdauern sehr unterschiedlich sein. Messungen haben gezeigt, dass die Belegungsdauern von Telefongesprächen in vielen Fällen negativ-exponentiell verteilt sind und der folgenden Verteilungsfunktion genügen:

$$F_H(t) = P(T_H \leq t) = 1 - e^{-\frac{t}{h}}.$$

Der Kehrwert der mittleren Belegungsdauer wird als Enderate ε bezeichnet:

$$\varepsilon = 1/h.$$

Der darauf aufbauende Zustandsprozess ist maßgeblich für die Dimensionierung von Leitungen und wird daher im nächsten Abschnitt beschrieben.

2.1.2 Leitungsdimensionierung

Wird die Anrufrate des Ankunftsprozesses der bei einem Bündel von n Leitungen eintreffenden Rufe über 24 Stunden eines Tages betrachtet, ergibt sich in den seltensten Fällen eine konstante Anrufrate, sondern diese wird erheblich schwanken. Höchstwerte ergeben sich bei der Telefonie z.B. häufig kurz vor oder nach der Mittagszeit. Bei der Dimensionierung von Leitungsbündeln müssen diese Raten berücksichtigt werden, um vorgegebene Verlustwahrscheinlichkeiten einzuhalten und ein intelligentes Management auf den Bündeln, wie Median es leistet, zu ermöglichen. In der Praxis wird die maximale Anrufrate meist auf die Stunde mit dem größten Durchschnittswert, die sogenannte *Hauptverkehrsstunde*, bezogen, eventuell zuzüglich eines Sicherheitsabstandes, um auch auf steigende, zukünftige Lasten vorbereitet zu sein.

Aus dem Verlauf des Zustandsprozesses über einen Tag ergibt sich, dass es sich nicht um einen homogenen Prozess handelt. Lediglich in der für die Dimensionierung interessanten Zeit der Hauptverkehrsstunde kann die Ankunftsrate als konstant angesehen werden und damit ist zu dieser Zeit der Zustandsprozess homogen und stationär. Für die übrigen Zeiten ist die Vorhersage der Lasten, das Thema dieser Diplomarbeit, von Bedeutung, worauf in späteren Kapiteln eingegangen wird.

Aus Anrufrate λ und mittlerer Belegungsdauer h ergibt sich das Produkt λh als Maß dafür, wie viele Belegungen im Mittel vorhanden wären, wenn keine Besetztfälle aufträten, d.h. die mittlere Anzahl an Belegungen bei unendlich vielen Leitungen ($n \to \infty$). Dieser Wert wird als *Verkehr* bezeichnet und gibt den „gewünschten" oder dem Leitungsbündel „angebotenen" Verkehr an. Das Vorhandensein einer unbegrenzten Anzahl von Leitungen ist Voraussetzung dafür, dass jeder Verbindungswunsch eines Telekommunikationsteilnehmers zu jeder Zeit durchgeschaltet werden kann, nur dann entspricht der Verkehr, beziehungsweise das *Angebot* A, λh.

Nach A.K. Erlang ergibt sich die Formel für die Zustandswahrscheinlichkeiten bei einem Bündel mit n Leitungen:

$$p(x) = \frac{\frac{A^x}{x!}}{\sum_{i=0}^{n} \frac{A^i}{i!}}.$$

Neben den Zustandswahrscheinlichkeiten sind noch andere Größen für eine Charakterisierung von Netzen ausschlaggebend. Die Wahrscheinlichkeit, dass ein ankommender Ruf nicht durchgeschaltet werden kann, weil keine freie Leitung mehr vorhanden ist, wird als *Verlustwahrscheinlichkeit* oder *Verlust* B bezeichnet. Die Wahrscheinlichkeit, dass ein eintreffender Ruf verloren geht, ist gleich der Wahrscheinlichkeit, dass zu dem Zeitpunkt des eintreffenden Rufes alle n Leitungen belegt sind. Da die Anrufe unabhängig voneinander zufällig eintreffen und die Anrufrate unabhängig von der Anzahl der bereits belegten Leitungen ist, gilt für die Verlustwahrscheinlichkeit:

$$B = p(n) = \frac{\frac{A^n}{n!}}{\sum_{i=0}^{n} \frac{A^i}{i!}} = E_{1,n}(A).$$

Diese Formel wird als *Erlangsche Verlustformel* oder als *1. Erlang-Formel* bezeichnet.

Erlang hat zusätzlich gezeigt, dass seine Formel nicht nur für negativexponentiell verteilte Belegungsdauern gilt, sondern auch für beliebige Verteilungen, was konstante Belegungsdauern für Datenübertragungen mit einschließt.

Die mittlere Anzahl gleichzeitig belegter Leitungen heißt *Verkehrswert* oder *Verkehr* Y (bei $n \to \infty$). Dieser ergibt sich nach:

$$Y = E(X) = \sum_{x=0}^{n} x \cdot p(x).$$

Alternativ lässt sich der Verkehr auch aus den beiden übrigen Größen Angebot und Verlust bestimmen:

$$Y = A(1 - B).$$

Prinzipiell sind die Größen A und Y dimensionslos, um sie allerdings als Verkehrswerte (Rufe/Belegungen zum jeweiligen Zeitpunkt) auszuzeichnen, wird ihnen die Pseudo-Einheit *Erlang* zugeordnet.

Der Verkehr, der nicht durchgeschaltet werden kann, wenn das Angebot A die Leitungskapazität überschreitet, wird als *Verkehrsrest* R bezeichnet und ergibt sich als

$$R = A - Y$$

oder

$$R = A \cdot B.$$

Um den Zustandsprozess besser einschätzen zu können, ist es nützlich, den Grenzfall für sehr viele, bzw. unendlich viele Leitungen zu betrachten. Für die Erlangsche Formel ergibt sich dann:

$$p(x) = \lim_{n \to \infty} \frac{\frac{A^x}{x!}}{\sum_{i=0}^{n} \frac{A^i}{i!}} = \frac{A^x}{x!} \cdot e^{-A},$$

woraus sich in der Gesamtheit der Wahrscheinlichkeiten die Poisson-Verteilung ergibt.

Details zur Umsetzung dieser Formeln zum Zweck der Simulation von Netzwerken lassen sich in [Tegtmeier99] nachlesen.

2.1.3 Netzwerksysteme

Bislang ist die Last auf dem Netzwerk durch die Anzahl belegter Leitungen in der Einheit Erlang beschrieben worden. Eine einzelne Belegung muss dabei nicht zwangsläufig auch einer physikalischen Leitung entsprechen. In modernen dienstintegrierenden Systemen werden in den meisten Fällen auf einer physikalischen Leitung mehrere Dienste durch Multiplexverfahren realisiert.

Ein Erlang eines einzelnen Dienstes entspricht einer Last, die der durchschnittlichen Bitrate dieses Dienstes zugeordnet ist. Die Bitraten der Dienste lassen sich in kleinste Einheiten von 64 kBit/s aufteilen. Für die Telefonie reicht beispielsweise einer dieser Kanäle aus, andere Dienste müssen für ihre Übertragungen mehrere dieser Kanäle bündeln. Bei Diensten, die keine konstante Bitrate aufweisen, ist es sinnvoll, die verschiedenen Parameter (z.B. maximale, geringste Bitrate und Burstfaktor[2]) in eine durchschnittliche, konstante Bitrate umzuwandeln. Hierfür gibt es verschiedene Methoden, die mehr oder weniger pessimistisch die durchschnittlich generierte Last berechnen und damit entweder Kapazität als Sicherheitsreserve ungenutzt lassen oder bei großen Bursts den Anforderungen der Quelle nicht mehr genügen.

Unter dem Aspekt der Vorhersage ist es sinnvoll, die Prognose in Erlang vorzunehmen und dann diesen Wert in die herkömmlichen Bitraten zu transformieren. Damit ist nicht die komplexe Vorhersage der aufzuwendenden und teilweise variablen Bitraten nötig, sondern lediglich die Anzahl der benötigten

[2] Der Burstfaktor ist eine Größe dafür, wie ungleichmäßig eine Quelle sendet. Ein hoher Faktor kennzeichnet Quellen, die zwischen langen, vergleichsweise ruhigen Phasen mit nur geringer Übertragungsrate hohe Spitzenlasten erzeugen.

Leitungen. Die Feststellung und Reservierung der nötigen Kapazität auf den Links erfolgt dann wieder durch eine gesonderte Optimierung.

Neben der Last zwischen zwei Punkten spielt auch der Weg durch das Netz eine Rolle. Die wenigsten Netze werden alle möglichen Wege zwischen den einzelnen Teilnehmer direkt implementieren, um den Aufwand und die Kosten in einem akzeptablen Rahmen zu halten. Die Suche und Festlegung optimaler Routen zwischen Quelle und Senke ist einer der wesentlichen Faktoren für einen effizienten Betrieb eines Netzes. Der Vorgang der Nachrichtenweiterleitung wird als *Routing* bezeichnet, Verbindungen mit einem oder mehreren Zwischenpunkten (Hops) als *Links*. Wird das Routing flexibel gehandhabt, ist dies ein dynamischer Vorgang, der sich im Verlauf des Betriebs ändert, d.h. die Wege durch das Netz bleiben nicht konstant. Viele herkömmliche Verfahren, wie beispielsweise im Internet realisiert, benutzen meist dieselben Wege zwischen zwei Punkten, lediglich im Fehlerfall oder zur Überlastabwehr kommen Alternativen in Frage.

Der nachfolgende Abschnitt beschreibt, wie die hier beschriebenen Konzepte durch die Spezifikation von ATM und innerhalb Medians umgesetzt sind.

2.2 ATM – Asynchronous Transfer Mode

Das System simuliert ein Netzwerk auf Basis des für Hochgeschwindigkeitsanwendungen ausgelegten ATM. In diesem Abschnitt werden die grundlegenden Eckdaten von ATM-Anwendungen aufgezeigt. Vertiefende Punkte finden sich in [Kyas96].

Im LAN[3]-Bereich, für den Leitungslängen zwischen den Teilnehmern wenige 100 m und (theoretische) Bitraten selten mehr als 1 GBit/s betragen, sind ATM-Netze relativ selten zu finden, typische Einsatzgebiete finden sich im Betrieb von leistungsstarken Backbone-Netzen.

Backbone-Netze stellen leistungsfähige Hintergrundnetze dar, die mehrere Teilnehmer mit hoher Übertragungskapazität oder leistungsschwächere Teilnetze miteinander verbinden und die Daten dazwischen transportieren. Für den Endteilnehmer sind diese Netze meist nicht sichtbar, da sie durch sein jeweiliges Teilnetz gekapselt werden.

Realisierbare Übertragungsraten liegen im Bereich von ca. 1 TBit/s. Ermöglicht wird dies durch den einfachen Aufbau einzelner ATM-Zellen. Im Gegensatz zu TCP/IP[4], das z.B. im Internet die Übertragung mit der sogenannten *store and forward*-Technik realisiert, basiert ATM auf der verbindungsorientierten Übertragung der Informationen. Auf Anwendungsebene werden die Daten vollständig zusammen mit der Zielinformation an die ATM-Schicht übergeben. Der sogenannte ATM-Adaption-Layer (AAL) übernimmt die Aufgabe, die Daten in einzelne Pakete aufzuteilen und entsprechend der Zielinformation weiterzuleiten. Die Paketgröße ist dabei fest vorgegeben, was eine schnelle Weiterleitung einzelner Pakete ermöglicht (Cell Switching Technologie). Siche-

[3]Local Area Network.

[4]Transfer Control Protocol/Internet Protocol.

rungsmechanismen (z.B. CRC[5]) existieren bei ATM lediglich für die Header-Informationen, um die richtige Übertragung bezüglich Absender/Ziel zu garantieren. Eine Sicherung der Anwendungsdaten muss auf einer höheren, nicht ATM spezifischen Ebene geschehen.

Mit diesen Eigenschaften gehören mittels ATM betriebene Netzwerke zu den dienstintegrierenden Netzwerken, d.h. Informationen völlig unterschiedlicher Art können transparent nebeneinander über eine Leitung übertragen werden. Für die ATM-Schicht sind lediglich die Verbindungscharakteristika von Bedeutung. In Median sind beispielhaft unterschiedliche Dienste wie z.B. Telefon oder digitales Fernsehen im selben Netzwerk vereint.

ATM weist zusätzlich weitere Besonderheiten auf, die für den Gesichtspunkt der Simulation und des Managements von Bedeutung sind. Zunächst ist dies die Verknüpfung von logischem und physikalischem Netzwerk. Grundsätzlich besteht für die Informationsübermittlung die Notwendigkeit echter Verbindungen auf Ebene der Hardware zwischen einzelnen Knoten. Aus Kosten- und Aufwandsgründen wird dieses Netz immer nur eine kleine Zahl aller verfügbaren Sender/Empfänger direkt miteinander verbinden. Um jedoch trotzdem eine Kommunikation aller Teilnehmer untereinander zu ermöglichen, setzt auf dem physikalischen Netzwerk das logische auf. Auf Ebene der Anwendung ist damit die Adresse, bzw. Identifikationsnummer, des Empfängers bekannt. Für die Abbildung dieser Adresse auf die wirklichen Verbindungen bis hin zum Empfänger ist jeder einzelne Knoten zuständig. Dieser *Routing* genannte Vorgang macht es möglich, alternative Wege für die Verbindung zwischen zwei Knoten anzubieten und im Bedarfsfall andere Wege zu nehmen, ohne dass die darüber liegende Anwendung Kenntnis von diesem Vorgang nimmt. Es ist ebenfalls möglich, die Auswahl an Alternativen für die möglichen Wege auf eine bestimmte Zahl festzuschreiben und bei entsprechender Notwendigkeit die Tabelle mit den Wegen neu zu generieren. Da diese *Rekonfiguration* des Netzwerks sehr aufwändig ist, besteht eines der Ziele von Median darin, die Anzahl und Häufigkeit der Änderung der Konfigurationen in einem sinnvollen Rahmen zu halten. Bei dem Netzwerk beispielsweise, das Median standardmäßig implementiert und das aus 24 Knoten[6], 6 Diensten und 3 möglichen Routen besteht, ist die Betrachtung von $24 \cdot 23 \cdot 6 \cdot 3 = 9.936$ Leitungswegen nötig, was beim Abgleich jeder Verbindung mit den übrigen zu $9.936^2 = 98.724.096$ Berechnungen führt.

Das Konzept von ATM sieht zwei Schichten logischer Netzwerke vor. Das *Virtual Path Network* (VPN) entspricht der obersten Schicht und sorgt für den Transport zwischen den Endknoten. Das VPN muss dabei nicht konstant sein, sondern kann flexibel bei Leitungsausfällen oder Lastschwankungen umkonfiguriert werden. Das darunter liegende *Virtual Channel Network* ermöglicht die Unterteilung der Pfade in kleinere Abschnitte. Die Aneinanderreihung einzelner Channels ergibt somit den kompletten Pfad. Dies ist z.B. dann sinnvoll, wenn der Transport der Daten durch heterogene Netze oder gekapselte Intranets führt. Jedes Teilnetz ist damit für den Transport durch die zugehörige

[5] Cyclic Redundancy Check.

[6] Zu beachten ist, dass Verbindungen nur zwischen den Knoten, nicht innerhalb eines Knotens betrachtet werden, d.h. es existieren $24 \cdot 23$ mögliche Verbindungen.

Region zuständig. Für den Rekonfigurationsvorgang spielt diese Thematik allerdings nur eine untergeordnete Rolle, so dass die Unterteilung der logischen Netze nicht näher betrachtet wird.

Die Übertragung von ATM-Paketen geht im sogenannten *slotted mode* vonstatten. Dabei ist die mögliche Übertragungskapazität in kleine Zeitschlitze (Slots) unterteilt, in denen die Übertragung jeweils eines Paketes stattfinden kann. Durch die Quantisierung der Zeit ist eine einfache Synchronisation zwischen Sender und Empfänger möglich. Der Vorteil von ATM liegt in der dynamischen Belegung der Zeitschlitze. STM[7], wie es beispielsweise im herkömmlichen ISDN[8] eingesetzt wird, bietet eine feste Zuordnung der Quellen zu den verfügbaren Zeitschlitzen. Sendet eine Quelle nicht, wird jeder dieser Quelle zugeordnete Slot trotzdem benutzt und nötigenfalls mit einem Null-Paket aufgefüllt.

Bei ATM kann diese Verschwendung von Bitrate durch die dynamische Belegung der Zeitschlitze vermieden werden. Eine Quelle bekommt auf einer ATM-Verbindung dann Slots zur Verfügung gestellt, wenn Daten gesendet werden sollen. Dies begünstigt burstartige Sender, die für kurze Zeiträume eine recht hohe Datenrate benötigen und in den Zeiträumen dazwischen nur wenige Slots belegen. Damit ergibt sich allerdings eine neue Problematik: Die effektiv auf einer Verbindung zur Verfügung gestellte Bitrate. Die Berechnung und Bestimmung effektiver Bitrate aus Angaben wie maximaler Übertragungsrate und Burstfaktor anhand unterschiedlicher Verfahren war ebenfalls Bestandteil des Median-Projekts ([Scheer99]).

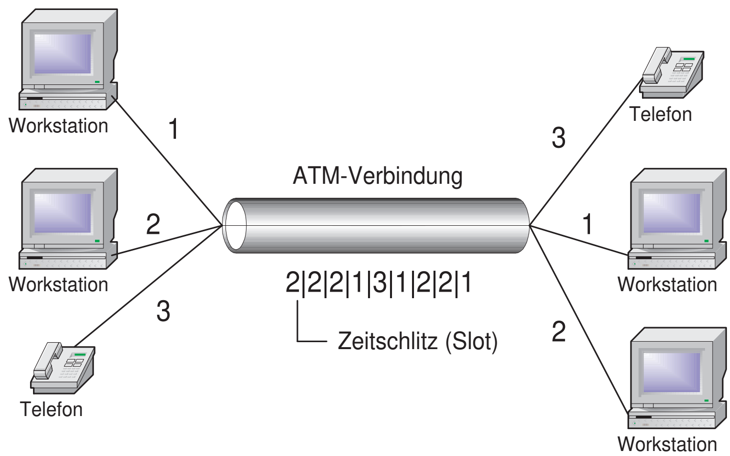


Abbildung 2.1: ATM-Verbindung zwischen mehreren Geräten

Analog ergibt sich die Problematik der Behandlung von QoS[9]-Parametern. Beim Aufbau einer Verbindung muss aus den geforderten Leistungen die Reservierung für die betreffenden Kanäle festgelegt werden.

[7] Synchronous Transfer Mode.
[8] Integrated Services Digital Network.
[9] Quality of Service — Leistung und Anforderungen an bestimmte Verbindungen.

2.3 Median

Das System Median stellt einen Prototypen mit dem Ziel dar, das dynamische Management eines Dimensionierungswerkzeugs für ATM-Netzwerke zu ermöglichen. Unter diesem Gesichtspunkt vereinigt Median eine Reihe von Vorgängen, die dieses Management ermöglichen.

2.3.1 Das Grundkonzept von Median

Ausgangspunkt aller Aktionen ist das zugrundeliegende physikalische Netzwerk. Auf diesen Verbindungen kann das logische Netzwerk aufgebaut und rekonfiguriert werden. Im realen System sind diese Leitungen entweder vorgegeben oder werden entsprechend ökonomischer Gesichtspunkte geplant. Median bietet die Möglichkeit, aus einem fiktiven Szenario heraus ein physikalisches Netz zu dimensionieren. Dieses Szenario ist als Ausschnitt in Abbildung 2.2 dargestellt.

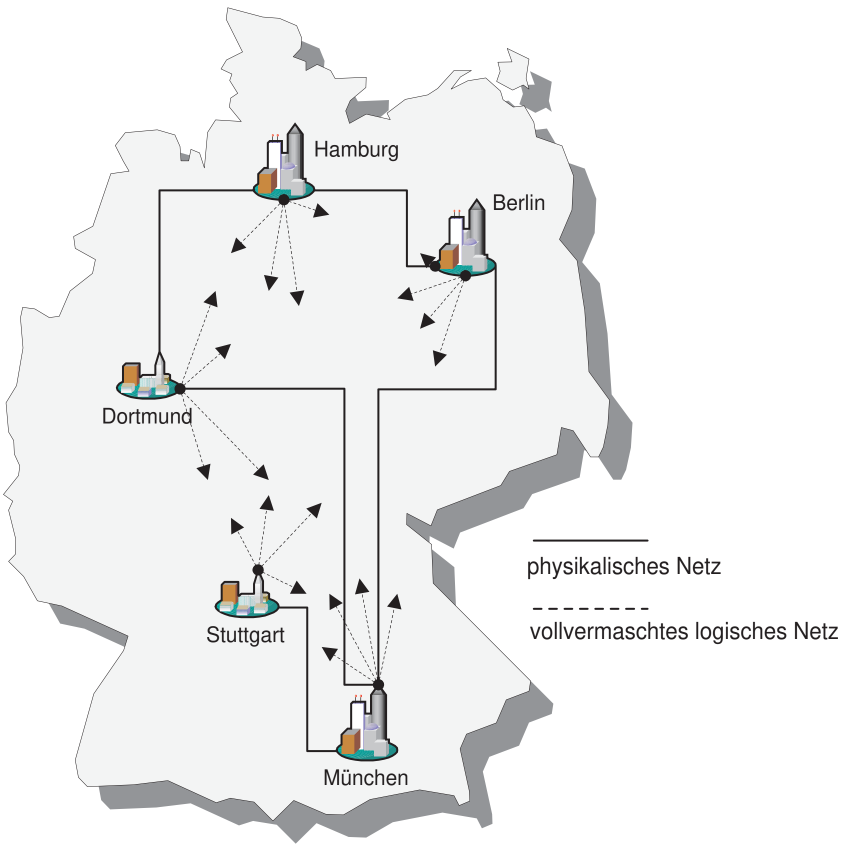


Abbildung 2.2: Das Hintergrund-Szenario von Median

Hintergrund ist der Aufbau eines Netzwerks zwischen verschiedenen deutschen Städten. Die Kapazität der Leitungen muss dabei ähnlich der in der Praxis angewandten Technik so ausgelegt werden, dass von der physikalischen Seite her eine Dimensionierung auf die höchsten zu erwartenden Lasten erfolgt, unter Umständen auch eine größere Dimensionierung, um eine gewisse

Zukunftssicherheit zu gewährleisten. In Median erfolgt die Bestimmung der benötigten Leitungskapazitäten beispielsweise proportional zu den Einwohnerzahlen der jeweiligen Städte, was dem Anwender ein anschauliches Bild der Größe einzelner Leitungen ermöglicht, oder auch durch die direkte Angabe von Lasttabellen.

Während das physikalische Netz aus Kostengründen in der Regel nicht für jedes Paar an Städten eine direkte Verbindung vorsieht, ist das logische Netz vollvermascht, um die Erreichbarkeit jedes Knotens von jedem anderen aus zu gewährleisten. In Abbildung 2.2 ist dieser Aspekt nur angedeutet. Die Verbindungen sind dabei gerichtet, so dass der Verkehr beispielsweise von Dortmund nach Hamburg nicht der gleiche sein muss wie in der umgekehrten Richtung. Die effiziente Verteilung aller Lasten und damit die optimale Ausnutzung des physikalischen Netzes ist die Hauptaufgabe von Median.

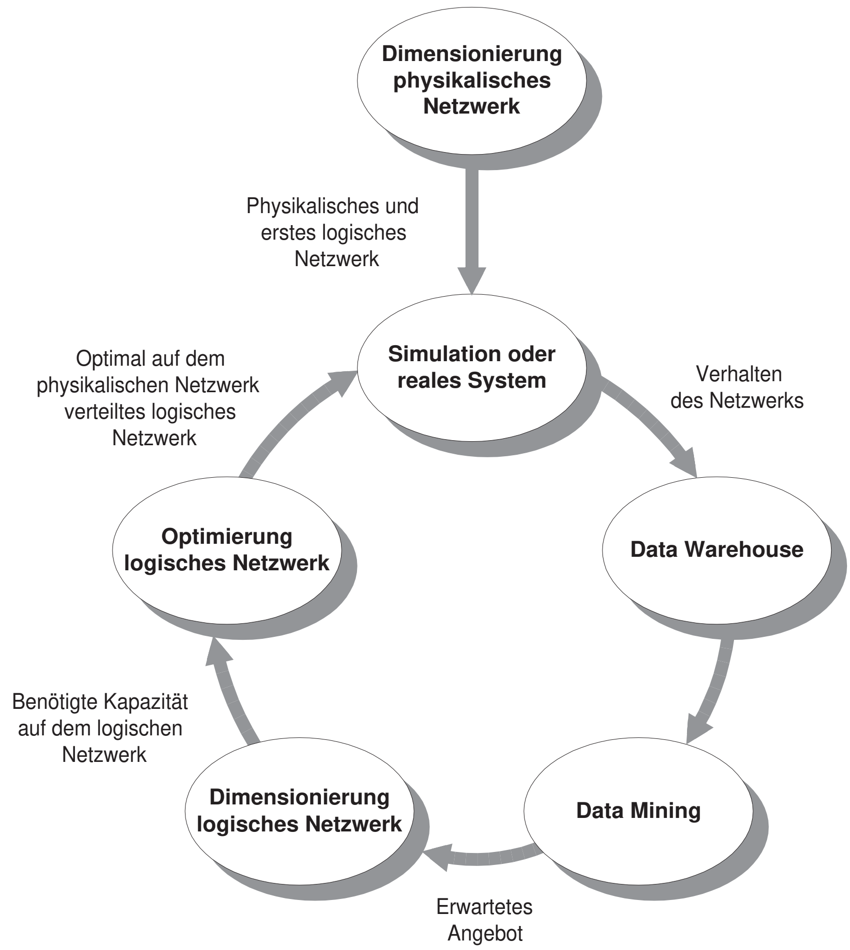


Abbildung 2.3: Lebenszyklus von Median

Der Zyklus, den Median zur Erreichung der Ziele durchläuft, ist in Abbildung 2.3 dargestellt. Der oberste Zustand ist der Startpunkt. Hier erfolgt die oben beschriebene Generierung des physikalischen Netzwerks und eine erste

Erstellung des logischen Netzes.

Die hier berechneten Daten fungieren als Eingabe in den nächsten Zustand, in dem das Netzverhalten aktiv beobachtet wird. Aufgrund vorgegebener Basiskurven werden unterschiedliche Verbindungen zwischen allen Knoten simuliert. Im Hinblick auf den Praxiseinsatz ist dies die Stelle, an der sich die Schnittstelle zu Messwerten aus den Vermittlungsstellen befindet. Durch die Überwachung der erzeugten Lasten lassen sich die wichtigsten Parameter Angebot, Verkehr und Blockierung für jeden einzelnen Link ermitteln. Befindet sich das System in einem stabilen, unkritischen Stadium, bleibt die Ausführung von Median grundsätzlich in diesem Zustand. Die Überwachung des Netzes geschieht mittels der sogenannten *Rekonfigurationszahl*. Diese stellt eine nach der Größe der Blockierung gewichtete Summe über alle Links dar. Dabei wird eine zu hohe Blockierung, welche zur Folge hat, dass der Verbindungsaufbau wegen zu vieler belegter Leitungen häufiger scheitert, genauso negativ bewertet wie eine geringe Blockierung, die auf eine Unterlast im Netzwerk hindeutet.

Der nächste Zustand im Zyklus deutet eine bestmögliche Art der Verwaltung der Daten an. Hinter dem Begriff *Data Warehouse* verbirgt sich eine Datenbank, die schnellen Zugriff auf die Netzdaten ermöglicht, so dass Abfragen (*Query*) entlang mehrerer Dimensionen und Attribute effizient erfolgen können, ohne bereits bei der Erstellung die genaue Art der Abfragen kennen zu müssen. Für den Prototypen Median verbirgt sich an dieser Stelle lediglich eine MS-Access-Datenbank für die Eingabe der Daten. Enthalten sind Informationen über das Netzwerk (Anzahl, Namen und Größe der beteiligten Städte) sowie Angebotskurven für die Änderungen der Last im Verlauf eines Tages. Die Ergebnisse der Simulation werden aufgrund der Größe von bis zu mehreren GByte (Access besitzt eine Grenze von 1 GByte) und der einfachen systemunabhängigen Verarbeitung in Textdateien gehalten, die den Netzzustand entsprechend des eingestellten Ausgabeintervalls chronologisch widerspiegeln und damit schnell sequenziell entlang der Zeitachse verarbeitet werden können.

Wird während des Betriebs ein schlechtes Netzverhalten anhand einer zu hohen Rekonfigurationszahl festgestellt, muss dementsprechend reagiert werden. Dazu tritt das Netz in eine Rekonfigurationsphase ein, in der versucht wird, eine optimale/hinreichend optimale Verteilung des logischen Netzes auf das physikalische zu erreichen. Zwei Parameter sind an dieser Stelle kritisch: Erstens muss die Rekonfiguration ausreichend schnell abgeschlossen sein, um die durch das Umlegen von Routen verursachten zusätzlichen Störungen auf ein Minimum zu reduzieren. Zweitens muss gewährleistet sein, dass die Rekonfiguration das Netz für einen hinreichend großen Zeitraum in einen stabilen Zustand versetzt. Ist der Abstand zwischen zwei Rekonfigurationsphasen annähernd Null oder wird eine neue Rekonfiguration des Netzes bereits zu einem Zeitpunkt angestoßen, während die aktuelle noch ausgeführt wird, gerät das Netzwerk in Schwingung. Unter Umständen steigen die Blockierungen über kritische Bereiche. Viele derzeit in Vermittlungsstellen eingesetzte Systeme besitzen Mechanismen zur Abwehr von Überlast, die in der Regel auf der Abschaltung von Teilnetzen beruhen, bis die Last wieder auf Normalniveau abgesunken ist. Baut sich in einem Netzwerk durch Schwingung eine hohe Last auf, kann es bis zum Ausfall des vollständigen Netzwerks kommen.

Im Betrieb mit Median hat sich ein Zeitraum von 3 Stunden als Horizont für die neue Konfiguration des Netzwerks als günstig erwiesen. Um für diesen Zeitraum den stabilen Betrieb zu gewährleisten, ist es notwendig, eine Aussage über die maximalen Lasten auf jeder Verbindung treffen zu können. Genau an dieser Stelle setzt diese Diplomarbeit an. Im bisherigen Betrieb von Median sind die Vorhersagen für das maximale Angebot durch Anfragen auf den Basistabellen für die Generierung des Angebots erfolgt. Diese Tabellen dienen als Eingabe für den Simulator und enthalten die durchschnittlichen Angebotswerte jeder Stunde eines Tages, wie es im simulierten Netzwerk auftritt. Durch Anfragen an diese Tabelle sind somit auch alle durchschnittlichen zukünftigen Angebote bekannt, eine Information die in der Realität selbstverständlich nicht vorliegt, weswegen diese Methode nicht anwendbar ist.

Eine Prognose des zukünftigen maximalen Angebots muss deshalb lediglich auf Basis der Vergangenheitswerte erfolgen. Um diese Vorhersage zu ermöglichen, werden unterschiedliche Techniken aus dem Bereich des *Data Mining* auf den Daten des Simulators angewendet.

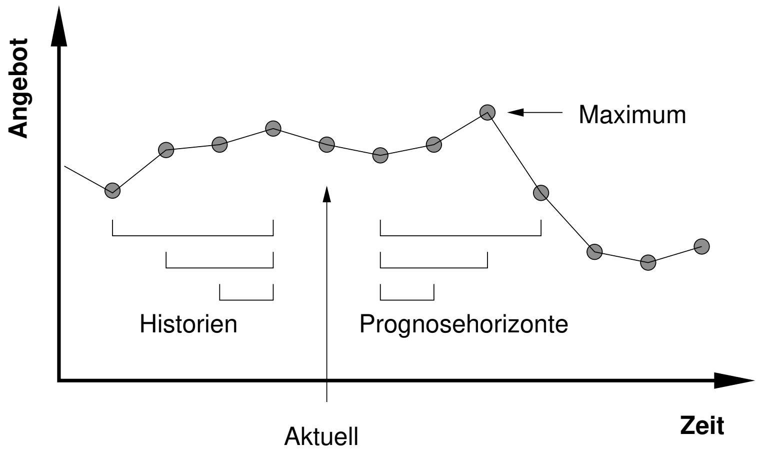


Abbildung 2.4: Beispiel einer Angebotskurve

Dieser Vorgang ist in Abbildung 2.4 dargestellt. Aus den Vergangenheitsdaten und dem entsprechenden Kurvenverlauf über die einzelnen Taktzeitpunkte wird versucht, das maximale Angebot über einen bestimmten Zeitraum zu finden. Die Prognose dient als Eingabe für die Optimierung. Ziel der Optimierung ist es, eine Abbildung des logischen auf das physikalische Netzwerk zu finden, die den Vorgaben hinsichtlich der Güte des Netzes genügt, d.h. die Auslastung hoch, die Blockierung dagegen gering hält. Dazu wird ein Mutations-Selektionsverfahren benutzt, das in [Scheer00] beschrieben ist. Vorteil dieses Verfahrens ist, dass die Optimierung jederzeit abgebrochen werden kann, was trotzdem in einem gültigen Ergebnis resultiert. Dieses Ergebnis wird in den wenigsten Fällen optimal sein, aber meist den Ansprüchen zwischen hinreichend genauer Berechnung des neuen Netzes und zeitlichem Rahmen der Berechnung genügen.

Das neue Netz wird anschließend wieder für die Betrachtung des System-verhaltens herangezogen und der Zyklus beginnt von vorn.

2.3.2 Implementierung von Median

Die konkrete Umsetzung des bislang beschriebenen Lebenszyklus des Netzwerkmanagements spaltet sich wie bereits angedeutet in zwei Systeme auf: Die eigentliche Managementsoftware Median und die Simulationssoftware ATMSim.

Die Programmierung des Systems erfolgt vollständig mit Visual C++ V6.0 unter MS Windows NT/2000.

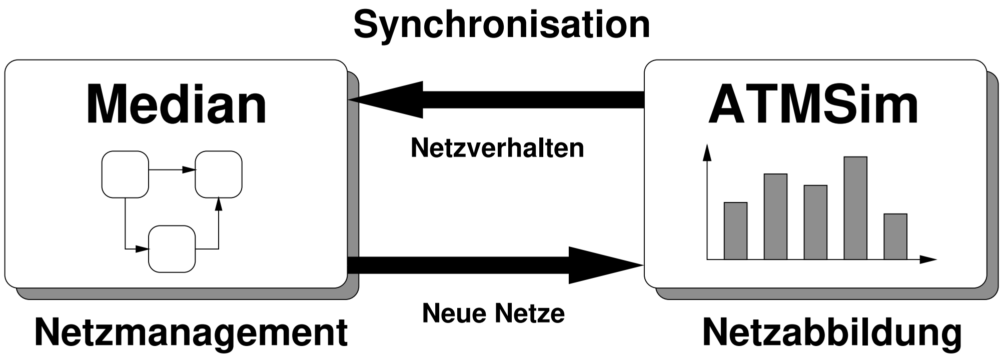


Abbildung 2.5: Zusammenhang zwischen Median und der ATM-Simulation

Die Simulation übernimmt die Abbildung des realen Netzwerks und stellt die Eingangsdaten für das Management dar. Parameter stellen die Anzahl der Durchläufe (gleichbedeutend mit Tagen) das Ausgabeintervall, die Anzahl verfügbarer Routen, die Anzahl der Dienste und die Gesamtlaufzeit der Simulation dar. Die Angebotswerte, die als Durchschnitt für ein Zeitintervall von der Simulation eingehalten werden müssen, werden durch Einträge in der zugehörigen Datenbank vorgegeben. Die Simulation der Belegungen erfolgt ereignisgesteuert. Jede Belegung wird durch ihr Start- und Endeereignis bestimmt, die wiederum durch Anrufrate und Haltedauer charakterisiert werden. Zu jedem Ausgabeintervall erfolgt die Speicherung der aktuellen Belegungen für jede Verbindung.

Das Programmpaket Median übernimmt bei dem im vorhergehenden Abschnitt vorgestellten Zyklus alle Managementaufgaben. Die Vorhersage zukünftiger Angebotswerte durch Data Mining-Techniken wird dementsprechend in dieses Paket integriert.

Zum Austausch der Daten synchronisieren sich Median und die ATMSim gegenseitig. Nach jeder Ausgabe der Simulation werden die Daten von Median kontrolliert und es wird geprüft, ob die Bedingungen für einen optimalen Betrieb des Netzes noch gegeben sind. Ist dies nicht der Fall, wird eine Vorhersage über den maximalen Netzzustand der nächsten drei Stunden eingeholt und das Netzwerk rekonfiguriert. Das neue Netz wird an die ATMSim zurückgegeben und damit weiter simuliert. Zu beachten ist, dass die Umschaltung zwischen neuem und altem logischen Netz nicht sofort erfolgen kann, da dies in den meisten Fällen durch alternatives Routing zum Abbruch bestehender Verbindungen und damit zum Datenverlust führt. Vielmehr werden die Differenzen zwischen aktueller und benötigter Kapazität auf einem Link betrachtet und im Falle einer Verkleinerung keine neuen Rufe zugelassen. Sind genügend Rufe abgebaut, d.h. beendet, kann der Link seine neue Größe zugewiesen bekommen. Prinzipiell ist es zwar möglich, bestehende Verbindungen während des laufenden Betriebs umzuschalten, die entsprechenden Verfahren erfordern allerdings entsprechende Reservekapazitäten und einen höheren Steuerungsaufwand, so dass dies in den meisten Fällen nicht praktiziert wird.

Neu ankommende Rufe werden auf alternativen freien Wegen durchgeschaltet. Median besitzt dazu die Fähigkeit, bis zu drei Routen pro Verbindung zwischen zwei Endknoten auszulasten und den sogenannten *Bündelungsgewinn* zu nutzen. Für jede Alternativroute kann eine Wahrscheinlichkeit angegeben werden, mit der diese Route bei eintreffenden Rufen benutzt wird. Dies sorgt für eine gleichmäßigere Auslastung der Wege, eine höhere Sicherheit bei Ausfällen, da nicht alle Datenströme gleichzeitig betroffen sind, und erhöht die Wahrscheinlichkeit, dass ein Link, der verkleinert werden muss, eher seine neue Größe erhalten kann.

Im nächsten Kapitel erfolgt eine allgemeine Einführung in die Thematik des Data Mining und eine Übersicht über Anwendungsgebiete und Verfahren.

Kapitel 3

Data Mining

In diesem Kapitel wird der Begriff und die zugrundeliegende Problematik des *Data Mining (DM)* dargestellt. Ziel ist es, eine Einführung in das Gebiet und einen Überblick über einzelne Bestandteile zu geben. Weitere tiefergehende Einführungen in die Thematik sind z.B. in [Holsheimer94] und [Fayyad96] zu finden, das erste Dokument beinhaltet eine Übersicht über unterschiedliche Verfahren des Data Mining und ihre konkreten Ausprägungen, das zweite neben einem allgemeinen Teil spezialisierte Aufgaben sowie realisierte Projekte.

Abschnitt 3.5 in diesem Kapitel beschreibt eine allgemeine Möglichkeit zur Vorgehensweise beim Data Mining in Form eines Prozessmodells.

3.1 Begriffsklärung

Data Mining oder auch *Knowledge Discovery in Databases (KDD)* beschreiben eine Klasse von Techniken, die dann zum Zuge kommen, wenn die Verarbeitung großer Datenmengen und die Extraktion gewünschter, teilweise auch vorher nicht bekannter, Merkmale nötig wird und herkömmliche intuitive oder statistische Methoden keine zufriedenstellenden Ergebnisse mehr liefern können.

In vielen Bereichen des täglichen Lebens haben sich große Datenmengen angesammelt (bzw. sammeln sich immer noch an), in denen verstecktes „Wissen" in Form unbekannter Beziehungen zwischen einzelnen Objekten der Datenbanken vorhanden ist. Populär geworden sind DM und KDD durch Anwendungen in betriebswirtschaftlichen (Direktmailing, Warenkorberstellung...) oder medizintechnischen (Diagnoseunterstützung, Eskalationsprävention...) Bereichen. Beispielsweise ließe sich die Effizienz von Direktmailingaktionen erheblich steigern, wenn vor Versendung der Angebote bekannt wäre, ob der Empfänger zum potenziellen Kundenkreis gehört. Durch Erstellung von Testmailingaktionen an bestimmte Kunden, Protokollierung der positiven und negativen Reaktionen und Verknüpfung mit Daten über die jeweiligen Kunden (Einkommen, Wohngebiet...) werden Charakteristika für mögliche Reagierer erstellt. Hintergrund sind an dieser Stelle Kostenreduktion und Gewinnmaximierung ([Kietz00]).

Eines der verbreitetsten Beispiele für die Erfolge von Wissensentdeckung steht im Zusammenhang mit der Warenkorbanalyse: Eine Untersuchung von Verkaufszahlen mehrerer Supermärkte hatte ergeben, dass kurze Zeit nachdem

Windeln gekauft wurden die gleichen Personen mit hoher Wahrscheinlichkeit direkt vor dem Wochenende ebenfalls Bier einkauften. Diese verblüffende und nicht auf den ersten Blick sichtbare Relation hat einige Zeit die unterschiedlichsten Diskussionsforen beschäftigt, sich aber letztendlich als Werbeaktion zur Vermarktung eines Data Mining-Werkzeugs offenbart.

Mit diesem Beispiel soll nicht die Leistungsfähigkeit von Algorithmen und Werkzeugen des DM bestritten werden, in vielen anderen Fällen konnten Beziehungen zwischen Faktoren gefunden werden, die nicht sofort offensichtlich waren. Aufmerksam gemacht werden soll dagegen auf eine der Gefahren des Data Mining: Repräsentativität und Aussagekraft der Schlussfolgerungen.

Das DM stellt mächtige Techniken zur Entdeckung von Mustern innerhalb der Daten zur Verfügung, anschließend muss allerdings jeweils die Plausibilität und der Ursprung der Muster hinterfragt werden.

3.2 Zielsetzung

Allgemeines Ziel des Data Mining ist es, ein Modell des zu untersuchenden Systems und seiner Umwelt zu erstellen, das Rückschlüsse auf Beziehungen zwischen unterschiedlichen Zuständen des Systems ermöglicht ([Holsheimer94]).

Definition 1
Ein *System* kann als Zustandsgraph aufgefasst werden, der aus zwei Mengen (S, T) besteht. S ist die Menge aller möglichen Zustände (States) und T (Transitionen) regelt die Übergänge $T : S \rightarrow S$ zwischen den Zuständen. Somit wird durch T der Übergang für jeden Zustand S_t in den Folgezustand S_{t+1} festgelegt. Die Übergänge zwischen den Zuständen können durch Einflüsse innerhalb oder Reize aus der Umgebung des Systems ausgelöst werden.

Der einfachste Weg ein Modell des zu beschreibenden Systems zu erhalten, wäre eine exakte Kopie der Zustände mit den zugehörigen Transitionen zu erstellen. Der naive Ansatz erfasst die Menge aller möglichen Zustände eines Systems und zeichnet die Transitionen zwischen den Zuständen auf. Um eine Aussage über den Folgezustand zu erhalten, wird das reale System mit der Datenbank der aufgezeichneten Zustände verglichen und daraus der folgende ausgegeben. Dieser Ansatz ist allerdings nur für überschaubare Systeme erfolgreich. Punkte, die gegen diese Art der Vorgehensweise sprechen, sind neben dem immensen Speicherbedarf die geringe Wahrscheinlichkeit, dass in realen Systemen ein späterer Zustand genau mit einem der Zustände innerhalb der Datenbank übereinstimmt oder alle möglichen Zustände erfassbar sind.

Im DM läuft deshalb die Erstellung eines Modells vielmehr auf die abstrahierte Überführung von in den Daten gefundenen Mustern in eine interne Darstellung hinaus. Das reale System stellt damit den Spezialfall dar und das DM-Modell versucht eine allgemeinere Beschreibung (*Generalisierung*) der Zusammenhänge im System. Benutzt wird dazu ein Teilausschnitt der jeweiligen Zustandsmenge. Die Vorhersage neuer Zustände erfolgt dann durch Vergleich der abstrakten Repräsentationen des realen Systems.

Im maschinellen Lernen wird dabei nicht von einer einzelnen Beobachtung oder Aussage über das zu untersuchende System ausgegangen, sondern vielmehr von einer endlichen Menge an Aussagen (*training set*).

Definition 2
Sei $A = \{a_1, \ldots, a_n\}$ eine Menge von Attributen mit den zugehörigen Wertebereichen $\mathbb{D}_1, \ldots, \mathbb{D}_n$. Ein *Trainingsdatensatz* T ist dann eine Menge möglicher Kombinationen c_i dieser Attribute. Jede dieser Kombinationen $c_i \in T$ wird als *Beispiel* bezeichnet. Die Gesamtheit aller möglichen Attributkombinationen $U = \mathbb{D}_1 \times \ldots \times \mathbb{D}_n$ heißt *Universum*. Der Trainingsdatensatz ist eine endliche Untermenge des Universums, $T \subset U$.

Da die Trainingsmenge endlich ist, können nicht alle Konzepte exakt gelernt werden. Bei bestimmten Problemen ist es aufgrund von Unentscheidbarkeit einzelner Aussagen selbst bei unendlich großen Trainingssätzen nicht möglich, eine exakte Lösung zu erhalten. In solchen Fällen muss eine angenäherte Lösung ausreichen, die die Kriterien hinsichtlich ihrer Genauigkeit erfüllt.

Desweiteren stellen die Trainingssätze in der Regel nur einen Teilbereich der möglichen Beobachtungen dar. Damit ist es möglich, mehrere unterschiedliche Modelle zu konstruieren, die für die gegebenen Beispiele eine korrekte Lösung anbieten. Korrektheit bedeutet, das Modell ist in der Lage alle Folgezustände für einen bereits bekannten vorauszusagen. Ziel ist allerdings, auch alle zukünftigen und unbekannten Zustände korrekt einordnen zu können. Eine vollständige Verifikation lässt sich nur in den wenigsten Fällen durchführen, da die Anzahl der möglichen Zustände meist nicht endlich ist. Auch hier muss auf eine Schätzung der Qualität des berechneten Modells zurückgegriffen werden.

3.3 Lernen

Zu Beginn jedes Modellbildungsprozesses existieren weder eine Repräsentation der möglichen Zustände noch die Transitionen dazwischen. Der Vorgang der Modellbildung wird als *Lernen* bezeichnet.

Wesentlicher Aspekt und Kerngegenstand der Forschung ist dieser Vorgang zur Extraktion des Wissens aus den gegebenen Datensätzen. Dieses Wissen kann je nach Aufgabenstellung in diskreter (Entscheidungsbäume...) oder kontinuierlicher (Regressionsverfahren...) Form präsentiert werden. Diskrete Ergebnisse werden dabei durch *Klassen* dargestellt, die den einzelnen möglichen Werten zugeordnet werden.

Definition 3
Eine *Klasse* ist eine Untermenge der Trainingsdatensätze und wird durch eine symbolische Repräsentation (Bezeichner, Zahl...) geprägt.

Definition 4
Die Menge aller Klassen C enthält einzelne Klassen C_i, denen eine eindeutige *Klassenbeschreibung* (class description) D_i zugeordnet ist. Aufgrund dieser Beschreibung ergibt sich eine Klassifizierungsfunktion f mit $f : S \rightarrow C$, die einen

Zustand S des betrachteten Systems auf eine Klasse C_i abbildet, wenn dieser der Beschreibung D_i genügt.

Prinzipiell ist es problemlos möglich, kontinuierliche Werte durch diskrete Klassifikation darzustellen; hierbei stellt sich allerdings die Frage nach der nötigen Anzahl an Klassen und damit der Genauigkeit. Eine „künstliche" Klassifikation bedeutet eine Verkleinerung des möglichen Wertebereichs an Lösungen. Zusätzliche Probleme bereiten Fälle, die sehr nah an der Grenze zwischen zwei Klassen liegen und bereits durch geringe Schwankungen von einer Klasse in die nächste wechseln können.

Der Vorgang des Lernens lässt sich in zwei Kategorien aufteilen: *Überwachtes* (supervised) und *unüberwachtes* (unsupervised) Lernen. Im überwachten Lernen wird der Vorgang der Modellierung durch Bereitstellen von Beispielen mit zugeordnetem Ergebnis unterstützt. Voraussetzung hierzu ist, dass sich für Daten des realen Systems das dazugehörige und damit das im Vorhersagefall zu bestimmende Ergebnis[1] den Werten zuordnen lässt. Aufgabe des Lernverfahrens ist es, aus diesen Beispielen charakteristische Muster und Regelmäßigkeiten für die Ergebnisse zu erkennen und in das interne Modell umzusetzen.

Im unüberwachten Lernen werden analog reale Beobachtungen über das System als Trainingsdaten vorgegeben. Den Datensätzen sind in diesem Fall dagegen keine Klassen oder Ergebnisse zugeordnet. Aufgabe des Lernverfahrens ist es, nur aufgrund dieser Daten Muster zu entdecken, die als Klassenbeschreibung herangezogen werden können um daraus selbst Klassen zu bilden. Der Vorgang führt somit zu einer abstrahierten Zusammenfassung der Muster, die allen Objekten innerhalb der Daten zugrunde liegen.

Da im konkreten Fall von Median die vorherzusagenden Ergebnisse — die Maxima der zukünftigen Intervalle — für die Trainingsdaten durch Betrachtung der aufgezeichneten Lastkurven bekannt sind, kann für die Lösung des Prognoseproblems hier auf überwachtes Lernen zurückgegriffen werden.

Um einen Eindruck von der Komplexität des Lernproblems zu vermitteln, wird an dieser Stelle ein Beispiel für den Fall überwachten Lernens mit vorgegebenen Klassen gegeben. Der naive Lernansatz geht von einer vollständigen Suche über den Raum der möglichen Beschreibungen für die einzelnen Klassen aus.

Für jede mögliche Beschreibung D_i wird dementsprechend untersucht, ob sich aus den Trainingsdaten eine Bestätigung für die Annahme einer Abbildung von D_i auf eine Klasse C_i: $D_i \to C_i$ finden lässt. Für eine vollständige Suche besteht der Thesenraum für eine Menge von $A = \{a_1, \ldots, a_n\}$ Attributen mit den jeweiligen Wertebereichen (Domänen) $Dom(a_i)$ aus der Kombination aller möglichen Attributausprägungen ([Holsheimer94]):

$$2^{(\prod_A |Dom(a_i)|)} - 1.$$

Zu beachten ist, dass der Aufwand nicht von der Anzahl der Daten abhängt, sondern von der Anzahl der Attribute und der Größe ihres Wertebereichs.

[1] z.B. Luftdruckwerte, Feuchte, Temperatur und als Ergebnis das Wetter, welches sich nach der Messung dieser Werte ergeben hat.

Zusätzlich wurde vorausgesetzt, dass keine widersprüchlichen oder fehlerhaften Beschreibungen existieren.

Data Mining-Verfahren reduzieren den Aufwand durch Verkleinerung des Suchraums möglicher Lösungen. Ansätze hierzu beruhen beispielsweise auf der Bewertung von Beschreibungen mittels spezieller Metriken und der Auswahl der vielversprechend erscheinenden Lösungen.

Verschiedene populäre DM-Verfahren, die in [Fayyad96] angesprochen werden, sind:

- *Entscheidungsbäume und Regeln* : In diesen Bereich gehören klassische Verfahren (z.B. C4.5, [Quinlan92]), die auf Basis von Unterteilungen der Daten anhand einzelner Attribute arbeiten. Die benutzten Methoden sind entsprechend einfach und schnell, berücksichtigen aber keine Relationen zwischen den Attributen, sondern trennen die Daten meist parallel zu den einzelnen Dimensionsachsen.

- *Nichtlineare Regressions- und Klassifikationsverfahren*: Diese Techniken benutzen verschiedene lineare und nichtlineare Kombinationen von Basisfunktionen in Abhängigkeit der Eingabeparameter zur Vorhersage. Typische Beispiele sind neuronale Netze oder Support Vector Machines (siehe auch [Vapnik98]).

- *Beispiel-basierte Methoden*: Hier werden die Trainingsdaten als Beispiele angesehen, wobei zur Prognose neuer Daten Ähnlichkeiten zu diesen Beispielen gesucht werden. Das Beispiel, dessen Ähnlichkeitsmaß am größten ist, wird als repräsentativer Vertreter für den zu bestimmenden Datensatz angenommen und das zugeordnete Ergebnis, bzw. die Klasse, ausgegeben. Größtes Problem bei dieser Methode ist die Bestimmung von Distanzmaßen, die den Grad der Ähnlichkeit zweier Datensätze angeben. Beispiel dieser Methoden ist das nearest-neighbour-Verfahren ([Wettschereck92]).

- *Grafische Modelle*: Diese Modelle ermöglichen die anschauliche Darstellung von Abhängigkeiten der Eingabevariablen untereinander. Grafische Strukturen stellen Beziehungen zwischen den unterschiedlichen Zuständen und die Wahrscheinlichkeiten für die Abhängigkeiten dar. Wegen der leichteren Interpretierbarkeit der Modelle hat diese Klasse von Verfahren sehr häufig in Expertensystemen Anwendung gefunden ([Borgelt98]).

- *Relationale Lernverfahren*: In diesem auch *inductive logic programming (ILP)*, [Morik96], genannten Bereich kommen komplexe Beschreibungssprachen wie *first-order logic (FOL)* zum Einsatz. Ziel ist es, die Begrenzungen durch fixe Beschreibungssprachen zu umgehen und Vorwissen über die Attribute in den Modellbildungsprozess mit einzubeziehen. Ergebnis ist ein FOL-Programm, das als logische Konsequenz die Trainingsdaten ergibt.

3.4 Probleme

Die bislang vorgestellten Definitionen haben in den meisten Fällen vorausgesetzt, dass sich eindeutige Beschreibungen und Muster innerhalb der Trainingsdaten zur Bestimmung der Ergebnisse finden lassen. Für nach bestimmten Verteilungen künstlich generierte Testdaten mag diese Annahme zutreffen, in realen Anwendungen dagegen wird dies häufig nicht erfüllt sein. An dieser Stelle wird auf einige möglicherweise auftretende Probleme hingewiesen.

Zunächst die Frage nach dem Umfang der Daten: Bei vielen Attributen mit jeweils sehr großem oder unbeschränktem Wertebereich ergibt sich zwangsläufig ein unüberschaubarer Raum möglicher Zustände. Demzufolge können die Trainingsdaten in den meisten Fällen nur einen Ausschnitt widerspiegeln und besitzen unter Umständen nicht den für exaktes Lernen nötigen Informationsgehalt. Dies betrifft die Repräsentativität der Daten, d.h. die enthaltenen Muster sollen den Schluss auf die Abhängigkeiten der Attribute untereinander erlauben, ohne z.B. durch unglückliche Verteilung der Klassen innerhalb der Datensätze ein verzerrtes Bild der Realität zu erzeugen (Auftreten von Klassen in deutlich höherer Konzentration als im realen System, falsche Verteilungen...). Ebenso kritisch sind die Daten, die bereits bei kleinen Schwankungen der Parameter große Änderungen der Ergebnisse/Klassifikation nach sich ziehen. Diese Beispiele bestimmen wesentlich Grenzen für die Klassifikation. Je besser sich Ergebnisse anhand der sie bestimmenden Attribute eingrenzen lassen, um so besser kann eine Bewertung neuer Daten erfolgen. Abhilfe bei diesen Problemen versprechen Verfahren wie *Sampling* oder *Feature Selection*, die als Vorverarbeitungsschritt die Trainingsdaten aufbereiten und z.B. repräsentative Beispiele auswählen.

Ein weiteres Problem betrifft nicht die Auswahl der Daten aus der möglichen Menge, sondern die Korrektheit der Daten in sich. Datenbanken, die durch menschliche Eingabe entstanden sind (z.B. mikrogeografische Datenbanken mit Kundendaten für das Direktmailing), enthalten häufig Eingabefehler, doppelte Datensätze oder hängen von subjektiven Einschätzungen ab. Analog können Daten, die durch automatisierte Vorgänge gewonnen wurden, Messfehler und Ungenauigkeiten — beispielsweise durch Bauteiltoleranzen — enthalten. Nicht systematische Fehler dieser Art führen zum sogenannten *Rauschen*.

An zwei Stellen führt Rauschen zu verschlechterten Ergebnissen: erstens bei der Modellbildung. Werden die Daten zu stark verzerrt, können vorhandene Muster in den Daten nicht mehr erkannt werden und die Vorhersagequalität des Modells sinkt. Zweitens erfolgt eine Verschlechterung bei der Bewertung unbekannter Daten. Es ist dabei Aufgabe des DM-Verfahrens, ein soweit generalisiertes Modell zu entwickeln, das auch bis zu einem gewissen Grad verzerrte Daten korrekt bewerten kann.

Teilweise fehlende Daten für einzelne Attribute erfordern eine spezielle Behandlung. Die einfachste Möglichkeit ist, unvollständige Beispiele im Datensatz nicht zu benutzen. Trifft dies jedoch auf relativ viele der Beispiele zu, ergibt sich wieder die Problematik der Repräsentativität. Andere Methoden versuchen, fehlende Attributwerte sinnvoll — z.B. entsprechend der Häufigkeit ihres Auftretens — zu ersetzen.

Abschließend sei hier ein Punkt erwähnt, den viele DM-Methoden im Gegensatz zu speziell für die Zeitreihenanalyse ausgelegten statistischen Verfahren aufweisen: die Repräsentation der Zeit innerhalb der Lerndaten und des Modells. Selten besteht die Möglichkeit für eine explizite Berücksichtigung der Zeit, so dass diese implizit in die Trainingsdaten, z.B. durch Darstellung von Kurvenverläufen (siehe auch Kapitel 4.4), eingearbeitet werden muss.

3.5 Prozessmodell für das Data Mining

Dieser Abschnitt bezieht sich auf das *Cross Industrial Software Process Model for Data Mining (CRISP-DM)*, nachzulesen in [Crisp99]. Gedacht ist das Prozessmodell als gebietsübergreifende Richtlinie und Entscheidungshilfe bei der konkreten Durchführung von umfangreichen Data Mining-Projekten. Die grundlegenden Phasen lassen sich allerdings in jedem DM-Projekt wiederfinden.

Die Vorgehensweise beginnt beim generalisierten Modell. Die allgemeinen Richtlinien im CRISP-DM-Dokument müssen an die speziellen Anforderungen der Aufgabe angepasst werden. Konkret bedeutet dies, die Umwelt der Aufgabe zu analysieren, ungeeignete Details entsprechend aus dem generalisierten Modell zu entfernen und die generischen Kontexte mit Inhalten zu instanziieren. Auf dem spezialisierten Prozessmodell setzt dann der Zeitplan zur Durchführung auf, der ebenfalls die Verteilung der Kapazitäten (Personal, Material...) regelt.

Abbildung 3.1 zeigt den Lebenszyklus eines DM-Projekts mit sechs einzelnen Phasen und den Beziehungen dazwischen. Die Phasen beinhalten die folgenden Aufgaben:

- *Business Understanding:* Die erste Phase beinhaltet die Problemanalyse. Die zu erfüllenden Anforderungen und das zu modellierende System müssen untersucht und verstanden werden. Dies wird durch die Einarbeitung in Median und die verkehrstheoretischen Grundlagen (Kapitel 2) realisiert. Als Ergebnis dieser Phase ergeben sich die Zieldefinitionen.

- *Data Understanding:* Hier werden die ersten Daten und Kenntnisse über ihre Art (Wertebereiche, kontinuierlich oder diskret...) gesammelt. Probleme, die möglicherweise später auftreten, sollen hier identifiziert werden (z.B. Daten, die während des Einschwingvorgangs des Netzes gesammelt wurden und deshalb nicht den Normalfall widerspiegeln).

- *Data Preparation:* Diese Phase bereitet die gesammelten Daten auf und stellt eine von den DM-Werkzeugen verwertbare Darstellung her. Dies beinhaltet sowohl die syntaktisch korrekte Form, als auch unterschiedliche Repräsentationen (Sortierung oder Generierung von Attributen...). Probleme innerhalb der Daten (Inkonsistenz, Widersprüche, Lücken...) werden in dieser Phase behoben.

- *Modeling:* An dieser Stelle kommen die eigentlichen DM-Werkzeuge zum Einsatz und werden mit den Daten aus der vorhergehenden Phase versorgt. Dementsprechend besteht zwischen dieser Phase und der vorheri-

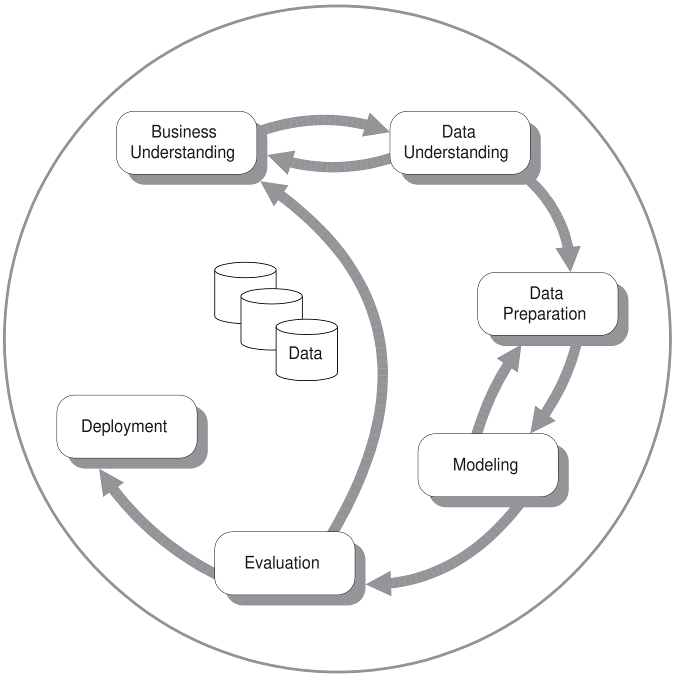


Abbildung 3.1: Die Phasen des CRISP-DM Modells

gen ein enger Zusammenhang und häufig erzwingen die Ergebnisse der Modellbildung neue Datenrepräsentationen. Modeling und Data Preparation finden sich gemeinsam in den Kapiteln 4 und 5 wieder.

- *Evaluation:* Die in der Phase Modeling gefundenen Parameter werden hier evaluiert. Für die besten Modelle erfolgt eine Bewertung entsprechend der in der ersten Phase aufgestellten Zielsetzung. In diesem Projekt erfolgt dies durch Kapitel 6, wo die Modelle miteinander verglichen und anhand von Tagesprognosen bewertet werden.

- *Deployment:* Die abschließende Phase integriert das endgültige Modell in das bestehende System (Kapitel 7). Die spezielle Data Mining-Umgebung wird in eine an die Systemumgebung und den Nutzer angepasste Version überführt. Dazu gehört sowohl die Implementierung der Software als auch die Erstellung des Abschlussberichts.

Die nächsten Kapitel beschäftigen sich mit dem konkreten Einsatz von Data Mining-Verfahren aus der Gruppe der in diesem Kapitel angesprochenen Methoden auf den Daten von Median.

Kapitel 4

Entscheidungsbäume

Dieses Kapitel beschreibt die Grundlagen des *C4.5*-Verfahrens von J.R. Quinlan. Entwickelt als Nachfolger des Systems ID3 stellt es eine Kombination unterschiedlicher Algorithmen mit dem Ziel dar, auf Basis vorgegebener Daten ein Modell zu entwickeln. Dieses Modell soll dem Anwender die Möglichkeit geben, bestimmte Aus- oder Vorhersagen über die dem Modell und den Daten zugrunde liegende Umwelt zu finden. Im folgenden Abschnitt wird die formale Darstellung des Modells (*Entscheidungsbäume*) und die Konstruktion mittels C4.5 erläutert. Die Beschreibung beschränkt sich dabei auf die für diese Diplomarbeit wesentlichen Teile, für einen vollständigen Einblick in C4.5 wird auf die zugehörige Dokumentation verwiesen ([Quinlan92]). Die anschließenden Abschnitte beschäftigen sich mit der Anwendung von C4.5 auf die konkreten Daten aus Median.

4.1 Entscheidungsbäume – Decision Trees

Entscheidungsbäume stellen das Modell in C4.5 dar, mit dessen Hilfe Aussagen über Zustände eines Systems getroffen werden können. Dem Entscheidungsbaum oder Decision Tree wird ein Problem in Form von Daten (*Testfälle* bestehend aus einzelnen *Attributen*) ähnlich den Trainingsdaten offeriert und anhand des Baumes wird den Daten eine Kategorie oder *Klasse* als Ergebnis zugeordnet.

Eine der wesentlichen Eigenschaften der Klassifizierung ist dabei die Diskretisierung. Die Klassen müssen scharf voneinander abgegrenzt sein, d.h. ein Fall bestehend aus einzelnen Werten gehört entweder zu einer Klasse oder nicht. Die Problematik im Hinblick auf die Aufgabenstellung dieser Diplomarbeit liegt darin, dass das Ziel, die Vorhersage von Angebotswerten, ein kontinuierliches Ergebnis verlangt. Zur Benutzung von Entscheidungsbäumen muss das zu prognostizierende Angebot künstlich diskretisiert werden. Konkret bedeutet dies die Einteilung in Intervalle, denen das Angebot zugeordnet wird. Untersuchungen an Simulationsdaten von Median sollen die Auswirkungen auf Genauigkeit und Geschwindigkeit zeigen, sowie unterschiedliche Möglichkeiten der Klasseneinteilung diskutieren. Die Ergebnisse werden im Abschnitt 4.5 vorgestellt.

Definition 5
Ein nichtleerer Entscheidungsbaum T ist von folgender Struktur:

1. Ein *Blatt* L_j, dem eine Klasse C_i zugeordnet ist und von dem keine weiteren Teilbäume ausgehen, oder

2. Ein *Entscheidungsknoten* (decision node) N, dem ein Test auf ein einzelnes Attribut des Datensatzes zugeordnet ist. Jedem möglichen Ergebnis des Tests ist ein Teilbaum $T_1 ... T_n$ zugeordnet. Ein Knoten ohne Vorgänger heißt *Wurzel*.

Der Entscheidungsbaum wird zur Klassifikation eines Datensatzes benutzt, indem beginnend beim ersten Entscheidungsknoten der Test auf das betroffene Attribut ausgeführt wird und je nach Ausgang des Tests der entsprechende Teilbaum weiterverfolgt wird. Der Baum wird solange durchlaufen, bis ein Blatt erreicht ist und die zugeordnete Klasse als Ergebnis ausgegeben werden kann.

Ein einfaches Beispiel eines Entscheidungsbaumes ist in Abbildung 4.1 gegeben. Dargestellt ist ein einzelner Fall mit den nummerierten Attributen A 1–A 4, den Bezeichnern für die Attribute und den konkreten Ausprägungen für diesen Fall. Der darunter liegende Baum soll anhand dieser Daten eine Wetterprognose in Form einer Klassifizierung in *Regen* oder *Sonne* liefern. An jedem Knoten des Baumes wird der Wert des zugeordneten Attributs getestet und die entsprechende Verzweigung gewählt. Für den speziellen Fall ergibt sich die im Bild markierte Klasse *Regen*. Zu beachten ist an dieser Stelle, dass für die Bewertung durch den Baum nicht alle Attribute benutzt oder im Baum vorhanden sein müssen.

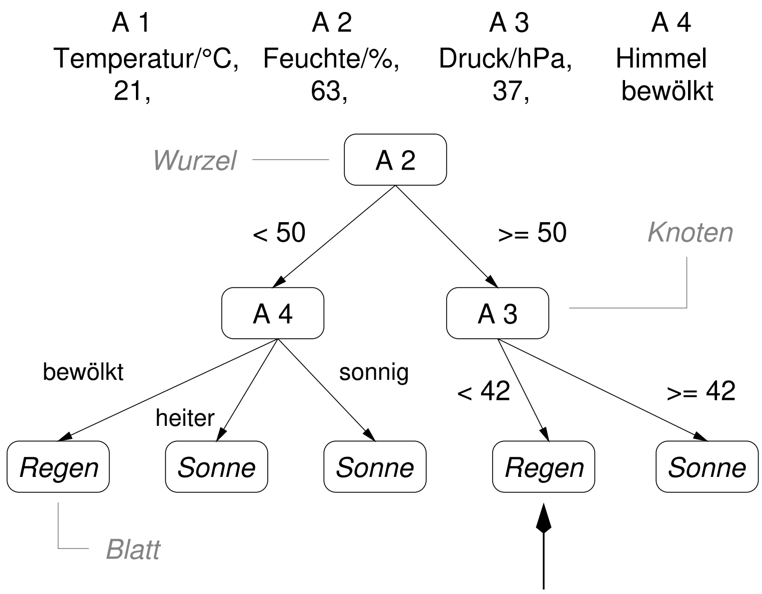


Abbildung 4.1: Einfaches grafisches Beispiel für einen Entscheidungsbaum

Die Vorgehensweise bei Bestimmung der Ergebnisklasse ist in diesem Beispiel folgende: Der Start erfolgt in der Wurzel, getestetes Attribut ist A 2. Da der Wert für die Feuchte mehr als 50 beträgt, wird anschließend der rechte Teilbaum weiterverfolgt. Im nächsten Knoten wird Attribut A 3 überprüft, der Test ergibt eine Verzweigung in den linken Teilbaum. Hier trifft der Algorithmus auf ein Blatt mit der zugeordneten Klasse *Regen* und gibt diese als Ergebnis aus.

4.2 Konstruktion der Bäume

Dieser Abschnitt beschreibt die Vorgehensweise des C4.5 bei der Konstruktion der Entscheidungsbäume.

Für einen sinnvollen Aufbau von Entscheidungsbäumen ist der Test an den einzelnen Knoten von großer Bedeutung für die Qualität der späteren Auswertungen. Prinzipiell müssen die Verzweigungen so gewählt werden, dass bei minimaler Baumgröße ein maximaler Informationsgewinn durch die Aufteilung entsteht. Ziel ist es, Gesetzmäßigkeiten innerhalb der Trainingsdaten zu entdecken und durch einen Baum darzustellen, der eine zuverlässige Prognose ermöglicht. Für eine optimale Generierung von Entscheidungsbäumen ist es theoretisch möglich, alle existierenden Aufteilungen hinsichtlich ihres Aussagegehalts zu überprüfen, praktisch ist dies allerdings nicht durchführbar. Das Problem, den kleinsten zum Trainingssatz konsistenten Entscheidungsbaum zu finden ist NP-vollständig[1] ([Hyafil76]).

C4.5 benutzt deshalb für die Generierung des Entscheidungsbaumes einen sogenannten *nonbacktracking greedy algorithm.* Dies bedeutet, dass die Entscheidung für eine Aufteilung des Trainingssatzes zur Klassifizierung auf Basis des aktuellen Wissensstandes gefällt wird. Nach der Wahl eines Tests zur Aufteilung (Split) der Daten am aktuellen Knoten bleibt dieser Test (bzw. Knoten), unverändert bestehen, selbst wenn sich später ergeben könnte, dass eine andere Aufteilung sinnvoller gewesen wäre. Die Wahl eines Tests zur Aufteilung des Baumes erfolgt in der Regel durch die Maximierung einer Funktion, die die Güte der möglichen Splits misst. Ein Split führt dementsprechend zur Aufteilung der Trainingsdaten in kleinere Teilmengen.

Im C4.5 werden für die Bewertung der Tests das *gain* und das *gain ratio* Kriterium benutzt, die in den folgenden Abschnitten erläutert werden. Es existiere ein Test, der auf den Trainingsdatensatz T angewandt wird und n Verzweigungen ergibt, die jeweils zu den Teilbäumen $T_1, T_2, ...T_n$ führen. Ein Test mit n Verzweigungen ergibt sich zum Beispiel dann, wenn auf ein Attribut in den Trainingsdaten getestet wird, das n verschiedene Werte annehmen kann (Die Größe der zugehörigen Domäne ist n). Bei kontinuierlichen Attributen, die eine unendliche Anzahl von möglichen Unterteilungen besitzen, wird ein binärer Test gewählt, d.h. es gilt $n = 2$. Die genaue Vorgehensweise für kontinuierliche Attribute wird später in diesem Kapitel näher erläutert. Zunächst besteht die Notwendigkeit, aus den möglichen Unterteilungen der Menge T die günstigste (oder zumindest eine dieser nahe kommende) auszuwählen. Soll der Raum der möglichen Unterteilungen nicht oder nicht vollständig untersucht werden, ist die einzige zur Verfügung stehende Information die Verteilung der Klassen in T. Quinlan benutzt zur Beschreibung seines Verfahrens folgende Notation: Sei

[1] Nicht-deterministisch polynomiell: Das Problem ist nicht mittels eines deterministischen Algorithmus in polynomieller Laufzeit lösbar.

T eine beliebige Menge von Fällen, dann bedeutet $freq(C_i, T)$ die Anzahl der Fälle in T, die zur Klasse C_i gehören. $|T|$ entspricht der Anzahl aller Fälle in T.

In der Informationstheorie gilt der Satz, dass die Information, die durch ein Ereignis[2] gewonnen wird, sich als negativer Logarithmus zur Basis 2 in Abhängigkeit der Wahrscheinlichkeit des Auftretens dieses Ereignisses ergibt. Für beispielsweise acht gleichwahrscheinliche Ereignisse mit paarweise voneinander unabhängigen Attributen ergibt sich ein Informationsgehalt von

$$-\log_2(1/8) = 3.$$

Für den konkreten Fall der Trainingsdaten ergibt sich für die Wahrscheinlichkeit, dass ein zufällig aus einer Menge T gewählter Fall zu einer Klasse C_j gehört, die Gleichung

$$\frac{freq(C_j, T)}{|T|}$$

mit dem Informationsgehalt

$$-\log_2(\frac{freq(C_j, T)}{|T|}).$$

Für den erwarteten Informationsgehalt aus der gesamten Menge T in Beziehung zu allen Klassen ergibt sich die Summe über alle Klassen in Beziehung zur Häufigkeit ihres Auftauchens in T:

$$info(T) = -\sum_{j=1}^{k} \frac{freq(C_j, T)}{|T|} \cdot \log_2(\frac{freq(C_j, T)}{|T|})$$

Auf den Trainingsdatensatz angewandt ist $info(T)$, die *Entropie*, das Maß für die durchschnittlich benötigte Informationsmenge, die notwendig ist, um die Klasse eines Falles in T bestimmen zu können.

Ein ähnliches Prinzip lässt sich auch für die aufgeteilten Untermengen T_i mit $i = 1, ..., n$ angeben. Wird T durch einen Test X mit n möglichen Ergebnissen unterteilt, ergibt sich die durchschnittlich benötigte Menge an Information als gewichtete Summe über die benötigte Information der Untermengen:

$$info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot info(T_i).$$

Letztendlich ergibt sich das gain-Kriterium als

$$gain(X) = info(T) - info_X(T).$$

Damit wird ein Test X gewählt, der den Informationsgewinn maximiert.

Das gain-Kriterium allein ist in den meisten Situationen bereits recht gut für die Auswahl eines Splits des Entscheidungsbaumes geeignet. In Ausnahmefällen, z.B. bei Tests, die eine große Anzahl an möglichen Ergebnissen aufweisen, ergibt sich eine unausgewogene Unterteilung der Trainingsdaten. Als

[2] In diesem Zusammenhang das Auftreten der Trainings- und Testfälle

Beispiel soll eine Patientenkartei dienen. Enthält eines der Attribute die Patientennamen als Wert, enthält nahezu jeder der Trainingssätze einen individuellen Wert. Erfolgt die Unterteilung anhand dieses Attributs, ergibt sich eine sehr große Menge an Unterbäumen, die alle lediglich einen Fall enthalten. Da alle dieser einelementigen Teilbäume nur Fälle beinhalten, die zu einer einzelnen Klasse gehören, gilt $info_X(T) = 0$. Damit ergibt sich hingegen ein maximaler Wert für den Informationsgewinn gain. Unter dem Aspekt der Vorhersage, bzw. Klassifizierung ist dagegen eine Unterteilung nach diesem Attribut zwecklos.

Um diesen Effekt abzumildern benutzt C4.5 das auf dem gain-Kriterium basierende gain ratio-Kriterium. Dies stellt eine Normalisierung hinsichtlich des Tests mit vielen möglichen Ergebnissen dar. Quinlan definiert eine Funktion *split info*, die den Informationsgehalt nicht in Bezug zu den Klassen, sondern relativ zur Größe des Splits misst:

$$splitinfo(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot \log_2(\frac{|T_i|}{|T|}).$$

Diese Funktion repräsentiert den potenziellen Informationsgewinn durch Aufteilung von T in n Teilmengen. Damit ergibt sich

$$gainratio(X) = gain(X)/splitinfo(X)$$

als Maß für den Informationsgewinn durch „sinnvolle“ Splits. Wenn die Aufteilung von T nahezu trivial ist, ergibt sich ein kleiner Wert für den gain ratio. Ziel ist wieder die Maximierung dieser Funktion, um einen guten Split zu finden.

Bezogen auf das Beispiel der Patientenkartei ergibt sich für den gain ratio folgendes: Existieren wie zuvor k Klassen, ist der Zähler (gain) ungefähr $\log_2(k)$. Für den Nenner gilt $\log_2(n)$. Unter der Voraussetzung, dass eine deutlich höhere Anzahl an Trainingsdaten als Klassen zur Verfügung steht ($n \gg k$), ergibt sich ein kleiner Wert für gain ratio und der Patientenname wird als Split-Attribut abgelehnt.

In den vorhergehenden Abschnitten wurde auf die Auswertungsfunktion für die Güte eines Splits hinsichtlich eines Tests X eingegangen, ohne diesen Test näher zu erläutern. Dies soll jetzt nachgeholt und die Tests zum Aufteilen des Entscheidungsbaumes beschrieben werden. C4.5 erlaubt die Verwendung diskreter und kontinuierlicher Attribute. Für jeden Test wird jeweils nur ein Attribut ausgewertet. Bei Attributen mit einem diskreten Wertebereich wird für jeden möglichen Wert ein eigener Zweig im Entscheidungsbaum angelegt. Da die Simulationsausgabe von Median lediglich kontinuierliche Attribute liefert, ist der Split bei dieser Attributart ebenfalls wichtig und wird in den folgenden Absätzen erläutert.

Für jedes kontinuierliche Attribut A wird grundsätzlich ein binärer Test der Art $A \leq Z$ und $A > Z$ durchgeführt. Z ist dabei ein Schwellwert, der aus den Trainingsdaten berechnet wird. Die Durchführung dieses Tests ist nicht wesentlich aufwändiger als für diskrete Attribute.

Zunächst werden alle Fälle innerhalb des Trainingsdatensatzes T nach dem Wert des zu testenden Attributs A sortiert. Da lediglich eine endliche Anzahl an

Trainingsdaten vorliegt, können die Werte eindeutig als $\{v_1, v_2, ..., v_m\}$ geordnet werden. Jeder Schwellwert zwischen einem Wertepaar v_i und v_{i+1} teilt die Fälle in die Untermengen $\{v_1, v_2, ..., v_i\}$ und $\{v_{i+1}, v_{i+2}, ..., v_m\}$. Zur Entscheidung für einen möglichen Split werden alle $m-1$ Unterteilungen untersucht. Da die Werte zuvor sortiert wurden, kann jede mögliche Unterteilung leicht in Laufzeit $\mathrm{O}(m-1)$ mit dem gain ratio-Kriterium untersucht werden. Dazu muss nur von Iteration zu Iteration das Element v_{i+1} aus der oberen Menge in die untere verschoben und das Kriterium auf die neuen Teilmengen angewandt werden.

Die Wahl des Schwellwertes Z zur Aufteilung der Menge T kann nach unterschiedlichen Gesichtspunkten erfolgen. In vielen Fällen wird einfach der Mittelpunkt des zu untersuchenden Intervalls gewählt:

$$\frac{v_i + v_{i+1}}{2}.$$

C4.5 wählt dagegen den größten Wert aus allen im Trainingssatz gegebenen Werten des Attributs A, dass nicht über diesem Schwellwert liegt (v_i). Damit ist sichergestellt, dass nur Werte als Schwellwert auftreten, die auch in den Trainingsdaten vorhanden sind.

4.3 Reduktion der Entscheidungsbäume

Der Vorgang der Unterteilung des aktuellen Entscheidungsbaumes wird solange fortgesetzt, bis alle einem Ast zugeordneten Fälle zu einer einzigen Klasse gehören oder eine weitere Unterteilung anhand eines Tests keine Verbesserung mehr bringt. An dieser Stelle ergibt sich die Gefahr, einen Baum zu erzeugen, der sich zu genau an die Trainingsdaten annähert. Die dem Lernverfahren offerierten Daten stellen prinzipiell immer einen Ausschnitt aus allen möglichen Daten dar und können somit nur einen kleinen Teil widerspiegeln. Besonders wichtig für einen guten Entscheidungsbaum ist die Qualität und die Repräsentativität der Daten. Zu den Trainingsdaten lässt sich in vielen Fällen ein sehr genau angepasster Baum erstellen. Sind allerdings die in den Trainingsdaten vorhandenen Attribute nicht aussagekräftig oder enthalten sie nur selten auftretende Fälle, ist die Gefahr einer Fehlklassifikation unbekannter Daten relativ hoch. Dieser Effekt wird als *Overfitting* bezeichnet.

Vergleichbar ist dies mit Testpersonen, die lernen sollen, wie ein Baum aussieht. Werden zu Ausbildungszwecken Fotos von Bäumen vorgelegt und besitzt die Testperson ein eidetisches Gedächtnis, können später nur Objekte identifiziert werden, die den Vorlagen bis auf das letzte Blatt hin gleichen. Werden andererseits nur Attribute wie „brauner Stamm“ und „grünes Blattwerk“ verwendet, kann ein Baum im Winter nicht erkannt werden.

C4.5 versucht diese Effekte durch einen *Pruning* genannten Vorgang abzumildern. Pruning bedeutet die Erzeugung eines einfacheren, weniger komplexen Baumes, der für unbekannte Fälle eine deutlich geringere Zahl an Fehlklassifikationen erzeugen soll. Die Vereinfachung besteht darin, Teile des Baumes, bei denen der normale Erstellungsvorgang anhand der Trainingsdaten eine weitere Unterteilung vorsieht, durch Blätter zu ersetzen.

Zwei Möglichkeiten bieten sich an: das Abbrechen der Baumentwicklung zu einem früheren Zeitpunkt als bei dem normalen Algorithmus oder das Rückentwickeln des Baumes bis zu einer bestimmten Grenze. Der erste Ansatz bietet den Vorteil, keine unnütze Zeit in eine Baumentwicklung stecken zu müssen, die später wieder verworfen wird. Normalerweise werden dazu ein oder mehrere Verfahren ausgewählt (gain ratio, statistische Signifikanz, Fehlerverringerung...), die entsprechend eines Schwellwertes zum Abbruch der Entwicklung führen. Problematisch ist hierbei die Wahl des Schwellwertes. Die meisten Verfahren zur Bestimmung der Güte eines Tests stützen sich auf Schätzungen über die Aussagekraft des Splits. Die Vorteile mancher Splits treten allerdings gelegentlich erst nach weiteren Unterteilungen auf. Bei einem zu hohen Schwellwert wird die Entwicklung zu früh abgebrochen. Ein zu niedriger Schwellwert dagegen kann zu einer zu geringen Vereinfachung führen.

Aus diesem Grund benutzt C4.5 die zweite Methode, die Rückentwicklung des Entscheidungsbaumes. Laut Quinlan führt dieses Vorgehen zu einer erhöhten Vorhersagequalität im Gegensatz zu der durch die Rückentwicklung geringfügig langsameren Erstellung der Pruned Decision Trees.

Da das Pruning ein Zusammenfassen benachbarter Zweige enthält, wird es in den meisten Fällen zu einer erhöhten Fehlklassifikation bereits auf den Trainingsdaten führen. Die zu einem Blatt vereinigten Zweige repräsentieren dann nicht mehr nur eine einzelne Klasse, sondern eine Menge von Klassen mit einer entsprechenden Verteilung, die die Wahrscheinlichkeit angibt, dass ein Trainingsfall zu dieser Klasse gehört.

Im folgenden Abschnitt wird die Strategie beschrieben, welche angewandt wird, um festzulegen, wann der Vereinfachungsvorgang gestoppt wird. Grundlegend hierfür ist die Berechnung, bzw. Abschätzung, der Fehlerraten. Die Vereinfachung des Baumes wird sich nur dann lohnen, wenn sich dadurch eine Verringerung des erwarteten Fehlers auf unbekannten Daten erzielen lässt.

Dazu wird vom Boden des Baumes aus begonnen, jeden Teilbaum, der mehr als ein Blatt enthält, zu untersuchen und für den Fall, dass sich die Fehlerrate durch Ersetzen des Teilbaumes durch ein Blatt oder einen Zweig verbessern lässt, der Baum dementsprechend vereinfacht. Die Fehlerrate anhand der bei der Erstellung des Baumes benutzten Trainingsdaten bietet für die Abschätzung nur ein unzureichendes Maß, da einerseits eine Vereinfachung des Baumes in den meisten Fällen zu einer höheren Rate an Fehlklassifikationen der Trainingsdaten führen muss und zusätzlich nur eine geringe Aussagesicherheit hinsichtlich unbekannter Daten besteht.

Eine wesentlich bessere Abschätzung des zu erwartenden Fehlers ist direkt mit dem Test des Baumes anhand unbekannter Daten möglich. Dazu muss allerdings von den Daten, die für das Training vorgesehen waren, ein Teil für den Test bewahrt werden, was im Fall nur weniger Daten wiederum zu einem ungenaueren Baum führen kann. Aus diesem Grund versucht C4.5 eine Schätzung des Fehlers nur anhand der Trainingsdaten. Ähnlich wird z.B. bei der Kreuzvalidierung vorgegangen, wozu alle Datensätze in mehrere unterschiedliche Pakete aus Trainings- und Testdaten aufgeteilt und getrennt Lern-/Testvorgänge ausgeführt werden.

Sind einem Blatt N Datensätze zugeordnet, von denen E falsch klassifiziert sind, so ergibt sich für dieses Blatt eine Fehlerrate von E/N. Werden die N Datensätze als Auszug aus den möglichen Datensätzen verstanden, stellt sich die Frage nach der Aussagekraft für den zu erwartenden Fehler. Quinlan wählt dazu eine pessimistische Schätzung auf Basis des oberen Konfidenzintervalls der Binomialverteilung ($U_{CF}(E, N)$). Auch wenn dieser Ansatz die nicht bekannte wirkliche Verteilung nur selten widerspiegeln wird, bietet diese Abschätzung für die meisten Fälle ein realistisches Ergebnis.

Für die Anzahl an Fehlern, bei einer gleich großen Anzahl an unbekannten Datensätzen wie für das Training benutzt, ergibt sich die geschätzte Fehlerrate multipliziert mit der Anzahl der durch das Blatt abgedeckten Fälle $N \cdot U_{CF}(E, N)$. Die geschätzte Fehlerrate für den gesamten Baum

$$\frac{1}{|T|} \sum_i N_i \cdot U_{CF}(E, N_i)$$

ergibt sich als Summe der Fehler über alle Blätter i dividiert durch die Anzahl an Trainingssätzen $|T|$.

4.4 Repräsentation der Daten

Neben dem eigentlichen Algorithmus zum Lernen und Prognostizieren von Werten kommt auch der Vorverarbeitung eine entscheidende Rolle zu. Je nach Art des benutzten Verfahrens eignen sich unterschiedliche Repräsentationen der Daten mehr oder weniger gut zur Analyse. In der Untersuchung von C4.5 nehmen demzufolge die einzelnen Datenaufbereitungen einen wichtigen Punkt ein.

Median stellt in seiner Simulation unterschiedliche Dienste mit ihren spezifischen Charakteristika wie Bit- und Burstrate zur Verfügung.

Abbildung 4.2 zeigt das Beispiel eines möglichen Verlaufs für den Dienst Telefonie. Dargestellt ist die dem Netzwerk für eine Verbindungsstrecke angebotene Last im Verhältnis zur Tageszeit. Vorstellbar ist die Kurve z.B. für das Telefonverhalten in überwiegend wirtschaftlich genutzten Gebieten. Die Anzahl der Rufe pro Zeiteinheit steigt morgens stark an, erleidet zur Mittagszeit einen kurzzeitigen Einbruch und fällt dann gegen Abend wieder ab. Unterteilt sind die Werte nach den einzelnen Stunden, die Balken geben die durchschnittliche Last für den jeweiligen Zeitraum wieder. Die Vertrauensintervalle an jedem Balken zeigen den Raum für die möglichen Schwankungen (+/-10%) um diesen Wert, d.h. auf die Daten wird ein zufällig bestimmtes, gleichverteiltes Rauschen (*Noise*) addiert. Die so berechneten Werte werden der Simulation des Netzes für jedes betrachtete Zeitintervall für die Berechnung als Basiskurve zur Verfügung gestellt. Die Werte zu den jeweiligen Zeiten auf der Kurve stellen den Mittelwert der Angebote für das Intervall dar. Die konkreten Werte im Verlauf der Simulation schwanken negativ-exponentiell verteilt um diesen Mittelwert. Um eine gute Konfiguration hinsichtlich der Routen der einzelnen Verbindungen zu ermöglichen, ist eine Prognose über das zukünftige Maximum der Kurve nötig.

Die Daten werden von der Simulation in einfachen Textdateien gespeichert, eine für jede mögliche Verbindung. Die simulierten Dienste werden mit ihren

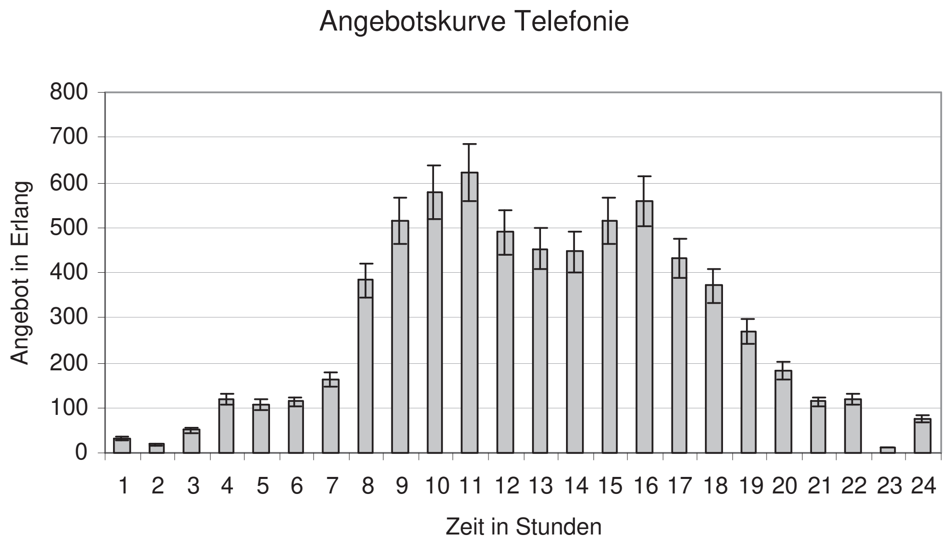


Abbildung 4.2: Beispiel einer Angebotskurve Telefonie über 24 Stunden

Daten in der Datei der benutzten Verbindung gehalten. Wichtig für die Vorhersage sind die Daten über die *logischen* Lasten auf dem Netzwerk, da hier der Verkehr zwischen allen Knoten gemessen wird. Das Netz ist auf dieser Ebene damit vollvermascht, denn jeder Knoten muss die Möglichkeit besitzen, alle anderen Knoten auf direktem Weg erreichen zu können. Die Abbildung des logischen Netzes auf das darunter liegende physikalische Netz besitzt für die Prognose der Lasten keine Bedeutung.

Die Textdateien mit den Ausgaben der Simulation (Abbildung 4.3) besitzen für jede Zeile jeweils folgendes Format:

1. Nummer des Durchlaufs: In der Regel wird die Simulation für mehrere Durchläufe, synonym zu Tagen, über die Angebotskurve ausgeführt. Dies ermöglicht eine genauere Prognose, da sich so z.B. statistisch der Mittelwert von 10 Durchläufen (bzw. Tagen) genauer bestimmen lässt. In Abbildung 4.3 ist für die Zeitindizes 600–3600 zu erkennen, wie die Angebotswerte um 30 Erlang schwanken, während ab Zeitindex 4200 (2. Stunde) ein Angebotswechsel stattfindet.

   Da sich das Netzwerk für den ersten Durchlauf noch nicht in einem eingeschwungenen Zustand befindet, kann dieser Durchlauf nicht zum Training des Systems verwendet werden.

2. Zeitindex in Sekunden: Dieser Wert gibt die bis zum aktuellen Zeitpunkt in diesem Durchlauf verstrichene Simulationszeit an. Sämtliche durchgeführten Berechnungen beziehen sich auf den Zeitraum vom letzten bis zum angegebenen Zeitpunkt. Der Abstand zwischen zwei Zeitindizes gibt den Simulationstakt an (in diesem Beispiel: 600 Sekunden).

```
2/600.0/33.600000/0.000000/0.000000/33.600000/0.000000/...
2/1200.0/28.000000/0.000000/0.000000/28.000000/0.000000/...
2/1800.0/28.600000/0.000000/0.000000/28.600000/0.000000/...
2/2400.0/34.000000/0.000000/0.000000/34.000000/0.000000/...
2/3000.0/36.000000/0.000000/0.000000/36.000000/0.000000/...
2/3600.0/26.200000/0.000000/0.000000/26.200000/0.000000/...
2/4200.0/17.400000/0.000000/0.000000/17.400000/0.000000/...

...0.000000/0.000000/-1.000000/-1.000000/...
...0.000000/0.000000/-1.000000/-1.000000/...
...0.000000/0.000000/-1.000000/-1.000000/...
...0.000000/0.000000/-1.000000/-1.000000/...
...0.000000/0.000000/-1.000000/-1.000000/...
...0.000000/0.000000/-1.000000/-1.000000/...
...0.000000/0.000000/-1.000000/-1.000000/...
```

Abbildung 4.3: Ausgabe der Simulation für einen Link

3. Die nächsten drei Felder enthalten das Verkehrsangebot in Erlang für den ersten Dienst über drei mögliche Routen. Diesen drei Routen entsprechen drei unterschiedliche konkrete Wege, auf denen die Verbindung zwischen zwei Knoten stattfinden kann. In Abbildung 4.3 wurde zur Simulation nur eine Route benutzt, daher sind die beiden anderen Einträge Null. Im Fall einer Berechnung mit alternativen Wegen verteilt sich das gesamte Angebot auf die drei Einträge.

4. Dementsprechend enthalten die darauf folgenden Felder den Verkehr des Dienstes 1 auf den drei Routen. Der Verkehr entspricht im Gegensatz zum Angebot der realen Last auf der Verbindung. Kann eine Verbindung die volle Last des angebotenen Verkehrs nicht verkraften, sinkt der reale Verkehrswert. Im Beispiel stimmen beide Werte überein, da die Belastung recht gering ist (in der Simulationszeit ist es kurz nach Mitternacht).

5. Die letzten drei Felder für den Dienst 1 geben die Blockierungswahrscheinlichkeit an, bzw. den Prozentsatz an Verbindungen, die auf der jeweiligen Route nicht geschaltet werden konnten. Bei nicht benutzten Routen wird hier eine -1 eingetragen.

6. Für jeden Dienst (max. 10) auf der Verbindung wiederholen sich die neun Felder für Angebot, Verkehr und Blockierung.

Die verfügbaren Informationen müssen in einem Vorverarbeitungsschritt in das Eingabeformat von C4.5 transformiert werden. Als minimale Voraussetzung benötigt C4.5 zwei Dateien: die eigentlichen Trainingsdaten und eine Definitionsdatei.

Die Trainingsdaten bestehen aus einer Sammlung von Attributen, die für eine Klassifizierung benutzt werden können, und der Klasse, die für den konkreten Fall angegeben ist. Die einzelnen Attribute eines Falles werden durch

Kommata getrennt, die gesamte Zeile mit der zugehörigen Klasse abgeschlossen. Das Beispiel in Abbildung 4.4 beginnt mit der Nummer des Durchlaufs als Attribut. Es folgen der Zeitindex und die letzten 11 Angebotswerte von diesem Zeitindex ausgehend. Am Ende der Zeile steht die Ergebnisklasse, die das Intervall angibt, in dem das Maximum der nächsten drei Stunden liegt (hier: zwischen 100 und 150 Erlang).

Damit befindet sich in jeder Zeile ein von den anderen Zeilen unabhängiger Trainingssatz, der im Wesentlichen die vergangenen Angebote enthält (*Historie*).

```
2,6600,33.6,28,28.6,34,36,26.2,17.4,18,16.6,17.6,21.2,[100-150[
2,7200,28,28.6,34,36,26.2,17.4,18,16.6,17.6,21.2,18,[100-150[
2,7800,28.6,34,36,26.2,17.4,18,16.6,17.6,21.2,18,44.8,[100-150[
2,8400,34,36,26.2,17.4,18,16.6,17.6,21.2,18,44.8,43.6,[100-150[
2,9000,36,26.2,17.4,18,16.6,17.6,21.2,18,44.8,43.6,51,[100-150[
2,9600,26.2,17.4,18,16.6,17.6,21.2,18,44.8,43.6,51,43,[100-150[
2,10200,17.4,18,16.6,17.6,21.2,18,44.8,43.6,51,43,55.2,[100-150[
```

Abbildung 4.4: Datendatei für C4.5

```
[0-50[,[50-100[,[100-150[,[150-200[,[200-250[...
Nummer:            ignore.
Zeit:              ignore.
Angebot0:          continuous.
Angebot1:          continuous.
Angebot2:          continuous.
```

Abbildung 4.5: Definitionsdatei für C4.5

Die Definitionsdatei enthält Informationen über die verwendeten Klassen und die Art der Attribute (*discrete* oder *continuous*). Zusätzlich lassen sich einzelne Attribute aus der betrachteten Menge von Attributen ausklammern (*ignore*), wenn diese beispielsweise keinen sinnvollen Beitrag zur Klassifizierung bringen, aber aus Kontrollgründen vorhanden sind. Abbildung 4.4 enthält den Ausschnitt einer zum vorangegangenen Beispiel gehörenden Datendatei, Abbildung 4.5 die entsprechende Definitionsdatei. Zu Beginn der Definitionsdatei werden alle möglichen Klassen vorgegeben. Die Wahl der Bezeichner ist frei, für die Zielsetzung der Klassifikation des maximalen Angebots der nächsten Stunden sind hier die Intervalle als jeweilige Klasse angegeben. Die Schrittweite der Intervalle stellt damit einen weiteren Freiheitsgrad bei der Suche nach einer genauen Vorhersage dar und kann z.B. durch Experimente bestimmt werden.

Die Zuordnung eines Falles zu einer Klasse bedeutet die Einschätzung des maximalen Angebots innerhalb des angegebenen Intervalls. Wie bereits in Abschnitt 4.1 angedeutet, bedeutet die Zuordnung von kontinuierlichen Werten zu einer Klasse eine künstliche Ungenauigkeit. Der maximale Fehler trotz richtiger Zuordnung zu einer Klasse entspricht damit der Größe eines solchen Intervalls, der durchschnittliche Fehler bei z.B. angenommener Gleichverteilung der Daten innerhalb des Intervalls der halben Größe. Dazu kommt noch der Fehler, der durch Fehlklassifikationen entsteht und wesentlich durch das Verfahren bestimmt wird.

Abbildung 4.6 demonstriert die beiden Fälle nochmals genauer: Der reale Angebotswert (grauer Kreis), der bestimmt werden, soll befindet sich bei 110 Erlang. Zwei verschiedene Klassifikationen 1 und 2 (schraffiert) sind gegeben. Da die Klassen Intervallen entsprechen, wird für Fall 1 der Bereich 0–50 Erlang abgedeckt, für Fall 2 entsprechend 100–150 Erlang. Um die diskrete Klasse mit dem realen Wert vergleichen zu können, muss die Klasse in einen kontinuierlichen Wert gewandelt werden, in der Abbildung ist das jeweils die Intervallobergrenze 50, bzw. 150 Erlang. Fall 1 klassifiziert falsch, so dass sich der Fehler als Abstand zwischen Klasse und realem Wert ergibt, d.h. 60 Erlang. In Fall 2 wird die richtige Klasse bestimmt, durch die Schrittweite von 50 Erlang für jede Klasse ergibt sich allerdings auch hier ein Fehler und zwar in der Größe von 40 Erlang.

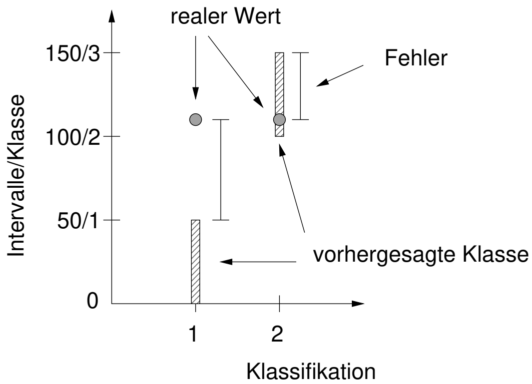


Abbildung 4.6: Fehlerarten bei Klassifikation

Die nachfolgende Aufzählung beschreibt nun verschiedene mögliche Ansätze für Umformungen, mit denen die Ausgaben der Simulation zum Training umgesetzt und anschließend die Fehler bestimmt werden können:

1. Berücksichtigung des Zeitindex: Die Zeit kann als eigenes Attribut in die zu lernenden Daten mit aufgenommen werden. In Abbildung 4.5 z.B. ist der Zeitindex durch die Option „ignore“ explizit ausgeklammert.

2. Bereich in der Vergangenheit: die Anzahl der letzten Angebotswerte, die den bisherigen Verlauf der Kurve wiedergeben (*Historie*).

3. Skaleneinteilung: die Behandlung der Historie. So können die Attribute entlang der Zeitachse in linearen oder auch logarithmischen Schritten ausgewählt werden, um den Einfluss von Werten abzumildern, die weiter zurück in der Vergangenheit liegen. Ein Beispiel für logarithmische Auswahl wäre z.B. das aktuelle Angebot als Attribut auszuwählen, dann das zwei Zeittakte weiter zurückliegende, dann das vier Takte zurückliegende usw.

4. Prognosehorizont: der Bereich, für den das maximale Angebot gesucht werden soll. Auf den ersten Blick wirkt sich dieser Wert nicht zwingend auf das Verfahren aus, da die Prognose nicht zu den Lernattributen gehört. Da sich hier aber in einigen Fällen die Klasse ändert, verändern sich auch die Pfade durch den Baum zu diesen Klassen. In Abbildung 2.4 sind unterschiedlich lange Prognosehorizonte dargestellt. Je nach Wahl des längsten oder kürzesten Horizonts verändert sich das betrachtete Maximum.

5. Schrittweite der Klasseneinteilung: die Größe des Wertebereichs, der einer Klasse zugeordnet ist (Diskretisierung der kontinuierlichen Prognose). In den bisherigen Beispielen betrug die Schrittweite immer 50 Erlang.

6. Variable Schrittweiten: In bestimmten Bereichen kann es sinnvoll sein, kleinere oder größere Intervalle für die Klassen zu bilden.

7. Konstruktion neuer Attribute:

   (a) Behandlung von Routen: Bei Verbindungen, die über mehr als einen Weg geschaltet werden können, ergibt sich die Frage, ob diese Routen getrennt oder gemeinsam behandelt werden können.

   (b) Berücksichtigung von Besonderheiten in den Ausgangskurven: steile Anstiege, Peaks, Leerlaufzeiten...

8. Kontinuierliche oder diskrete Attribute: Obwohl die Ausgaben der Simulation grundsätzlich kontinuierliche Werte liefern, könnte eine künstliche Diskretisierung der Attribute Vorteile bringen. Beispielsweise könnte jeder Angebotswert auf Vielfache der Schrittweite gerundet werden.

9. Berücksichtigung der Daten über die Grenzen einzelner Verbindungen hinweg: Anstatt des Verlaufs der Last über einen einzelnen Link können auch die Werte aller Links eines Zeitpunkts zum Lernen genutzt werden. So lassen sich z.B. Erkenntnisse über Zeiten gewinnen, an denen allgemein viel Verkehr im Netz vorhanden ist. Abbildung 4.7 gibt einen grafischen Eindruck der Verhältnisse. Die Prognose ist für beide Arten der Bewertung von Bedeutung.

10. Kreuzvalidierung: Zur Bewertung des Entscheidungsbaumes wird ein aus den vorhandenen Daten extrahierter Testdatensatz benutzt. In der Regel wird dieser aus mindestens einem vollständigen Tagesverlauf bestehen. Um allerdings die Repräsentativität der Testdaten zu gewährleisten, ist es sinnvoll, die Testdaten auch verteilt über alle Tage auszuwählen.

Der C4.5-Algorithmus bietet zusätzlich Möglichkeiten zur Behandlung von Datensätzen mit fehlenden Werten. Für die Bewertung von Telekommunikationsnetzwerken spielt dies allerdings nur eine untergeordnete Rolle. Zwar führt der Ausfall von Vermittlungen und damit auch von Verkehrslastmessungen für Betreiber und Anwender zu unangenehmen Komplikationen, wenn allerdings der Abstand zwischen zwei Lernzyklen des Prognosewerkzeugs groß ist

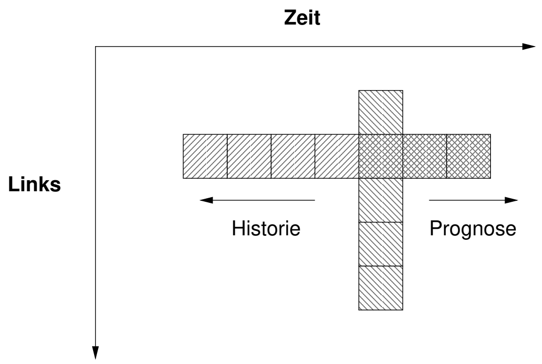


Abbildung 4.7: Bewertungen der Angebote entlang der Zeit und der Links

gegenüber dem Ausfallzeitraum, kann auf die verlorenen Trainingsdaten verzichtet werden. Da in der Regel die Ausfallzeiten so kurz wie möglich gehalten werden (die Deutsche Telekom hat beispielsweise eine Vorgabe von maximal 10 min Ausfallzeit eines Vermittlungsknotens pro Jahr), hält sich die Menge an verlorenen Daten in Grenzen. Ausfälle physikalischer Leitungen (der vielzitierte „Bagger“) wirken sich nicht auf das Angebot, d.h. die Anzahl der Verbindungwünsche, aus und verändern daher nicht den Lern-/Prognosezyklus. Das Umkonfigurieren des logischen Netzes und damit die Kompensation des Ausfalls gehört zu den Aufgaben des Managementteils von Median.

Neben der Umwandlung der Daten zur Eingabe in das C4.5-Programm gibt es noch andere zu berücksichtigende Aspekte. Um die Güte des generierten Entscheidungsbaumes zu messen, ist es erforderlich, diesen auf einem Datensatz zu testen, in dem für jeden einzelnen Fall die zugehörige Ergebnisklasse bekannt ist. Dieser Datensatz muss aus den vorhandenen Daten und damit aus den Daten, die für das Training zur Verfügung stehen, entnommen werden. Hier ergibt sich die Frage nach den Größenverhältnissen zwischen beiden Mengen. Je kleiner der verbleibende Trainingssatz, umso geringer wird auch die Aussagekraft der Daten und damit die Wahrscheinlichkeit, dass C4.5 entscheidende Merkmale aus den Daten extrahieren kann. Andererseits bedeutet eine größere Testdatenmenge eine höhere Aussagegenauigkeit über die Güte des Baumes. In der Regel wird aber die Menge an Fällen im Trainingssatz deutlich höher sein als im Testdatensatz. Eine weitere Faustregel besagt, dass die Anzahl an Trainingsfällen (z.B. > 1000) groß gegenüber der Anzahl an Klassen (z.B. < 20) sein muss. Nur so kann eine zu jeder Klasse hinreichende Menge an Fällen gewährleistet werden.

Ein weiteres Merkmal von C4.5 ist, dass der Algorithmus aus einer evtl. vorhandenen Sortierung der Ergebnisklassen keinen Nutzen ziehen kann. Dieser Effekt macht sich insbesondere bei Fehlklassifizierungen bemerkbar, da dort Ausreißer in Regionen liegen können, die weit von der richtigen Klasse entfernt sind. Bei Berücksichtigung der Reihenfolge ließen sich z.B. Erkenntnisse aus den zuvor prognostizierten Klassen gewinnen, je nachdem wie starken Schwankungen die Angebotskurve unterworfen ist. Bei relativ sanften Verläufen sind starke Unterschiede in der Quantität zweier aufeinander folgend prognostizierten Klassen eher unwahrscheinlich. Die Probleme bei der Klassifizierung treten u.a. dann auf, wenn sich einzelne Kurvenverläufe stark ähneln, aber eine völlig andere Voraussage für das maximale Angebot der folgenden Stunden aufweisen. In solchen Bereichen kann selbst geringes Rauschen der Eingabedaten eine völlig unterschiedliche Klassifizierung erzeugen. Abhilfe ließe sich hierbei dadurch schaffen, dass mehrere Bäume auf den selben Daten, aber mit unterschiedlicher Auflösung der Klassifizierung betrachtet werden; dann überlappen sich die vorhergesagten Klassen in einzelnen Intervallen. Dabei sinkt die Wahrscheinlichkeit, dass ein Ausreißer sich auch bei unterschiedlichen Intervallgrößen für jeden Baum im selben Bereich befindet.

4.5 Ergebnisse

Nachdem der vorherige Abschnitt auf prinzipielle Möglichkeiten der Datenrepräsentation eingegangen ist, zeigt dieser Abschnitt die Ergebnisse der konkreten Anwendung der Transformationsverfahren auf die Simulationsdaten aus Median. Beschrieben werden die grundlegenden Umwandlungen, die dabei angewandten Parameter und die erreichte Genauigkeit, sowohl geschätzt als auch anhand von Testdaten. Zum Einsatz kam eine angepasste Version von C4.5 für Windows NT 4.0 und ein Transformationsprogramm, das aus den Simulationsdaten von Median die Eingabeformate generiert hat. Die einzelnen Berechnungen wurden auf einem PentiumPro 233 MHz mit 64 MByte Hauptspeicher durchgeführt.

Nach der Berechnung des Entscheidungsbaumes gibt C4.5 diesen im Textformat zur Anschauung wieder. Damit erhält der Anwender die Möglichkeit, sich einen Überblick über die Verzweigungen innerhalb des Baumes oder über nicht benutzte Attribute zu machen. Abbildung 4.8 zeigt einen Ausschnitt eines einfachen erzeugten Baumes.

```
Angebot10 <= 131 :
|   Angebot10 <= 83 :
|   |   Angebot0 > 83 : [50-100[ (94.0)
|   |   Angebot0 <= 83 :
|   |   |   Angebot10 <= 58.2 : [100-150[ (170.0)
|   |   |   Angebot10 > 58.2 : [50-100[ (18.0)
|   Angebot10 > 83 :
|   |   Angebot4 <= 77 : [150-200[ (60.0)
|   |   Angebot4 > 77 :
|   |   |   Angebot0 > 131 : [100-150[ (100.0)
|   |   |   Angebot0 <= 131 :
|   |   |   |   Angebot4 <= 110.8 :
```

Abbildung 4.8: Entscheidungsbaum zur Prognose von Angebotswerten

Entgegen der normalen Anordnung von Bäumen befinden sich hier Knoten einer Ebene auf den vertikalen Linien. Jedem Knoten wird ein Attribut aus der

Eingabemenge zugeordnet und dort wird ein Vergleich über den jeweiligen Wert durchgeführt. Den Knoten, an denen eine weitere Aufteilung nicht möglich oder sinnvoll gewesen wäre, ist eine konkrete Klasse zugeordnet. Die Werte in runden Klammern geben die Anzahl der dieser Klasse zugeordneten Testfälle an.

Die nächsten Abschnitte in diesem Kapitel beschäftigen sich mit konkreten Umformungen, Trainings und Tests auf den Daten.

4.5.1 Verteilung der Last auf einer Route

Bei den ersten Versuchen wurde die explizite Berücksichtigung der Zeit für den Lernvorgang vernachlässigt. Ziel war es, evtl. vorhandene Muster in den Angebotskurven zu erkennen und anhand dieser Merkmale die Klassifizierung vorzunehmen. Vorteil dieser Vorgehensweise ist, dass Zeitverschiebungen keine Rolle spielen, was z.B. im internationalen Verkehr von Bedeutung sein kann. In diesem Fall lassen sich die Prognosen auf die reinen Lasten im Netzwerk reduzieren, weswegen dieser Ansatz bei ähnlichen zu erreichenden Fehlerraten bevorzugt wird.

Median besitzt für die Berechnung Standardeinstellungen, die weitgehend an reale Systeme angepasst wurden. Das betrachtete Netz besteht aus 24 Knoten, die auf der für die Prognose relevanten logischen Ebene vollvermascht[3] sind. Die normale Taktung für die Berechnung des Netzzustandes beträgt 10 min, d.h. im Verlauf eines Tages fallen 144 Ausgaben (im realen System Messungen) an. Eine normale Simulation berechnet das Verhalten für 11 Durchläufe, jeder Durchlauf entspricht einem Tag mit 24 Stunden. Der erste Durchlauf kann für die Bewertung nicht herangezogen werden, da zu diesem Zeitpunkt das Netzwerk noch frei von aufgebauten Verbindungen[4] ist und das Verhalten nicht korrekt widerspiegelt. Somit stehen standardmäßig 10 Durchläufe mit insgesamt 1440 Datensätzen für Training und Test zur Verfügung.

Die Vergangenheitswerte (Historie) wurden alle gleich behandelt, es fand keine Gewichtung unterschiedlicher Werte statt. Die damit erzeugten Attribute verteilen sich dementsprechend linear über die Zeitskala. Für jede mögliche Verbindung stand bei diesem Beispiel nur eine Route zur Verfügung, die das gesamte auf dem Link anfallende Verkehrsaufkommen aufnehmen musste. Der Prognosehorizont für das anzugebende maximale Angebot betrug 3 Stunden.

Tabelle 4.1 stellt die Ergebnisse für unterschiedliche Entscheidungsbäume dar. Historie gibt dabei die Anzahl der Angebotswerte an, die als Attribute herangezogen und linear aus den bisherigen Angebotskurven ausgewählt wurden. Schrittweite bezeichnet die Größe des Intervalls für die Klassifizierung des maximalen Angebots in Erlang. Dieser Faktor ist neben der Anzahl an Fehlklassifikationen u.a. maßgebend für die Genauigkeit der Prognose. Je kleiner der Wert für die Schrittweite, umso höher die Genauigkeit (siehe Abbildung 4.6). Die folgenden drei Spalten geben den Fehler auf den Trainingsdaten, den

[3] Auf der logischen Ebene vollvermascht heißt in diesem Kontext, dass jeder einzelne Knoten mit jedem anderen kommunizieren kann, auch wenn keine direkte physikalische Verbindung besteht.

[4] Das Netz ist noch lastfrei, insbesondere existieren keine bestehenden Verbindungen aus vorherigen Durchläufen.

geschätzten Fehler und den Fehler auf den Testdaten an. Der Fehler auf den Trainingsdaten ergibt sich daraus, dass es in einigen Fällen nicht möglich ist, für manche Datensätze einen eindeutigen Weg durch den Entscheidungsbaum zu generieren. Dies tritt beispielsweise bei sehr ähnlich aussehenden Teilen des Kurvenverlaufs auf, die allerdings für die nächsten drei Stunden unterschiedliche Maxima besitzen. Dann widerspricht sich ein Teil der Datensätze.

Alle in der Tabelle gemachten Angaben beziehen sich auf die Werte, die nach dem Pruning des Entscheidungsbaumes erzielt wurden. Für den Test des generierten Baumes wurde jeweils ein Datensatz für einen kompletten Tag mit 144 Einträgen herangezogen. Die jeweiligen Werte sind die Mittel aus Versuchen, die mit unterschiedlichen Testdatensätzen erzeugt wurden.

Die durchschnittliche Rechenzeit für die Erzeugung eines Baumes lag bei ca. 5 s.

Historie	Schrittweite	Fehler Training	gesch. Fehler	Fehler Testdaten
5	25	17,9%	30,4%	44,4%
10	25	17,7%	27,1%	38,2%
15	25	18,3%	25,9%	36,1%
20	25	17,2%	25,3%	33,5%
25	25	16,1%	24,1%	30,9%
30	25	14,6%	24,0%	32,6%
5	50	1,5%	6,2%	7,2%
10	50	1,0%	5,1%	6,3%
15	50	1,3%	4,3%	6,3%
20	50	0,8%	4,0%	4,9%
25	50	0,4%	3,6%	4,8%
30	50	0,3%	3,2%	6,3%

Tabelle 4.1: Prognose ohne Berücksichtigung des Zeitindex

Wie aus Tabelle 4.1 zu ersehen ist, spielt die gewählte Schrittweite für die klassifizierten Intervalle eine entscheidende Rolle. Ist der Wert zu gering, so ist eine glaubwürdige Prognose nicht mehr möglich. Eine Schrittweite von 50 erzielt dagegen bereits akzeptable Fehlerraten. Allgemein lässt sich der Effekt erkennen, dass die Fehlerzahl auf den für den Entscheidungsbaum unbekannten Testdaten deutlich über der Anzahl an Fehlern für die Trainingsdaten liegt. Auch die anhand der Trainingsdaten geschätzte Rate ist in allen Fällen zu optimistisch ausgefallen.

Die Verlängerung der Historie führt in den meisten Fällen zu einer Verbesserung des Ergebnisses. Lediglich für eine Schrittweite von 50 kehrt sich der Effekt bei einer Anzahl von 30 Attributen wieder um. Zusätzlich steht dem Rechenaufwand für mehr als 15 Attribute eine prozentual geringere Verbesserung gegenüber.

C4.5 bietet die Möglichkeit, bis zu einer gewissen Anzahl an Klassen eine grafische Verteilung der Zuordnung der Testdaten zu den Klassen anzuzeigen. Abbildung 4.9 gibt eine solche Matrix wieder.

Vorteil dieser Darstellung ist, dass der Anwender Ausreißer und ihre Art (zu hoch, zu niedrig, extrem vom realen Ergebnis entfernt) sofort erkennen kann. Bei der falschen Zuordnung im Beispiel zur Klasse (d) käme es zu einer massiven Überdimensionierung des Netzwerks, bei Klasse (k) ist das genaue Gegenteil der Fall.

```
classified as
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m) (n) (o)
--- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
                                                           (a)
     13                                                    (b)
         30                                                (c)
             11                                 1          (d)
                                                           (e)
                      6                                    (f)
                                                           (g)
                              8   4                        (h)
                                  6                        (i)
                                                           (j)
      1                                  12                (k)
                                          1  26            (l)
                                              2  23        (m)
                                                           (n)
                                                           (o)
                                                    real class

(a): class [0-50[               (i): class [400-450[
(b): class [50-100[             (j): class [450-500[
(c): class [100-150[            (k): class [500-550[
(d): class [150-200[            (l): class [550-600[
(e): class [200-250[            (m): class [600-650[
(f): class [250-300[            (n): class [650-700[
(g): class [300-350[            (o): class [700-
(h): class [350-400[
```

Abbildung 4.9: Verteilungsmatrix der Klassifikation

Bei der Hinzunahme des Zeitindex als Attribut ergibt sich ein verändertes Bild. Es besteht die Möglichkeit, die Zeit als alleiniges Attribut oder in Kombination mit den herkömmlichen Angebotswerten zu verwenden. Beim letzteren Fall kommt der Zeit unterstützende Wirkung bei der Klassifikation zu. Dieser soll als erster betrachtet werden. Tabelle 4.2 greift aus Platzgründen nur einige der Fälle aus 4.1 wieder auf. Die Tabelle ist in drei Abschnitte unterteilt. Der Erste beinhaltet die Ergänzung der normalen Attribute um die Zeit, im Zweiten waren lediglich noch das Angebot zum jeweils aktuellen Zeitindex und die Zeit selbst vorhanden. Der dritte Teil berücksichtigt die Zeit als alleiniges Attribut. Im ersten Teil zeigen sich kaum Unterschiede zu der Prognose ohne Berücksichtigung der Zeit, insbesondere auf den Testdaten bei Fehlerraten von 6,3%. Die anderen beiden Abschnitte dagegen zeigen ein besseres Verhalten hinsichtlich der Fehlerrate, das allerdings erst bei der Berücksichtigung der Testdaten zu Tage tritt. Hier sind Fehlerraten von 2% und weniger erzielbar. Die Werte für die genauere Klasseneinteilung bei einer Schrittweite von 25 Erlang bewegen sich dagegen mit mehr als 25% immer noch außerhalb eines akzeptablen Rahmens für den praktischen Einsatz; die Rate der Fehlklassifikationen macht einen großen Teil des Genauigkeitsgewinns für die kleinere Schrittweite wieder zunichte.

Historie	Schrittweite	Fehler Training	gesch. Fehler	Fehler Testdaten
Gesamte Historie und Zeitindex				
15	50	1,4%	3,8%	6,3%
15	25	22,1%	26,8%	32,5%
Aktuelles Angebot und Zeitindex				
15	50	1,9%	3,9%	2,1%
15	25	25,8%	30,2%	27,1%
Zeit als Attribut				
15	50	2,0%	3,8%	1,4%
15	25	25,1%	29,9%	27,2%

Tabelle 4.2: Prognose ohne Berücksichtigung des Zeitindex

Alle bislang angegebenen Experimente sind mit einem Prognosehorizont von drei Stunden[5] durchgeführt worden. Aus Tests mit Median und der Simulation des Netzwerks hat sich dieser Wert als günstig erwiesen, da sich innerhalb dieses Zeitraumes sowohl das Netzwerk sinnvoll rekonfigurieren lässt, als auch der Prognosewert den zukünftigen Zustand des Netzes wiedergibt. Trotzdem ist ebenfalls interessant, ob und wie sich die Prognosegenauigkeit verändert, wenn andere Prognosezeiträume gewählt werden, falls unterschiedliche Anforderungen längerfristige Planung oder schnelle Reaktionen erfordern. Tabelle 4.3 gibt die Raten für unterschiedliche Prognosehorizonte wieder (siehe zur Bedeutung unterschiedlich langer Horizonte auch Abbildung 2.4 auf Seite 17). Die Versuche wurden für eine Historie von 15 Werten und eine Schrittweite von 50 Erlang durchgeführt. Der erste Teil der Tabelle enthält die Ergebnisse für die Bewertung ohne Berücksichtigung der Zeit, der zweite Teil mit Zeitindex zusätzlich zum aktuellen Angebot.

Im Vergleich zu diesen Werten zeigt sich der Standardhorizont von 3 Stunden ohne Zeitindex erstaunlicherweise als besser oder zumindest gleichbleibend stabil. Mit Zeitindex kann eine Verbesserung erzielt werden, die Einstellung von 3 Stunden wird allerdings für die zukünftigen Versuche als Standard beibehalten, da dieser Wert bei den Rekonfigurationen von Median als maßgebend bestimmt wurde.

[5] Bei einem Ausgabetakt der Werte durch Median von 10 min entspricht dies 18 Werten.

Horizont	Fehler Training	gesch. Fehler	Fehler Testdaten
Ohne Zeitindex			
6	1,0%	5,9%	7,6%
12	1,1%	5,9%	7,6%
24	1,1%	4,8%	6,9%
Mit Zeitindex			
6	3,7%	6,5%	3,5%
12	3,8%	6,3%	2,8%
24	1,9%	3,8%	2,1%

Tabelle 4.3: verkürzter/verlängerter Prognosehorizont

4.5.2 Verteilung der Last über drei Routen

Im vorhergehenden Abschnitt stand für den Verbindungsaufbau zwischen zwei Endpunkten jeweils nur ein möglicher Weg zur Verfügung. Die gesamte anfallende Last muss somit von einer einzelnen Route aufgefangen werden, die zwar über mehrere Zwischenstationen (Hops) geleitet werden kann, deren Verlauf aber bei der Dimensionierung des Netzwerks festgelegt ist. In der Realität führt dies zu einem schlechteren und unflexibleren Verhalten. Auf Engpässe, die entstehen, wenn zwei breitbandige Dienste denselben Leitungsabschnitt benutzen, kann nicht reagiert werden. Leitungsausfälle führen zum vollständigen Ausfall der Verbindung zwischen zwei Knoten.

Aus diesen Gründen bietet Median die Möglichkeit, für jede Punkt-zu-Punkt Verbindung bis zu drei Wege anzubieten, die aus Gründen der Ausfallsicherheit knotendisjunkt sind und alternativ zueinander benutzt werden können. Für das Routing über die Wege lässt sich eine Verteilung angeben. Damit ergeben sich für jeden Dienst auf jeder Verbindung zwei zusätzliche Angebotswerte. In Abbildung 4.3 auf Seite 37 waren diese Werte noch mit 0 angegeben.

In diesem Abschnitt wird der Fragestellung nachgegangen, wie sich die Fehlerrate von C4.5 verhält, wenn die dreifache Anzahl an Attributen zum Lernen zur Verfügung steht. Prinzipiell besteht die Chance, dass sich durch die Verteilung des Angebots auf drei Links für den Lernprozess eher eindeutige Muster ergeben. Ein großer Nachteil ist allerdings sofort absehbar: Ändert sich aus zwingenden Gründen die Verteilung der Last auf den drei Routen, kann der Algorithmus keine Prognose liefern, da andere Voraussetzungen herrschen. Somit wäre ein neuer Lernprozess nötig. Je nach Häufigkeit der Änderungen im Netzwerk kann u.U. die Verfügbarkeit einer wertvollen Prognose nicht gewährleistet werden.

Durch die Benutzung aller drei möglichen Alternativwege verdreifacht sich dementsprechend die Anzahl an Attributen und damit steigt auch die Größe und Komplexität der Entscheidungsbäume. Die Rechenzeit erhöht sich pro Baum von ca. 5 Sekunden auf 8 Sekunden bei einer Lernmenge von 10 Durchläufen.

Die Ergebnisse zeigen, dass es sinnvoll ist, die Verteilung des Angebots auf den drei Routen zu einem Angebot zusammenzufassen. Die Behandlung von einer oder mehreren Routen im Rahmen der Vorhersage wäre somit äquivalent.

Historie	Schrittweite	Fehler Training	gesch. Fehler	Fehler Testdaten
Ohne Zeitindex				
15	50	4,4%	20,1%	43,6%
20	50	3,8%	16,4%	41,2%
Mit Zeitindex				
15	50	4,1%	18,1%	38,2%
20	50	3,9%	18,2%	37,5%

Tabelle 4.4: Prognose mit Berücksichtigung von 3 Routen

Ähnlich wie bei der Einteilung der Vorhersage in kleinere Intervalle liegt hier eine starke Diskrepanz zwischen dem geschätzten Fehler und demjenigen auf den Evaluierungsdaten. Vorteil bei der Zusammenfassung ist, dass die Aufteilung des Gesamtangebots auf die Routen an die Managementmodule von Median übertragen werden kann, was eine flexiblere Verteilung und damit die Ausnutzung des Bündelungsgewinns ermöglicht. Bei Betrachtung des Angebots auf den einzelnen Routen im Gegensatz zum Angebot des Links wäre zusätzlich nach jeder Rekonfiguration ein neuer Lernvorgang nötig, da sich die Verteilungen des Angebots auf den Routen und dementsprechend auch die zugrunde liegenden Muster verändert haben.

4.5.3 Diskrete Attribute

Während C4.5 nur die Vorhersage diskreter Klassen ermöglicht, sind für die Attribute sowohl diskrete als auch kontinuierliche Werte zugelassen. Für kontinuierliche Werte ergeben sich bei jeder Aufteilung des Baumes zwei Wege, die anhand eines Schwellwertvergleichs ermittelt werden. Bei diskreten Attributen wird für jede mögliche Ausprägung der Klasse ein Pfad bereitgestellt.

Historie	Schrittweite	Fehler Training	gesch. Fehler	Fehler Testdaten
Ohne Zeitindex, kontinuierliche Attribute				
15	50	1,3%	4,3%	6,3%
20	50	0,8%	4,0%	4,9%
Ohne Zeitindex, diskrete Attribute				
15	50	2,9%	8,4%	7,6%
20	50	2,2%	8,1%	8,3%
Mit Zeitindex, kontinuierliche Attribute				
15	50	1,4%	3,8%	6,3%
20	50	0,3%	3,1%	4,9%
Mit Zeitindex, diskrete Attribute				
15	50	2,8%	7,9%	7,7%
20	50	2,5%	7,1%	7,1%

Tabelle 4.5: Diskrete Attributwerte

Auch wenn das betrachtete Problem grundsätzlich kontinuierlicher Natur ist, stellt sich die Frage, ob eine künstliche Diskretisierung der zur Prognose herangezogenen Attribute eine Verbesserung der Vorhersage mit sich bringt. Unter Umständen könnten relevante Muster eher zu Tage treten als im verrauschten Originalzustand.

Dazu werden analog zur Schrittweite für die Zielklassen Intervalle für die Einteilung der Angebotswerte eingeführt und bei der Transformierung der Daten als Eingabe für C4.5 die neuen Werte ausgegeben (Beispiel: Angebotswerte 14 und 32 bekommen den diskreten Wert #50, 101 und 149 analog #150 zugeordnet). Anschließend werden die Versuche mit den diskreten Attributen wiederholt. Um einen bequemeren Vergleich zu ermöglichen, wiederholen sich in der Tabelle 4.5 die entsprechenden Einträge mit kontinuierlichen Attributen aus vorherigen Tabellen.

Durch die Verfolgung jeder Ausprägung an den Knoten des Baumes besitzt dieser eine wesentlich größere Breite.

Die Rechenzeit zur Generierung dieser Bäume steigt um ca. 50% im Vergleich zur Benutzung der kontinuierlichen Attribute und es lässt sich bei den durchgeführten Experimenten keine Verbesserung der Fehlerrate erkennen, so dass sich der Einsatz von diskretisierten Attributwerten nicht empfiehlt.

4.5.4 Adaptive Zielklassengrößen

In allen bisherigen Untersuchungen ist die Intervallgröße der Zielklassen als statisch betrachtet worden. Anhand der zuvor festgelegten Schrittweite für die Unterteilungen sind die Ergebnisklassen fest bestimmt worden. Bei näherer Betrachtung der Angebote zu unterschiedlichen Zeitpunkten zeigt sich, dass die Schwankung und damit die Fluktuation an belegten Leitungen bei höheren Angeboten verhältnismäßig stärker ist als bei niedrigeren. Da stärkere Schwankungen in der Regel eine Vorhersage schwieriger machen, ist es zumindest einen Versuch wert, diese Schwankung im Vorhersagewert durch eine angepasste Intervallgröße der Zielklassen zu kompensieren. Damit erhöht sich für die höheren Angebote die Ungenauigkeit der Vorhersage, da der durch das Intervall abgedeckte Bereich für das Maximum der nächsten Stunden entsprechend steigt.

Historie	Basisschrittw.	Fehler Training	gesch. Fehler	Fehler Testd.
Ohne Zeitindex				
15	50	1,5%	5,9%	4,9%
15	40	8,3%	10,2%	9,9%
Mit Zeitindex				
15	50	10,0%	12,3%	4,9%
15	40	17,4%	20,8%	23,8%

Tabelle 4.6: Adaptive Zielklassengröße

Das Transformationsprogramm bietet dazu eine Option, die es erlaubt, für höhere Angebote einen Steigerungsfaktor anzugeben. Die Intervallgrößen werden entsprechend diesem Faktor erhöht. Tabelle 4.6 gibt die Fehler für eine

Steigerung von 10% analog zur künstlichen Schwankung des Simulators wieder. Bei einer Basisschrittweite von 50 deckte somit die erste Klasse den Bereich 0–50 Erlang, die zweite 50–55 usw. ab. In Tabelle 4.6 ist jeweils die Schrittweite, die als Ausgangsgröße eingesetzt wurde, angegeben. Ein Wert von 40 sorgt dabei dafür, dass die Steigerung sich erst im mittleren Angebotsbereich auswirkt, d.h. zunächst sind die Intervalle ein wenig kleiner als zuvor üblich und übersteigen im weiteren Verlauf die Schrittweite 50.

Die Basisschrittweite von 40 führt zu einer starken Verschlechterung der Ergebnisse. Die übrigen Fehlerraten bewegen sich in ähnlichen Größenordnungen wie mit fester Schrittweite von 50. Durch die veränderliche Schrittweite wird für die Generierung der Daten und der Entscheidungsbäume kein zusätzlicher Aufwand erzeugt, so dass von Seite der Performanz aus gesehen nichts gegen den Einsatz adaptiver Klassifikationsintervalle spricht.

4.5.5 Kürzere Taktung

Standardmäßig liefert die Simulation Ausgaben in Abständen von 600 Sekunden, womit sich für einen vollständigen Tag von 86400 Sekunden 144 Werte ergeben. Dieses Intervall ist in gewissen Grenzen frei wählbar, so dass zu untersuchen ist, ob eine kürzere Taktung und damit eine höhere Anzahl an Attributen zu einer Verbesserung des Vorhersageergebnisses führt. Die Ergebnisse sind in Tabelle 4.7 wiedergegeben. Die Taktung für die Ausgabe der Angebote im Netz durch die Simulation war um den Faktor 10 schneller bei 60 s.

Historie	Schrittweite	Fehler Training	gesch. Fehler	Fehler Testdaten
Ohne Zeitindex				
15	50	9,8%	20,1%	33,8%
20	50	8,2%	18,9%	33,0%
25	50	7,7%	18,0%	31,5%
100	50	4,1%	11,1%	24,9%
150	50	2,6%	9,5%	23,6%
200	50	1,9%	8,8%	23,8%
250	50	1,1%	7,8%	25,6%
Mit Zeitindex				
15	50	13,8%	17,2%	19,3%
20	50	12,1%	15,1%	16,4%
25	50	11,2%	14,6%	16,5%
100	50	3,7%	9,7%	22,0%
150	50	2,6%	8,9%	22,4%
200	50	1,3%	8,2%	25,5%
250	50	1,0%	7,5%	23,5%

Tabelle 4.7: Prognose auf Daten mit kurzem Ausgabetakt

Neben der bereits bekannten Unterteilung mit und ohne Berücksichtigung des Zeitindex kamen hier stark unterschiedliche Größenbereiche für den Vergangenheitszeitraum zum Einsatz. Zunächst wird die Historie für einen Bereich von 15–25 Ausgabezeitpunkten betrachtet. Da sich im Gegensatz zu den übrigen Versuchen der Takt um den Faktor 10 erhöht hat, decken diese Intervalle einen wesentlich kleineren Bereich über der Zeit ab. Die Anzahl der für den Lernvorgang zur Verfügung stehenden Attribute bleibt dabei konstant im Vergleich zur 600 Sekunden-Taktung. Der zweite Bereich wurde dementsprechend um den Faktor 10 ausgeweitet, um sicherzustellen, dass Besonderheiten im Angebotsverlauf (wie z.B. grundlegende Veränderungen, die zuvor durch die Attributauswahl abgedeckt wurden) weiterhin berücksichtigt werden.

Problematisch erweist sich die kürzere Taktung insbesondere in Bezug auf Speicherbedarf und Rechenzeit. Sowohl die Transformierung der Simulationsdaten für C4.5 als auch die eigentliche Baumgenerierung benötigen für die Berechnungen mit einer Historie von mehr als 100 Werten eine Rechenzeit von mindestens 5 Minuten. Der erhöhte Speicherbedarf verringert — endlichen Speicherplatz vorausgesetzt — gleichzeitig die maximale Anzahl an zur Verfügung stehenden Durchläufen.

Im Vergleich von Ressourcenbedarf und erzielbarer Fehlerrate empfiehlt sich der Einsatz einer kürzeren Taktung als der ursprünglich vorgesehenen nicht.

4.5.6 Kreuzvalidierung

Um gute Abschätzungen zukünftiger Fehlklassifikationen zu ermöglichen, ist die Wahl geeigneter Testdaten entscheidend. Die Daten spiegeln nur dann die späteren Verhältnisse wider, wenn die Repräsentativität ausreichend ist. Sicherheit in diesem Bereich kann eine Kreuzvalidierung (engl. Crossvalidation) bieten. Dazu werden aus der Menge der Gesamtdaten zufällig einzelne Daten herausgesucht und zur Evaluierung bereitgestellt. Der Vorgang wiederholt sich für eine gegebene Anzahl an Blöcken, so dass jeweils Paare aus Lern- und Testmengen entstehen, wobei die Mengen jedes Paares zueinander disjunkt sein müssen, damit die einzelnen Trainings-/Testläufe voneinander unabhängig sind.

Anschließend werden für jedes Paar getrennt Lern- und Testläufe durchgeführt. Die Fehlerrate wird über alle Durchläufe gemittelt um eine stabilere Einschätzung zukünftiger Fehler zu ermitteln. Der durch Kreuzvalidierung ermittelte Fehler sollte sich dabei nicht stark von den zuvor ermittelten Fehlern unterscheiden. In günstigen Fällen ist er nur leicht schlechter als der Wert bei der Bereitstellung ganzer Durchläufe zur Überprüfung der generierten Bäume. Beispiele für die Versuche mit Kreuzvalidierung gibt Tabelle 4.8 wieder, zum besseren Vergleich wieder mit den Ergebnisse eines einzelnen Testdurchlaufs. Angaben über den geschätzten Fehler liegen bei der automatischen Kreuzvalidierung durch C4.5 nicht vor.

Die Ergebnisse bestätigen weitestgehend die Tendenzen, die bereits ohne Kreuzvalidierung zu erkennen waren. So erzielt die kurze Schrittweite zur Klasseneinteilung von 25 auch hier Fehlerraten von mehr als 30%. Andererseits konnten in manchen Fällen durch die Kreuzvalidierung die Fehler auf unbekannten Testdaten nach unten korrigiert werden.

Historie	Basisschrittw.	Fehler Training	Fehler Testdaten
Ohne Zeitindex			
15	50	1,3%	6,3%
15	25	18,3%	36,1%
Ohne Zeitindex, Kreuzvalidierung			
15	50	0,8%	3,6%
15	25	16,1%	38,2%
Mit Zeitindex			
15	50	1,4%	6,3%
Mit Zeitindex, Kreuzvalidierung			
15	50	2,2%	2,8%

Tabelle 4.8: Kreuzvalidierung

4.5.7 Logarithmische Attributauswahl

Der Grundgedanke hinter dieser Art der Attributvorverarbeitung ist, weiter in der Vergangenheit zurückliegende Zeitpunkte vermindert für das Lernverfahren einzusetzen. Je weiter ein Wert vom aktuellen Zeitpunkt aus gesehen liegt, um so geringer ist seine Aussagekraft bezogen auf die Prognose für die naheliegende Zukunft. Für die in Tabelle 4.9 dargestellten Ergebnisse wurde die Auswahl der Attribute logarithmisch gestaffelt, d.h. entsprechend zur Basis des gewählten Logarithmus werden in Richtung der Vergangenheit immer weniger Attribute ausgewählt.

Da sich die Schrittweite von Attribut zu Attribut exponentiell erhöht, wird bei linearer Schrittweite teilweise die gleiche Lerndatei wie zuvor erzeugt, was dementsprechend auch zu demselben Ergebnis führt. Konkret heißt dies, dass z.B. ein Logarithmus zur Basis 10 für Historien von 15 und 20 die gleichen Attribute auswählt, da als Attribute die Angebotswerte 1, 11 und dann erst wieder 101 für den Lernprozess ausgewählt werden. Für den Logarithmus zur Basis 2 ist dies in der Tabelle beispielhaft ersichtlich, in den anderen Fällen erfolgt keine lineare Auswahl der Länge der Historie. Für den bereits recht großen Logarithmus zur Basis 10 zeigt sich eine Änderung erst wieder für den Bereich von 120 Takten, was sich lediglich in einer Verschlechterung der Fehlerzahl auf den Testdaten auswirkt. Problematisch sind große Schritte entlang der Vergangenheit zusätzlich dadurch, dass die Transformation der Daten eine Verarbeitung aller Takte benötigt, von denen anschließend nicht alle für den Lernabschnitt eingesetzt werden. Der Lern- und Vorhersageprozess selbst benötigt dagegen durch die verringerte Auswahl von Attributen eine kürzere Rechenzeit als bei konstanter Einteilung über einen gleich großen Bereich entlang der Zeitachse, bietet aber teilweise deutlich schlechtere Ergebnisse, so dass Zeitbedarf zur Auswahl dieser Umformung nicht das alleinige Kriterium sein sollte.

Historie	Logarithmus	Fehler Training	gesch. Fehler	Fehler Testdaten
Ohne Zeitindex				
10	2	4,2%	10,1%	13,9%
15	2	4,4%	9,4%	11,8%
20	2	4,3%	9,4%	11,8%
25	2	4,0%	9,2%	11,1%
10	e	8,0%	15,8%	18,8%
15	e	4,2%	7,9%	8,3%
10	10	17,4%	25,1%	25,0%
15	10	2,3%	6,2%	4,9%
120	10	2,3%	5,4%	4,9%
Mit Zeitindex				
10	2	1,9%	3,9%	2,1%
10	e	1,9%	3,9%	2,1%
10	10	1,9%	3,9%	2,1%
120	10	1,9%	3,9%	2,8%

Tabelle 4.9: Logarithmische Auswahl der Attribute

4.5.8 Berücksichtigung der Steigung

Ein großer Vorteil von Techniken des maschinellen Lernens hinsichtlich der Qualität der Vorhersage gegenüber einfacheren statistischen Verfahren liegt in der hohen Flexibilität der Eingabemöglichkeiten. Es ist ohne weiteres möglich, quantitativ nur schlecht bewertbare Attribute[6] mit Größen wie hier dem Angebot zusammen zu bewerten und Relationen zu erkennen. Daraus folgt, dass ein nicht zu vernachlässigender Bereich des Data Mining in der Konstruktion von gänzlich neuen Attributen besteht.

Im Zusammenhang mit der Bewertung von Lasten in ATM-Netzwerken fällt dieser Ansatz zunächst nicht ganz leicht. Es stehen nur reine Angebotswerte mit ihrem Zeitbezug zur Verfügung. Denkbar wäre es aber auch, zusätzliche benutzerabhängige Daten mit in die Bewertung einzubeziehen. Möglich wäre, dass sich z.B. das Telefonverhalten in Abhängigkeit vom Wetter, von der Jahreszeit (Sonnenscheindauer...) oder vor und nach großen Ereignissen (Fußballspiele...) ändert. Solche Daten werden von der Simulation aber weder bereitgestellt noch berücksichtigt, so dass als einziger Ansatz die Verwertung der vorhandenen Daten bleibt.

Hier wird zu den aufeinanderfolgenden Angebotswerten die Steigung berechnet und als zusätzliches Attribut zur Verfügung gestellt. Der Gedanke dahinter ist, C4.5 Besonderheiten im Kurvenverlauf, wie schnelle, plötzliche Anstiege oder relativ gleichbleibende Werte, besser erkennen zu lassen. Neben den herkömmlichen Mustern des Kurvenverlaufs aus dem Angebot besteht die

[6]Beispiel: Für den Erfolg von Direktmailing-Aktionen werden die potenziellen Empfänger häufig in Klassen entsprechend ihres Lebenswandels, persönlicher Einstellung usw. unterteilt, die nur subjektiv bewertbar sind.

Möglichkeit, signifikante Muster (sehr steile Anstiege, flache Abschnitte...) im Verlauf der Steigungen zu entdecken.

Historie	Schrittw.	Fehler Training	gesch. Fehler	Fehler Testdaten
Ohne Zeitindex				
10	50	0,9%	4,9%	6,9%
15	50	1,3%	4,2%	4,2%
20	50	0,5%	4,5%	6,3%
25	50	1,7%	5,2%	4,9%
30	50	0,8%	4,6%	4,9%
Mit Zeitindex				
10	50	0,5%	3,3%	4,9%
15	50	1,5%	4,2%	3,5%
20	50	0,9%	3,4%	4,2%
25	50	1,3%	3,5%	3,5%
30	50	0,2%	2,7%	4,9%

Tabelle 4.10: Berücksichtigung der Steigung als Attribut

Die erzielbaren Fehlerraten bewegen sich im allgemeinen Rahmen. Auffällig ist die Verschlechterung bei einer Historie von 20 Takten. Eine Wiederholung des Experiments mit anderen Testdaten im Sinne der Kreuzvalidierung lieferte allerdings ähnliche Ergebnisse, was bedeutet, dass an dieser Stelle Anomalien im Kurvenverlauf eine Rolle spielen könnten. Dadurch, dass die Steigung als Attribut dazugekommen ist, verdoppelt sich die Anzahl an benutzten Attributen, was sich entsprechend auf die Rechenzeit auswirkt.

4.5.9 Auswertung entlang mehrerer Links

Der hier gewählte Ansatz fällt gegenüber den bisher dargestellten aus dem Rahmen. Anstatt die Attribute streng entlang des Zeitstrahls auszuwählen, wird hier der Zeitpunkt „räumlich" über mehrere Links ausgedehnt (vergleiche Abbildung 4.7 auf Seite 41). Es ist vorstellbar, dass aussagekräftige Zeitpunkte im Netz bestehen, anhand derer sich in globaler Sicht über mehrere Links Prognosen erstellen lassen, z.B. Zeiten, zu denen allgemein viel telefoniert wird. Prinzipiell lassen sich dazu alle 552 im Standardnetz zur Verfügung stehenden Links benutzen. An dieser Stelle werden daraus lediglich die 23 vom jeweiligen Knoten abgehenden Verbindungen berücksichtigt, um die Anzahl an Attributen einzugrenzen. Einen Auszug der Fehlerraten für unterschiedliche Links enthält Tabelle 4.11.

Beim Versuch für mehrere Links eine Vorhersage zu ermöglichen, zeigt sich, dass die Fehlerrate auch für verschiedene Links recht stabil bleibt, gegenüber den anderen Datenaufbereitungen aber keinen Gewinn an Genauigkeit bietet. Der Aufwand zur Transformation der Daten für C4.5 steigt um ein Vielfaches an, da für jeden berücksichtigten Link die zugehörige Textdatei verarbeitet werden muss. Beschleunigen ließe sich der Zugriff, wenn die Daten nicht in

Von	Nach	Fehler Training	gesch. Fehler	Fehler Testdaten
0	1	3,4%	8,5%	9,7%
0	2	2,3%	7,4%	9,1%
0	3	3,0%	8,2%	9,6%
1	5	3,9%	9,2%	9,7%

Tabelle 4.11: Angebotswerte mehrerer Links

Textdateien, sondern in einer entsprechend aufbereiteten mehrdimensionalen Datenbank gehalten würden. Damit könnten auf einfache Art zusätzlich weitere Kombinationen von Links gebildet werden, die über das hier benutzte Verfahren mit der Berücksichtigung aller Links des betroffenen Knotens hinausgehen.

4.5.10 Zusammenfassung

Vorteil von C4.5 ist die schnelle und einfache Verfügbarkeit, weswegen dieses Verfahren als erstes zum Einsatz gekommen ist. Großer Nachteil ist, dass lediglich eine Unterstützung für diskrete Attribute besteht, was neben der Fehlerrate der Klassifikation die Ungenauigkeit erhöht. Vorteil ist die recht hohe Verarbeitungsgeschwindigkeit von C4.5, die dafür sorgt, dass Lernergebnisse bei einer moderaten Anzahl von Attributen auch für Rechner im mittleren Leistungsbereich bereits nach Sekunden vorliegen, Prognosen für neue Daten bereits nach Sekundenbruchteilen.

Wie die einzelnen Experimente mit verschiedenen Datentransformationen gezeigt haben, lassen sich bereits mit recht einfachen Mitteln akzeptable Fehlerraten erzeugen. Gleichzeitig sind Grenzen erkennbar, z.B. bei der Schrittweite der Diskretisierung der Klassifikationsintervalle. Der Versuch, hier die Genauigkeit zu steigern, führt zu einem starken Ansteig der Fehlerrate bezüglich der Klassifikation und kehrt den gewünschten Effekt damit wieder um.

Wie für die Prognose einer Zeitreihe schon fast zu erwarten, hat die Hinzunahme des Zeitindex als Lernattribut in den meisten Fällen die Vorhersagegenauigkeit weiter gesteigert, weswegen dieser Ansatz mit zu den vielversprechendsten gehört.

Als stabil, auch für unterschiedliche Diensttypen, haben sich Historien von 15–20 Attributen, Schrittweiten von 50 Erlang (was bei einem maximalen Angebot innerhalb der Testdaten von 750 Erlang 15 verschiedenen Klassen entspricht) und die Hinzunahme des Zeitindex zur Lernmenge erwiesen. Vom Bedarf an Rechenzeit sind diese Einstellungen ebenfalls besser als aufwändigere Transformationen, die z.B. deutlich mehr Attribute verarbeiten müssten.

Kapitel 5

Support Vector Machine

Dieses Kapitel widmet sich den Experimenten mit der *Support Vector Machine (SVM)*. Auch wenn die zugrundeliegenden Theorien schon seit längerem bekannt sind, wird gerade in der letzten Zeit der SVM vermehrt Aufmerksamkeit geschenkt. Einsatzgebiete sind weit gestreut, existieren doch Varianten sowohl zur Klassifikation als auch zur Regression. Für die Durchführung der hier angegebenen Experimente wurde die Regressionsvariante benutzt, zum einen, da diese Form der Art der Prognose eher gerecht wird, zum anderen, um auch in diesem Punkt zwischen C4.5 und SVM zu differenzieren. Wie bei den meisten Lernverfahren üblich, spielt auch hier die Generalisierung eine wesentliche Rolle. Orientiert sich das Verfahren zu stark an den Trainingsdaten, kann das Ergebnis nicht zwangsläufig auf unbekannte Daten übertragen werden, im anderen Fall der zu starken Generalisierung lässt sich keine Aussage über zugrundeliegende Muster machen.

Der Teil dieses Kapitels, der sich den theoretischen Grundlagen widmet, baut weitgehend auf [Burges98], einem Tutorial zur Support Vector Machine, auf. Eine Diplomarbeit, die sich hauptsächlich mit der SVM zur Zeitreihenanalyse und asymmetrischen Kostenfunktionen beschäftigt, ist in [Rüping99] zu finden.

5.1 Messung des Risikos

Eine der grundsätzlichen Fragen, die sich während des Lernvorgangs stellt, ist, wie gut die Maschine bereits trainiert ist. Zu diesem Zweck werden unterschiedliche Maße eingeführt.

Vorausgesetzt sei eine Menge von l Beobachtungen, bzw. in diesem Zusammenhang Trainingsdaten. Jede Beobachtung i bestehe aus einem Vektor $\vec{x}_i \in \mathbb{R}^n$, $i = 1, \ldots, l$ und dem zugeordneten Ergebnis y_i, das als korrekt angesehen wird. Weiter wird angenommen, dass eine Wahrscheinlichkeitsverteilung $P(\vec{x}, y)$ existiert, die für die Generierung der Daten maßgebend ist. Die Datensätze werden als unabhängig „gezogen" und identisch verteilt betrachtet.

Allgemein ist die Aufgabe einer lernenden Maschine, die Abbildung $\vec{x}_i \to y_i$ zu finden, was in der Regel auf eine Menge möglicher Funktionen $\vec{x} \to f(\vec{x}, \alpha)$ zurückzuführen ist. α ist der zu lernende Parameter; eine Maschine mit bestimmtem α wird als *trainiert* bezeichnet.

Der erwartete Testfehler einer solchen Maschine ergibt sich nach:

$$R(\alpha) = \int \frac{1}{2}|y - f(\vec{x}, \alpha)| dP(\vec{x}, y).$$

Die Größe $R(\alpha)$ wird *erwartetes Risiko* oder einfach *Risiko* genannt.

In Ergänzung zum erwarteten Risiko wird das *empirische Risiko* definiert als die Größe des gemessenen mittleren Fehlers auf dem Trainingsdatensatz:

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^{l} |y_i - f(\vec{x}_i, \alpha)|.$$

Die Größe $\frac{1}{2}|y_i - f(\vec{x}_i, \alpha)|$ wird als *Loss* bezeichnet und gibt die Abweichung zwischen vorhergesagtem und realem Wert an, weswegen es ebenfalls zur Fehlerbewertung herangezogen wird.

Für die Berechnung des empirischen Risikos wird keine konkrete Wahrscheinlichkeitsverteilung benötigt, deren Bestimmung in vielen Fällen Probleme bereitet, sondern es besteht lediglich eine Abhängigkeit von α und einem bestimmten Trainingsdatensatz $\{\vec{x}_i, y_i\}$.

Das empirische Risiko besitzt allerdings einige Schwächen, so z.B. hinsichtlich Genauigkeit oder Konsistenz in Bezug zum realen Fehler. Eine Frage beschäftigt sich deshalb mit einer Grenze für das erwartete Risiko. Vapnik ([Vapnik98]) gibt als Grenze für ein η mit $0 \leq \eta \leq 1$ und ein Loss z mit $0 \leq z \leq 1$ und der Wahrscheinlichkeit $1 - \eta$

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log_2(2l/h) + 1) - \log_2(\eta/4)}{l}}$$

an. Zur Bestimmung dieser Grenze wird eine beliebige Verteilungsfunktion $P(\vec{x}, y)$ vorausgesetzt, die allerdings nicht bekannt sein muss. Notwendig ist dagegen die Wahl eines fixen, hinreichend kleinen η, um sich dann für die Maschine zu entscheiden, welche die rechte Seite der Gleichung minimiert und damit die kleinste obere Grenze für das erwartete Risiko liefert. Ebenfalls bekannt sein muss der Parameter h, die VC Dimension. Dieser Begriff wird im nächsten Abschnitt näher erläutert. Der Term unterhalb der Wurzel auf der rechten Seite der Ungleichung wird dementsprechend als *VC Konfidenz* bezeichnet.

5.2 Die VC Dimension

Die Vapnik-Chervonenkis-Dimension ist eine Eigenschaft einer Funktionsschar $\{f(\alpha)\}$, wobei α eine Menge von Parametern darstellt, die eine bestimmte Funktion spezifizieren. Die nachfolgenden Erläuterungen beziehen sich auf den einfachen Spezialfall einer Mustererkennung für den Fall zweier Klassen, d.h. $\{f(\vec{x}, \alpha) \in \{-1, 1\} | \forall \vec{x}, \alpha\}$. Eine Menge von l Punkten kann demzufolge auf 2^l Arten diesen Klassen zugeordnet werden. Ziel einer Klassifikation ist es, die möglichen Punktmengen durch eine trennende Funktion f den richtigen Klassen zuzuordnen, bestimmt durch die Lage der Punkte in Relation zu f. Für die VC Dimension ergibt sich die folgende Definition:

Definition 6
Die *VC Dimension* einer Funktionsschar $\{f(\alpha)\}$ ist die maximale Anzahl an Trainingspunkten, die durch $\{f(\alpha)\}$ entsprechend der Zugehörigkeit zu den Klassen $\{-1, 1\}$ getrennt werden können.

Zu beachten ist, dass diese Definition lediglich garantiert, für eine VC Dimension von h mindestens *eine* Menge von h Punkten finden zu können, die durch f trennbar ist, nicht jedoch allgemein für *jede* Menge von h Punkten. Abbildung 5.1 gibt ein Beispiel für drei Punkte innerhalb des $\mathbb{R}^2$, die durch eine Gerade entsprechend ihrer Klasse voneinander getrennt werden. Es ist kein Problem, für eine Geradenschar drei Punkte zu finden, die getrennt werden können, für vier Punkte ist dies allerdings nicht möglich. Dementsprechend ist die VC Dimension für diesen Fall 3, oder allgemein für eine Menge orientierter Hyperebenen im $\mathbb{R}^n$ $n+1$. Das Prinzip der VC Dimension zur Trennung in zwei Klassen läßt sich auch bei reellen Werten anwenden, wozu diese Funktionen wieder auf den einfachen Klassifikationsfall zurückgeführt werden.

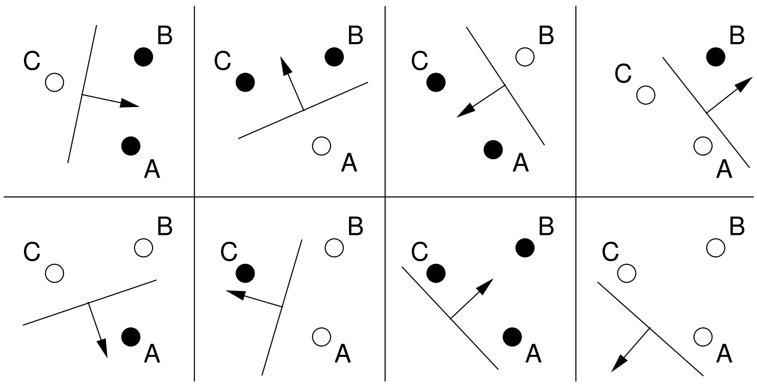


Abbildung 5.1: Alle 8 Möglichkeiten, drei Punkte mit zugeordneter binärer Klasse linear voneinander zu trennen und entsprechend zu klassifizieren.

Zwei Parameter, die somit die obere Schranke für das erwartete Risiko maßgeblich mitbestimmen, sind die Anzahl der Trainigsdaten l und die VC Dimension h. Der Trainingsvorgang besteht nun darin, diese obere Schranke in der Hoffnung zu minimieren, damit auch das reale Risiko zu minimieren. Für $l \gg h$ wird die obere Schranke weitgehend durch das empirische Risiko bestimmt, bei großem h dagegen ergibt sich eine monoton wachsende Funktion für die Schranke. Für eine Menge möglicher lernender Maschinen fällt die Wahl bei gleichem empirischen Risiko damit auf die Maschine, welche die minimale VC Dimension besitzt. Dies führt, wie Gegenbeispiele ([Burges98]) zeigen, nicht in jedem Fall zu einer Minimierung des Risikos, Versuche bescheinigen dieser Taktik allerdings häufig gute Ergebnisse.

Die beschriebene Vorgehensweise führt zum Prinzip der *strukturellen Risikominimierung (SRM)*. Der VC Konfidenz-Term in der oberen Schranke hängt von der gewählten Klasse von Funktionen ab, während das empirische Risiko durch die konkret während des Trainings der Maschine ausgewählte Funktion bestimmt wird. Für die strukturelle Risikominimierung wird die Klasse der in Frage kommenden Funktionen in verschachtelte Untermengen geteilt (siehe Abbildung 5.2).

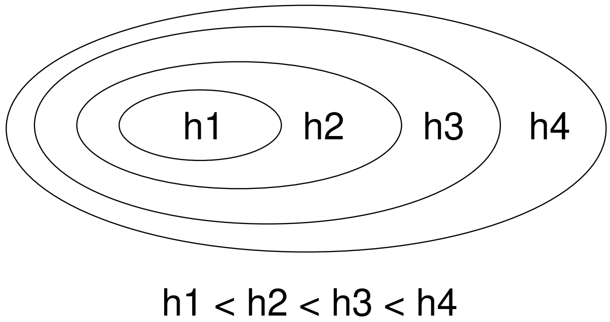


Abbildung 5.2: Verschachtelung der Funktionsklassen in Bezug zur VC Dimension

Für jede dieser Teilmengen muss die VC Dimension h oder eine obere Schranke für h ermittelt werden. Die Methodik der SRM besteht darin, die Teilmenge der Funktionen zu ermitteln, welche die obere Grenze für das erwartete Risiko minimiert. Dies kann beispielsweise durch mehrere Trainingsläufe (für jede Teilmenge einen) mit dem Ziel, das empirische Risiko zu minimieren, und anschließende Auswahl der Maschine mit der geringsten Summe von empirischem Risiko und VC Konfidenz erreicht werden.

Dieses Verfahren bildet die Grundlage für die Vorgehensweise der Support Vector Machine, die konkrete Ausprägung wird in den nächsten Abschnitten besprochen.

5.3 Support Vektoren

An dieser Stelle beginnen die Erläuterungen zur Support Vector Machine mit dem einfachen Fall der linearen Maschine, trainiert auf eindeutig voneinander trennbaren Daten. Die Trainingsdaten werden mit $\{\vec{x}_i, y_i\}$, $i = 1, \ldots, l$, $y_i \in \{-1, 1\}$ und $\vec{x}_i \in \mathbb{R}^d$ bezeichnet. Gesucht ist die Hyperebene, die beide Klassen innerhalb der Trainingsdaten voneinander abgrenzt. Für Punkte $\vec{x}$, die auf dieser Hyperebene liegen, gilt $\vec{n} \cdot \vec{x} + b = 0$, wobei $\vec{n}$ die Normale der Ebene und b die Verschiebung vom Ursprung darstellt. d_+ (d_-) bezeichnet den kleinsten Abstand der Ebene zu dem nächstliegenden positiven (negativen) Beispiel einer Klasse. Ziel ist es, diesen Abstand bei korrekter Trennung der Beispiele zu maximieren. Damit muss folgende Bedingung erfüllt sein:

$$y_i(\vec{x}_i \cdot \vec{n} + b) - 1 \geq 0 \quad \forall i$$

bei gleichzeitigem Minimieren von $||\vec{n}||$.

Diese Bedingungen sind für die Korrektheit zur Trennung der Daten in ihre Klassen maßgeblich, gleichzeitig allerdings auch für die Güte der Generalisierung bestimmend. Eine Maschine besitzt dann eine gute Generalisierung, wenn sie auch bei leichten Verschiebungen der Daten noch richtig klassifizieren kann. Dazu muss der Abstand der trennenden Hyperebene zu den Beispielen beider Klassen maximiert werden, um eine gewisse „Sicherheitszone“ zwischen den Beispielen zu schaffen. Abbildung 5.3 stellt Beispiele für die Fälle beider Klassen (Punkte), die trennenden Ebenen (H_1 und H_2), die den Rand der Daten begrenzen, und die optimal trennende Ebene (H) dazwischen dar.

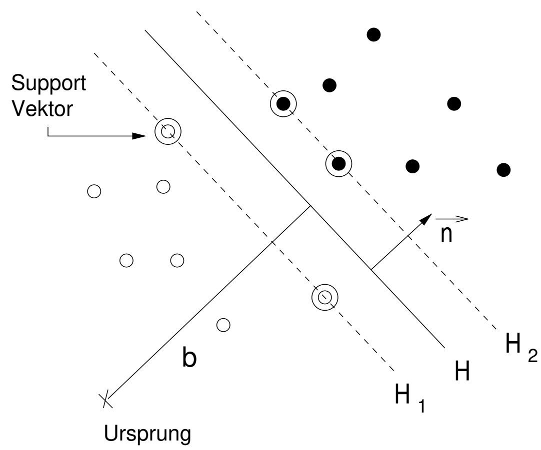


Abbildung 5.3: Durch eine Gerade getrennte Datenpunkte

Bezeichnend für die Support Vector Machine ist, dass nicht alle Trainingsdaten für die Berechnung der optimal trennenden Ebene notwendig sind. Diejenigen Vektoren, welche die oben angegebenen Bedingungen erfüllen und deren Wegfall die Lösung verändern, werden als *Support Vektoren* bezeichnet.

Das Prinzip der strukturellen Risikominimierung wird dadurch wieder aufgegriffen, dass das Maximieren des Abstands der durch die Support Vektoren laufenden Ebenen dem Minimieren von $||\vec{n}||$ und laut Vapnik ([Vapnik98]) damit dem Minimieren der VC Dimension entspricht.

Die Bedingungen zur Maximierung des Randes lassen sich mit Hilfe von Lagrange-Multiplikatoren $\alpha_i, i = 1 \dots l$ in die Lagrange-Darstellung überführen:

$$L_P = \frac{1}{2}||\vec{n}||^2 - \sum_{i=1}^{l} \alpha_i y_i (\vec{x}_i \cdot \vec{n} + b) + \sum_{i=1}^{l} \alpha_i$$

unter den Bedingungen

$$\vec{n} = \sum_i \alpha_i y_i \vec{x}_i$$

und

$$\sum_i \alpha_i y_i = 0.$$

Eingesetzt ergibt sich:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j.$$

Zur Lösung des Problems muss entweder L_P minimiert oder L_D maximiert werden; in Abhängigkeit von α_i. Für jeden Vektor aus der Trainingsmenge existiert ein eigener Lagrange-Faktor, für Support Vektoren gilt $\alpha_i > 0$, für alle übrigen $\alpha_i = 0$.

Damit hat sich ein quadratisches Programmierungsproblem ergeben, das entsprechend gelöst werden muss. Zur Lösung des Problems können die sogenannten *Karush-Kuhn-Tucker Bedingungen (KKT)* herangezogen werden, die für die Lösungen von $\vec{n}, b$ und α sowohl hinreichend als auch notwendig sind:

$$\begin{aligned}
\frac{\partial}{\partial n_\nu} L_P = n_\nu - \sum_i \alpha_i y_i x_{i,\nu} &= 0 \quad \nu = 1 \dots d \\
\frac{\partial}{\partial b} L_P = -\sum_i \alpha_i y_i &= 0 \\
y_i(\vec{x}_i \cdot \vec{n} + b) - 1 &\geq 0 \quad i = 1 \dots l \\
\alpha_i &\geq 0 \quad \forall i \\
\alpha_i(y_i(\vec{n} \cdot \vec{x}_i + b) - 1) &= 0 \quad \forall i.
\end{aligned}$$

ν läuft dabei von 1 bis zur Dimension der Daten.

Diese Bedingungen lassen sich z.B. benutzen, um die Verschiebung b der Hyperebenen vom Ursprung zu bestimmen. Dazu braucht lediglich in der letzten Bedingung für ein $\alpha_i \neq 0$ das durch Training bestimmte $\vec{n}$ eingesetzt und nach b aufgelöst zu werden.

Für die Ausführungen in den vorhergehenden Abschnitten wurde eine Bedingung vorausgesetzt, die ebenfalls in Abbildung 5.3 galt: Die Daten mussten linear voneinander trennbar sein. In der Realität wird diese Voraussetzung nur in den wenigsten Fällen erfüllbar sein, die Daten werden durch Inkonsistenzen (Rauschen, Messfehler...) in einzelnen Fällen im Bereich der anderen Klasse liegen. Mit der bisherigen Methode ist eine Lösung des Problems nicht möglich, weswegen ein Maß $\xi_i, i = 1 \dots l$ mit $\xi_i \geq 0$ für den Fehler jedes Trainingsbeispiels eingeführt wird. Die Bedingung für die Bestimmung der Hyperebene erweitert sich damit zu:

$$y_i(\vec{x}_i \cdot \vec{n} + b) \geq 1 - \xi_i.$$

Statt jetzt lediglich $||\vec{n}||^2/2$ zu minimieren, ist es sinnvoll, eine Funktion f in Abhängigkeit der Fehler hinzuzunehmen, die eine gewisse Form von „Kosten" für Klassifikationsfehler einführt: $||\vec{n}||^2/2 + C \cdot f(\sum_i \xi_i)$. C, die Kapazität der Maschine, ist dabei ein durch den Anwender festgelegter Faktor, der die Gewichtung der Fehler im Verhältnis zur Minimierung von $||\vec{n}||$ festlegt. Ein höheres C führt somit zu einer stärkeren Bestrafung von Fehlern.

Für die Wahl von $f \to x^k$ mit $k = 1$ und $k = 2$ ergibt sich wieder ein einfaches quadratisches Optimierungsproblem mit der folgenden Lagrange-Darstellung:

$$L_P = \frac{1}{2}||\vec{n}||^2 + C\sum_i \xi_i - \sum_i \alpha_i(y_i(\vec{x}_i \cdot \vec{n} + b) - 1 + \xi_i) - \sum_i \mu_i \xi_i.$$

Zusätzlich müssen wieder die KKT-Bedingungen gelten, mit folgenden Erweiterungen:

$$\begin{aligned}
\frac{\partial}{\partial \xi_i} L_P = C - \alpha_i - \mu_i &= 0 \\
y_i(\vec{x}_i \cdot \vec{n} + b) - 1 + \xi_i &\geq 0 \\
\xi_i &\geq 0 \\
\mu_i &\geq 0 \\
\alpha_i(y_i(\vec{x}_i \cdot \vec{n} + b) - 1 + \xi_i) &= 0 \\
\mu_i \xi_i &= 0.
\end{aligned}$$

Die Entscheidung, ob ein unbekannter Fall $\vec{x}$ zur Klasse -1 oder 1 gehört, wird mittels einer trainierten SVM bestimmt, indem die Lage von $\vec{x}$ in Bezug zur trennenden Hyperebene ermittelt wird. Dies erfolgt durch die Berechnung von $f(\vec{x}) = \mathrm{sign}(\vec{n} \cdot \vec{x} + b)$.

5.4 Kostenfunktionen zur Regression

Bislang wurde die Support Vector Machine nur für den Fall der binären Klassifikation betrachtet. Das Problem der Lastprognose verlangt dagegen die Vorhersage kontinuierlicher Werte oder zumindest die Unterscheidung in eine hinreichend große Anzahl von Klassen.

Die bekannte Variante der SVM lässt sich allerdings erweitern, so dass auch reelle Werte prognostiziert werden können. Grundprinzip ist, eine Hypergerade zu ermitteln, die die Funktionswerte approximiert. Dies entspricht dem Finden einer Geraden, die den Abstand zu allen Trainingsbeispielen minimiert. Um dieses Ziel zu erreichen wird eine Kostenfunktion, ähnlich der Berücksichtigung von Fehlklassifikationen aus dem vorhergehenden Abschnitt, eingeführt. Der Verlust über diese Kostenfunktion soll entsprechend minimiert werden. Typische Beispiele sind lineare, wie hier angegeben, oder quadratische Verlustfunktionen:

$$L(y, f(\vec{x}, \alpha)) = \begin{cases} 0 & \text{bei } |y - f(\vec{x}, \alpha)| \leq \varepsilon \\ |y - f(\vec{x}, \alpha)| - \varepsilon & \text{sonst} \end{cases}.$$

Der Parameter ε bezeichnet die *Insensitivität* der Kostenfunktion, d.h. für Abweichungen innerhalb dieses Bereichs werden keine Kosten veranschlagt. Dies dient dazu, einen gewissen Bereich hinsichtlich Genauigkeit und Schwankungen innerhalb der Daten auszuklammern.

Auf ähnliche Weise lassen sich auch asymmetrische Kostenfunktionen implementieren, die es ermöglichen, Über- und Unterschätzen auf unterschiedliche Arten, bzw. Gewichtungen, zu berücksichtigen.

5.5 Kernelfunktionen

Ein weiterer Nachteil der bislang vorgestellten SVM besteht im Versuch, die Daten grundsätzlich linear voneinander zu trennen. Nur in wenigen einfachen

Fällen wird diese Methode akzeptable Vorhersagen ermöglichen. Durch *Kernelfunktionen* werden deshalb die Fähigkeiten der SVM so erweitert, dass Funktionen anderer Kategorien gelernt werden können.

Zunächst sei vorausgesetzt, dass die Ursprungsdaten durch eine Transformation $\Phi : \mathbb{R}^d \to \mathbb{H}$ in einen anderen Datenraum (u.U. mit der Dimension ∞) überführt werden. Innerhalb des neuen Datenraums kann dann wieder mit der herkömmlichen Methode eine linear trennende Funktion bestimmt werden; die real gelernte Funktion ist damit nicht linear, kann aber mit vergleichbarem Aufwand wie im rein linearen Fall gefunden werden.

Da in der Lagrange-Formulierung L_D des Problems die Daten nur in Produktform $\vec{x}_i \cdot \vec{x}_j$ auftreten, muss die Transformation Φ nicht direkt bekannt sein, es genügt eine Kernelfunktion $K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$ zu kennen.

Die Ergebnisse für einen Testpunkt $\vec{x}$ ergeben sich dann nach:

$$f(\vec{x}) = \text{sign}(\sum_{i=1}^{N_s} \alpha_i y_i K(\vec{s}_i, \vec{x}) + b),$$

wobei s_i die Support Vektoren, N_s die Anzahl der Support Vektoren repräsentieren.

Nicht jede Funktion K lässt sich als Kernelfunktion verwenden. Für jeden Kernel muss die *Mercer-Bedingung* erfüllt sein: Es existiert ein K und ein Φ mit

$$K(\vec{x}, \vec{y}) = \sum_i \Phi(\vec{x})_i \cdot \Phi(\vec{y})_i$$

genau dann, wenn für jedes $g(\vec{x})$ mit

$$\int g(\vec{x})^2 d\vec{x} \quad \text{ist finit},$$

gilt:

$$\iint K(\vec{x}, \vec{y}) g(\vec{x}) g(\vec{y}) d\vec{x} d\vec{y} \geq 0.$$

Die Version der SVM, die für die nachstehenden Experimente benutzt wurde, bietet neben der Möglichkeit eigene Kernel mittels Vorwissen über die Art und Verteilung der Daten zu erstellen, standardmäßig folgende Kernel:

1. Dot: $K(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y}$.

2. Polynomiell: $K(\vec{x}, \vec{y}) = ((\vec{x} \cdot \vec{y}) + 1)^d$ mit Grad $d \in \mathbb{N}$.

3. Radial: $K(\vec{x}, \vec{y}) = e^{-\gamma |\vec{x} - \vec{y}|^2}, \quad \gamma \in \mathbb{R}_0^+$.

4. Zweischichtiges neuronales Netz: $K(\vec{x}, \vec{y}) = \tanh(a(\vec{x} \cdot \vec{y}) + b), \quad a, b \in \mathbb{R}$.

5. Anova: $K(\vec{x}, \vec{y}) = (\sum_i e^{-\gamma(\vec{x}_i - \vec{y}_i)})^d, \quad \gamma \in \mathbb{R}_0^+, d \in \mathbb{N}$.

5.6 Ergebnisse

Nach den Ausführungen zur Support Vector Machine und der zugrunde liegenden Theorie folgt in diesem Abschnitt die Darstellung der praktischen Ergebnisse im Einsatz der SVM. Da es sich im Gegensatz zu C4.5 um ein regressionsfähiges Verfahren handelt, fallen einige der Experimente aus dem vorhergehenden Kapitel weg, andere kommen durch die unterschiedlichen Parameter (Kernel, Kapazität...) neu hinzu. Zusätzlich fehlt die Wiederholung einiger Experimente mit C4.5, die für den späteren Einsatz lediglich von theoretischem Interesse hinsichtlich des Verhaltens des Data Mining-Verfahrens waren, wie z.B. die getrennte Betrachtung von drei Routen. Da lediglich das gesamte Angebot auf einer Verbindung interessiert und die Verteilung auf die alternativen Wege durch den Managementteil des Systems erfolgt, bringt die einzelne Berücksichtigung der Routen für die Prognose keinen Vorteil. Die zugrundeliegende Intention bei allen durchgeführten Experimenten ist aber auch hier, die Abweichung vom realen Wert zu minimieren.

Zum Einsatz gekommen ist die *mySVM* in der Version 1.3.6 vom Lehrstuhl für Künstliche Intelligenz der Universität Dortmund. Diese Support Vector Maschine implementiert Regression und asymmetrische Kostenfunktionen. Bezogen werden kann sie unter [mySVM]. Da als Plattform für den Einsatz Unix vorgesehen ist, empfiehlt es sich beim Betrieb unter Windows eine entsprechende gcc-Variante einzusetzen, die das problemlose Ausführen des Programms ermöglicht ([Cygwin]). Da die SVM stark durch Attribute mit sehr unterschiedlichen Wertebereichen beeinflusst wird (Zeitindex von 600 bis 86400, Angebote z.B. von 0 bis 1000 Erlang), werden vor dem Anstoß des Lernprozesses alle Attribute linear auf den Bereich 0 bis 1 skaliert.

Interessant ist die Möglichkeit von mySVM, asymmetrische Kostenfunktionen zu berücksichtigen. Damit lassen sich Unter- oder Überschätzen unterschiedlich starke Gewichtungen zuordnen, was die Vorhersage in die eine oder andere Richtung verschiebt. Konkret für den Netzbetrieb bedeutet dies, dass sich der Anbieter entscheiden kann, ob er eher ein überdimensioniertes Netz bevorzugt, bei dem das Überschätzen des Maximums der nächsten Stunden weniger stark bestraft wird, oder ein unterdimensioniertes, dass dann eine bessere prozentuale Auslastung der vorhandenen Kapazitäten aufweist. In den meisten Fällen wird die erste Variante bevorzugt, da Kundenzufriedenheit und damit eine hohe Wahrscheinlichkeit für eine verfügbare Leitung ein starkes Kriterium für den Betrieb eines Netzwerks ist. Letztendlich entscheiden über diesen Punkt im Detail betriebswirtschaftliche Kriterien. Entsprechend zur Festlegung der Rekonfigurationszahl für die Bewertung von zu hohen oder zu niedrigen Blockierungen wurde das Loss für das Unterschätzen des Angebots um den Faktor drei höher angesetzt als für eine überschätzte Prognose.

Für den insensitiven Bereich ε hinsichtlich des Loss wurde 1% des maximalen Angebots innerhalb des betrachteten Zeitraums gewählt.

Prinzipiell lassen sich auch viele andere Verfahren des maschinellen Lernens um die Berücksichtigung von Kostenfunktionen erweitern, indem spezielle Vorverarbeitungsschritte — wie in [Domingos99] vorgeschlagen — eingesetzt werden. Großer Nachteil dieser aufgesetzten Lösung ist allerdings, dass die

Umwandlung der Trainingsdaten in ihre alternative Repräsentation wieder entsprechend Zeit benötigt und damit nur bedingt in Frage kommt.

Die Transformation der Daten in das Eingabeformat der SVM erfolgt ähnlich wie bei C4.5 und führt zu, im vorhergehenden Kapitel beschriebenen, äquivalenten Datenrepräsentationen (siehe Kapitel 4). Beispiele für die Formate zur Eingabe in die Support Vector Machine werden deshalb hier nicht nochmals gegeben, stattdessen sollte bei Bedarf die entsprechende Dokumentation zu Rate gezogen werden.

5.6.1 Unterschiedliche Kapazitäten

Die Kapazität der Support Vector Machine bestimmt, wie gut sich das Verfahren an die Trainingsdaten anpasst, wodurch dieser Parameter mitbestimmend für die spätere Aussagequalität ist. Die direkte Bestimmung der Kapazität (oder auch eines guten Kernels, siehe Abschnitt 5.6.3) ist aktueller Forschungsgegenstand. In diesem Anwendungsfall wurden Kapazitäten für einen Bereich von 10^{-3} bis 10^{3} mit logarithmischer Skalierung der Schrittweite untersucht. Tabelle 5.1 gibt die Daten beispielhaft für Dienst 1, eine Historie von 20 Takten und den Dot-Kernel wieder.

Kapazität	Average Loss Training	Average Loss Test	Iterationen
Ohne Zeitindex			
0,001	224,3	223,5	95
0,01	230,0	228,7	83
0,1	504,0	505,0	27
1	176,7	177,7	1143
10	1012,6	1018,9	709
100	1092,6	1098,7	2441
1000	927,5	933,1	10000
Mit Zeitindex			
0,001	204,2	202,8	287
0,01	129,8	131,5	497
0,1	113,6	114,1	747
1	114,8	120,3	10000
10	1083,7	1086,8	663
100	403,7	312,3	10000
1000	312,3	307,8	10000

Tabelle 5.1: Unterschiedliche Kapazitäten der SVM

Zu beachten ist z.B. der Wert von 10.000 Iterationen[1] für eine Kapazität von 1.000. Dieser Wert kommt als Grenzwert für die maximale Anzahl an zulässigen Iterationen zustande. Ab einer Größe von 10.000 Iterationen ist davon

[1]Die Anzahl der Berechnungsschritte des numerischen Lösungsverfahrens, mit dem die Parameter der SVM bestimmt werden.

auszugehen, dass das Kosten–Nutzen Verhältnis nicht mehr in einer vernünftigen Relation steht und trotz eventueller Genauigkeitsgewinne die aufgewandte Rechenzeit nicht mehr rechtfertigt. Zusätzlich besteht die Möglichkeit, dass für diese Parameter die Maschine nicht konvergiert und so bei einer höheren Iterationstiefe schlechtere Ergebnisse erzielt.

Die Versuche zeigen, dass die Wahl der Kapazität relativ kritisch für Rechenzeit und Genauigkeit ist. Prinzipiell weisen nicht alle Kernel dieses Verhalten auf, der polynomielle Kernel beispielsweise ist für einen Grad größer oder gleich 3 stabil in seiner Vorhersagegenauigkeit, liegt aber mit einem durchschnittlichen Loss von 375 auch deutlich höher als die mit dem Dot-Kernel erzielbaren Ergebnisse.

Ähnlich wie bei C4.5 lassen sich mit Berücksichtigung des Zeitindex bessere Werte erzielen.

Für andere Dienste zeigen die Kapazitäten vergleichbare Auswirkungen, allerdings ist es schwierig vorherzusagen, bei welchen Kapazitäten die Grenzwerte für die maximale zulässige Anzahl an Iterationen erreicht werden. Damit lässt sich auch der benötigte Zeitbedarf nur schwer im Vorfeld abschätzen und ist bereits bei oberflächlich erscheinend kleinen Änderungen Schwankungen unterlegen. Für einzelne Dienste, bei denen andere Kernel als der Dot-Kernel bessere Ergebnisse liefern, gelten leicht andere Ergebnisse; dies wird aber speziell auf die Kernel bezogen noch in Abschnitt 5.6.3 dargestellt.

5.6.2 Verschiedene Vergangenheitsbereiche

Eine große Rolle in Bezug auf die Qualität der Prognose hat bei C4.5 die Auswahl der Vergangenheitswerte gezeigt. Insbesondere war der Effekt zu beobachten, dass bis zu einer gewissen Länge des betrachteten Vergangenheitsbereichs die Genauigkeit verbessert werden konnte, diese ab einem bestimmten Grenzwert dagegen wieder nachließ. In diesem Abschnitt wird versucht, diesen oder ähnliche Effekte über die Länge der Historie zu untersuchen.

Historie	Average Loss Training	Average Loss Test	Iterationen
Ohne Zeitindex			
10	203,0	202,0	114
15	218,6	217,6	90
20	230,0	228,0	83
25	229,6	228,6	73
Mit Zeitindex			
10	150,3	152,0	608
15	135,8	139,0	1796
20	129,8	131,5	497
25	122,4	129,1	802

Tabelle 5.2: Unterschiedlich große Historie

Tabelle 5.2 zeigt die Abhängigkeiten für Dienst 1, Dot-Kernel und eine Kapazität von 0,01, die sich sowohl im Bereich ohne/mit Zeitindex als auch bei der Ausführungszeit in einem guten Bereich bewegt hat (siehe Abschnitt 5.6.1).

Bei Betrachtung der Ergebnisse in Tabelle 5.2 fällt auf, dass sich der bei C4.5 beobachtete Effekt nicht, bzw. nur bedingt wiederfinden lässt. In der Repräsentation ohne Zeitindex findet die SVM das minimale Loss bereits bei der kleinsten Historie. Für die anderen Bereiche zeigt sich entweder eine Verschlechterung oder keine Änderung. Anders sieht es bei der Repräsentation mit Zeitindex aus, hier kehrt sich der Effekt um. Die ersteren Ergebnisse (ohne explizite Berücksichtigung der Zeit) zeigen sich auch bei anderen Kapazitäten, wobei sich das minimale Loss nicht immer direkt an der Stelle der kleinsten Historie befindet. Für die in der Tabelle gegebenen Ergebnisse mit Zeitindex sind sie eher die Ausnahme, Fälle wie bei der Variante ohne Zeitindex überwiegen.

Andere Kernel zeigen sich gegenüber der Länge der Historie als äußerst unempfindlich und ändern sich tendenziell zwar ähnlich wie der Dot-Kernel, insgesamt aber auf die Größe der Abweichung bezogen in weit geringerem Ausmaß.

Bei Betrachtung anderer Dienste bietet sich ein anderes Bild, beispielhaft für Dienst 2 (Musikübertragung/Audio-Streaming) in Tabelle 5.3 wiedergegeben.

Historie	Average Loss Training	Average Loss Test	Iterationen
Ohne Zeitindex			
10	85,4	85,6	72
15	74,2	74,4	58
20	46,9	47,6	155
25	49,8	51,2	168
Mit Zeitindex			
10	58,0	67,3	108
15	43,2	57,9	218
20	45,2	43,8	154
25	52,4	51,6	79

Tabelle 5.3: Unterschiedlich große Historie für Dienst 2

Hier zeigt sich ein Anstieg der Fehlerrate sowohl ohne als auch mit Berücksichtigung des Zeitpunkts, bei den längeren Historien sind die Werte für beide Varianten vergleichbar.

Bei einigen Versuchen zeigen sich recht deutliche Unterschiede zwischen Trainings- und Testdaten. Solche Parameter sind somit Kandidaten für eine aufwändigere Kreuzvalidierung um eine sicherere Aussage für den erwarteten Fehler treffen zu können.

Andeutungsweise zeigt sich hier auch, dass die Dienste nur schwer global gleich behandelt werden können. Mit denselben Parametern ergeben sich für einen anderen Dienst abweichende Muster. Für die spätere Integration in Median erscheint es deshalb sinnvoll, jeden Dienst getrennt von den übrigen zu betrachten und einzeln die besten Parameter zu verwenden. Da sich die qualitativen Verläufe der Dienste auf unterschiedlichen Links stark ähneln, ist es nicht erforderlich, für jeden Link einen getrennten Parametersatz zu wählen.

5.6.3 Vergleich der Kernel

Die Wahl der Kernelfunktion bestimmt die Transformation der Daten zur Anwendung der Maschine im linearen Raum. Sie legen damit die Klasse der Funktionen fest, für die entsprechende Parameter durch Training ermittelt werden. Dieser Abschnitt greift die Versuche mit unterschiedlichen Kernelfunktionen auf und untersucht die Auswirkungen der zu den Funktionen gehörenden Parameter. Benutzt wurden die fünf durch die SVM bereitgestellten Standardkernel (dot, polynomial, radial, neural, anova), bei Kernelfunktionen mit Parametern kamen jeweils die Wertebereiche $1 \dots 5$ zum Einsatz.

Tabelle 5.4 stellt im Vergleich zu dem in den bisherigen Tabellen benutzten Dot-Kernel den Anova-Kernel mit den zugehörigen Parametern dar bei einer Kapazität von 0,01. Es werden nicht alle getesteten Parameterkombinationen angegeben, sondern lediglich ein Auszug.

Historie	γ	d	Av. Loss Training	Av. Loss Test	Iterationen
Ohne Zeitindex					
10	1	1	266,1	263,9	352
10	1	2	263,9	261,6	103
10	1	3	263,8	261,5	369
10	1	4	260,4	258,4	163
10	1	5	262,9	260,7	123
15	1	1	259,0	257,4	142
20	1	1	259,0	257,5	147
25	1	1	263,9	261,7	90
Mit Zeitindex					
10	1	1	266,1	263,9	352
10	1	2	266,1	264,0	446
10	1	3	260,5	258,5	404
10	1	4	263,7	261,5	427
10	1	5	260,3	258,4	152
15	1	1	263,8	261,8	114
20	1	1	263,8	261,9	115
25	1	1	263,9	261,7	85

Tabelle 5.4: Fehlerraten mit dem Anova-Kernel

Das durchschnittliche Loss zeigt sich relativ unempfindlich gegenüber der Auswahl der Parameter, Änderungen treten nur in geringem Maße und keinesfalls im Rahmen ganzer Größenordnungen auf. Bei der Betrachtung anderer Kernel wird dies noch deutlicher, der polynomielle Kernel beispielsweise erzielt nur für einen Grad von 2 sehr schlechte Werte, die ab dem Grad 3 deutlich besser wurden. Die Änderungen des Fehlers für Grade höher als 3 sind dagegen minimal.

In Bezug zur Konvergenz des Verfahrens hat es mit diesem Kernel keine Probleme gegeben, in keinem Fall ist die Grenze von 10.000 Iteration erreicht worden. Teilweise weisen die Berechnungen mit diesem Kernel eine deutlich geringere Iterationszahl als diejenigen mit dem Dot-Kernel auf.

Eine Berücksichtigung des Zeitindex unter den Trainingsdaten führt zu keiner Verbesserung des Ergebnisses.

Die gegebene Tabelle dient als Beispiel für das Verhalten eines anderen Kernels; im Vergleich zum Dot-Kernel ergibt der Anova-Kernel in diesem Fall schlechtere Ergebnisse. An dieser Stelle soll allerdings angemerkt werden, dass dies nicht für jeden Dienst gilt, sondern sich auch bessere Ergebnisse als mit dem Dot-Kernel erzielen lassen. Dieser Punkt zeigt sich nochmals in der Zusammenfassung dieses Kapitels.

5.6.4 Logarithmische Attributauswahl

Die logarithmisch gestaffelte Auswahl der Attribute diente dazu, weniger Angebotsdaten zu berücksichtigen, je weiter sie in der Vergangenheit vom aktuellen Zeitpunkt aus gesehen liegen. Tabelle 5.5 stellt einige der besten Ergebnisse vor, die mit diesem Ansatz erzielt werden konnten, um die Vergleichbarkeit zur Leistungsfähigkeit der linearen Auswahl der Angebotswerte herzustellen.

Dienst	Log	Zeit	Kernel	Kapazität	Historie	Loss
1	2	J	1	1	20	110,3
1	e	N	1	10	15	32,2
1	10	J	1	0,1	25	102,4
2	2	J	1	10	25	15,3
2	e	N	1	1000	15	21,6
2	10	N	1	100	15	22,3
3	2	J	1	0,1	20	23,4
3	e	J	1	0,01	10	27,6
3	10	J	1	0,1	25	24,1
4	2	N	1	1000	25	24,2
4	e	N	1	10	25	26,0
4	10	N	1	1000	15	25,6
5	2	J	1	0,01	15	3,2
5	e	N	1	1000	10	3,4
5	10	J	1	0,001	25	3,3

Tabelle 5.5: Die besten Parameter für jeden Dienst bei logarithmischer Auswahl

Herausragende Ergebnisse wie beispielsweise für Dienst 1 bei Auswahl der Attribute durch den natürlichen Logarithmus bedürfen genauerer Prüfung hinsichtlich Repräsentativität und Stabilität auf anderen Daten. Dienst 6 erscheint nicht in der Tabelle, da die besten Werte für diesen Dienst entweder nur bei 10.000 Iterationsschritten auftraten oder ein Unterschied zwischen Test- und Trainingsfehler von mehreren Größenordnungen bestand. Dieser Punkt sollte gesondert ebenfalls genauer untersucht werden.

5.6.5 Berücksichtigung der Steigung

In diesem Abschnitt wird die Auswirkung auf das Loss untersucht, wenn die Steigung zwischen den Angebotswerten zu Training und Vorhersage mit herangezogen wird (Tab. 5.6).

Dienst	Zeit	Kernel	Kapazität	Historie	Loss
1	J	1	0,1	20	113,4
2	N	1	10	25	19,3
3	J	1	0,1	15	31,4
4	J	1	0,01	20	23,0
5	N	1	0,001	25	3,3

Tabelle 5.6: Die besten Parameter für jeden Dienst bei Berücksichtigung der Steigung

Wie bei der logarithmischen Attributauswahl bereitet die Validierung des tatsächlichen Loss für Dienst 6 Probleme, so dass dieser Fall gesondert untersucht werden sollte.

5.6.6 Kreuzvalidierung

Die Kreuzvalidierung stellt an dieser Stelle ebenfalls sicher, dass die Datenauswahl den Fehler repräsentativ widerspiegelt. mySVM bietet die Möglichkeit, direkt eine Kreuzvalidierung über einer gegebenen Anzahl von Blöcken (*chunks*) durchzuführen. Die Versuche wurden entsprechend zu C4.5 mit einer 10-fach Kreuzvalidierung durchgeführt. Betrachtet wurden die Einstellungen, die das niedrigste durchschnittliche Loss, eine zu große Diskrepanz zwischen Trainings- und Testfehler aufwiesen oder an die Grenze von 10.000 Iterationen gestoßen sind.

Dienst	Zeit	Kernel	Kapazität	Historie	Loss Training	Loss Test
1	J	1	0,1	20	116,2	128,9
1	J	1	1	20	123,8	123,0
1	J	1	100	20	241,1	235,8
6	N	2	10	10	0,14	0,15
Steigung						
6	J	1	100	10	0,1	193,8
6	J	1	100	25	0,4	153,4
Logarithmus zur Basis 2						
6	J	1	100	20	0,2	11,9
6	J	1	100	30	1,8	104,7

Tabelle 5.7: Kreuzvalidierung

In vielen Fällen ergibt die Kreuzvalidierung eine Abweichung zu einem höheren Loss, selten allerdings um ganze Größenordnungen. Die Rangfolge der

Parameter ändert sich nur bei wenigen Beispielen. Interessante Ergebnisse zeigen sich für die Parameter, welche die Grenze von 10.000 Iterationen erreichen (Tabelle 5.7, Dienst 1, Kapazitäten 1 und 100). Diese konnten das Ergebnis aus Tabelle 5.1 bestätigen, bzw. verbessern, obwohl der Iterationsvorgang der SVM vor Erreichen der Konvergenzkriterien abgebrochen wurde.

Als problematisch erweist sich Dienst 6. Die Werte verschlechtern sich bei Durchführung der Kreuzvalidierung nicht nur (vgl. Tabelle 5.8), sondern weisen in Teilen eine so große Differenz zwischen Trainings- und Testfehler auf, dass eine zuverlässige Aussage über den zu erwartenden Fehler nicht getroffen werden kann.

5.6.7 Zusammenfassung

Die obigen Abschnitte haben einzelne der besonderen Muster in den berechneten Abweichungen wiedergegeben. Letztendlich zählen allerdings die besten Ergebnisse hinsichtlich Genauigkeit und Zeitaufwand. Hier werden die besten Parameterkombinationen gestaffelt nach den Diensten nochmals hervorgehoben. Ausgespart sind Trainingsläufe mit 10.000 Iterationen, da keine Aussage über die weitere Entwicklung des Loss bei unbeschränkter Iterationszahl getätigt werden kann, und die besonderen Transformationen mit logarithmischer Attributauswahl oder Berücksichtigung der Steigung (s. die Ergebnisse dort).

Dienst	Zeit	Kernel	Kapazität	Historie	Loss
1	J	1	0,1	10	110
2	N	1	10	25	20,4
3	J	1	1	15	16
4	J/N	5	10–1000	15–25	13–15
5	N	5	10	30	3
6	N	2	10	10	0,09

Tabelle 5.8: Die besten Parameter für jeden Dienst

Es zeigt sich, dass eine globale Parametereinstellung nur bedingt gewählt werden kann. Die besten Parameter unterscheiden sich zwischen den Diensten, es ist daher empfehlenswert, für jeden Dienst getrennte Einstellungen vorzunehmen. Innerhalb einer Dienstkategorie, aber auf unterschiedlichen Links treten diese Probleme nur bedingt auf, da die Dienste auch auf unterschiedlichen Verbindungen einen qualitativ ähnlichen Verlauf besitzen.

Meist gibt es mehrere Parametereinstellungen, die ein ähnlich großes Loss verursachen. In diesem Fall sind Einstellungen zu bevorzugen, die eine deutlich kürzere Trainingszeit besitzen.

Bei der Berücksichtigung des Zeitindex ergeben sich unterschiedliche Fälle: Manche Dienste erzielen mit Zeit deutlich bessere Ergebnisse, andere ohne und teilweise hat die Berücksichtigung nur geringe Auswirkungen. Dienst 4 nimmt eine Sonderrolle ein, hier ist die Wahl des Kernels maßgeblich, die anderen Parameter spielen nur eine untergeordnete Rolle.

Insgesamt hat sich gezeigt, dass die Support Vector Machine in der Wahl der Parameter deutlich schwieriger zu handhaben ist. Zeigt C4.5 für viele Bereiche vergleichbares Verhalten in Bezug auf die Fehlerrate und den Zeitbedarf, sind die Einstellungen für die SVM deutlich sensibler vorzunehmen. Andererseits bietet die SVM gerade durch die weitreichenden Parametereinstellungen eine höhere Flexibilität und Anpassungsfähigkeit, so dass mit dem entsprechenden Aufwand an Voruntersuchungen und späterer Pflege in der Regel gute Ergebnisse erzielbar sind. Wie sich die bislang vorgestellten Verfahren auf die Abweichung der Prognose während eines vollständigen Simulationslaufs verhalten ist Thematik des nächsten Kapitels, in dem die Verfahren im direkten Vergleich betrachtet werden.

Kapitel 6

Vergleich der Verfahren

Nachdem die Aufgabe der vorhergehenden Kapitel war, jedes Data Mining-Verfahren für sich getrennt zu betrachten und den Klassifikationsfehler/das Loss durch Testen verschiedener Datenrepräsentationen und Parameter zu minimieren, werden in diesem Kapitel die Verfahren im direkten Vergleich getestet. Im Vordergrund steht hier dementsprechend auch nicht mehr die dem jeweiligen Verfahren zugrundeliegende Bewertungsmetrik (Risikomessung...), sondern die Abweichung des vorhergesagten Wertes vom realen Angebot auf den einzelnen Verbindungen.

In Abschnitt 6.1 werden dazu Berechnungen für jedes Ausgabeintervall der Simulation durchgeführt und somit eine Fehlerkurve über das Netzverhalten des gesamten Tagesdurchlaufs erstellt. Die übrigen Abschnitte beschäftigen sich mit Veränderungen der Ausgangsdaten (z.B. durchschnittlich steigende Angebote, die Trends im wachsenden Telekommunikationsmarkt widerspiegeln), um die Stabilität der Prognosemethoden hinsichtlich leichteren oder stärkeren Schwankungen der Voraussetzungen zu untersuchen. Stabile Verfahren gleichen kleine Veränderungen in den Attributwerten aus, ohne gravierende Verschlechterungen in den Prognosen aufzuweisen.

6.1 Tagesprognosen

Da die Verfahren C4.5 und SVM völlig unterschiedliche Methoden benutzen und die Ergebnisse bei ersterem diskret, bei letzterem kontinuierlich sind, lassen sich die Vorhersagen nicht direkt vergleichen. Hier wird für jeden Ausgabezeitpunkt (alle 600 s) eine Prognose für das Maximum der nächsten drei Stunden erstellt und mit dem realen Wert, der innerhalb des Horizonts erreicht wurde, für eine neue, den Verfahren unbekannte, Tageskurve verglichen.

Die von C4.5 ermittelten Klassen müssen dazu in kontinuierliche Werte gewandelt werden. Jeder Klasse in C4.5 ist ein Intervall zugeordnet, innerhalb dessen das geschätzte maximale Angebot der nächsten drei Stunden liegt. Unter der Annahme, dass die Kundenzufriedenheit ein wesentlicher Faktor für einen gewinnorientierten Betrieb von Netzwerken ist, bietet das Überschätzen des zukünftigen Angebots eine zusätzliche Sicherheit bezüglich der Verfügbarkeit freier Leitungen. Als Eingabewert für die Dimensionierung durch Median wird deshalb die obere Grenze des der Ergebnisklasse zugeordneten Intervalls benutzt. Bei korrekter Bestimmung der Klassen und unter der Voraussetzung, dass sich das Netzverhalten zwischen den Tagen nicht wesentlich ändert, kann demzufolge kein Unterschätzen des maximalen Angebots auftreten. Die Auslastung des gesamten Netzes wird dadurch zwangsläufig sinken, ist allerdings die dadurch aufgebaute Reservekapazität nicht zu groß, überwiegt der wirtschaftliche Nutzen, jederzeit einen Verbindungswunsch erfüllen zu können gegenüber der brachliegenden Kapazität.

Dieselbe Argumentation wurde bei Bestimmung der asymmetrischen Kosten für den Einsatz der SVM angewandt.

Ein weiterer Punkt, der sich möglicherweise bei der Übertragung der Erfahrungen mit dem Simulator auf das reale System als Problem erweisen könnte, hat sich bei Betrachtung der Datentransformationen mit C4.5 gezeigt. Die direkte Berücksichtigung des Zeitpunkts der Simulationsausgabe hat eine deutliche Verbesserung der Ergebnisse mit sich gebracht, wobei C4.5 das Attribut „Zeitindex“ dementsprechend häufig für die Erstellung der Bäume verwendet hat. Dies lässt die Vermutung zu, dass eine starke Abhängigkeit des Maximums der nächsten Stunden vom aktuellen Zeitpunkt besteht und der Punkt der Mustererkennung durch Betrachtung der Kurvenverläufe vernachlässigt wird.

Die ATM-Simulation geht im normalen Betrieb für jeden Durchlauf/Tag von denselben durchschnittlichen Angebotswerten aus, wodurch sich ähnliche Kurvenverläufe zuzüglich der Schwankung um diesen Durchschnittswert ergeben. Im realen Telekommunikationsnetzwerk muss diese Abhängigkeit von der Zeit nicht zwangsläufig gegeben sein, sondern es sind Verschiebungen oder Deformationen der Kurven durch besondere Ereignisse (Feiertage, Großveranstaltungen...) denkbar.

Um solche Effekte beurteilen zu können, wird ein weiteres Verfahren für die Vorhersage der Maxima eingeführt, die *Mittelwertbildung*. Bei dieser Methode wird innerhalb des gleichen Fensters, das auch für die anderen Verfahren zur Verfügung steht, der Mittelwert für jeden Zeitpunkt gebildet. Dadurch entsteht eine Art „musterhafter Angebotskurvenverlauf“, so dass zur Vorhersage das Maximum innerhalb dieser Kurve gesucht wird und sich damit eine absolute Abhängigkeit vom Zeitpunkt ergibt. Durch die einfache Mittelwertbildung ist diese Methode sehr empfindlich gegenüber Veränderungen der Angebotskurven einzelner Durchläufe, wie sie u.a. bei Steigerungen im Angebot oder Ausreißern auftreten, bietet aber eine gute Basis, um die Auswirkungen dieser Veränderungen auf die anderen Verfahren vergleichen und einschätzen zu können.

Die Kurven in den Abbildungen 6.1 bis 6.5 geben den absoluten quadratischen Fehler zwischen vorhergesagtem und realem Maximalwert für den jeweiligen Zeitpunkt wieder, sowohl gemittelt über alle Dienste, als auch beispielhaft für einzelne Dienste, um Unterschiede, die zwischen den qualitativ verschiedenen Angebotskurven bestehen, in ihren Auswirkungen betrachten zu können. Die Angebotswerte der Dienste können sich dabei in sehr unterschiedlichen Bereichen bewegen, so sind für den Dienst Telefonie Werte im Bereich mehrerer 1000 Erlang üblich, während es für HDTV nur ein geringes Angebot von wenigen Erlang Größe gibt. Bei Berechnung der quadratischen Abweichung über alle Dienste und Links werden Dienste mit niedrigem durchschnittlichen Verkehr entsprechend weniger stark in das Ergebnis einfließen. Im Gegensatz dazu weisen Dienste mit geringem Verkehr häufig einen hohen Bedarf an Bitrate auf, so dass für jede geschaltete Verbindung mehr Bitrate auf den benutzten physikalischen Leitungen reserviert werden muss als für andere Dienste. Kosten für den Betrieb werden allerdings auch maßgeblich durch die Höhe der benötigten Bitrate bestimmt, da diese auf dem Link nicht für andere Verbindungen zur Verfügung steht. Deshalb wird an dieser Stelle eine Kostenfunktion mit Gewichtungen für jeden Dienst eingeführt, welche diesen Bedarf berücksichtigt. Für die Bestimmung der Gewichte bietet sich die mittlere Bitrate an, die ebenfalls von Median für Dimensionierung und Optimierung des Netzes benutzt wird und Dienste mit dynamischer Bitrate betrachtet. Die durchschnittliche Bitrate lässt sich durch unterschiedliche Methoden ermitteln, einzelne Verfahren wurden auf ihre Tauglichkeit für den Dimensionierungsprozess in [Scheer00] betrachtet. Als stabilstes Verfahren hat sich dabei die Methode von *Chiotis* erwiesen, andere Verfahren zeigen in der Praxis die Tendenz, die tatsächliche Bitrate zu über- oder unterschätzen.

Tabelle 6.1 zeigt die Bitrate der einzelnen Dienste und gibt die Gewichtung an, wobei Dienst 1 mit 54 kBit/s als Basis dient. Die Gewichte fließen ebenfalls quadratisch in die Fehlerbewertung mit ein, was letztendlich die sehr hohen Werte für den in den Abbildungen dargestellten Fehler erklärt.

Nr.	Dienst	Bitrate	Gewichtung
1	Telefonie	54 kBit/s	1
2	Musik	847 kBit/s	15,7
3	Videokonferenz	1115 kBit/s	20,6
4	Fernsehen	7623 kBit/s	141,2
5	Datenübertragung	27650 kBit/s	512
6	HDTV	74500 kBit/s	1379,6

Tabelle 6.1: Durchschnittliche Bitraten der Dienste

Um bei der quadratischen Fehlerbestimmung weiterhin eine Unterscheidung zwischen Über- und Unterschätzen zu ermöglichen, geht dies direkt in die Fehlerberechnung mit ein, d.h. negative Abweichungen in den Diagrammen entsprechen einem überschätzten Angebot. Die X-Achse als Basis gibt damit den optimalen Fall der hundertprozentig genauen Prognose an.

An dieser Stelle sei noch eine Besonderheit der Support Vector Machine erwähnt, die auftritt, da es sich um ein Regressionsverfahren handelt: Durch Schwankungen, die in der gelernten Zielfunktion natürlicherweise auftreten, können negative Werte vorhergesagt werden ohne dass zuvor in den Lerndaten negative Werte vorhanden gewesen wären. In der Realität kann es keinen angebotenen Verkehr kleiner Null geben, so dass negative Werte gesondert behandelt werden müssen. Dies geschieht, indem davon ausgegangen wird, dass jeder vorhergesagte negative Angebotswert real ein sehr kleines Angebot darstellen sollte. Um zu vermeiden, dass die Optimierungsverfahren in Median den Link vollständig entfernen und damit jeder eintreffende Ruf auf diesem

Link verloren geht, wird das Angebot für den betroffenen Link nicht einfach auf Null gesetzt, sondern eine geringe Kapazität als Sicherheit reserviert. Vergleichbar ist dies mit einer Vorhersage durch C4.5, bei der die kleinste Klasse ermittelt wurde.

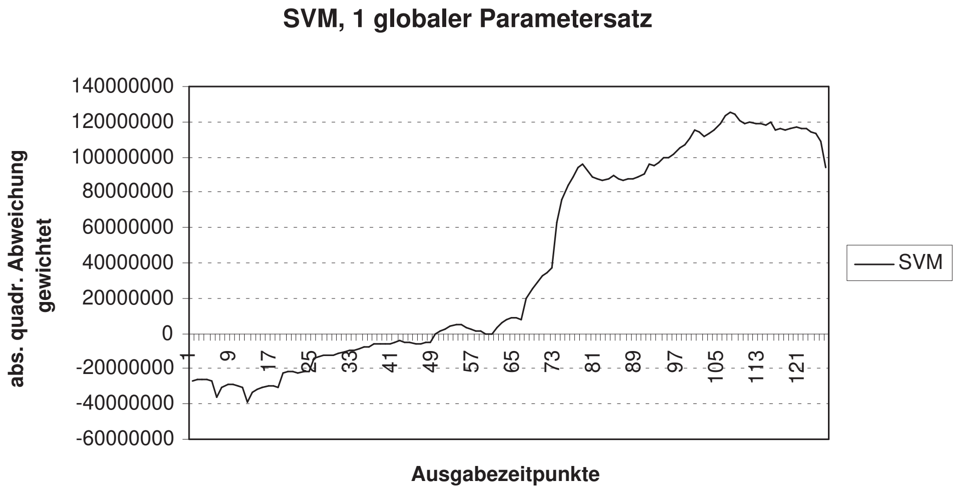


Abbildung 6.1: Prognose mit der SVM und einem Parametersatz für alle Dienste

Abbildung 6.1 stellt die Prognose über einen Tag mit der Support Vector Machine dar. Die Einheiten entlang der X-Achse entsprechen dem Ausgabetakt der Simulation, in diesem Fall 600 s je Einteilung. Der vorhergehende Lernprozess beinhaltete den Versuch, lediglich eine globale Parameterauswahl (Kernel, Kapazität...) zu benutzen, die sich bei den Versuchen im vorhergehenden Kapitel wenn auch nicht für jeden Dienst als optimal erwiesen, so doch zumindest einen Kompromiss erlaubt hat. Bei Betrachtung der Prognose zeigt sich jedoch, dass die realen Angebote entgegen der Einstellung, unterschätzte Werte um den Faktor drei stärker zu bestrafen als überschätzte, in großen Teilen weit unterschätzt werden.

Deutlich verbesserte Ergebnisse ließen sich mit für jeden Dienst getrennten Parametersätzen erzielen. Abbildung 6.2 zeigt die Abweichungen, wobei für jeden Dienst die in Kapitel 5 ermittelten Konfigurationen eingesetzt wurden. Insgesamt werden die realen Angebote grundsätzlich überschätzt, was zu einer leichten Überdimensionierung des Netzwerks führt. Schwächen offenbaren sich in den frühen/späten Stunden des Tages (Zeit ca. Ausgabeintervalle 1–42 oder 0–7 Uhr), wo das Angebot deutlich überschätzt wird, und zur kritischen Zeit der Hauptverkehrsstunde der Telefonie (Ausgabeintervalle 66–84 oder 11–14 Uhr). Zu den folgenden Zeiten, an denen teilweise andere Dienste ihre Hauptverkehrsstunde besitzen, findet dagegen eine deutliche Annäherung der Prognose an das reale Angebot statt.

Abbildung 6.3 stellt im Vergleich dazu die Abweichungen für C4.5 und zum

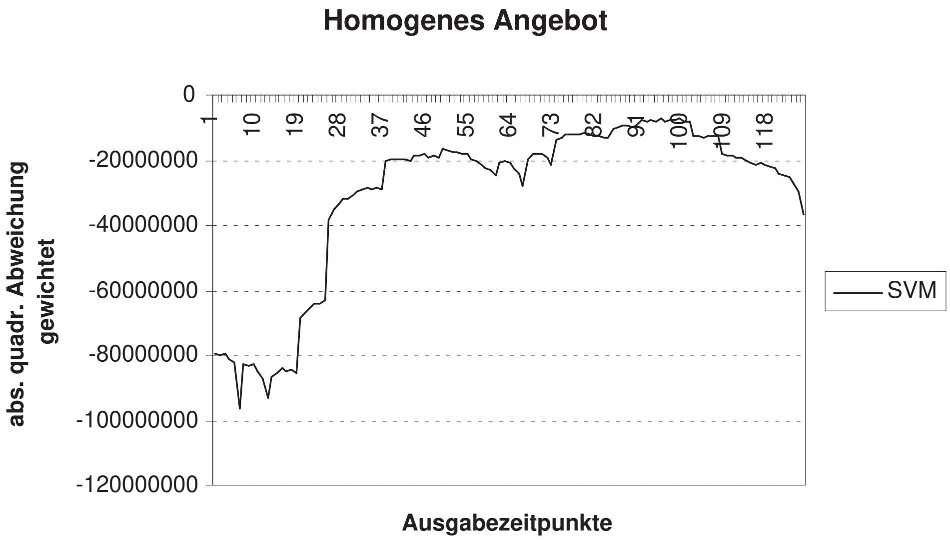


Abbildung 6.2: Prognose mit der SVM und einem Parametersatz je Dienst

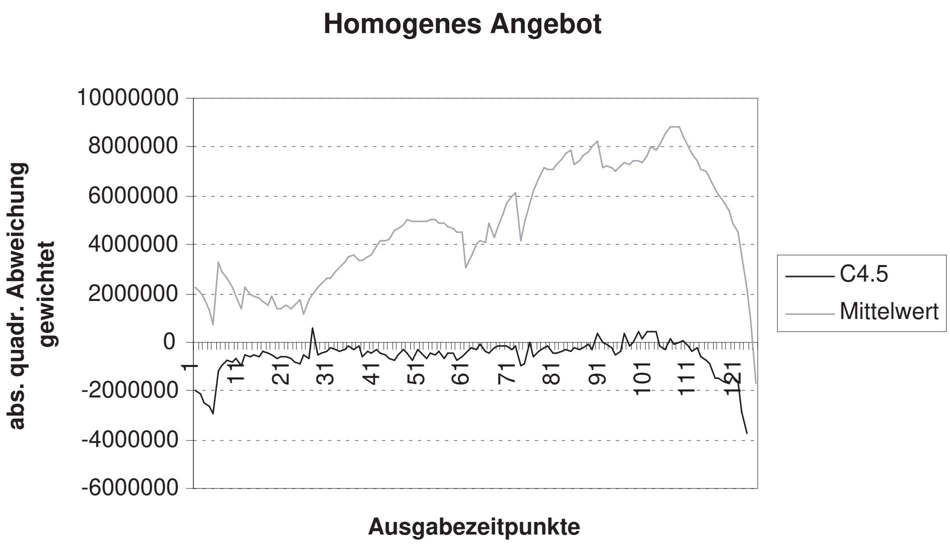


Abbildung 6.3: Prognose mit C4.5 und Mittelwert

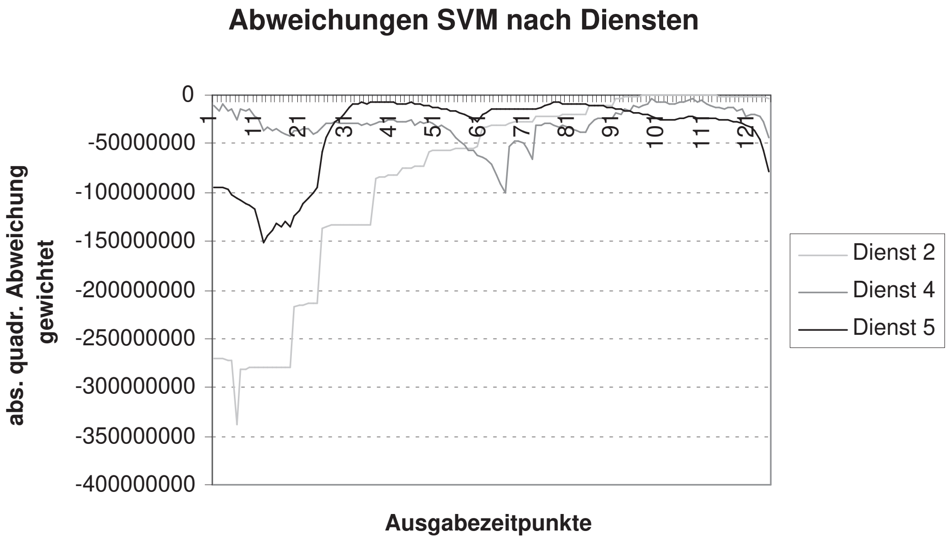


Abbildung 6.4: Prognose mit der SVM, aufgeteilt nach Diensten

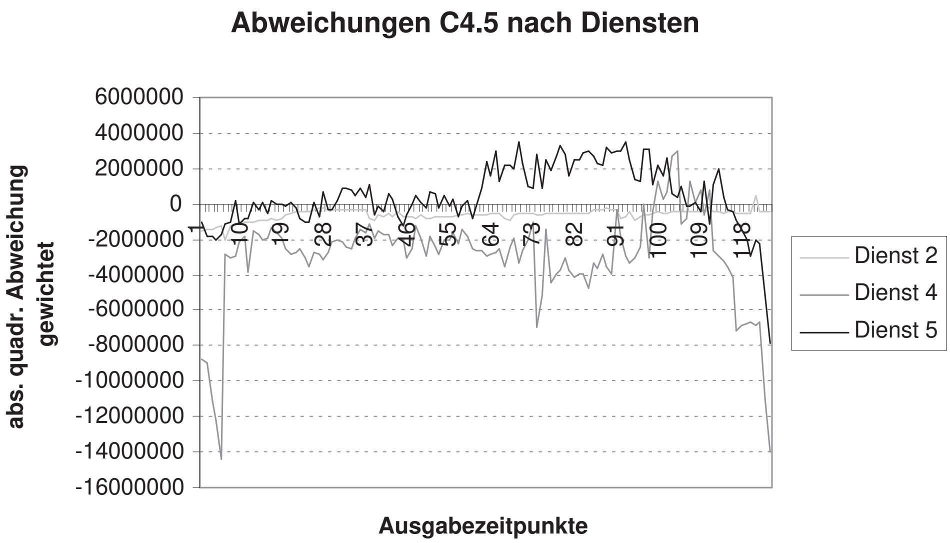


Abbildung 6.5: Prognose mit C4.5, aufgeteilt nach Diensten

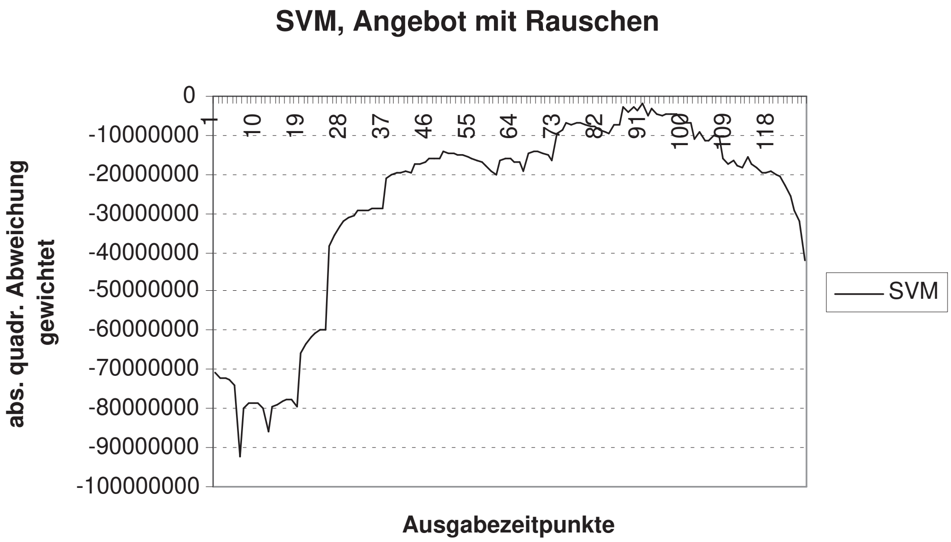


Abbildung 6.6: Prognose mit der SVM auf zusätzlich verrauschten Daten

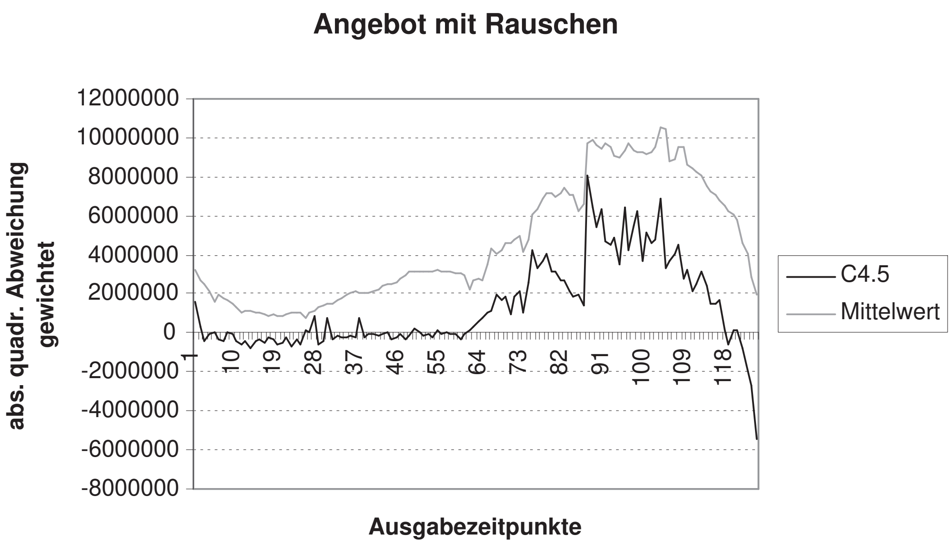


Abbildung 6.7: Prognose mit C4.5 und Mittelwert auf zusätzlich verrauschten Daten

Mittelwert über den Trainingsdurchläufen dar. Diese beiden Kurven sind in einem von der SVM getrennten Diagramm dargestellt, da die Abweichungen von einer deutlich kleineren Größenordnung sind. Bei Betrachtung der zum Mittelwert gehörigen Kurve wird ersichtlich, dass die Testdaten (die Angebotswerte des Tages, für den eine Prognose erstellt und mit dem verglichen wurde) höher als der Durchschnitt liegen, da das reale Angebot zu fast jeder Zeit unterschätzt worden ist. Trotzdem haben weder SVM noch C4.5 die Werte in einem Maß wie bei der Mittelwertbildung unterschätzt. Die Kurve für C4.5 zeigt in diesem Experiment die genaueste Annäherung an die realen Werte und überschätzt das Angebot bis auf wenige Stellen. Es fällt auf, dass die Abweichungen der SVM um den Faktor 10 höher liegen als bei den beiden anderen Verfahren. Da allerdings der quadratische Fehler betrachtet wird, ergibt sich lediglich eine Abweichung um einen Faktor ≈ 3, was aus dem gewählten Verhältnis von 1:3 zwischen Unter- und Überschätzen folgt.

Abbildung 6.4 zeigt die Abweichungskurven aufgeschlüsselt für drei der Dienste bei Prognose durch die SVM, Abbildung 6.5 analog für C4.5, um die Bewertung der Vorhersage für qualitativ unterschiedliche Angebotskurven zu ermöglichen. Daraus lässt sich erkennen, dass die Gesamtabweichung der SVM maßgeblich durch Dienst 2 bestimmt wird, der in den ersten Stunden des Tages die größte Abweichung aufweist. Durch seine Gewichtung im Verhältnis zum Angebot gehört dieser Dienst zu den „teueren“ Diensten, die einen hohen realen Bedarf an Bitrate aufweisen, und wirkt sich deshalb stark auf die Abweichung aus. Damit bietet es sich für diesen Dienst unter Umständen an, nochmals alternative Parametereinstellungen zu untersuchen. Dienst 4 dagegen beginnt bereits mit einer vergleichsweise kleinen Differenz zwischen realem und vorhergesagtem Wert. Ab der Mittagszeit nähern sich die Kurven insgesamt aneinander an.

Die Aufschlüsselung nach Diensten für C4.5 zeigt, dass in Teilen das Angebot mancher Dienste ebenso unter- (Dienst 5) wie überschätzt (Dienst 4) wird. Andere Dienste (Dienst 2) dagegen zeigen im Vergleich zu den anderen Verfahren die beste Anpassung an die realen Angebote, ohne diese jedoch zu unterschätzen.

Ein weiteres Experiment mit den Verfahren ist in den Abbildungen 6.6 und 6.7 dargestellt. Sowohl zu den Trainings- als auch zu den Testdaten wurde ein geringes Rauschen in Form von Sprüngen um +/-30% im Mittel und einer durchschnittlichen Rate von vier dieser Sprünge pro Tag zusätzlich zu den bereits durch die Simulation vorhandenen Schwankungen hinzuaddiert. Mittelwert und SVM zeigen sich hierbei als stabil und verändern ihre Abweichungen kaum. C4.5 dagegen wird durch die Schwankungen stark in seiner Prognosequalität beeinflusst und verschlechtert sich in der zweiten Hälfte der Tagesprognose deutlich.

6.2 Angebotssteigerungen

Um dem wachsenden Telekommunikationsmarkt Rechnung zu tragen, bietet Median die Möglichkeit, ein steigendes Angebot im Netz zu simulieren. Dazu werden bei Start von Median bis zu zwei Faktoren angegeben, die den durchschnittlichen Zuwachs in jeder der zwei Steigerungsphasen bestimmen. Die Zeitabschnitte, in denen die Steigerungen durchgeführt werden, verteilen sich gleichmäßig über den Simulationszeitraum, d.h.: Wird eine Gesamtzeit von 30 Durchläufen angesetzt und Faktoren 1,05 und 1,1 angegeben, beginnt die erste Steigerungsphase bei Tag 11 mit einer durchschnittlichen Steigerung von 5%, Phase zwei bei Tag 21 mit 10% Steigerung.

Um die Stabilität der Verfahren hinsichtlich ihrer Vorhersagegenauigkeit zu testen, wurden die Lernzyklen zunächst mit den normal großen Angebotskurven gestartet und anschließend daran unter Berücksichtigung einer Angebotskurve mit Steigerung die Tagesprognose erstellt. Abbildung 6.8 zeigt die Ergebnisse bei 5% Steigerung, Abbildungen 6.9 und 6.10 bei 20%.

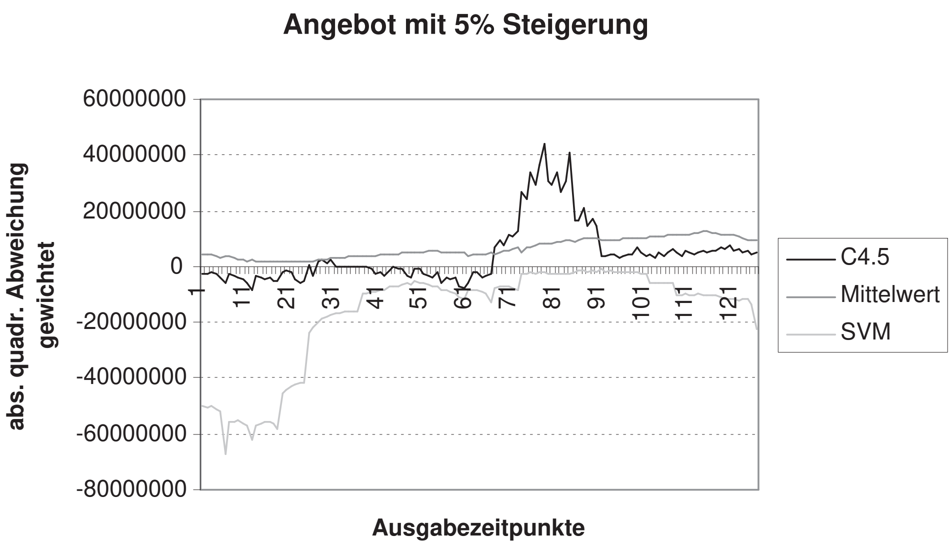


Abbildung 6.8: Prognose mit SVM, C4.5 und Mittelwert bei 5% Angebotssteigerung

Wie zu erwarten vollzieht die Prognose durch Mittelwert diese Trends einfach nach. C4.5 dagegen weist für den Fall der 5-prozentigen Steigerung im Vergleich zur Vorhersage unveränderter Angebotskurven große Abweichungen im Bereich 67–91 auf. Für den Fall der 20-prozentigen Steigerung tritt zwar ebenfalls ein deutliches Unterschätzen des realen Angebots auf, dieses fällt aber nicht so deutlich aus wie bei 5%.

Die SVM erweist sich bei der Vorhersage steigender Angebote als stabilstes Verfahren. Zwar wirkt sich an einigen Stellen auch hier die Steigerung auf die Abweichung aus, allerdings bleibt die Kurve auch bei 20% noch unterhalb des realen Angebots.

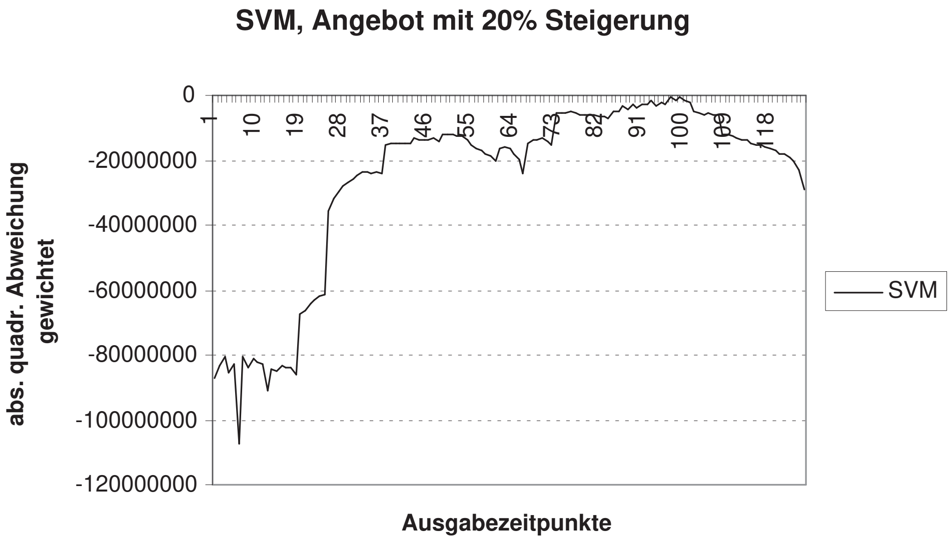


Abbildung 6.9: Prognose mit der SVM bei 20% Angebotssteigerung

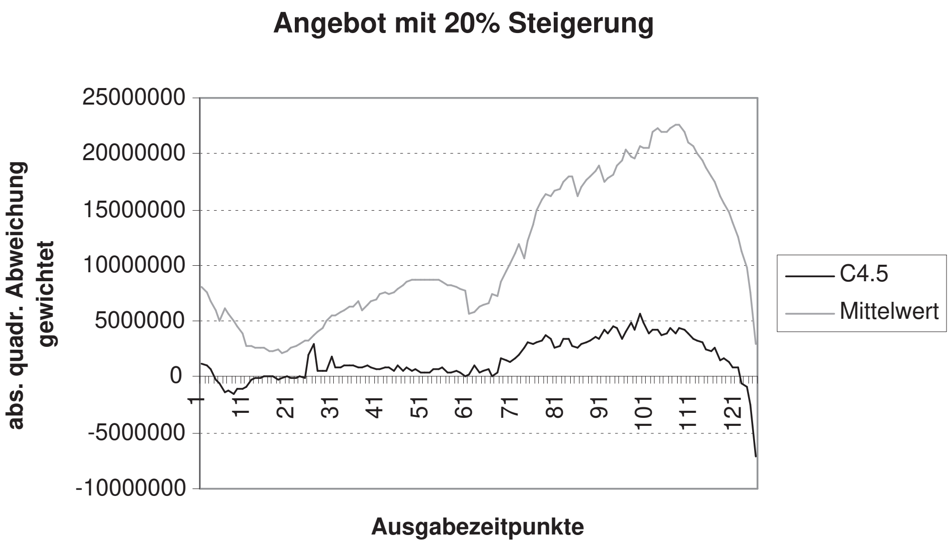


Abbildung 6.10: Prognose mit C4.5 und Mittelwert bei 20% Angebotssteigerung

6.3 Einführung von Wochentagen

In seiner aktuellen Version sind die Fähigkeiten des Simulators noch sehr begrenzt, sind doch nur gleichförmige oder stufenweise gesteigerte Angebote möglich. Dadurch besteht die Gefahr, dass die Data Mining-Verfahren lediglich die im Simulator eingesetzten Methoden lernen und die gewonnenen Erkenntnisse nicht direkt auf die Realität übertragbar sind.

Denkbar sind z.B. unterschiedliche Verhaltensmuster hinsichtlich des Telefonverkehrs in Abhängigkeit des Tages. An Wochenenden bestehen ganz andere Bedürfnisse und Hintergründe für Telefongespräche, so dass dann auch qualitativ andere Angebotskurven zugrunde liegen. Um diesem Umstand gerecht zu werden, wurden bestehende Ausgaben des Simulators über das Netzverhalten so transformiert, dass bestimmte Durchläufe auf einer alternativen Angebotskurve basieren. Abbildung 6.11 zeigt das Beispiel einer Kurve für den Dienst Telefonie, wie sie für einen Samstag (jeden sechsten Durchlauf) denkbar wäre.

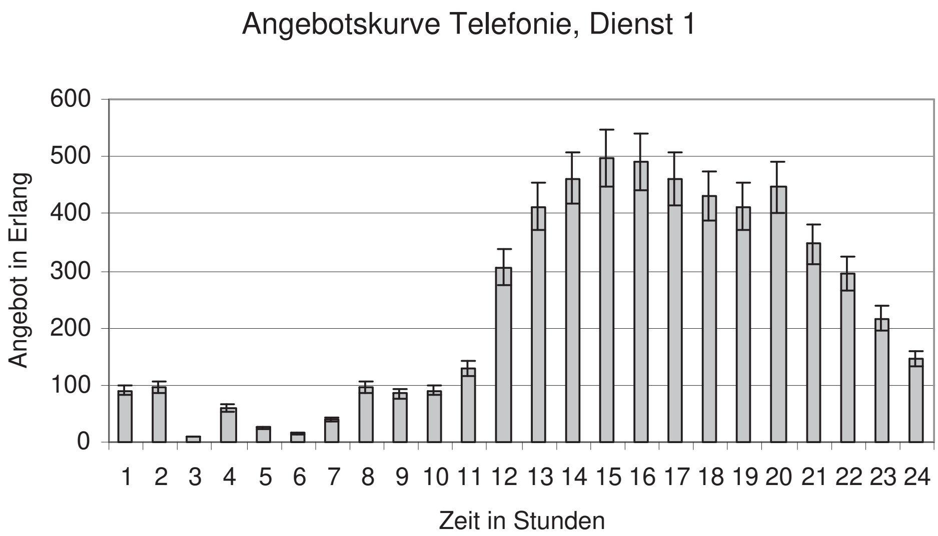


Abbildung 6.11: Beispiel einer an das Wochenende angepassten Angebotskurve

Der Anstieg im Telefonverkehr findet hierbei später statt im Vergleich zum herkömmlichen Verlauf in Abbildung 4.2 auf Seite 36 und zieht sich weiter in die Abendstunden hinein. Insgesamt fällt der angebotene Verkehr deutlich geringer aus als an einem Werktag.

In den durchgeführten Experimenten wurde zwischen drei Kurven unterschieden: Werktag, Samstag, Sonntag. Im ersten Experiment erfolgte die Bereitstellung der Trainingsdaten wie zuvor bei homogenem Angebot als Sequenz der Durchläufe hintereinander, im zweiten ist den DM-Verfahren als zusätzliche Information die Art des Tages zur Verfügung gestellt worden. Im Fall von C4.5 heißt dies die Bereitstellung eines weiteren Attributs *Wochentag*, das angibt, ob es sich um Werktag, Samstag oder Sonntag handelt; bei der SVM wurde für jeden Tag ein eigenes Attribut eingeführt (*Binärisierung*).

Beide Möglichkeiten sind in den Abbildungen 6.12 und 6.13 dargestellt, jeweils mit und ohne eigene Wochentags-Attribute.

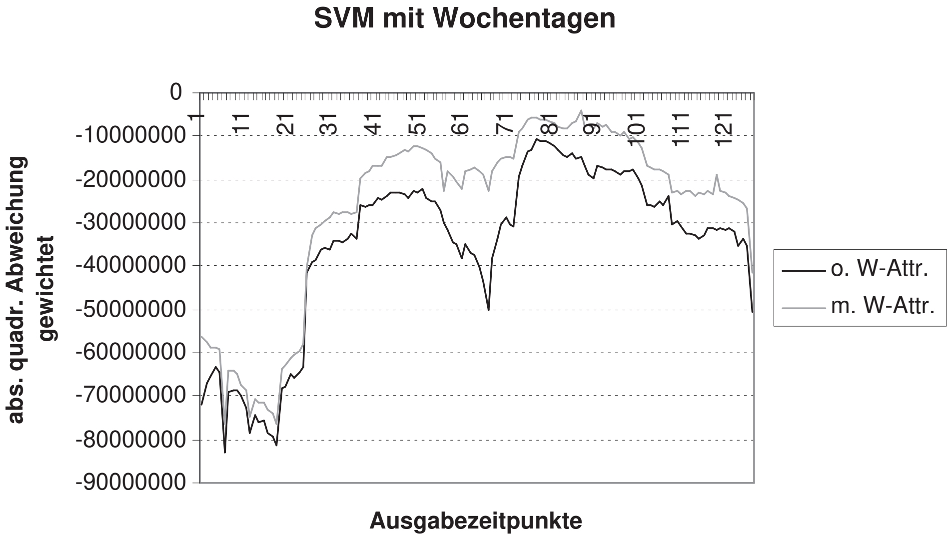


Abbildung 6.12: Prognose mit der SVM bei Angebotskurven in Abhängigkeit des Wochentags

Wie zu erwarten bereitet die Verwendung qualitativ unterschiedlicher Zeitreihen den Verfahren mehr oder weniger stark Probleme, da die unterschiedlichsten Muster in den Daten auftreten. Davon betroffen war dabei auch der hier aus Platzgründen nicht angegebene Mittelwert, da der Durchschnitt über alle Tage entsprechend stark vom Durchschnitt einzelner Wochentage abweicht.

Wie die Grafiken zeigen, konnte die SVM durch eigenständige Wochentags-Attribute profitieren und das Ergebnis etwas verbessern. Kaum eine Wirkung hat die Einführung dieses Attributs bei C4.5 gezeigt, wo sich die Kurve nicht an entscheidenden Stellen ändert.

Insgesamt gesehen scheint die Support Vector Machine eher für einen Einsatz bei unterschiedlichen zugrunde liegenden Angebotskurven, wie sie in der Realität vermutlich auftreten, geeignet zu sein.

6.4 Zusammenfassung

Die Verfahren zeigen auf den jeweiligen Daten unterschiedliche Stärken und Schwächen. Die einfache Mittelwertbildung besitzt in ihrer eingesetzten Form den Nachteil, jede Änderung (z.B. Steigerungen) in den Daten über die Durchläufe gesehen gleich zu behandeln, während die anderen Verfahren vorhandene Muster berücksichtigen und daher zyklisch auftretende Schwankungen (Wochentage...) teilweise erkennen können.

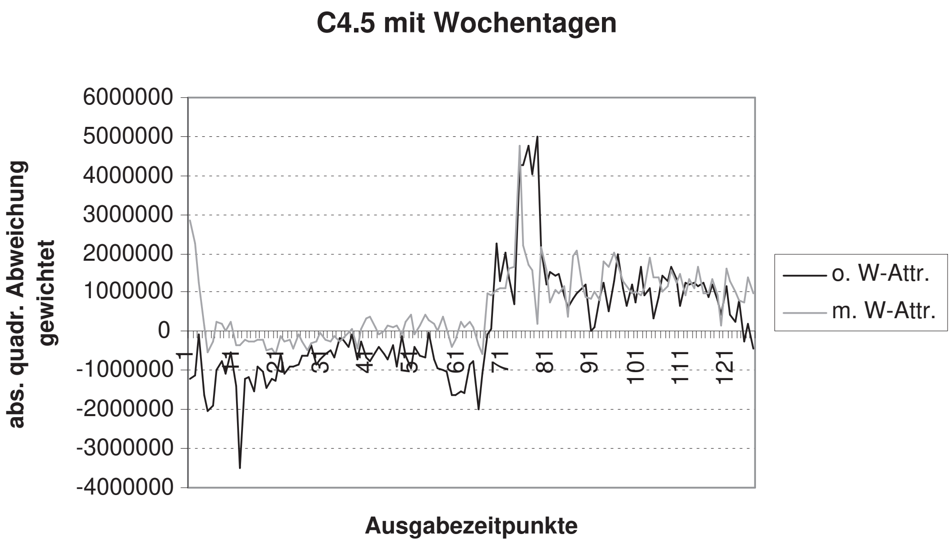


Abbildung 6.13: Prognose mit C45 bei Angebotskurven in Abhängigkeit des Wochentags

Eigene Erfahrungen mit Telekommunikationsanbietern bestätigen, wie sensibel der Kunde auf Engpässe in den vorhandenen Kapazitäten reagiert. Als Schlussfolgerung daraus ergibt sich die unterschiedliche Bewertung von über- und unterschätzten Angeboten. Wie die Prognose durch den Mittelwert gezeigt hat, ergeben sich hier schnell Abweichungen, wenn das reale Angebot deutlich über dem Durchschnitt liegt. Erweiterungen, wie z.B. zusätzliche pauschal bestimmte Sicherheitsreserven, könnten hier Abhilfe schaffen. Die Bedeutung von Sicherheitsreserven darf auch unter dem Aspekt des Rekonfigurationsaufwands nicht vernachlässigt werden, da ständige Engpässe im Netzwerk die Notwendigkeit zum Umschalten der betroffenen Verbindungen verursachen.

Als stabilstes Verfahren hinsichtlich des Kriteriums eher zu Überschätzen, hat sich die SVM erwiesen, bei der auch Steigerungen von 20% kaum zum Unterschätzen führen. Die genauesten Prognosen im Test hat C4.5 erzeugt, im Gegensatz dazu war die Qualität der Prognose starken Schwankungen bei Veränderungen der Daten unterworfen, so dass es schwer fällt, die Güte einer Prognose im Vorfeld abzuschätzen.

Im Endeffekt lässt sich kein Verfahren pauschal besser als die übrigen bewerten. Für den konkreten Einsatz kann es unter Umständen auch sinnvoll sein, unterschiedliche Methoden parallel auf unterschiedlichen Links und Diensten einzusetzen. Wichtig dabei bleibt die permanente Überwachung der Vorhersagequalität und daraus resultierend die Verbesserung der Prognosen.

Kapitel 7

Integration in MEDIAN

Als abschließende Phase dieses Data Mining-Projekts erfolgt die Integration in das Gesamtsystem Median um die Anforderungen an das dynamische Netzwerkmanagement vollständig zu erfüllen. Zwei Gesichtspunkte sind dabei besonders zu berücksichtigen: erstens der bei den vorhergehenden Betrachtungen nur eine untergeordnete Rolle spielende Geschwindigkeitsaspekt. Als prototypische Anwendung bietet Median grundsätzlich viele Ansatzpunkte zur Effizienz- und Geschwindigkeitssteigerung; grundlegende Konzepte wie die in Abschnitt 7.1 dargestellte Parallelisierung sollten allerdings bereits in der Konzeptionsphase beachtet werden. Skalierbarkeit und Speed-Up sind dabei wichtige Punkte für den Echtzeit-Charakter von Median. Zweitens stellt die Integration besondere Ansprüche an Flexibilität und Modularisierung um die einfache Anpassung an zukünftige Entwicklungen zu ermöglichen. Sinnvollerweise werden die Verfahren so weit wie möglich voneinander gekapselt und Programmcode, der allgemein benutzbare Teile enthält, wiederverwendet (siehe Abschnitt 7.3).

Abschnitt 7.2 erläutert die Einbindung in die grafische Oberfläche von Median und die Möglichkeiten zum Setzen der Parameter für die DM-Verfahren.

7.1 Parallel Virtual Machine

Für einen vollständigen Lernvorgang über die Verbindungen eines Netzwerks mit 24 Knoten und 6 Diensten müssen 3312 einzelne Trainingsvorgänge gestartet werden, was jeweils die Transformierung der Daten in die an das Verfahren und die Parameter angepasste Form und darauffolgend die eigentliche Modellbildung durch das Verfahren beinhaltet. Da die Angebote jedes Links/Dienstes unabhängig von den übrigen sind, bietet es sich an, die Berechnungen gleichzeitig parallel durchzuführen und nur die Ausgaben der Simulation an die einzelnen Prozesse zu verteilen.

Zu diesem Zweck bedient sich Median der *Parallel Virtual Machine (PVM)*. Dieses System stellt Bibliotheken und Funktionen zur Verfügung, die das Parallelisieren eigener Programme in heterogenen Rechnerumgebungen realisieren. Abbildung 7.1 zeigt das zugrundeliegende Konzept.

Die Implementierung der PVM erfolgt dabei vollkommen transparent im Hintergrund. In den Programmcode der zu parallelisierenden Programme werden Anweisungen eingefügt, welche die Steuerung der Maschine übernehmen. Unterstützung besteht für die unterschiedlichsten Programmiersprachen, Prozessormodelle und Betriebssysteme, so dass Probleme in heterogenen LAN-Umgebungen gelöst werden können. Die Rechner im vorhandenen herkömmlichen Netzwerk werden durch PVM zu einem sogenannten *Cluster* verbunden, der sich nach außen wie ein MPP[1]-System mit verteiltem Speicher verhält. Jeder im Cluster enthaltene Rechner beherbergt den PVM-Dämon, der die Verwaltung und Koordinierung zwischen den einzelnen Maschinen übernimmt. Innerhalb des Clusters sind alle Maschinen gleichberechtigt, Unterscheidungen und Prioritätsvergaben werden durch die zu parallelisierende Anwendungssoftware geregelt. Zur Nutzerinteraktion mit dem Cluster existiert eine Konsole, über die Rechner im Cluster hinzugefügt, entfernt werden können oder ihr Status abgefragt werden kann.

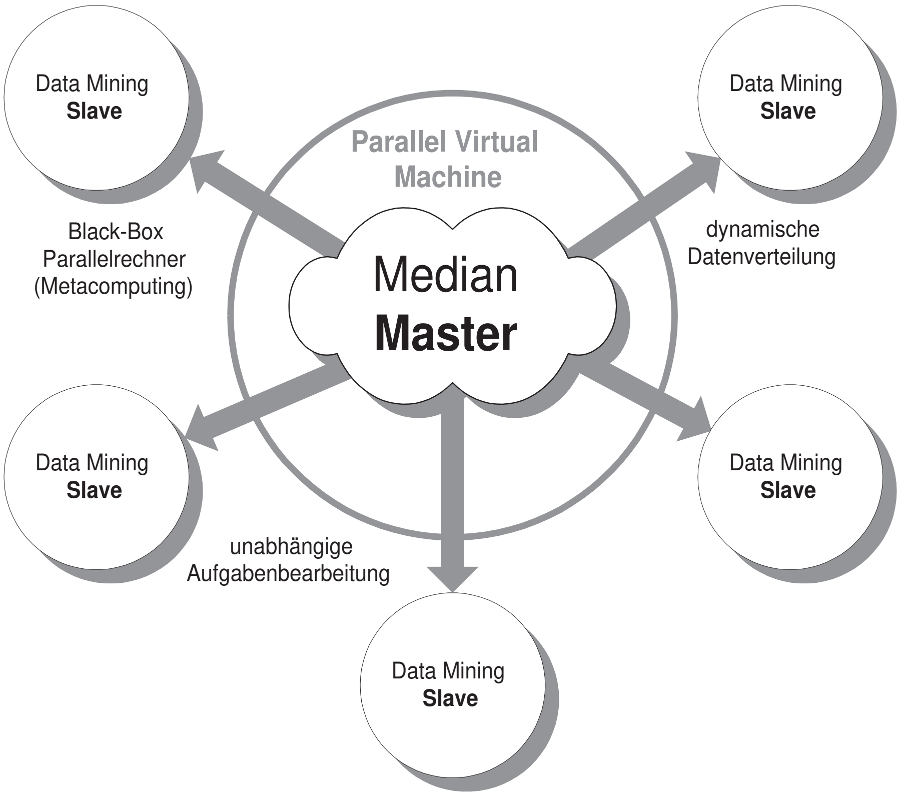


Abbildung 7.1: Konzept der Parallel Virtual Machine

In Abbildung 7.1 ist ein möglicher Cluster für dieses Projekt schematisch skizziert. Median übernimmt hierin die Funktion des *Masters* und somit die Koordinierung der Aufgaben. Die *Slaves* stellen von Median eigenständige Programme dar, die die Data Mining-Verfahren integrieren und mit diesen die von Median verteilten Aufträge ausführen.

Die PVM arbeitet nachrichtenorientiert, d.h. die Koordination zwischen den

[1] Massive Parallel Processing.

Prozessen erfolgt über Sende- und Empfangsoperationen, die sowohl blockierend als auch nicht blockierend sein können. Empfangsoperationen sind für Slaves in der Regel blockierend, da diese auf die notwendigen Daten warten müssen. Sendeoperationen sind im Gegensatz dazu meist nicht blockierend, da in diesem Fall der sendende Prozess weiter seine Aufgaben bearbeiten, bzw. Aufträge verteilen kann.

Wichtig für die effiziente Ausführung der Prognose über alle Links des Netzwerks ist die asynchrone Verteilung der Prognosen der einzelnen Links. Die Geschwindigkeiten der einzelnen Rechner in einem heterogenen Cluster können sich stark unterscheiden, weswegen die Aufträge von Median an die einzelnen Slaves nicht in einer festen Reihenfolge verteilt werden. Meldet ein Rechner die erfolgreiche Berechnung einer Vorhersage, erhält dieser sofort den nächsten Auftrag. In Median werden die beendeten Prognosen gesammelt und die Vorhersage für das vollständige Netz an die Dimensionierungs- und Optimierungsfunktionen übergeben.

Um die Vor- und Nachteile der Parallelisierung bewerten zu können und den Aufwand für Implementierung und Rechenkapazität zu rechtfertigen, wird der Begriff des *Speed Up* eingeführt:

Definition 7
Der *Speed Up* S_i ist definiert als Quotient aus der Zeit T_s für die Programmausführung auf einem einzelnen Prozessor (seriell) und der Zeit $T_{p,i}$ für die Ausführung parallel auf i Prozessoren:

$$S_i = \frac{T_s}{T_{p,i}}.$$

Speed Up ist damit ein Maß für den zeitlichen Gewinn, der durch die Parallelverarbeitung erreicht werden kann. Für eine perfekte Parallelisierung, d.h. ohne algorithmisch oder hardwarebedingte Engpässe, auf absolut homogenen Maschinen ergibt sich ein zur Anzahl der Prozessoren linearer Speed Up. Im Normalfall wird sich der Zeitgewinn für homogene Rechner sublinear verhalten, da z.B. bei der Verteilung großer Datenmengen das lokale Netz limitierender Faktor ist oder sich einige Algorithmen nur schlecht parallelisieren lassen. Superlineare Speed Ups treten unter anderem dann auf, wenn Maschinen mit unterschiedlicher Rechenleistung vorhanden sind und als Basis für die sequenzielle Ausführungszeit T_s nicht die schnellste ausgewählt wird.

Abbildung 7.2 zeigt den Speed Up für Berechnungen mit bis zu sechs Rechnern, jeweils für einen Lernvorgang mit C4.5 und der SVM über das gesamte Netzwerk.

Die ersten vier verwendeten Rechner beinhalten jeweils eine 233 MHz Pentium Pro/ MMX-CPU mit 64 MB Hauptspeicher, verhalten sich bei grober Betrachtung dementsprechend homogen. Als fünfter Rechner kam ein Pentium III mit 733 MHz und 256 MB Hauptspeicher zum Einsatz, als sechster Rechner ein Pentium II mit 300 MHz und 256 MB, was die hohen Speed Up-Werte für die letzten beiden Rechner erklärt.

Die Werte für den Lernzyklus der SVM zeigen einen nahezu linearen Speed Up für die ersten vier Rechner, was darauf hindeutet, dass die Rechenzeiten

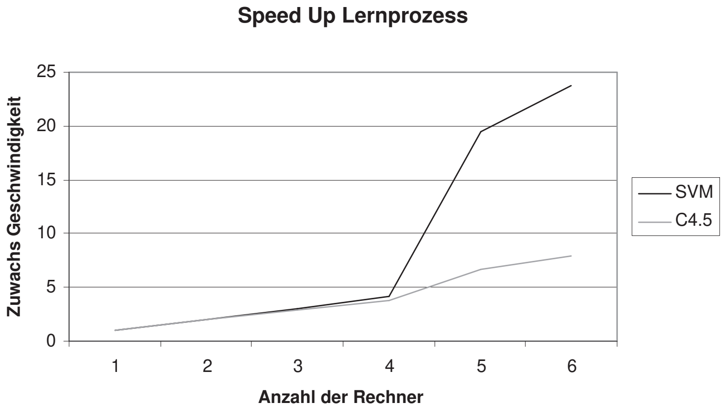


Abbildung 7.2: Speed Up Berechnungen

für das Training im Vergleich zu anderen Faktoren wie Übertragungsrate des Netzwerks und Festplattenzugriffen überwiegen und die Parallelisierung deutlichen Zeitgewinn gegenüber der sequenziellen Berechnung bringt. Die Speed Up-Messungen für die Berechnungen mit C4.5 zeigen für die ersten vier Rechner einen leicht sublinearen Geschwindigkeitsgewinn, d.h. auch hier überwiegt der Verbrauch von Rechenzeit gegenüber anderen Faktoren.

Ein deutlicher Unterschied zeigt sich für Clustergrößen von 5 und 6 Rechnern. Hier fällt der Speed Up für C4.5 deutlich geringer aus als bei Berechnungen mit der SVM. Durch die kürzeren Rechenzeiten für einzelne Links mit C4.5 spielen begrenzende Faktoren wie Festplattenzugriffe und Netzwerkgeschwindigkeit bereits früher eine Rolle als bei der SVM, weswegen die Support Vector Machine stärker von einer Parallelisierung profitiert als C4.5. Konkrete Rechenzeiten lagen bei 106 h, 55 min für die SVM und 5 h, 56 min für C4.5 auf einem 233 MHz-Computer beim Training über 10 Tage. Diese konnten mit 6 Rechnern auf 4 h, 31 min für die SVM und 45 min für C4.5 verringert werden.

Die Experimente mit verschiedenen Clustergrößen haben zusätzlich gezeigt, dass mit steigender Rechnerzahl das Risiko für den Ausfall eines Rechners entsprechend mit ansteigt. Um trotzdem eine erfolgreiche Berechnung aller Maxima zu gewährleisten, wird die Wartezeit für eine Bestätigung eines Slaves begrenzt. Überschreitet einer der Rechner diese Zeit (Timeout), überprüft Median den Status innerhalb der virtuellen Maschine. Slaves, deren Überprüfung ein negatives Ergebnis geliefert hat, werden aus der Liste der zur Verfügung stehenden Slaves entfernt und der an diese Maschine vergebene Auftrag bei Gelegenheit an die nächste freie Maschine nochmals vergeben. Wichtig ist hierbei, dass das Intervall für einen Timeout nicht zu klein gewählt wird, da ansonsten das Risiko besteht, eine intakte Maschine, die ihre Berechnung noch nicht abgeschlossen hat, von der Bearbeitung ihres Auftrags auszuschließen. Dies ist besonders für Berechnungen mit der Support Vector Machine kritisch, da diese stark schwankende Rechenzeiten für die numerischen Lösungsverfahren aufweist.

7.2 Programmdokumentation

Durch die Integration des Data Mining-Prozesses in das bestehende Median-System ergeben sich weitere Parametereinstellungen, die vom Anwender beim Start des Programms festgelegt werden müssen. Die Einstellung der Parameter erfolgt zur Laufzeit und nicht direkt im Programmcode, um nicht auf die in den vorhergehenden Kapiteln ermittelten Konfigurationen festgelegt zu sein, sondern beispielsweise flexibel auf grundlegend geänderte Angebotskurven reagieren zu können.

In diesem Abschnitt erfolgt die Beschreibung der neu hinzugekommenen Dialoge sowie die Erklärung zu den einzelnen Einstellmöglichkeiten. Abschließend erfolgt die anschauliche Darstellung eines „Lernen–Vorhersage–erneut Lernen“–Zyklus.

Abbildung 7.3 zeigt den Verfahrensauswahldialog, der sich direkt an den bereits zuvor bestehenden Dialog zur Bestimmung der aktiven PVM-Rechner anschließt. Hier wird das Verfahren ausgewählt, das für den Einsatz in Median benutzt werden soll. Der nächste Eintrag gibt die Anzahl an Tagen (synonym zu Durchläufen) an, nach denen genügend Daten für einen Lernvorgang gesammelt wurden und dieser gestartet werden kann. Gleichzeitig bestimmt dieser Wert das Fenster, das für erneute Lernvorgänge benutzt wird (innerhalb dieses Fensters vom aktuellen Zeitpunkt an zurückgerechnet werden die Durchläufe zum Lernen genutzt).

Die Checkbox „Bereits trainierte Daten verwenden“ kann für Testzwecke genutzt werden und ist dann zu aktivieren, wenn zuvor Median mit denselben Parametern betrieben wurde und aus diesem Grund problemlos die gelernten Angebotskurven aus den vorhergehende Durchläufen für die aktuelle Prognose verwendet werden können. Der normalerweise bei einem Neustart von Median nötige initiale Lernprozess für das gesamte Netzwerk entfällt somit und es können auf Basis der vorherigen gelernten Daten sofort Prognosen erstellt werden.

Im nächsten Eingabefeld wird der „Prognosehorizont“ festgelegt. Der Wert gibt die Anzahl der Ausgabeintervalle der Simulation an, die für die Bestimmung des Maximums herangezogen werden, was bei einem Ausgabeintervall von z.B. 600 s und einem Wert von 18 einem Prognosehorizont von 3 Stunden entspricht. Dieser Wert wird nicht von den Lernverfahren, sondern wesentlich durch die Dimensionierungs- und Optimierungsanforderungen von Median bestimmt.

Die letzten drei Felder in diesem Dialog steuern die Qualitätsüberwachung der Vorhersage. Das „Fenster für Genauigkeitsüberwachung“ gibt an, wie viele der letzten Vorhersagen auf ihre Qualität hin überprüft werden sollen. Überschreitet die mittlere relative Abweichung innerhalb dieses Fensters einen der Grenzwerte für Über- oder Unterschätzen, wird für die betroffenen Links ein neues Lernverfahren initiiert.

Im Gegensatz zum vorherigen Kapitel wird an dieser Stelle nicht die absolute Abweichung zwischen vorhergesagtem und realem Angebot gewählt, sondern die relative Abweichung, d.h. das Verhältnis von absoluter Differenz zwischen Vorhersage und Realwert zum Realwert. Diese Vorgehensweise weist die Schwäche auf, dass Abweichungen von 10% bei einem realen Wert von 1.000 Erlang genauso stark bewertet werden wie Abweichungen von 10% bei 1 Erlang und damit die Berücksichtigung der Abweichung nicht genau die durch den Bedarf an Bitrate entstandenen Kosten wiedergibt. Im Gegensatz dazu ermöglicht die Berücksichtigung des relativen Fehlers die Angabe von lediglich zwei Grenzen, die unabhängig von der quantitativen Größe des Angebots sind und somit nicht für jeden Link gesondert angegeben werden müssen. Der zeitliche Aufwand für das erneute Lernen einzelner Links, deren Vorhersagefehler die Grenzwerte verletzt hat, ist dabei im Vergleich zum Aufwand für einen Lernvorgang über das vollständige Netzwerk deutlich geringer. Ziel des erneuten Lernens ist es, die Vorhersagequalität zu verbessern sowie die vorhergesagte an die reale Kurve anzunähern und damit die Abweichung wieder unter die Grenzwerte zu bringen.

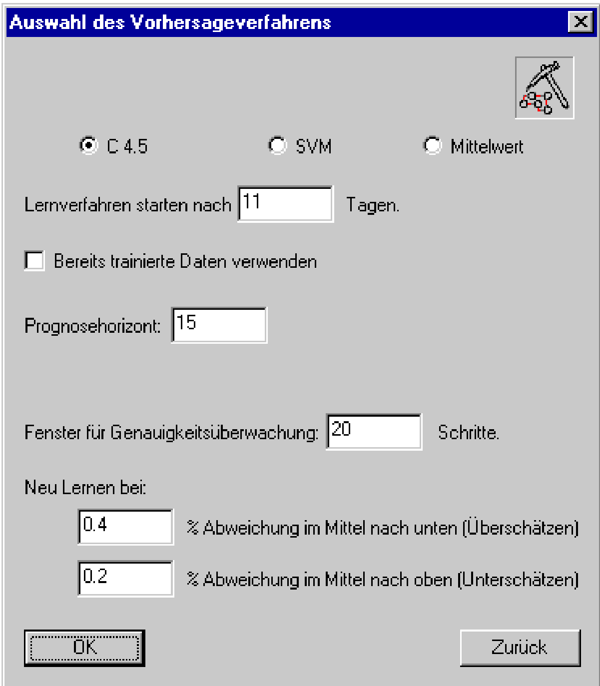


Abbildung 7.3: Dialog zur Auswahl des Data Mining-Verfahrens

Die übrigen Dialoge sind verfahrensabhängig und betreffen die jeweiligen Parametereinstellungen. Abbildung 7.4 zeigt die Einstellungen bei Wahl von C4.5. Eingabefeld „Berücksichtigte Vergangenheitswerte“ gibt die Länge der Historie an, d.h. die Anzahl der letzten Angebotswerte, die C4.5 für die Baumgenerierung zur Verfügung stehen.

Im nächsten Eingabefeld „Schrittweite Klassifizierung“ wird die Größe des Diskretisierungsintervalls angegeben, mit dem die Klassen für C4.5 gebildet werden. Da es nicht sinnvoll ist, diesen Wert global vorzugeben, wenn Dienste mit stark unterschiedlichen quantitativen Kurvenverläufen behandelt werden müssen, wird diese Schrittweite nur dann benutzt, wenn der folgende Eintrag „Anzahl Ergebnisklassen“ Null ist. Ansonsten wird der maximale Angebotswert für den jeweiligen Dienst und die Trainingsdaten bestimmt und die Schrittweite so festgelegt, dass sich die eingestellte Anzahl an Klassen ergibt.

„Zuwachsfaktor Klasseneinteilung“ bestimmt die Steigerung der Schrittweite von Klasse zu Klasse, beginnend bei der Klasse für das kleinste Angebot. Ein Wert von 1 hat dementsprechend keinen Effekt.

In „logarithmische Auswahl“ kann die Basis des Logarithmus, bzw. der Faktor, mit dem sich der Abstand zwischen den als Attribut ausgewählten Werten erhöht, angegeben werden (vergleiche die Experimente mit den Verfahren in den Kapiteln 4 und 5).

Die Checkboxen am unteren Rand des Dialoges aktivieren jeweils die Bereitstellung der Steigung oder des Zeitindex als Attribut, oder wandeln die kontinuierlichen Angebotswerte entsprechend der Schrittweite in diskrete Werte um.

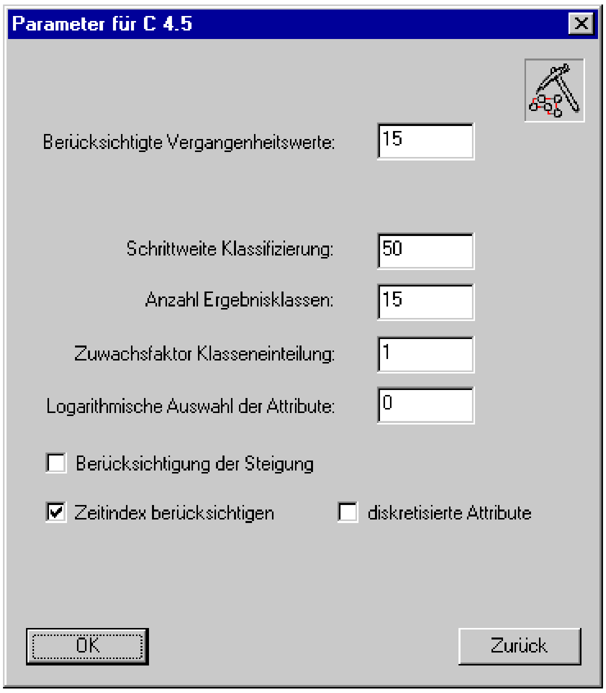


Abbildung 7.4: Dialog für die Parametereinstellungen von C4.5

Analog zu den Einstellungen für C4.5 besitzt der Dialog für die SVM (Abbildung 7.5) Eingabefelder für Historie, Steigung, Zeitindex und Logarithmus. Hinzu kommen Felder für Kapazität und Kernel. Die Kernelparameter lassen sich ebenfalls angeben, werden aber nur von Kerneln benutzt, die diese Werte benötigen. „Loss“ gibt die Gewichtung an, mit der Überschätzen (positiv) und Unterschätzen (negativ) bewertet wird. „Epsilon“ gibt den Bereich der Insensitivität für die Kostenfunktion an. Um hier Dienste mit durchschnittlich niedrigem Angebot nicht anders als die übrigen zu bewerten, gibt Epsilon die relative Insensitivität hinsichtlich des maximalen Angebots des Dienstes an.

Da sich bei den Experimenten mit der SVM herausgestellt hat, dass ein globaler Parametersatz über alle Dienste unzureichende Ergebnisse erzielt, wird dieser Dialog für jeden Dienst einzeln aufgerufen. Aus jedem Dialog kann über die Anwahl von „Zurück“ der vorherige Dialog angesprungen werden.

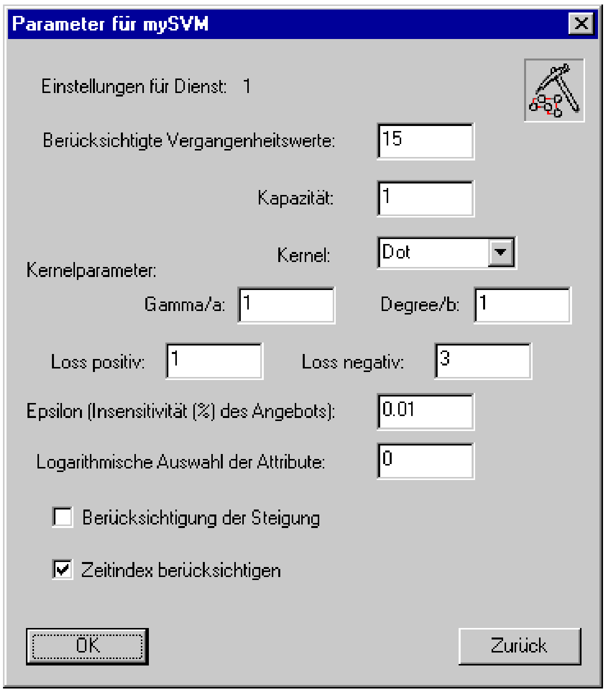


Abbildung 7.5: Dialog für die Parametereinstellungen der SVM

Die Mittelwertbildung besitzt keine besonderen Parameter und kommt daher ohne eigenen Dialog aus.

Nach der Einstellung der Parameter startet Median die Simulation des Netzwerks und überwacht die Blockierungen auf den Verbindungen. Abbildung 7.6 stellt dar, wie sich die Data Mining-Verfahren in den Ablauf von Netzwerkdimensionierung und -optimierung eingliedern.

Nach dem Start befindet sich das System in Zustand 1, in dem die Simulation auf dem ersten von Median generierten logischen Netz arbeitet. Solange noch nicht genügend Angebotsdaten gesammelt werden konnten (entsprechend dem zu Beginn vom Anwender eingestellten Fenster), existieren keine trainierten Daten und es kann keine Vorhersage über zukünftige Angebote getroffen werden. Treten in diesem Zeitraum hohe Blockierungen auf, die zu einer Verletzung der Rekonfigurationszahl führen, kann das Netzwerk nicht rekonfiguriert werden.

Sind genügend Daten über den Verlauf der Angebote gesammelt, tritt das System in Zustand 2 ein und startet einen Lernvorgang über alle Daten des vollständigen Netzwerks. Anschließend wird die Programmausführung wieder an die Simulation zurückgegeben und Median überwacht in Zustand 3 den Betrieb. Überschreitet die Blockierung nicht die Rekonfigurationszahl, verbleibt das System passiv in Zustand 3, andernfalls wird vom Data Mining-Modul in Zustand 4 eine Vorhersage der Angebotswerte innerhalb des Prognosehorizonts angefordert, auf deren Basis die Dimensionierung und Optimierung eines neuen logischen Netzwerks erfolgt.

Die in Zustand 4 erstellten Prognosen werden gespeichert und mit den realen Werten verglichen sobald diese vorliegen. Überschreitet der durchschnittliche Fehler nicht die Grenzwerte, kehrt Median wieder in Zustand 3 zurück und kann die Prognose verwenden, andernfalls geht das System in Zustand 5 über und startet hier für die Links und Dienste, die einen zu großen Fehler aufwiesen, einen neuen Lernvorgang.

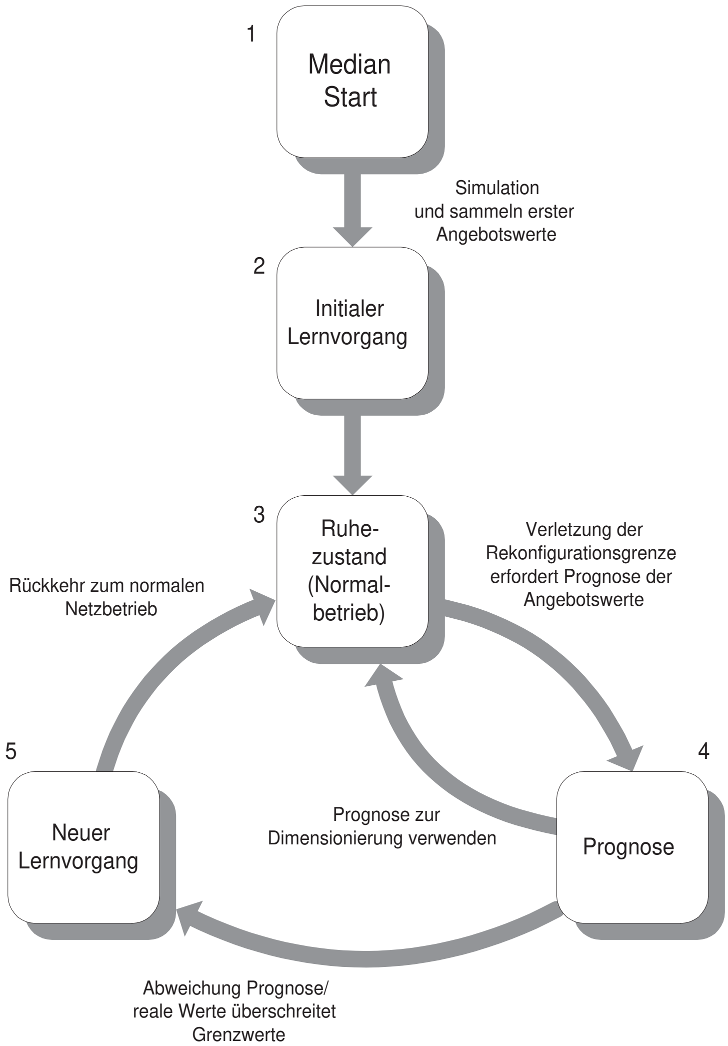


Abbildung 7.6: „Lernen–Vorhersage–Neu Lernen“—Zyklus

Sind die nachgebesserten Modelle gelernt, kehrt Median in Zustand 3 zurück und kontrolliert wieder die Blockierungen im Netz. Zu beachten ist, dass für die Verbindungen, die einen neuen Lernvorgang erforderten, noch die alten mit dem vorherigen Modell erstellten Prognosen vorhanden sind. Um zu vermeiden, dass die Grenzwerte für die Abweichung bereits bei der nächsten Prognose erneut überschritten werden, bleiben die betroffenen Links aus der Überwachung so lange ausgespart, bis genügend Prognosen mit dem neuen Modell erfolgt sind. Somit wird sichergestellt, dass für die folgenden Prognosen nur gemessene Abweichungen berücksichtigt werden, die mit dem neuen, korrigierten Modell erfolgt sind.

7.3 Klassenübersicht

Um Wartung und Pflege für zukünftige Versionen von Median zu erleichtern, erfolgt die modulare Integration der Lernverfahren innerhalb eigener Klassen. Um die Wiederverwertbarkeit der für allgemeine Verwaltungszwecke benötigten Programmteile zu gewährleisten, enthält die Basisklasse *DataMining* die vollständige Implementation der Funktionen, die verfahrensunabhängig sind. Dies betrifft beispielsweise Methoden zur Bestimmung von realen Maxima und Überwachung der Abweichungen zwischen realen und prognostizierten Werten. Die abgeleiteten Klassen implementieren die verfahrensspezifischen Funktionen, wie z.B. die Transformation der Daten in das jeweilige Format anhand der relevanten Parameter. Abbildung 7.7 stellt eine Übersicht der Klassen mit Methoden, Beziehungen und der Vererbungshierarchie dar, aus Platzgründen fehlen die Parameter der einzelnen Methoden.

Im folgenden werden die entsprechenden Methoden und Attribute in ihrer grundlegenden Funktionalität beschrieben:

1. Die einzelnen Attribute der Klasse DataMining (**m_ihistory**[ ], **m_ifuture**[ ], **m_iAnzDienste**...) spiegeln die Parameterauswahl wider, die in den vorherigen Kapiteln für die Transformationen der Daten zum Lernen benutzt wurden. Zu beachten ist, dass Parameter, die auch von der SVM benutzt werden, für jeden Dienst einzeln vorgesehen sind, da sich die Wahl eines globalen Parametersatzes für jeden Dienst als instabil hinsichtlich der Vorhersagequalität erwiesen hat.

2. Attribute mit Zugriffstatus *protected* dienen der internen Verwaltung, z.B. zur Verteilung der Daten an die PVM-Rechner (**RechnerSchlange**) oder zur Fehlerüberwachung (**dabweichungenplus**[ ][ ][ ] und **dabweichungenminus**[ ][ ][ ]).

3. Die Methode **Lernen()** startet den Lernvorgang über alle Dienste und Links. Da dies verfahrensspezifisch erfolgt, wird Lernen() in den von DataMining abgeleiteten Klassen implementiert.

4. Analog dazu enthält **Prognose()** die Vorhersage des Maximums innerhalb des Prognosehorizonts (m_ifuture[ ]) für das gesamte Netzwerk. Die Werte werden an die Dimensionierungs- und Optimierungsmethoden von Median übergeben. Diese Funktion ist ebenfalls verfahrensspezifisch, weshalb die Implementierung in den abgeleiteten Klassen erfolgt.

5. Für die Überwachung der Qualität der Prognose und das erneute Lernen stehen die folgenden Funktionen zur Verfügung:

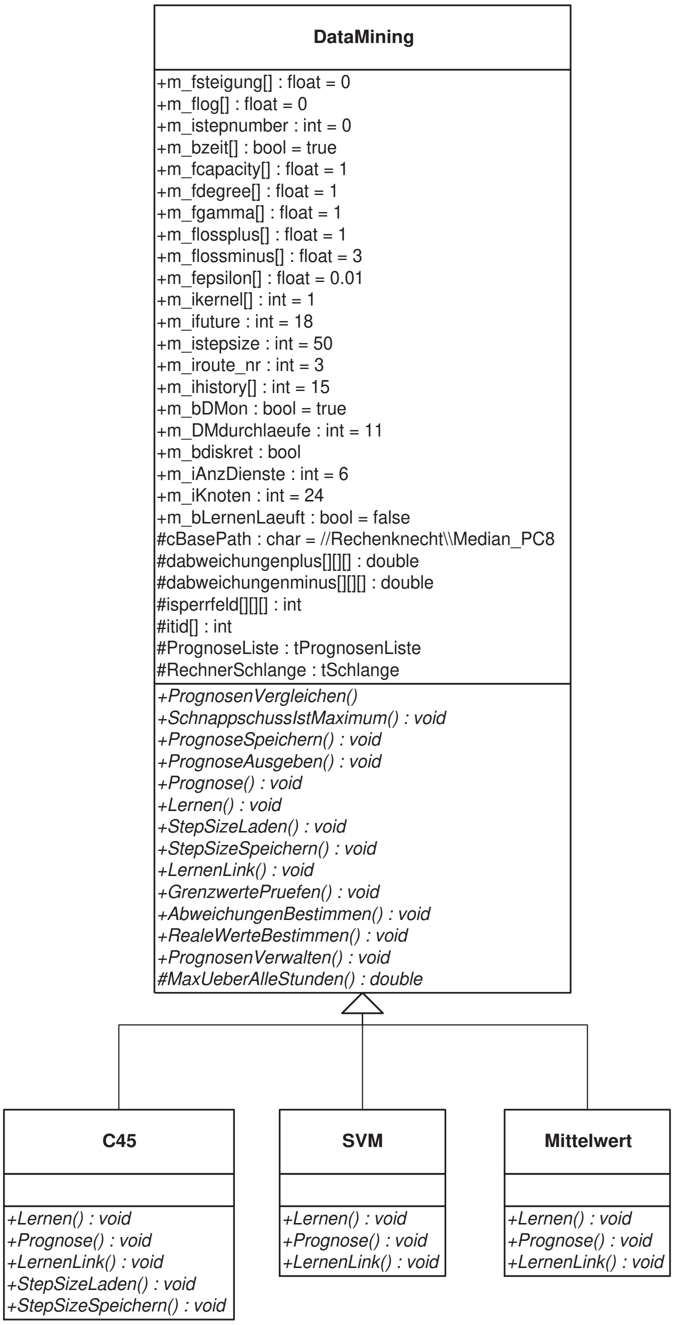


Abbildung 7.7: Klassenübersicht der Data Mining-Verfahren

(a) **PrognosenVerwalten():** Die Überwachung der Vorhersagequalität erfolgt innerhalb eines durch den Anwender festgelegten Fensters. PrognosenVerwalten() übernimmt die Speicherung aktueller Prognosen und löscht veraltete, die aus dem Fenster herausfallen.

(b) **RealeWerteBestimmen():** Zur Qualitätsüberwachung müssen die real aufgetretenen Maxima innerhalb des Prognosehorizonts ermittelt werden. Zum Zeitpunkt der aktuellen Prognose ist das zukünftige Verhalten des Netzes selbstverständlich noch nicht bekannt, so dass diese Methode die Aufgabe hat, für alle gespeicherten Prognosen, die weit genug in der Vergangenheit zurückliegen, die realen Maxima aus dem Netzverhalten seit der betrachteten Prognose zu bestimmen.

(c) **AbweichungenBestimmen():** Befinden sich innerhalb des Überwachungsfensters genügend Prognosen sowie die zugehörigen realen Werte, berechnet diese Methode die Differenz zwischen beiden Werten und daraus den *relativen* Fehler. Der relative Fehler wird hier aus zwei Gründen gewählt: Erstens kann somit eine einheitliche Grenze für die maximale Abweichung über alle Links und Dienste gewählt und muss nicht für jede Verbindung einzeln angegeben werden. Zweitens berücksichtigt der relative Fehler die verhältnismäßige Abweichung der Werte, d.h. über mehrere Zeitpunkte betrachtet, wie eng aneinander prognostizierte und reale Angebotskurve zueinander verlaufen. Da bei Betrachtung der Abweichung einzelner Links und selektivem erneuten Lernen nur auf den Links, deren Fehler den Grenzwert überschreitet, der zeitliche Aufwand verhältnismäßig niedrig ist, kann ein neuer Lernvorgang bereits dann von Vorteil sein, wenn die absolute Abweichung noch keine kritische Größe erreicht hat.

(d) **GrenzwertePruefen():** Sind die Abweichungen im Fenster bestimmt, wird jede Verbindung auf eine Verletzung des durch den Anwender festgelegten Grenzwertes überprüft. Bei Verbindungen, für die die Prognose die Qualitätskriterien nicht mehr erfüllt, wird ein erneuter Lernprozess mit dem gewählten Lernverfahren durchgeführt. Die Verteilung der Aufträge erfolgt wie für den Lernenvorgang über das gesamte Netzwerk dynamisch mittels der Parallel Virtual Machine.

(e) **LernenLink():** Im Gegensatz zu Lernen() startet diese Methode den Lernvorgang lediglich für einen übergebenen Link und Dienst. Sie dient dazu, gezielt einzelne Angebotskurven neu zu Lernen, für welche die Prognose den Grenzwert der Abweichung verletzt hat, und damit den Rechenaufwand nur auf das unbedingt notwendige Maß zu beschränken. Angestoßen wird die Methode aus GrenzwertePruefen() heraus, die Implementierung erfolgt verfahrensspezifisch in der abgeleiteten Klasse des benutzten Data Mining-Verfahrens.

6. In der Klasse DataMining sind allgemeine Methoden implementiert, die unabhängig vom gewählten Verfahren sind und den Lernvorgang, bzw.

die Prognose unterstützen:

(a) **SchnappschussIstMaximum():** Diese Funktion ermittelt das Maximum für einen gewünschten Simulationsdurchlauf und ab einem Zeitindex innerhalb des Prognosehorizonts, was für die Ermittlung der realen Maxima im Netz notwendig ist.

(b) **PrognoseSpeichern():** Mit den Data Mining-Verfahren berechnete Angebotsprognosen können mit dieser Methode dauerhaft für spätere Vergleiche gesichert werden.

(c) **PrognoseAusgeben():** Zu Testzwecken kann hiermit eine Prognose auf dem Bildschirm ausgegeben werden.

(d) **PrognosenVergleichen():** Diese Methode vergleicht zwei Prognosen, bzw. eine Prognose mit den realen Werten, und berechnet verschiedene Abweichungen (relativer/absoluter Fehler, jeweils quadratisch und linear; Varianz; durchschnittliche Fehler über alle Links und Dienste; Gesamtfehler)

(e) **MaxUeberAlleStunden():** Ermittelt das Maximum unabhängig von Zeitindex und Prognosehorizont innerhalb eines vollständigen Durchlaufs. Benötigt wird dieser Wert z.B. für die Klasseneinteilung bei der Benutzung von C4.5. Aus Maximalwert und vorgegebener Klassenanzahl ergibt sich die Schrittweite für die Intervallbildung der Klassen.

7. **StepSizeSpeichern()** und **StepSizeLaden()** sind Funktionen, die nur von C4.5 benötigt und dementsprechend auch nur in dieser Klasse implementiert werden. Da C4.5 als Ergebniswerte lediglich die Nummern der ermittelten Klassen zurückliefert, müssen daraus noch entsprechend der Schrittweiten die jeweiligen Intervalle bestimmt werden. Um die Prognose auch zwischen zwei Programmausführungen zu ermöglichen, kann über die StepSize-Funktionen die Schrittweite persistent im Speicher gehalten werden.

Kapitel 8

Ausblick

Bei der Arbeit mit den verschiedenen Data Mining-Verfahren haben sich deutliche Unterschiede in Handhabung, Genauigkeit und Geschwindigkeit gezeigt.

C4.5 als Beispiel aus dem Bereich der Entscheidungsbäume hat sich besonders durch seine Geschwindigkeit ausgezeichnet. Als einfaches Verfahren ist es leicht einzusetzen, bietet allerdings nur begrenzt Möglichkeiten, durch Wahl geeigneter Parameter an die Anforderungen der Daten angepasst werden zu können.

Wesentlich flexibler, damit aber auch wesentlich schwieriger in der Vorbereitungsphase, war die Support Vector Machine einzusetzen. Die vielfältigen Parameter machen die Suche optimaler Einstellungen äußerst aufwändig. Gerade an dieser Stelle (z.B. in Bezug zur Auswahl des Kernels direkt anhand der Daten ohne Trainings/Test-Zyklen) besteht enormer Bedarf und dieser Bereich ist damit auch Gegenstand aktueller Forschungen. Wie die Experimente gezeigt haben, reagiert die SVM damit empfindlich auf die Daten, qualitativ gute Ergebnisse lassen sich so nur erzielen, wenn für jeden Dienst getrennte Parameterauswahlen getroffen werden. Stabil hinsichtlich der Kriterien für das Loss erweist sie sich dagegen bei steigendem Angebot und Wochentagen.

Als problematisch hat sich zusätzlich der stark schwankende Zeitaufwand für Training der SVM erwiesen. Rechenzeiten zwischen wenigen Sekunden und im Bereich von Minuten sind bereits bei geringen Änderungen der Parameter aufgetreten. C4.5 dagegen ändert seine Rechenzeit im wesentlichen nur in Abhängigkeit der Anzahl der Attribute.

Sehr viel Zeit im Vorfeld der Integration der DM-Verfahren haben die Untersuchungen für gute Repräsentationen der Simulationsdaten benötigt. Trotzdem sind die Möglichkeiten in diesem Bereich keinesfalls voll ausgeschöpft, grundsätzlich sind unzählige weitere Transformationen denkbar, die mit den vom Simulator bereitgestellten Daten nicht durchführbar waren. Diesem Umstand zufolge ist die Einführung von Wochentagen entstanden. Interessante Möglichkeiten ergeben sich beispielsweise dann, wenn nicht nur reine Angebotsdaten der Zeitreihe vorliegen, sondern weitere mögliche Attribute gefunden werden können.

Vorstellbar sind weitere Kenngrößen des Netzwerks, z.B. auf physikalischer Ebene (Latenzzeiten bei der Paketvermittlung, Auslastung von Zwischenspeichern...), aber auch von Faktoren, die nicht offensichtlich Einfluss auf das Netzverhalten nehmen. Dies könnten große Veranstaltungen, Fernsehsendungen oder das Wetter sein. Im Bereich der Bewertung des Stromverbrauchs sind solche Zusammenhänge bereits länger bekannt. Problem hierbei ist wieder, wie die möglichen Attribute ausgewählt und mit den großen Datenmengen umgegangen werden kann.

Als Ergebnis dieser Diplomarbeit hat sich Median als vollständig geschlossenes System zum dynamischen Management von ATM-Netzwerken ergeben. Damit lässt sich die Machbarkeit solcher Management-Systeme zeigen. Durch die Bewertung der Dienste anhand ihrer mittleren Bitrate synonym zu Kosten, lässt sich das Netzmanagement auch nach betriebswirtschaftlichen Kriterien durchführen. Hierzu gehört ebenfalls die asymmetrische Bewertung von Unter- und Überschätzen, die es ermöglicht, Leitungsreserven zur Kundenzufriedenheit einer höheren Auslastung gegenüber vorzuziehen.

Ein weiterer wichtiger Punkt betrifft die Fähigkeit, sich adaptiv an Veränderungen im Angebot anzupassen. Die Prognosen sind nicht auf die im ersten Lernzyklus gewonnenen Erkenntnisse festgelegt, sondern werden bei Bedarf in weiteren Zyklen angepasst und verbessert. Die Kapselung des jeweiligen Programmcodes in logisch getrennte Einheiten erleichtert das Verständnis und die Wiederverwertbarkeit.

An dieser Stelle sei nochmals die Berücksichtigung der Zeit erwähnt: Die Verfahren selbst waren ursprünglich nicht für die Zeitreihenanalyse vorgesehen, weswegen die Zeit in die Repräsentation der Daten mit eingearbeitet werden musste. Bei den durchgeführten Experimenten hat sich gezeigt, dass dieser Ansatz mit Erfolg realisierbar ist und damit aufwändigere Zeitreihenverfahren aus der Statistik ersetzen kann.

Der aktuelle Stand von Median spiegelt die Fähigkeit zum dynamischen, semi-automatischen Management wider und dementsprechend könnten zukünftige Entwicklungen von Median auf die voll-automatisierte Steuerung ohne Nutzereingriffe, wie die Voranalyse optimaler Parametereinstellungen, abzielen. Der Prognoseprozess erfordert dann neben der permanenten Qualitätsüberwachung die dynamische Anpassung der Parameter oder auch des Zeitfensters, über das gelernt werden soll. Diese Vorgehensweise stellt neue Anforderungen an Rechenleistung und Verfahren, müssen doch viele Einstellungen auf ihre Tauglichkeit hin parallel abgeschätzt werden. Helfen können an dieser Stelle neue Methoden, die einfache Schätzungen optimaler Parameter, wie z.B. für die Wahl des Kernels bei der SVM, anhand schneller Analysen der Daten ermöglichen.

Denkbar ist auch die in Abbildung 2.3 auf Seite 15 getrennt dargestellten Phasen Data Mining zur Lastvorhersage, Dimensionierung und Optimierung in eine Phase zu integrieren, die mit Techniken aus der künstlichen Intelligenz arbeitet. Der Lernprozess verlagert sich dann von der einfachen Zeitreihenprognose zur direkten Abbildung des Netzverhaltens auf ein neues Netz. Dadurch ließen sich u.U. Geschwindigkeitsvorteile erzielen, allerdings stiege die Komplexität entsprechend an, so dass die Auswahl und Pflege der benutzten Verfahren nochmals an Bedeutung gewinnt.

Abschließend betrachtet erfüllt Median die grundlegenden Anforderungen an das dynamische Netzwerkmanagement in Echtzeit. Die Experimente haben gezeigt, dass sich mittels intelligentem Management die Auslastungen verbessern lassen und die Notwendigkeit zum Einsatz neuer Hardware und Überdimensionierung der Leitungskapazitäten im Netz reduziert werden. Die Ergebnisse sind selbstverständlich deutlich unter den optimal zu erreichenden anzusiedeln, die vollständiges und fehlerfreies Wissen über das zukünftige Netzverhalten voraussetzen, wie es in der Realität durch Prognosefehler, Zeitbeschränkungen usw. nicht vorliegt. Ziel ist dementsprechend auch nicht das bis ins letzte Detail perfekte Management des Netzes, sondern die im Gegensatz zum nicht oder lediglich manuell gesteuerten Betrieb verbesserte Auslastung und Entdeckung ungenutzter Potenziale bei gleichzeitiger Berücksichtigung der größtmöglichen Erreichbarkeit der Teilnehmer. Der Weg dahin ist noch lange nicht vollständig, aber Median kann einen Beitrag zur weiteren Entwicklung leisten.

Anhang A

Installation und Betrieb

In diesem Kapitel des Anhangs werden einige Besonderheiten und Programme beschrieben, die im Verlauf der Arbeit mit Median bekannt geworden, bzw. entstanden und noch nicht in anderen Diplomarbeiten dokumentiert oder vorhanden sind. Sie dienen als Werkzeuge für weitere Arbeiten am Projekt Median und können helfen, in einigen Fällen Arbeit und unnötigen Aufwand zu ersparen.

A.1 Stabilität

Die Simulation ATMSim stellt gewisse Bedingungen an die Umgebung um einen stabilen Betrieb zu gewährleisten. Insbesondere sind folgende Punkte zu beachten:

1. Ein gemeinsamer Betrieb von Simulation und dem zu Visual Studio gehörenden Debugger erfordert die Installation des Service Packs 4. Ohne das SP kann es zu Problemen beim Anspringen von Break-Points kommen.

2. In den Speicherüberwachungsfunktionen von Visual Studio, die in einer Debug-Version zum Einsatz kommen, befindet sich ein Fehler, der nach einiger Laufzeit der Simulation zum Überlauf eines Zählers führt, was einen internen Break-Point auslöst. Um diesen Effekt zu vermeiden, muss direkt die Datei „CRTdbgheap.c“ im Bezug zum Zähler „_CRTAllocHeap“ korrigiert werden. Dieser Fehler tritt in der Release-Version nicht auf, so dass die Verwendung dieser Version im Normalbetrieb empfohlen wird.

3. Bei Verwendung von Windows 2000 kann es zu Problemen bei der Speicherverwaltung durch einen neuen Heap-Manager kommen. Dieser muss durch Verwendung des bei den Windows-Support-Tools vorhandenen Programms „ApCompat.exe“ abgeschaltet werden.

4. Die Support Vector Machine mySVM arbeitet nicht unter Windows 2000, wenn sie mittels der PVM als Remote-Prozess gestartet wird. Unter Windows NT 4 besteht dieses Problem nicht. Die übrigen Verfahren arbeiten auf beiden Betriebssystemen stabil.

A.2 Hilfsprogramme

Hier werden die zusätzlich im Rahmen der Diplomarbeit entstandenen Programme dokumentiert, was sowohl die Programme zur Datentransformation mit den jeweiligen Parametern als auch die Skripte zur Manipulation der Daten außerhalb Medians betrifft.

Für die Umwandlung der Ausgaben der Simulation in die für die Data Mining-Verfahren geeigneten Repräsentationen stehen die folgende Programme zur Verfügung:

1. *Transform:* Wandelt die Daten in ein für C4.5 lesbares Format um.

2. *TransformSVM:* Ermöglicht den Trainingsvorgang durch mySVM.

3. *TransformMW:* Bildet den durchschnittlichen Mittelwert auf den Daten.

Die möglichen Parameter (Prognosehorizont, Dienst, Anzahl der Routen...) werden jeweils angezeigt, wenn das Programm ohne Parameter aufgerufen wird.

Für die Vorhersage von Klassen durch C4.5 steht standardmäßig nur das interaktive Programm *consult* zur Verfügung, das Werte für einzelne Attribute vom Anwender erfragt und sich damit durch den Entscheidungsbaum zur Zielklasse bewegt. Um die automatisierte Vorhersage zu ermöglichen, ist eine modifizierte Version *consultm* entstanden, die einen zusätzlichen Parameter „-c“ besitzt. Als Wert für den Parameter wird die Datei angegeben, die Werte für die Attribute des zu klassifizierenden Falles enthält.

Neben den oben angegebenen Transformationsprogrammen müssen sich für den Betrieb von Median noch die folgenden Programme im Verzeichnis *Programme* des Median-Hauptverzeichnisses befinden: *C4.5.exe, consultm.exe, mysvm.exe, predict.exe.*

Für die Aufnahme der Prognosedateien wird in das Median-Hauptverzeichnis ein weiteres Verzeichnis *Prognosedaten* eingefügt. Darin müssen sich weitere Verzeichnisse *Prog_0* bis *Prog_23*, die die Prognose für jeden Knoten aufnehmen, sowie *Vergleiche*, das die Daten zur Überwachung der Prognosequalität enthält, befinden.

Die Slave-Programme *DM_01s.exe* bis *DM_06s.exe* müssen sich im *bin*-Verzeichnis der jeweiligen PVM-Installation befinden.

Neben den Programmen, die für den Betrieb mit Median zwingend notwendig sind und die aus Geschwindigkeitsgründen in *C* programmiert wurden, existieren weitere Hilfsprogramme in *Perl*, die überwiegend zur Arbeit auf den Ausgaben der Simulation dienen, z.B. zur Extraktion von Testdurchläufen. An dieser Stelle werden die Skripte kurz erwähnt, um bei späteren Arbeiten mit Median die Wiederverwertung zu ermöglichen und damit die Manipulation der Daten zu erleichtern. Die Funktion der einzelnen Programme ist kurz im Quelltext erläutert und die meisten Programme geben die Aufrufsyntax an, wenn sie ohne Parameter gestartet werden.

1. *Extract.pl:* Schneidet den angegebenen Durchlauf zwischen zwei Zeitindizes aus den logOutput-Dateien im aktuellen Verzeichnis aus und legt

den extrahierten Durchlauf in separaten Dateien ab. Die extrahierten Durchläufe können dann z.B. als Testdaten verwendet werden.

2. *ExtractMulti.pl:* arbeitet wie Extract.pl, schneidet aber mehrere aufeinander folgende Durchläufe aus.

3. *Concat.pl:* hängt beliebige Zeilen aus den zuvor extrahierten Dateien an die logOutput-Dateien an, z.B. um Prognosen für die extrahierten Durchläufe und Zeitindizes zu ermöglichen.

4. *DeleteLine.pl:* löscht eine Zeile (abhängig vom übergebenen Durchlauf und Zeitindex) aus allen logOutput-Dateien im aktuellen Verzeichnis.

5. *DeleteLineMulti.pl:* wie DeleteLine.pl, aber für mehrere Zeile hintereinander.

6. *BatchSVM.pl:* führt mehrere Lern-/Testdurchläufe mit mySVM hintereinander aus und speichert die Ausgaben in Textdateien zur späteren Auswertung. Damit lassen sich beispielsweise die Auswirkungen der Wahl des Kernels bestimmen.

7. *ReportSVMlog.pl:* fasst die wichtigsten Kenngrößen aus den mit BatchSVM.pl erstellten Ausgaben in einer Datei zusammen (Loss, Iterationszahl...).

8. *BatchC45.pl:* führt mehrere Lern-/Testdurchläufe für unterschiedliche Transformationen mit C4.5 aus.

9. *AcrossLinks.pl:* In Kapitel 4 wurde als Datenrepräsentation für das Lernverfahren nicht nur die Auswahl der Attribute entlang der Zeitreihe, sondern über mehrere Links vorgeschlagen. Da sich diese Art der Bereitstellung der Attribute wesentlich von derjenigen entlang der Zeitachse unterscheidet, wurde diese Möglichkeit nicht als Parameter in die Transformationsprogramme integriert, sondern ein eigenes Skript AcrossLinks.pl geschaffen.

10. *ReportVergleicheAbs.pl:* extrahiert den absoluten quadratischen Fehler aus den Vergleichsdateien zu jedem DM-Verfahren und berechnet den Gesamtfehler/Fehler je Dienst entsprechend der Gewichtung in Abhängigkeit der mittleren Bitrate.

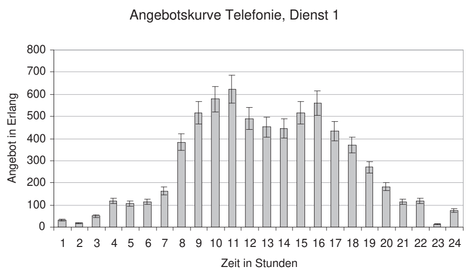
11. *Resort.pl:* Bei Simulationen mit steigendem Angebot sortiert Median die Durchläufe so, dass auf zwei Durchläufe mit normalem Angebot (1) ein Durchlauf (2) mit dem ersten Steigerungsfaktor und ein Durchlauf (3) mit dem zweiten Steigerungsfaktor folgen (1-1-2-3-1-1-2-3...). Um daraus ein insgesamt monoton steigendes Angebot entlang der Zeitachse zu erzeugen, sortiert Resort.pl die Durchläufe so um, dass zuerst alle normalen Angebote, dann die ersten Steigerungen und abschließend die zweite Steigerungsstufe auftreten (1-1-1-...2-2-2-...3-3-3-...).

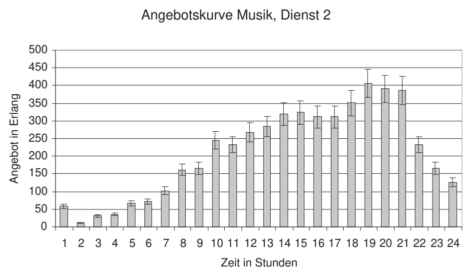
12. *AddNoise.pl:* fügt den logOutput-Dateien künstliches Rauschen hinzu, indem in wählbaren Schritten Ausreißer in einstellbarer Stärke hinzugefügt werden.

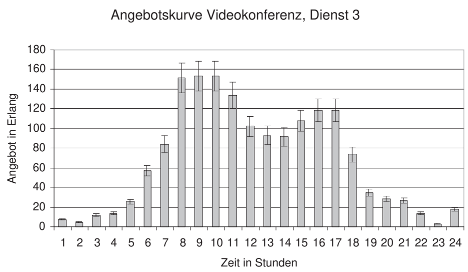
13. *Weekdays.pl:* Der Simulator beherrscht in seiner aktuellen Version lediglich die Optionen, für jeden Durchlauf dieselben zugrundeliegenden Angebotskurven zu benutzen oder eine Steigerung in bis zu zwei Schritten zu integrieren. Um eine weitere Annäherung an die Realität zu ermöglichen, können mit diesem Programm bestehende logOutput-Dateien so verändert werden, dass sie unterschiedliche Angebotskurven in Abhängigkeit der Durchlaufnummer (im Sinne von Wochentagen) erzeugen. Die Kurven sehen z.B. so aus, dass an Samstagen der durchschnittliche Konsum des Dienstes Fernsehen ansteigt und sich zeitmäßig weiter in die späten Abendstunden verlagert.

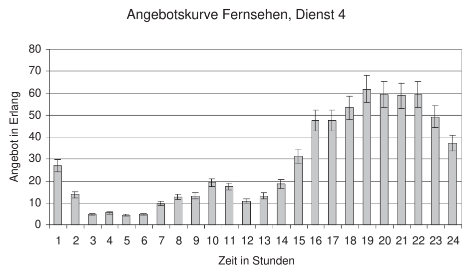
Anhang B

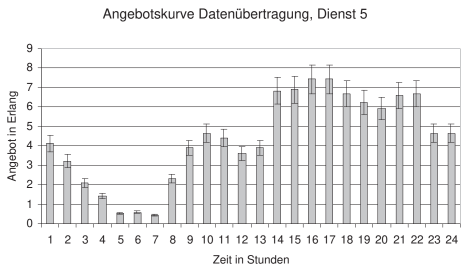
Angebotskurven der Dienste

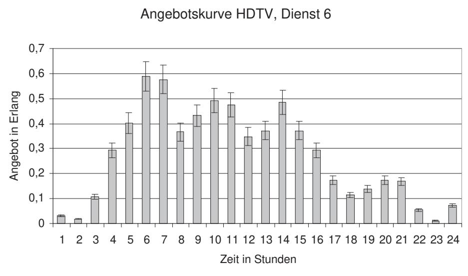












Literaturverzeichnis

[Bindrim00] W. Bindrim: *Teletraffic Theory meets Data Mining* in NOMS 2000: The Networked Planet: Management Beyond 2000, S. 997–998, IEEE Operations Center 2000, Piscataway New Jersey

[Borgelt98] C. Borgelt et al.: *Lernen probabilistischer und possibilistischer Netze aus Daten: Theorie und Anwendung* in Künstliche Intelligenz 1/98, S. 11–77

[Burges98] C. Burges: *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery Volume 2/No. 2, 1998

[Cohen95] P. Cohen: *Empirical Methods for Artificial Intelligence*, The MIT Press 1995, Cambridge Massachusetts

[Crisp99] P. Chapman et al.: *The CRISP-DM Process Model*, The CRISP-DM consortium, 1999

[Cygwin] Cygwin: *gcc-Implementierung für Windows*, Red Hat, http://sources.redhat.com/cygwin/

[Domingos99] P. Domingos: *Data Pre-Processing for Cost-Sensitive Learning*, Proceedings of the ICML-99 Workshop on: From Machine Learning to Knowldege Discovery in Databases

[Fayyad96] U. Fayyad et al.: *Advances in Knowledge Discovery and Data Mining*, A Bradford Book, The MIT Press 1996, Cambridge Massachusetts

[Glow00] A. Glow, B. Rahali: *Diplomarbeit: Implementierung von Verfahren zur Dimensionierung von ATM-Netzwerken unter Berücksichtigung paralleler Prozesse*, Universität Dortmund 2000, Lehrstuhl für Elektronische Systeme und Vermittlungstechnik

[Holsheimer94] M. Holsheimer, A.P.J.M. Siebes: *Data Mining: The search for knowledge in databases*, Centrum voor Wiskunde en Informatica 1994, Amsterdam

[Hyafil76] L. Hyafil, R.L. Rivest: *Constructing optimal binary decision trees is NP-complete*, Information Processing Letters 5, 1, 15–17, 1976

[Kietz00] J.-U. Kietz et al.: *Mining Mart: Combining Case-Based-Reasoning and Multi-Strategy Learning into a Framework to reuse KDD-Application* in Proceedings of the fifth International Workshop on Multistrategy Learning 2000

[Krahl98] Daniela Krahl et al.: *Data Mining - Einsatz in der Praxis*, Addison-Wesley 1998, Bonn

[Kyas96] Othmar Kyas: *ATM-Netzwerke, Aufbau–Funktion–Performance*, DATACOM-Buchverlag 1996, Bergheim

[Morik96] K. Morik, P. Brockhausen: *Direct Access of an ILP Algorithm to a Database Management System* in Data Mining with Inductive Logic Programming (ILP for KDD), S. 95–110, 1996

[mySVM] mySVM: *Support Vector Machine*, Lehrstuhl für Künstliche Intelligenz, Universität Dortmund, http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/

[Pyle99] D. Pyle: *Data Preparation for Data Mining*, Morgan Kaufmann Publishers 1999, San Francisco CA

[Quinlan92] J. R. Quinlan: *C4.5: Programs for machine learning*, Morgan Kaufmann Publishers 1992, San Mateo (USA)

[Rüping99] S. Rüping: *Diplomarbeit: Zeitreihenprognose für Warenwirtschaftssysteme unter Berücksichtigung asymmetrischer Kostenfunktionen*, Universität Dortmund 1999, Lehrstuhl für Künstliche Intelligenz

[Scheer99] R. Scheer, C. Steinweg: *Studienarbeit: Entwicklung einer Oberfläche und Implementierung von speziellen Verfahren zur Erzeugung und Dimensionierung von physikalischen ATM-Kernnetzen*, Universität Dortmund 1999, Lehrstuhl für Elektronische Systeme und Vermittlungstechnik

[Scheer00] R. Scheer, C. Steinweg: *Diplomarbeit: Erstellung einer Windowsapplikation mit Datenbankanbindung zur parallelen Dimensionierung und Optimierung von ATM-Kernnetzwerken*, Universität Dortmund 1999, Lehrstuhl für Elektronische Systeme und Vermittlungstechnik

[Schehrer99] R. Schehrer: *Vermittlungssysteme I/II*, Universität Dortmund 1999

[Schlittgen97] R. Schlittgen, B. Streitberg: *Zeitreihenanalyse*, Oldenbourg Verlag 1997, München

[Tegtmeier99] G. Tegtmeier: *Diplomarbeit: Erstellung einer Simulationssoftware für ATM-Kernnetze zur Realisierung von VP-Rekonfigurationen während des Betriebes*, Universität Dortmund 1999, Lehrstuhl für Elektronische Systeme und Vermittlungstechnik

[Vapnik98] V. Vapnik: *Statistical Learning Theory*, Wiley 1998

[Wettschereck92] D. Wettschereck, T.G. Dietterich: *Improving the performance of Radial Basis Function Networks by Learning Center Locations* in Neural Processing Systems, Morgan Kaufmann 1992

[Wüthrich95] Beat Wüthrich: *Knowledge Discovery in Databases*, University of Science and Technology 1995, Kowloon Hong Kong

Index

Angebot, 9
Ankunftsprozess, 7
Anova, 63
Anrufrate, 8
ATM-Adaption-Layer, 11
ATMSim, 18
Attribut, 28

Bündelungsgewinn, 19
Backbone, 11
Binärisierung, 84
Bitrate, 10
Blatt, 29
Blockierung, 16
Burstfaktor, 10

Chiotis, 75
Chunk, 70
Cluster, 87

Data Mining-Verfahren
- beispiel-basiert, 24
- Entscheidungsbäume, 24
- graphisch, 24
- nichtlineare, 24
- Regeln, 24
- relational, 24

Dialog
- C4.5, 91
- SVM, 92
- Verfahrensauswahl, 90

Dienst, 7
Dienstintegrierend, 7
Direktmailing, 20
Dot, 63
Durchschnittliche Bitrate, 75

Empirisches Risiko, 57
Endeprozess, 7
Entropie, 31
Entscheidungsbaum, Def., 28
Entscheidungsknoten, 29
Erlang, Einheit, 10
Erlangsche Verlustformel, 9
Erwartetes Risiko, 57

Feature Selection, 25
Fehlerabschätzung, 34

gain ratio(), 32
gain(), 31
Generalisierung, 21
Gewichtung, 75

Hauptverkehrsstunde, 8
Hop, 11
Hyperebene, 58

info(), 31
Informationsgehalt, 31
Insensitivität, 62

Kapazität, 61
KDD, 20
Kernel, 63
KKT-Bedingungen, 61, 62
Klasse, 22
Klassenbeschreibung, 22

Lagrange-Darstellung, 60, 61
Lernen
- überwacht, 23
- unüberwacht, 23

Link, 11
Loss, 57

Master, 87
Mercer-Bedingung, 63
Mittelwert, 74
Modell, 21
MPP, 87

mySVM, 64

Neuronales Netz, 63

Overfitting, 33

Polynomiell, 63
Pruning, 33

Radial, 63
Rauschen, 25, 35
Rechenzeit, 89
Rekonfiguration, 12
Rekonfigurationszahl, 16
Risiko, 57
Routing, 11, 12
Ruf, 8

Sampling, 25
Simulationsausgaben, 36
Slave, 87
Slot, 13
Speed Up, 88
Split, 30
splitinfo(), 32
Strukturelle Risikominimierung, 58
Support Vektor, 60
System, 21

Teilnehmer, 7
Timeout, 89
Trainingsdaten, 22
Transition, 21

Universum, 22

VC Dimension, Def., 58
VC Konfidenz, 57
Verbindungsdauer, 8
Verbindungswunsch, 8
Verkehr, 9
Verkehrsrest, 10
Verlust, 9
Virtual Channel Network, 12
Virtual Path Network, 12

Wurzel, 29

Zeitschlitz, 13
Zufallsprozess, 7
Zustand, 21
Zustandsprozess, 7