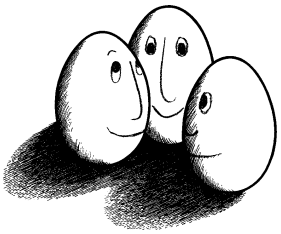
Diplomarbeit

Plant Asset Management unterstützt durch empirische Datenanalyse

Andrej Pawlenko



Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

13. Oktober 2005

Betreuer:

Prof. Dr. Katharina Morik
Dipl.-Inform. Ingo Mierswa

Die sich heutzutage im Einsatz befindenden *Plant Asset Management*-Systeme (PAM-Systeme) dienen zur Verwaltung von Feldgeräten. PAM-Systeme erfassen die Arbeitsparameter der Feldgeräte und sind in der Lage, die erfassten Daten nach Kundenwünschen auszuwerten. Ein nächster Schritt ist es, die PAM-Systeme um Prognosemodule zu erweitern, damit sie in der Lage sind, nicht nur die Ist-Zustände der Feldgeräte auszuwerten, sondern auch das Verhalten der Feldgeräte in der Zukunft zu prognostizieren. Im Rahmen dieser Arbeit werden maschinelle Lernverfahren analysiert. Aufbauend auf diesen Verfahren wird ein Prognosemodul für das PAM-System entworfen, implementiert und eingesetzt. Anschließend wird empirisch nachgewiesen, dass das Prognosemodul zuverlässige Ergebnisse liefert.

Zunächst möchte ich Katharina Morik und Serhiy Mordvinov für die Ermöglichung dieser Arbeit danken. Weiterhin danke ich Ingo Mierswa für die große Unterstützung, hilfreiche Kommentare und insgesamt großartige Betreuung. Anschließend geht mein besonderer Dank an meine Eltern, die mir die wunderbare Arbeitsumgebung geschaffen haben, und – ein ganz besonderer – an Evgeniya Slozina, die die Sprachkorrektur dieser Arbeit durchgeführt hat.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Zum Erfolg eines Unternehmens führen verschiedene Wege. Guter Name, Qualität, günstiger Preis der Produkte und Kundenzufriedenheit sind einige Beispiele dafür. Kundenzufriedenheit spielt dabei eine wesentliche, wenn nicht die wichtigste Rolle, denn zentraler Erfolgsfaktor aller Industrieunternehmen wird in Zukunft die Fähigkeit sein, sich dauerhaft durch qualitativ hochwertige Dienstleistungen von Wettbewerbern zu differenzieren.

Eine Untersuchung [23] zeigte mögliche Gründe auf, warum Unternehmen Kunden verlieren. Bei 68 Prozent der Befragten waren es Unzufriedenheit mit dem Service und dem Unternehmensdialog. Unzufrieden mit dem Produkt bzw. dem Preis waren nur 15 Prozent und auch alle anderen genannten Gründe – wie Abwerbung durch die Konkurrenz oder freundschaftliche Beziehungen zu anderen Firmen – waren wesentlich weniger wichtig.

Um die Kundenzufriedenheit zu gewährleisten, reicht es nicht aus, nur dafür zu sorgen, dass die Qualität bzw. der Preis des Produktes besser bzw. niedriger als bei den Wettbewerbern ist. Vielmehr ist es wichtig, auch produktbegleitende Leistungen (Kundendienstleistungen) zu erbringen, und zwar nicht nur vor, sondern auch nachdem der Kunde ein Produkt gekauft hat (*after sales service*).

Ein Industieunternehmen wurde gegründet und beginnt mit seiner Produktion. Es wächst zusammen mit steigender Anzahl seiner Produkte, Varianten der Produkte und Kunden, die diese Produkte kaufen und einsetzen. Erfolgreiche Hersteller werden den Übergang von Produkt- zur vollständigen Kundenorientierung vollziehen. Bei sämtlichen Handlungen gilt als oberster Grundsatz, die Bedürfnisse der Kunden in den Mittelpunkt zu stellen. Spätestens dann ergibt sich die Notwendigkeit des *after sales service*.

Die durchdachte *after sales*-Politik verschafft den Kunden einen Zusatznutzen. Das anbietende Unternehmen kann damit unterschiedliche Ziele verfolgen, die jedoch alle in einer Verbesserung von Umsatz und Gewinn münden [13]:

- Kundendienstumsatz steigern (durch den Verkauf von Kundendienstleistungen)
- Nutzung / Betrieb von Produkten sichern
- Präferenzen für Unternehmen und Produkte schaffen
- Kundenbindung aufbauen (Nachkauf / Inanspruchnahme weiterer Produkte, wie z.B. Hard- und Softwarekomponenten für die *after sales*-Leistungen)

Eine mögliche Lösung dieser Aufgabe ist ein anlagennahes Asset Management System (*Plant Asset Management system* oder PAM-System).

1.1 *Plant Asset Management*

Unter *asset management* werden Tätigkeiten und Maßnahmen verstanden, die dazu dienen, den Wert einer Anlage zu erhalten oder zu steigern. *Plant Asset Management*, also die werterhaltende und wertsteigernde Instandhaltung [22], wird inzwischen als kritische Komponente eines jeden Echtzeit-Performance-Management-Konzepts gesehen [4]. Um hier erfolgreich zu sein, muss ein verfahrenstechnisch arbeitender Hersteller (und nicht nur dieser) heute *asset management* als aktiven Geschäftsprozess behandeln, und zwar mit einer Vision und einer Zielsetzung, die seiner Geschäftsstrategie entsprechen [4].

Plant Asset Management beginnt mit Zustandserkennung und Diagnose von Feldgeräten (*field devices*) und verfahrenstechnischen Anlagenkomponenten. Zustandserkennung von Feldgeräten beginnt mit der Fähigkeit zur Selbstdiagnose; sie kann erweitert werden durch übergeordnete *asset management*-Systeme. Beides wird heute kommerziell angeboten. In dem Maße, in dem heute bei Feldgeräten die Möglichkeiten zunehmen, neben der Selbstdiagnose im engeren Sinne (Gerätefehler) auch Anwendungsfehler diagnostizieren zu können, wächst bei Anbietern und Nutzern die Bereitschaft, als nächste Stufe die integrierte Zustandserkennung von verfahrenstechnischen Anlagenkomponenten in Betracht zu ziehen. Wie weit dazu Intelligenz vor Ort als integrierter Teil derartiger Komponenten von deren Anbietern (optional) mitgeliefert wird, ob neue Messstellen und neue Sensoren als standardisierte Diagnose-Pakete von den gleichen oder anderen Komponentenanbietern zur Verfügung gestellt werden können oder von der technischen Betriebsbetreuung eingerichtet und installiert werden, oder ob Diagnose anhand vorhandener, heute noch nicht verwerteter Informationen in den Leitsystemen oder in *asset management*-Systemen erfolgt, ist heute im Fluss. Der eine oder andere Anbieter, Nutzer oder Instandhalter bereitet auch die standardisierte Zustandsüberwachung von Prozessabschnitten oder ganzen Prozessen vor, wobei die Situation bei größeren Anlagenkomponenten wie z.B. Wärmetauschern, Turbinen oder Kesseln schon fortgeschritten ist. Und vereinzelt wird schon an die Verknüpfung derartiger Überwachungs- und Diagnose-Fähigkeiten mit den Planning & Scheduling-Systemen der Leittechnik gedacht, an Energiemanagement oder eine wirtschaftliche Prozessbewertung oder gar an eine Prozessoptimierung auf Basis von in Echtzeit verfügbaren Prozess- und Diagnosedaten, von *key performance indicators* [1].

Die heutzutage angebotenen Softwareprodukte im Bereich *Plant Asset Management* dienen also zur Konfiguration, Diagnose, Verwaltung und Optimierung von intelligenten Feldgeräten und Komponenten. Damit verfügt der Anwender über ein offenes Werkzeug für vielfältige Aufgaben von der Vor-Ort-Parametrisierung bis hin zum anlagennahen Asset Management, das über den gesamten Lebenszyklus der Anlage hinweg jeweils die aktuell benötigten Informationen zu den eingesetzten Geräten liefert.

1.2 Kundenbezogenes *Plant Asset Management*-System

Im Rahmen der Vorbereitung zur Diplomarbeit wurde ein kundenbezogenes *Plant Asset Management*-System (PAM-System) entwickelt. Das System umfasst Kunden- und Produktverwaltung. Produkte sind intelligente Feldgeräte, deren Zustand zu jeder Zeit über entsprechende Schnittstellen ausgelesen werden kann. Der Zustand der Feldgeräte hängt in erster Linie von ihrer Leistung ab, die keinesfalls unter den in der Spezifikation des Gerätes vorgegebenen Wert sinken darf. Solche Unterschreitung der Leistung muss sofortigen Geräteaustausch nach sich ziehen.

1.3 Problemstellung

Zum guten *after sales service* gehört außer Hotline und Vor-Ort-Einsätze auch präventive Wartung des Produktes. Es genügt zum Beispiel nicht, festzustellen, dass ein Feldgerät in einer Anlage ausgefallen ist und ersetzt werden muss – viel besser wäre es, vorhersagen zu können,

wann ein Feldgerät ausfallen könnte. Erst dann wird es möglich sein, das Gerät in dem zur Ersetzung vorgesehen Zeitabschnitt rechtzeitig auszutauschen, und nicht erst dann, wenn durch das versagte Feldgerät der Produktionsbetrieb des Kunden beeinträchtigt wird.

Der Punkt Prognose ist aber bis jetzt unberücksichtigt geblieben, wobei er oft als wichtiger Bestandteil eines PAM-Systems gesehen wird. Das System sollte mit Hilfe bereits vorhandener Messungen in der Lage sein, das zukünftige Verhalten der Feldgeräte vorauszusagen.

Es wurde festgestellt, dass die Feldgeräte nicht willkürlich und sporadisch ausfallen, sondern dass ein Zusammenhang zwischen dem Verhalten und Versagen eines Feldgerätes existiert. Daher ist Vorhersage möglich.

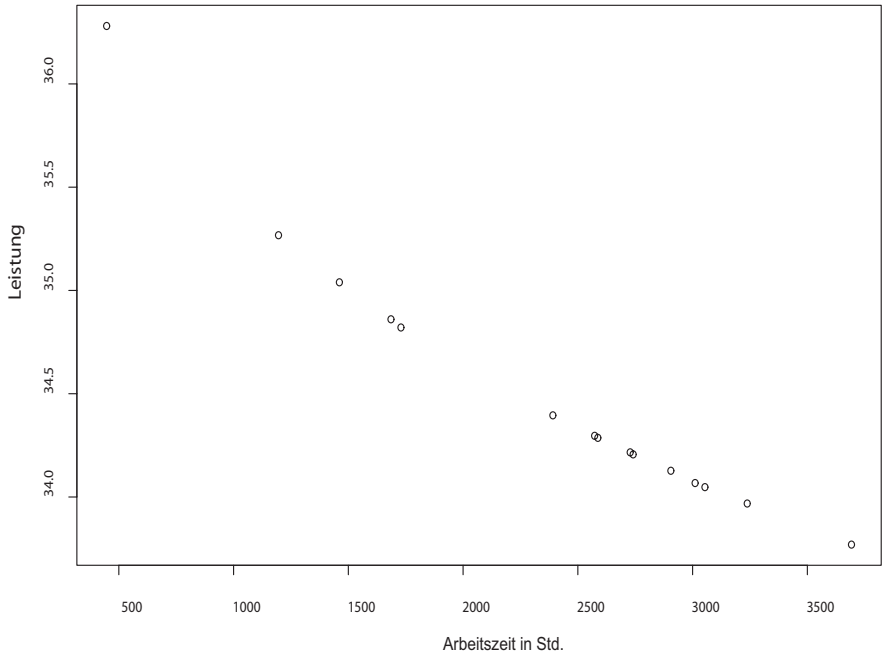


Abbildung 1.1: Lebenszyklus eines Feldgerätes. Fällt die Leistung unter einen kritischen Wert (in diesem Beispiel – unter ca. 34.75 W), so muss das Feldgerät getauscht werden.

Aus der Abbildung 1.1 wird ersichtlich, dass die Leistung eines Feldgerätes im Laufe seines Lebenszyklus abfällt, bis sie irgendwann nicht mehr ausreichend ist und das Gerät ausgetauscht werden muss. Es handelt sich offensichtlich um eine Leistungsregression.

Als wichtigste Anforderung für eine Prognose gilt, dass ein Über- und ein Unterschätzen des tatsächlich realisierten Wertes nicht dieselben Auswirkungen haben [24]. Schätzt man die Leistung eines Feldgerätes zu einem bestimmten Zeitpunkt niedriger ein, führt das unter Umständen zum überflüssigen Austausch des Gerätes. Wird die Leistung aber zu hoch eingeschätzt, so wird das Feldgerät versagen, bevor es ausgetauscht wird. Um das Problem des Über- und Unterschätzen zu berücksichtigen, kann ein asymmetrishes Kostenmass eingesetzt werden. Somit würde man z.B. das Überschätzen des Lebenszyklus höher bestrafen als das Unterschätzen.

Das Prognoseverfahren muss in der Lage sein, verschiedene Faktoren zu berücksichtigen. Da alle Feldgeräte unmittelbar nach ihrer Produktion den Ausgangstest passieren, unterscheiden sie sich bei der kleinen Anzahl der gearbeiteten Stunden im Punkt Leistung nur geringfügig. Im Laufe ihres Einsatzes können diese Unterschiede immer deutlicher werden. Es ist aber durchaus möglich, dass ein Zusammenhang zwischen den Feldgeräten auch weiter besteht. Dieser Aspekt sollte in die Prognose mit einfließen. Die Leistung eines Feldgerätes zum Zeitpunkt der Prognose hängt z.B. auch davon ab, ob die höchstzulässige Stromstärke während des Einzatzes nicht

überschritten und ob das Feldgerät in optimalen Temperaturbereichen betrieben wurde.

Die Anzahl der Feldgeräte, deren Zustände gemessen werden, ist sehr gross. Hieraus resultiert ein riesiges Datenvolumen, das einem Prognoseverfahren zur Verfügung steht. Somit braucht man ein Prognoseverfahren, das sowohl Zusammenhänge zwischen den Verhalten der Feldgeräte berücksichtigt als auch mit großen Datenmengen arbeiten kann.

1.4 Ziele

Diese Arbeit verfolgt folgende Ziele:

- Verschiedene Prognoseverfahren für die Anwendungslogik des in das kundenbezogene PAM-System zu integrierenden Prognosemoduls einsetzen
- Ergebnisse vergleichen und analysieren
- Einen Algorithmus für die Prognose erstellen und implementieren
- Verschiedene Interpolationsmöglichkeiten beim Einsatz der Prognoseverfahren ausprobieren
- Empirisch nachweisen, dass der Algorithmus zuverlässige Prognosen liefert.

Man kann den Inhalt dieser Arbeit in fünf Bereiche unterteilen:

- Beschreibung des kundenbezogenen PAM-Systems (Kapitel 2)
- Erarbeitung der Grundlagen des maschinellen Lernens (Kapitel 3)
- Grundlagen und Arbeitsweise von R-Project (Kapitel 4)
- Erweiterung des PAM-Systems um das Prognosemodul (Kapitel 5)
- Anwendung des Prognosemoduls: Tests und deren Interpretation (Kapitel 6)

Kapitel 2 beschäftigt sich mit dem kundenbezogenen PAM-System. Es werden die Software-Bausteine des Systems vorgestellt, wobei der Akzent insbesondere auf die Architektur gelegt wird. Am Ende dieses Kapitels ist der Leser in der Lage, sich die Ausgangssituation beim Kunden vorzustellen.

In Kapitel 3 wird ein Überblick über das maschinelle Lernen und die damit verbundenen Prognoseverfahren verschafft. Dabei wird das maschinelle Lernen als Begriff erklärt, wonach die Formen des maschinellen Lernens und deren Ziele vorgestellt werden.

In Kapitel 4 wird die Software R-Project vorgestellt. In einer Einweisung in die Eigenschaften des R-Project Softwarepakets wird gezeigt, warum R-Project für den Einsatz im Rahmen dieser Arbeit gut geeignet ist. Darauffolgend wird die Schnittstelle zu dem PAM-System beschrieben.

Der Schwerpunkt dieser Arbeit liegt auf dem Kapitel 5, das sich mit der konzeptionellen Erweiterung des kundenbezogenen PAM-Systems um das Prognosemodul beschäftigt. In diesem Kapitel werden die in Kapitel 3 vorgestellten Prognoseverfahren analysiert, indem sie auf die Kundendaten angewendet werden. Die daraus gewonnenen Ergebnisse spielen eine wichtige Rolle für den Entwurf des Prognosemoduls. Es wird ein Algorithmus ausgearbeitet, der anschließend in die Geschäftslogik des PAM-Systems integriert wird.

Das um das Prognosemodul erweiterte PAM-System wird in Kapitel 6 getestet. Dabei wird die Arbeitsweise sowie die Zuverlässigkeit seiner Prognose gezeigt.

Den Abschluss dieser Arbeit bildet Kapitel 7 mit einer Zusammenfassung der Inhalte und einem Resümee der Ergebnisse.

Kapitel 2

Struktur des PAM-Systems

Die Anforderungen an das PAM-System sind klar definiert worden: Neben der Kunden- und Produktverwaltung sollen die Auswerte- und Prognosetools für die Feldgeräte zur Verfügung gestellt werden. Aufgrund sehr großer Anzahl der Feldgeräte und deren weltweiter Standorte ist die 3-schichtige Architektur von Vorteil, die im Rahmen der Einführung in die Softwarebausteine und -technologien des PAM-Systems zusammen mit der Servlet-Technologie vorgestellt wird. Im Folgenden werden die einzelnen Schichten des PAM-System näher betrachtet.

2.1 Softwarebausteine und -technologien

Die Architektur des kundenbezogenen PAM-Systems ist 3-schichtig. Die zumindest grobe Einführung in diese Architektur ist daher notwendig. Servlets werden für die Berechnung und Darstellung der Ergebnisse eingesetzt. Gleichzeitig wird durch sie die webbasierte Diagnose und Kommunikation mit den Feldgeräten ermöglicht. Durch die Schnittstelle zum externen Programmpaket R-Project (wird in Kapitel 4 beschrieben) wird die grafische Darstellung der Ergebnisse vorbereitet sowie einige statistische Auswertungen durchgeführt. In diesem Abschnitt werden die Grundlagen über die verwendete 3-Schicht-Architektur und Servlets vermittelt.

2.1.1 3-Schicht-Architektur

Nachdem der PC sein Inseldasein durch das Auftauchen leistungsfähiger *local area network* hinter sich liess, begann man bald, die PCs nicht nur untereinander sondern auch mit zentralen Servern zu koppeln. Das *client/server computing* entstand. Bei den Servern handelt es sich bis heute mehrheitlich um File- und Datenbankserver; Applikationsserver sind die Ausnahme. Beim Zugriff auf einen Datenserver liegen nur die Daten auf dem Server, die gesamte Applikationsintelligenz wird auf dem Client implementiert. Da bloss die architektonischen Schichten Datenserver und Client vorhanden sind, spricht man hier von einer 2-schichtigen Architektur. Dieses Modell ist auch heute noch das vorherrschende. Es ist eigentlich genau das Gegenteil seines populären, terminalbasierenden Vorläufers, bei dem die ganze Intelligenz auf dem Host lag.

Doch zeigt das 2-schichtige Modell eklatante Schwächen, die die Entwicklung und Wartung solcher Applikationen massiv verteuern:

- *Fat*-Client
  - Die gesamte Entwicklung kumuliert sich auf den PC. Dieser übernimmt sowohl die Verarbeitung als auch die Darstellung von Information, was zu großen, monolithischen und nur teuer zu wartenden Applikationen führt.

- Erhöhte Netzwerkbelastung
  - Da die eigentliche Verarbeitung der Daten auf dem Client passiert, müssen alle Daten über das Netzwerk transportiert werden. Dies führt in der Regel zu erhöhten Netzwerkbelastungen.
- Ineffiziente Transaktionssteuerung
  - Das ganze Transaktionsverhalten muss vom Client aus gesteuert werden. Verteilte Updates sind fragwürdig.
- Übertragung sensibler Daten
  - PCs gelten sicherheitsmäßig als *untrusted*, d.h. ihre Authentisierung gestaltet sich viel schwieriger als diejenige von Servern. Trotzdem werden in 2-schichtigen Architekturen gezwungenermaßen auch sehr sensible Daten auf den PC übertragen.
- Keine serverseitige Datenverareitung
  - Auf dem Server werden nur Daten angeboten, keine Verarbeitungen. *Stored procedures* sind eine Hilfslösung der Datenbankanbieter mit beschränktem Einsatzbereich und proprietärer Natur.
- Keine Wiederverwendbarkeit der Applikationslogik
  - Applikationslogik kann nicht wiederverwendet werden, weil sie an einzelne PC-Programme gebunden ist.
- Drastische Einflüsse auf das *change management*
  - Müssen aufgrund von geschäftspolitischen oder gesetzlichen Änderungen (z.B. Einführung der Mehrwertsteuer) Verarbeitungen geändert werden, so sind evtl. dutzende von PC-Programmen anzupassen, die die gleiche Logik implementieren. Dass alle diese Programme wiederum einzeln qualitätsgesichert werden müssen, und sie nach der Änderung wieder alle die gleichen Ergebnisse liefern sollten, versteht sich von selbst.
- Aufwändge Ausbreitung von Software
  - Gerade in größeren Umfeldern ist dies im 2-schichtigen Modell sehr aufwändig. Neue Programmversionen müssen entweder über das Netzwerk oder geeignete Medien (Disketten oder CD) verbreitet werden. Das Erste kann das Netzwerk verstopfen, das Letztere ist ein aufwändiger manueller Prozess. Ist die neue Version einmal beim Endbenutzer angekommen, muss dieser sie installieren und testen. Die Betriebsverantwortlichen können bei diesem Vorgang letztendlich keine Garantien geben, ob, wann und in welcher Qualität die Version auf allen betroffenen PCs eingespielt wurde.

3- und n-schichtige Architekturen versuchen, diese Problematik zu lösen. Erreicht wird dieses Ziel primär durch eine Rückverlagerung von Applikationslogik vom Client auf den Server.

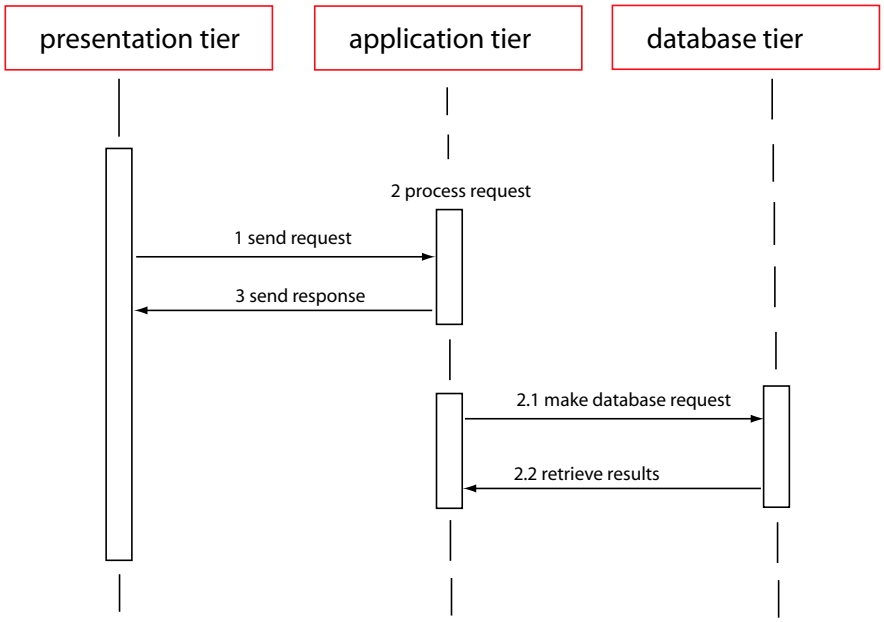


Abbildung 2.1: Aufbau eines 3- und mehrschichtigen Systems (aus [7]).

Abbildung 2.1 zeigt eine vereinfachte Referenzarchitektur, die aber dem Prinzip nach schon alle Möglichkeiten aufzeigt. Ein Client (*presentation*-Schicht) sendet eine Anfrage an den Anwendungsserver (*application*-Schicht, Schritt 1). Der Anwendungsserver bearbeitet diese Anfrage (Schritt 2) und operiert – z.B. durch Datenbankabfragen – auf den Daten, die sich auf einem Datenbankserver (*database*-Schicht) befinden (Schritt 2.1). Die von dem Datenbankserver erhaltenen Ergebnisdaten (Schritt 2.2) werdem - evtl. nach ihrer weiteren Verarbeitung, z.B. Aufbereitung für die Präsentation - an den Client weitergeleitet (Schritt 3).

Es ist wichtig festzuhalten, dass es sich bei diesen Schichten um logische Grenzen handelt. Es ist also ohne Weiteres möglich, dass alle drei Ebenen auf ein und derselben physischen Maschine laufen. Zentral ist aber das saubere Strukturieren des Systems und eine wohldurchdachte Definition der Schnittstellen zwischen den Ebenen.

presentation-Schicht

Diese ist zuständig für die Präsentation der darzustellenden Daten, der Entgegennahme von Benutzerereignisse sowie für die Steuerung des Benutzeroberfläche. Die eigentliche Geschäftslogik ist ausgelagert worden. Mit verteilten Objekttechnologien wie CORBA und RMI kann heute der Client nicht nur eine traditionell geschriebene PC-Applikation sein, sondern auch ein Applet für einen Java-fähigen Browser.

application-Schicht

Diese Schicht ist neu, d.h. in einer 2-schichtigen Architektur nicht in dieser expliziten Form vorhanden. Hier befinden sich *business*-Objekte, welche die Geschäftsregeln implementieren und der *presentation*-Schicht zur Verfügung stellen. Diese Ebene bildet nun den zentralen Schlüssel zur Lösung von 2-Schichten-Problemen. Diese Schicht schutzt die Datenbestände auch vor direktem Zugriff der Clients.

Die objektorientierte Analyse (OOA, beschrieben in [5]) zielt von ihrem Wesen her auf diese Schicht ab: Die Erfassung und Abstraktion von Geschäftsprozessen in *business*-Objekten. Es

ist also möglich, die *application*-Schicht direkt aus CASE-Tools, welche OOA unterstützen, zu mappen.

Weiter trifft man nun auch hier auf den Begriff *Komponenten*. Bisher hat sich der Begriff hauptsächlich auf der Client-Seite für visuelle Komponenten etabliert. Hier, im nicht-visuellen Bereich des Systems, sind Komponenten für spezifische Zwecke konfigurierbare Objekte, welche zu neuen applikatorischen Prozessen zusammengefügt werden können.

database-Schicht

Die für die Datenspeicherung zuständige Schicht ist die *database*-Schicht. Neben den weitverbreiteten relationalen Datenbanksystemen können hier auch bestehende Applikationen in Frage kommen, welche zur Datenspeicherung und vielleicht auch zur Service-Erbringung eingebunden werden können.

2.1.2 Servlets

Servlets sind Java-basierte Webkomponenten. Durch die Verwendung von Java ist die volle Plattformunabhängigkeit gegeben. Dies ermöglicht daher einen flexiblen Einsatz auf allen gängigen Plattformen.

Servlets sind dazu gedacht, den Inhalt einer Website dynamisch zu generieren. Sie sind rein serverseitige Komponenten und bilden daher eine starke Erweiterung der Serverfunktionalität. Diese können dabei dynamisch vom Webserver geladen werden.

In ihrer Funktionalität liegen Servlets zwischen der CGI-Technologie (*common gateway interface*) und proprietären Servererweiterungen wie das Netscape Server API (NSAPI) oder Apache Module. Servlets zeichnen sich gegenüber anderen Servererweiterungen durch folgende Vorteile aus [10]:

- Effizienz
  - Bei traditionellen CGI Skripts (*fast CGI* bietet bereits eine Verbesserung) wird für jeden *HTTP request* ein neuer Prozess gestartet. Für ein relativ kleines CGI-Skript ist das Starten des Prozesses schon viel aufwändiger, als das Skript selbst. Bei Servlets hingegen erzeugt die *java virtual machine* JVM für jeden *request* einen eigenen *lightweight java thread* und keinen eigenen Betriebssystemprozess.
  - Kommen n gleiche *requests* an dasselbe CGI-Skript, so wird der Programmcode n mal in den Speicher geladen. Bei den Servlets hingegen existiert nur eine Kopie der Servletklasse im Speicher, wobei n *threads* mit dieser Klasse arbeiten.
  - Ist ein *request* abgearbeitet, so wird der Programmcode des CGI-Skripts wieder aus dem Speicher entfernt. Dadurch ist es besonders schwer, Daten zu cachen, Datenbankverbindungen aufrecht zu erhalten oder andere Optimierungsaufgaben, die persistente Daten bearbeiten, durchzuführen. Servlets bleiben nach der Bearbeitung eines *request* im Speicher. Dadurch ist es möglich, komplexe Daten zwischen den *requests* auszutauschen.
- Einfachheit
  - Servlets besitzen eine umfangreiche Infrastruktur für das automatische Parsen und Dekodieren von HTML-Formulardaten, um HTML-Headerdaten zu lesen und zu ändern, mit *cookies* zu arbeiten, *session tracking* zu betreiben, und viele andere *high level*-Anwendungen.

- Leistungsfähigkeit

  - Servlets unterstützen Techniken, welche u. a. mit herkömmlichen CGI-Skripten sehr beschränkt oder überhaupt nicht realisierbar sind (z B. den sogenannten *garbage collector*, der sich darum kümmert, dass der Speicher von ungenutzen Objekten bereinigt wird, womit der benötigte Speicher minimiert wird). Weiters können Servlets direkt mit dem Webserver kommunizieren. Dadurch wird es erleichtert, Daten zwischen Servlets auszutauschen, das Verbingungspooling von Datenbankverbindungen leichter zu implementieren und viele andere Optimierungen zu realisieren, bei denen Ressourcen geteilt werden sollen.

- Portierbarkeit

  - Durch die Programmierung in Java ist die Servlet Technologie absolut plattformunabhängig. Es brauchen Servlets daher nur einmal entwickelt zu werden und können ohne großen Aufwand auf den verschiedensten Webservern eingesetzt werden.

- Sicherheit

  - Herkömmliche CGI-Programme werden oft mittels einfacher Shell-Skripts erstellt. Der Programmierer muss dabei sehr darauf achten, dass Shellbefehl, Klammern oder Strichpunkte, sorgfältig ausgefiltert werden. Weiters gibt es Sprachen, die keine ordentliche Überprüfung von Array- und Stringgrenzen durchführen. Es ist z. B. in C oder C++ möglich, ein 100 elementiges Array zu allokieren und anschließend z.B. auf das 199 Element zu schreiben. Werden solche Dinge von den Programmierern nicht sorgfältig überprüft, kann das ein hohes Sicherheitsrisiko für das System darstellen.

    Bei Servlets kann dies nicht passieren. Wird von einem Servlet ein Systembefehl ausgeführt, so geschieht dies nicht über eine Shell. Genauso kann man in Java nicht auf Speicherstellen zugreifen, die nicht allokiert wurden.

Mehr Informationen über die Servlet-Technologie gibt es in [6] und [10]. Die darin beschriebene Servlet-Spezifikation bietet einen genauen Überblick über Servlets und deren Anforderungen.

Für den Einstieg in die Servlet-Technologie bieten sich die Online Tutorien [16] und [3] an. Des Weiteren gibt es bereits viele Webseiten (z.B. [15]), welche sich voll und ganz den Servlets verschrieben haben.

2.2 Datenbank Management System

Die Aufgabe des Datenbank Management Systems (DBMS) besteht in der Speicherung der Daten, die die Arbeitsgrundlage für das PAM-System darstellen. Für alle Daten gilt der Grundsatz der zentralen Datenhaltung. So wird eine komfortable Administration möglich; Einfache Datensicherung durch zentrale Backups ist eine weitere Stärke, die hierdurch erreicht wird. Auf das Datenbank Management System wird über eine *java database connectivity* (JDBC) -Schnittstelle zugegriffen [25]. Somit kann das System leicht auf eine andere Datenbank als die vom Entwicklerteam vorgesehene angepasst werden.

Dem Datenbank Management System liegt eine relationale Datenbank zugrunde. Durch den Einsatz einer relationalen Datenbank ist das PAM-System für zukünftige Erweiterungen offen. Die Daten werden in den Tabellen abgelegt. Dafür wurde das entsprechende Datenmodell entworfen, das in der Abbildung 2.2 vorgestellt wird.

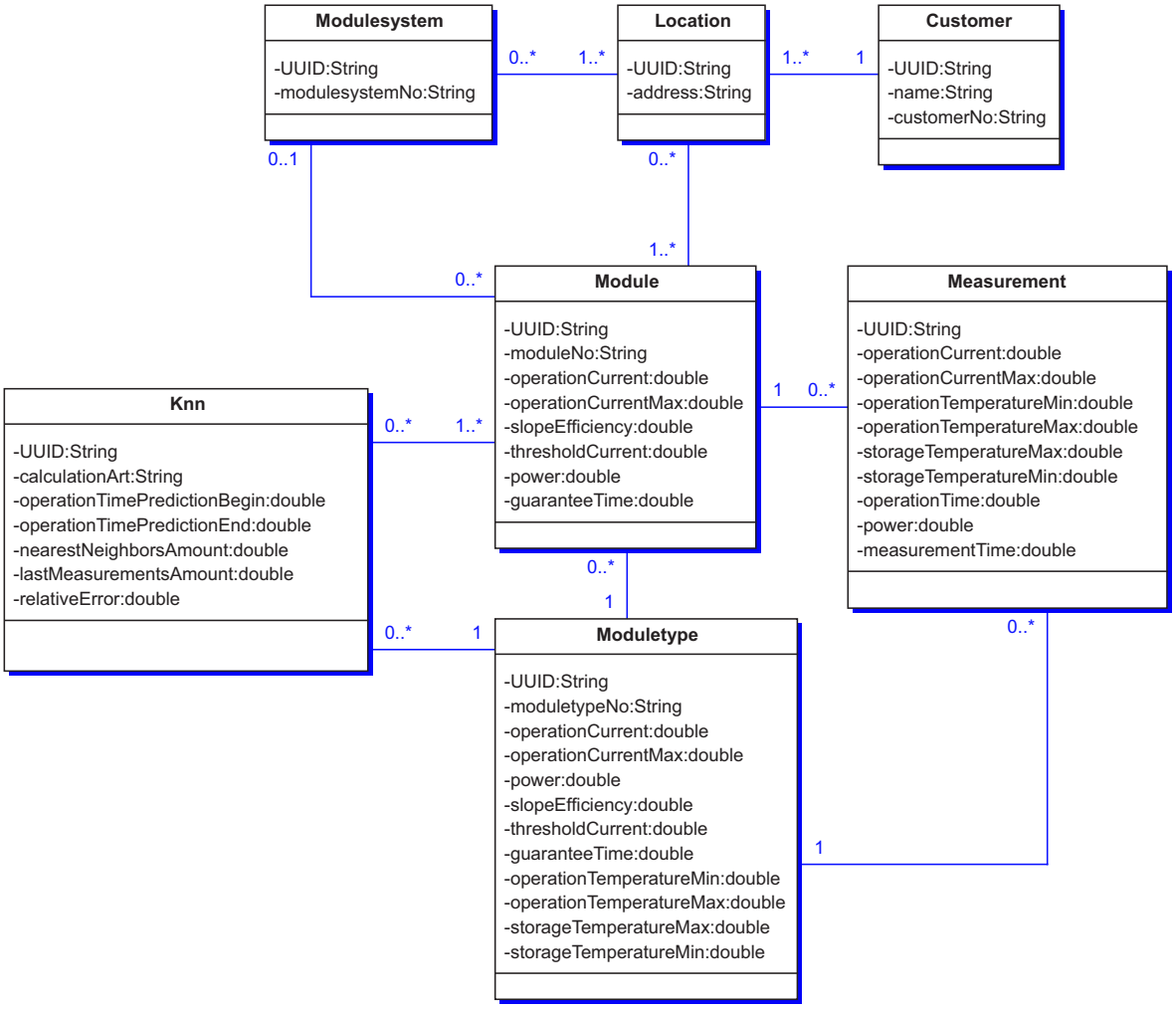


Abbildung 2.2: Das Datenmodell des PAM-Systems

Die Bedeutung der einzelnen Klassen (Entitäten) lässt sich wie folgt beschreiben: Die Klasse *customer* repräsentiert einen Kunden. Der Kunde kann einen oder mehrere Standorte (*location*) haben. Die Feldgeräte (*module*) werden nach ihrer Produktion dem Kunden ausgeliefert und werden an einem seiner Standorte in Betrieb genommen. Optional können mehrere Feldgeräte in ein Gerätesystem (*modulesystem*) eingebunden werden, um z.B. stärkere Leistung zu erreichen. Die Gerätesysteme werden – genauso wie die Geräte selbst – an einem Standort eingesetzt.

Jedes der Feldgeräte gehört zu einem Feldgerätetyp (*moduletype*). Feldgerätetypen besitzen bestimmte Parameter wie z.B. Leistung, Arbeitsstromstärke und Aufbewahrungstemperatur. Das sind die Vorgabewerte, die Soll-Werte für alle Feldgeräte eines bestimmten Typs. Nachdem ein Feldgerät produziert wurde, passiert es seinen ersten Test. Die Messergebnisse, die unter *measurement* gespeichert werden, dürfen gewisse Abweichungen von den Richtwerten des Feldgerätetyps aufweisen, da es in der Praxis nahezu unmöglich ist, alle Feldgeräte mit exakt denselben Werten zu produzieren. Die Rahmen der Abweichungen sind aber entsprechend definiert.

Die Tabelle *knn* ist für das zu implementierende Prognosemodul vorgesehen.

2.3 Anwendungslogik

Die Anwendungslogik bildet das Kernstück des Systems. Ihre Aufgabe ist es, alle im System anfallenden Arbeitsprozesse abzubilden und bei entsprechendem Aufruf abzuarbeiten. Ein Arbeitsprozess ist dabei eine zusammengehörige Folge von Aufgaben, welche nach bestimmten Regeln auf ein bestimmtes Ziel hin durchgeführt werden. Als Beispiel ist die Durchführung einer Prognose für ein Feldgerät zu nennen. Von der Beauftragung der Prognose durch den Kunden bis zum Empfang einer Statusmeldung am Ende der Prognose sind verschiedene Teilaufgaben zu erfüllen, welche als Gesamtheit einen Arbeitsprozess darstellen.

Ein Arbeitsprozess kann auf drei verschiedenen Weisen gestartet werden. Die erste Möglichkeit ist der Aufruf durch die Benutzeroberfläche im Rahmen einer Benutzerinteraktion. Mögliche Interaktionen sind zum Beispiel die Anweisung einer Auswertung oder die Pflege der angegebenen Kundendaten. Die zweite Möglichkeit ist der Prozessstart durch den Datenkonverter, zum Beispiel nach dem erfolgten Import neuer Messungsdaten. Die dritte Möglichkeit ist die automatische Ausführung von Standard-Aufträgen, wie zum Beispiel die Überprüfung, ob neue Messungsdaten hinzugekommen sind.

2.4 Benutzeroberfläche

Die technische Seite der Benutzeroberfläche bereitet die darzustellenden Daten in das HTML-Format auf. Der Zugriff auf das System erfolgt von Seiten des Kunden nur per Internet; hierfür wird eine HTML-Darstellung der Benutzeroberfläche bereitgestellt. Wählt der Benutzer eine Funktion aus, leitet die grafische Bentzerschnittstelle die zur Ausführung nötigen Schritte bei der Anwendungslogik ein.

Die Benutzeroberfläche erhält von der Anwendungslogik fachliche Objekte, welche zur Anzeige ermittelt werden. Dies geschieht mit Hilfe der Servlets.

Bei der Gestaltung der grafischen Oberfläche wird besonderer Wert auf Benutzerfreundlichkeit und Softwareergonomie gelegt. Vor allem ein modernes, ausgewogenes Design ist eine maßgebende Eigenschaft dieses Systems. Die für alle PC-Anwender vertrauten Webbrowser-Bedienkonzepte garantieren intuitive Bedienung und kurze Einschulungszeiten.

Die Idee ist das Schaffen einer Oberfläche, die es ihren Benutzern ermöglicht, die Bedienung und Navigation möglichst einfach und unkompliziert zu erleben. Eine große Rolle spielen eine überschaubare Seitenstruktur sowie die einfache Möglichkeit, die verschiedenen Funktionen, wie beispielsweise die Übersicht über Auswertungen, direkt von der Startseite aus zu erreichen.

Das einheitliche Design aller Seiten des Systems trägt dazu bei, es allen Benutzern zu erleichtern, das System wiederzuerkennen und sich darin zurechtzufinden. Die feste Breite der Seiten unterstützt dabei dieses Ziel, indem dafür gesorgt wird, dass alle Elemente der Seiten unabhängig von der Bildschirmauflösung ihre feste Position behalten. Somit wird einem versehentlichen Verrutschen der Textabsätze oder der Formularelemente vorgebeugt, wodurch eine Verwirrung des Benutzers vermieden wird.

In ihrem Anfangszustand verfügt das PAM-System über die Möglichkeiten, Entitäten (Kunden, ihre Standorte, Feldgeräte und deren Messungen) zu verwalten. Die Grundfunktionen der Verwaltung dieser Entitäte (Neuanlage, Suchen, Bearbeiten und Speichern) sind ebenfalls implementiert.

Die Abbildung 2.4 zeigt, wie eine Entität (in diesem Beispiel ein Feldgerät) gefunden werden kann. Sehr einfach navigiert man über die Menüleiste links, bis das entsprechende Suchfenster in dem Frame rechts erscheint. Darüber definiert man die Suchkriterien, mit deren Bestätigung eine Abfrage an die Datenbank geschickt wird.

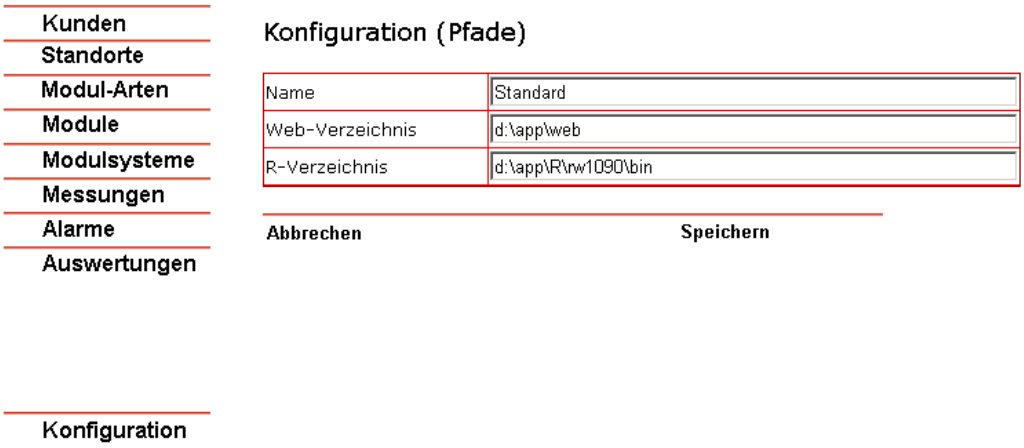


Abbildung 2.3: Benutzeroberfläche des PAM-Systems.

Die zurückgelieferten Daten werden mit Hilfe eines Servlets in eine HTML-Seite überführt, die danach angezeigt wird (Abbildung 2.6).

Über den entsprechenden Link hat man die Möglichkeit, die Eigenschaften einer Entität anzeigen zu lassen. In diesem Beispiel erreicht man die Eigenschaften eines Feldgerätes, indem man auf die Feldgerätenummer klickt (Abbildung 2.7).

Neben den Möglichkeiten, die Eigenschaften zu löschen, verändern und speichern, wäre es vorteilhaft, den Benutzern den Zugriff zu Analyse-Programmen zu verschaffen. Am komfortabelsten erreicht man sie durch die entsprechende Schaltflächen oberhalb von den aufgelisteten Eigenschaften. So kann man sich die letzte Messung des Feldgerätes anschauen, um z.B. seinen Ist-Zustand feststellen zu können. Oder man lässt sich den Lebenszyklus des Gerätes anzeigen. Für die Zustandsprognose des Feldgerätes ist die entsprechende Schaltfläche vorgesehen, bei deren Betätigung ein Prognosemodul aktiviert werden wird, das seinerseits die Prognose erstellt.

2.5 Zusammenfassung

In diesem Kapitel wurde das PAM-System vorgestellt. Nach der Einführung in die 3-Schicht-Architektur wurde gezeigt, wie man durch die Betrachtung des kundenorientierten PAM-Systems die einzelnen Schichten wiedererkennt. Dabei wurde jede der Schichten detailliert beschrieben. Es wurden auch die Elemente der Benutzeroberfläche erläutert, die für die Erweiterung des Systems um die Prognose- und Auswertungsfunktionen vorgesehen sind.

Die Vorstellung des PAM-Sytems ist als weitere, praktische Grundlage für die Bearbeitung der Ziele dieser Diplomarbeit anzusehen. Zusammen mit dem im nächsten Kapitel vermittelten theoretischen Grundwissen bildet das vorgestellte Material eine Basis zum Entwurf des Prognosemoduls.

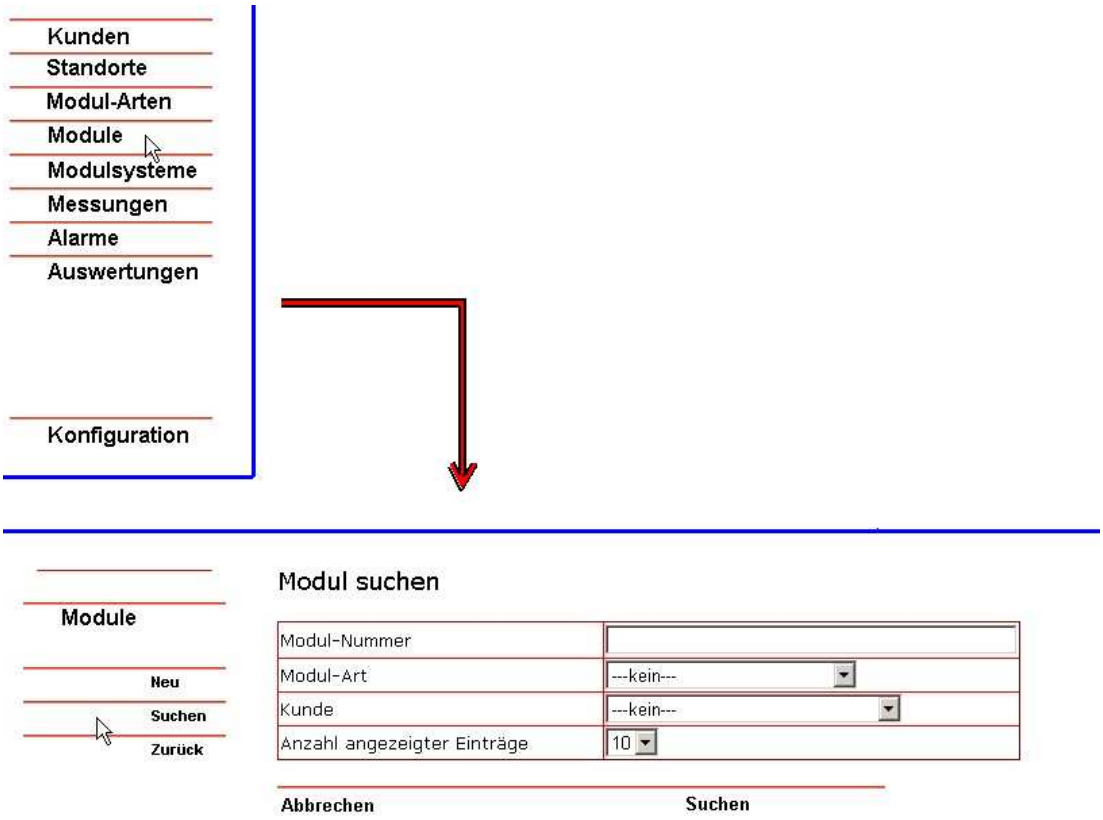


Abbildung 2.4: Suchaktion eines Feldgerätes.

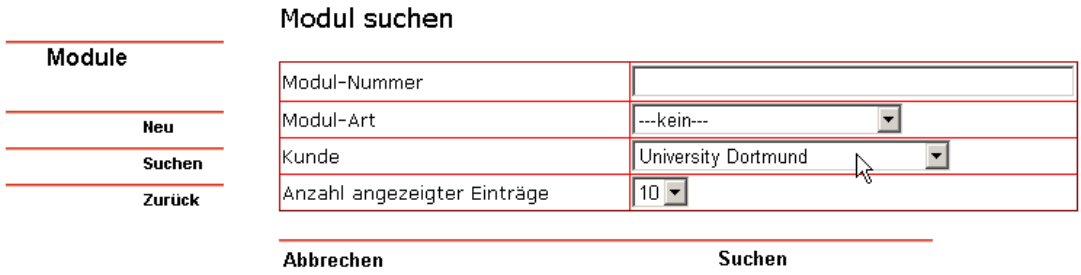


Abbildung 2.5: Auswahl der Kriterien für die Suche nach einem Feldgerät.

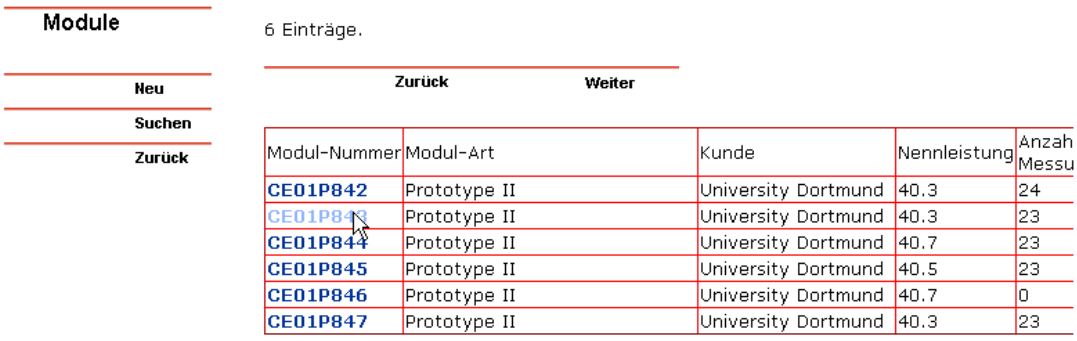


Abbildung 2.6: Präsentation der Suchergebnisse.

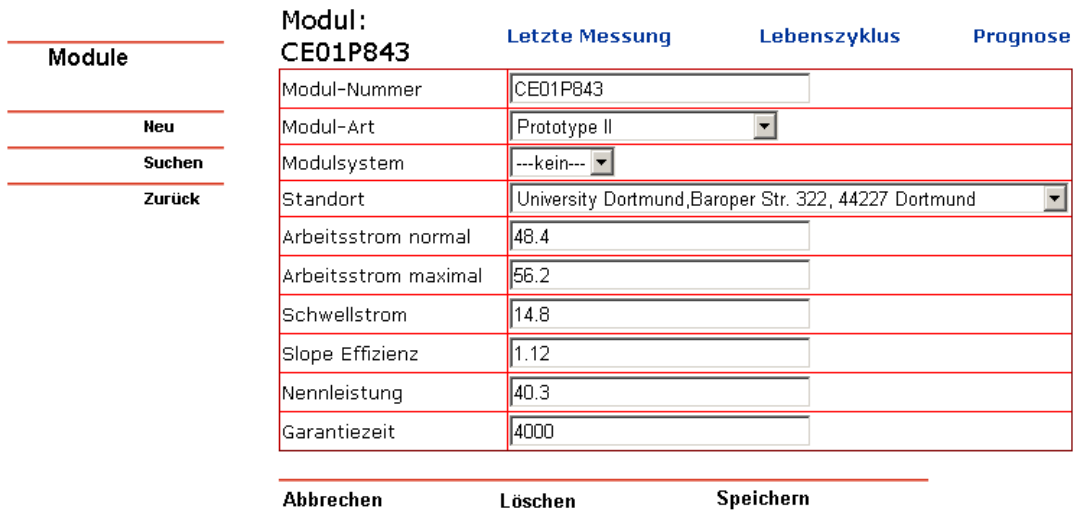


Abbildung 2.7: Eigenschaften eines Feldgerätes.

Kapitel 3

Maschinelles Lernen

In der Ausgangsituation hat man große, strukturierte Bestände von Daten, in denen interessante, aber schwer aufzuspürende Zusammenhänge vermutet werden. Nun möchte man aus den vorhandenen Daten neue, ungeahnte Erkenntnisse automatisch extrahieren. Es handelt sich also um den Einsatz eines der Wissensgewinnungsverfahren, deren Ziel es ist, Entscheidungsprozesse dadurch zu verbessern, dass Regelmäßigkeiten des betrachteten Themengebiets erkannt und genutzt werden – und zwar durch Maschinen und Menschen. Beim Menschen und anderen Lebewesen spricht man bei einem solchen Vorgang im Allgemeinen von Lernen. Man kann diesen Begriff auch auf Maschinen anwenden und vom maschinellen Lernen (*machine learning*) sprechen [9]. Das Forschungsziel des maschinellen Lernens sind allgemein verwendbare, effiziente Verfahren, die autonom aus großen Datenmengen die bedeutsamsten und aussagekräftigsten Zusammenhänge identifizieren und sie dem Anwender als interessantes Wissen präsentieren. In folgenden Abschnitten wird maschinelles Lernen als Begriff näher betrachtet. Des Weiteren werden die auf dem maschinellen Lernen basierenden Prognoseverfahren vorgestellt, die später für die Bestimmung der Leistungswerte der Feldgeräte in der Zukunft eingesetzt werden. Anschließend werden die Interpolationsverfahren vorgestellt, die für die Ermittlung der unbekannten Leistungswerte aus den benachbarten – bekannten – Leitungswerten benötigt werden.

3.1 Was ist maschinelles Lernen?

Am sinnvollsten ist es, das maschinelle Lernen durch die einzelnen Typen von Lernaufgaben zu defnieren. Der Vorteil von diesem Vorgehen liegt darin, dass die einzelnen Lernaufgaben präzise definiert werden können.

> *Eine Lernaufgabe wird definiert durch eine Beschreibung der dem lernenden System zur Verfügung stehenden Eingaben (ihrer Art, Verteilung, Eingabezeitpunkte, Darstellung und sonstigen Eigenschaften), der vom lernenden System erwarteten Ausgaben (ihrer Art, Funktion, Ausgabezeitpunkte, Darstellung und sonstigen Eigenschaften) und den Randbedingungen des Lernsystems selbst (z.B. maximale Laufzeiten oder Speicherverbrauch).* [21]

Man unterscheidet bei den Lernverfahren zwischen verschiedenen Arten des Lernens. Beim nichtüberwachten Lernen sind im Gegensatz zum überwachten Lernen die Werte der vorherzusagenden Attribute nicht vorgegeben. Der Algorithmus muss z.B. eine Einteilung in Kategorien selbst finden. Hat man Ähnlichkeitsmaße auf der Menge der Tupel definiert, so handelt es sich bei diesen Verfahren um Cluster-Verfahren in einem sehr allgemeinen Sinn. Diese Verfahren sind vor allem dann sinnvoll, wenn auf der Menge der Tupel Ähnlichkeitsstrukturen vorhanden sind oder vermutet werden, die inhaltlich interpretiert werden können [9].

Es sei X eine Menge möglicher Instanzenbeschreibungen. Beim unüberwachten Lernen geht es um folgende Aufgabe:

- Gegeben:
  - Eine Menge E von Beispielen der Form $x \in X$.
- Finde:
  - Ein optimales Modell für x.

Nichtüberwachte Lernverfahren haben den Vorteil, dass keine kategorisierten oder bewerteten Trainingsdaten vorliegen müssen. Allerdings braucht man auch für solche Verfahren eine Bewertung der Ergebnisse der Kategorisierung. Eine solche Bewertung kann z.B. durch die nachträgliche Begutachtung der gefundenen Kategorisierung durch Sachverständige oder eine experimentelle Überprüfung geliefert werden [9].

Sind bei den Beispielen aus der Trainingsmenge alle Attributwerte und damit auch die der vorherzusagenden Attribute bekannt und werden später von den Prognosealgorithmen benutzt, spricht man von *supervised learning*, überwachtem Lernen oder Lernen aus Beispielen.

Es sei X eine Menge möglicher Instanzenbeschreibungen, D eine Wahrscheinlichkeitsverteilung auf X, und Y eine Menge möglicher Zielwerte. Es sei weiterhin H eine Menge zulässiger Funktionen (auch als Hypothesensprache LH bezeichnet). Eine Lernaufgabe vom Typ Funktionslernen aus Beispielen sieht dann wie folgt aus [21]:

- Gegeben:
  - Eine Menge E von Beispielen der Form $(x, y) \in X \times Y$, für die gilt: $y = f(x)$ für eine unbekannte Funktion f.
- Finde:
  - Eine Funktion $h \in H$, so dass der Fehler $error_D(h, f)$ von h im Vergleich zu f bei gemäss der Verteilung D gezogenen Instanzen aus X möglichst gering ist.

Diese Form des Lernens hat den Vorteil, dass die bekannten Werte der vorherzusagenden Attribute der Beispiele der Trainingsmenge bei der Konstruktion eines Algorithmus verwendet werden können und dieser dadurch im Allgemeinen wesentlich effektiver wird. Sie hat den Nachteil, dass eine (im Allgemeinen große) Trainingsmenge und eine Testmenge bereitgestellt werden müssen, bei denen die Werte der vorherzusagenden Attribute bekannt sind. Solche Datensammlungen sind häufig nur unter großen Kosten zu beschaffen [9].

Die oben beschriebene Lernaufgabe vom Typ Funktionslernen aus Beispielen ist eine Regressionsaufgabe. *Regression* bedeutet im Allgemeinen die fuktionelle Beschreibung der Abhängigkeit von Zufallsgrößen:

$$Y = \mathbb{R}.$$

Sind etwa einige Messwerte eines Feldgerätes bekannt, ein anderer jedoch nicht, so kann mit Hilfe verschiedener Regressionsmethoden der unbekannte Wert als Funktion der bekannten Werte dargestellt und somit vorhergesagt werden.

Vollständigkeitshalber muss auch der Begriff *Klassifikation* erwähnt werden. Das Ziel der Klassifikation ist es, eine Menge von Objekten in eine Menge von Klassen einzuteilen. Dabei sind die Klassen vorgegeben:

$$Y = \{0; 1\}$$

oder

$$Y = \{-1; 1\}.$$

3.2 Risikominimierung

Klassifikations- und Regressionsaufgaben lassen sich in einem gemeinsamen Rahmen fassen, nämlich den der Minimierung des Risikofunktionals. Gegeben ist eine Kostenfunktion $c(f(x), y)$, die bestimmt, wie sehr Abweichungen zwischen den Vorhersagten $f(x)$ und den wirklichen Reaktionen y des Systems zu bestrafen sind, sowie eine Wahrscheinlichkeitsverteilung $p(x, y)$, die allen Beobachtungen zugrunde liegt, soll das erwartete Risiko minimiert werden:

$$R[f] = \int_{X \times Y} c(f(x), y) p(x, y) dx dy.$$

Bei Klassifikation ist $c(f(x), y) = 0$ für den Fall einer korrekten Vorhersage, d.h. falls $f(x) = y$, und ansonsten 1. Bei Regression könnte es sich bei $c(f(x), y)$ um den quadratischen Fehler $(f(x) - y)^2$, den absoluten Fehler $|f(x) - y|$ oder den absoluten Fehler mit ε Toleranz (also $max(0, |f(x) - y| - \varepsilon)$) handeln [30].

Das erwartete Risiko ist meistens nicht explizit berechenbar. Dies liegt daran, dass fast immer $p(x, y)$ unbekannt ist und nur die Beobachtungen $(x_1, y_1), ..., (x_n, y_n) \in X \times Y$ zur Verfügung stehen, die unabhängig und gleichverteilt von $p(x, y)$ erzeugt wurden. Gewöhnlich besteht der Ausweg dann darin, p durch die empirische Dichte $\frac{1}{m} \sum_{i=1}^{n} \sigma_{x_i, y_i}(x, y)$ auszuwerten und das sich daraus ergebende empirische Risiko

$$R_{emp}[f] = \frac{1}{m} \sum_{i=1}^{m} c(f(x_i), y_i)$$

zu minimieren. Dabei wird angestrebt, ein Modell zu wählen, das nicht zu einfach und nicht zu komplex ist. Man versucht also, die Variante des empirischen Risikos zu minimieren, die bereits der Komplexität der Lösung Rechnung trägt. Dies führt zum regularisierten Risiko:

$$R_{reg}[f] = R_{emp}[f] + \lambda \Omega[f] = \frac{1}{m} \sum_{i=1}^{m} c(f(x_i), y_i) + \lambda \Omega[f].$$

$\Omega[f]$ ist Stabilisierungsterm [29] und $\lambda > 0$ die Regularisierungskonstante. Je nachdem, wie groß λ gewählt wird, werden mehr oder weniger einfachere Funktionen favorisiert. Die Wahl von $\Omega[f]$ selbst beschreibt die Komplexität der Funktionenklasse. Im Fall der SVM ist die Komplexität durch die Breite der Hyperebene beschrieben.

Die Hyperebene (genauer gesagt: die *trennende* Hyperebene) ist definiert durch

- den Normalvektor w und
- die Verschiebung b:

$$\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}.$$

$\langle ., . \rangle$ nennt man *inneres Produkt* oder *Skalarprodukt*.

Man wählt w und b so, dass die Hyperebene die Daten trennt. In einfachen Fällen (Abbildung 3.1) ist es machbar: die Daten sind linear trennbar. Der Punkt, für den entschieden werden soll, ob er zu den positiven oder negativen Beispielen gehört, liegt dann entweder in der Richtung des Normalenvektors (wird als positives Beispiel erkannt) oder in der anderen, entgegengesetzten Richtung (wird entsprechend als negatives Beispiel eingestuft).

Bei einem linear separierbaren Problem gibt es unendlich viele Möglichkeiten lineare Aufteilungen vorzunehmen ohne dabei einen Fehler zu machen (bezüglich der Trainingsmuster). Allerdings sollte man eine Aufteilung wählen, die einen möglichst großen Trennungsstreifen (*margin*) zwischen den Trennungsklassen realisiert (Abbildung 3.2).

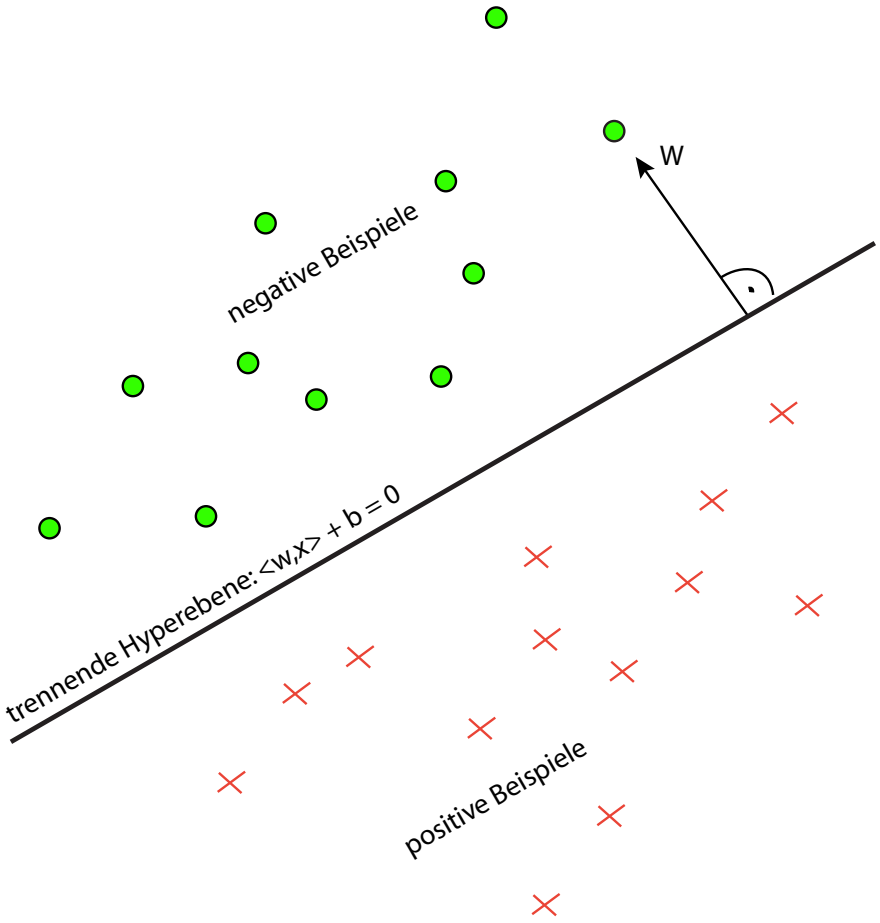


Abbildung 3.1: Hyperebene bei den linear trennbaren Daten.

Somit wird *margin* als Abstand eines Punktes x_i der Klasse $y_i \in \{-1; 1\}$ zu einer Hyperebene $\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$ definiert. Der Gesamtabstand d ist

$$d = \frac{|1-b|}{||w||} + \frac{|-1-b|}{||w||} = \frac{2}{||w||}.$$

Die Punkte, die der Hyperebene am nähesten liegen, nennt man *support vectors* (tragende Vektoren). Sie allein bestimmen die Lage der Hyperebene, alle anderen Punkte haben darauf keinen Einfluss.

margin maximieren heisst $||w||$ minimieren. Außerdem müssen die Nebenbedingungen

$$y_i(\langle w, x_i \rangle + b) \geq 1$$

eingeführt werden um sicherzustellen, dass die Hyperebene die Trainingsdaten auch korrekt trennt. Deswegen werden Lagrange-Multiplikatoren $\alpha \geq 0$ eingeführt und das Optimierungsproblem in der sog. *Lagrange-Funktion* $L(w, b, \alpha)$ zusammengefasst:

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 \ - \sum_{i=1}^{n} \alpha_i (y_i(\langle w, x_i \rangle + b) - 1) = -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_{i=1}^{n} \alpha_i.$$

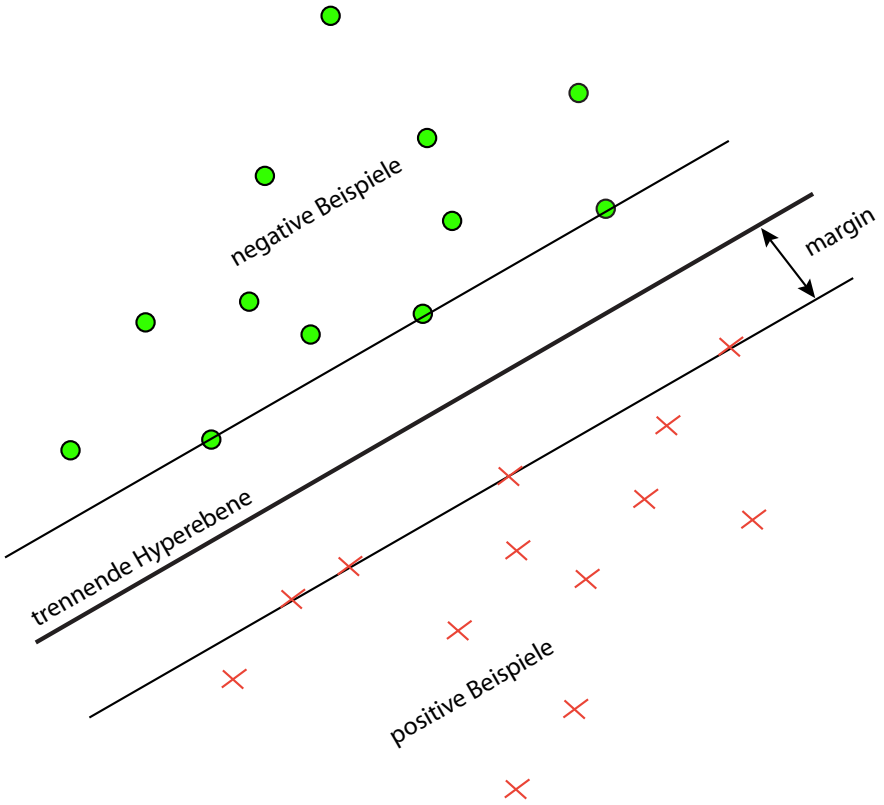


Abbildung 3.2: Daten sollen mit maximaler Trennspanne aufgeteilt werden.

Setzt man das Ergebnis $L(w, b, \alpha)$ ein und formt es um, so erhält man das *duale Problem*

- maximiere

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

- unter den Nebenbedingungen

$$\alpha \geq 0$$

und

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

Man löst das duale Problem und erhält die α_i, die $W(\alpha)$ maximieren. Damit berechnet man den Normalenvektor w mit der Formel

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

und hat die Trennebene mit maximaler Trennspanne gefunden.

3.3 Nichtlinearität und Kernel-Trick

Oft sind Daten nicht linear trennbar. Eine mögliche Lösung wäre eine Entfernung einer minimalen Menge der Datenpunkte, wodurch die Daten linear trennbar werden. Diese Lösung hat allerdings den Nachteil, dass der Algorithmus zu deren Realisierung schnell exponentiell wird.

Das Konzept der Support-Vektor-Maschinen lässt sich jedoch auch auf nichtlineare Daten ausweiten. Um eine nichtlineare Trennung zu erlauben, wird eine nichtlineare Funktion Φ (*feature map*) angewendet.

Bei nichtlinearen Funktionen werden die Trainingsdaten über eine nichtlineare Abbildung

$$\Phi : \mathbb{R}^N \to \mathcal{H}$$
$$x \mapsto \Phi(x)$$

in einen höherdimensionalen Merkmalsraum $\mathcal{H}$ (*feature space*) abgebildet ($\mathcal{H}$ ist ein Raum, in dem ein Skalarprodukt erklärt ist), um eine Nichtlinearität zu erzeugen. Anschließend trennt man die Punkte $\Phi(x)$ (s. Abbildung 3.3).

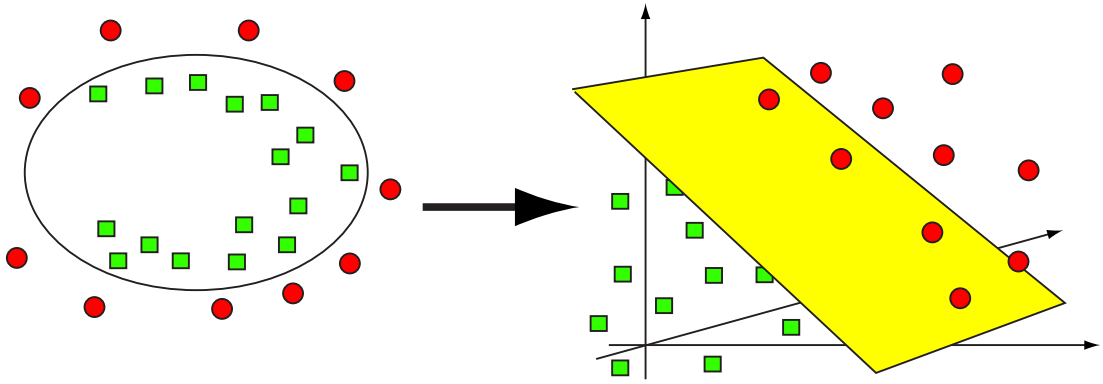


Abbildung 3.3: Einfachere Trennung der Datenpunkte in höheren Dimensionen.

Das Hauptproblem bleibt dabei linear und wird für

$$(\Phi(x_1), y_1), ..., (\Phi(x_n), y_n) \in \mathcal{H} \times Y$$

behandelt. Für $x, y \in \mathbb{R}^N$ und $N, d \in \mathbb{N}$ berechnet

$$k(x, y) = (x \cdot y)^d$$

ein Skalarprodukt im Raum aller Produkte von d Einträgen der Vektoren x, y [27, 30]. Ist $k : C \times C \to R$ stetiger Kern eines positiven Integraloperators auf $L_2(C)$(für eine kompakte Menge $C \subset \mathbb{R}^g$, die die Muster enthält), d.h.

$$\forall f \in L_2(C) : \int_{C \times C} k(x, y) f(x) f(y) dx dy \geq 0,$$

so existieren ein Merkmalsraum $\mathcal{H}$ und eine Abbildung $\Phi : \mathbb{R}^N \to \mathcal{H}$, sodass $k(x, y) = (\Phi(x) \cdot \Phi(y))$. Man kann also das Skalarprodukt in $\mathcal{H}$ berechnen, ohne explizit die Abbildung nach $\mathcal{H}$ zu berechnen. Daraus folgt, dass jeder Algorithmus, der nur Skalarprodukte verwendet, implizit in $\mathcal{H}$ ausgeführt werden kann.

Die Schwierigkeit besteht aber nun darin, eine günstige Kernel-Funktion auszuwählen. Oft verwendete Kernel, die diesen Kernel-Trick ausnutzen, sind:

- Polynome: $k(x,y) = (\gamma\langle\vec{x},\vec{y}\rangle + c_0)^d$
- Radiale Basisfunktion: $k(x,y) = exp(-\gamma(\parallel \vec{x} - \vec{y} \parallel)^2)$

Dabei gilt Folgendes:

- c_0: additive Konstante im Polynom
- γ: Weite der Gaußschen Glocken

3.4 Prognoseverfahren

Wie in der Einleitung bereits angedeutet wird, braucht man ein Verfahren, das mit großen Datenmengen und vielen Attributen arbeiten kann. Ein solches Verfahren ist die *Support Vector Machine* (SVM) [19, 24]. Die *Support Vector Machine* basiert auf der Arbeit von Vladimir Vapnik [30] und ist ein Lernverfahren, das effiektiv zum Lernen mit großen Datenmengen und vielen Attributen genutzt werden kann. Die SVM ist ein universelles Lernverfahren zur Klassifikation, Regression und Dichteschätzung. Sie wurde in [19, 24] beschrieben. Entscheidend für ein erfolgreiches Lernen der SVM ist eine ausreichende Zahl an Lernbeispielen. Mit Zunahme der Beispiele steigt die Genauigkeit, aber auch die benötigte Trainingszeit.

Des Weiteren kommt auch der k-Nächster-Nachbar Algorithmus (*k-nearest neighbor algorithm*) in Frage, da man vermuten kann, dass Objekte in Daten, die eine gewisse Nähe zueinander aufweisen, auch den gleichen Vorhersagewert haben werden. Das bedeutet, wenn man den Vorhersagewert einer Menge von Objekten kennt, kann der Vorhersagewert der anderen Objekte abgeleitet werden.

3.4.1 Supportvektor-Regression (SVR)

Regression wird auch als Funktionsschätzung (*function estimation*) oder Funktionsannäherung (*function approximation*) bezeichnet. Im Rahmen dieser Arbeit ist der Einsatz der *Support Vector Machine* in der Regression von Daten besonders interessant. Hier besteht die Aufgabe darin, gegebene Datenpunkte

$$(y_1, \vec{x_1}, ..., y_i, \vec{x_i}), \vec{x} \in \mathbb{R}^N, y \in \mathbb{R}$$

im einfachsten Falle durch eine lineare Funktion

$$f(x) = (w \cdot \Phi(x)) + b$$

möglichst gut zu approximieren [14]. Die Problemstellung verlangt die Festlegung einer Fehlerfunktion

$$|y - f(x)|_\varepsilon = max\{0, |y - f(x)| - \varepsilon\},$$

die angibt, in welchem Maße Datenpunkte, die nicht auf der Regressionskurve liegen, bestraft werden. Die Funktion f findet man wiederum durch Lösung eines quadratischen Optimierungsproblems, das die Eigenschaft hat, nur von Skalarprodukten abzuhängen. Für die Klasse allgemeiner konvexer Kostenfunktionen kann ebenfalls effizient eine Lösung gefunden werden. Somit können Kernfunktionen verwendet werden. Im Rahmen der Analyse mehrerer Lernverfahren (zwecks deren späteren Übernahme in die Anwendungslogik des Prognosemoduls) wird in Kapitel 5 untersucht, welche Arten der SVM-Regression den Verlauf der Leistung der Feldgeräte am besten approximieren.

Die Gauß- oder quadratische Fehlerfunktion hat den Nachteil, bereits bei etwas größeren Abweichungen sehr große Bestrafungen zu erzeugen. In Kombination mit der linearen Laplace-

oder *least modulus*-Fehlerfunktion ergibt sich die Huber-Fehlerfunktion. Weitere Vereinfachung ergibt die ε-insensitive Fehlerfunktion (entsprechend – ε-SVR). Diese ist bei Abweichungen im Intervall $[-\varepsilon, \varepsilon]$ unempfindlich; eine Bestrafung findet erst außerhalb dieses Intervalls statt. Mit anderen Worten geht es bei der $\varepsilon - SVR$ darum, eine Funktion zu bestimmen, die an allen gegebenen Stützstellen x_i nicht mehr als um ein vordefiniertes ε von y_i abweicht. Im Gegensatz zu der quadratischen Fehlerfunktion werden nur Fehler größer ε bestraft. Die ε-insensitive Fehlerfunktion erfüllt damit genau das Prinzip der strukturellen Risikominimierung (SRM, mehr darüber findet man in [24]) und wird bei der Supportvektor-Regression als Fehlerfunktion verwendet. Durch den Einsatz ε-insensitiver Fehlerfunktion reduziert sich die Zahl der Support-Vektoren und damit die Komplexität des Modells. Es werden somit nicht mehr alle Beispiele benötigt um eine Funktion zu beschreiben.

3.4.2 k-Nächster-Nachbar

Hierbei handelt es sich um eine Klasse einfachster instanz-basierter Algorithmen. Mittels instanzbasierter Verfahren ist es möglich, diskrete und stetige Funktionen zu approximieren, wobei allerdings vorausgesetzt wird, dass sich alle Werte als Punkte im mehrdimensionalen Raum darstellen lassen. Dabei geht man von der Annahme aus, dass Funktionswerte einer Instanz ähnlich zu den Funktionswerten von benachbarten Instanzen sind. Mit anderen Worten werden Objekte in Daten, die eine gewisse Nähe zueinander aufweisen, auch den gleichen Vorhersagewert haben. D.h. wenn man den Vorhersagewert eines dieser Objekte kennt, kann der Vorhersagewert der anderen Objekte abgeleitet werden. Das Lernen instanz-basierter Verfahren geschieht durch Speichern von Daten (*lazy learning*). Mit *lazy learning* bezeichnet man das praktische Entfallen der Lernphase: alle Trainingsdaten werden nur zwischengespeichert und erst ausgewertet, wenn neue Objekte vorherzusagen sind [12]. Die Klassifikation neuer Instanzen wird in der Anwendungsphase direkt aus den Beispielen bestimmt.

Eine Instanz wird über ihre Attribute definiert. Jedes Attribut stellt eine Achse im mehrdimensionalen Raum, die Anzahl des Auftretens der Attribute wird zu einem Punkt-Vektor zusammengefügt,

$$x = (a_1(x), a_2(x), ..., a_n(x)).$$

Die Anzahl des Vorkommens des Attributs x wird durch a_r definiert.

Entschieden wird bei der k-Nächster-Nachbar Methode, wie der Name schon erahnen lässt, anhand der nächstliegenden Daten zum neu einzuordnenen Punkt. Definiert wird der nächste Nachbar über die euklid'sche Abstandsfunktion

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_i(x_r) - a_j(x_r))^2}.$$

Als Rückgabewert beim k-Nächster-Nachbar erhält man den am häufigsten vorkommenden Wert unter den in Betracht gezogenen Werten.

Zusammengefasst ergibt sich folgender Algorithmus [26]:

- *Lernphase*:
  - Füge Beispiele $(x, f(x))$ zur Menge der Beispiele S hinzu.
- *Anwendung*: Klassifikation von neuer Instanz x
  - Bestimme die k nächsten Nachbarn $x_1, ..., x_k$ zu x.
  - k_i Nachbarn haben die Klasse i.

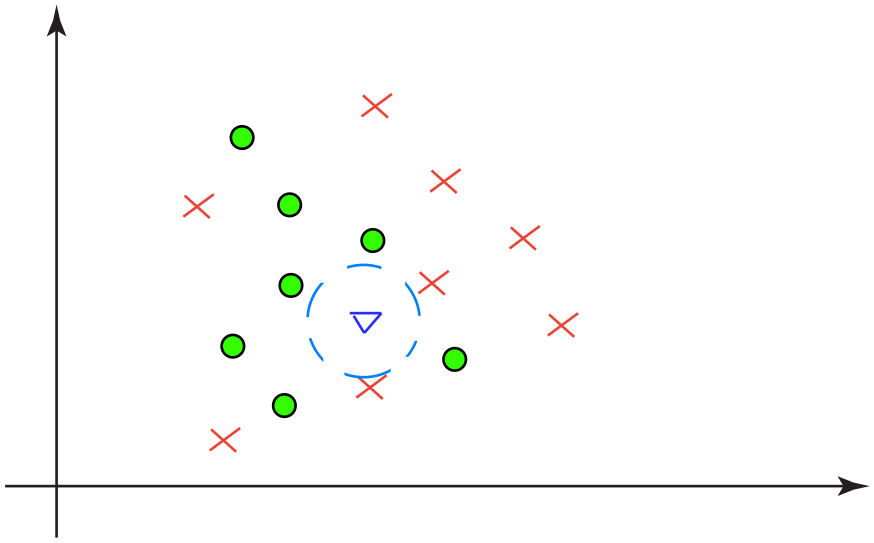


Abbildung 3.4: k-Nächster-Nachbar.

– Mehrheitsentscheidung unter den Nachbarn:

$$f(x) = \arg\max_i \frac{k_i}{k}.$$

Wie das Beispiel in Abbildung 3.4 zeigt, kann es beim k-Nächster-Nachbar durchaus vorkommen, dass die Funktion, die nur einen Nachbarn mit einbezieht, einen anderen Wert liefert als die, die mehrere Nachbarn mit einbezieht. Im dargestellten Beispiel würde der 1-NN den Punkt als Kreuz klassifizieren, der 3- sowohl 5-NN würden ihn als Kreis einstufen [11].

Das Beispiel soll keinesfalls die schwache Zuverlässigkeit der k-Nächster-Nachbar-Methode verdeutlichen. Hier ging es um die Notwendigkeit, eine Gewichtung in Abhängigkeit vom Abstand von der Instanz zu geben. In dieser Diplomarbeit wird k-Nächster-Nachbar nicht für die Klassifizierung benutzt, sondern lediglich für die Bestimmung der Nachbar-Feldgeräte eingesetzt, deren Leistungswerte später für die Prognose benutzt werden.

Den Nachteil des k-Nächster-Nachbar-Algorithmus sieht man darin, dass für jeden einzelnen Vorgang die gesamte Trainingsmenge zur Verfügung stehen und nach ähnlichen Objekten durchgearbeitet werden muss. Ausserdem muss die Anzahl der zu berücksichtigenden Nachbarn von außen festgelegt werden. Für größere Werte von k nimmt der Aufwand noch zu.

3.5 Interpolation und Approximation

Interpolation[1] und Approximation[2] werden in vielfältigen Zusammenhängen benötigt. Dabei ist der Begriff Approximation eigentlich der übergeordnete Begriff. Interpolationsverfahren sind spezielle Approximationsverfahren, bei denen die Näherungskurven durch vorgegebene Punkte verlaufen [20].

Eine typische Aufgabe der Interpolation besteht darin, durch diskret gegebene Punkte, die beispielsweise aus Messungen gewonnen werden, eine geeignete Kurve zu legen. Im einfachsten Fall werden die Punkte durch Geradenstücke verbunden. Mit einer Interpolationsfunktion

[1] lateinlisch für Zwischenschaltung, Schluss von bekannten Werten auf Zwischenwerte
[2] lateinisch für Annäherung

werden also die vorgegebenen Funktionswerte exakt dargestellt, der Näherungsfehler (Approximationsfehler) ist dort Null.

Ist ein fester Funktionstyp bekannt, etwa wenn in der Statistik ein linearer Zusammenhang zwischen Größen vermutet wird, dann lässt sich im Allgemeinen keine Interpolationsfunktion mehr bestimmen, die durch alle vorgegebenen Punkte verläuft, und man muss andere, geeignete Gütekriterien für eine Näherungskurve aufstellen. Beispielsweise lässt sich durch mehr als zwei Messpunkte im Allgemeinen keine Gerade legen, die durch alle Punkte verläuft. Man wählt je nach Anforderung andere Kriterien, um eine möglichst gute Approximation zu erhalten (z.B. das Prinzip der kleinsten Fehlerquadrate).

Von Approximation spricht man auch, wenn der Funktionswert einer bekannten Funktion mit einer Näherungsfunktion gefunden werden soll, die bestimmte Gütekriterien erfüllt. Dies geschieht zum Beispiel bereits dann, wenn mit einem Rechner Werte von Funktionen ermittelt werden sollen, die nicht nur aus den vier Grundrechenarten zusammengesetzt sind, wie die Sinusfunktion oder Logarithmusfunktionen.

Liegen also wenige korrekte Daten vor, können die Werte dazwischen durch Interpolation gewonnen werden. In vielen Anwendungen von Interpolationsverfahren wird behauptet, dass durch Interpolation neue Daten aus bestehenden Daten hinzugewonnen werden. Dies ist aber falsch. Durch Interpolation (wie bereits oben beschrieben) kann nur der Verlauf einer kontinuierlichen Funktion zwischen bekannten Abtastpunkten abgeschätzt werden. Diese Abschätzung basiert meist auf der Annahme, dass der Verlauf einigermaßen glatt ist, was in den meisten Fällen zu plausiblen Resultaten führt. Die Annahme muss aber nicht notwendigerweise zutreffen. Außerdem muss die Interpolationsmethode sorgfältig gewählt werden, damit die Realdaten möglichst gut abgeschätzt werden und sich keine Widersprüche zu Grundvoraussetzungen ergeben.

Lineare Interpolation

Lineare Interpolation ist sehr schnell und einfach und wird in der Praxis wohl am häufigsten benutzt. Der Grundgedanke ist, dass man den Funktionsgraph zwischen zwei Stützstellen durch eine lineare Funktion ersetzt und die Zwischenwerte dadurch approximiert.

Aus der Abbildung 3.5 entnimmt man, dass $\frac{y_p - y_1}{x_p - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$ gilt. So erhält man die Interpolationsformel für die y-Koordinate an einer Stelle x_p:

$$y_p = y_1 + \frac{(y_2 - y_1)(x_p - x_1)}{x_2 - x_1}.$$

SVM-Interpolation

Alternativ kann die *Support Vector Machine* für die Bestimmung eines Funktionswertes eingesetzt werden. In diesem Fall interpoliert man mit Hilfe von der erlernten SVM-Funktion $f(x)$ (Abbildung 3.6).

Die Interpolationsformel für die y-Koordinate an einer Stelle x_p ist im Falle der SVM-Interpolation einfach:

$$y_p = f(x_p).$$

3.6 Zusammenfassung

In diesem Kapitel wurde der Begriff des maschinellen Lernens erklärt. Aufbauend darauf wurden die Grundlagen der Verfahren und Technologien vorgestellt, die später im Rahmen dieser Arbeit analysiert bzw. verwendet werden. Die Prognoseverfahren werden für den Entwurf des Prognosemoduls (s. Kapitel 5) benötigt.

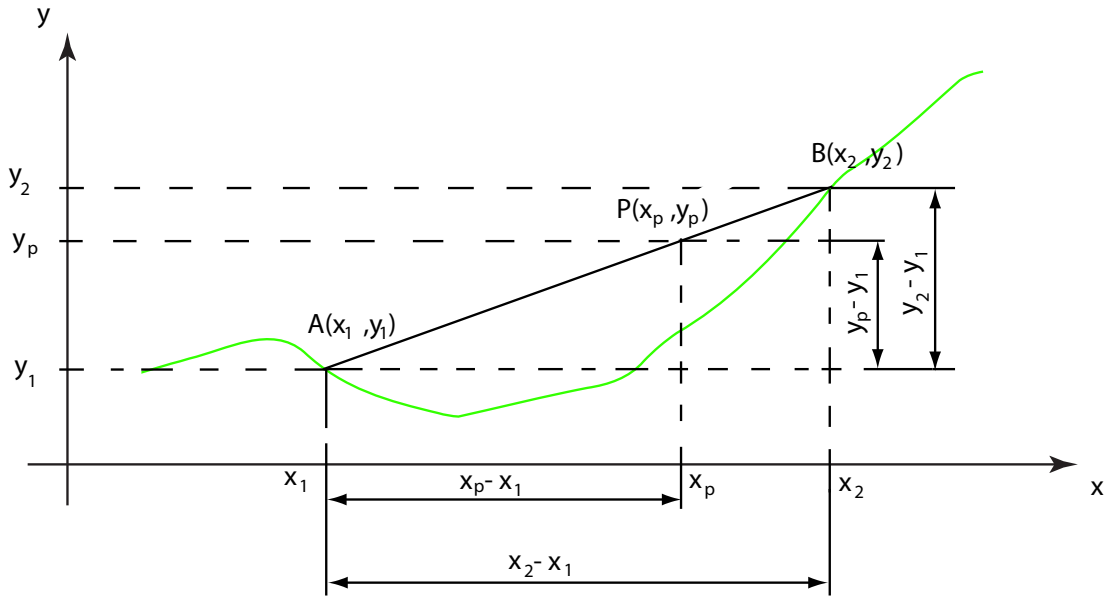


Abbildung 3.5: Lineare Interpolation: aus den Koordinaten bekannter Punkte A und B soll die y-Koordinate des Punktes P ermittelt werden.

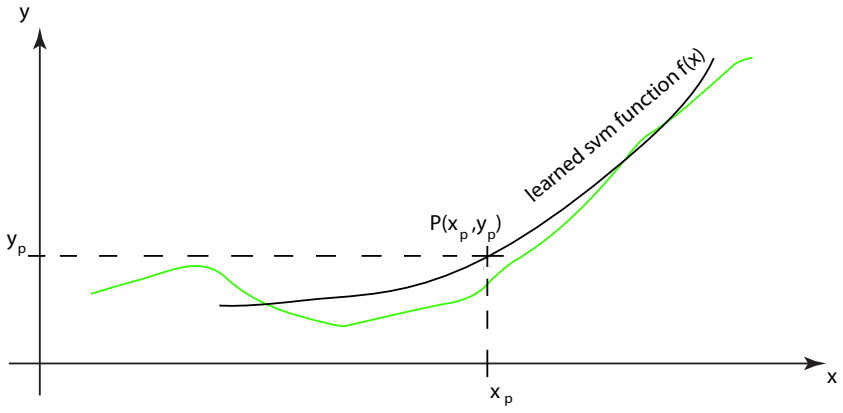


Abbildung 3.6: SVM-Interpolation: aus der erlernten SVM-Funktion $f(x)$ soll die y-Koordinate des Punktes P ermittelt werden.

Kapitel 4

R-Project

Um die Aufgaben dieser Arbeit zu bewältigen, kann man die bereits in Kapitel 3 erwähnten Prognoseverfahren selbst implementieren und nachher, nachdem die entsprechenden Anwendungstests durchgeführt worden sind, die geeigneten Verfahren auswählen und die Ergebnisse ihres Einsatzes zur Anzeige bringen (in Form von Grafiken, Tabellen etc.). Dies ist aus mehreren Gründen nachteilig:

- Aufwändige Programmierung der Visualisierung
- Aufwändige Programmierung der Prognoseverfaheren
- Mühsam implementierte Verfahren werden evtl. später doch nicht gebraucht.

Alternative verschafft das Programmpaket R-Project. R-Project ist eine Software für statistische Berechnungen und Auswertungen. Das Programm ist sowohl für ökonometrische Analysen als auch für graphische Darstellungen geeignet und steht als *free software* auf dem *Comprehensive R Archive Network* (CRAN)[1] zum Download zur Verfügung. Ausführliche Informationen über die Entwicklungen und Autoren sind auf der offiziellen Homepage[2] zu finden.

R-Project verfügt sowohl über bereits implementierte Prognoseverfahren als auch über hervorragende Auswertungmöglichkeiten. Er ist in der aktuell eingesetzten Version (1.9.0) nicht Java-fähig, so dass eine entsprechende Schnittstelle entworfen werden muss. Im Rahmen dieser Arbeit verwendet man R-Software als externes Programm, dessen Aufgabe ist, Berechnungen bzw. Auswertungen der übergebenen Daten durchzuführen und die Ergebnisse an das PAM-System zu liefern.

Als Erstes wird betrachtet, wie R-Project die Daten importiert. Danach wird erklärt, auf welche Weise die Daten ausgewertet werden und wie die Ergebnisse dargestellt werden. Schließlich wird das Konzept der Schnittstelle zwischen R-Project und dem PAM-System veranschaulicht. Das zu integrierende Prognosemodul braucht diese Schnittstelle zur Durchführung der Prognoseberechnungen und Visualisierung der Ergebnisse.

4.1 Datenimport

Die Daten können eingegeben oder eingelesen werden. Das Einlesen von Daten kann von einer Datei oder von einer Datenbank geschehen. Da es in unserem Fall um große Datenmengen geht, greift R-Project direkt auf die Datenbank zu. Zum Import aus relationalen Datenbanksystemen steht eine Reihe von Treibern zur Verfügung, die Daten aus mSQL, MySQL, Oracle und Postgres importieren können. Zusätzlich stehen mit dem Paket *RODBC* alle Datenquellen zur Verfügung, für die ein ODBC-Treiber existiert.

[1] siehe http://cran.r-project.org/
[2] siehe http://www.r-project.org/

4.2 Ökonometrisches Modell

Nachdem die Daten zur Analyse bereitstehen, wird zunächst auf Basis einer theoretischen Vorüberlegung ein ökonometrisches Modell konstruiert. Dieses ist im einfachsten Falle ein lineares und mit der Funktion *lm()* darstellbares Modell. Die Funktion schätzt auf Grundlage der Methode kleinster Quadrate (KQ) [8, 28] die beste Anpassungsgerade[3] an die Daten. Neben des linearen existieren auch weitere Modelle, die in Kapitel 5 vorgestellt und analysiert werden.

4.3 Vorhersage

Neben dem Ziel des Verstehens der Realität durch Modelle kann man auf der Grundlage des Modells Prognosen erstellen. Nachdem die Effekte der Einflußgrößen geschätzt wurden, kann nun bei Kenntnis der Ausprägung bzw. erwarteter Entwicklung der erklärenden Variable der Verlauf der zu erklärenden Größe vorhergesagt werden. Dies geschieht mit der Funktion *predict()*.

4.4 Grafische Darstellung

R ist in der graphischen Darstellung sehr leistungsfähig. Mit dem Befehl *x11()* öffnet sich ein neues Fenster zur grafischen Ausgabe. Falls mehrere Grafiken gleichzeitig dargestellt werden sollen, bietet die Funktion layout() viele Möglichkeiten zur Unterteilung des Grafikfensters. Zur weiteren Analyse sollten auch grafische Elemente zur Interpretation der Ergebnisse herangezogen werden. Mit dem Befehl *plot()* sind bereits einige Plots vordefiniert. Weiterhin stellen u.a. die Funktionen *boxplot()*, *piechart()*, *sunflowerplot()*, *hist()*, *dotplot()* zum Teil nützliche Ergänzungen dar. Zur Verfeinerung ist es mit Hilfsmitteln wie z.B. *points()*, *lines()*, *title()*,*legend()* möglich, dem Diagramm weitere Elemente hinzuzufügen.

4.5 *Support Vector Machine* in R-Project

Als *Support Vector Machine* benutzt R-Project die Implementierung von *libsvm*. Die Entwicklung von *libsvm* kann man bis zu SVM^{light} zurückverfolgen. SVM^{light} ist die Implementierung von Vapnik's *Support Vector Machine* [30]. Die Optimierungsalgorithmen, die in SVM^{light} benutzt wurden, sind in [17, 18] beschrieben. Die schlechte Performanz von SVM^{light} in etwas komplizierteren Fällen führte zur Entwicklung von *bsvm*, die ihrerseits auch Nachteile mit sich brachte: sehr komplex, außerdem – Benutzung von Optimisierungslösungen von Drittanbietern. Das war der Grund für die Entstehung von *libsvm*. *libsvm* gewann bereits mehrere Wettbewerbe, wie z.B.:

- EUNITE 2001
- IJCNN Challenge 2001

R-Project bietet eine Schnittstelle zu *libsvm* über ein Package *e1071*[4](benannt nach der Statistik-Abteilung der TU Wien). Dabei werden die im Rahmen dieser Arbeit benötigten Funktionen zu Trainieren (*training*) und Vorhersagen (*predicting*) unterstützt.

4.6 Schnittstelle zum PAM-System

Das R-Project arbeitet die an es übergebenen Befehle sequentiell ab. Man kann diese Befehle entweder in der grafischen Benutzeroberfläche von dem R-Project eingeben oder man speichert

[3] Eine Gerade entsteht nur für Modelle mit einer Regressorvariable.
[4] siehe http://cran.r-project.org/

alle Befehle in einer R-Datei (z.B. *test.r*) und startet die Abarbeitung dieser Befehle, indem man per Kommandozeile das ausführbare Programm von dem R-Project *Rterm.exe* mit der R-Datei als Übergabeparameter aufruft. In der Abbildung 4.1 ist dargestellt, wie die Informationen von einem Client (Benutzer-PC) an das PAM-System fließen, das seinerseits den R-Project benutzt, um bestimmte Arbeiten durchzuführen.

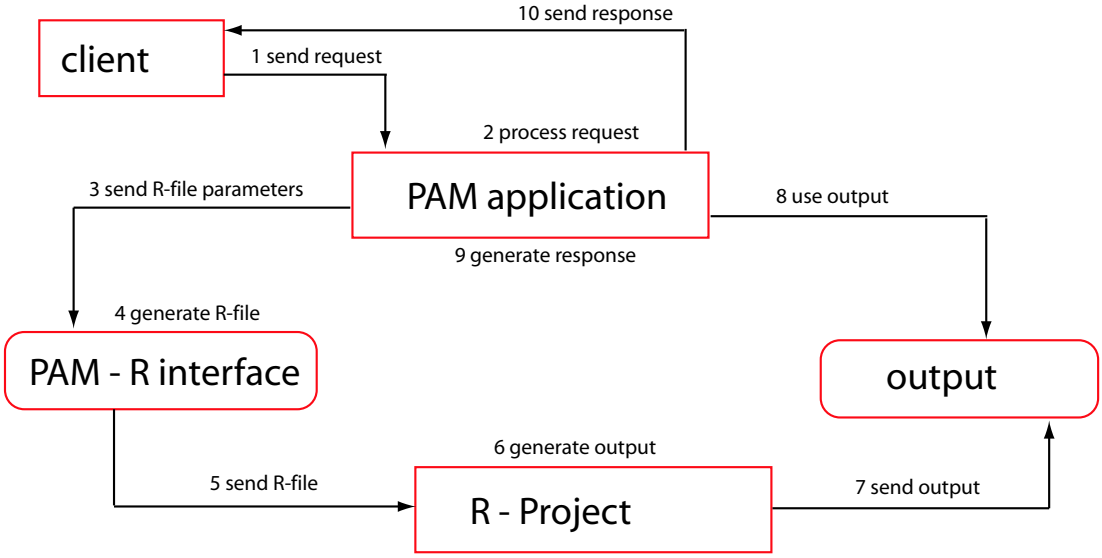


Abbildung 4.1: PAM-Schnittstelle zu R-Project.

Nachdem ein Client eine Anfrage an das PAM-System gesendet hat (z.B. Anzeige des Lebenszyklus eines Feldgerätes, *send request*), wird diese Anfrage von der Anwendungslogik im PAM-System bearbeitet (*process request*). Das PAM-System erkennt, dass die Bearbeitung der Anfrage auch Hilfe von R-Project erfordert, und sendet die für das R-Project relevanten Informationen an die Schnittstelle zu R-Project (*send R-file parameters*). Aus denen generiert die Schnittstelle die R-Datei (*generate R-file*) und startet R-Project mit dieser R-Datei (*send R-file*).

R-Project wird gestartet und arbeitet die R-Datei ab. Währenddessen wartet das PAM-System auf das R-Project. Die Ausgabe (Arbeitsergebnis von R-Project, *generate output*), wird in einer Datei abgelegt (*send output*), womit die Arbeit von R-Project an dieser Stelle auch beendet ist. Anschließend benutzt das PAM-System die Ausgabe von R-Project (*use output*) und generiert mit Hilfe von Servlets die Antwort (*generate response*), die an den Client zurückgeschickt wird (*send response*).

4.7 Zusammenfassung

In diesem Kapitel wurde ein kurzer Einblick in die R-Project Software vermittellt. Anschließend wurde das Zusammenspiel von PAM-System und R-Project gezeigt. Daraus ist ersichtlich, dass R-Project hervorragende Werkzeuge zu statistischen Auswertungen sowie Möglichkeiten zu deren grafischen Darstellung bietet. Diese komfortablen Analysetools werden als externe Programme für die Auswertungen in dem PAM-System eingesetzt. Somit sieht die Anwendung von R-Project im Rahmen dieser Arbeit sehr vielversprechend aus.

Kapitel 5

Prognosemodul

Was erwartet ein Kunde zu sehen, wenn er mitgeteilt bekommt, das sein PAM-System jetzt in der Lage ist zu prognostizieren und die Prognosen auszuwerten? Die Anwort ist in den meisten Fällen immer gleich: neben der sehr einfachen Handhabung (jeder Techniker sollte die Prognose- und Auswerteprogramme bedienen können) ist auch Schnelligkeit des Programms und Zuverlässigkeit der gelieferten Prognosen gefragt. All diese Aspekte sollten entsprechend berücksichtigt werden.

Die einfache Handhabung ist durch die vordefinierten und bereits in Kapitel 4 beschriebenen Schnittstellen gewährleistet. Nachdem man ein Feldgerät ausgewählt hat, kann die Prognose mit einem Mausclick auf die entsprechende Schaltfläche erstellt werden. Dagegen können Aspekte wie Schnelligkeit und vor allem Zuverlässigkeit nicht so einfach nachgewiesen werden. Von hier aus ist gedacht, die bereits in Kapitel 3 vorgestellte Prognoseverfahren zuerst an den Kundendaten einzusetzen. Die daraus resultierenden Ergebnisse lassen dann hoffentlich die Verfahren herauskristallisieren, die sich für die Kundendaten besonders eignen.

5.1 Analyse der Prognoseverfahren

In der Ausgangssituation hat man eine sehr große Menge von Messungen. Jede Messung kann grob durch zwei Werte repräsentiert werden:

- t – Anzahl der Stunden, die das Feldgerät zum Zeitpunkt der Messung geleistet hat
- $p(t)$ – Leistung des Gerätes bei der Anzahl der geleisteten Stunden t.

In Anbetracht dessen kann jede Messung M als ein Punkt im zweidimensionalen Raum abgebildet werden:

$$M(t, p(t)).$$

In der Abbildung 1.1 wurde bereits das Lebeszyklus eines Feldgerätes dargestellt. Darin sieht man gewisse Abhängigkeit zwischen t und $p(t)$. Die Idee ist, diese Abhängigkeit zu erlernen und später für die Prognose zu benutzen.

Das Erlernen der Abhängigkeit aus den Messungen eines einzelnen Feldgerätes ist aus zwei Gründen nicht sinnvoll. Zum Ersten, es existieren zu wenig Messungen pro Feldgerät, zum Zweiten muss das Arbeitszeit-Leistungsverhältnis eines ausgewählten Feldgerätes nicht unbedingt für alle anderen Feldgeräte desselben Typs massgebend sein: Es könnte durchaus ein Ausreißer sein, der das gesamte Bild verfälscht.

Das Problem kann man umgehen, indem man die Messungen aller Feldgeräte eines Typs nimmt. Dadurch trifft man die Aussage über *alle* Messungen, die zum gegebenen Feldgerätetyp gehören.

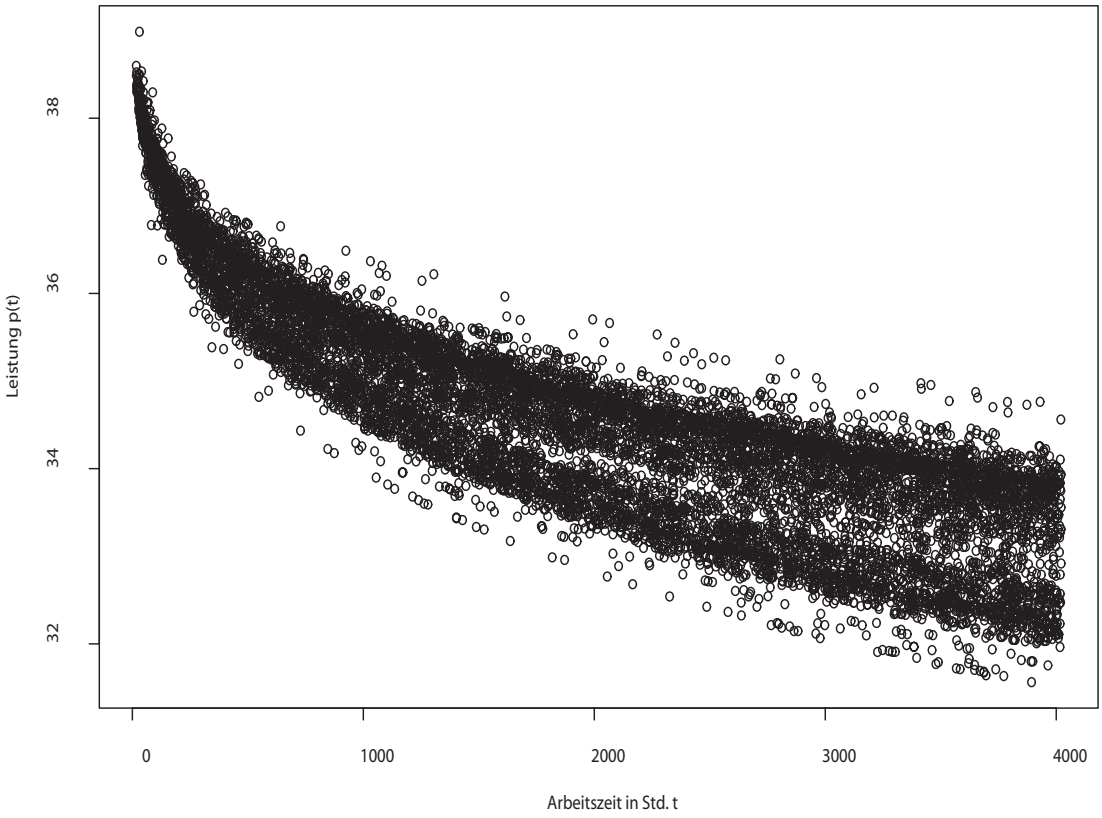


Abbildung 5.1: Messungen aller Feldgeräte eines bestimmten Typs (hier – *module type I*).

Aus der Abbildung 5.1 entnimmt man, dass alle Feldgeräte eines bestimmten Typs sich im Laufe ihrer Arbeitszeit ungefähr gleich verhalten. Allerdings sieht man nicht, welche Messung zu welchem Feldgerät gehört, was sofort die Frage aufwirft, ob die Leistung der Feldgeräte wirklich kontinuierlich abfällt. Denn es kann auch sein, dass die Leistung infolge anderer, bis jetzt noch nicht berücksichtigter Faktore, ab und zu auch ansteigt (Abbildung 5.2).

Durch die stichprobenartige Betrachtung der Lesitungskurven bei den Feldgeräten ist aber bestätigt worden, dass es keine vorübergehende Ansteigung der Leistung gibt. Die seltenen Fälle, wo sie tatsächlich, gleichwohl unbedeutend klein, vorkam, sind auf die ungenauen Messungen zurückzuführen. Somit können alle Messungen eines Feldgerätetyps benutzt werden, um daraus eine Funktion zu erlernen, die das Verhalten der Feldgeräte dieses Typs beschreibt. Diese Funktion wird für die Prognose benötigt. Bevor man aber mit dem Anwenden der Prognoseverfahren anfängt, sollte man sich zusätzlich noch über die Ausreißer-Problematik Gedanken machen. Ausreißer sind in unserem Fall die Feldgeräte, die z.B. aufgrund des Materialfehlers ziemlich schnell kaputt gehen. Ihre Leistung sinkt im Laufe der Zeit viel stärker als bei allen anderen Geräten, was wiederum auf die Zuverlässigkeit der Prognose Auswirkungen haben kann. Dieser Aspekt wird im nächsten Abschnitt ausführlicher erläutert.

5.1.1 Die *Support Vector Machine*

Wie bereits in Kapitel 4 beschrieben, werden SVM-Prognoseverfahren unter R-Project benutzt. Dies ist bequem, denn R-Project wird in Rahmen dieser Arbeit auch für die Durchführung der Auswertungen und Visualisierung der Ergebnisse eingesetzt. Der Ablauf eines SVM-

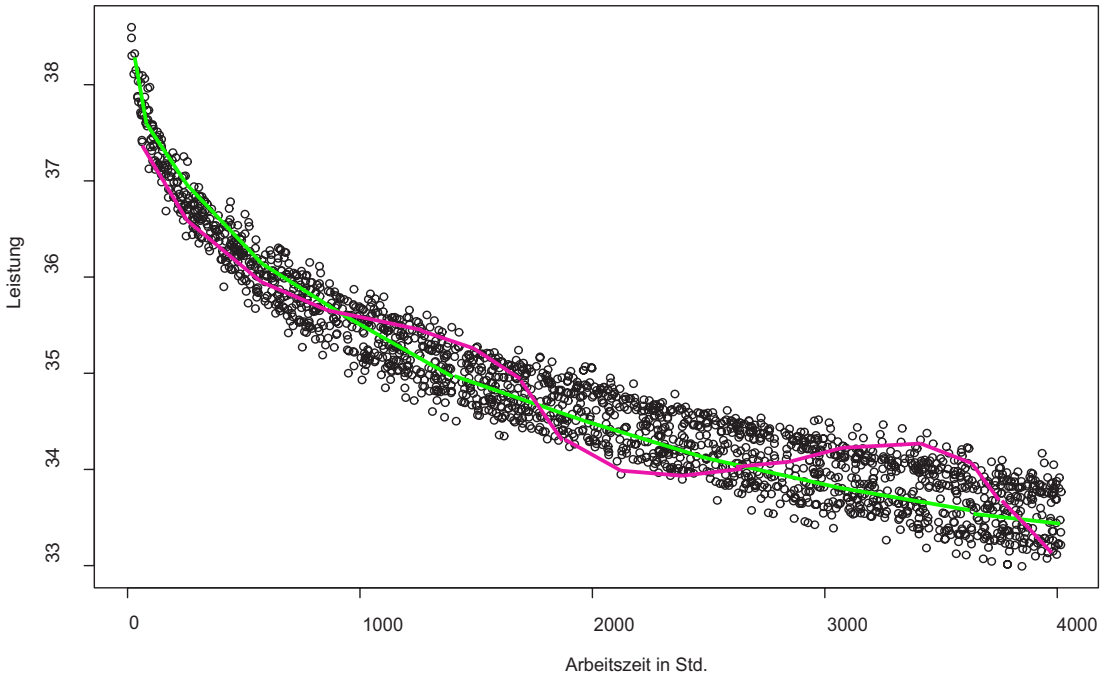


Abbildung 5.2: Mögliche Leistungsverläufe der Feldgeräte.

Prognoseverfahrens im Rahmen dieser Arbeit ist in der Tabelle 5.1 dargestellt.

Nr.	Aktion	Befehl
1	RODBC-Bibliothek laden	library(RODBC)
2	Verbindung zu der Datenbank herstellen	channel <- odbcConnect(<parameters>)
3	Daten aus der Datenbank holen	queryResult <- sqlQuery(channel, <query>)
4	Daten in ein *data frame* packen	dataFrame <- data.frame(queryResult)
5	*data frame* anbinden	attach(dataFrame)
6	Einzelne Spalten den Variablen x und y zuweisen	x <- <Spalte1> y <- <Spalte2>
7	Verbindung zu der Datenbank schließen	odbcClose(channel)
8	e1071-Bibliothek laden	library(e1071)
9	SVM-Modell bauen	svmmodel <- svm(x, y)
10	Prognose erstellen	pred <- predict(svmmodel, x)

Tabelle 5.1: Ablauf einer Prognose mit Hilfe von SVM unter R-Project.

Standardparameter. Radialer Kernel

Der zentrale Punkt des oben dargestellten Ablaufs ist das Bauen eines SVM-Modells (Schritt 9). Beim Aufruf des entsprechenden Befehls *svm(x, y)* wird die SVM mit Standardparametern aufgerufen (Tabelle 5.2).

kernel	radial basis
type	eps-regression (ϵ-SVR)
γ	1
ϵ	0.1

Tabelle 5.2: Die Standardparameter des SVM-Aufrufs.

Man bekommt dadurch die Funktionsapproximation, die in Abbildung 5.3 dargestellt wird.

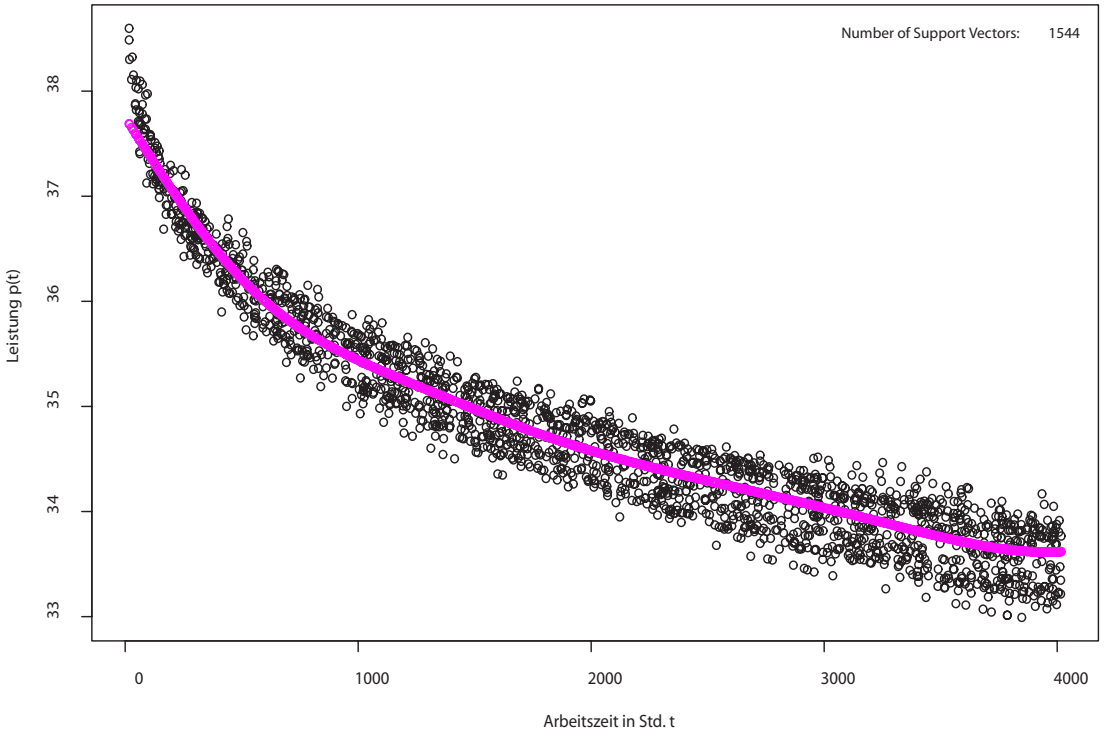


Abbildung 5.3: ϵ-SVR. Funktionsapproximation mit Standardparametern.

Aus der Abbildung 5.3 ist ersichtlich, dass die ε-SVR bereits eine gute Approximation des Leistungsverlaufs der Feldgeräte bietet. Die Tatsache, dass die approximierte Funktion etwas ungenau am Anfang verläuft, wodurch die Leistungswerte oberhalb von ca. 37,5 - 37,7 unberücksichtigt bleiben, ist zwar (wie später bei den Tests zu sehen ist) nicht sehr relevant, kann aber mit Hilfe des γ-Parameters angepasst werden (Abbildung 5.4).

Die Funktion in der Abbildung 5.4 approximiert den Leistungsverlauf der Feldgeräte im Anfang ihres Lebenszyklus etwas genauer, ist aber für den gesamten Verlauf zu genau angepasst (*overfitted*). Unter *overfitting* versteht man eine Überanpassung eines Modells. In manchen Fällen (z.B. wenn das Modell außer der Menge der trainierten Daten sonst niergendwo eingesetzt werden soll, was im Rahmen dieser Arbeit nicht zutreffend ist) ist solche Überanpassung kein Problem. Man möchte aber das trainierte Modell für die Prognose der zukünftigen Leistungswerte einsetzen, so dass die Überanpassung vermieden werden sollte. Der Standardwert für γ ist also der richtige.

Der ϵ-Parameter, wie bereits in Kapitel 3 beschrieben, sorgt für die Unempfindlichkeit der Funktion bei den Abweichungen. Die Abbildung 5.5 veranschaulicht die Funktionsverläufe bei verschiedenen Werten von ϵ.

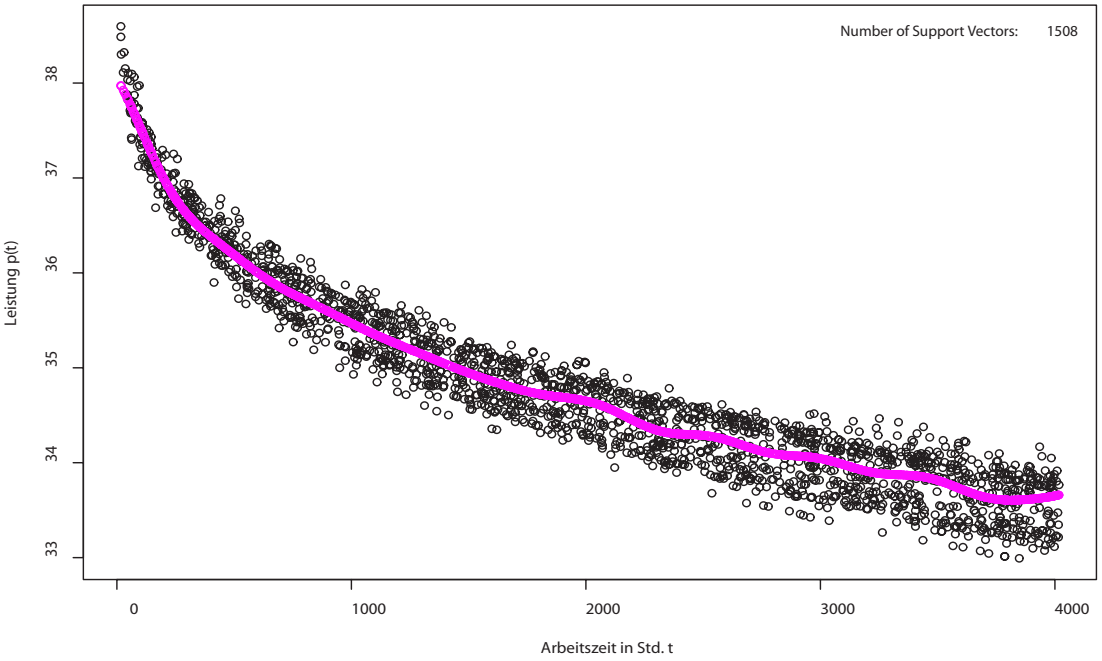


Abbildung 5.4: ϵ-SVR. Funktionsapproximation mit $\gamma = 10$ (radialer Kernel).

Je größer ist ϵ, desto linearer ist der Funktionsverlauf, desto ungenauer ist das Modell. Ein kleines ϵ reduziert den Trainingsfehler, führt aber zu erhöhter Modellkomplexität und größerer Anzahl von Support-Vektoren.

Die Anzahl der Support-Vektoren beläuft sich bei dem Beispiel in der Abbildung 5.3 auf 1544. Dies ist etwas zu hoch und hat negative Auswirkung auf die Rechenzeit. Deshalb wird versucht, die ν-SVR einzusetzen, denn die ν-SVR erlaubt es, die maximale Zahl der Ausreißer im Verhältnis zu der Gesamtzahl der Trainingsbeispiele fest vorzugeben. Damit erhält man die Möglichkeit, eine sehr präzise Toleranz zu erzielen, ohne sich vorher auf eine bestimmte Toleranz ϵ festlegen zu müssen. Damit ist das Verfahren in der Praxis robuster als die ϵ-SVR. Mehr darüber erfährt man aus [2].

Da die Anzahl der Ausreißer eigentlich sehr gering sein soll, soll die Funktionsapproximation mit der ν-SVR ähnliche Ergebnisse liefern wie die mit der ϵ-SVR. Die Anzahl der Support-Vektoren ist aber deutlich geringer geworden (Abbildung 5.6).

Damit sieht man, dass die ν-SVR für Aufgaben dieser Arbeit besser geeignet ist, als die ϵ-SVR. Zur Vervollständigung der Analyse werden noch die Untersuchungen mit dem polynomiellen Kernel durchgeführt.

Polynomieller Kernel

Der Aufruf des SVM-Befehls mit dem polynomiellen Kernel geschieht mit Parametern, die in Tabelle 5.3 angegeben sind.

Das Ergebnis dieses Aufrufs ist in der Abbildung 5.7 zu sehen. Die Aufrufe des SVM-Befehls mit dem polynomiellen Kernel und anderen Werten von den Parametern *degree* und *coef0* brachten ein noch schlechteres Ergebnis. Vollständigkeitshalber werden noch die Aufrufe des SVM-Befehls mit dem polynomiellen und radialem Kernel verglichen (Abbildung 5.9).

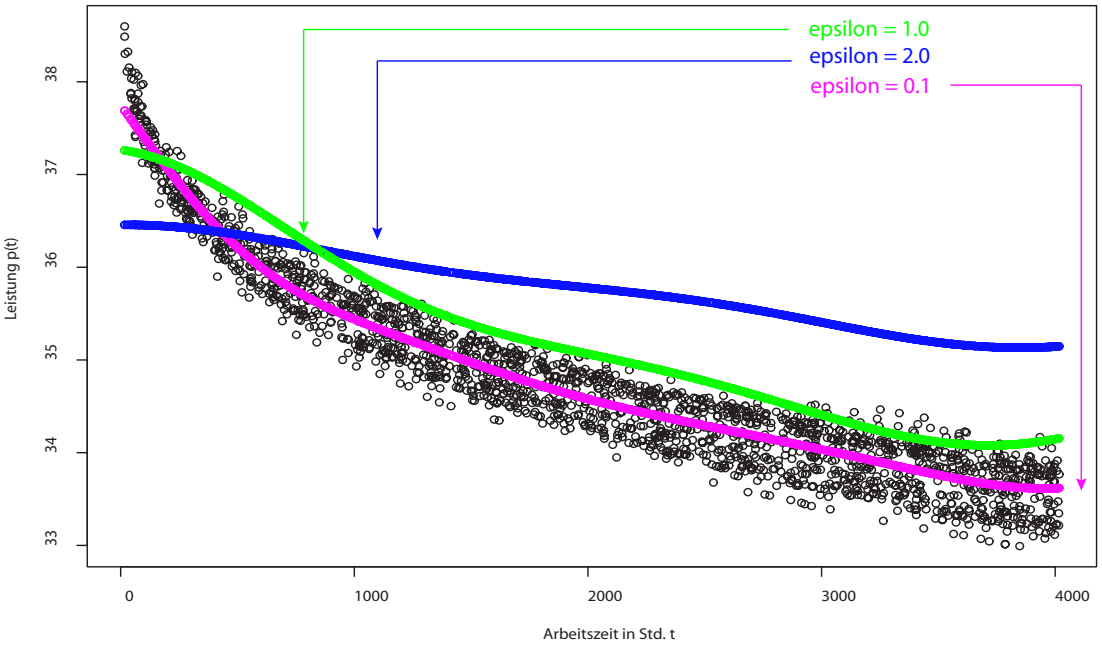


Abbildung 5.5: ϵ-SVR. Funktionsapproximation mit veschiedenen ϵ-Werten (radialer Kernel).

kernel	polynomiell
type	nu-regression (ν-SVR)
degree	2
γ	1
ν	0.1
coef0	1.5

Tabelle 5.3: Die Parameter des SVM-Aufrufs mit dem polynomiellen Kernel.

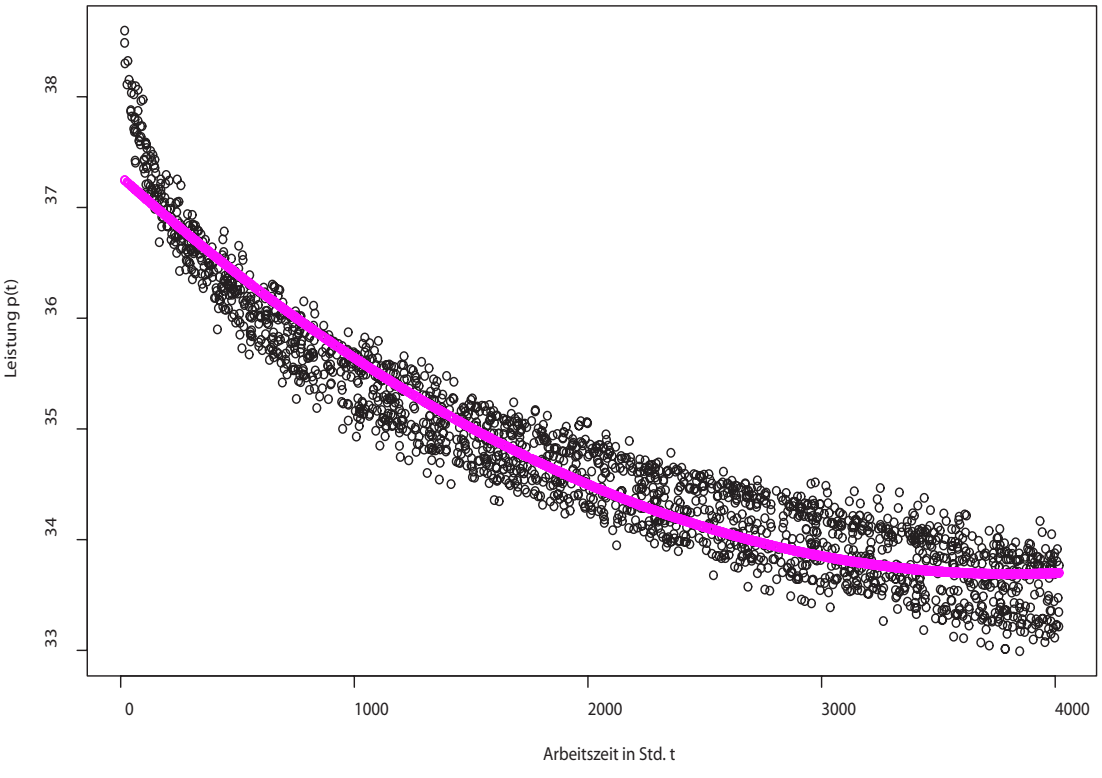


Abbildung 5.7: Funktionsapproximation mit polynomiellem Kernel.

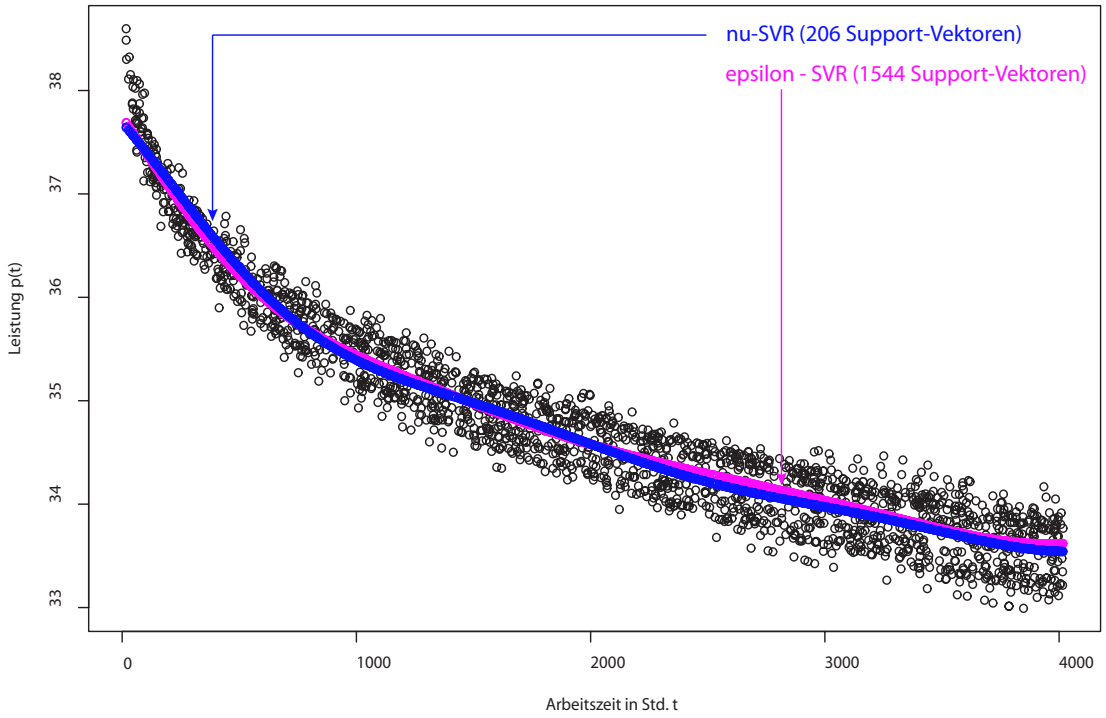


Abbildung 5.6: Funktionsapproximation mit ϵ- und ν-SVR im Vergleich (radialer Kernel).

Aus diesem Vergleich resultiert sich ein klarer Vorteil für den Einsatz eines ν-SVR-Prognoseverfahrens mi einem radialen Kernel (verglichen mit ϵ-SVR und polynomiellem Kernel). Die ν-SVR sorgt für eine kleine Anzahl (in dieser Arbeit – um ca. 7 Mal kleiner!) der Support-Vektoren und der radiale Kernel – für ein besseres Verhalten der Approximationsfunktion bei der Arbeitszeit der Feldgeräte unmittelbar am Anfang ihrer Einsätze.

5.1.2 k-Nächster-Nachbar

Der Einsatz von SVM approximiert den Leistungsverlauf der Feldgeräte in Laufe der Zeit ziemlich gut, was uns natürlich nicht verbietet, auch weitere Verfahren auszuprobieren. Zumal werden die überdurchschnittlich guten Feldgeräte (deren Leistung im Vergleich zu den anderen Geräten weit über die Garantiezeit nahezu konstant bleibt) von den SVM-Prognoseverfahren höchstwahrscheinlich zu pessimistisch und die überdurchschnittlich schlechten – zu optimistisch eingeschätzt. Ob dieser Aspekt in der Praxis eine bedeutende Rolle spielt, mag dahingestellt sein. Dennoch wird ein Verfahren gesucht, mit dem man die negativen Auswirkungen dieses Problems minimieren kann.

Beim k-Nächster-Nachbar Algorithmus geht es um die Bestimmung der Feldgeräte, deren Verhalten sich am meisten dem Verhalten des Gerätes ähnelt, für das die Prognose erstellt werden soll. Somit werden auch die Ausreißer fairer betrachtet, denn als ihre Nachbarn agieren die Feldgeräte, deren Messwerte am nähesten zu den der Ausreißer liegen, wodurch das Verhalten der Ausreißer durch das Verhalten anderer Ausreißer approximiert wird (Abbildung 5.8).

Für den k-Nächster-Nachbar Algorithmus sind folgende Parameter wichtig:

- k (Anzahl nächster Nachbarn)
- c_{lm} (Anzahl letzter Messungen)
- *type* (Typ der Interpolation zwischen den Messungen)

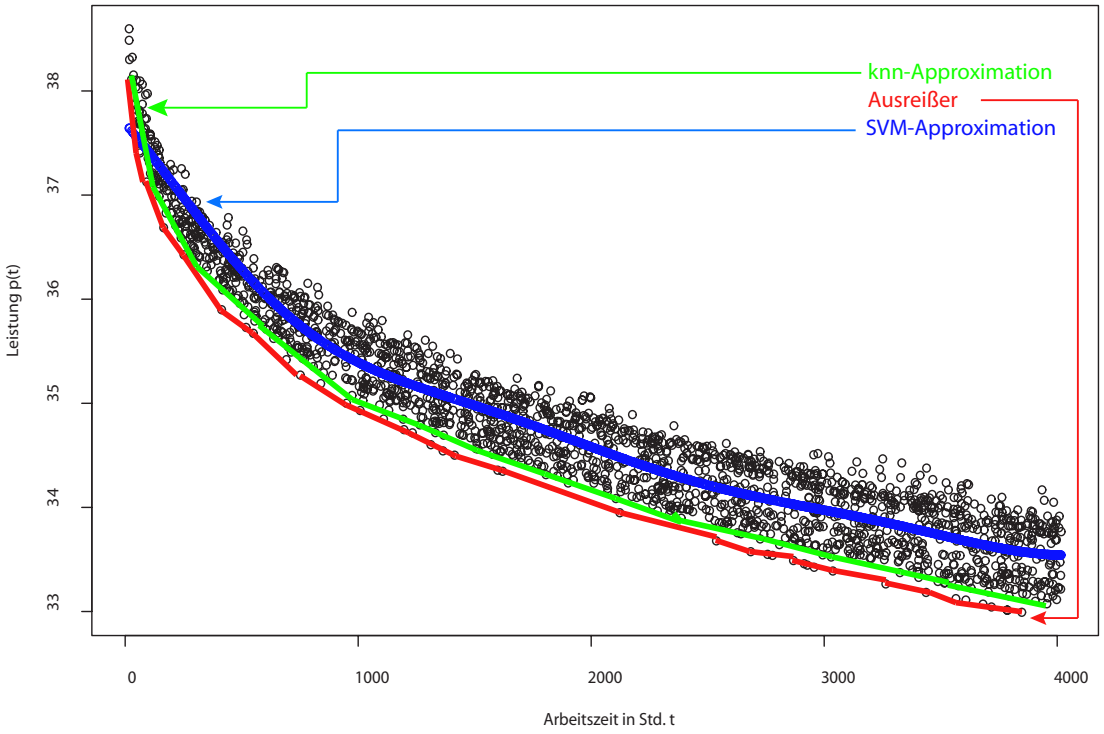


Abbildung 5.8: Funktionsapproximation mit SVM und k-Nächster-Nachbar (Zur Vereinfachung: k = 1) im einem Ausreißer-Fall.

Anzahl nächster Nachbarn

Anzahl nächster Nachbarn ist ein unentbehrlicher Parameter. Er bestimmt die Anzahl der zu ermittelnden Feldgeräte, dessen Verhalten sich am meisten dem Verhalten des Feldgerätes ähnelt, für das die Prognose erstellt werden soll. Eine zu kleine Anzahl der nächsten Nachbarn kann zu ungenauen Prognose führen (Abbildung 5.10), wobei es allerdings sein kann, dass eine große Anzahl der nächsten Nachbarn auch nicht unbedingt sehr viel zu der Zuverlässigkeit der Prognose beiträgt. Es steht allerdings außer Frage, dass die Rechenzeit mit zunehmender Anzahl der nächsten Nachbarn auch zunimmt. Die Abhängigkeit der Zuverlässigkeit der Prognose von der Anzahl nächster Nachbarn wird später durch entsprechende Tests erforscht.

Anzahl letzter Messungen

Für die Bestimmung der Nachbarn eines Feldgerätes ist die Anzahl der zuletzt bei diesem Feldgerät durchgeführten Messungen c_{lm} von Bedeutung. Denn je größer ist die Anzahl der Messungen (die später den Messungen der anderen Geräte zu denselben Zeitpunkten gegenübergestellt werden), desto genauer kann man die Leistungskurven des Feldgerätes und die

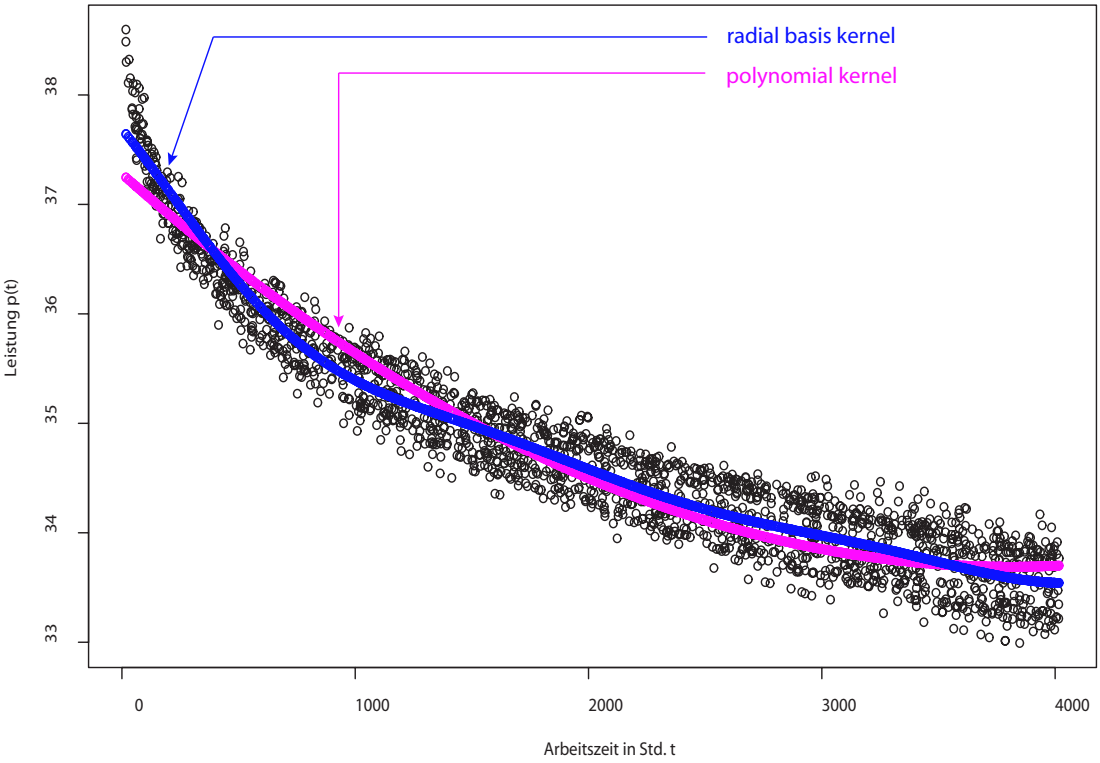


Abbildung 5.9: Funktionsapproximation mit radialem und polynomiellem Kernel im Vergleich.

seiner potentiellen Nachbarn vergleichen. Es kann aber auch sein, dass zu große Anzahl der Messungen bedeutend mehr Rechenzeit in Anspruch nimmt.

In der Abbildung 5.11 sieht man, wie unzuverlässig es sein kann, den nächsten Nachbarn anhand nur einer letzen Messung zu bestimmen. Wählt man einen ungünstigen Messzeitpunkt aus, wo der Verlauf des Feldgerätes, für das die Prognose erstellt werden soll, kurzzeitig den Verlauf eines anderen Feldgerätes ähnelt (in diesem Beispiel – bei ca. 1600 Stunden Arbeitszeit), so kann es passieren, dass die Prognose total falsche Ergebnisse liefert (siehe bedeutenden Unterschied bei 4000 Stunden).

Interpolation

Jedes Feldgerät wird im Laufe seines Lebenszyklus getestet. Es ist aber in der Praxis unmöglich, an allen Feldgeräten in einem bestimmten Zeitpunkt gleichzeitig Messungen durchzuführen. Einerseits reicht die Systemperformance nicht, andererseits dürfen die Feldgeräte im Laufe ihres Einsatzes nicht für die Tests verwendet werden. Man braucht also ein Verfahren zur Ermittlung der Leistung der Feldgeräte in den Zeitpunkten, für die keine Messungen durchgeführt wurden (Abbildung 5.12). Die in Kapitel 3 vorgestellten Interpolationsverfahren werden diese Aufgabe übernehmen.

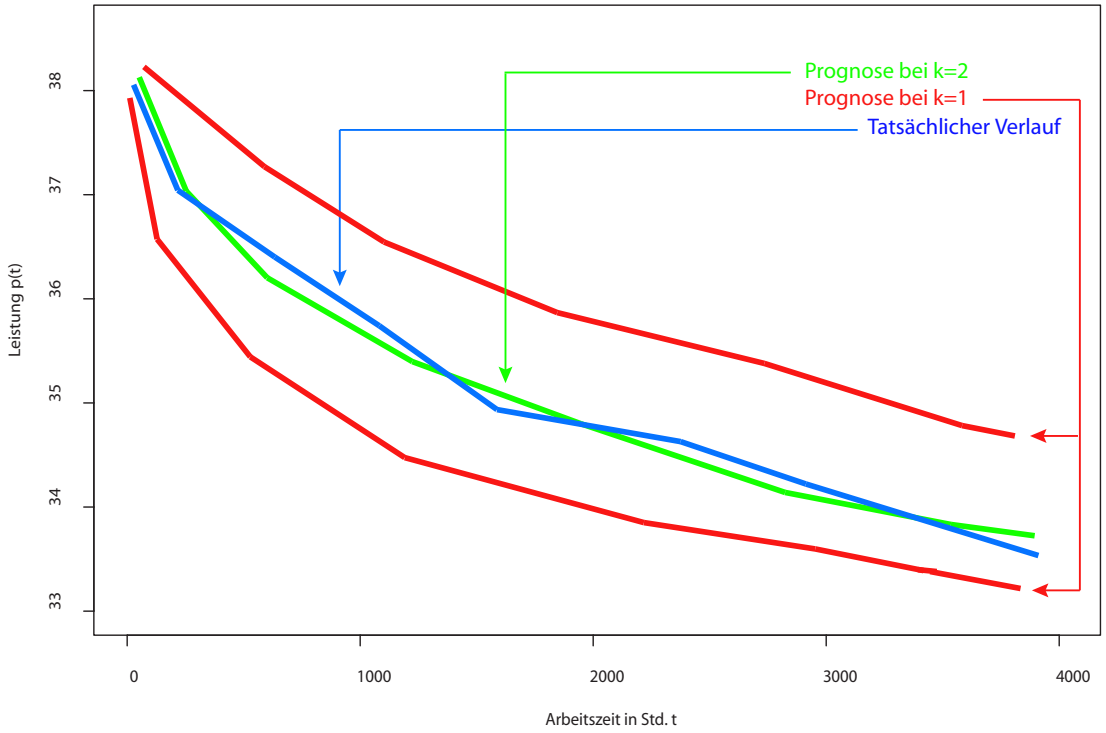


Abbildung 5.10: Nächster-Nachbar-Prognose bei $k = 1$ und $k = 2$: Nimmt man nur einen nächsten Nachbar (entweder obere oder untere rote Kurve), so ist die Approximation ungenauer als der Mittel von zwei nächsten Nachbarn (durch grüne Kurve dargestellt).

Der zuvor beschriebene k-Nächster-Nachbar Algorithmus lässt sich hier folgendermaßen anwenden:

- Wähle ein Feldgerät fd_{pred}[1], für das die Prognose erstellt werden soll.
- Wähle den Zeitpunkt der Prognose t_{pred}.
- Bestimme die k nächsten Nachbarn $fd_1, ..., fd_k$ zu fd_{pred}.
- Bestimme die Leistung p_i zum Zeitpunkt t_{pred} für jeden Nachbarn fd_i.
- $p_{pred} = \frac{\sum_{i=1}^{k} p_i}{k}$

Nachdem der Ablauf des k-Nächster-Nachbar Algorithmus in dem PAM-System klar geworden ist, kann man sich seiner praktischen Umsetzung unter eventueller Inanspruchnahme der SVM widmen.

5.2 Praktische Umsetzung der Prognosealgorithmen

Der zentrale Punkt dieser Arbeit ist der Argorithmus für die Prognose des Lebenszyklus eines Feldgerätes. Der Bediener des PAM-Systems stellt alle nötigen Parameter ein und sendet die Anfrage per Knopfdruck an den Server, wonach er erwartet, eine entsprechende Prognose angezeigt zu bekommen. Der Ablauf der Prognoseerstellung ist in der Abbildung 5.13 veranschaulicht.

[1] fd: für *field device*, pred: für *prediction*

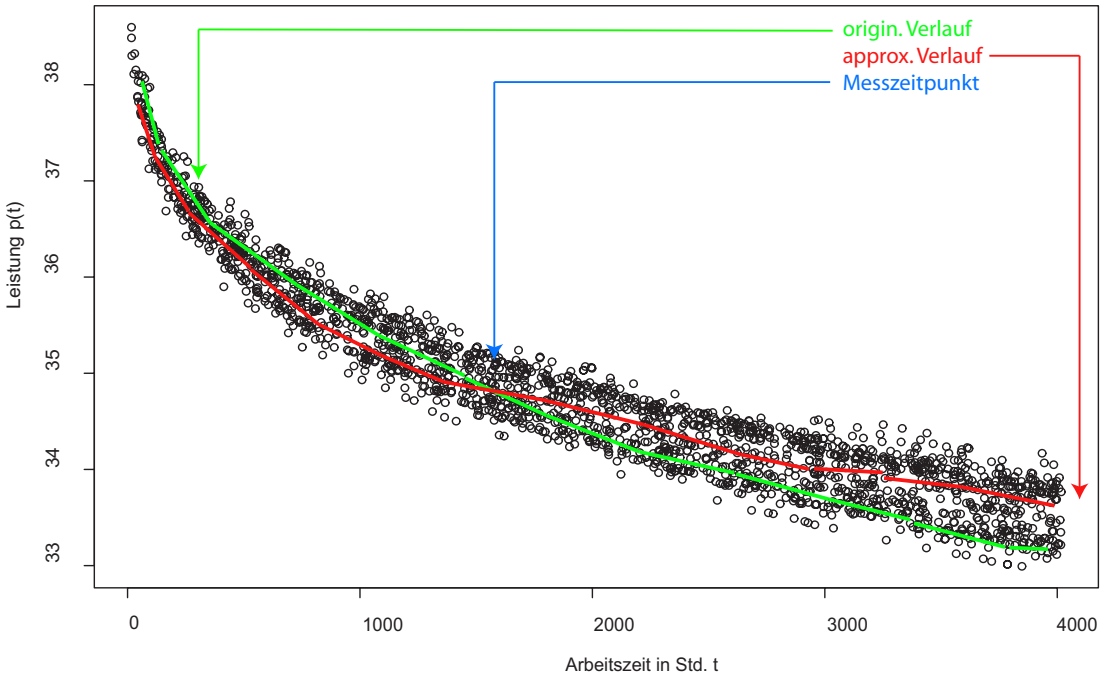


Abbildung 5.11: Nächster Nachbar eines Feldgerätes bei $c_{lm} = 1$.

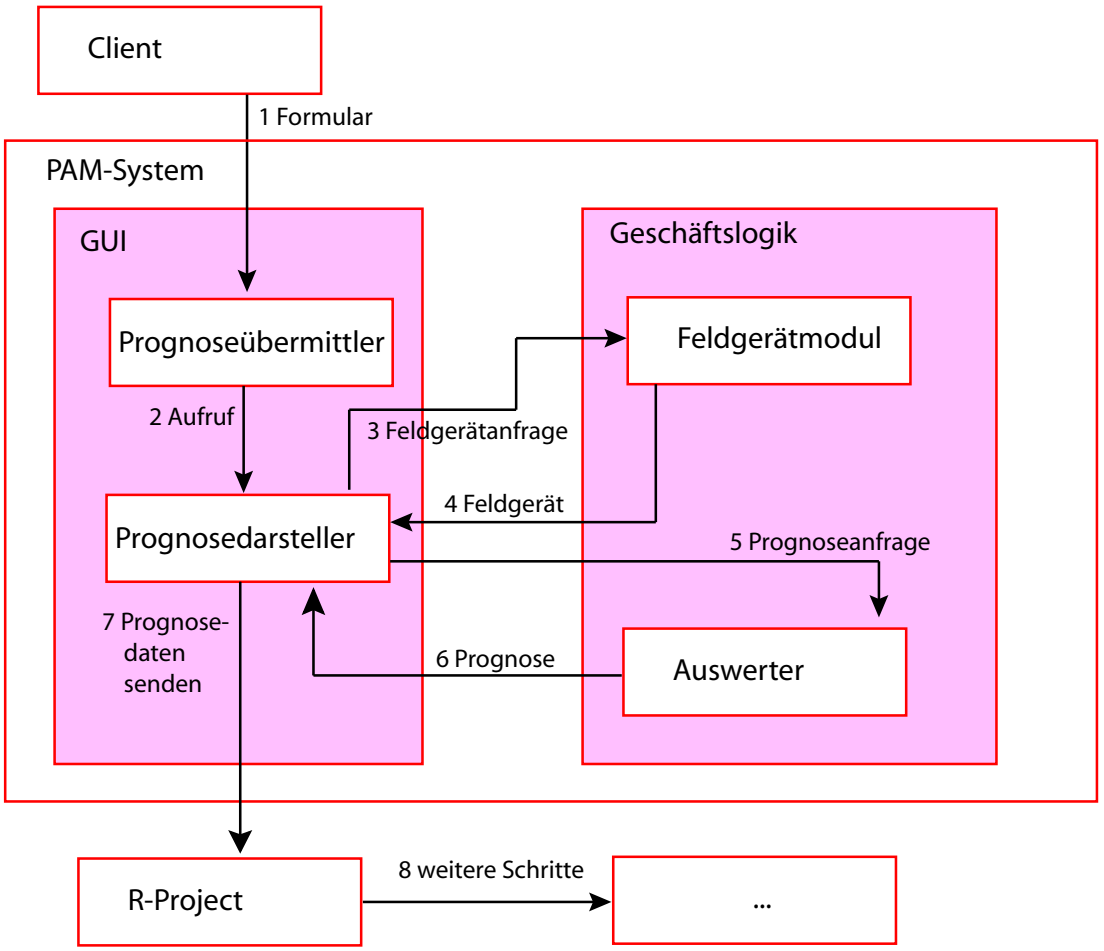


Abbildung 5.13: Ablauf einer Prognose-Abfrage in einem PAM-System bis zur Übergabe an die Schnittstelle zum R-Project.

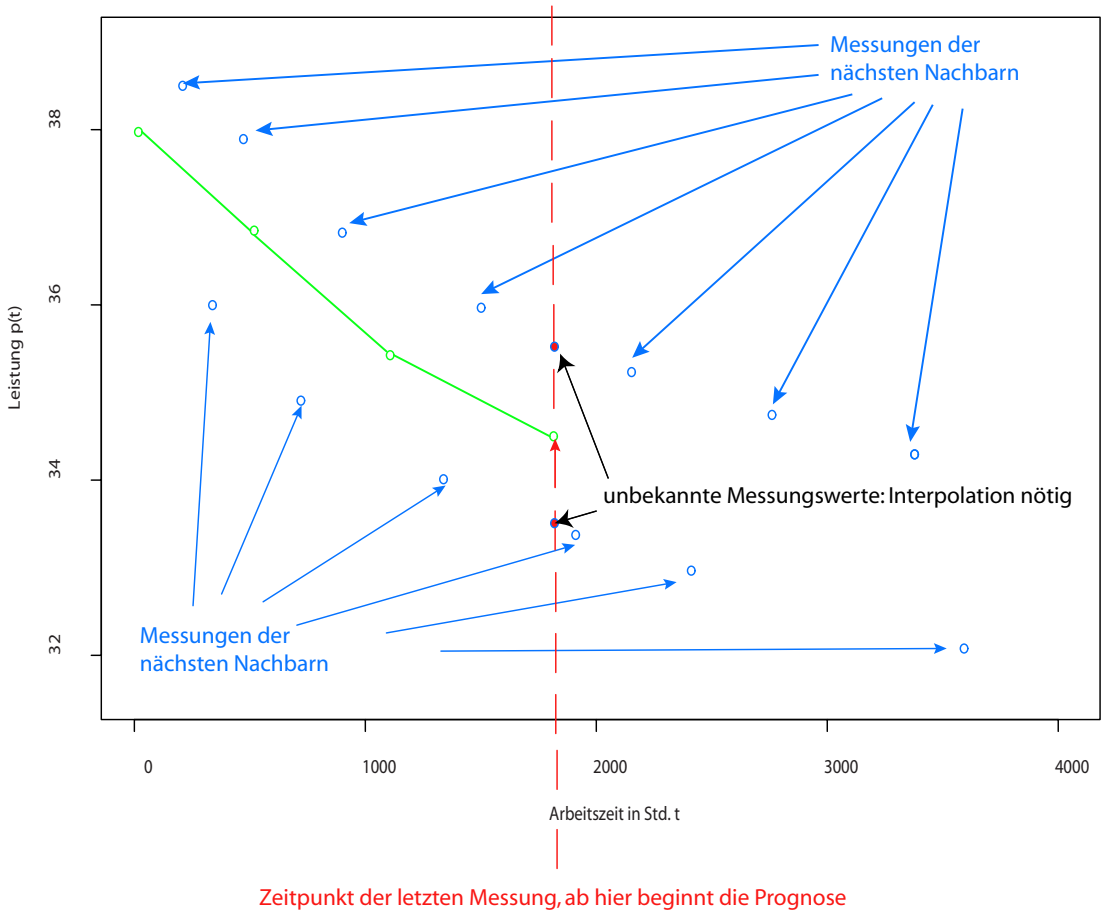


Abbildung 5.12: Notwendigkeit einer Interpolation zur Ermittlung unbekannter Messungswerte.

Nachdem der Bediener die Prognose-Parameter eingestellt (Schritt 1) und die Anfrage an das PAM-System gesendet hat, stellt der Prognoseübermittler fest, um welches Typ der Prognose es sich handelt, und leitet die Anfrage weiter (im Falle der Erstellung der Lebenszyklusprognose eines Feldgerätes – Prognosedarsteller (Schritt 2). Dieser übermittelt die ID des Feldgerätes, für das die Prognose erstellt werden soll (Schritt 3), an das Feldgerätemodul und holt sich das Feldgerät (Schritt 4). Das Feldgerät wird – zusammen mit den Parametern, die zuvor von dem Prognoseübermittler geliefert wurden, – an den Auswerter übergeben (Schritt 5). Darin befindet sich der eigentliche Prognosealgorithmus, der die Prognosedaten als Ergebnis zurückliefert(Schritt 6). Der weitere Verlauf – Übergabe der Prognosedaten an die R-Schnittstelle mit nachfolgender Erstellung und Ausführung der R-Datei – ist im Rahmen dieses Abschnittes nicht von Bedeutung. Viel wichtiger ist das in den Auswerter eingebaute Modul, das den Prognosealgorithmus beinhaltet.

Im folgenden Unterabschnitt wird der Algorithmus vorgestellt. Aufbauend auf diesem Algorithmus werden später dessen Erweiterungen ausgearbeitet, die die Zuverlässigkeit dieses Algorithmus analysieren.

5.2.1 Algorithmus für die Lebenszyklusprognose eines Feldgerätes

Aufgrund der Komplexität des Prognosealgorithmus sowie seiner engen Zusammenarbeit mit anderen Modulen ist es sinnvoll, den Algorithmus auf mehrere Ebenen zu projezieren und diese Ebenen einzeln zu betrachten. So werden zuerst die Parameter des Prognosealgorithmus vorgestellt. Danach betrachtet man die Variablen und Datenstrukturen, die in dem Algorithmus benutzt werden. Anschließend wird der grobe Ablauf des Algorithmus grafisch präsentiert, um einen Überblick über die Interaktion der Module zu verschaffen, die sich an diesem Algorithmus beteiligen.

Parameter

Für die Realisierung dieses Algorithmus werden Parameter benötigt, die in der Tabelle 5.4 aufgelistet sind.

fd_{pred}	UUID des Feldgerätes, für das die Prognose erstellt werden soll
$type$	Typ der Interpolation zwischen den Messungen. Linear oder SVM
t_{begin}	Anzahl der von einem Feldgerät geleisteten Arbeitsstunden am Anfang der Prognose
t_{end}	Anzahl der von einem Feldgerät geleisteten Arbeitsstunden am Ende der Prognose
k	Anzahl nächster Nachbarn
c_{lm}	Anzahl letzter Messungen

Tabelle 5.4: Die Parameter des Prognosealgorithmus.

Variablen und Datenstrukturen

Nachdem man sich mit den Parametern vertraut gemacht hat, die an den Prognosealgorithmus übergeben werden, steht man sozusagen am Eingangstor des Algorithmus und möchte nun endlich rein. Nachfolgend sind die Variablen und Datenstrukturen, die der Algorithmus verwendet, tabellarisch dargestellt (Tabelle 5.5). In der ersten Spalte steht der Name der Variablen bzw. Datenstruktur. Die zweite Spalte informiert über den Variablentyp bzw. den Aufbau einer Datenstruktur. In der dritten Spalte befinden sich kurze Erklärungen zu den Variablen bzw. Datenstrukturen. Die einfacheren Datenstrukturen werden zuerst aufgelistet und von komplexeren Datenstrukturen verwendet. Deshalb ist die Tabelle so aufgebaut, dass die Variablen und Datenstrukturen ihrer Komplexität nach dargestellt sind.

Name	Struktur	Bezeichnung
fd	s. *Module* im Datenmodell, Abbildung 2.2	Feldgerät
t	Zahl	Anzahl der von einem Feldgerät geleisteten Arbeitsstunden zum Zeitpunkt einer Messung
p	Zahl	Leistung eines Feldgerätes
id_{fd}	Text	ID eines Feldgerätes
d	Zahl	Abstand zwischen interpolierter Leistung eines Feldgerätes, das als potentieller Nachbar agiert, und Leistung des Feldgerätes, für das die Prognose erstellt werden soll, an einer gegebenen t
$fd-type$	s. *Moduletype* im Datenmodell, Abbildung 2.2	Feldgerätetyp
$id_{fd-type}$	Text	ID eines Feldgerätetyps
m	s. *Measurement* im Datenmodell, Abbildung 2.2	Messung eines Feldgerätes
$list_m^{fd}$	Liste aus m	Liste aller Messungen eines Feldgerätes
$list_m^{fd-type}$	Liste aus m	Liste aller Messungen aller Feldgeräte eines Feldgerätetyps
mm	id_{fd} $list_m^{fd}$	Struktur, in der sich alle Messungen eines Feldgerätes in Form einer Liste befinden, wobei jede Messung zusätzlich mit einer Feldgeräte-ID, zu der sie gehört, versehen ist
$list_{mm}$	Liste aus mm	Oberstruktur, die mehrere mm-Strukturen in Form einer Liste enthält
knn	id_{fd} t p d	Enthält Informationen über den Leistungswert bei einer bestimmten Anzahl der geleisteten Arbeitsstunden. Dieser Wert wird durch Interpolation berechnet. Ein weiterer Attribut ist der Abstand zwischen interpolierter Leistung eines Feldgerätes, das als potentieller Nachbar agiert, und Leistung des Feldgerätes, für das die Prognose erstellt werden soll
$list_{knn}$	Liste aus knn	Liste von knn, sortiert nach d

Tabelle 5.5: Variablen und Datenstrukturen des Prognosealgorithmus.

fd stellt ein Feldgerät dar. Dessen Attribute sind bereits in dem Datenmodell (Abbildung 2.2, Entität *Module*) erläutert worden. Eins dieser Attribute – id_{fd} – bezeichnet die Unique-ID

eines Feldgerätes und ist für die eindeutige Identifizierung des Feldgerätes zuständig. *ModuleType* bildet ein Feldgerätetyp ab. Jedes Feldgerät gehört zu genau einem Typ (s. Abbildung 2.2). Durch m wird eine Messung repräsentiert. t und p sind Attribute einer Messung. In t wird protokolliert, wie lange (in Stunden) das Feldgerät insgesamt bereits gearbeitet hat, bis die Messung stattgefunden hat, und p speichert den Leistungswert einer Messung zum Zeitpunkt t.

Die Datenstrukturen $list_m^{fd}$ und $list_m^{fd-type}$ sind in ihrem Aufbau identisch. Beide verwalten Mengen von Messungen. Der Unterschied liegt an verschiedenen Abfragen der Datenbank: in $list_m^{fd}$ befinden sich alle Messungen nur eines Feldgerätes (abgefragt über id_{fd}), während in $list_m^{fd-type}$ alle Messungen aller Feldgeräte gespeichert sind, die zu einem bestimmten Feldgerätetyp gehören (abgefragt über $id_{fd-type}$). Die Messungen in $list_m^{fd-type}$ sind nach Feldgeräte-ID sortiert, was später bei der Konvertierung dieser Messungen zu mm-Struktur von großer Bedeutung ist.

Etwas komplexer ist die Struktur mm. Sie besteht aus Liste aller Messungen eines Feldgerätes und der ID dieses Feldgerätes. Bekommt man mm übergeben, so ist über die Abfrage der id_{fd} sehr leicht herauszufinden, zu welchem Feldgerät die Liste von Messungen gehört. Die mm kann man ebenfalls in einer Liste speichern. Diese Liste heisst $list_{mm}$.

knn speichert die (t, p)-Paare. Dies ist für die einfachere Verwaltung dieser Werte in Rahmen der Ermittlung der k nächsten Nachbarn erwünscht. Zusätzlich enthält knn noch zwei Attribute: id_{fd} und d. Die id_{fd} ist – wie auch in allen anderen Fällen – zur Bestimmung der Zugehörigkeit eines knn zu einem Feldgerät notwendig. Die Aufgabe des Attributes d wurde oben in der Tabelle bereits erläutert, die entsprechende grafische Erläuterung ist aus der Abbildung 5.14 zu entnehmen.

Für die Bestimmung der k nächsten Nachbarn braucht man eine Struktur, die alle knn in sortierter Reihenfolge in sich speichert. Die Sortierung braucht man nach dem Abstand der Leistung eines Feldgerätes, das als potentieller Nachbar agiert, zu der Leistung des Feldgerätes, für das die Prognose erstellt werden soll. Werden die knn unter dieser Sortierbedingung (nach aufsteigendem d) in diese Struktur eingefügt, so braucht man später nur noch die ersten k Elemente zu betrachten. Die id_{fd}-Attribute dieser Elemente geben uns die k nächsten Nachbarn.

Bekannt sind jetzt alle Bausteine des Prognosealgorithmus in Form von Eingangsparametern, Variablen und Datenstrukturen. Ein gewisser Zusammenhang zwischen diesen Daten wurde auch schon vermittelt, ist allerdings zu oberflächlich. Das Gesamdbild des Algorithmus, das den Ablauf veranschaulicht, fehlt immer noch. Dies wird in dem folgenden Unterabschnitt nachgeholt.

Prognosealgorithmus auf der Methodenebene

Die Idee des Algorithmus ist, zuerst alle Messungen eines Feldgerätes zu ermitteln, für das prognostiziert werden soll; demnächst werden alle Messungen des Feldgerätetyps ermittelt, zu dem dieses Feldgerät gehört. Anschließend iteriert man durch die letzten Messungen des Feldgerätes, das für die Prognose ausgewählt wurde. Für jede seiner Messungen werden die Messungen anderer Feldgeräte gefunden, die zu derselben Anzahl der bereits geleisteten Stunden gemacht wurden. Falls es nicht der Fall ist – was allerdings sehr oft passieren kann – wird der gesuchte Leistungswert durch Interpolation der Leistungswerte benachbarter Messungen bestimmt. Die k von $(k + 1)$ Feldgeräte, deren Leistungswerte am nähesten zu dem Leistungswert des zu prognostizierten Feldgerätes liegen, sind seine k nächste Nachbarn (das erste Feldgerät ist das zu prognostizierte Feldgerät selbst). Der Algorithmus ist mit Hilfe der grafischen Darstellung leichter zu verstehen. Auf der Abbildung 5.15 wird der Datenfluss des Algorithmus gezeigt.

Der Prognosealgorithmus ist u.a. eine Interaktion zwischen zwei Komponenten der Geschäftslogik des PAM-Systems: der Auswertungslogik (zuständig für die Auswertungen und Prognosen), und der Messungslogik (stellt die Module zur Verwaltung der Messungen zur Verfügung). Im

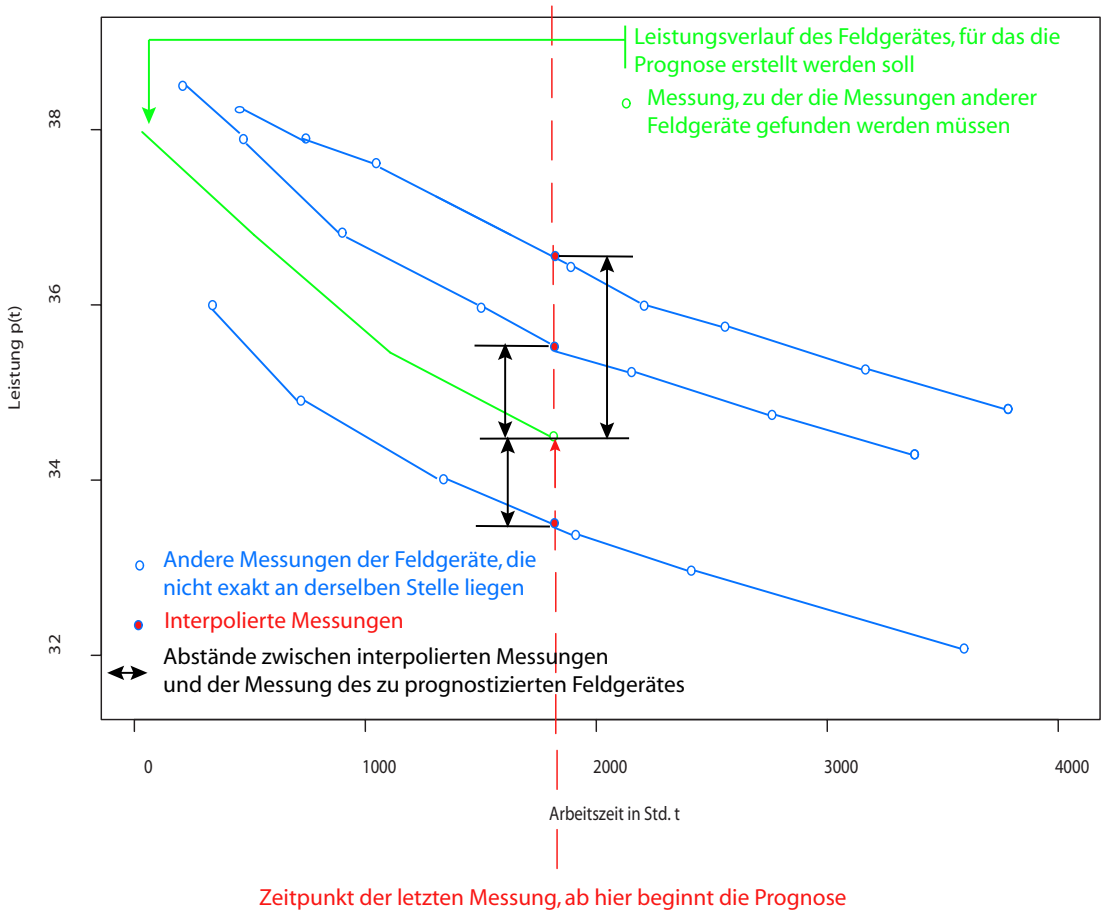


Abbildung 5.14: Interpolierte Messungen und deren Abstände zu der Messung des Prognose-Feldgerätes, die in $list_{knn}$ eingefügt werden.

- Schritt 1
  - werden die Parameter an den DB-Zugreifer übergeben. Als Parameter wird die ID des Feldgerätes übergeben, für das die Prognose erstellt werden soll (id_{fd}), sowie die Anzahl der von einem Feldgerät geleisteten Arbeitsstunden zum Anfang der Prognose (t_{begin}).
- Schritt 2
  - Der DB-Zugreifer findet anhand der übergebenen Parametern alle Messungen dieses Feldgerätes bis zum Zeitpunkt, ab dem prognostiziert werden soll. Dies geschieht durch die Datenbankabfrage. Anschließend packt er die gefundenen Messungen in eine Liste. Diese Liste wird dann als Ergebnis zurückgeliefert und in der Struktur $list_m^{fd}$ gespeichert.
- Schritt 3
  - Analog zu Schritt 1 wird der DB-Zugreifer mit den Parametern $id_{fd-type}$ (ID des Feldgerätetyps) und t_{end} (Anzahl der Arbeitsstunden am Ende der Prognose) aufgerufen.

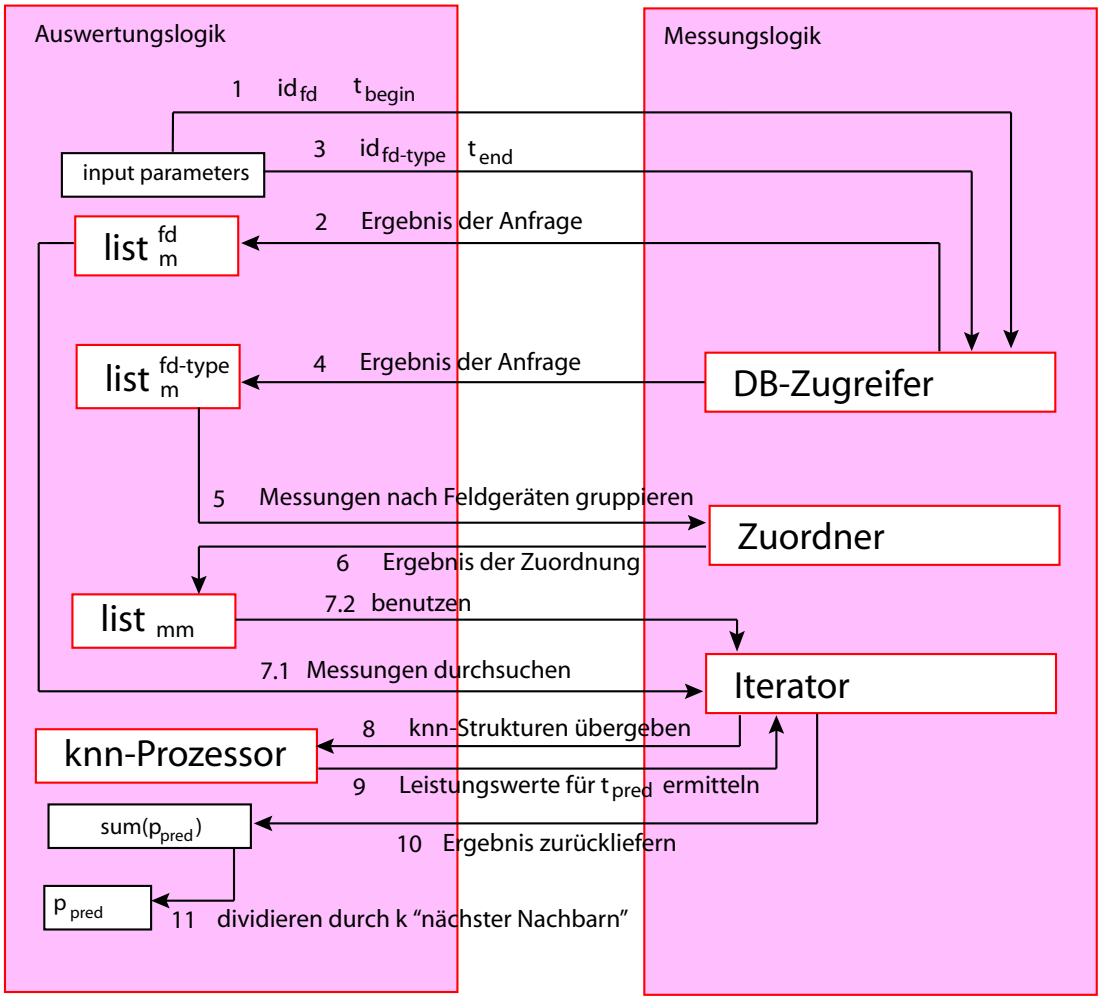


Abbildung 5.15: Vorhersage des Verhaltens eines Feldgerätes in der Zukunft: der Prognosealgorithmus.

- Schritt 4
  - Analog zu Schritt 2 wird von dem DB-Zugreifer eine Liste von allen Messungen (bis zur t_{end}) zurückgeliefert, die bei den Feldgeräten eines bestimmten Typs durchgeführt wurden. Diese Liste wird in der Struktur $list_m^{fd-type}$ gespeichert.
- Schritt 5
  - Jetzt müssen die Messungen, die sich in $list_m^{fd-type}$ befinden, in eine andere Form gebracht werden. Für die spätere Ermittlung der nächsten Nachbarn wäre es sehr hilfreich, eine Liste von Strukturen zu haben, in der jede Struktur alle Messungen eines Feldgerätes in sich kapselt. Um zu erkennen, zu welchem Feldgerät die Messungen gehören, ohne sich jede Messung extra anschauen zu müssen, wird jeder Struktur zusätzlich eine entsprechende Feldgeräte-ID als ein weiterer Attribut hinzugefügt. Um diese Aufgabe durchzuführen, wird $list_m^{fd-type}$ an den Zuordner (Algorithm 1) übergeben.

- Schritt 6

  - Der Zuordner iteriert durch alle Messungen. Vor der Iteration erzeugt er ein leeres mm. Bei der Betrachtung der ersten Messung wird deren id_{fd}-Attribut die Feldgeräte-ID der Messung zugewiesen. Ist die Messung nicht die erste, und es existieren schon die Messungen desselben Feldgerätes, die dem mm entsprechend zugewiesen wurden, wird auch sie diesem mm zugewiesen. Falls allerdings bei der Iteration durch Messungen festgestellt wird, dass die Feldgeräte-ID sich geändert hat, so wird speziell für diese Konstellation eine Neuerzeugung eines mm mit anschließender Zuweisung des id_{fd}-Attributes durchgeführt. Die Tatsache, dass die Messungen nach Feldgeräte-ID geordnet sind, schafft eine große Abhilfe, denn dadurch wird die Erzeugung mehrerer mm mit gleichen id_{fd}-Attributen unterdrückt. Die mm werden in eine Liste ($list_{mm}$) eingefügt, die anschließend zurückgeliefert wird.

- Schritt 7

  - Die Struktur $list_m^{fd}$ wird an den Iterator übergeben (Schritt 7.1). Der Iterator durchläuft alle Messungen. Jetzt ist der Parameter der Anzahl letzter Messungen wichtig, denn für den Algorythmus sind nur die letzten Messungen von Interesse: Nur für die Leistungswerte dieser Messungen müssen Messungen anderer Feldgeräte desselben Typs durchsucht und anschließend interpoliert werden. So extrahiert der Iterator aus jeder Messung dessen t- und p-Wert und übergibt sie (zusammen mit dem Parameter für den Typ der Interpolation) an eine seiner inneren Funktionen, die eine geordnete Liste von knn-Strukturen zurückliefert. Dabei wird die Struktur $list_{mm}$ (Schritt 7.2) benutzt.

  - Als Zwischenstand hat man zwei Datenstrukturen: Die $list_m^{fd}$ mit Messungen von dem zu prognostizierten Feldgerät und $list_{knn}$, die von allen Feldgeräten des gegebenen Feldgerätetyps eine Messung für einen bestimmten Zeitpunkt t der Prognose in Form von knn-Strukturen enthält.

  - Im Falle, dass $c_{lm} = 1$ (die nächsten Nachbarn werden anhand nur einer, letzter Messung bestimmt), ist die Arbeit im Schritt 7 praktisch beendet. Ist es nicht der Fall, so werden die inneren Funktionen, die die interpolierten Leistungswerte liefern, mehrmals aufgerufen, wobei jedes Mal die Abstände, die in neu zurückgelieferten knn enthalten sind, zu den entsprechnden Abständen in denselben knn in der $list_{knn}$ dazuaddiert werden.

- Schritt 8

  - Als Ergebnis von Schritt 7 wird also eine $list_{knn}$ zurückgeliefert. Darin befinden sich unter anderem die Differenzen zwischen einem Leistungswert an einer vorher als Parameter übergebenen t-Stelle desjenigen Feldgerätes, für das prognostiziert werden soll, und des Feldgerätes, das als möglicher nächster Nachbar betrachtet wird (Abbildung 5.14). Jetzt müssen die Ergebnisse nach dem d sortiert werden, damit später die ersten k Elemente auch die k nächste Nachbarn sind. Dazu wird eine Hilfsstruktur $list_{knn}^{tmp}$ mit der Größe k benutzt. Die ersten k Elemente werden aus $list_{knn}^{tmp}$ in $list_{knn}$ zurückgeschrieben. Dies ist erforderlich, um später, bei der Ermittlung der Leistung zum Prognosezeitpunkt t_{end}, nicht den ganzen $list_{knn}$ durchlaufen zu müssen.
  - Die Aufgaben von Schritten 7 und 8 übernimmt der knn-Prozessor. Der dazugehörige Ablauf ist in Algorithm 2 dargestellt.

- Schritt 9
  - $list_{knn}$, in dem sich die k nächsten Nachbarn befinden, wird an den Iterator übergeben. Der Iterator durchläuft alle nächsten Nachbarn und ermittelt den Leistungswert eines Feldgerätes p_{pred} an einer bestimmten t-Stelle (vorausgesetzt, es existieren Messungen vor und nach diesem t-Wert) anhand der an sie übergebenen Feldgeräte-ID (die praktischerweise in jeder *knn*-Struktur gespeichert ist) und t_{end}.
- Schritt 10
  - Der Iterator summiert alle Leistungswerte und weist die Summe der temporären Variablen $sum(p_{pred})$ zu.
- Schritt 11
  - Die Variable $sum(p_{pred})$ wird durch die Anzahl der nächsten Nachbarn dividiert und das Ergebnis wird in der Variable p_{pred} gespeichert. p_{pred} ist der Prognosewert der Leistung an einer Stelle t_{end}.

Algorithm 1: Der Zuordner

input : Liste aller Messungen eines gegebenen Feldgerätetyps $list_m^{fd-type}$
output: $list_{mm}$
begin
  create new $list_{mm}$
  $first \longleftarrow true$
  $id_{fd}^{tmp} \longleftarrow NONE$
  $mm \longleftarrow NONE$
  $id_{fd}^{tmp} \longleftarrow NONE$
  foreach m *of* $list_m^{fd-type}$ **do**
    if $id_{fd}^{tmp} \neq m.id_{fd}$ **then**
      if $first = false$ **then**
        Add mm to $list_{mm}$
      else
        $first \longleftarrow false$
        create new mm;
      $id_{fd}^{tmp} \longleftarrow m.id_{fd}$
      $mbm.id_{fd} \longleftarrow id_{fd}^{tmp}$
    add m to mm
  return $list_{mm}$
end

5.2.2 Integration der Interpolationsverfahren

Im vorherigen Abschnitt wurde das Prognosemodul vorgestellt, dessen Aufgabe es ist, Leistungswerte vorherzusagen. Dabei werden sehr oft die Leistungswerte von Messungen benötigt, die physikalisch nicht vorhanden sind und deshalb aus benachbarten Messungen ermittelt werden müssen. In Kapitel 3 wurden bereits die Verfahren für die Interpolation vorgestellt. Diese Verfahren müssen in das Prognosemodul integriert werden, um später – durch den Auswahl des Typs der Interpolation (wird durch den Benutzer gemacht, Abbildung 5.16) – eingesetzt zu werden.

Algorithm 2: Der knn-Prozessor

```
input  : Liste aller Messungen eines Feldgerätes list_m^fd
input  : Liste von mm-Strukturen list_mm
input  : Anzahl letzter Messungen c_lm
output: Liste von knn-Strukturen list_knn
begin
    count ⟵ 0
    list_knn^tmp ⟵ NONE
    foreach m of list_m^fd do
        if count < c_lm then
            count ⟵ (count + 1)
            list_knn ⟵ getInterpolatedPowerForOperationTime(
            m.id_fd, m.t, m.p, list_mm)
            if c_lm > 1 then
                if list_knn^tmp = NONE then
                    create new list_knn^tmp with the same size k as list_knn
                    list_knn^tmp ⟵ list_knn
                k = k − 1
                for i ← 0 to n do
                    foreach knn^tmp of list_knn^tmp do
                        foreach knn of list_knn do
                            if knn^tmp.id_fd = knn.id_fd then
                                d_old ⟵ knn.d
                                d_new ⟵ (d_old + knn.d)
                                knn^tmp.d ⟵ d_new
                list_knn ← list_knn^tmp
    return list_knn
end
```

Abbildung 5.16: Möglichkeit der Auswahl einer Interpolationsmethode (Typ der Berechnung, oberste Zeile) in der Benutzeroberfläche.

Interpoliert wird in einem Interpolator, der sich innerhalb des knn-Prozessors befindet. Er wird benutzt um Leistungswerte an bestimmten Stellen zu bekommen, an denen aber keine Messungen existieren. Der Algorithmus des Interpolators ist in Algorithm 3 vorgestellt.

5.3 Zusammenfassung

In diesem Kapitel wurde das Prognosemodul vorgestellt. Während der Analyse der Prognoseverfahren hat sich herausgestellt, mit welchen Einstellungen der Einsatz der SVM am effizientesten ist, und dass die SVM in dem Prognosealgorithmus nur als eine Alternative zu linearer Interpolation in dem k-Nächster-Nachbar-Algorithmus benutzt wird. Basierend darauf wurde der k-Nächster-Nachbar-Algorithmus in das PAM-System integriert und als Algorithmus für die Lebenszyklusprognose eines Feldgerätes eingesetzt.

Algorithm 3: Der Interpolator

input : UUID desjenigen Feldgerätes, dessen Leistung gefunden werden muss id_{fd}
input : Anzahl der geleisteten Arbeitsstunden desjenigen Feldgerätes, dessen Leistung gefunden werden muss t_{input}
input : Interpolationsmethode (linear oder svm) *type*
output: Interpolierte Leistung p_{int}
begin
  if $type = SVM$ **then**
    get learned svm function f
    $p_{int} = f(t)$
  if $type = LINEAR$ **then**
    $t_{min} \longleftarrow -\infty$
    $t_{max} \longleftarrow \infty$
    $p_{min} \longleftarrow 0$
    $p_{max} \longleftarrow 0$
    $list_m^{fd} \longleftarrow NONE$
    get $list_m^{fd}$ from the database by id_{fd}
    foreach m *of* $list_m^{fd}$ **do**
      if $m.t < t_{input}$ **then**
        if $m.t > t_{min}$ **then**
          $t_{min} \longleftarrow m.t$
          $p_{max} \longleftarrow m.p$
      if $m.t > t_{input}$ **then**
        if $m.t < t_{max}$ **then**
          $t_{max} \longleftarrow m.t$
          $p_{min} \longleftarrow m.p$
    $p_{int} \longleftarrow \frac{(p_{max}-p_{min})(t_{max}-input)}{(t_{max}-t_{min})} + p_{min}$
  return p_{int}
end

Kapitel 6

Evaluation

Nach der Integration des Prognosemoduls in das kundenbezogene PAM-System soll nun untersucht werden, ob der implementierte Prognoseagorithmus zuverlässige Ergebnisse liefert. Des Weiteren ist die Antwortzeit des PAM-Systems von Interesse – also die Zeit zwischen dem Abschicken der Prognoseanfrage und der Präsentation der Ergebnsse. In diesem Kapitel werden einige Versuche durchgeführt, mit deren Hilfe die oben erwähnten Fragen beantwortet werden können. Dazu wird zuerst die Versuchsumgebung eingeführt, indem die Datenbasis und die Hardwareumgebung beschrieben wird. Aufbauend darauf werden die Ergebnisse der Versuche analysiert und interpretiert.

6.1 Testumgebung

Die Testumgebung besteht aus der Datenbasis, die von dem Kunden zur Verfügung gestellt wurde, sowie aus der Hardwareausstattung des Versuchsystems.

6.1.1 Datenbasis

Für die Durchführung der Tests werden Daten verwendet, die der Kunde, für den das Prognosemodul entwickelt wurde, zur Verfügung stellte. Die Daten betrafen zwei Bereiche: kundenbezogene und feldgerätebezogene Daten. Aus Datenschutzgründen war es nicht möglich, eine Beziehung zwischen diesen Bereichen herzustellen, was allerdings auch im Rahmen dieser Arbeit nicht nötig war. Später sind die kundenbezogenen Daten gar nicht verwendet worden. Vielmehr wichtig waren die Soll- und Messdaten von Feldgeräten. Aufgrund der damals nicht vorhandenen Interesse an der Auswertung der Daten lagen diese in sehr unterschiedlichen Formaten vor und wiesen daher keine Struktur auf. Es wurden Messungsdaten genommen, die im MS-Word-Format vorhanden waren, und in die Datenbank überführt. Speziell dafür wurde ein Konverter geschrieben, dessen Aufgabe es war, MS-Word-Dokumente an bestimmten Stellen auszulesen und die daraus gewonnenen Daten in die Datenbank zu schreiben.

Die Datenbasis besteht aus 5 Feldgerätetypen. Es existieren ca. 600 - 1000 Feldgeräte pro Typ. Für jedes Feldgerät wurden ca. 10 - 20 Messungen durchgeführt. Die Messungen erstrecken sich im Zeitbereich von 0 Std. bis zum Garantieende. Das Garantieende unterscheidet sich je nach Feldgerätetyp (von 4000 bis 10000 Std.).

6.1.2 Hardwareausstattung

Alle Versuche sind auf einem Notebook Dell Inspiron 9100 (Intel Pentium 4, 3.0 GHz mit HT-Technologie, 1024 MB RAM) durchgeführt worden. Als Datenbank wurde MySQL (in der Version 4.0.18) sowie den MySQL ODBC-Treiber (in der Version 3.51.06) unter Windows XP

Home Edition. Die Ausführung der Versuche erfolgte auf Java 1.4 Enterprise Edition. Als Web-Server agierte Apache Tomcat (in der Version 5.5.9).

6.2 Testdurchführung

Um nachzuweisen, dass der Algorithmus zuverlässige Prognosen liefert, ist die Durchführung einiger Versuche notwendig, die die prognostizierten Werte für einen bestimmten Zeitpunkt ermitteln und nachher mit tatsächlichen Werten vergleichen. Dabei wird wie folgt vorgegangen:

- Es wird ein Feldgerätetyp ausgewählt.
- Es werden die Prognosekriterien ausgewählt, wie die Anzahl der von einem Feldgerät geleisteten Arbeitsstunden zum Anfang und zum Ende der Prognose (t_{begin} und t_{end}), Anzahl nächster Nachbarn k, Anzahl letzter Messungen c_{lm} und Interpolationstyp *type* (linear oder SVM). Der Parameter t_{end} sollte so ausgewählt werden, dass für alle Feldgeräte des gegebenen Typs zum Zeitpunkt t_{end} Messungen existieren. Dies erhöht die Genauigkeit des Tests.
- Für jedes Feldgerät des ausgewählten Feldgerätetyps wird eine Prognose durchgeführt. Dabei werden die Messungen, die zwischen den geleisteten Arbeitsstunden zum Anfang und zum Ende der Prognose erfolgten, erst mal nicht betrachtet.
- Als Ergebnis der Prognose kommt der vorhergesagte Leistungswert p_{pred} zum Zeitpunkt t_{end} heraus. Dieser Wert wird mit der tatsächlich gemessenen Leistung $p_{measured}$ verglichen. Falls an der Stelle t_{end} eine Messung existiert, so ist der Vergleich zweier Werte einfach. Ansonsten wird zwischen zwei benachbarten Messungen interpoliert (linear oder mit SVM, je nach der Einstellung) und anschließend verglichen. Sollte es nicht möglich sein, den tatsächlichen Leistungswert eines Feldgerätes an der Stelle t_{end} zu ermitteln (z.B. wenn $t_{end} = 3000$ Std., aber die Leistung für dieses Feldgerät nur bis 2700 Std. gemessen wurde, dann gibt es keine Möglichkeit einen reelen Leistungswert für 3000 Std. zu ermitteln), so wird die Prognose für das gegebene Feldgerät verworfen, und sie wird in die Gesamtbewertung nicht einfließen.
- Die aus dem Vergleich ermittelte Differenz zwischen den vorhergesagten und tatsächlichen Leistungswerten wird durch das Minimum dieser Werte dividiert. Daraus resultiert sich der relative Prognosefehler r_{fd} für jedes Feldgerät:

$$r_{fd} = |\frac{p_{pred} - p_{measured}}{\min(p_{pred}, p_{measured})}|.$$

- Anschließend wird der relative Prognosefehler für ein Feldgerätetyp $r_{fd-type}$ berechnet:

$$r_{fd-type} = \frac{\sum_{i=1}^{N} r_{fd}}{N}.$$

  Dabei bedeutet N die Anzahl der Feldgeräte, deren Prognosen in die Gesamtbewertung eingeflossen sind.

Beim Durchführen der Tests wurde versucht herauszufinden, wie die Veränderung des Wertes eines jeden Prognoseparameters sich auf den relativen Fehler auswirkt. Getestet wurde mit zwei Feldgerätetypen. Von dem ersten (*module type I*) sind ca. 600 Feldgeräte in der Datenbank enthalten, von dem zweiten (*module type II*) – ca. 3500. Die Leistung dieser Feldgeräte wurde bei ihrer Arbeitszeit von 0 bis 4000 Std. (bei *module type I*) bzw. von 0 bis 10000 Std. (*module type II*) regelmäßig gemessen. Nachfolgend werden die Tests einzeln beschrieben.

6.2.1 Test 1: Veränderung der Anzahl nächster Nachbarn

Das Ziel des ersten Tests war herauszufinden, welche Auswirkung die Anzahl der nächsten Nachbarn auf den relativen Fehler der Prognosen hat. Der Test verlief wie folgt ab:

- Für jede Anzahl der nächsten Nachbarn im Bereich von 1 bis 10 wurde die Prognose für die Feldgerätetypen *module type I* und *module type II* durchgeführt. Die Werte restlicher Prognoseparameter sind in der Tabelle 6.1 aufgelistet.

Parameter	Wert
Anzahl letzter Messungen	5
Interpolationsmethode	linear
Prognoseanfang	2000 Std.
Prognoseende	3200 Std. (bei *module type I*) 8600 Std. (bei *module type II*)

Tabelle 6.1: Parameter für die Ermittlung des relativen Prognosefehlers bei der variablen Anzahl nächster Nachbarn.

- Die daraus resultierenden Ergebnisse wurden in der Tabelle *knn* gespeichert.
- Mit Hilfe von R-Project wurden Grafiken erstellt, die die Abhängigkeit des relativen Prognosefehlers von der Anzahl der nächsten Nachbarn darstellen (Abbildung 6.1 und 6.2).

Aus der Grafik ist ersichtlich, dass die Prognose sowohl beim Feldgerätetyp *module type I* als auch beim Feldgerätetyp *module type II* bei der Anzahl nächster Nachbarn $k = 2$ den kleinsten relativen Fehler hat und somit am zuverlässigsten ist. Dieses Ergebnis lässt sich wie folgt erklären:

- Sei fd_{pred} ein Feldgerät, für das die Prognose erstellt werden soll.
- Sei $power_{measured}$ die tatsächliche Leistung des fd_{pred} an der Prognosestelle.
- Seien n_1 und n_2 die zwei nächsten Nachbarn von fd_{pred}.
- Seien d_1 und d_2 die Differenzen zwischen $p_{measured}$ und der Leistung der nächsten Nachbarn n_1 bzw. n_2.
- d_1 (und daraus resultierender relativer Prognosefehler) ist in den meisten Situationen größer als der Mittelwert zweier Abstände d_1 und d_2 (Abbildung 6.3).

  Somit erklärt sich der kleinere relative Prognosefehler $r_{fd-type}$ bei $k = 2$ gegenüber von $k = 1$. Wie man in den Abbildungen 6.1 und 6.2 sieht, ist $r_{fd-type}$ ab $k = 3$ fast konstant und dabei etwas größer als bei $k = 2$. Das nahezu konstante Verhalten von $r_{fd-type}$ lässt sich durch die Annhame erklären, dass im Falle $k = 2$ das erste Feldgerät, dessen Leistung oberhalb bzw. unterhalb der Leistung des Feldgerätes fd_{pred} liegt, für das die Prognose erstellt werden soll, als erster (n_1) bzw. zweiter (n_2) Nachbar genommen wird. Der Mittelwert ihrer Abstände d_1 und d_2 (Abbildung 6.3) ist in diesem Fall am exaktesten. Kommt (im Falle $k = 3$) noch ein Nachbar hinzu, so kann es sein, dass er den Prognosewert durch sein im Vergleich zu anderen Nachbarn etwas fremderes Verhalten negativ beeinflusst. Denn bei $k = 2$ hat man in fast allen Fällen zwei Abstände, wobei einer davon etwas pessimistischere und anderer etwas optimistischere Prognose fördert. Damit kann ein verhältnissmäßig extakter Wert prognostiziert werden. Der Abstand d_3,

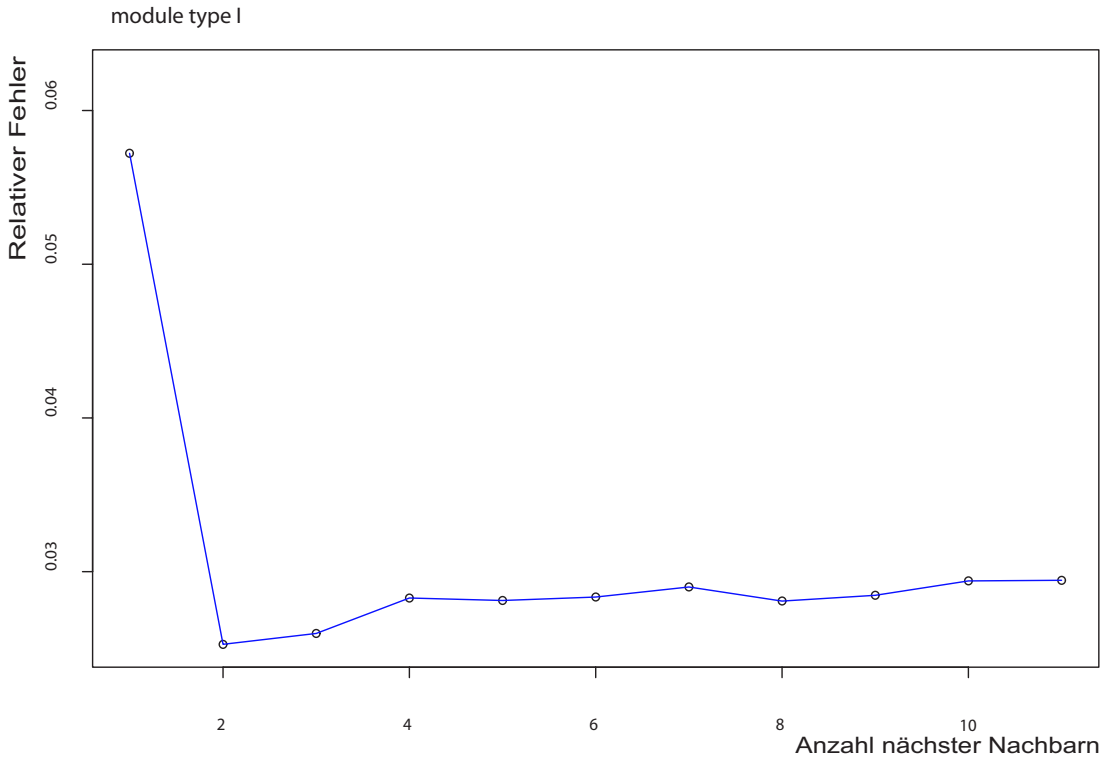


Abbildung 6.1: Abhängigkeit des relativen Prognosefehlers $r_{fd-type}$ von der Anzahl nächster Nachbarn k (Feldgerätetyp *module type I*). $r_{fd-type}$ ist bei $k = 1$ am größten, bei $k = 2$ – am kleinsten, ab $k = 3$ – nahezu konstant.

der von dem dritten Nachbarn kommt, zieht das Gesamtergebnis in eine bestimmte – pessimistische oder optimistiche – Richtung und vergrößert dadurch den $r_{fd-type}$. Bei der größeren Anzahl von k kommen neue Feldgeräte in Betracht, deren Leistung aber immer mehr und mehr von der Leistung des Feldgerätes fd_{pred} abweicht. Insgesamt gleichen sich diese Auswirkungen etwa aus, was die nahezu konstante Verhaltensweise des $r_{fd-type}$ erklären kann. Nur der Rechenaufwand nimmt zu.

Die obenbeschriebene Vermutung setzt voraus, dass die Feldgeräte sich im Laufe ihrer Lebenszyklen ähnlich verhalten: Ihre Leistungskurven dürfen sich fast nie überschneiden lassen, wodurch eine sehr exakte Prognose ermöglicht wird. Die hohe Genauigkeit (und somit auch die Zuverässigkeit) der Prognose lässt sich zumindest teilweise durch den aus diesem Versuch ermittelten sehr kleinen $r_{fd-type}$ (2.2 ... 2.9% bei *module type I* und 2.8 ... 3.3% bei *module type II*) bestätigen. Im folgenden Test wird untersucht, ob $r_{fd-type}$ sich durch den günstigen Auswahl der Anzahl letzter Messungen noch weiter verkleinern lässt. Interessant wäre außerdem zu sehen, ob das Verhalten der Feldgeräte durch den Einsatz von SVM-Interpolation präziser eingeschätzt werden kann.

6.2.2 Test 2: Veränderung der Anzahl letzter Messungen

Analog zum ersten Test wurden für Feldgerätetypen *module type I* und *module type II* Prognosen durchgeführt, wobei alle Prognoseparameter außer der Anzahl der letzten Messungen konstant blieben und folgendermaßen aussahen:

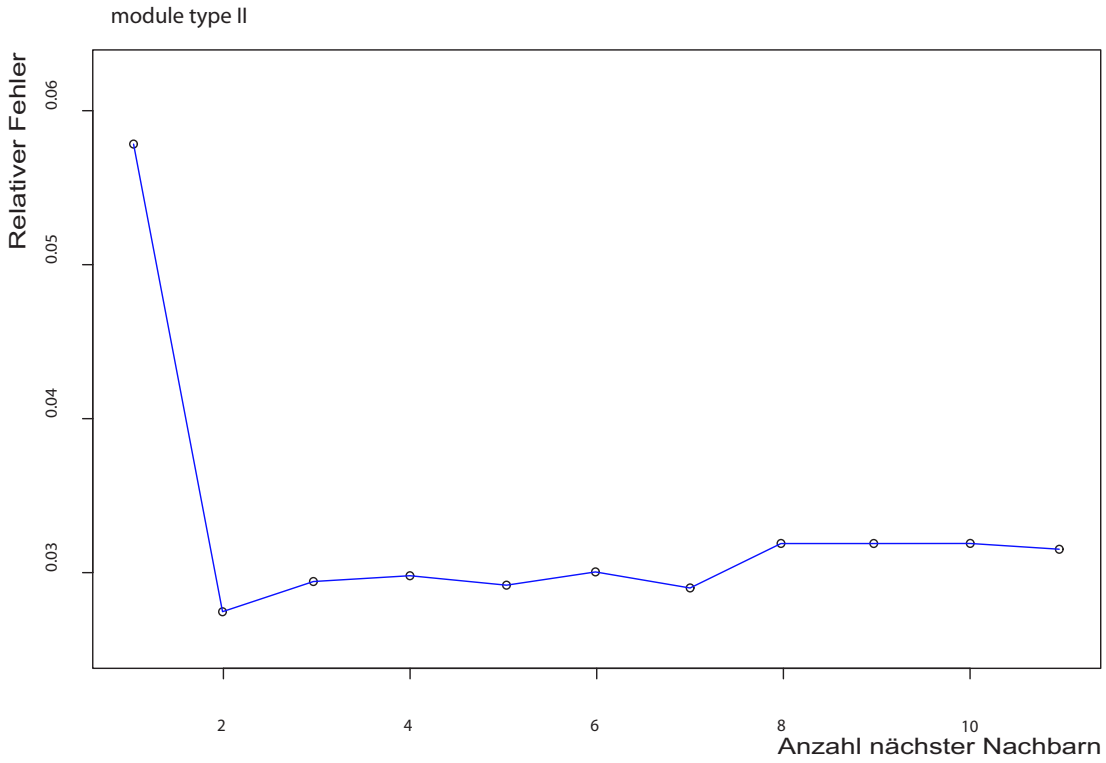


Abbildung 6.2: Abhängigkeit des relativen Prognosefehlers $r_{fd-type}$ von der Anzahl nächster Nachbarn k (Feldgerätetyp *module type II*). $r_{fd-type}$ ist etwas größer als bei *module type I*, verhält sich aber ähnlich.

Parameter	Wert
Anzahl nächster Nachbarn	2
Interpolationsmethode	linear
Prognoseanfang	2000 Std.
Prognoseende	3200 Std. (bei *module type I*) 8600 Std. (bei *module type II*)

Tabelle 6.2: Parameter für die Ermittlung des relativen Prognosefehlers bei der variablen Anzahl letzter Messungen.

Anschließend – wie im ersten Test – wurden Grafiken erstellt, die die Abhängigkeit des relativen Prognosefehlers von der Anzahl der letzten Messungen darstellen (Abbildungen 6.4 und 6.5).

Die Abhängigkeiten des relativen Prognosefehlers von der Anzahl letzter Messungen sehen bei beiden Feldgerätetypen ähnlich aus. Der relative Prognosefehler ist bei einer kleineren Anzahl letzter Messungen c_{lm} verhältnissmäßig gross. Bei $c_{lm} \in \{5, 6\}$ (je nach Feldgerätetyp) ist der relative Prognosefehler am kleinsten. Dann steigt der relative Prognosefehler wieder an, eine weitere Verkleinerung des relativen Prognosefehlers war nicht mehr zu erreichen.

Der Leistungsverlauf eines Feldgerätes lässt sich um so genauer beschreiben, je mehr Messpunkte betrachtet werden (je größer der Wert von c_{lm} ist). Somit kann genauer überprüft werden, ob der Leistungverlauf eines Feldgerätes fd in der Tat dem Leistungsverlauf des Feld-

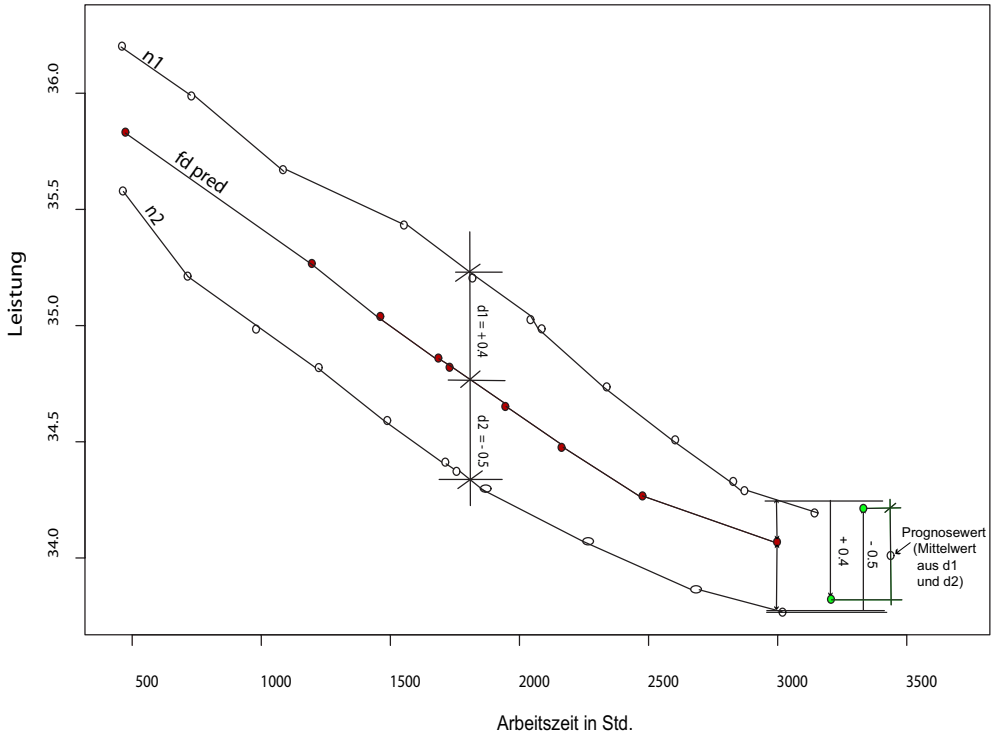


Abbildung 6.3: Der Prognosewert aus dem nächsten Nachbar n_1 (ausgerechnet aus dem Abstand d_1 und als unterer grüner Punkt dargestellt) weicht mehr von dem tatsächlichen Wert ab, als der Mittelwert der Abstände d_1 und d_2.

gerätes fd_{pred} ähnelt. Denn es kann auch sein, dass nur die letzten 1 – 2 Messungen von fd zufälligerweise den Messungen von fd_{pred} an gleichen Stellen ähneln, wobei die Leistungsverläufe insgesamt (vor und nach diesen Messungen) total verschieden sind. Der Algorithmus betrachtet aber nur diese Messungen und geht davon aus, dass das Feldgerät fd sehr gut zu dem Feldgerät fd_{pred} passt, und nimmt fd als einen der Nachbarn von fd_{pred} auf, was aber eigentlich nicht sein dürfte.

Daraus lässt sich Folgendes schließen: Je größer c_{lm} ist, desto kleiner soll $r_{fd-type}$ sein. Warum steigt dann der $r_{fd-type}$-Wert wieder (und zwar teilweise sehr drastisch) an?

Angenommen wurde zuerst, dass die Steigung des relativen Fehlers bei größeren Anzahl der letzten Messungen auf eine große Varianz zurückzuführen ist. Denn wäre die Varianz gross, könnte man sich durchaus vorstellen, dass der Verlauf in Abbildungen 6.4 und 6.5 etwas ebener ist und der relative Fehler bei großer Anzahl letzter Messungen unter Berücksichtigung der Varianz nicht so gross ist. Die Varianz ist aber sehr klein, was die Richtigkeit der Verläufe in Abbildungen 6.4 und 6.5 bestätigt.

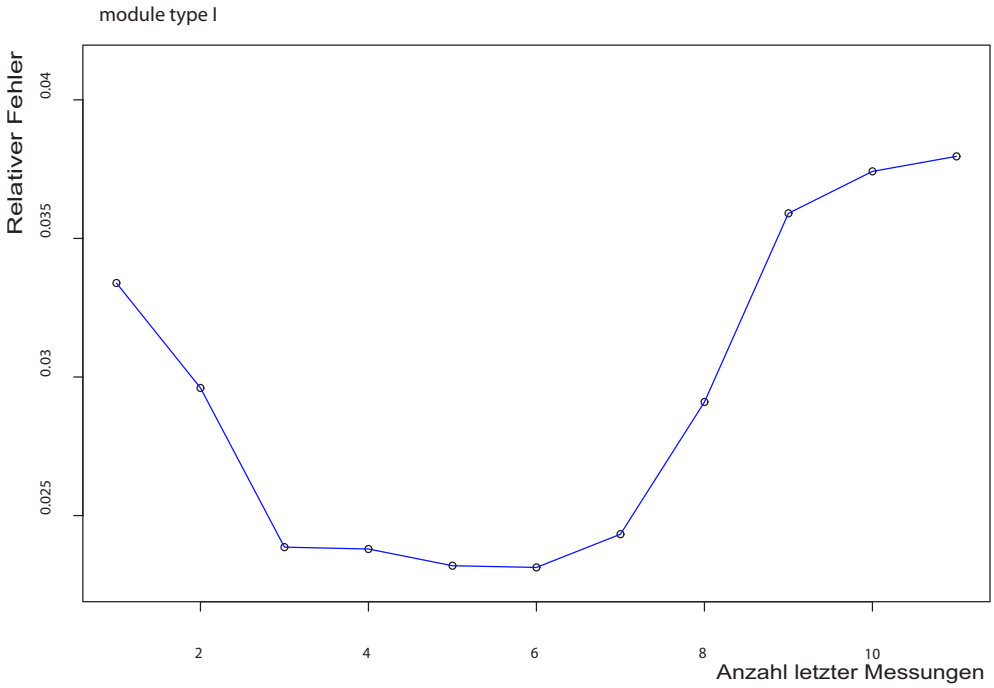


Abbildung 6.4: Abhängigkeit des relativen Prognosefehlers von der Anzahl letzter Messungen c_{lm} (Feldgerätetyp *module type I*). Der relative Prognosefehler erreicht bei $c_{lm} = 6$ seinen Tiefpunkt.

Der Grund eines solchen Verhaltens ist eine begrenzte Anzahl der Messungen. Nicht jedes Feldgerät wurde öfter als 7 - 9 Mal gemessen, nicht bei jedem Feldgerät liegen diese Messungen innerhalb des betrachteten Bereichs. Als Folge kommt, dass viele Feldgeräte nicht in die Prognose mitreingenommen wurden. Wenn aber viele von diesen Geräten einen dem Feldgerät fd_{pred} ähnlichen Leistungsverlauf aufweisen, hat das eine ziemlich negative Auswirkung auf den $r_{fd-type}$-Wert.

Ein aufmerksamer Leser würde jetzt sagen, dass es aber schon ziemlich viele Feldgeräte sein sollen, die in die Gesamtprognose nicht hereinfließen, dessen Leistungsveräufe aber am besten zu dem Leistungsverlauf von fd_{pred} passen, was eigeltlich sehr merkwürdig und eher unwahrscheinlich sei. Die von dem aufmerksamen Leser beschriebene Situation dürfte etwa wie in Abbildung 6.6 aussehen.

Die Situation in der Abbildung 6.6 ist zur Veranschaulichung gedacht und deshalb etwas übertrieben. Es ist aber teilweise tatsächlich so, dass es zu einer ähnlichen Kostellation kommen kann. Bei der Herstellung der Feldgeräte wird jedem Gerät eine Feldgerätenummer zugewiesen. Diese Nummer wird nicht zufällig generiert, sondern sie setzt sich aus verschiedenen Komponenten zusammen, eine von welchen eine fortlaufende Zahl ist. Es kann also die Situation eintreffen, dass bei einer Menge der Feldgeräte – z.B. einer Serie der Feldgeräte, die innerhalb einer Woche produziert worden waren und an einen Kunden verkauft wurden, – zu der fd_{pred} gehört, die Messungen nur bis 2000 Std. gemacht wurden (z.B. weil die Geräte relativ wenig im Einsatz waren). Wenn innerhalb der Woche, in der diese Menge der Feldgeräte produziert wurde, aufgrund eines Materialdefektes (oder Mitarbeiterversagen, z.B. Schutzmaske nicht auf) die Feldgeräte produziert wurden, deren Leistung im Laufe der Zeit schneller nachlassen wird als bei anderen, so ist eine Situation, die der Konstellation in der Abbildung 6.6 ähnlich ist, durchaus vorstellbar. Nach entsprechender Auswertung könnte man notfalls auf die Feldgerä-

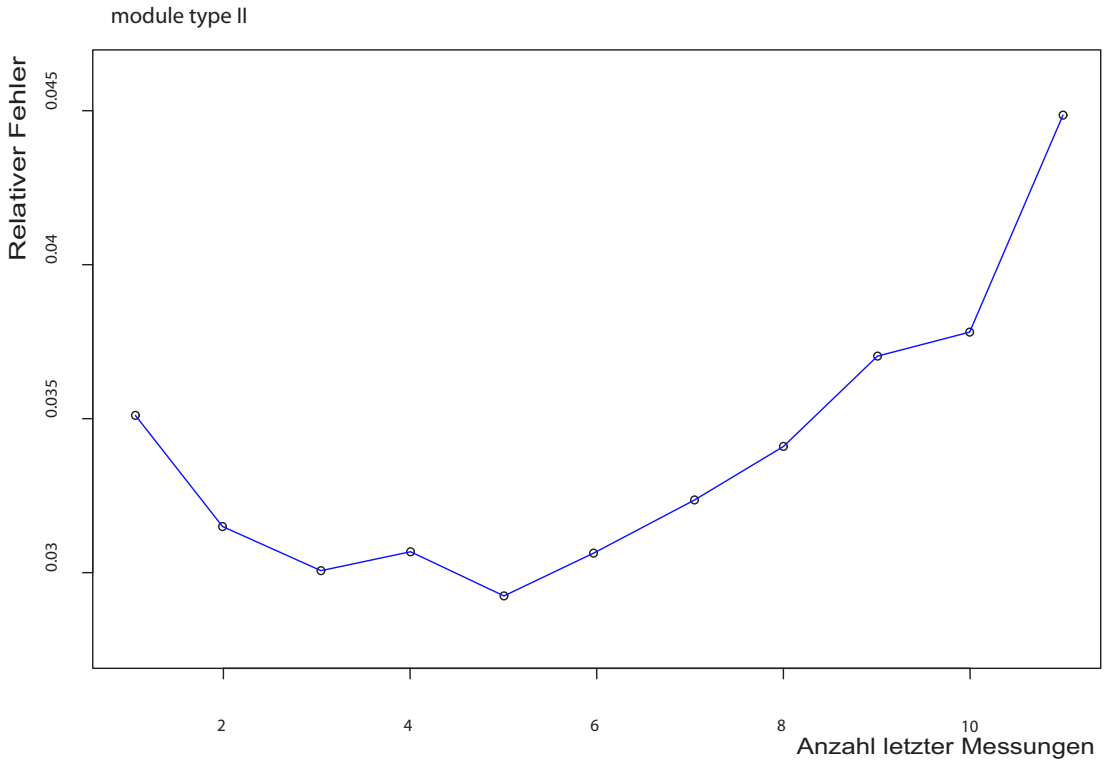


Abbildung 6.5: Abhängigkeit des relativen Prognosefehlers von der Anzahl letzter Messungen c_{lm} (Feldgerätetyp *module type II*). Der relative Prognosefehler erreicht bei $c_{lm} = 5$ seinen Tiefpunkt.

tenummern zurückgreifen, ermitteln, wann diese Geräte hergestellt wurden, und gezielter nach der Ursache der mangelnden Leistung suchen.

Um zu beweisen, dass der relative Fehler sich ausgerechnet aus dem Grunde der begrenzten Anzahl der Messungen ansteigt, wurden für jedes Feldgerät der Typen *module type I* und *module type II* zusätzliche Messdaten angefordert, so dass sichergestellt war, dass alle Feldgeräte nicht weniger als 9 Mal gemessen wurden, bevor sie 2000 Std. Einsatzgrenze erreicht haben. Somit konnte man genauestens die obengenannte Hypothese untersuchen.

Die Ergebisse der Untersuchung sieht man in den Abbildungen 6.7 und 6.8. Der relative Prognosefehler ist sowohl beim Feldgerätetyp *module type I* als auch beim Feldgerätetyp *module type II* konstant klein und fängt an zu steigen erst ab $c_{lm} > 9$. Da alle anderen Prognoseparameter unverändert geblieben sind, ist die Annahme der begrenzten Anzahl der Messungen somit bestätigt.

In der Situation, die in Abbildungen 6.4 und 6.5 (bzw. in Abbildungen 6.7 und 6.8) dargestellt ist, sind die Unterschiede zwischen den relativen Prognosefehlern nicht gravierend. Sie bewegen sich im Bereich von 2.3% bis 3.8% bei *module type I* und 2.8% ... 4.5% bei *module type II*, was als zuverlässig eingestuft werden darf. Es bleibt noch zu klären, in wie weit der relative Prognosefehler sich verändert, wenn man nicht linear, sondern mit Hilfe von SVM interpoliert. Dies wird in dem nächsten Test untersucht.

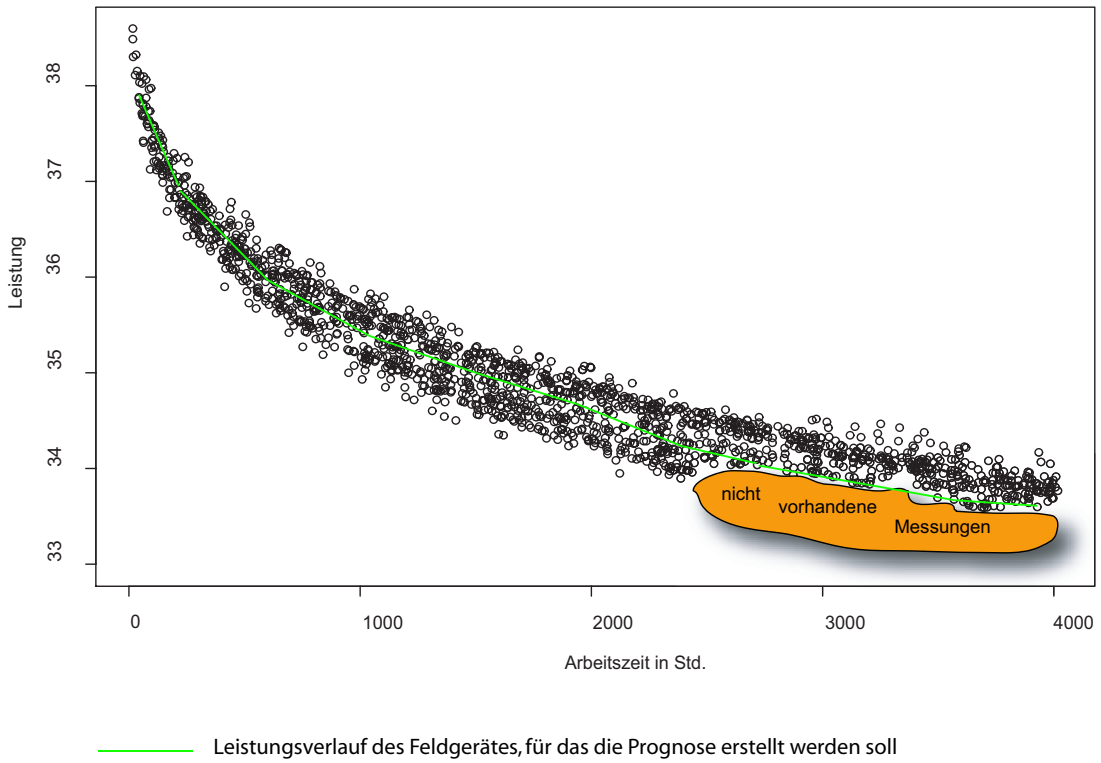


Abbildung 6.6: Unglückliche Situation für eine zu große Anzahl letzter Messungen: Sollte z.B. der Leistungsverlauf eines Feldgerätes fd_{pred} (grün dargestellt) im Bereich der Arbeitszeit von 2000 bis 4000 Std. prognostiziert werden, so werden fast alle Feldgeräte, deren Leistungsverläufe sich unterhalb des Leistungsverlauf von fd_{pred} befinden, nicht betrachtet, da sie nach 200 Std. kaum gemessen wurden. Als Ergebnis kommt eine optimistischere und ungenauere Prognose.

6.2.3 Test 3: Vergleich der Interpolationsarten

Wie bereits in Kapitel 3 geschrieben, ist es gestrebt herauszufinden, wie effiziert die Verfahren der linearen und SVM-Interpolation sind. Beide Verfahren werden eingesetzt, um unbekannte Werte zwischen den Messungen zu ermitteln. Das Ziel dieses Tests war es, beide Verfahren zu vergleichen und auszuwerten. Zwei Kriterien spielen dabei eine wichtige Rolle: Bearbeitungszeit einer Prognose und Zuverlässigkeit dieser Prognose.

Als Variationskriterium kann wieder entweder die Anzahl nächster Nachbarn und letzter Messungen ausgewählt werden. Vollständigkeithalber wird der Test mit beiden Variationskriterien durchgeführt.

Vergleich der Interpolationsarten bei der Veränderung der Anzahl nächster Nachbarn

Einstellungen:

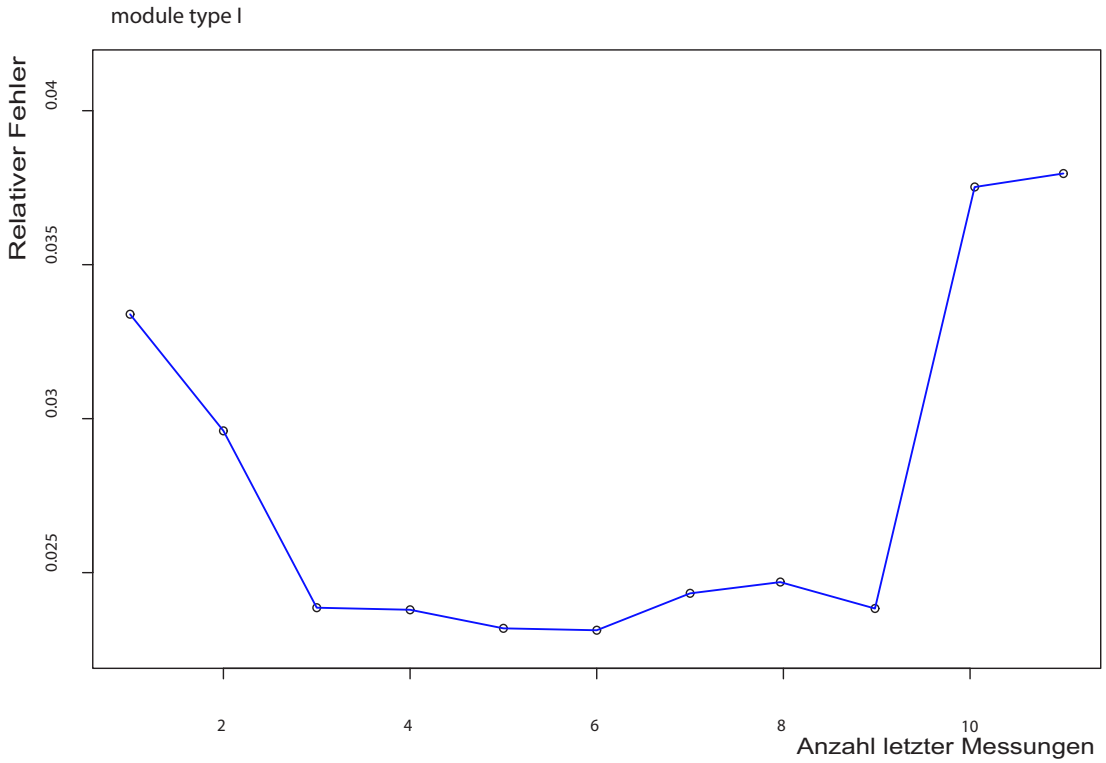


Abbildung 6.7: Abhängigkeit des relativen Prognosefehlers von der Anzahl letzter Messungen c_{lm} (Feldgerätetyp *module type I*). Nachdem die zusätzlichen Messungen angefordert wurden, ist der relative Prognosefehler bis $c_{lm} = 9$ ungefähr gleich klein.

Parameter	Wert
Anzahl letzter Messungen	5 (bei *module type I*) 6 (bei *module type II*)
Prognoseanfang	2000 Std.
Prognoseende	3200 Std. (bei *module type I*) 8600 Std. (bei *module type II*)

Tabelle 6.3: Parameter für den Vergleich linearer und SVM-Interpolationsmethode bei variirender Anzahl nächster Nachbarn.

In den Abbildungen 6.9 und 6.10 werden die Ergebnisse des Tests präsentiert.

Lineares und SVM-Interpolationsverfahren verhalten sich bei der variirenden Anzahl der nächsten Nachbarn ungefähr gleich. Bei *module type I* ist die lineare Interpolation sogar etwas zuverlässiger als die SVM-Interpolation. Dabei ist die lineare Interpolation bedeutend schneller als SVM-Interpolation. Die Laufzeitabschätzung furr die lineare Interpolation ist $O(n)$, wo n die Anzahl der Messungen eines Feldgerätes ist. Die Laufzeitabschätzung für die SVM-Interpolation ist aber in *worst case* $O(n^3)$. Ein weiterer, weniger bedeutender Faktor ist die Tatsache, dass der Aufruf von SVM mit Hilfe von R-Project erfolgt, und das Aufrufen eines externen Programms kostet Rechenzeit – dabei wäre die lineare Interpolation schon längst fertig. Die SVM wäre aber auch sonst viel langsamer (denn die Behauptung, dass die SVM nur deshalb langsamer ist, weil sie als externes Programm läuft, ist kein prinzipieller Grund).

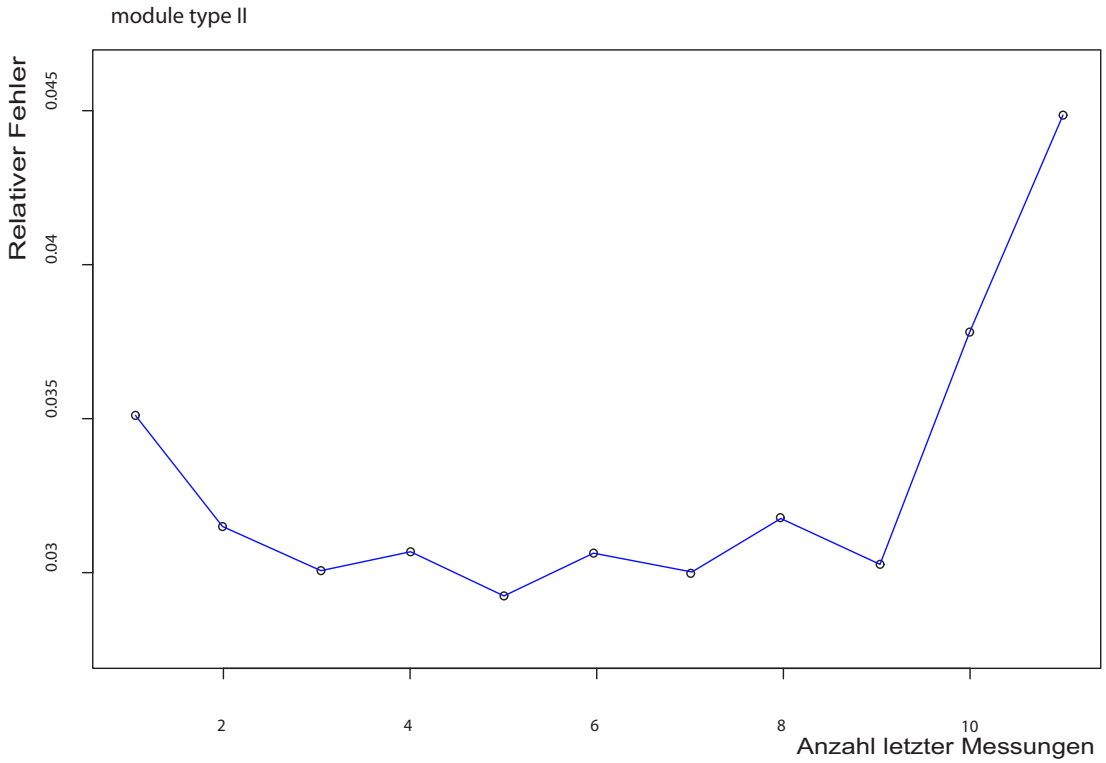


Abbildung 6.8: Abhängigkeit des relativen Prognosefehlers von der Anzahl letzter Messungen c_{lm} (Feldgerätetyp *module type II*). Nachdem die zusätzlichen Messungen angefordert wurden, ist der relative Prognosefehler bis $c_{lm} = 9$ – genauso wie bei dem Feldgerätetyp *module type I* – ungefähr gleich klein.

SVM-Interpolation verfeinert den Verlauf der Approximationsfunktion zwischen zwei Messungen. Dies ist zwar allgemein vorteilhafter als lineare Approximation und liefert in der Regel auch zuverlässigere Ergebnisse. Im Fall von diesen zwei Feldgerätetypen wird zwischen Messungen interpoliert, die vergleichweise ziemlich nah zueinander liegen. Diese Tatsache verstärkt die Position des linearen Interpolationsverfahren – je kleiner der Abstand zwischen den Messungen ist, desto näher ist der lineare Verlauf dem tatsächlichen, desto kleiner ist der relative Fehler, desto weniger Vorteile bringt die SVM-Interpolation gegenüber der linearen Interpolation.

Man könnte vermuten, dass die SVM-Interpolation bei einer kleineren Anzahl letzter Messungen ihre Stärke zeigt, da die Abstände zwischen Messungen im Bereich von 1500 bis 2000 Std. etwas größer sind. Diese Annahme wird in dem nächsten Test überprüft.

Vergleich der Interpolationsarten bei der Veränderung der Anzahl letzter Messungen

Einstellungen:

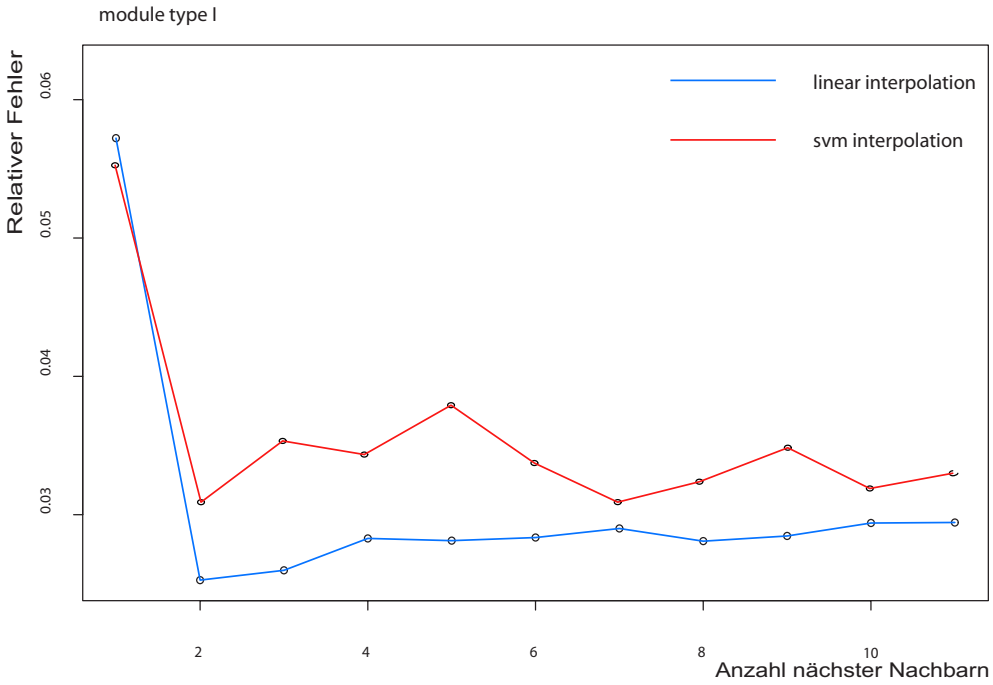


Abbildung 6.9: Vergleich linearer und SVM-Interpolationsmethode bei variirender Anzahl nächster Nachbarn (*module type I*).

Parameter	Wert
Anzahl nächster Nachbarn	2
Prognoseanfang	2000 Std.
Prognoseende	3200 Std. (bei *module type I*) 8600 Std. (bei *module type II*)

Tabelle 6.4: Parameter für den Vergleich linearer und SVM-Interpolationsmethode bei variabler Anzahl letzter Messungen.

Aus den daraus resultierenden Grafiken (Abbildungen 6.11 und 6.12) kann man sehen, dass die SVM-Interpolation keine besondere Verbesserungen gegenüber der linearen Interpolation bringt. Der bereits vorher erwähnte Nachteil – Rechenzeit – verfolgt die SVM auch in diesem Fall, sodass man die SVM-Interpolation nur in folgenden Fällen einsetzen könnte:

- Feldgerätetyp: *module type II*. Bei *module type I* lohnt sich der Einsatz von SVM nicht (Abbildung 6.9)
- Die Zuverlässigkeit der Prognose spielt die wichtigste Rolle
- Die Rechenzeit hat die niedrigste Priorität
- Die Anzahl nächster Nachbarn ist 2
- Die Anzahl letzter Messungen ist 5

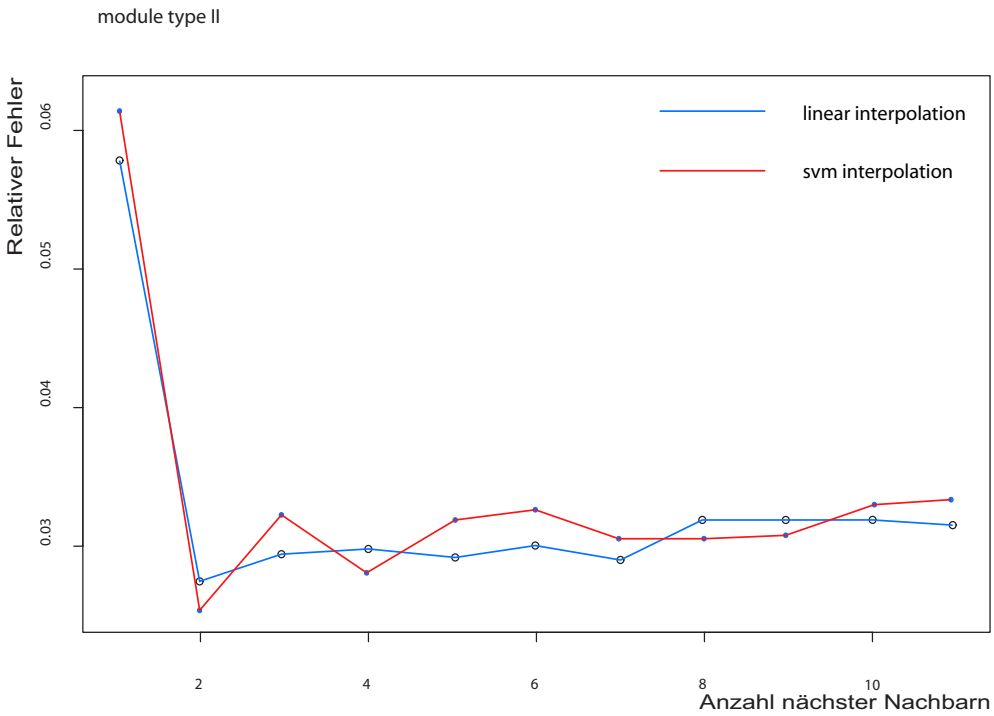


Abbildung 6.10: Vergleich linearer und SVM-Interpolationsmethode bei variirender Anzahl nächster Nachbarn (*module type II*).

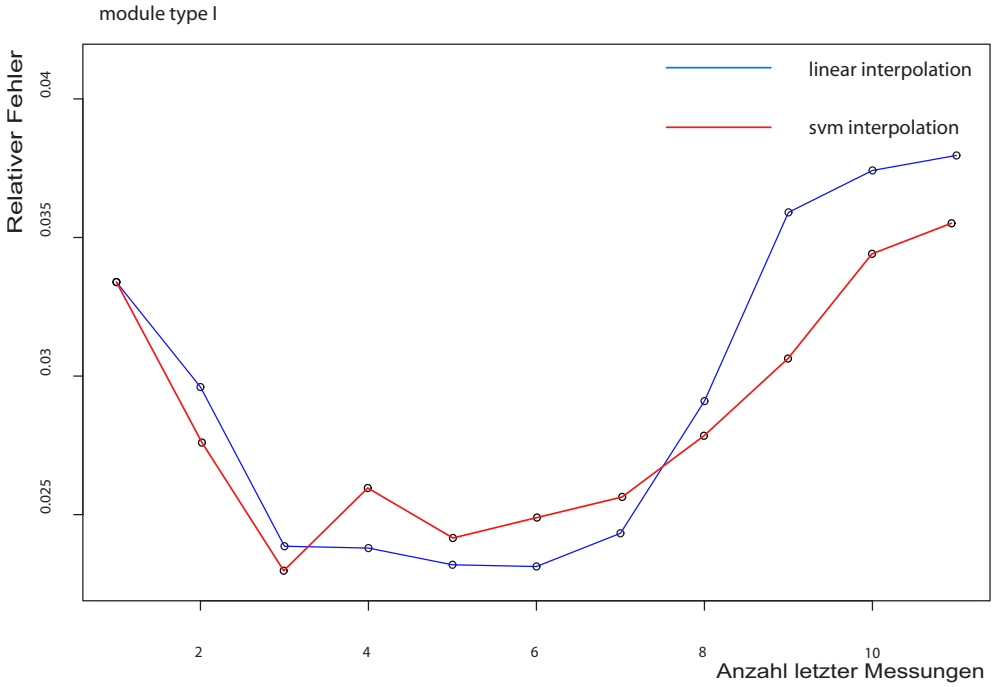


Abbildung 6.11: Vergleich linearer und SVM-Interpolationsmethode bei variabler Anzahl letzter Messungen (*module type I*).

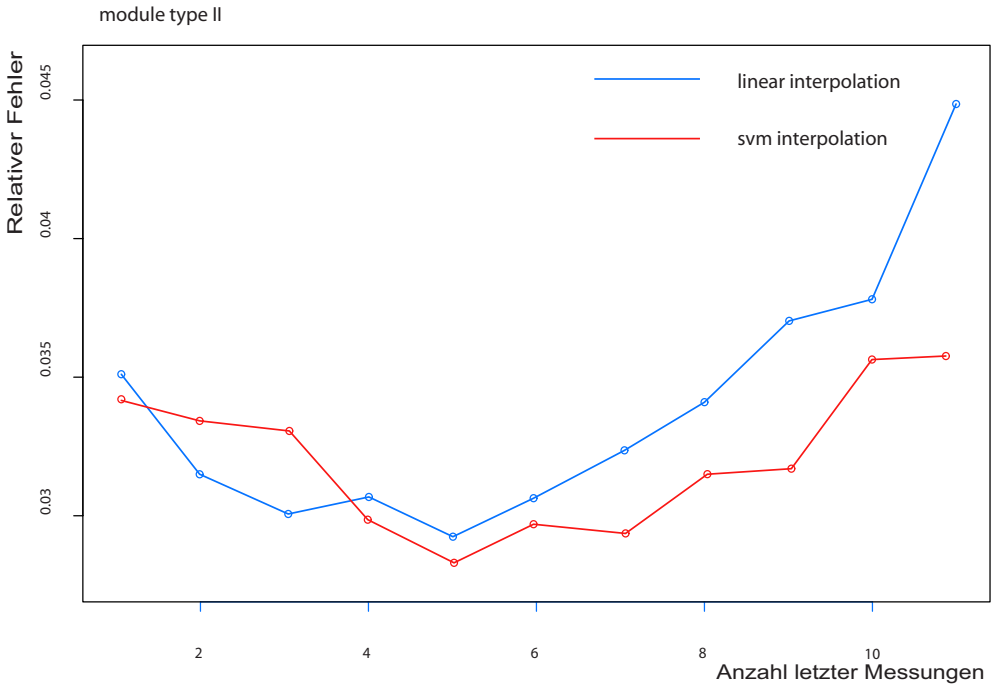


Abbildung 6.12: Vergleich linearer und SVM-Interpolationsmethode bei variabler Anzahl letzter Messungen (*module type II*).

Kapitel 7

Zusammenfassung

Zum Abschluss werden die Ergebnisse zusammengefasst und diskutiert, in wie weit die Ergebnisse dieser Arbeit die vom Kunden gestellte Ziele abdecken.

7.1 Erfüllung der Ziele

In der Einleitung wurden fünf wesentliche Ziele dieser Diplomarbeit formuliert:

- Verschiedene Prognoseverfahren für die Anwendungslogik des in das kundenbezogene PAM-System zu integrierenden Prognosemoduls einsetzen
- Ergebnisse vergleichen und analysieren
- Einen Algorithmus für die Prognose erstellen und implementieren
- Verschiedene Interpolationsmöglichkeiten beim Einsatz der Prognoseverfahren ausprobieren
- Empirisch nachweisen, dass der Algorithmus zuverlässige Prognosen liefert.

Ob diese Ziele erfüllt wurden, wird jetzt genauer betrachtet.

- Verschiedene Prognoseverfahren für die Anwendungslogik des in das kundenbezogene PAM-System zu integrierenden Prognosemoduls einsetzen
  - Für das kundenbezogene PAM-System kamen zwei Prognoseverfahren in Frage: die SVM und der k-Nächster-Nachbar-Algorithmus (beschrieben in Kapitel 3). Die SVM zeichnet sich dadurch aus, dass sie globale Lösungen findet, was auf dem Gebiet des maschinellen Lernens nicht selbstverständlich ist. Ihre Anwendungsmöglichkeiten in der Praxis sind sehr vielfältig. Dazu zählen beispielsweise Bild- und Schrifterkennung und Umsatzprognosen für Unternehmen. Interessant war zu sehen, ob die SVM auch für die Prognose der Feldgeräte gute Leistung erbringen kann.
  - Mit dem k-Nächster-Nachbar-Algorithmus kann das Verhalten eines Feldgerätes durch das Verhalten benachbarter Feldgeräte vorhergesagt werden. Dabei ist er sehr einfach zu implemetieren und erfordert keine Schnittstellen zu externen Programmen (wie bei der SVM), was positive Auswirkungen im bezug auf Rechenzeit hat.

- Ergebnisse vergleichen und analysieren
  - Die Analyse der Prognoseverfahren (Kapitel 5) zeigte, dass die SVM sich als reines Prognoseverfahren nicht ganz eignet, da sie die Feldgeräte, deren Verhalten sich etwas bedeutender von dem Verhalten anderer, überwiegenderer Anzahl der Feldgeräte unterscheidet (sogenannte Ausreißer) falsch einschätzt. Deshalb wurde entschieden, den k-Nächster-Nachbar-Algorithmus für das Prognosemodul zu benutzen.
- Einen Algorithmus für die Prognose erstellen und implementieren
  - Der Prognosealgorithmus basiert auf dem k-Nächster-Nachbar-Algorithmus. Der Algorithmus findet die k Feldgeräte, deren Verhalten sich am meisten dem Verhalten des Feldgerätes ähnelt, für das die Prognose erstellt werden soll (Nachbarn). Er orientiert sich dabei an solche Parameter wie die Anzahl letzter Nachbarn und die Anzahl letzter Messungen sowie die Arbeitszeit der Feldgeräte zum Anfang und zum Ende der Prognose. Dann wird geschaut, welchen Leistungswert jeder Nachbar am Zeitpunkt der Prognoseende hat, die Werte werden aufsummiert und durch die Anzahl der Nachbarn dividiert. Als Ergebnis kommt der prognostizierte Leistungswert.
- Verschiedene Interpolationsmöglichkeiten beim Einsatz der Prognoseverfahren ausprobieren
  - Die unbekannten Werte in dem Prognosealgorithmus müssen mit Hilfe der Interpolation ermittelt werden. Dabei kam – neben der linearen Interpolation – wieder die Möglichkeit in Frage, die SVM einzusetzen. Die SVM wird mit den Messungswerten der von dem k-Nächster-Nachbar-Algorithmus als Nachbarn identifizierten Feldgeräte gefüttert. Daraus wird eine Funktion erlernt, die man benutzt, um einen Leistungswert an einer Stelle zu ermitteln, an der keine Messung stattfand.
- Empirisch nachweisen, dass der Algorithmus zuverlässige Prognosen liefert.
  - Die Zuverlässigkeit der Prognose wurde an dem relativen Fehler gemessen. Aus den Versuchen (Kapitel 6) wurden die Parameter ermittelt, bei denen der relative Fehler sehr klein gehalten werden kann (2.5% bis 3%).

7.2 Fazit

Mit dem Einsatz von SVM ist man nicht in allen Fällen am besten bedient. Auch andere Prognoseverfahren, wie z.B. der k-Nächster-Nachbar-Algorithmus, liefern nicht unbedingt schlechtere – und in dieser Arbeit sogar bessere – Ergebnisse. Bei der Interpolation liefert die SVM zwar in bestimmen Fällen genauere Werte als die lineare Interpolation, ist aber aufgrund ihrer Komplexität und aufwändigerer Einsatzart eher selten zu empfehlen.

Literaturverzeichnis

[1] Abel, Olaf und Boll, Marko: *Beispiele für Asset Management in Betrieben der chemischen Industrie*. Wiesbaden, Juni 2002.

[2] Bang, Christian: *Inkrementelle Support-Vektor Maschinen im GAILS Framework: Integration und Evaluierung*. Diplomarbeit, Technische Universität Darmstadt, Darmstadt, September 2004.

[3] Bodoff, Stephanie: *Java Servlet Technology*. http://java.sun.com/webservices/docs/1.0/tutorial/doc/Servlets.html.

[4] Clayton, David: *Plant Asset Management is Critical to Real-time Performance Management*. ARC Insights, 2003.

[5] Coad, Peter und Yourdon, Edward: *Objektorientierte Analyse OOA*. VMI Buch AG, Bonn, 1992.

[6] Coward, Danny: *Java Servlet Specification*. http://jcp.org/aboutJava/communityprocess/first/jsr053/index.html, 2005.

[7] Distributed Technologies GmbH d-tec. http://www.d-tec.ch, 1997.

[8] Draper, Norman R. und Smith, Harry: *Applied Regression Analysis*. Wiley-VCH Verlag, 3 Auflage, Mai 1998.

[9] Ferber, Reginald: *Information Retrieval*. Dpunkt Verlag, Heidelberg, Oktober 2003.

[10] Hall, Marty: *Core Servlets and JavaServer Pages (JSP)*. Prentice Hall PTR, Mai 2000.

[11] Hallama, Peter: *Maschine Learning Nonparametric Techniques*, November 2003.

[12] Han, Jiawei und Kamber, Micheline: *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[13] Harms, Volker: *Kundendienst – Serviceleistungen für Kunden und Produkte*. Carl Hanser Verlag, 2004.

[14] Hübner, Kai: *Neuere Verfahren der Neuroinformatik*, Januar 2001.

[15] Hunter, Jason. http://www.servlets.com, 2005.

[16] IBM: *Servlets*. http://www-4.ibm.com/software/webservers/appserv/doc/v30/ae/web/doc/whatis/icsrvlet.html, 2005.

[17] Joachims, Thorsten: *Advances in Kernel Methods - Support Vector Learning*. In: Schölkopf, B. und Burges, C. und Smola A. (Herausgeber): *Making large-Scale SVM Learning Practical*, Nummer 11. MIT Press, 1999.

[18] Joachims, Thorsten: *Learning to Classify Text using Support Vector Machines.* Kluwer Academic Publishers, Mai 2002.

[19] Kunze, Katja: *Support Vector Machines, Kernels*, Januar 2004.

[20] Lutz, Monika: *Aufgabenstellung bei Interpolation und Approximation.* http://www.fh-friedberg.de/users/mlutz/JavaKurs/applets/Splines/AufgabeInterpolation.htm.

[21] Morik, K. und Wrobel, S. und Joachims T.: *Maschinelles Lernen.* In: Görz, G. (Herausgeber): *Handbuch der Künstlichen Intelligenz.* Addison Wesley, 2000.

[22] NAMUR-Empfehlung: *Anforderungen an Systeme fur Anlagennahes Asset Management.* NE91, 2001.

[23] Ponader, Michael: *E-Business im Bereich Service.* 9. 2002.

[24] Rüping, Stefan: *Zeitreihenprognose für Warenwirtschaftssysteme unter Berücksichtigung asymmetrischer Kostenfunktionen.* Diplomarbeit, Universität Dortmund, Dortmund, September 1999.

[25] Saake, Günter und Sattler, Kai-Uwe: *Datenbanken & Java – JDBC, SQLJ, ODMG und JDO.* Dpunkt Verlag, Heidelberg, 2 Auflage, 2003.

[26] Scheffer, Tobias und Bickel, Steffen: *Instanzbasiertes Lernen (und Regression)*, 2004.

[27] Schölkopf, B. und Smola, A. und Müller K.: *Nonlinear component analysis as a kernel eigenvalue problem.* Nummer 10, Seiten 1299–1319. Neural Computation, 1998.

[28] Schönfeld, Peter: *Methoden der Ökonometrie*, Band 1. Vahlen Verlag, Berlin, 1969.

[29] Tikhonov, A.N. und Arsenin, V.Y.: *Solution of Ill-Posed Problems.* Winston, Washington, DC, 1977.

[30] Vapnik, Vladimir: *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

Abbildungsverzeichnis

Tabellenverzeichnis