Das Programmsystem SOLVES zur Suche und Optimierung von Leitstrukturen

Dissertation

der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Dipl.-Inform. Fred-Reiner Rapp

aus Reutlingen

Tübingen
2005

Tag der mündlichen Qualifikation:	08.06.2005
Dekan:	Prof. Dr. Michael Diehl
1. Berichterstatter:	Prof. Dr. Andreas Zell
2. Berichterstatter:	Prof. Dr. Günter Gauglitz

Kurzfassung

Das Aufkommen des rationalen Wirkstoffdesigns in der Pharmaforschung gegen Mitte des letzten Jahrhunderts markiert einen Paradigmenwechsel, der erstmals die Vorhersage der Verwendbarkeit einer chemischen Substanz als pharmazeutischen Wirkstoff ermöglichte. Von relativ einfachen Berechnungsverfahren hat sich dieses Gebiet mit der steigenden Leistungskraft der eingesetzten Computer hin zu immer komplexeren Verfahren entwickelt. Methoden des sogenannten *Softcomputings*, zu denen u. a. die Neuronalen Netze und die Evolutionären Algorithmen zählen, haben sich inzwischen fest etabliert. Neu entwickelte Methoden der Datenverarbeitung werden aufgrund des in der Pharmaforschung herrschenden hohen Zeit- und Kostendrucks schnell aufgegriffen und im praktischen Einsatz erprobt.

Problematisch hierbei ist, dass sich das Spektrum der im Einsatz befindlichen Methoden stetig erweitert. Einmal etablierte Methoden werden nur selten von neueren vollständig abgelöst, auch wenn letztere sich in vielen Fällen als überlegen erweisen. Nach dem Prinzip von *Occam's Razor* ist von mehreren vergleichbaren Modellen stets dasjenige mit der geringsten Anzahl an freien Parametern vorzuziehen, so dass je nach der inneren Komplexität eines vorliegenden Datensatzes auch relativ einfache Methoden das beste Ergebnis liefern können. Liegen diese unterschiedlichen Methoden zudem noch in Form von separaten Programmen mit zueinander nicht kompatiblen Datenformaten vor, so wird eine vergleichende, explorative Datenanalyse zusätzlich erschwert.

Das Hauptziel dieser Arbeit liegt daher in der Konzeption und Implementierung eines Hilfsmittels, welches die genannten Methoden mit weiteren anwendungsspezifischen Verfahren kombiniert und im Rahmen einer einheitlichen Benutzungsoberfläche dem in der Pharmaforschung tätigen Wissenschaftler zur Verfügung stellt.

Es handelt sich dabei um ein verteiltes Programmsystem namens SOLVES, welches eine einfache, flexible und intuitive Zusammenstellung von Berechnungsabläufen (hier als „Prozessketten“ bezeichnet) ermöglicht und die erzielten Ergebnisse dauerhaft und nachvollziehbar archiviert. Neben den in der Pharmaforschung typischen Anwendungen des Wirkstoffdesigns und der Leitstruktursuche soll dieses auch in anderen Anwendungsbereichen einsetzbar sein.

Nach einem einleitenden ersten Kapitel beschäftigt sich das zweite mit den Grundlagen der Datenauswertung und des rationalen Wirkstoffdesigns, sowie mit relevanten technischen Standards. Im dritten Kapitel wird das Programmsystem SOLVES vorgestellt und mit vergleichbaren Ansätzen verglichen. Das vierte Kapitel beschäftigt sich mit der Auswertung von beispielhaft ausgesuchten Datensätzen und der Deskriptorselektion. Im fünften Kapitel wird das STEEDS-Verfahren zur Optimierung von Substruktur-Deskriptoren vorgestellt. Abschließend werden die in der Arbeit gewonnenen Erkenntnisse zusammengefasst und ein Ausblick auf weitere Entwicklungsmöglichkeiten gegeben.

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Rechnerarchitektur an der Universität Tübingen. Ich danke Prof. Dr. Andreas Zell für die freundliche Unterstützung und für die Möglichkeit, an seinem Lehrstuhl in einem sehr guten und motivierenden Arbeitsklima promovieren zu dürfen.

Prof. Dr. Johann Gasteiger danke ich für die stets konstruktiven Anregungen und für die Leitung des SOL-Projekts zur Suche und Optimierung von Leitstrukturen. Dem Bundesministerium für Bildung und Forschung (BmBF) sei herzlich für die Förderung des Projekts unter der Kennziffer 0311681 gedankt.

Ich danke Dr. Andreas Dominik für die gute und umfangreiche Zusammenarbeit von Seiten der Altana Pharma AG (vormals Byk Gulden GmbH) im Rahmen des SOL-Projekts. Dies gilt im gleichen Maße für Dr. Claude Ostermann und Martin Kraus. Auch den Vertretern der anderen teilnehmenden Firmen, besonders Frau Dr. Soheila Anzali, Dr. Michael Krug und Dr. Jochen Antel sei für ihre stete Diskussionsbereitschaft gedankt. Von den Mitarbeitern der anderen am Projekt beteiligten Arbeitskreise danke ich besonders Michael Bieler, Thomas Kleinöder und Dr. Lothar Terfloth.

Meinen Kollegen im Arbeitskreis von Prof. Zell bin ich für die stets anregenden Diskussionen, für die fachlichen Ratschläge sowie für die Grillabende auf der Loggia dankbar. Hervorgehoben erwähnen möchte ich Dr. Badreddin Abolmaali, Jörg Kurt Wegner, Felix Streichert und Dr. Simon Wiest.

Ebenso danke ich Prof. Dr. Günter Gauglitz für die Möglichkeit der wissenschaftlichen Mitarbeit in seinem Arbeitskreis vor und nach der Fertigstellung meiner Diplomarbeit. Das gute und offene Arbeitsklima wird mir stets in Erinnerung bleiben. Die hier gewonnenen Einblicke in die Gebiete der physikalischen Chemie und der analytischen Auswertung von Messdaten stellten eine wichtige Grundlage für meine späteren Tätigkeiten dar.

Und nicht zuletzt geht mein Dank natürlich an Frau Dr. Kerstin Länge, die mich bei der Anfertigung dieser Arbeit stets moralisch unterstützt hat und deren Geduld und positiver Zuspruch wesentlich zum Gelingen der Arbeit beigetragen haben.

Fred-Reiner Rapp

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Entwicklung von neuen Arzneistoffen war bis Mitte des letzten Jahrhunderts weitgehend von zufälligen Entdeckungen und von der Weiterentwicklung bereits bekannter Naturstoffe bestimmt. Erst mit dem Aufkommen des sogenannten „rationalen Wirkstoffdesigns“ wurden Methoden zur Vorhersage der Verwendbarkeit einer chemischen Substanz als Wirkstoff entwickelt. Während die ersten Zusammenhänge zwischen der chemischen Struktur einer Substanz und deren Wirkstärke noch von Hand berechnet werden konnten (Hansch-Analyse), stieg mit der Leistungskraft der Computer auch die Komplexität der eingesetzten Methoden. Heute haben sich neuere Methoden der Informationsverarbeitung, wie z. B. die Neuronalen Netze und die Evolutionären Algorithmen, in der Arzneistoffentwicklung fest etabliert. Zusätzlich wurden eine große Anzahl von für dieses Anwendungsgebiet spezifischen Methoden abgeleitet und teilweise auch neu entwickelt. Das Gebiet des Wirkstoffdesigns zeichnet sich heute dadurch aus, dass neue Methoden aus der Informatik mit relativ geringer Verzögerung übernommen und auf ihre praktische Anwendbarkeit überprüft werden.

Seit dem Aufkommen des rationalen Wirkstoffdesigns hat sich das Spektrum der eingesetzten Methoden stetig erweitert. Zwar haben sich neu eingeführte Methoden in manchen Einsatzbereichen als deutlich leistungsfähiger als zuvor bekannte Methoden erwiesen, in den meisten Fällen haben sie die bereits etablierten Methoden aber nicht vollständig abgelöst, sondern nur ergänzt. So sind beispielsweise Neuronale Netze zwar deutlich leistungsfähiger als die seit langem bekannte Multivariate Regression, bei ausschließlich linearen Zusammenhängen im Datensatz ist letztere jedoch aufgrund besserer statistischer Eigenschaften vorzuziehen. Die *Support Vector Machines* haben sich wiederum in vielen Fällen als überlegen gegenüber den Neuronalen Netzen erwiesen, stoßen bei großen Datensätzen allerdings schnell an ihre Grenzen.

Die Frage, welche Methode für welches Problem einzusetzen ist, lässt sich meist nicht allgemeingültig beantworten. Sie hängt in der Regel von der inhärenten Komplexi-

tät des betrachteten Datensatzes ab. Nach dem Prinzip von *Occam's Razor*[1] ist von mehreren Modellen mit vergleichbarer Leistung stets dasjenige zu wählen, welches die geringste Anzahl an freien Parametern (und damit die geringste Komplexität) aufweist. Das ist dann der Fall, wenn das Modell der eingesetzten Methode dem inneren Zusammenhang des Datensatzes gut entspricht.

Dies hat zur Folge, dass bei einem bisher unbekannten Datensatz zuerst mehrere Methoden miteinander verglichen und danach meist noch die Werte der Eingabeparameter der erfolgreichsten Methode weiter verfeinert bzw. optimiert werden müssen. Diese Vorgehensweise ist mühsam, wenn die hierbei eingesetzten Methoden als separate, unzusammenhängende Programme vorliegen und zudem noch jeweils eigene, untereinander nicht kompatible Datenformate verwenden.

Die grundlegende Idee dieser Arbeit besteht daher in der Erstellung eines modularen Programmsystems, welches möglichst alle bereits als nützlich erkannten Methoden unter einer gemeinsamen Benutzungsoberfläche integriert und somit eine schnelle, flexible und sichere Durchführung der oben beschriebenen Vorgehensweise zur explorativen Datenanalyse ermöglicht.

1.2 Einordnung

Diese Arbeit entstand im Rahmen des SOL-Projekts zur „**S**uche und **O**ptimierung von **L**eitstrukturen“, an dem neben drei universitären Arbeitskreisen[2] auch drei Pharmafirmen[3] beteiligt waren. Dabei konnte auf den Erfahrungen des Vorgängerprojekts ÄBAV mit dem Ziel der „**Ä**hnlichkeitsanalyse **B**iologisch **A**ktiver **V**erbindungen“ aufgebaut werden, in dem Neuronale Netze bereits erfolgreich zur Wirkstoffsuche eingesetzt werden konnten. In dem Schlusswort seiner im Rahmen des ÄBAV-Projekts entstandenen Arbeit weist H. SIEMENS [Sie98a, S. 165] allerdings auch auf die beim praktischen Einsatz von Neuronalen Netzen auftretenden Probleme hin:

> *Ganz anders ist der Fall bei den nichtlinearen Methoden gelagert. Ihr Einsatz ist wesentlich aufwändiger und bietet viele Stolperfallen: Statt des globalen Optimums kann meistens nur ein lokales gefunden werden, das gefundene Resultat hängt häufig empfindlich von der anfänglichen*

[1] Das Leben und Werk des Philosophen *William of Ockham* (ca. 1287–1347, lat. auch *Occam*) wird beispielsweise von SPADE [Spa04] beschrieben.

[2] Uni Erlangen-Nürnberg, Institut für Organische Chemie / Computer-Chemie-Centrum (Prof. J. Gasteiger); Uni Marburg, Institut für Pharmazeutische Chemie und Lebensmittelchemie (Prof. G. Klebe); Uni Tübingen, Lehrstuhl Rechnerarchitektur / Zentrum für Bioinformatik (Prof. A. Zell)

[3] ALTANA Pharma AG (vormals Byk Gulden GmbH, Konstanz); Merck KGaA (Darmstadt); Solvay AG (Hannover)

> *Parameterinitialisierung ab, mehrere Untersuchungen des gleichen Problems können zu wiedersprüchlichen Ergebnissen führen, die Interpretation selbst einfacher nichtlinearer Modelle kann sich als schwierig erweisen, und zum Einsatz solcher Verfahren müssen oftmals spezielle Softwarepakete verwendet werden.*

Am Ende des ÄBAV-Projekts standen also eine beachtliche Anzahl von Verfahren mit erwiesener Nützlichkeit zur Verfügung, deren praktischer Einsatz jedoch durch die in dem Zitat aufgeführten Einschränkungen zumindest erschwert, wenn nicht gar verhindert wurde. Als Abhilfe wurde die Entwicklung eines integrierten Programmsystems vorgeschlagen, in welches die Verfahren als Module eingebunden sind und mit Methoden zur Validierung der Ergebnisse kombiniert werden können: SOLVES.

1.3 Aufbau der Arbeit

Das **erste Kapitel** beschreibt die grundlegende Motivation der vorliegenden Arbeit und ordnet sie in ihr Umfeld ein.

Im **zweiten Kapitel** werden die für das Verständnis der Arbeit notwendigen Grundlagen dargelegt. Dies umfasst zum einen Methoden der Datenauswertung, mit den Teilgebieten Modellbildung, Optimierung und Validierung. Insbesondere wird hier auf Neuronale Netze, Evolutionäre Algorithmen und andere Methoden des *Softcomputings* eingegangen. Zum anderen wird der aktuelle Stand des rationalen Wirkstoffdesigns kurz umrissen. Es wird begründet, warum neuere Entwicklungen wie *High Throughput Screening* (HTS) und die kombinatorische Chemie zu einem Paradigmenwechsel geführt haben und wie sich dadurch auch die Anforderungen an Hilfsmittel zur Datenauswertung geändert haben. Besonderes Gewicht wird hierbei auf die für das Liganden-basierte Wirkstoffdesign benötigten Methoden gelegt. Zusätzlich werden die eingesetzten technischen Standards beschrieben. Ihnen kommt in einer Zeit, in denen heterogene Systeme koordiniert zusammenarbeiten sollen, eine nicht zu vernachlässigende Bedeutung zu. Bei der Auswahl wurde darauf geachtet, nur offene und Plattform-übergreifend verfügbare Standards zu verwenden.

Im **dritten Kapitel** wird das im Rahmen dieser Arbeit entwickelte Programmsystem SOLVES beschrieben. Nach einer kurzen Darlegung der Ausgangslage und der relevanten Anforderungen wird der Entwurf und die darauf folgende Implementierung behandelt. Es werden die verschiedenen Programmteile zur Erstellung von Prozessketten, zur Verwaltung der Module, zur Anbindung an Moleküldatenbanken, sowie zur Archivierung und Visualisierung der Ergebnisse vorgestellt. Auf die Eigenschaften der unterschiedlichen Modultypen wird eingegangen. Schließlich werden die Alleinstellungskriterien von SOLVES im Vergleich mit anderen verfügbaren Programmsystemen herausgearbeitet.

Das **vierte Kapitel** beschäftigt sich mit einer Auswahl von Anwendungen des SOLVES Programmsystems. Es werden beispielhaft Möglichkeiten zur Klassifizierung nach chemischen Eigenschaften, nach pharmazeutischen Indikationsgebieten und allgemeiner nach der Wirkstoffähnlichkeit von Molekülen aufgezeigt. Hierbei wird auf das stets auftauchende Problem der Deskriptorselektion eingegangen und es wird untersucht, welche Klassen von Deskriptoren generell zur Lösung vergleichbarer Aufgaben geeignet sind. Die gewonnenen Erkenntnisse werden dann auf das Problem der Vorhersage der Wasserlöslichkeit von chemischen Substanzen angewandt. Die dabei verwendeten SOLVES-Prozessketten werden schematisch dargestellt und jeweils kurz diskutiert.

Im **fünften Kapitel** wird eine neue Methode zur Evolution von Molekül-Substrukturen vorgestellt. Mit dieser kann das in Kapitel 4 häufig auftretende Problem der Auswahl von relevanten Moleküldeskriptoren in gewisser Hinsicht umgangen werden: Für eine bestimmte Klasse von Deskriptoren, den Substruktur-basierten Deskriptoren, kann nun die für eine vorliegende Problemstellung optimale Teilmenge direkt evolviert werden. Hierdurch wird der Zwischenschritt der Generierung möglichst großer Deskriptormengen als Grundlage einer darauf folgenden Selektion vermieden. Nach der Beschreibung der Vorgehensweise werden verschiedene Anwendungsmöglichkeiten im Liganden-basierten Wirkstoffdesign beschrieben.

Im **sechsten Kapitel** werden schließlich die Ergebnisse noch einmal zusammengefasst und ein Ausblick auf weitere mögliche und wünschenswerte Entwicklungen in diesem interessanten Gebiet gegeben.

1.4 Hinweis zu Fachbegriffen und Notation

Fachbegriffe zeichnen sich durch ihre wohldefinierte Bedeutung aus. Unglücklicherweise können dieselben Begriffe in verschiedenen Fachdisziplinen unterschiedliche Bedeutungen haben. So wird beispielsweise der Begriff „Entropie“ in der Informatik anders definiert als in der physikalischen Chemie. Manche Begriffe haben wiederum in einer Fachdisziplin eine genau spezifizierte Bedeutung, werden außerhalb dieses Kontextes jedoch schwammig oder mehrdeutig gebraucht (z. B. der Begriff der „Information“). Das in Anhang B dieser Arbeit befindliche Glossar dient somit nicht nur der Erklärung von Begriffen, welche dem aus einer anderen Fachrichtung stammenden Leser unbekannt sein mögen. Wichtiger fast noch definiert es den Kontext, in dem ein Begriff im Rahmen dieser Arbeit auftaucht. So taucht die Entropie hier nur im Kontext der Informatik auf, was über den entsprechenden Eintrag im Glossar ersichtlich wird. Dasselbe gilt für die verwendeten Symbole und Abkürzungen, welche im Anhang A tabellarisch aufgelistet sind.

Neu eingeführte deutsche Begriffe werden in Anführungszeichen, englische dagegen *kursiv* gesetzt. Firmen-, Produkt- und Eigennamen werden in KAPITÄLCHEN, Programmcode wird in `Maschinenschrift` dargestellt.

Kapitel 2

Grundlagen

In diesem Kapitel werden zum einen die Grundlagen der im Rahmen dieser Arbeit bedeutsamen Methoden zur Auswertung von Daten vorgestellt, wobei die Schwerpunkte auf der Modellierung der inhärenten Zusammenhänge in einem Datensatz (z. B. mit Hilfe von Neuronalen Netzen) und auf der Optimierung dieser Modelle (z. B. mit Evolutionären Algorithmen) liegen. Auch auf die Vorverarbeitung von Daten und die Validierung der erhaltenen Ergebnisse wird eingegangen. Zum anderen wird eine kurze Einführung in das Anwendungsgebiet der Arzneistoffentwicklung bzw. des Teilgebiets der Suche nach neuen Leitstrukturen gegeben. Ziel ist hier die Vermittlung der zum Verständnis notwendigen Grundlagen, wobei ein Schwerpunkt auf die unterschiedlichen Möglichkeiten zur Berechnung von Moleküldeskriptoren gelegt wird, da diese in den folgenden Kapiteln eine zentrale Rolle spielen. Auch auf die Kodierung und Optimierung von Molekülstrukturen wird näher eingegangen. Schließlich werden die im Rahmen der Implementierung verwendeten technischen Standards im Umfeld von XML und CORBA erläutert. Die wichtigsten Fachbegriffe sollen dargelegt und bereits erste Vor- und Nachteile der Technologien aufgezeigt werden.

Sowohl die Neuronalen Netze als auch die Evolutionären Algorithmen haben in den letzten Jahren eine weite Verbreitung in industriellen Anwendungen gefunden. Dabei hat sich gezeigt, dass eine Kombination der beiden Gebiete für viele Aufgabenstellungen sinnvoll ist. Im folgenden wird daher sowohl auf Anwendungsbeispiele als auch auf Beispiele für gelungene Kombinationen eingegangen.

Es sei hier zudem angemerkt, dass eine klare Trennung zwischen modellbildenden und optimierenden Verfahren nicht immer eindeutig durchführbar ist. So werden für das Training eines Neuronalen Netzes in der Regel Gradientenabstiegsverfahren eingesetzt, also vergleichsweise einfache, auf der Richtung des steilsten Gradientenabstiegs basierende Optimierungsverfahren. Andererseits lässt sich mit Hilfe des Genetischen Programmierens ein Regressionsproblem symbolisch lösen, wobei die resultierende Formel das Modell darstellt. Als Überbegriff für auf dem Prinzip der Evolution basierende Optimierungsmethoden wird von LANGDON & POLI [LP02] der Ausdruck

„Suchverfahren“ vorgeschlagen. Unabhängig davon bezeichnen WITTEN & FRANK [WF00] die Methoden des maschinellen Lernens ebenfalls als Suchverfahren. Beides erscheint sinnvoll, da ein wesentliches Merkmal von modellbildenden und optimierenden Verfahren die Suche (nach einem generalisierungsfähigen Modell bzw. einem globalen Optimum) ist. Bedenkt man zudem, dass das erklärte Ziel des Wirkstoffdesigns in der Suche nach neuen Leitstrukturen (und damit nach potentiellen neuen Medikamenten) liegt, so wird ein gemeinsamer Nenner dieses einführenden Kapitels erkennbar.

2.1 Modellbildende Verfahren

In diesem Abschnitt werden Verfahren beschrieben, mit denen der innerhalb eines Datensatzes vorhandene inhärente Zusammenhang modelliert werden kann. Von den vielen bekannten klassischen und modernen Ansätzen werden nur die in dem betrachteten Anwendungsgebiet des Wirkstoffdesigns häufig eingesetzten Verfahren vorgestellt:

- Feedforward-Netze (Multi Layer Perceptrons, MLP)
- Learning Vector Quantization (LVQ)
- Selbstorganisierende Karten (SOM)
- Radiale Basis-Funktionen (RBF)
- Entscheidungsbaum-Verfahren (C4.5)
- Support Vector Machines (SVM)
- Hauptkomponenten-Analyse (PCA)

Mit Ausnahme der PCA handelt es sich dabei um Verfahren aus dem Bereich des maschinellen Lernens.

Eine Einteilung der genannten Verfahren kann nach der Zielstellung erfolgen, mit der sie angewendet werden. Bei den unüberwachten Verfahren werden nur die Eingabevariablen vorgegeben, die jeweilige Methode muss eigenständig einen inneren Zusammenhang (sofern vorhanden) erkennen. Bei den überwachten Verfahren wird zu jedem Eingabemuster entweder die Zugehörigkeit zu einer von mehreren Klassen oder ein numerischer Zahlenwert mitgeliefert. In einer Trainingsphase muss die jeweilige Methode versuchen, den diesen Vorgaben zugrunde liegenden Zusammenhang in ein möglichst passendes Modell zu fassen. Somit ergeben sich die folgenden möglichen Aufgabenstellungen:

Klassifizierung Die Eingabemuster müssen jeweils einer von mehreren Klassen zugeordnet werden. Je nach Verfahren kann zu den vorgegebenen Klassen noch eine weitere Klasse für „unbekannte“ Muster, die keiner der trainierten Klassen zugeordnet werden können, hinzugenommen werden. Manche Verfahren geben Wahrscheinlichkeitswerte für jede Klasse zurück, von diesen kann z. B. der höchste Wert als Zuordnungskriterium verwendet werden (*winner-takes-all*).

Binäre Klassifizierung Ein Sonderfall der Klassifizierung, bei dem zwischen genau zwei Klassen unterschieden wird. Das Ergebnis wird oft als Zuordnungswahrscheinlichkeit angegeben (z. B. mit 70% Wahrscheinlichkeit zu Klasse A gehörig). Durch geeignete Wahl eines Schwellwerts können entweder falsch positive oder falsch negative Zuordnungen reduziert werden.

Regression Das Ziel einer Regression ist die Approximierung eines im Vorfeld nicht ersichtlichen Funktionsverlaufs anhand einer Menge von Eingabemustern und einer numerischen Zielgröße. Eine gute Generalisierung liegt vor, wenn für bislang unbekannte Eingabemuster ein vergleichbarer Vorhersagefehler wie für die trainierten Muster erreicht wird.

Projektion bzw. Dimensionsreduktion Eine Projektion von einem hoch- in einen niederdimensionalen Raum kann beispielsweise mit den Verfahren PCA und SOM durchgeführt werden. Ein Ziel kann dabei die Erklärung der im Datensatz vorkommenden Varianz sein (PCA), oder es kann eine Erhaltung der Nachbarschaft von ähnlichen Datenpunkten im niederdimensionalen Raum angestrebt werden (SOM).

Eine geeignete Vorgehensweise bei der Suche nach dem für eine gegebene Problemstellung besten Modell ist es, stets zuerst den Erfolg von „einfachen“ Verfahren zu prüfen. Diese erkennen triviale Abhängigkeiten im Trainingsdatensatz bzw. liefern nur dann gute Vorhersageergebnisse, wenn solche vorliegen. Komplexere Verfahren können dagegen in vielen Fällen gute Vorhersagen erzeugen, machen triviale Abhängigkeiten aber nicht notwendigerweise erkennbar. So konnte HOLTE [Hol93] zeigen, dass für viele der bis dahin als Vergleichsstandard dienenden Datensätze bereits mit sehr einfachen Verfahren (‘*one rule*’, entspricht dem Vergleich einer Variablen mit einem oder mehreren Schwellwerten) ähnlich gute Vorhersagen wie mit komplexen, als *state-of-the-art* geltenden Verfahren erzielbar waren. Eine Auswahl solch einfacher Verfahren wird beispielsweise von WITTEN & FRANK [WF00] beschrieben.

2.1.1 Feedforward-Netze (Multi Layer Perceptrons)

Die Grundlagen des Gebiets wurden von MCCULLOCH & PITTS [MP43] gelegt. Sie zeigten, dass prinzipiell jede logische oder arithmetische Funktion mit Hilfe eines

Netzwerks aus binären Neuronen mit festem Schwellwert und hemmenden oder verstärkenden Eingangsverbindungen berechnet werden kann. Ein erstes Lernverfahren wurde von HEBB [Heb49] vorgestellt. Das erste praktisch einsetzbare Netzwerk wurde von ROSENBLATT [Ros58] in analoger Hardware entwickelt. Das aufkommende Interesse an Neuronalen Netze wurde von MINSKY & PAPERT [MP69] gebremst, die in dem Buch „Perceptrons“ bewiesen, dass die damals verwendeten einstufigen Netze aus linearen Perceptrons für viele Klassifizierungsprobleme ungeeignet waren. Der große Durchbruch wurde dann durch die Entwicklung der selbstorganisierenden Karte von TEUVO KOHONEN [Koh82] und des Backpropagation-Lernverfahrens durch RUMMELHART [RHW86] markiert. In den darauf folgenden Jahren sind allein zur Kohonen-Karte mehr als 1 500 Artikel veröffentlicht worden [Koh95].

2.1.1.1 Definition und Eigenschaften

Unter dem Begriff „Künstliche Neuronale Netze“[1] werden eine Anzahl zum Teil recht unterschiedlicher Verfahren zusammengefasst. Gemeinsam ist ihnen, dass sie sich an der Funktionsweise von natürlichen Neuronen, wie sie z. B. im Gehirn des Menschen vorliegen, orientieren. Dabei müssen schon aus Gründen der Effizienz und der benötigten Rechenzeit stets Vereinfachungen gegenüber der biologischen Realität vorgenommen werden. Die Hoffnung, dass Neuronale Netze Aufgaben eigenständig oder gar „intelligent“ lösen können, hat sich daher als nicht zutreffend erwiesen.

Auch bei den verwendeten Begriffen ist in gewisser Hinsicht Vorsicht geboten. So wie die NN selber sind viele Fachbegriffe in Analogiebildung zum jeweiligen Vorbild aus der Natur entstanden. Man sollte sich jedoch vor Augen halten, dass Ausdrücke wie „Lernen“ oder „Neuron“ hier nur als Platzhalter für abstrakte Algorithmen und Definitionen stehen, welche höchstens einen Bruchteil der natürlichen Komplexität abbilden können.

Die folgenden charakteristischen Eigenschaften der Neuronalen Netze haben einerseits zu ihrer weiten Verbreitung beigetragen, haben andererseits zum Teil aber auch limitierende Auswirkungen:

Lernfähigkeit: Neuronale Netze können die Lösung eines Problems erlernen, wenn es durch einen ausreichend großen Trainingsdatensatz repräsentiert und durch das Neuronale Netz prinzipiell lösbar ist. Dieses hängt wiederum vom gewählten Netztyp, der Topologie des Netzes und dem verwendeten Lernverfahren ab. Da am Anfang des Lernvorgangs meist von einem zufällig erzeugten Netz ausgegangen wird, spielt auch die Initialisierung eine große Rolle. Der Lernerfolg kann während des Trainings mit einem nur zur Abfrage verwendeten Validierungsdatensatz verfolgt werden. Die abschließende Beurteilung erfolgt mit einem weiteren, unabhängigen Testdatensatz.

[1] Im restlichen Text werden diese vereinfachend als „Neuronale Netze“ (NN) bezeichnet, da eine Verwechslung mit biologischen neuronalen Netzen in diesem Kontext nicht zu befürchten ist.

Robustheit gegen Rauschen: Ein leichtes Rauschen, wie es z. B. bei Messvorgängen stets erzeugt wird, stört die Generalisierungsfähigkeit eines Neuronalen Netzes normalerweise nicht. Die Betonung liegt hierbei auf „leicht“: Tritt eine unvorhergesehen starke Störung des Eingabesignals auf, die durch den Lernprozess nicht abgedeckt wurde, kann daraus eine fast beliebige, unbrauchbare Ausgabe resultieren.

Black-Box-Effekt: Das trainierte Netz erscheint wie eine *black box*, der man den erlernten Lösungsweg nicht ansehen kann. Dadurch wird eine Abschätzung des Grenzbereiches, bis zu dem noch gewünschte Ausgaben erzeugt werden, erschwert. Auch unerwünschte Ausgaben innerhalb des durch das Training abgedeckten Bereiches können nicht generell ausgeschlossen werden, da bei nichttrivialen Aufgabenstellungen in der Regel nicht alle möglichen Eingabewerte durchgetestet werden können. Eine Extraktion des erlernten Wissens oder eine Ableitung von Regeln ist zumindest als schwierig anzusehen, beispielhaft sei die von CECHIN [Cec98] vorgestellte Methode zur Erzeugung von Fuzzy-Regeln aus trainierten Netzen genannt.

Hohe Parallelität: Die Verarbeitung der durch das NN hindurchpropagierten Signale kann parallel in vielen kleinen Recheneinheiten (den Neuronen) erfolgen. Eine Abbildung dieser Struktur auf leistungsfähige Parallelrechner ist daher konzeptionell gesehen vergleichsweise einfach, verschiedene Möglichkeiten wurden von ZELL [Zel94, Kap. 31] zusammenfassend beschrieben. Bei den meisten in der Praxis eingesetzten Software-Simulatoren wird diese Fähigkeit zur Parallelisierung jedoch nicht oder nur in geringem Umfang ausgenutzt.

Neuronale Netze lassen sich zum einen nach der zu lösenden Aufgabenstellung einteilen:

Klassifizierung: Die Eingabevektoren (*patterns*) sind in mehrere diskrete Klassen eingeteilt. Während des Lernvorgangs sollen die Klassengrenzen modelliert werden. Bei der Abfrage kann neben der Angabe der Klasse auch die Ausgabe „unbekannt“ erwünscht sein, wenn das Abfragepattern zu weit von den trainierten Patterns abweicht. Ein Vergleich von 33 verschiedenen Klassifizierungsverfahren (u. a. LVQ und RBF) wurde von LIM, LOH & SHIH [LLS99] erstellt.

Funktionsapproximation: Eine durch die Eingabepatterns vorgegebene, oft unbekannte Funktion soll quantitativ approximiert werden. Dabei ist die Generalisierungsfähigkeit des Netzes wichtig, da der Vorhersagefehler auch für unbekannte Patterns gering sein soll. Werden nur die gelernten Patterns mit geringem Fehler wiedergegeben, so sagt man, dass das Netz die Daten „auswendig gelernt“ hat. Dieser Effekt tritt meist auf, wenn das Netz zu groß gewählt wird, also zu viele freie Parameter enthält. Zur Verkleinerung von Netzen können Pruning-Verfahren eingesetzt werden, siehe z. B. [LDS90].

Zum anderen kann eine Einteilung nach der Art des Lernvorgangs erfolgen:

Überwachtes Lernen: Das zu lernende Ergebnis wird während des Lernens vorgegeben. In jedem Lernschritt wird die Differenz zwischen aktueller und erwünschter Ausgabe ermittelt, und das Netz entsprechend angepasst.

Unüberwachtes Lernen: Es wird keine erwünschte Ausgabe vorgegeben, das Netz muss die vorliegenden Patterns selbständig in Klassen einteilen. Das bekannteste unüberwachte Lernverfahren ist die selbstorganisierende Karte [Koh82]. Bei dem Counterpropagation-Verfahren von HECHT-NIELSEN [HN87] wird aus einer Kohonen-Karte durch Aufsetzen einer weiteren Schicht von Neuronen wieder ein überwachtes Lernverfahren gebildet. Ein Vergleich von verschiedenen unüberwachten Lernverfahren wurde von UPAL & NEUFELD [UN96] vorgenommen.

Bestärkendes Lernen: Ein weitere Variante stellt das bestärkende Lernen (*reinforcement learning*) dar, bei dem während des Trainingsprozesses nur angegeben wird, ob die aktuelle Ausgabe richtig oder falsch ist. Obwohl biologisch plausibler als das überwachte Lernen, kommt es in der Praxis aufgrund der erhöhten Lerndauer seltener zum Einsatz.

2.1.1.2 Aufbau eines Feedforward-Netzes

Feedforward-Netze (oft auch als Multilayer-Perceptrons bezeichnet) zählen zu den am einfachsten aufgebauten und zugleich am häufigsten eingesetzten Arten von NN. Sie können sowohl zur quantitativen Vorhersage als auch zur Klassifizierung eingesetzt werden. Der Lernvorgang erfolgt überwacht, d. h. die gewünschte Ausgabe muss beim Training vorgegeben werden. Als Lernverfahren werden meist modifizierte Versionen des 1986 von Rummelhart vorgestellten Backpropagation-Algorithmus verwendet.

Ein Feedforward-Netz besteht aus Neuronen, die in mehreren Schichten angeordnet sind (vgl. Abb. 2.1). Man unterscheidet dabei zwischen der Eingabeschicht, einen oder mehreren versteckten Schichten und der Ausgabeschicht. In der Eingabeschicht findet keine Verarbeitung der Daten statt, sie dient lediglich zur Einspeisung der Daten in das Netz.[2]

In den Neuronen werden die einkommenden Signale aufsummiert und dann über eine sogenannte Aktivierungsfunktion abgebildet, wobei das Ergebnis vom Wert eines in den Neuronen enthaltenen Schwellwertes θ abhängt (vgl. Abb. 2.1). Dieser Schwellwert kann auf zwei unterschiedliche, jedoch äquivalente Arten dargestellt werden: Auf der linken Seite der Abb. 2.1 ist der Schwellwert θ implizit in den Neuronen enthalten, auf der rechten Seite wird der Schwellwert über Verbindungen zu einem separaten On-Neuron, welches stets die Ausgabe 1 liefert, erzeugt.

[2] Daher wird die Eingabeschicht meist nicht mitgezählt, wenn die Tiefe eines Netzwerks (also die Anzahl der Schichten) angegeben wird.

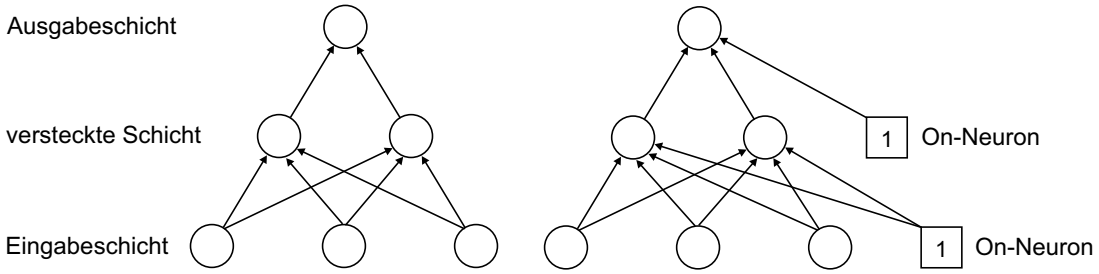

Abbildung 2.1: Zwei alternative Darstellungen eines Feedforward-Netzes mit drei Eingabeneuronen, zwei versteckten und einem Ausgabeneuron.

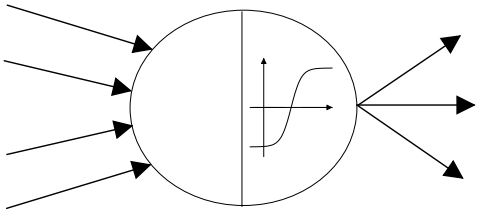

Abbildung 2.2: Schematische Darstellung eines Neurons.

Die Neuronen sind über gewichtete Verbindungen (*links*) verbunden, wobei bei den Feedforward-Netzen nur Verbindungen von den Neuronen einer Schicht zur nächsten zugelassen sind, nicht zu Neuronen der gleichen oder einer vorhergehenden Schicht. Auf diese Weise werden Rückkopplungen, wie sie z. B. bei Jordan- und Elman-Netzen auftreten, ausgeschlossen.

Ein Netz wird als „vollverdrahtet“ bezeichnet, wenn stets sämtliche Neuronen einer Schicht mit allen Neuronen der nächsten Schicht verbunden sind. Eine Besonderheit sind die *shortcuts* genannten Direktverbindungen von der Eingabe- zur Ausgabeschicht.

In den frühen Tagen der NN kamen vor allem binäre Aktivierungsfunktionen zum Einsatz. Diese liefern die Ausgabe 0, solange die Summe der Eingangswerte kleiner als der Schwellwert ist, und die Ausgabe 1 bei Überschreiten des Schwellwerts – das Neuron „feuert“. Für die moderneren, auf dem Prinzip des Gradientenabstiegs beruhenden Lernverfahren werden dagegen sigmoide, stetig differenzierbare Aktivierungsfunktionen eingesetzt.

Die logistische Aktivierungsfunktion beruht auf Gleichung 2.1 und besitzt einen Ausgabebereich von 0 bis 1. Die in Gleichung 2.2 gezeigte *tangens hyperbolicus*-Funktion hat eine ähnliche Form, besitzt jedoch eine etwas stärkere Steigung im Wendepunkt und einen Ausgabebereich von -1 bis 1 (vergleiche Abb. 2.3).

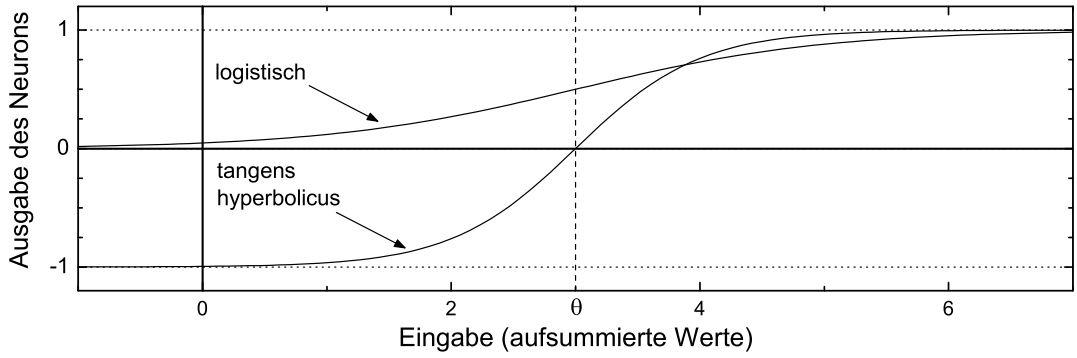

Abbildung 2.3: Zwei sigmoide Aktivierungsfunktionen bei einem Schwellwert $\theta = 3$.

$$f_{\log}(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

$$f_{\tanh}(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

In der Praxis macht es meist keinen großen Unterschied, welche der beiden Aktivierungsfunktionen verwendet wird, da sie ineinander umgerechnet werden können. Wichtig ist jedoch, dass die gewünschten Ausgabewerte in einen vom Netz erreichbaren Bereich skaliert werden. Dieser wird meist etwas kleiner als der Ausgabebereich der Aktivierungsfunktion gewählt (bei der tanh-Funktion z. B. von -0.8 bis 0.8), um den Sättigungsbereich am Rand zu vermeiden.

Auch eine lineare Funktion kann als Aktivierungsfunktion verwendet werden. Zwar sind mehrstufige Netze, die nur aus linearen Neuronen bestehen, nicht sinnvoll – es kann gezeigt werden, dass diese in einstufige Netze[3] transformierbar sind. Wählt man jedoch für die versteckten Neuronen sigmoide und für die Ausgabeneuronen lineare Aktivierungsfunktionen und fügt *shortcuts* hinzu, dann erhält man ein Netz, dass neben nicht-linearen Zusammenhängen zwischen den Ein- und Ausgabedaten auch evtl. vorhandene lineare Anteile sehr gut approximieren kann.

2.1.1.3 Das Backpropagation-Lernverfahren

Während des Lernprozesses wird der Vorhersagefehler des Netzes minimiert. Oft wird die Summe der Fehlerquadrate (*sum square error*, SSE) als Fehlermaß verwendet. Der Lernvorgang entspricht einem Gradientenabstieg auf einer Fehlerkurve, wie in Abb.

[3] Einstufige Netze bestehen nur aus der Eingabe- und der Ausgabeschicht, haben also keine versteckten Neuronen.

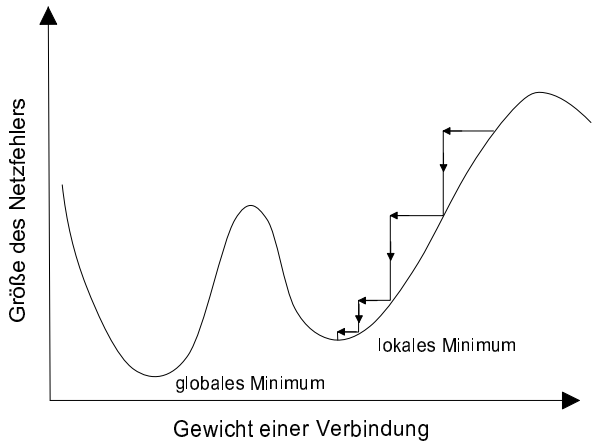

Abbildung 2.4: Gradientenabstieg beim Training von NN mit Backpropagation.

2.4 für ein Verbindungsgewicht (nur ein freier Parameter) dargestellt. Im allgemeinen Fall (mehrere Verbindungsgewichte werden gleichzeitig adaptiert) spricht man auch von einer Fehlerhyperebene.

Im Backpropagation-Verfahren wird üblicherweise eine feste Schrittweite vorgegeben, die sich während des Lernprozesses nicht ändert. Dies kann zu einer Reihe von Problemen führen, z. B. zum Oszillieren in tiefen Tälern [Zel94, Kap. 8]. Daher liegt die wichtigste Verbesserung bei späteren Lernverfahren, wie RPROP (*resilient propagation*) von RIEDMILLER & BRAUN [RB93] und SCG (*scaled conjugate gradient*) von MØLLER [Møl93], in einer adaptiven Anpassung der Schrittweite während des Trainings.

Wie in Abb. 2.4 dargestellt, strebt das Backpropagation-Verfahren während des Trainings stets auf das nächstgelegene lokale Minimum zu. Ohne weiteren Aufwand (wie z. B. einer Kombination mit genetischen Algorithmen) kann nicht erwartet werden, dass ein gutes lokales oder gar das globale Minimum erreicht wird. Zur Validierung sollte ein Feedforward-Netz daher mehrmals mit verschiedenen unterschiedlichen Anfangsbelegungen der Gewichte trainiert und die Ergebnisse statistisch beurteilt werden.

2.1.1.4 Vermeidung des Übertrainierungseffekts

Um den bei Feedforward-Netzen häufig auftretenden Übertrainierungseffekt zu vermeiden, können die folgenden beiden Strategien angewendet werden:

Vorzeitiger Abbruch des Trainings Der Trainingsdatensatz wird um einen gewissen Prozentsatz verkleinert (z. B. 20%), wobei die entfernten Eingabemuster als Validierungsdatensatz verwendet werden. In einem ersten Trainingslauf wird mit

dem verkleinerten Trainingsdatensatz bis zu einer maximalen Anzahl an Lernschritten trainiert. Dabei wird die Lernrate auf dem Validierungsdatensatz mitverfolgt. Im zweiten Schritt wird erneut bis zu derjenigen Anzahl von Lernschritten trainiert, die im ersten Durchlauf den minimalen Validierungsfehler ergab. Der zweite Schritt kann wahlweise mit dem reduzierten oder dem vollen Trainingsdatensatz durchgeführt werden.[4]

Optimierung der Netztopologie Ein Übertrainierungseffekt ist nur dann zu befürchten, wenn das Netz prinzipiell zu einem Auswendiglernen der Eingabemuster fähig ist. Wird es soweit wie möglich verkleinert, so tritt auch nach einem langen Lernvorgang kein Übertrainieren mehr auf. Bei der Topologie-Optimierung kann entweder von einem ausreichend großen Netz ausgegangen werden, welches schrittweise immer weiter verkleinert wird. Ein solches Pruning-Verfahren bricht ab, wenn der Vorhersagefehler zu stark ansteigt. Ein andere Möglichkeit besteht darin, ausgehend von einem leeren Netz Neuronen und Verbindungen schrittweise hinzuzufügen, bis sich keine signifikante Verbesserung mehr ergibt. Beispiele sind das einfache *magnitude based* Pruning-Verfahren, welches stets die Verbindung mit dem absolut kleinsten Gewicht entfernt [Zel94], und das Greedy-Verfahren zum Netzaufbau, welches in jedem Schritt alle Einfügemöglichkeiten prüft und stets „gierig“ diejenige mit der größten Verbesserung des Lernfehlers auswählt [Rap96].

2.1.2 Lernende Vektorquantisierung (LVQ)

Bei der *Learning Vector Quantization* (LVQ) handelt es sich um ein überwachtes Klassifizierungsverfahren. Es wurde von KOHONEN [Koh90] entwickelt und stellt einen direkten Vorläufer des im nächsten Abschnitt beschriebenen SOM-Verfahrens dar.

Aufbau des Neuronalen Netzes Für das LVQ-Verfahren wird eine fest vorgegebene Menge von *Codebook-Vektoren* eingesetzt. Diese sind mit den aktiven Neuronen eines einschichtigen MLPs vergleichbar, welche mit den Eingabeneuronen vollständig verbunden sind. Die Codebook-Vektoren sind durch einen Index j gekennzeichnet, bestehen aus einem Gewichtsvektor $\boldsymbol{w}_j$ und sind jeweils mit dem Namen einer Klasse versehen. Jede der im Eingabedatensatz vorhandenen Klassen wird durch einen oder mehrere Codebook-Vektoren beschrieben. Diese Zuteilung ändert sich nach der Initialisierungsphase nicht mehr.

Abfrage eines Musters Für jedes Eingabemuster $\boldsymbol{x}$ wird nach dem *winner-takes-all*-Prinzip der nächstgelegene Codebook-Vektor $\boldsymbol{w}_c$ ermittelt. Als Entfernungsmaß wird dabei meist die euklidische Distanz (L_2-Norm) verwendet. Es gilt also:

[4] Besonders bei kleinen Trainingsdatensätzen ist letzteres empfehlenswert, um mit so vielen Eingabemustern wie möglich trainieren zu können.

$$\|\boldsymbol{x} - \boldsymbol{w}_c\| = \min_j(\|\boldsymbol{x} - \boldsymbol{w}_j\|), \quad \text{mit} \quad j \in \{1, \ldots, m\} \tag{2.3}$$

Dabei steht m für die Anzahl der Codebook-Vektoren.

Lernverfahren Während des Trainings wird in jedem Lernschritt der zu dem aktuellen Trainingspattern $\boldsymbol{x}$ nächstliegende Codebook-Vektor bestimmt. Gehört er der gleichen Klasse wie das Pattern an, dann wird er proportional zu einer festgelegten Lernrate (die im Laufe des Lernprozesses abnehmen kann) in Richtung des Patterns verschoben. Ist er einer anderen Klasse zugeordnet, dann wird er entsprechend in die Gegenrichtung bewegt.[5] Für die Änderung $\Delta\boldsymbol{w}_c$ der Gewichte dieses Codebook-Vektors gilt:

$$\Delta\boldsymbol{w}_c = \begin{cases} +\eta(t)\,(\boldsymbol{x} - \boldsymbol{w}_c) & : \quad \text{Klasse}(\boldsymbol{w}_c) = \text{Klasse}(\boldsymbol{x}) \\ -\eta(t)\,(\boldsymbol{x} - \boldsymbol{w}_c) & : \quad \text{Klasse}(\boldsymbol{w}_c) \neq \text{Klasse}(\boldsymbol{x}) \end{cases} \tag{2.4}$$

Dabei steht $\eta(t)$ für die Lernrate zum Zeitpunkt t, wobei $0 < \eta(t) < 1$ gelten muss. Die Gewichte der weiter entfernten Codebook-Vektoren werden in diesem Lernschritt nicht verändert, es gilt also $\Delta\boldsymbol{w}_j = 0$ für alle $j \in \{1, \ldots, m\}$ mit $j \neq c$.

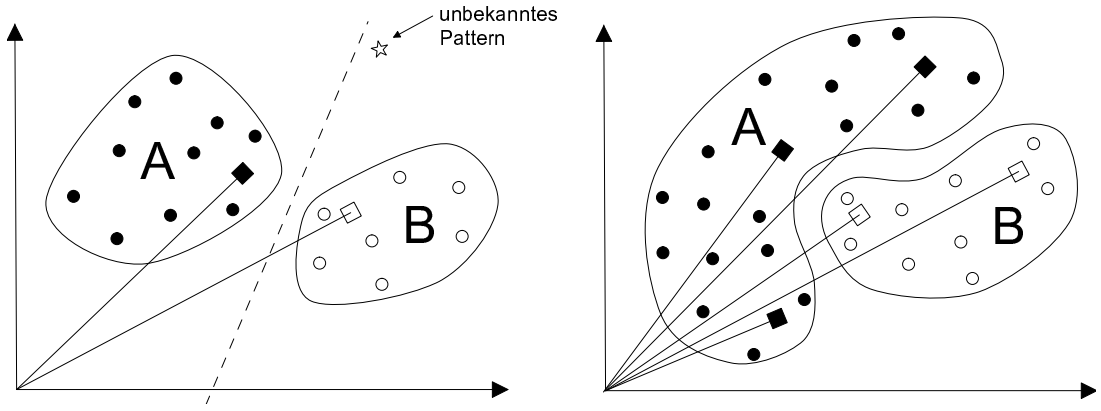

Abbildung 2.5: Separierung zweier Klassen A und B mit Hilfe von LVQ anhand zweier unterschiedlicher Pattern-Verteilungen. Die Patterns sind als Kreise dargestellt, die Codebook-Vektoren als Quadrate. Auf der linken Seite ist zusätzlich ein nicht zugeordnetes Pattern sternförmig dargestellt. Die gestrichelte Linie links markiert die Klassengrenze.

[5] Neben dieser recht einfachen, als LVQ1 bekannten Lernregel wurden von KOHONEN [Koh90] noch weitere Varianten namens LVQ2.1, LVQ3 und OLVQ1 entwickelt, auf die hier nicht näher eingegangen wird (vgl. [Zel94, Kap. 14]).

In Abb. 2.5 ist auf der linken Seite eine einfache Verteilung zweier Klassen A und B gezeigt, für deren Separierung zwei Codebook-Vektoren ausreichen. Für die kompliziertere Aufteilung auf der rechten Seite werden fünf Codebook-Vektoren benötigt.

Problematisch an LVQ ist, dass die Anzahl der zur Verfügung stehenden Codebook-Vektoren vor Beginn des Trainings festgelegt werden muss. Eventuell müssen mehrere Versuche mit jeweils unterschiedlicher Anzahl gestartet werden, um auf ein gutes Ergebnis zu kommen. In der LVQ-Implementierung von KOHONEN *et al.* [KHK+96] kann zu Beginn gewählt werden, ob die Codebook-Vektoren auf die Klassen gleich oder entsprechend der Anzahl der Samples pro Klasse verteilt werden sollen. Zusätzlich kann ein *balancing* genannter Schritt durchgeführt werden, bei dem die Abstände der Codebook-Vektoren in den einzelnen Klassen angeglichen werden können.[6] Im Stuttgarter Neuronale Netze Simulator [ZMV+95] ist eine DLVQ (dynamisches LVQ) genannte Implementierung enthalten, bei der eine passende Anzahl von Codebook-Vektoren automatisch bestimmt wird. Zu Beginn des Trainings wird hier jeder Klasse nur der Durchschnittsvektor aller Patterns dieser Klasse zugeordnet. Nach einer Lernrunde werden weitere Codebook-Vektoren eingefügt, bis entweder alle Samples richtig zugeordnet sind oder eine vorgegebene Grenze erreicht ist.

Ein grundsätzliches Problem von LVQ liegt darin, dass die Dichteverteilung der Patterns keine Berücksichtigung findet. Auch ein Pattern, das sehr weit vom „Zentrum“ einer Klasse entfernt ist, kann dieser noch zugeordnet werden (vgl. das unbekannte Pattern in Abb. 2.5). Wenn dieses Verhalten nicht tolerierbar ist, dann sollten andere Verfahren wie RBF zum Einsatz kommen.

2.1.3 Selbstorganisierende Karten (SOM)

Kohonen-Karten (*self organizing maps*, SOM) stellen ein unüberwacht lernendes Klassifizierungsverfahren dar [Koh82, Koh95]. Ähnlich wie bei LVQ werden Codebook-Vektoren verwendet, die hier jedoch über eine Nachbarschaftsfunktion miteinander verbunden sind. Während des Lernvorgangs wird also nicht nur der zum Eingabemuster nächstliegende Codebook-Vektor, sondern auch die in einem bestimmten Umfeld liegenden Nachbarn verschoben. Somit wird eine topologieerhaltende Abbildung von einem hochdimensionalen Raum (dem der Eingabevektoren) in einen meist zweidimensionalen Raum (den der Karte) erreicht. Muster, die im Eingaberaum im selben Cluster liegen, sollen auch auf benachbarte Neuronen der Kohonen-Karte abgebildet werden.

Ob dieses Ziel erreicht wird, hängt von der Verteilung der Eingabemuster ab. Sind diese zufällig oder in vielen kleinen Clustern verteilt, so wird die Karte wahrscheinlich kein interpretierbares Ergebnis liefern. Wenige, gut abgegrenzte Cluster sollten aber auch auf der Karte erkennbar sein.

[6] Genauer gesagt werden die Median-Werte der kleinsten Abstände der Codebook-Vektoren einer Klasse angeglichen.

Aufbau des Neuronalen Netzes Die Kohonenkarte besteht nur aus einer Schicht von aktiven Neuronen. Jedes Neuron der Karte ist mit sämtlichen Eingabeneuronen verbunden. Die Neuronen der Karte sind in einem ein- oder mehrdimensionalen Gitter angeordnet (meist ist dieses zweidimensional), sind aber nicht miteinander verbunden. Die geometrische Anordnung der Neuronen wird bei der Auswertung der Nachbarschaftsfunktion während des Trainings berücksichtigt.

Abfrage eines Musters Bei dem Anlegen eines Eingabemusters $\boldsymbol{x}$ an die Karte wird nach dem *winner-takes-all*-Prinzip dasjenige Neuron der Karte ermittelt, welches diesem Muster am ähnlichsten ist. Als Metrik wird meist die euklidische Norm verwendet. Für ein Gewinnerneuron c mit Gewichtsvektor $\boldsymbol{w}_c$ gilt also:

$$\|\boldsymbol{x} - \boldsymbol{w}_c\| = \min_j(\|\boldsymbol{x} - \boldsymbol{w}_j\|), \quad \text{mit} \quad j \in \{1, \ldots, m\} \tag{2.5}$$

Dabei steht m für die Anzahl der Kartenneuronen. Bei normierten Vektoren kann auch das Skalarprodukt $\langle \boldsymbol{x} \,|\, \boldsymbol{w}_j \rangle = \sum_{i=1}^{k} x_i w_{ij}$ verwendet werden, wobei entsprechend das Neuron mit dem maximalen Skalarprodukt ausgewählt wird:

$$\langle \boldsymbol{x} \,|\, \boldsymbol{w}_c \rangle = \max_j(\langle \boldsymbol{x} \,|\, \boldsymbol{w}_j \rangle), \quad \text{mit} \quad j \in \{1, \ldots, m\} \tag{2.6}$$

Lernverfahren In jedem Lernschritt wird der Gewichtsvektor des Gewinnerneurons verändert. Zusätzlich werden in einem geringerem, nach topologischen Abstand abgestuften Maße die Gewichtsvektoren der benachbarten Neuronen angepasst. Für die Änderung $\Delta \boldsymbol{w}_j$ der Gewichtsvektoren gilt:

$$\Delta \boldsymbol{w}_j = \eta(t)\, h(z,t)\, (\boldsymbol{x} - \boldsymbol{w}_j), \quad \text{wobei} \quad z = \|\boldsymbol{r}_c - \boldsymbol{r}_j\| \tag{2.7}$$

Für die Lernrate $\eta(t)$ zum Zeitpunkt t muss $0 < \eta(t) < 1$ gelten. Im Laufe des Trainings wird sie kontinuierlich verringert. Die Nachbarschaftsfunktion $h(z,t)$ hängt vom Zeitpunkt t und von der Entfernung z ab, welche zwischen der Position $\boldsymbol{r}_c$ des Gewinnerneurons c auf der Karte und der Position $\boldsymbol{r}_j$ eines benachbarten Neurons j liegt. Als Nachbarschaftsfunktion wird häufig die normalisierte Gaußfunktion eingesetzt:

$$h_{\text{gauss}}(z,t) = \frac{1}{\sigma(t)\sqrt{2\pi}} \cdot \exp\left(\frac{-z^2}{2\sigma^2(t)}\right) \tag{2.8}$$

Die Varianz $\sigma(t)$ der Gaußfunktion ist ein Maß für die Ausdehnung der Nachbarschaftsfunktion zum Zeitpunkt t. Sie wird im Laufe des Trainings reduziert mit $\sigma(t) \to 0$ für $t \to \infty$.

Wird eine Kohonen-Karte mehrmals trainiert, so entstehen unterschiedliche Cluster-Verteilungen, die sich jedoch in der Regel durch entsprechende Symmetrieoperationen

(spiegeln, drehen, verschieben) ineinander überführen lassen. In der Implementierung von KOHONEN *et al.* [KHKL96] gibt es die Möglichkeit, Eingabemuster an bestimmte Regionen der Karte „anzuheften“, so dass eine stabile Verteilung erreicht wird.

2.1.4 Radiale Basisfunktionen-Netze (RBF)

Bei dem RBF-Verfahren handelt es sich um ein überwachtes Lernverfahren, welches zur Klassifizierung und zur Funktionsapproximation geeignet ist [PG89]. Die Approximation einer Zielfunktion erfolgt durch Überlagerung von radialsymmetrischen Funktionen, deren Stützstellen (Positionen im Eingaberaum) während des Lernvorgangs bestimmt werden.

Aufbau des Neuronalen Netzes RBF-Netze enthalten in der Regel nur eine Schicht von versteckten Neuronen. Die versteckten Neuronen sind mit den Eingabe- und den Ausgabeneuronen jeweils voll verdrahtet. Sie haben jeweils eine radialsymmetrische Aktivierungsfunktion h_j, wobei häufig eine Gaußfunktion verwendet wird:

$$h_j(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{w}_j\|}{2\sigma_j^2}\right) \tag{2.9}$$

Dabei bezeichnet $\boldsymbol{x}$ das anliegende Eingabemuster, $\boldsymbol{w}_j$ den Gewichtsvektor des versteckten Neurons j und σ_j^2 die Varianz der Gaußfunktion. Die Ausgabeneuronen besitzen eine lineare oder sigmoide Aktivierungsfunktion.

Abfrage eines Musters Ein an das Netz angelegtes Eingabemuster wird wie bei den MLPs erst zu den Neuronen der versteckten Schicht, dann zu den Ausgabeneuronen propagiert. Die Ausgabe wird entsprechend der Problemstellung (Approximation oder Klassifizierung) ausgewertet.

Lernverfahren Ein RBF-Netz kann prinzipiell mit einem modifizierten Backpropagation-Verfahren trainiert werden. Effizienter ist eine hybride Lernstrategie, bei der zuerst die Stützstellen und weitere Parameter der Basisfunktionen ermittelt werden. In einem zweiten Schritt werden dann die Ausgabeneuronen mit Backpropagation trainiert [MD89]. Wird als radialsymmetrische Funktion die Gaußfunktion verwendet, so kann das Netz sogar in einem einzigen, allerdings aufwändigen Schritt trainiert werden [Zel94, Kap. 20].

Als ein Vorteil von RBF-Netzen lässt sich die Lokalität der verwendeten Basisfunktionen ansehen. Außerhalb des trainierten Bereiches verringert sich die Ausgabe des Netzes und geht schließlich gegen Null. Die Gefahr einer Fehlklassifizierung von nicht durch das Training abgedeckten Abfragemustern verringert sich dadurch im Vergleich

zu MLPs. Ein weiterer Vorteil gegenüber MLPs liegt darin, dass die in einem trainierten RBF-Netz enthaltenen Informationen (vor allem die Lage der Stützstellen der Basisfunktionen) einfach interpretierbar sind. Der bei MLPs auftretende *black box*-Effekt wird so vermieden. Ein Problem des Verfahrens ist die oft schlechtere Generalisierung. Im Fall der quantitativen Approximation kann das Ergebnis durch Überlagerung mit Polynomen verbessert werden [ZMV+95].

2.1.5 Entscheidungsbaumverfahren

Die weit verbreiteten Entscheidungsbäume (*decision trees*) eignen sich besonders für die Vorhersage von diskreten Werten und werden daher meist für Klassifizierungsaufgaben eingesetzt. Eine Vorhersage von numerischen Werten ist ebenfalls möglich (vgl. [WF00, Kap. 6.5]), wird aber selten eingesetzt und daher an dieser Stelle nicht weiter vertieft.

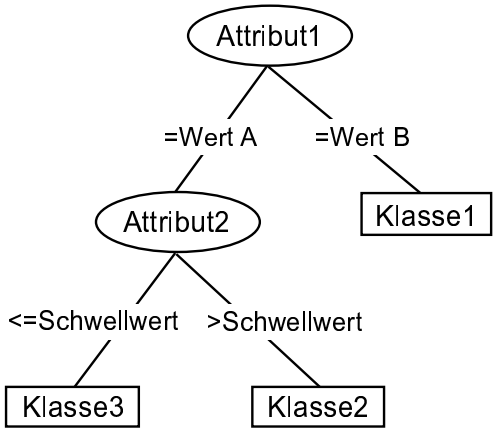

Abbildung 2.6: Beispiel für einen Entscheidungsbaum. Das zuerst getestete Attribut ist nominal mit den möglichen Werten A und B. Das zweite Attribut ist numerisch, daher wird auf der zweiten Stufe nach einem Schwellwert getestet.

Als Eingabe können nominale Attribute (mit einem diskreten Wertebereich) oder numerische Attribute (mit einem ganzzahligen oder reellwertigen Wertebereich) verwendet werden. Die Erstellung eines Entscheidungsbaums erfolgt rekursiv, indem vom Wurzelknoten ausgehend wiederholt das geeignetste Attribut für einen Test ausgewählt wird (vgl. Abb. 2.6). Als Kriterium dient hierbei die Maximimierung des „Informationsgewinns“ (*information gain*). Die Menge der Eingabemuster wird dem Testergebnis entsprechend auf die Kindknoten verteilt. Das Verfahren endet, wenn sich in einem Kindknoten nur noch Muster einer Klasse befinden, oder wenn sich kein weiterer Informationsgewinn mehr erzielen lässt.

Zu den grundlegenden Konzepten für die Beurteilung eines neuen Knotens in einem Entscheidungsbaums zählt das der „Entropie“. Diese ist nach MITCHELL [Mit97] wie folgt definiert:

$$\text{entropy}(S) = \sum_{i=1}^{k} -p_i \log_2 p_i \tag{2.10}$$

S steht für die Menge der Eingabemuster und k für die Anzahl der Klassen. p_i bezeichnet den Anteil der zur Klasse i gehörenden Eingabemuster. Die erwartete Abnahme der Entropie bei Einführung einer neuen Regel wird Informationsgewinn (*information gain*) genannt. Es gilt:

$$\text{gain}(S, A) = \text{entropy}(S) - \sum_{v \in V_A} \frac{|S_v|}{|S|} \text{entropy}(S_v) \tag{2.11}$$

Dabei ist V_A die Menge aller möglichen Werte des Attributs A. S_v ist diejenige Untermenge von S, welche alle Eingabemuster mit Attributwert v enthält, für die also $S_v = \{s \in S \mid A(s) = v\}$ gilt.

Zu den bekannten Entscheidungsbaumverfahren zählen ID3 und dessen Nachfolger C4.5 [Qui86, Qui93]. Eine wichtige Erweiterung von C4.5 im Vergleich zu ID3 sind die Knoten-Reduzierungsverfahren (*node pruning*), welche nach dem Aufbau des Entscheidungsbaums unnötige Knoten entfernen und damit die Generalisierungsfähigkeit verbessern sollen. Dabei können Teilbäume durch eines ihrer Blätter ersetzt werden (*subtree replacement*) oder es kann ein Teilbaum an die Stelle seines Elternknotens treten (*subtree raising*) [WF00].

2.1.6 Support Vector Machines

Das Gebiet der *Support Vector Machines* (SVM) wurde von VAPNIK [Vap95, Vap98] begründet. SVM eignen sich in ihrer ursprünglichen Form für binäre Klassifizierungsaufgaben. Es wurden zahlreiche Methoden für den Einsatz mit mehr als zwei Klassen vorgeschlagen, wobei meist mehrere SVM-Läufe nach einem bestimmten Schema kombiniert werden. Von SCHÖLKOPF *et al.* [SBSW99] erfolgte die Erweiterung des Verfahrens für numerische Regressionsprobleme.

SVM können als eine Weiterentwicklung der Linearen Diskriminanz-Analyse (LDA) betrachtet werden. Die LDA trennt die Instanzen zweier Klassen in einem n-dimensionalen Raum durch eine $(n-1)$-dimensionale Hyperebene (*hyperplane*). Diese hat die Eigenschaft, dass sie den maximal möglichen Abstand zu beiden Klassen besitzt. Der Nachteil von LDA besteht darin, dass die meisten nicht-trivialen Klassifizierungsaufgaben in ihrem Ursprungs-Koordinatensystem nicht linear separierbar sind.

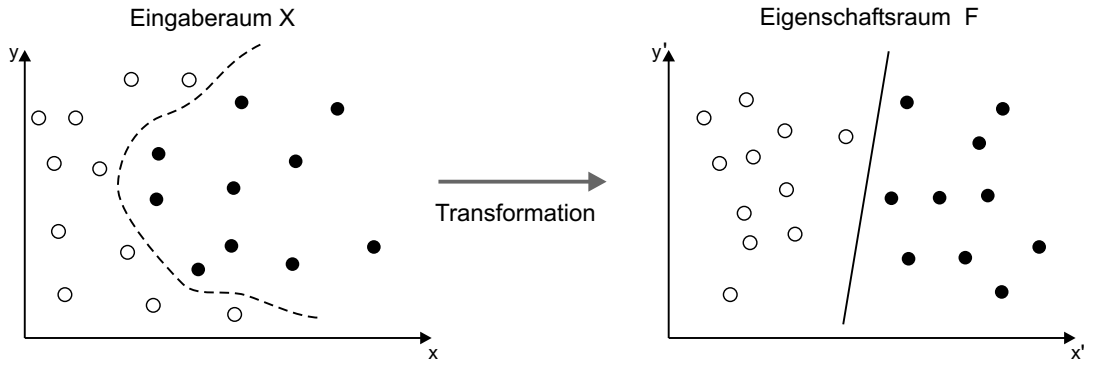

Abbildung 2.7: Schematische Darstellung der Transformation des Eingaberaums X in den Eigenschaftsraum F. Die hellen und dunklen Kreise stellen die Eingabemuster zweier Klassen dar. Eine lineare Separierung der beiden Klassen ist nur in F möglich.

Daher werden beim SVM-Verfahren die Eingabemuster (bzw. Eingabevektoren) über eine nichtlineare Abbildung in einen höherdimensionalen Eigenschaftsraum F transformiert. Die abbildende Funktion ϕ wird so gewählt, dass eine lineare Separierung der Klassen in F prinzipiell möglich ist.[7] Die eingesetzten Algorithmen bzw. Lernverfahren werden umformuliert (in eine sogenannte „Dualform“ gebracht), so dass die Transformation der Eingabevektoren in den Eigenschaftsraum nur innerhalb eines Skalarprodukts in die Berechnungen mit eingeht. Mit Hilfe von speziellen „Kernelfunktionen“ kann dieses Skalarprodukt direkt berechnet werden, auf die eigentliche Transformation der Eingabemuster kann verzichtet werden. Neben polynomiellen Kerneln werden häufig auch sigmoide oder auf radialen Basisfunktionen basierende eingesetzt.

Sei $X \subseteq \mathbb{R}^n$ die Menge der Eingabevektoren $\boldsymbol{x} = (x_1, \ldots, x_k)'$ und $Y = \{-1, 1\}$ die Ausgabemenge einer binären Klassifizierung. Ein Trainingsdatensatz besteht aus der Menge von Tupeln $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)) \subseteq (X \times Y)^n$. Das Problem der binären Klassifizierung kann als das Erlernen einer Funktion $f : X \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ angesehen werden, wobei mit der Vorzeichenfunktion $\mathrm{sgn} : \mathbb{R} \rightarrow Y$ die Abbildung des reellwertigen Ausgabewerts auf die Klasse vorgenommen wird:

$$\mathrm{sgn}(x) = \begin{cases} 1 & : \quad x \geq 0 \\ -1 & : \quad x < 0 \end{cases} \tag{2.12}$$

Die Transformationsfunktion $\phi : x \in X \rightarrow \phi(x) \in F$ bildet den Eingaberaum X in einen höherdimensionalen Eigenschaftsraum F ab.

Neben statistischen Verfahren (wie LDA oder PCA) lassen sich auch viele Verfahren des Maschinellen Lernens in einer „Primärform“ (meist ist damit die bislang etablier-

[7] Allerdings ist eine lineare Separierung nicht immer erstrebenswert, da viele reale Datensätze Rauschen und evtl. auch Messfehler enthalten. Eine vollständige lineare Separierung würde in diesen Fällen einem Auswendiglernen der Daten (*overfitting*) entsprechen.

te Form gemeint) und in einer äquivalenten „Dualform“ ausdrücken. Das wesentliche Merkmal der Dualform ist, dass der Eigenschaftsraum F nur über ein Skalarprodukt in die Berechnungen mit eingeht. Dieses Skalarprodukt wird indirekt mit einer Kernelfunktion $K(\boldsymbol{x}, \boldsymbol{z})$ berechnet. Für diese muss gelten, dass der Ausgabewert von K stets dem Skalarprodukt der beiden Eingabevektoren $\boldsymbol{x}$ und $\boldsymbol{z}$ im Eigenschaftsraum F entspricht, somit:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}) \,|\, \phi(\boldsymbol{z}) \rangle \,, \qquad \text{für alle } \boldsymbol{x}, \boldsymbol{z} \in X \tag{2.13}$$

Im folgenden sollen die Primär- und die Dualform eines einzelnen Perzeptron-Ausgabeneurons dargestellt werden (vgl. [BH03, Kap. 5]).[8] Ein Perzeptron bildet einen Eingabevektor $\boldsymbol{x}$ über eine lineare Funktion $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ auf einen reellen Ausgabewert $f(\boldsymbol{x})$ ab:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w} \,|\, \boldsymbol{x} \rangle + b = \sum_{i=1}^{n} w_i x_i + b \tag{2.14}$$

Dabei steht $\boldsymbol{w}$ für den Gewichtsvektor und b für den *bias* des Perzeptrons.[9] Die Klasse des Ausgabewertes wird wieder über die Vorzeichenfunktion $\operatorname{sgn}(f(\boldsymbol{x}))$ ermittelt.

Im folgenden wird ein einfaches, bereits von ROSENBLATT [Ros58] beschriebenes Lernverfahren für das Perzeptron verwendet. Es geht initial von einem auf Null gesetzten Gewichtsvektor und Bias aus, die Eingabevektoren seien normiert. Es wird solange abwechselnd mit allen Eingabemustern trainiert, bis sich in einer Runde keine Änderung mehr ergibt. In jedem Lernschritt wird die für das Eingabemuster j vorhergesagte Klasse $\hat{y}_j$ mit der vorgegebenen Klasse y_j verglichen.

Weichen diese voneinander ab, so werden der Gewichtsvektor mit $\Delta\boldsymbol{w} = y_j \boldsymbol{x}_j$ und der Bias mit $\Delta b = y_j$ verändert.[10]

Für die Herleitung der Dualform dieser einfachen Lernregel ist nun die Beobachtung entscheidend, dass der Gewichtsvektor $\boldsymbol{w}$ nach dem Training eine Linearkombination der Eingabemuster darstellt, es gilt $\boldsymbol{w} = \sum_{j=1}^{m} \alpha_j y_i \boldsymbol{x}_j$. Der Wert α_j steht dabei für die Anzahl der Änderungen des Gewichtsvektors, die vom Eingabemuster j während des Trainings verursacht wurden. Wenn die Trainingsmenge S festgelegt ist, so stellt der Vektor $\boldsymbol{\alpha}$ eine alternative (duale) Repräsentation des Gewichtsvektors $\boldsymbol{w}$ dar [CST00].

[8] Im folgenden wird nur noch kurz von „Perzeptron“ gesprochen, womit ein einzelnes Ausgabeneuron eines einstufigen, historischen Perzeptrons gemeint (vgl. [Zel94, Kap. 7]). Im Gegensatz zum historischen Perzeptron verwendet das hier besprochene reellwertige Eingabevektoren.

[9] Anstelle des Bias b kann auch der Schwellwert (*threshold*) $\theta = -b$ eingesetzt werden.

[10] Bei nicht-normierten Eingabevektoren berechnet sich die Änderung des Bias zu $\Delta b = y_j r_{\max}^2$, wobei $r_{\max} = \max_j \|\boldsymbol{x}_j\|$.

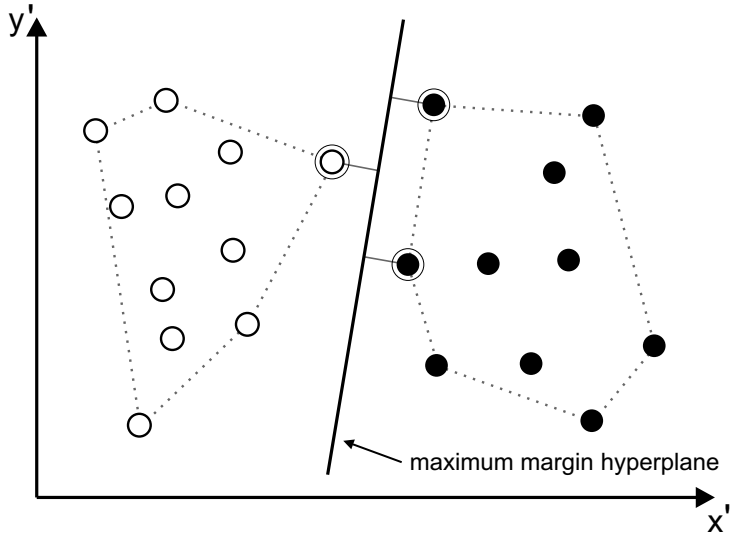

Abbildung 2.8: Beispiel für eine maximal trennende Hyperebene (*maximum margin hyperplane*). Die hellen und dunklen Kreise stellen die Eingabemuster zweier Klassen nach der Kernel-Transformation dar. Die gestrichelten Linien zeigen die konvexen Hüllen der beiden Klassen an. Die markierten Eingabemuster dienen als Support-Vektoren.

2.2 Evolutionäre Optimierung

Die Grundidee der Evolutionären Algorithmen (EA) liegt darin, das von DARWIN [Dar59] beschriebene Prinzip der natürlichen Evolution zu abstrahieren und auf den Computer abzubilden. Hierzu wurden parallel unterschiedliche Vorgehensweisen entwickelt, die heute als Teilgebiete der EA angesehen werden. Das Gebiet der Evolutionären Algorithmen teilt sich auf in:

- Genetische Algorithmen (GA)
- Evolutionsstrategien (ES) und Evolutionäres Programmieren (EP)
- Genetisches Programmieren (GP)

Die Genetischen Algorithmen (GA) bilden die Beschreibung des jeweils zu optimierenden Problems auf einen Binärstring ab. Bei den Evolutionsstrategien (ES) werden reelle Zahlen zur Problembeschreibung eingesetzt, bei der Genetischen Programmierung (GP) wird das Individuum in einer Baumstruktur kodiert.

Die einzelnen Teilgebiete werden im folgenden noch detaillierter beschrieben. Auf das von FOGEL [FOW66] geprägte Verfahren der Evolutionären Programmierung wird

jedoch nicht näher eingegangen, da es in praktischen Anwendungen heutzutage nur eine vergleichsweise geringe Rolle spielt.

Die obige kurze Charakterisierung deutet bereits an, dass je nach gegebener Problemstellung das eine oder andere Verfahren vorzuziehen ist. In den letzten Jahren ist ein Trend zu „hybriden“ Evolutionären Algorithmen zu beobachten. Dabei werden die Methoden untereinander oder mit anderen Verfahren kombiniert. Die Repräsentierung erfolgt in der für die Problemstellung angemessensten Form, die Operatoren werden speziell auf die jeweilige Repräsentierung zugeschnitten.

Um solchen pragmatischen, dem „gesunden Menschenverstand“ folgenden Ansätzen zur Wahl der jeweils problemspezifisch besten Kodierung und Selektion mehr Gewicht zu verleihen, entstanden schon relativ früh Vorschläge für das ganze Gebiet umfassende und teilweise vereinheitlichende Methodiken. So wurde von MICHALEWICZ [Mic96] der Begriff *Evolution Programs* für auf komplexeren Datenstrukturen arbeitende Genetische Algorithmen geprägt. Das Einbringen von fachspezifischem Vorwissen wird durch die Wahl einer geeigneten Kodierung oft wesentlich erleichtert oder überhaupt erst ermöglicht. Deutlich wird dies beispielsweise bei den in Abschnitt 2.4.8 beschriebenen Verfahren zur Optimierung von Molekülstrukturen.

Welches der hier vorgestellten Optimierungsverfahren ist das „beste“? Diese Frage lässt sich mit Hilfe des *no free lunch theorems* (NFL) beantworten. Dabei handelt es sich um einen theoretischen Beweis der Aussage, dass kein Optimierungsverfahren auf der Menge aller möglichen Optimierungsprobleme besser als ein beliebiges anderes Optimierungsverfahren sein kann [WM97]. Eine zufallsgesteuerte Suche ist damit im Durchschnitt genauso erfolgreich wie die hier besprochenen Verfahren. Die praktische Konsequenz des NFL besteht allerdings lediglich darin, dass für unterschiedliche Klassen von Optimierungsproblemen jeweils dasjenige Verfahren gesucht werden muss, welches von seinen Stärken und Einschränkungen her am besten geeignet und somit für diese spezielle Problemklasse am erfolgreichsten ist.

Evolutionäre Algorithmen haben eine weite Verbreitung in den verschiedensten Anwendungsbereichen in Forschung und Industrie gefunden [BVM97]. Auf Anwendungen aus dem Bereich der Chemoinformatik wird in Kapitel 4 noch ausführlich eingegangen.

2.2.1 Genetische Algorithmen

Das Gebiet der Genetischen Algorithmen (GA) wurde mit den Arbeiten von HOLLAND [Hol62, Hol75] über adaptive Systeme begründet. Die erste Anwendung auf Optimierungsprobleme erfolgte von DE JONG [De 75]. Eine weite Verbreitung im Sinne eines „endgültigen Durchbruchs“ des Gebiets wurde durch das vielzitierte Textbuch von GOLDBERG [Gol89] erreicht.

2.2.1.1 Beschreibung des Verfahrens

Im folgenden werden die grundlegenden Eigenschaften der Genetischen Algorithmen kurz umrissen, wobei besonders die zur Abgrenzung gegenüber anderen Verfahren geeigneten Eigenschaften angesprochen werden.

Repräsentation der Individuen Die Individuen werden in Form eines Bitstrings kodiert. Dieser kann als „Genotyp“ angesehen werden, auf den die den „Phänotyp“ darstellenden Datentypen (beispielsweise ganze oder reellwertige Zahlen) mittels einer geeigneten Kodierung abgebildet werden. Gut geeignet sind Kodierungen, bei denen kleine Änderungen des Bitstrings geringe Auswirkungen auf den Wert des Datentyps haben (wie z. B. der *gray-code* für ganze Zahlen).

Evolutionäre Operatoren Den wichtigsten Operator stellt die Rekombination zweier Bitstrings (*cross over*) dar, der Mutationsoperator spielt nur eine untergeordnete Rolle. Begründet wird dies mit Hilfe des Schema-Theorems von HOLLAND [Hol75], in welchem die Hauptstärke von GAs in der Anreicherung von für die Problemlösung vorteilhaften *building blocks* gesehen wird. Diese Anreicherung wird durch Crossover begünstigt, durch Mutation jedoch tendenziell vermindert. Der Mutationsoperator wird hauptsächlich zum Erhalt der genetischen Vielfalt benötigt, wobei die hierfür notwendige Mutationsrate auch von weiteren Faktoren wie der Populationsgröße abhängt. Daher spricht sich MITCHELL [Mit96] für eine automatische Anpassung der Crossover- und Mutationsraten aus, wie sie typischerweise eher bei ES zu finden ist.

Selektion der Nachkommen Die Selektion der erzeugten Nachkommen kann durch Fitness-proportionale Selektionsverfahren (z. B. *roulette wheel*) erfolgen, wobei eine vorzeitige Konvergenz (*premature convergence*) auftreten kann, wenn anfangs wenige sehr gute Individuen die Population dominieren. Dieser Effekt kann durch eine zur Rangordnung proportionale Selektion (*rank selection*) vermindert werden. Ein ähnlicher Effekt wird durch die *tournament selection* ereicht, bei der wiederholt zwei zufällig ausgewählte Individuen gegeneinander zum Fitnessvergleich antreten. Das Prinzip des Elitismus, bei dem stets nur die besten Individuen überleben, wird aus Konvergenzgründen seltener eingesetzt, wobei häufig jedoch die bislang besten Individuen an separater Stelle vermerkt werden. Mit der *steady-state selection* wird vom Prinzip der Generationenfolge Abstand genommen und der *generation gap* [De 75] minimiert, indem jeweils nur wenige der schlechtesten Individuen einer Population ausgetauscht werden.

2.2.1.2 Bemerkungen

GAs können systematisch durch Problemstellungen in die Irre geführt werden, bei denen sich die Gesamtlösung nicht durch Anreicherung von Teillösungen erreichen lässt.

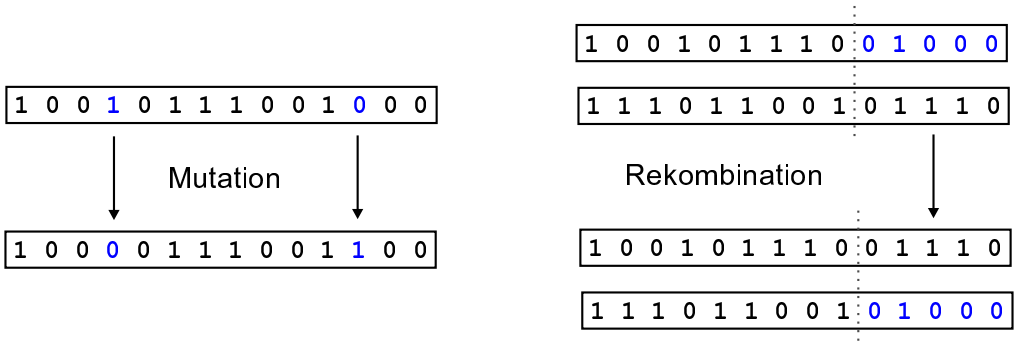

Abbildung 2.9: Links ist ein Mutationsoperator gezeigt, bei dem zwei zufällig gewählte Positionen im Bitstring verändert werden. Bei der Rekombination (rechts) wird ein Teil des Bitstrings zwischen zwei Individuen ausgetauscht.

Solche *deceptive functions* weisen meist eine sprunghafte, nicht-stetige Fitnesslandschaft auf und sind auch für andere Verfahren entsprechend schwierig zu optimieren.

2.2.2 Evolutionsstrategien

Die Entwicklung der Evolutionsstrategien (ES) begann in den sechziger Jahren an der Technischen Universität Berlin. Sie wurden von SCHWEFEL [Sch65] und RECHENBERG [Rec65, Rec73] zur Lösung von ingenieurstechnischen Fragestellungen eingesetzt. Vertieft wurde das Gebiet neben den weiterführenden Arbeiten der beiden Entwickler [Sch95, Rec94] wesentlich von BÄCK [Bäc96].

2.2.2.1 Beschreibung des Verfahrens

Repräsentation der Individuen Im Unterschied zu den GA werden die Individuen bei ES nicht mit binären, sondern mit reellwertigen Zahlen kodiert. Die Kodierung kann damit für viele Problemstellungen bereits auf der Phänotyp-Ebene erfolgen, eine evtl. mit einem Verlust behaftete Umwandlung in eine andere Darstellung ist oft nicht erforderlich.

Evolutionäre Operatoren Bei ES steht die Mutation im Vordergrund. Die Mutationsschrittweiten können über sogenannte Strategievariablen im Laufe der simulierten Evolution angepasst werden. Diese Strategievariablen werden einfach zu den der Evolution unterworfenen Variablen hinzugefügt und passen sich somit „automatisch“ den jeweiligen Bedürfnissen an.[11]

[11] Ein Beispiel für eine solche Anpassung ist der selbstgeregelte Übergang von einer anfangs großen zu einer im späteren Verlauf kleiner werdenden Mutationsschrittweite, d. h. der Wechsel von einer Exploration des Suchraums zu einer lokalen Optimierung.

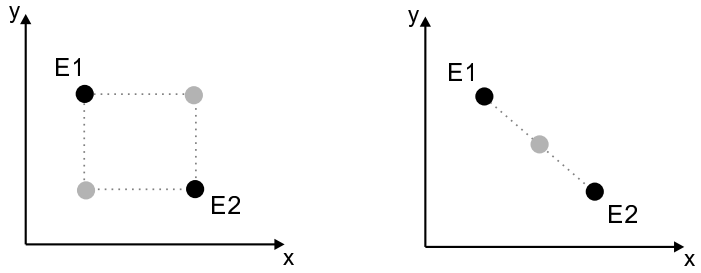

Abbildung 2.10: Crossover-Operatoren bei ES in einem zweidimensionalen Raum. Die Elternindividuen sind dunkel, mögliche Kindindividuen hellgrau gezeichnet. Links ist eine diskrete, rechts eine intermediäre Rekombination dargestellt.

Die Schrittweiten von Mutationen können global oder separat sein, die zugehörigen Mutationsverteilungen sind jeweils kreisförmig bzw. ellipsoid. Zudem werden unkorrelierte und korrelierte Schrittweiten unterschieden, die entsprechenden Ellipsen der Mutationsverteilungen sind achsenparallel oder frei gedreht (siehe Abb. 2.11).

Operatoren für eine Rekombination waren anfänglich nicht vorgesehen. Sie wurden erst nachträglich hinzugefügt und spielen nach wie vor eine untergeordnete Rolle. Bei der diskreten Rekombination wird pro Dimension jeweils einer der in den Elternindividuen vorhandenen Werte zufällig ausgewählt. Das Kindindividuum besteht also ausschließlich aus bereits in den Eltern vorkommenden Zahlenwerten.[12] Bei der intermediären Rekombination wird pro Dimension aus den in den Elternindividuen vorhandenen Werten ein (evtl. gewichteter) Mittelwert gebildet (vgl. Abb. 2.10).

Selektion der Nachkommen und Generationswechsel Für den Übergang von einer Eltern- zur nächsten Kindgeneration stehen zwei Vorgehensweisen zur Wahl. Bei der sogenannten „Komma“-Strategie erfolgt die Selektion nur auf der Menge der erzeugten Kinder, sämtliche Individuen der Elterngeneration werden verworfen. Bei der „Plus“-Strategie nehmen sowohl die Eltern als auch die Nachkommen an der Selektion teil. Während bei einer Komma-Strategie sich die durchschnittliche Fitness einer Population auch verschlechtern kann, ist dies bei einer Plus-Strategie nicht möglich. Bei letzterer besteht allerdings die Gefahr einer vorzeitigen Konvergenz, da anfangs überlegene Individuen, die eventuell nur ein lokales Optimum repräsentieren, die anderen schnell verdrängen und so die Diversität reduzieren können. Eine grobe Charakterisierung des Ablaufs einer Evolutionsstrategie kann mit der von RECHENBERG [Rec94] eingeführten Notation $(\mu, \lambda)^\gamma$ bzw. $(\mu + \lambda)^\gamma$ erfolgen, wobei μ die Anzahl der Eltern, λ die

[12] Diese Vorgehensweise ähnelt einem multiplen Crossover bei GAs mit $n-1$ möglichen Kreuzungspunkten bei einem Bitstring der Länge n.

Anzahl der Kinder und γ die Anzahl der Generationswechsel angibt.

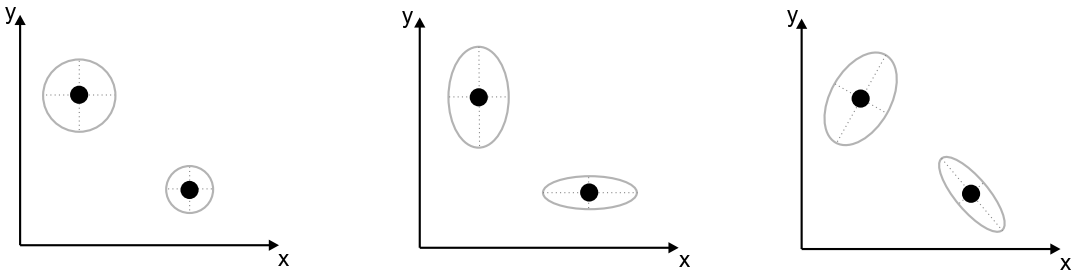

Abbildung 2.11: Mutations-Operatoren bei ES in einem zweidimensionalen Raum. Die Elternindividuen sind dunkel, die Mutationsradien hellgrau gezeichnet. Neben globalen Schrittweiten (links) werden auch separate Schrittweiten eingesetzt, die unkorreliert (mitte) oder korrelliert (rechts) sein können.

2.2.2.2 Eigenschaften

Der Erfolg eines ES-Laufs wird wesentlich von einer erfolgreichen Adaption der Strategieparameter (vor allem der Mutationsraten) während des Evolutionsprozesses beeinflusst. Möglich wäre z. B. eine schrittweise Verkleinerung der Parameter während des Laufs (ähnlich wie beim *simulated annealing*). Von Rechenberg wurde die $1/5$-Erfolgsregel vorgeschlagen [Rec94]. Sie besagt, dass die Mutationsschrittweite optimal gewählt ist, wenn durchschnittlich 20% aller Mutationen erfolgreich sind. Diese Erfolgsrate wird in regelmässigen Abständen geprüft. Wird sie unterschritten, so wird die Mutationsrate σ mit $\sigma' = 1/\alpha \cdot \sigma$ verkleinert, sonst mit $\sigma' = \alpha \cdot \sigma$ erhöht, wobei $\alpha > 1$ gelten muss. Häufig wird $\alpha = 1.3$ oder $1/\alpha = 0.85$ (bzw. $\alpha \approx 1.18$) gesetzt [Rec94, Sch95].

Ein anderer Ansatz besteht darin, den evolvierten Vektor (das „Chromosom“) um die Mutationsschrittweiten-Parameter zu erweitern. Sie werden dadurch im Laufe des Evolutionsprozesses angepasst und können indirekt einen Selektionsvorteil für ein Individuum darstellen, da es bei gut eingestellten Parametern im Durchschnitt bessere Nachkommen erzeugen wird als Individuen mit unpassenden Parameterwerten. Es kann eine globale oder pro Dimension eine seperate Mutationsrate verwendet werden. Eine weitere Verbesserung stellt die Korrelation dieser separaten Schrittweiten dar. Die hierfür benötigte Kovarianzmatrix kann entweder indirekt über Rotationswinkel erzeugt [Sch81] oder direkt mit der *Covariance Matrix Adaption* (CMA) von HANSEN & OSTERMEIER [HO96] angepasst werden. Auf diese Verfahren wird hier nicht näher eingegangen, es soll nur erwähnt werden, dass CMA nicht zusammen mit den Standard-Rekombinationsoperatoren eingesetzt werden kann.

2.2.3 Genetisches Programmieren

Das Gebiet des Genetischen Programmierens (GP) entstand Anfang der neunziger Jahre mit dem Ziel, Computerprogramme mit den Mitteln der Evolution zu erzeugen.[13] Die Programme werden dabei in einem Syntax-Baum (*parse tree*) dargestellt. Die einzelnen Knoten dieses Baumes bestehen aus Funktionen, die Blätter aus Variablen oder konstanten Werten. In den ersten Arbeiten auf diesem Gebiet von KOZA [Koz92, Koz94] wird LISP als Programmiersprache verwendet. Seitdem wurde das Prinzip der Syntax-Bäume auch auf andere Anwendungen erweitert, beispielsweise auf das Problem der symbolischen Regression. Je nach Art der Anwendung enthalten die Syntax-Bäume dann arithmetische, prädikatenlogische oder andere (problemspezifische) Funktionen.

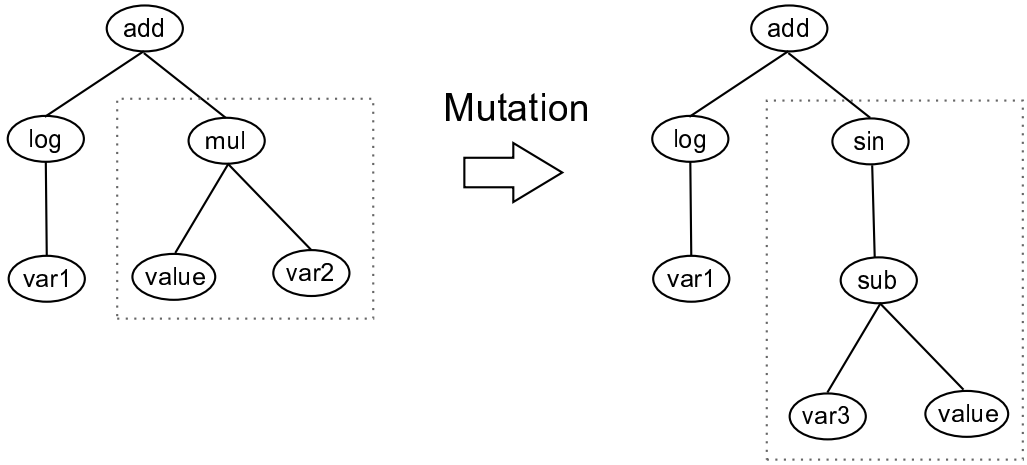

Abbildung 2.12: Funktionsweise des Mutationsoperators beim Genetischen Programmieren. Ein Teil des ursprünglichen Baums (links) wird durch einen zufällig generierten Teilbaum ersetzt (rechts).

2.2.3.1 Beschreibung des Verfahrens

Repräsentation der Individuen Die Individuen werden hierarchisch in Form einer Baumstruktur kodiert. Die Knoten des Baumes enthalten Funktionen, welche auf die in den Blättern enthaltenen Terminale (Variablen oder Konstanten) angewandt werden. Die erlaubten Funktions- und Terminalwerte werden in der Funktionsmenge F bzw. in der Terminalmenge T definiert. Für die Bäume in Abb. 2.12 und 2.13 gilt beispielsweise $F = \{\mathrm{add}, \mathrm{sub}, \mathrm{mul}, \mathrm{div}, \exp, \log, \sin, \cos\}$ und $T = \{\mathrm{value}, \mathrm{var1}, \ldots, \mathrm{var4}\}$.

[13] Dieses Ziel wurde von KOZA als '*programming of computers by means of natural selection*' und von BANZHAF als '*automatic evolution of computer programs*' formuliert [Koz92, BNKF98].

Die Baumstrukturen lassen sich auch in LISP-artigen symbolischen Ausdrücken (*s-expressions*) darstellen, der linke Baum in Abb. 2.12 würde beispielsweise als `(add (log var1) (mul value var2))` geschrieben werden.

Initialisierung der Individuen Es gibt zwei häufig eingesetzte Verfahren, um Bäume mit einer maximalen Tiefe $D_{\max}$ zu erzeugen. Bei der *full method* hat jeder generierte Baum die maximale Tiefe. Für einen Knoten der Tiefe $d < D_{\max}$ wird eine Funktion aus der Menge F gewählt. Hat der Knoten die Tiefe $d = D_{\max}$, so wird ein Terminal aus der Menge T eingefügt. Bei der *grow method* wird dagegen für einen Knoten der Tiefe $d < D_{\max}$ zufällig eine Funktion oder ein Terminal aus der Menge $\{F \cup T\}$ ausgewählt. Die Tiefe der generierten Bäume ist bei dieser Methode variabel. Eine Kombination der beiden Verfahren wird als *ramped half-and-half* bezeichnet.

Evolutionäre Operatoren Der wichtigste Operator ist das Crossover. Beim Standard-Crossover werden zwei Nachkommen aus je zwei Elternteilen durch den Austausch von zufällig ausgewählten Teilbäumen erzeugt (siehe Abb. 2.13). Der Mutationsoperator wählt zufällig einen Knoten aus der Baumstruktur des Elternindividuums aus. In der Baumstruktur des Nachkommens wird an dieser Stelle ein neuer, ebenfalls zufällig generierter Teilbaum erzeugt (vgl. Abb. 2.12). Bei beiden Operatoren wird darauf geachtet, dass die vorgegebene maximale Baumtiefe $D_{\max}$ nicht überschritten wird.[14]

Selektion der Elternindividuen Es wird ein fitness-proportionales Selektionsschema eingesetzt (z. B. *roulette wheel selection*). Bei sehr großen Populationen ($\mu > 1000$) wird gelegentlich auch ein vereinfachtes Verfahren namens *over-selection* verwendet [ES03]. Dabei werden die Individuen nach ihrer Fitness sortiert und mittels eines Schwellwerts in zwei Gruppen aufgeteilt. Aus der deutlich kleineren Gruppe der besseren Individuen werden 80% der Eltern ausgewählt, aus der Gruppe der schlechteren Individuen dagegen nur 20%.

Generationswechsel In der Regel wird ein generationsbasierter Ansatz verwendet, bei dem die Menge der erzeugten Nachkommen ohne weitere Selektion die nächste Generation bildet. Alle Individuen sterben nach einer Generation, es wird kein Elitismus eingesetzt [Koz92]. Alternativ kann ein *steady state*-Schema zum Einsatz kommen, bei dem auf einen Generationswechsel verzichtet wird. Dabei wird jeweils ein zufällig ausgewählter Operator auf ein bis zwei Elternindividuen angewandt. Weisen die Nachkommen einen höheren Fitnesswert auf, so ersetzen sie die Eltern [BNKF98]. Diese elitistische Vorgehensweise mildert die tendenziell oft eher destruktiven Auswirkungen der oben beschriebenen Standard-Operatoren.[15]

[14] Beim Crossover kann beispielsweise die zufällige Auswahl der Teilbäume aus den beiden Elternindividuen solange wiederholt werden, bis beide Nachkommen eine zulässige Tiefe $d < D_{\max}$ haben.

[15] Eine andere Möglichkeit besteht in der Definition von problemspezifischen Operatoren, welche

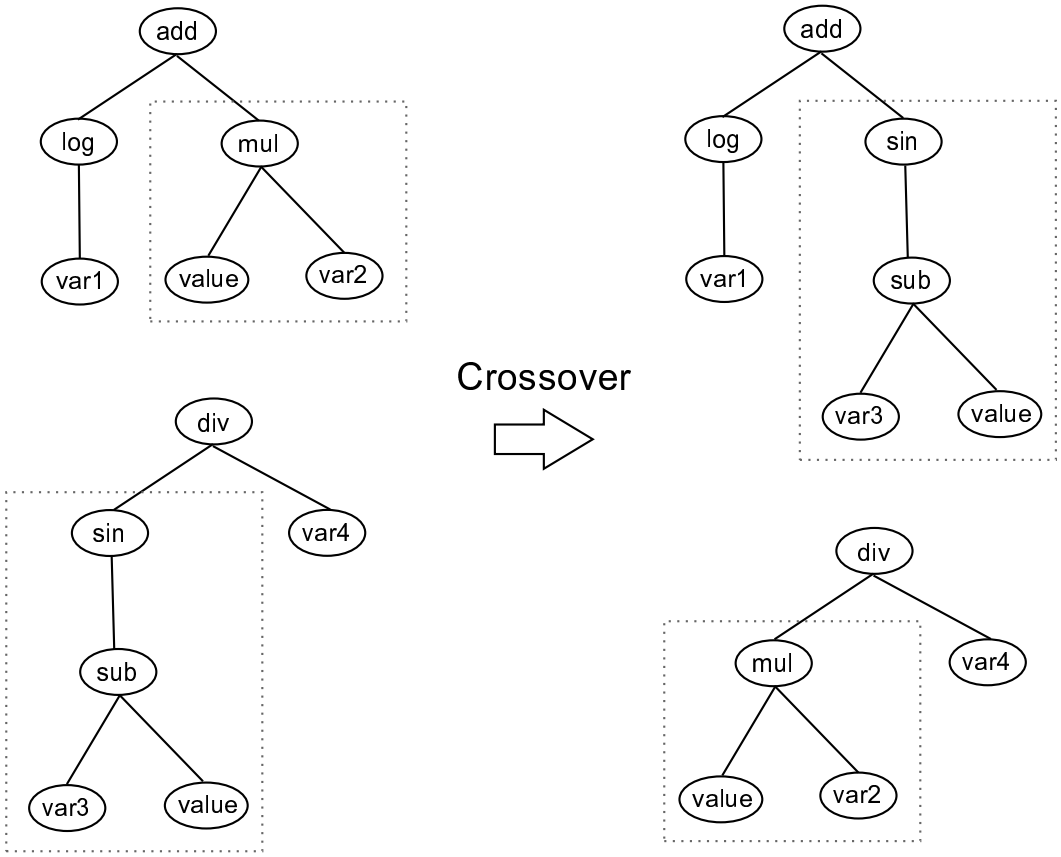

Abbildung 2.13: Funktionsweise des Crossover-Operators beim Genetischen Programmieren. Die markierten Teile der beiden ursprünglichen Bäume (links) werden untereinander ausgetauscht (rechts).

2.2.3.2 Eigenschaften

Ein bei GP häufig auftretendes Problem ist das übermässige Wachstum der generierten Bäume. Die Ursachen dieses auch als *bloat* bezeichneten Phänomens wurden u. a. von SOULE [SFD96] untersucht.[16] Es ist beispielsweise daran zu erkennen, dass einzelne Knoten im Baum nur konstante Werte liefern, obwohl sie einen sehr komplexen Teilbaum repräsentieren (ein solcher Teilbaum wird als „neutral“ oder „nicht-effektiv“ bezeichnet).[17] Zwei Hypothesen bezüglich der Entstehung des übermässigen Baumwachstums wurden von BANZHAF & LANGDON [BL02] betrachtet und in Simulationsläufen bestätigt: Die erste Hypothese (*fitness causes bloat*) sagt voraus, dass bei

auch kleinere Änderungen an den Individuen vornehmen können und damit z. B. für eine lokale Suche geeignet sind.

[16] Eine alternative, leicht ironische Bezeichnung für dieses Phänomen lautet ‘*survival of the fattest*’.

[17] Gelegentlich wird auch der Begriff „Introns“ für neutrale Teilbäume verwendet. Diese Analogiebildung zu biologischen Gen-Sequenzen ist im Bereich des GP jedoch selten wirklich treffend.

Individuen mit hoher Fitness der Anteil an neutralen Teilbäumen tendenziell zunimmt, wenn durch Mutation und Crossover überwiegend keine besseren Lösungen, sondern nur Variationen mit gleicher Funktionsweise gefunden werden. Dies liegt daran, dass es insgesamt viel mehr Variationen mit großer als mit kleiner Baumtiefe gibt. Die zweite Hypothese (*neutral code is protective*) besagt zudem, dass neutrale Teilbäume eine Schutzfunktion haben, da Mutationen oder Crossover in diesem Bereich keine Auswirkung auf die Fitness des Baumes haben. Individuen mit hoher Fitness bleiben somit mit höherer Wahrscheinlichkeit erhalten, je größer der Anteil der neutralen Teilbäume ist.

Eine Möglichkeit zur Vermeidung dieses Effekts ist eine Beschränkung der maximalen Baumtiefe. Bei einer zu klein gewählten maximalen Baumtiefe werden jedoch auch sinnvolle komplexe Lösungen generell ausgeschlossen. Besser geeignet sind daher Belohnungsterme in der Fitnessfunktion, welche die Tiefe des Baumes oder die Anzahl der Knoten in die Bewertung mit einbeziehen und auf diese Weise kleinere Lösungen bevorzugen [SFD96]. Diese Vorgehensweise wird als *parsimony pressure* bezeichnet [ES03].

2.2.3.3 Streng-typisiertes Genetisches Programmieren

Beim streng-typisierten Genetischen Programmieren (STGP) wird jedem Knoten eines Baumes ein Typ zugewiesen. Mit Hilfe eines Satzes von Regeln wird bestimmt, welche Arten von Knoten als Elternknoten und welche als Kindknoten eines bestimmten Typs einsetzbar sind. Entwickelt wurde dieses Verfahren von MONTANA [Mon93], um Funktionen mit Übergabeparametern im Rahmen eines evolvierten Computerprogramms definieren zu können. Generell lassen sich mit Hilfe von STGP komplexere Probleme auf Bäume abbilden, als dies mit Standard-GPs möglich wäre [Mon95].

2.2.4 Weitere Verfahren

Der Vollständigkeit halber soll hier noch ein weiteres, nicht auf evolutionären Prinzipien beruhendes Verfahren aufgeführt werden.

2.2.4.1 Simulated Annealing

Bei dieser von KIRKPATRICK *et al.* [KGV83] entwickelten Methode wird der Prozess des langsamen Abkühlens simuliert, welcher z. B. bei Metallen zu einer energetisch günstigen, regelmässigen Kristallstruktur führen kann. Das Verfahren basiert auf der Beobachtung, dass abhängig von der Umgebungstemperatur T_U in einem flüssigen Medium vorliegende Atome ihr Energieniveau nicht nur verkleinern, sondern unter bestimmten Umständen auch wieder erhöhen bzw. „verschlechtern“ können. Für die Wahrscheinlichkeit p dieser Verschlechterung gilt:

$$p = \exp\left(\frac{-\Delta E}{k_B \cdot T_U}\right) \tag{2.15}$$

Dabei steht ΔE für die Energiedifferenz zwischen dem alten und dem potentiellen neuen Energiezustand, k_B ist die Boltzmann-Konstante.

Um dieses Prinzip auf Optimierungsaufgaben übertragen zu können, werden k_B und T_U zu einer neuen Variablen T zusammengefasst, welche keinen direkten Bezug zu physikalischen Größen mehr besitzt. Der Wert von T wird im Laufe des Optimierungsprozesses gemäß eines „Kühlschemas“ verringert. Dabei muss T im positiven Wertebereich bleiben, für jedes Kühlschema gilt also $T \rightarrow 0_+$. Bei einer arithmetischen Verringerung wird in jedem Schritt ein konstanter Wert von T abgezogen, bei einer geometrischen Verringerung wird T pro Schritt mit einem Wert $\alpha < 1$ multipliziert. Der Vorteil des Verfahrens liegt darin, dass in der „heißen“ Anfangsphase eines Optimierungslaufes mit einer gewissen Wahrscheinlichkeit lokale Optima wieder verlassen werden können, auch wenn dafür eine zeitweilig schlechtere Bewertung in Kauf genommen werden muss. Gemäß der Formel 2.15 ist diese Wahrscheinlichkeit für ein flaches lokales Optimum höher als für ein steiles, tiefes lokales Optimum. In stark zerklüfteten Fitnesslandschaften kann die Erfolgswahrscheinlichkeit durch eine Verlangsamung des Abkühlprozesses erhöht werden [GKK04].

2.3 Validierung

Die Güte eines Modells lässt sich nicht direkt anhand des auf den Trainingsdaten erreichten Modellfehlers quantifizieren. Dieses als *resubstitution error* bezeichnete Fehlermaß liefert meist eine zu optimistische Abschätzung des tatsächlichen Fehlerwertes, besonders bei „übertrainierten“ Modellen. Modelle werden als übertrainiert (*overtrained*) bezeichnet, wenn der bei realen Datensätzen stets vorhandene Messfehler approximiert wird. Die Vorhersagekraft auf unabhängigen Testdaten ist in einem solchen Fall entsprechend gering. Mit Hilfe von Validierungsverfahren wird daher versucht, die Generalisierungsfähigkeit eines Modells zu ermitteln. Es lassen sich dabei „externe“ von „internen“ Validierungen unterscheiden.

Die nachfolgend beschriebenen Validierungsmethoden sind bereits seit den siebziger Jahren bekannt. Trotzdem werden deren Aussagekraft und Verläßlichkeit in Bezug auf spezielle Einsatzgebiete wie QSAR-Analysen auch in der aktuellen Literatur noch diskutiert. So spricht sich HAWKINS [HBM03] für den alleinigen Einsatz von Kreuzvalidierung bei kleineren Datensätzen aus. SUTHERLAND [SOW04] beobachtete deutliche Abweichungen zwischen Kreuzvalidierungs- und Testfehler beim Einsatz von kleinen und besonders diversen Testdatensätzen (weniger als 30 Patterns). Generell lässt sich sagen, dass mit einer Kreuzvalidierung stets nur die Generalisierungsleistung innerhalb des trainierten Bereichs bestimmt werden kann. Mit einem geeignet gewählten Testdatensatz lässt sich dagegen die Generalisierungsfähigkeit auch über den trainierten Bereich hinaus abschätzen.

2.3.1 Externe Validierung

Bei der sogenannten externen Validierung wird eine unabhängige Menge von Testdaten zur Bestimmung des Modellfehlers herangezogen. Liegen ausreichend viele Datenpunkte (*patterns*, *samples*) vor, so kann beispielsweise ein Drittel der vorhandenen Daten während der Trainingsphase „zurückgehalten“ werden (*holdout dataset*). Diese Methode ist allgemein als aussagekräftig und verlässlich anerkannt. In der Praxis treten allerdings häufig Probleme durch zu kleine Datensätze auf. Liegen insgesamt nur wenige Patterns vor, so gilt es abzuwägen, ob von diesen nicht möglichst viele in den Trainingsdatensatz aufgenommen werden sollen, um das Modell zu verbessern. Die verbleibende geringe Anzahl von Testpatterns kann zu einer ungenauen Vorhersage des Testfehlers führen. Bei QSAR-Analysen werden Testsets mit weniger als 50 Patterns als fragwürdig angesehen, weniger als 20 Patterns gelten als nicht aussagekräftig [Haw04, HBM03, SOW04].

2.3.2 Interne Validierung

Das am häufigsten angewandte Verfahren zur internen Validierung ist die sogenannte Kreuzvalidierung (*cross validation*, CV). Hierbei wird das Trainingsverfahren mehrmals durchlaufen, wobei jeweils ein bestimmter Anteil der Patterns ausgelassen und zum Testen verwendet wird. Insgesamt wird jedes Pattern bei genau einem der Durchläufe getestet. Bei dem *leave one out*-Verfahren (LOO) wird bei jedem Durchlauf nur ein einzelnes Pattern ausgelassen und getestet. Dieses Verfahren kommt nur bei kleinen Datensätzen zum Einsatz, da der Rechenaufwand linear mit der Anzahl der Patterns ansteigt. Werden bei größeren Datensätzen mehrere Patterns pro Durchlauf ausgelassen, so wird dies als n-fache CV bezeichnet. Dabei gibt n die Anzahl der Durchläufe und damit auch die Anzahl der Testsets (*folds*) an. Eine weitere Verbesserung der Vorhersage des CV-Fehlers ist durch eine Mittelung über mehrere komplette CV-Durchläufe möglich, wobei die einzelnen *folds* jeweils unterschiedlich zusammengestellt werden. In der Praxis bewährt hat sich eine zehnfache Kreuzvalidierung.

Bei dem sogenannten *bootstrap*-Verfahren werden ebenfalls in mehreren wiederholten Durchläufen Trainings- und Testdatensätze zusammengestellt, wobei die zufällige Auswahl der Trainingspattern jedoch „mit Zurücklegen“ erfolgt. Ein Pattern kann somit mehrfach in der Trainingsmenge enthalten sein. Werden k Trainingspatterns aus m Mustern ausgewählt, so verbleibt ein Pattern mit der Wahrscheinlichkeit von $P_{\text{test}} = (1 - 1/m)^k$ in der Testmenge. Häufig wird $k = m$ gesetzt, die Testmenge enthält dann im Durchschnitt 36.8% der Patterns (da $P_{\text{test}} = (1-1/m)^m \approx e^{-1} \approx 0.368$). Der Gesamtfehler ergibt sich durch eine mit P_{test} gewichtete Kombination der auf der Testmenge und der Trainingsmenge erzielten Fehlerwerte [WF00].

2.4 Wirkstoffdesign

In diesem Kapitel werden die für das Verständnis der späteren Ausführungen notwendigen Grundlagen des Wirkstoffdesigns vorgestellt. Hierbei kann es sich nur um eine stark vereinfachende und verkürzte Darstellung handeln, für eine umfassende Einführung sei auf das Standardwerk von BÖHM, KLEBE & KUBINYI [BKK96] verwiesen. Nach einer Beschreibung der generellen Vorgehensweise im Abschnitt 2.4.1 wird der Schwerpunkt auf die Suche und Optimierung von Leitstrukturen im Bereich des Liganden-basierten Wirkstoffdesigns gelegt.

2.4.1 Die Entwicklungsstadien eines Arzneistoffes

In diesem Abschnitt werden die heutzutage bei der Entwicklung eines neuen Arzneistoffes durchlaufenen Stadien (der sogenannte *drug-design-cycle*, vgl. Abb. 2.14) beschrieben. Dieser hat sich erst in den letzten Dekaden in der dargestellten Form entwickelt und ist sicher auch zukünftig noch Änderungen und Erweiterungen unterworfen.

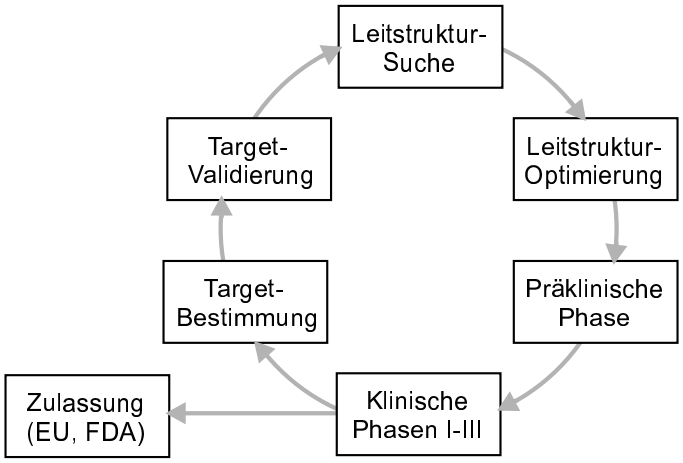

Abbildung 2.14: Die Phasen der Entwicklung eines Arzneistoffes (*drug-design-cycle*) von der Target-Bestimmung (*target identification*) bis zu einer eventuellen Zulassung.

Ganz am Anfang steht die Bestimmung einer körpereigenen Zielstruktur, welche in Verbindung mit dem Auftreten eines bestimmten Krankheitsbildes beim Menschen (auch als „Indikation“ bezeichnet) gebracht werden kann. Dieses sogenannte *target* ist im Idealfall ursächlich an der Entstehung der Krankheit beteiligt. Wenn ein solcher einzelner Faktor jedoch nicht bekannt oder wie bei manchen komplexen Krankheitsbildern schlichtweg nicht bestimmbar ist, so kann beispielsweise ein einzelnes Glied einer längeren Signalkette als Target in Angriff genommen werden. Auf die Bestimmung und Validierung von Targets soll hier nicht weiter eingegangen werden, da sie

das Thema der strukturellen und funktionalen Proteomics und zu einem Teil auch der Bioinformatik sind. Es sei nur erwähnt, dass von den über 30 000 im menschlichen Genom vorhandenen potentiellen Targets rund 3 000 als mögliche Angriffspunkte (*druggable targets*) in Frage kommen [GE03].

In den meisten Fällen handelt es sich bei den Targets um Proteine oder Enzyme. Diese werden auch als „Rezeptoren“ bezeichnet, da ihre jeweilige Funktion durch Anlagerung eines in der Regel kleineren Moleküls, dem natürlichen „Liganden“, ausgelöst wird. Dieser Rezeptor soll nun in seiner Funktion beeinflusst werden, was beispielsweise durch kompetetive Inhibition geschehen kann. Hierbei wird nach einem neuen, bisher nicht im Körper auftretenden Liganden gesucht, der die Bindungstasche des Rezeptors zwar belegt, nicht aber die vorgesehene Wirkung auslöst. Dieser „Inhibitor“ unterdrückt so die Wirkung des Rezeptors. Auch andere Wirkmechanismen sind bekannt, z. B. kann der Ligand auch als Agonist oder als Antagonist dienen.[18]

Bei solchen neu entwickelten Liganden werden mehrere, aufeinander aufbauende Entwicklungstadien unterschieden. Am Anfang der Suche stehen meist vergleichsweise schwach an den Rezeptor bindende Moleküle, welche zudem noch unerwünschte Eigenschaften (wie z. B. schlechte Löslichkeit oder Toxizität) aufweisen können. Diese werden als „Leitstrukturen“ bezeichnet. Gefunden werden sie beispielsweise mittels eines *High Throughput Screenings* (HTS), wobei die resultierende Liste der Treffer (*hits*) noch ausgewertet und konsolidiert werden muss, z. B. mit Hilfe von Clustering-Verfahren. Alternativ können virtuelle Screening-Methoden *in silico* unter Einsatz von sogenannten Docking-Verfahren durchgeführt werden, welche allerdings noch mit hohen Unsicherheiten und entsprechend geringer Vorhersagegenauigkeit verbunden sind.

Sobald eine oder möglichst mehrere strukturell diverse Leitstrukturen vorliegen, können diese in einem weiteren Schritt optimiert werden. Dabei werden einzelne Substrukturen entfernt oder gegen ähnliche Substrukuren ausgetauscht, um die gewünschte Wirkung zu erhöhen und Nebenwirkungen zu minimieren. Die neuen Strukturvorschläge müssen in der Regel mühsam mit Hilfe der präparativen organischen Chemie erstellt werden. Daher sollten neben der unersetzbaren Erfahrung und Intuition eines Chemikers unterstützend Docking-Verfahren zur Überprüfung von strukturellen Hypothesen zum Einsatz kommen.

Die in Hinsicht auf ihre Eigenschaften optimierten Strukturen werden nun an die extrem zeit- und kostenaufwändigen klinischen Phasen übergeben. Der genaue Ablauf der insgesamt drei klinischen Phasen soll hier nicht weiter ausgeführt werden. Erwähnt sei nur, dass in diesen Abschnitten der Arzneistoffentwicklung zum einen die größten Kosten entstehen und zum anderen die Wahrscheinlichkeit des Scheiterns immer noch bei 90% liegt. Hieraus ergibt sich die Wichtigkeit der sorgfältigen Durchführung der vorhergehenden Stufen der Leitstruktursuche und -optimierung.

Sämtliche Entwicklungsstufen bis hin zu einem marktreifen Arzneimittel dauern zusammengenommen derzeit 12 bis 15 Jahre und verursachen Kosten in Höhe von rund

[18] Eine Definition dieser und weiterer nicht explizit im Text erläuterter Begriffe findet sich im Glossar.

800 Millionen Euro, bei stetig ansteigender Tendenz [GE03, Kap. 10]. Aus 100 frühen Leitstruktur-Ideen entsteht im statistischen Durchschnitt höchstens ein einziges Arzneimittel, die Erfolgswahrscheinlichkeit liegt also bei weniger als einem Prozent.

2.4.2 Liganden-basiertes Wirkstoffdesign

Im Gegensatz zum hier nicht näher beschriebenen Rezeptor-basierten Wirkstoffdesign (meist als *molecular modelling* bezeichnet), wird bei dem Liganden-basierten Wirkstoffdesign der Rezeptor nicht als bekannt vorausgesetzt. Es liegen lediglich Informationen über die Bindungsstärke von einer Menge von Substanzen an den Rezeptor vor. Die Bindungsstärke kann dabei qualitativ oder (semi-)quantitativ vorliegen. Die Bezeichnung semi-quantitativ bezieht sich hierbei auf mit einem großen Messfehler behaftete Werte, wie sie z. B. bei HTS-Läufen entstehen.

Ein Ziel des Liganden-basierten Wirkstoffdesigns besteht darin, aus der Menge der bekannten Liganden einen sogenannten „Pharmakophor“ abzuleiten (vgl. Abb. 2.15). Dieser stellt eine komprimierte Darstellung der Molekülstruktur dar, bei dem nur noch die an Wechselwirkungen mit dem Rezeptor beteiligten Gruppen dargestellt werden. Unterschieden wird dabei meistens zwischen wasserstoffbrückenbildenden Gruppen (Akzeptor und Donor), hydrophoben Gruppen und positiv oder negativ geladenen Gruppen.

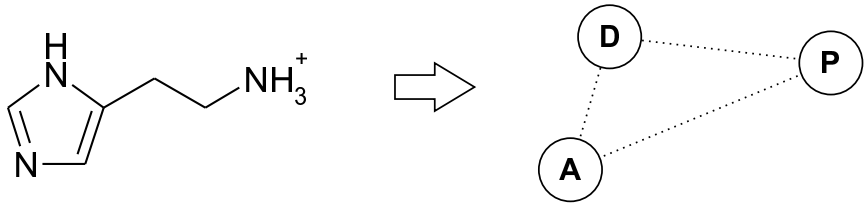

Abbildung 2.15: Der Wirkstoff Histamin (links, bei $pH = 7$) und dessen aus je einem Akzeptor (A), Donor (D) und positiv geladener Gruppe (P) bestehenden Pharmakophor (nach [BKK96, S. 159]).

2.4.3 Eigenschaften von Arzneistoffen

Die Haupteigenschaft eines Liganden besteht darin, ausreichend stark und selektiv an einen als Zielstruktur ausgewählten Rezeptor zu binden. Ein solcher Ligand zeigt eine gewünschte Wirkung und kann daher als „Wirkstoff“ bezeichnet werden. Um die klinischen Phasen erfolgreich durchlaufen zu können und damit zum „Arzneistoff“ zu werden, muss ein Wirkstoff weitere, in diesem Abschnitt aufgeführte Eigenschaften aufweisen. Diese werden gerne unter dem Kürzel ADMET zusammengefasst (*absorption, distribution, metabolism, excretion, toxicity*) und beschreiben den Weg, den eine

Substanz von der möglichst oralen Verabreichung bis an den Wirkort und weiter bis zur Ausscheidung über Leber oder Niere im Körper zurücklegt. Auch die Geschwindigkeit (Kinetik) der Abläufe spielt hierbei eine wichtige Rolle.

Da eine Messung dieser Parameter im Tierversuch zeitaufwändig und teuer ist, besteht ein großes Interesse an der frühzeitigen Vorhersage der ADMET-Werte. Eine genaue und stets zuverlässige Vorhersage ist aufgrund der hochkomplexen und teilweise noch unverstandenen Vorgänge im Körper nicht möglich, eine grobe Abschätzung („Wert liegt im grünen Bereich“) ist jedoch machbar und in den frühen Entwicklungsstadien eines Arzneistoffes völlig ausreichend. Im folgenden werden einige Ansätze beschrieben und miteinander verglichen.

2.4.3.1 Orale Verfügbarkeit

Von den meisten Patienten wird die orale Verabreichung eines Arzneistoffes (beispielsweise in Form einer Tablette) anderen Applikationsarten vorgezogen, so dass sich für oral verfügbare Medikamente größere Marktchancen ergeben. Entsprechend wird bereits bei der Entwicklung von neuen Liganden auf diese Eigenschaft geachtet.

Eine Abschätzung der oralen Verfügbarkeit einer Substanz ist mit der vielbeachteten *rule of 5* von LIPINSKY *et al.* [LLDF97] möglich. Diese Regeln wurden durch die Untersuchung von ca. 2 300 Wirkstoffen des World Drug Index (WDI) abgeleitet, welche zum Zeitpunkt der Auswertung mindestens die klinische Phase 1 passiert hatten. Sie lauten:

1. Molekulargewicht ≤ 500.
2. Lipophilie-Wert $\log P \leq 5$.
3. Anzahl der Wasserstoffbrücken-Donoren ≤ 5.
4. Anzahl der Wasserstoffbrücken-Akzeptoren ≤ 10.

Die statistische Auswertung der betrachteten Moleküldatenbank ergab, dass 90% der bekannten Arzneistoffe keine oder nur eine der aufgelisteten Regeln verletzen.

Spätere Untersuchungen verwenden wiederum andere Kriterien, die teilweise ähnliche Zusammenhänge ausdrücken, zum Teil aber auch echte Ergänzungen darstellen. So hat eine von VEBER *et al.* [VJC$^+$02] durchgeführte Studie der oralen Verfügbarkeit von 1 100 Wirkstoffen bei Ratten ergeben, dass die folgenden zwei Kriterien ausreichend sind:

1. Anzahl der rotierbaren Bindungen ≤ 10.
2. Größe der polaren Oberfläche (*polar surface area*, PSA) ≤ 140 Å^2.

Als eine Alternative zur zweiten Regel wird eine Gesamtzahl von Wasserstoffbrücken-Donoren und -Akzeptoren ≤ 12 angegeben. Das Molekulargewicht spielt nach dieser Untersuchung keine Rolle. Als Erklärung wird darauf hingewiesen, dass durch die Beschränkung des Molekulargewichts indirekt zugleich auch die Anzahl der H-Donoren und H-Akzeptoren begrenzt wird, wohingegen eine Obergrenze für die Anzahl der Donoren und Akzeptoren nicht notwendigerweise beschränkend auf das Molekulargewicht wirkt.

Solche vergleichsweise einfachen, statistisch abgeleiteten Regelsätze können zwar nur grobe Abschätzungen liefern, haben aber dafür den Vorteil der unmittelbaren Verständlichkeit und Nachvollziehbarkeit. Sie stellen keine *black box* dar, im Gegensatz zu nichtlinearen Verfahren wie z. B. den Neuronalen Netzen. Sie werden häufig in Kombination miteinander eingesetzt und bei Bedarf auch leicht modifiziert, falls einzelne Kriterien nicht unmittelbar verfügbar sind [BBR+04].

2.4.4 Ligand-Rezeptor-Wechselwirkung

Die Stärke der Bindung eines Liganden an einen Rezeptor (meist handelt es sich dabei um ein Protein) wird mit der Bindungskonstanten K_i beschrieben. K_i ist ein Maß für den Anteil der vorliegenden Liganden, welcher im thermodynamischen Gleichgewicht an die vorhandenen Rezeptoren gebunden ist.[19]

$$\mathrm{Ligand{-}Rezeptor} \rightleftharpoons \mathrm{Ligand} + \mathrm{Rezeptor} \tag{2.16}$$

$$K_i = \frac{k_d}{k_a} = \frac{[\mathrm{Ligand}]\,[\mathrm{Rezeptor}]}{[\mathrm{Ligand{-}Rezeptor}]} \tag{2.17}$$

Dabei steht k_a für die Geschwindigkeitsrate der Komplexbildung (Assoziation) und k_d für die Geschwindigkeitsrate des Zerfalls (Dissoziation). Bei K_i handelt es sich um einen Konzentrationswert mit der Einheit mol/l (M). Bei einer Ligandkonzentration von K_i ist die Hälfte der Rezeptoren durch Ligandenmoleküle belegt. Dies lässt sich direkt aus Formel 2.17 ableiten:

$$K_i = [\mathrm{Ligand}] \quad \Rightarrow \quad [\mathrm{Rezeptor}] = [\mathrm{Ligand{-}Rezeptor}] \tag{2.18}$$

Anstelle der Bindungskonstanten kann auch die freie Bindungsenthalpie ΔG angegeben werden. Im Gleichgewichtszustand gilt:

$$\Delta G = \Delta G^0 - R\,T\,\ln K_i = 0 \tag{2.19}$$

[19] Bei dieser eher im pharmazeutischen Bereich anzutreffenden Definition der Bindungskonstante wird K_i kleiner, je mehr Liganden im Gleichgewicht an die Rezeptoren binden. In der physikalischen Chemie wird die Bindungskonstante meist reziprok als $K' = k_a/k_d$ definiert [Moo90, Kap. 8.12].

$$\text{und somit:} \qquad \Delta G^0 = R\,T\,\ln K_i \tag{2.20}$$

Dabei steht R für die Gaskonstante und T für die absolute Temperatur in Kelvin.

Einfacher experimentell zugänglich ist dagegen der IC_{50}-Wert. Er gibt die Konzentration an Liganden an, bei der die Aktivität des Rezeptors auf die Hälfte sinkt. Dieser Wert verläuft in erster Annäherung parallel zum K_i-Wert und wird daher in der Praxis oft verwendet.

Im folgenden werden die häufigsten Arten der Wechselwirkung zwischen Ligand und Rezeptor vorgestellt. Auf seltener anzutreffende Wechselwirkungen (wie z. B. Metallkomplexierungen) wird an dieser Stelle nicht weiter eingegangen.

Wasserstoffbrücken Eine elektrostatische Wechselwirkung, die sich zwischen einer mit einem Proton versehenen Gruppe (z. B. $>$NH oder $-$OH, auch „Donor“ genannt) und einem Atom mit einer negativen Partialladung („Akzeptor“) ausbilden kann. Die Bildung einer Wasserstoffbrücke tritt nur innerhalb gewisser Abstands- und Winkelmaße auf.

Ionische Wechselwirkungen Eine geladene Gruppe des Liganden kann eine elektrostatische Anziehung bei einer entgegengesetzt geladenen Gruppe des Rezeptors bewirken. Diese auch als „Salzbrücken“ bezeichneten Wechselwirkungen treten nur innerhalb begrenzter Abstandsmaße auf.

Van-der-Waals Wechselwirkungen Durch diese zwischenmolekularen Kräfte kann eine schwache Wechselwirkung zwischen inerten Atomen und gesättigten Molekülen entstehen. Bei der Wechselwirkung zwischen Atomen treten nur die sehr schwachen Dispersionskräfte (London-Kräfte) auf. Bei Molekülen sind die Wechselwirkungen von induzierten oder permanent vorhandenen Dipolmomenten als zusätzliche Anziehungskräfte wirksam.

Hydrophobe Wechselwirkungen Diese schwachen und ungerichteten Wechselwirkungen bilden sich zwischen unpolaren Seitenketten des Rezeptors und lipophilen Gruppen des Liganden aus, sofern diese räumlich dicht beieinanderliegen. Sie können einen wesentlichen Anteil zur Gesamt-Bindungsstärke eines Liganden beitragen, welcher jedoch häufig überwiegend mit der Verdrängung von Wassermolekülen aus der Bindetasche zu erklären ist (Entropieeffekt).

2.4.5 Moleküldeskriptoren

Die rechnergestützte Verarbeitung von Molekülstrukturen mit dem Ziel der Vorhersage von biologischen Eigenschaften setzt eine Kodierung der Moleküle in Form von

numerischen Werten voraus. Diese Kodierung erfolgt mit Hilfe von sogenannten „Deskriptoren“. Eine möglichst allgemein gehaltene Definition dieses Begriffes würde jegliche Art von auf einer Molekülstruktur basierenden Mess- oder Rechenvorschriften umfassen. Die von TODESCHINI & CONSONNI [TC00] in ihrem Standardwerk über Deskriptoren gegebene Definition enthält dagegen eine wichtige Einschränkung:

> *The molecular descriptor is the final result of a logical and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardized experiment.*

Es wird gefordert, dass ein Deskriptor eine *useful number* sein soll. Diese Nützlichkeit steht jedoch in einem engen Zusammenhang mit dem jeweils verfolgten Ziel bzw. der betrachteten Aufgabenstellung. Deskriptoren, die sich für die Vorhersage einer biologischen Eigenschaft eignen, sind nicht notwendigerweise auch für andere QSAR-Probleme geeignet. Letztlich muss also für jede Aufgabenstellung erst einmal untersucht werden, welche Arten von Deskriptoren dieses Nützlichkeitskriterium erfüllen.

Deskriptoren können danach unterteilt werden, ob sie auf experimentellen Messungen beruhen (wie $\log P$, Dipolmoment oder Polarisierbarkeit), oder ob sie mit Hilfe eines Computers berechnet werden. Die berechneten Deskriptoren spielen in der Praxis eine größere Rolle, da nur selten experimentelle Messwerte in ausreichender Anzahl und Qualität vorliegen.

Eine der ersten Rechenvorschriften für einen Deskriptor wurde von WIENER [Wie47] vorgeschlagen. Seitdem wuchs die Zahl der für unterschiedliche Problemstellungen erdachten und publizierten Deskriptoren stetig an. Eine weit verbreitete Einteilung dieser Vielzahl an berechenbaren Deskriptoren geschieht anhand der Menge an Informationen, die in die Berechnung eingeht. Es wird dabei unterschieden, ob z. B. die dreidimensionale Struktur des Moleküls bekannt sein muss, oder ob der Molekülgraph (die sogenannte 2D-Struktur) ausreicht. Diese Einteilung anhand der verwendeten Dimensionen eines Moleküls ist allerdings nur für den 2D- und den 3D-Fall wirklich schlüssig, die weiteren „Dimensionen“ sind durch eine nur teilweise geglückte Analogiebildung entstanden:

0D-Deskriptoren Verwenden nur die chemische Formel einer Substanz als Eingabe (also z. B. $\mathrm{C_2H_5OH}$ für Ethanol). Beispiele sind die Auftrittshäufigkeiten von einzelnen oder von zu Gruppen zusammengefassten Atomtypen (z. B. Heteroatome), oder aufsummierte Eigenschaften dieser Atomtypen (wie das Molekulargewicht, die Gesamtladung oder die Summe der van der Waals-Volumen).

1D-Deskriptoren Es wird von einer Auflistung der in einem Molekül vorkommenden Substrukturen ausgegangen. Diese Liste muss nicht vollständig sein, enthält also nicht notwendigerweise alle Informationen über die Molekülstruktur (daher die

Bezeichnung „eindimensional“). Bei den Substrukturen kann es sich um funktionelle Gruppen, Seitenketten, Ringstrukturen oder Grundgerüste (*scaffolds*) handeln.

2D-Deskriptoren Als Eingabe wird der zweidimensionale Graph des Moleküls (die *connection table*) verwendet. In diese Kategorie fallen beispielsweise die topologischen und die 2D-Autokorrelations-Deskriptoren. Zur Eingabe des Molekülgraphen sind auch Zeilennotationen wie SMILES geeignet (vgl. Abschnitt 2.4.7.1).

3D-Deskriptoren Verwenden die zuvor berechnete 3D-Struktur eines Moleküls als Eingabe. Probleme ergeben sich aus den unterschiedlichen Konformationen, in denen ein Molekül vorliegen kann, besonders wenn viele rotierbare Bindungen vorliegen. Da die Berechnung der 3D-Struktur *ab initio* zu aufwändig ist, können im besten Falle nur semiempirische Methoden eingesetzt werden. Bei großen Substanzbibliotheken wird oft auf eine Optimierung verzichtet und auf regelbasierte 3D-Strukturgeneratoren (wie z. B. CORINA) zurückgegriffen.[20] Beispiele sind geometrische, Oberflächen-basierte und auf einem Gitter[21] berechnete Deskriptoren.

4D-Deskriptoren Hier bildet nicht eine einzelne 3D-Struktur des Moleküls, sondern eine Menge von energiegünstigen Konformationen die Grundlage für die Berechnungen. Diese kann beispielsweise durch mehrmaliges, schrittweises Verändern der Winkelmaße an den rotierbaren Bindungen erzeugt werden. Nimmt man das Molekül als in einer wässrigen Lösung vorliegend an, dann kann die Zeit als die vierte Dimension dieser Art von Deskriptoren gesehen werden.

Im folgenden sollen die wichtigsten Arten von Deskriptoren vorgestellt werden.

2.4.5.1 Topologische Deskriptoren

Einer der ersten topologischen Deskriptoren wurde von WIENER [Wie47] vorgestellt. Zur Berechnung wird die Distanzmatrix eingesetzt:

$$W = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} = \sum_{i=1}^{n}\sum_{j>i}^{n} d_{ij} \tag{2.21}$$

Dabei steht n für die Anzahl der Atome im Molekül, d_{ij} bezeichnet die topologische Distanz zwischen zwei Atomen. Die Vereinfachung im zweiten Teil der Formel wird

[20] Solche nur über eine angenäherte 3D-Struktur berechneten Deskriptoren werden gelegentlich auch als „zweieinhalb-dimensional“ bezeichnet.

[21] Diese beispielsweise mit dem CoMFA-Verfahren generierbaren Deskriptoren setzen eine aufwändige Überlagerung (*alignment*) aller Molekülstrukturen voraus.

durch die Symmetrie der Distanzmatrix ermöglicht. Da W von der Größe des Atoms abhängt, wird oft auch der mittlere Wiener-Index $\bar{W}$ verwendet:

$$\bar{W} = \frac{2 \cdot W}{n \cdot (n-1)} \tag{2.22}$$

Die „Kompaktheit“ eines Moleküls ist durch $2 \cdot \bar{W}$ definiert. Die Anzahl der Verbindungen innerhalb eines Moleküls wird durch den *connectivity index* ${}^{m}\chi$ beschrieben. Der erste, auch *branching index* genannt, wurde von RANDIC [Ran75] vorgeschlagen:

$${}^{1}\chi = \sum\nolimits_{b} (\delta_i \cdot \delta_j)_b^{-1/2} \tag{2.23}$$

Dabei läuft b über alle Verbindungen eines Moleküls, δ_i und δ_j stellen den Verzweigungsgrad der an der Verbindung $i-j$ beteiligten Atome dar. Dieser Index erster Ordnung wurde von KIER & HALL [KH77, KH86] auf Pfadlängen von 0 bis m verallgemeinert:

$${}^{2}\chi = \sum_{k=1}^{{}^{2}P} (\delta_i \cdot \delta_j \cdot \delta_l)_k^{-1/2} \qquad \text{für } m = 2, \text{ bzw.} \qquad {}^{m}\chi = \sum_{k=1}^{{}^{m}P} \left(\prod_{a=1}^{n} \delta_a \right)_k^{-1/2} \tag{2.24}$$

Wobei ${}^{m}P$ für die Anzahl aller Pfade der Länge m im Molekül und n für die Anzahl der Atome innerhalb eines solchen Pfades steht.

2.4.5.2 Autokorrelations-Deskriptoren

Eine Autokorrelation kann zur (nicht verlustfreien) Transformation von Informationen unterschiedlicher Länge in eine Repräsentierung mit fester Länge verwendet werden. Dieses Prinzip machten sich MOREAU & BROTO [MB80a, MB80b] bei der Entwicklung ihres Verfahrens der *autocorrelation of topological structures* (ATS) zunutze. Zur Berechnung des Autokorrelationswertes $a(d)$ für eine topologische Distanz d wird die folgende Formel verwendet:

$$a(d) = \sum_{i=1}^{n} \sum_{j=1}^{n} \delta(d_{ij}, d) p_i p_j \tag{2.25}$$

Dabei steht n für die Anzahl der Atome im Molekül und d_{ij} für die kürzeste topologische Distanz zwischen den Atomen i und j. Der Wert einer vorgegebenen Eigenschaft des Atoms i wird von p_i angegeben. Die Funktion $\delta(x, y)$ entspricht dem Kronecker-Delta, für sie gilt:

$$\delta(x,y) = \begin{cases} 1 & : \quad x = y \\ 0 & : \quad x \neq y \end{cases} \qquad \text{bzw.} \qquad \delta(d_{ij}, d) = \begin{cases} 1 & : \quad d_{ij} = d \\ 0 & : \quad d_{ij} \neq d \end{cases} \tag{2.26}$$

Autokorrelations-Deskriptoren können für die physiko-chemischen Eigenschaften eines Atoms berechnet werden, wie z. B. die σ-Ladung, Partialladung, σ- und π-Elektronegativität und die Atompolarisierbarkeit [BZB+96], aber auch für topologische Eigenschaften, wie der Anzahl der Verbindungen eines Atoms.

2.4.5.3 Substruktur-Deskriptoren

Ein *substructure descriptor* (auch *fragment count* genannt) gibt an, ob und wie häufig eine vorgegebene Substruktur in einem Molekül auftritt. Diese Angabe kann entweder quantitativ als ganzzahliger oder qualitativ als binärer Wert erfolgen.

Entstanden ist diese Deskriptor-Klasse aus der Optimierung der Substruktur-Suche in Moleküldatenbanken. Eine *molecular fingerprint* genannte Menge von vorberechneten Substruktur-Deskriptoren dient zur Ausfilterung der aufgrund fehlender struktureller Merkmale nicht in Frage kommenden Moleküle der Datenbank. Nur in den verbleibenden Molekülen muss die zeitaufwändige Suche nach der jeweils vom Benutzer eingegeben Substruktur noch durchgeführt werden. Die Länge des *fingerprints* hängt jeweils von der Anzahl der vorberechneten Deskriptoren ab.

Eine Vereinheitlichung der Länge kann mit Hilfe der *hashed fingerprints* erreicht werden [IG94]. Hier werden die Auftrittshäufigkeiten von mehreren Substrukturen über einen Hashcode auf ein gemeinsames Feld (*bin*) des *fingerprints* abgebildet und aufaddiert. Dies kann sinnvoll sein, wenn die zugrundeliegenden Deskriptoren im Mittel nur selten vorkommen, was bei einem normalen *fingerprint* zu vielen leeren Feldern führen würde.

Zur Auswahl der den Deskriptoren zugrunde liegenden Substrukturen können folgende Kriterien herangezogen werden:

- Funktionelle Gruppen (wie Säuregruppen)
- Pharmakologisch relevante Gruppen (wie Akzeptor / Donor)
- Einzelne Atome und deren unmittelbare Umgebung (Ghose/Crippen)

2.4.5.4 Kriterien zur Bewertung

Das wichtigste Kriterium für die praktische Verwendbarkeit eines Deskriptors stellt dessen Informationsgehalt in Bezug auf eine bestimmte Problemstellung dar. In der folgenden Auflistung steht es daher an erster Stelle. Ergänzend dazu lassen sich noch eine Reihe von weiteren Kriterien identifizieren:

Informationsgehalt Die Frage, ob und wieviel Informationen ein Deskriptor zur Lösung einer Aufgabenstellung beiträgt, lässt sich meist nicht von vorneherein analytisch klären. Vielmehr kann erst durch den praktischen Einsatz des Deskriptors (allein oder im Zusammenspiel mit anderen) ein quantitatives Gütemaß generiert werden. Dieses kann als Fitness des betrachteten Deskriptors angesehen und beispielsweise zur Deskriptorselektion eingesetzt werden.

Auf einen konkreten Datensatz bezogen lässt sich der „Informationsgehalt“ mit Hilfe der Entropie bestimmen (analog zu Formel 2.10).[22] Da nach dieser Definition ein Deskriptor, der stets zufällige Werte ausgibt, den höchsten Entropiewert besitzt, ist dieses Maß nur eingeschränkt verwendbar. Sinnvoll ist jedoch die Festlegung einer unteren Schranke für die Entropie, um Deskriptoren, die für fast alle Strukturen einen konstanten Wert liefern, auszuschließen.[23]

Konformationsunabhängigkeit Einerseits besitzen auf der 3D-Struktur eines Moleküls basierende Deskriptoren große Vorteile, da das Volumen und die Ausdehnung eines Liganden wesentlich dessen Fähigkeit zur Einlagerung in die Bindungstasche eines Rezeptors beeinflussen. Auch die dabei wichtigen Wasserstoff-Brückenbindungen hängen sehr stark von der räumlichen Orientierung ab. Da die 3D-Struktur eines Moleküls jedoch in der Regel nicht starr ist, müssen bei einem von der Konformation abhängigen Deskriptor stets mehrere Werte für unterschiedliche Konformationen berechnet werden. Erschwerend kommt hinzu, dass die tatsächlich im Rezeptor vorliegende Konformation energetisch vergleichsweise ungünstig sein kann (z. B. bei einem *induced fit*). Dies führt dazu, dass nicht konformationsunabhängige Deskriptoren stets mit einem schwer abschätzbaren Unsicherheitsfaktor behaftet sind.

Rotations- und Translationsinvarianz Des Weiteren muss für 3D-Deskriptoren die Invarianz gegenüber Drehung und Verschiebung des Moleküls in seinem Koordinatensystem gefordert werden. Ist dies nicht der Fall, so kann über Regeln für eine eindeutige Positionierung des Moleküls sichergestellt werden, dass strukturell gleiche Moleküle dieselben Deskriptorwerte ergeben. Damit ist jedoch nicht sichergestellt, dass sich die für unterschiedliche Moleküle berechneten Werte sinnvoll vergleichen lassen. Vergleichsweise unbedeutende strukturelle Abweichungen im Molekül können zu einer Verschiebung im Koordinatensystem führen und damit zu völlig unterschiedlichen Deskriptorwerten. Von der Verwendung von Deskriptoren, die dieses Kriterium nicht erfüllen, ist daher im Allgemeinen abzuraten.

2.4.6 Wirkstoffähnlichkeit

Die in der Literatur beschriebenen Verfahren zur Bestimmung der „Wirkstoffähnlichkeit“ (*drug-likeness*) eines Moleküls lassen sich zum einen nach dem eingesetzten Klassifizierungsverfahren, zum zweiten nach den für das Training verwendeten Datenbanken und zum dritten nach den zur Beschreibung der Moleküle eingesetzten

[22] Die Entropie als Maß für den Informationsgehalt wurde bereits im Abschnitt 2.1.5 in Zusammenhang mit den Entscheidungsbaumverfahren besprochen.

[23] Diese untere Schranke ist insbesondere für Substruktur-Deskriptoren relevant, da viele komplexe Substrukturen nur sehr selten auftreten.

Deskriptoren unterscheiden. Ein direkter Vergleich der Ansätze wird dabei vor allem durch die unterschiedlichen Datensätze erschwert. Als *non-drug*-Datenbank wird meist eine Untermenge des *Available Chemicals Directory* (ACD) eingesetzt, als *drug*-Datenbank wahlweise der *World Drug Index* (WDI), der *MACCS-II Drug Data Report* (MDDR) oder die *Comprehensive Medicinal Chemistry*-Datenbank (CMC).

Aufgrund des von ANZALI *et al.* [ABC+01] für eine einzelne Methode durchgeführten Vergleichs der auf unterschiedlichen Datensätzen erzielbaren Klassifizierungsraten ist zumindest eine grobe Abschätzung der Schwankungsbreite auch für andere Methoden möglich.

2.4.6.1 Klassifizierung mit Neuronalen Netzen

In dem Ansatz von SADOWSKI & KUBINYI [SK98] wird ein vollständig verbundenes *feed forward*-Netz mit fünf versteckten Neuronen und einem Ausgabeneuron verwendet. Das Training erfolgt mit Hilfe des im Neuronale Netze-Simulators von ZELL *et al.* [ZMV+95] enthaltenen *backpropagation with momentum*-Lernverfahrens. Als Trainingsdaten kommen je 5 000 Substanzen zum Einsatz, die aus vorgefilterten Versionen des WDI (*drugs*) und ACD (*non-drugs*) zufällig ausgewählt wurden. Es wird ein Klassifizierungsfehler von 23% auf dem WDI und von 17% auf dem ACD erreicht.

Eine sehr gute Klassifizierungsrate von je 90% auf jeweils 3 500 zufällig ausgewählten Substanzen aus den CMC- und ACD-Datenbanken, welche die *drugs* und die *non-drugs* repräsentieren, konnte dagegen von AJAY, WALTERS & MURCKO [AWM98] erreicht werden. Auch die Generalisierungsleistung war zufriedenstellend, es wurden 80% der MDDR-Datenbank korrekt als *drug-like* erkannt. Hierbei wurde ein Bayessches Neuronales Netz (BNN) eingesetzt. Als Eingaben wurden sieben nach subjektiven Kriterien gewählte Deskriptoren in Kombination mit 166 ISIS-Keys verwendet. Diese Gesamtzahl an Deskriptoren konnte mit Hilfe des Bayesschen Lernverfahrens auf 78 reduziert werden. Ein Vergleich mit dem Entscheidungsbaum-Verfahren C4.5 ergab für letzteres deutlich schlechtere Ergebnisse.

Eine vergleichbar gute Klassifizierungsrate von 88% konnte von FRIMURER *et al.* [FBN+00] unter Verwendung der MDDR- und ACD-Datenbanken erreicht werden. Als Eingabe wurden 80 Substruktur-Deskriptoren verwendet, deren Anzahl durch systematisches Auslassen von jeweils einem Deskriptor auf 15 reduziert werden konnte. Das Training erfolgte mit einem *feed forward*-Netz mit 200 Neuronen in einer versteckten Schicht, wobei unterschiedliche Anzahlen von Neuronen getestet wurden. Die Vorhersage der *drug-likeness* wurde über ein einzelnes Ausgabeneuron realisiert, bei dem durch entsprechende Wahl eines *cut off*-Wertes zwischen 0 und 1 die Anzahl der falsch positiven oder falsch negativen Vorhersagen beeinflusst werden konnte.

Dass eine solche Unterscheidung auch mit anderen Deskriptorarten möglich ist, wurde von MURCIA-SOLER *et al.* [MPG+03] gezeigt. Mit Hilfe eines *feed forward*-Netzes

mit zwei versteckten und einem Ausgabeneuron, das mit dem Standard-Backpropagation-Verfahren des SNNS trainiert wurde, konnten Klassifizierungsraten von 93% auf der Traingsmenge von 780 Substanzen und von 82% auf einer unabhängigen Validierungsmenge bestehend aus 86 Substanzen erreicht werden. Als Eingabe wurden insgesamt 62 topologische Indizes verwendet.

2.4.6.2 Entscheidungsbaum-Verfahren

Der Vorteil von Entscheidungsbaumverfahren liegt laut WAGENER & VAN GEERESTEIN [WvG00] darin, dass sich aus den generierten *decision trees* direkt Kriterien zur Unterscheidung von *drugs* und *non-drugs* ablesen lassen. In ihrer Studie werden je 5 000 Substanzen zufällig aus dem ACD und WDI ausgewählt. Als Eingabe dienen 121 Ghose-Crippen-Deskriptoren. Der Klassifizierungsfehler des mit C5.0 trainierten Baumes liegt insgesamt bei 17.4%. Wenn *false positives* stärker bestraft werden als *false negatives*, so lässt sich ein Fehler von nur 8.1% auf den Wirkstoffen erzielen, der Fehler bei den nicht-Wirkstoffen erhöht sich entsprechend auf 34.3%. Die Analyse des trainierten Entscheidungsbaums ergibt eine überraschend kleine Menge von einfachen Features, auf denen diese Klassifizierung beruht: Es wird vor allem das Vorhandensein oder die Abwesenheit von funktionellen Gruppen, die Wasserstoffbrücken ausbilden können, bewertet.

2.4.6.3 Selbstorganisierende Karten

Von BRÜSTLE *et al.* [BBS$^+$02] wird als Argument für die Verwendung von Kohonen-Karten die Tatsache angeführt, dass eine repräsentative Zusammenstellung der benötigten *drug* und insbesondere der *non-drug*-Datenbanken schwierig ist. Daher eignen sich die nicht von einer solchen Unterteilung abhängigen unüberwachten Verfahren besonders gut. Aus einer Menge von 66 Deskriptoren werden mittels *recursive partitioning* (FIRM) lediglich drei ausgewählt, mit denen nach einem Training einer 100x100-Knoten großen SOM deutlich ein zentraler *drug*-Cluster erkennbar ist. Auch gewisse Untermengen (wie z. B. alle Hormone) werden zusammenhängend auf kleinere Cluster abgebildet.

2.4.6.4 Statistische Methoden

Die von POROIKOV *et al.* [PFB$^+$00] zur Vorhersage von biologischen Aktivitätsspektren entwickelte Methode PASS (*Prediction of Activity Spectra for Substances*) wurde von ANZALI *et al.* [ABC$^+$01] zur Bestimmung der Wirkstoffähnlichkeit eingesetzt. Es wurden fragmentbasierte Deskriptoren verwendet, die nach dem *multilevel neighborhoods of atoms*-Verfahren (MNA) von FILIMONOV *et al.* [FPBG99] zusammengestellt

wurden. Dabei handelt es sich um eine Verallgemeinerung des von Ghose und Crippen verwendeten Nachbarschaftsradius auf mehrere Stufen.

Der von XU & STEVENSON [XS00] definierte *drug-like index* (DLI) basiert auf der Bewertung von Substanzen mit Hilfe von 25 strukturellen Deskriptoren. Zur Auswahl der später eingesetzten Deskriptoren wird eine Menge von unter chemischen Gesichtspunkten als subjektiv sinnvoll betrachteten strukturellen Deskriptoren getestet. Bei dem Test werden die Werteverteilungen (Histogramme) zwischen einer wirkstoffähnlichen und einer -unähnlichen Substanzdatenbank verglichen. Ergibt sich ein als signifikant angesehener Unterschied, so wird der Deskriptor selektiert. Steht die Menge der Deskriptoren fest, so wird jeweils der Schwerpunkt der Werteverteilung für die wirkstoffähnliche Substanzdatenbank bestimmt (der sogenannte *cluster center*). Aus dem Abstand einer Substanz vom *cluster center* wird der DLI berechnet. In dem Histogramm des als *non-drug*-Datenbank verwendeten ACD werden zwar 28.3% der Substanzen mit einem relativen DLI von weniger weniger als 1% bewertet, andererseits werden 60% der Substanzen als zu mehr als 50% wirkstoffähnlich angesehen. Daraus kann gefolgert werden, dass sich mit der berichteten Vorgehensweise ohne weitere Verbesserungen z. B. bei der Auswahl der Deskriptoren keine befriedigende Unterscheidung erzielen lässt.

2.4.6.5 Wissensbasierte Methoden

In der von MUEGGE [MHB01] beschriebenen Methode wird ein einfaches, aus chemischen Vorwissen abgeleitetes Filterkriterium vorgestellt. Der Vorteil eines solchen „Expertensystems“ gegenüber Neuronalen Netzen wird darin gesehen, dass über die Regelsätze die Einordnung der Moleküle im Einzelfall begründet werden kann. Es werden sämtliche potentiell in einem Pharmakophor vorkommenden Gruppen definiert (*pharmcophore points*), von denen bei Wirkstoffen mindestens 2 und nicht mehr als 7 vorkommen dürfen. Andernfalls sind die Moleküle unter- bzw. überfunktionalisiert. Zusätzlich werden vier weitere Regeln definiert, eine davon besagt beispielsweise, dass Moleküle ohne Ringstrukturen nicht zu den Wirkstoffen zu zählen sind. Mit diesem Regelsatz konnten je nach gewählter Teilmenge 61–68% der CMC-Datenbank als *drug like* erkannt werden, 64% des ACD wurden als nicht wirkstoffähnlich bewertet.

2.4.7 Kodierung von Molekülstrukturen

In diesem Abschnitt werden verschiedene Möglichkeiten zur Kodierung von Molekülstrukturen aufgezeigt. Die Kodierung in ein wohldefiniertes Datenformat stellt einen wesentlichen Teilschritt bei jeder Computer-basierten Verarbeitung von Molekülen dar. Die Konvertierung zwischen unterschiedlichen Datenformaten stellt ein häufig

auftretendes Folgeproblem dar und führte zur Entstehung von zahlreichen Konvertierungsprogrammen, von denen wohl BABEL den größten Bekanntheitsgrad erlangt hat. Zu den aktuellen Nachfolgern dieses Programms zählen OELIB bzw. OPENBABEL (www.eyesopen.com) und JOELIB (joelib.sourceforge.com) . Eine ausführliche Abhandlung über chemische Datenformate ist in dem Standardwerk von GASTEIGER & ENGEL [GE03, Kap. 2] zu finden.

2.4.7.1 Zeilennotationen

Der Vorteil von sogenannten Zeilennotationen liegt vor allem in ihrer Kompaktheit. Eine komplette Molekülstruktur passt „in eine Textzeile“ hinein, lässt sich also im Regelfall mit deutlich weniger als 80 Zeichen kodieren. Damit sind solche Formate einerseits ideal für Eingabemasken z. B. von Internet-basierten Abfragen. Andererseits lassen sich umfangreiche Molekülbibliotheken platzsparend ablegen. Der Nachteil der Formate ist darin zu sehen, dass eine Vorarbeitung vor den eigentlichen Berechnungen notwendig wird, bei der z. B. die 2D- oder 3D-Strukturen der Moleküle berechnet werden müssen.

SMILES In der von WEININGER [Wei88, WWW89] entwickelten SMILES-Notation (*Simplified Molecular Input Line Entry System*) werden Atome durch ihre Symbolnamen bezeichnet. Wasserstoffatome werden nicht explizit angegeben, sondern an allen freien Bindungsstellen implizit ergänzt. Einfachbindungen zwischen zwei Atomen werden durch Konkatenation der entsprechenden Atomsymbole kodiert, Mehrfachbindungen dagegen explizit angegeben. Seitenketten von Molekülen werden in verschachtelbare Klammern gesetzt. Ringschlüsse werden durch eine eindeutige, paarweise Nummerierung der beteiligten Atome ermöglicht. Beispiele für einfache SMILES-Ausdrücke sind in Abb. 2.16 dargestellt.

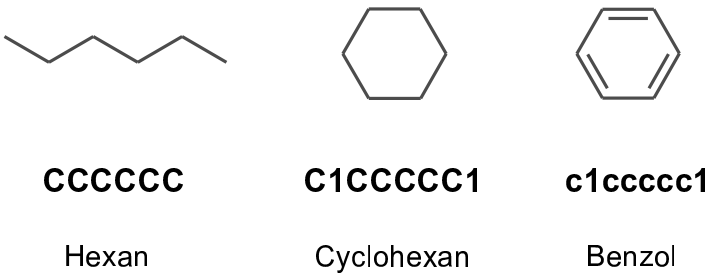

Abbildung 2.16: Beispiele für die SMILES-Zeilennotation zur Kodierung von Molekülen.

SMARTS Die SMARTS-Notation (*SMiles ARbitrary Target Specification*) wurde von der Firma DAYLIGHT (www.daylight.com) entwickelt und stellt eine Erweiterung von SMILES für Substruktursuchen dar. Hinzugefügt wurden für beliebige

Atomtypen stehende Wildcards (∗) und Listen von alternativen Atomtypen, welche in eckigen Klammern notiert werden. Ein Nachteil von SMARTS ist, dass Flexibilität nur auf der Ebene von einzelnen Atomen möglich ist. Es ist daher z. B. nicht möglich, sogenannte *spacer* (als Abstandshalter dienende Kohlenstoffketten) von variabler Länge zu definieren. Zwei beispielhafte SMARTS-Patterns sind in Abb. 2.17 abgebildet.

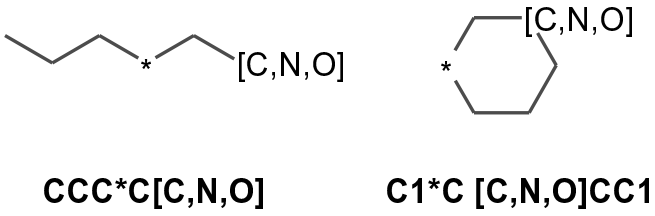

Abbildung 2.17: Eine Kette und ein Ring in der SMARTS-Notation. Der Stern steht für ein beliebiges Atom, der Ausdruck in eckigen Klammern für die Auswahl aus einer Liste von Atomtypen.

2.4.8 Optimierung von Molekülstrukturen

In diesem Abschnitt wird auf unterschiedliche Ansätze zur Evolution von Molekülstrukturen eingegangen. Die Einteilung erfolgt in erster Linie nach dem eingesetzten Optimierungsverfahren (GA, ES oder GP). Dabei ist anzumerken, dass in der Literatur oft auch nicht auf Bitstrings arbeitende Verfahren als GA bezeichnet werden. Diese besonders im amerikanischen Raum anzutreffende Begriffsaufweichung wird hier beibehalten, sofern ansonsten ausreichend viele GA-Merkmale vorliegen. Es wird zudem stets gesondert auf die jeweils gewählte Kodierung der Molekülstrukturen hingewiesen bzw. danach unterteilt.

Auf Verfahren, die die Bindungseigenschaften von Liganden innerhalb einer Rezeptortasche optimieren (*de novo design*), wird an dieser Stelle nicht näher eingegangen. Ein Überblick über dieses Gebiet wird beispielsweise von GILLET *et al.* [GNM+94] gegeben.

2.4.8.1 Zufallsgesteuerte Suche

Der wohl erste Versuch der Computer-gestützten Generierung von Molekülstrukturen wurde von DERRINGER & MARKHAM [DM85] unternommen. Ziel war die Herstellung eines Polymers mit gewünschten Eigenschaften. Aus einer Menge von sieben

funktionellen Gruppen wurden per Zufall Ketten von bis zu vier Gruppen zusammengestellt. Lagen die additiv berechneten Eigenschaften nahe genug am vorgegebenen Optimum, dann wurde die Struktur ausgegeben.

Schon wesentlich komplexer ist das von NILAKANTAN *et al.* [NBV91] vorgestellte Verfahren. Auch hier wird über eine reine Zufallssuche ein Molekül aus Fragmenten aufgebaut, das Ergebnis kann in diesem Fall allerdings auch nichtlinear sein. Aus einer Menge von Fragmenten wird wiederholt zufällig eines ausgewählt und durch Elimination von Wasserstoffatomen an das bisher aufgebaute Molekül angefügt. Das Verfahren bricht ab, wenn ein vorgegebenes Molekülgewicht erreicht ist. Die generierten Moleküle werden schließlich auf ihre Ähnlichkeit zu einer Menge von Zielmolekülen getestet.

2.4.8.2 Evolution mit Genetischen Algorithmen

Das Problem einer rein durch den Zufall gesteuerten Suche ohne Beschränkung des Suchraums besteht darin, dass die generierten Moleküle in den meisten Fällen nicht oder nur mit sehr großem Aufwand synthetisierbar sind. Eine entsprechende Beschränkung kann bei Optimierungsverfahren wie den GAs entweder direkt über die Kodierung oder aufwändiger über Bestrafungsterme in der Bewertungsfunktion erfolgen.

Die folgenden Ansätze sind zum einen nach der Klasse der optimierbaren Moleküle sortiert (z. B. Polymere), sofern das Verfahren nur für diese Klasse geeignet ist. Bei allgemeineren Verfahren, welche die Optimierung beliebiger Molekülstrukturen erlauben, wird nach der jeweiligen Kodierung eingeordnet.

Polymere Ziel der Arbeit von VENKATASUBRAMANIAN *et al.* [VCC94, VCC95] ist die Optimierung der physiko-chemischen Eigenschaften von Polymeren. Diese werden wahlweise in einer LISP-artigen Notation oder als lineare Aneinanderreihung von einzelnen Atomen oder Substrukturen repräsentiert. Es werden fünf Mutations- und Crossover-Operatoren definiert. In großen Suchräumen konnten die durch ihre Eigenschaften vorgegebenen Zielstrukturen in acht von neun Fällen gefunden werden.

Tripeptoide Die Optimierung von Tripeptoiden erfolgt bei SHERIDAN *et al.* [SK95] über die Kombination von Fragmenten. Diese werden über ihren Index kodiert und in einem Bitstring repräsentiert. Es werden spezielle Mutations-Operatoren verwendet, die die Ähnlichkeit von Fragmenten berücksichtigen. Zur Bewertung der Fitness eines Moleküls werden *atom pair*-Deskriptoren als Maß der Ähnlichkeit zu einem Zielmolekül verwendet. Zudem werden mit Trend-Vektoren die Deskriptorwerte mit einer biologischen Aktivität korreliert.

Hexapeptide In der Arbeit von SINGH *et al.* [SAJ+96] wird die Fitness eines zu optimierenden Hexapeptids direkt über einen biologischen Test bestimmt. Die Kodierung erfolgt über die Abbildung des Index der einzelnen Aminosäuren auf

jeweils 5 Bits des Bitstrings. Die Synthetisierung von 300 Hexapeptiden (aus 20^6 möglichen) in 5 Generationen deutet eine Steigerung der Fitness pro Generation an, der Versuch konnte allerdings nicht zu Ende geführt werden.

Kombinatorische Misch-Bibliotheken Bei dem von BROWN & MARTIN [BM97] vorgestellten Verfahren sind nicht einzelne Moleküle, sondern ganze Bibliotheken (*mixture libraries*) das Ziel der Optimierung. Dementsprechend wird für alle an einem Substitutionspunkt einsetzbaren Fragmente ein eigenes Bit im Bitstring reserviert, welches das Vorkommen des Fragments im Gemisch anzeigt. Die Gesamtzahl der Moleküle ergibt sich dann durch die möglichen Fragmentkombinationen. Als Fitness wird die Diversität und die Dekonvolutionsfähigkeit verwendet.[24]

Kombinatorische Einzel-Bibliotheken Das Treffen einer Auswahl von einzeln zu synthetisierenden Substanzen aus einer virtuellen kombinatorischen Bibliothek ist das Ziel der Arbeiten von GILLET *et al.* [GWBG99, GKW$^+$02]. Dabei setzt sich der Gencode aus den als Integer-Zahlen kodierten Indices der ausgewählten Fragmente zusammen. Die Gesamtgröße der Bibliothek ist über die Länge des Gencodes vorgegeben. Die Optimierung kann nach mehreren Kriterien (wie Diversität, *drug likeness*) gleichzeitig erfolgen.

SMILES Das von DOUGET *et al.* [DTG00] vorgestellte Verfahren verwendet die SMILES Zeilennotation zur Kodierung der Moleküle. Insgesamt werden dafür zwei Crossover- und 13 Mutationsoperatoren definiert. Bei einer Mutation werden Fragmente, Aminosäuren, Ringe oder funktionelle Gruppen eingefügt oder gelöscht, Ringe geschlossen oder einzelne Atome mutiert.

3D-Koordinaten Direkt auf der 3D-Struktur des Moleküls arbeitet das Verfahren von GLEN & PAYNE [GP95]. Dazu werden 2 Crossover- und 12 Mutationsoperatoren verwendet. Da in dem gewählten Ansatz die räumliche Lage des Moleküls von Bedeutung ist, dienen drei der Mutationsoperatoren der Translation und Rotation des Moleküls und der Drehung um eine Bindung. Mit den anderen werden Fragmente eingefügt oder gelöscht, Ringe geschlossen oder geöffnet und Atom- oder Bindungstypen mutiert. Ein Fragment besteht aus einer Substruktur, die über Einfachbindungen mit dem Rest des Moleküls verbunden werden kann. Bei der Berechnung der Fitness werden 15 Arten von *constraints* beachtet. Eines davon ist eine *look-up table* mit chemisch instabilen Vierer-Atomketten.

Baumstruktur mit fester Topologie In dem Ansatz von GOH & FOSTER [GF00] werden die in eine Bindungstasche einzupassenden Liganden in Form einer fest vorgegebenen Baumstruktur repräsentiert. Elternknoten haben drei Kinder, mindestens zwei davon sind Terminalknoten. Jeder Knoten enthält eine von sieben

[24] Die Dekonvolutionsfähigkeit ist ein Maß dafür, ob sich das Molekülgewicht des aktiven Bestandteils im Gemisch noch über massenspektroskopische Verfahren ermitteln lässt.

möglichen funktionellen Gruppen. Die feste Topologie erlaubt die Abbildung des Baums auf einen Bitstring und den Einsatz von Standard-Operatoren. Die Fitness wird anhand eines schematischen zweidimensionalen Modells der Bindetasche berechnet.

Genetischer Graph Von GLOBUS *et al.* [GLW99, GLW01] wird ein auf dem Molekülgraphen arbeitendes Verfahren vorgestellt. Ein neu entwickelter Crossover-Operator sorgt dafür, dass die resultierenden Strukturen chemisch sinnvoll und zusammenhängend sind.

Binäre Kodierung von Fragmenten Das von WEBER *et al.* [WWBG95] vorgestellte Verfahren optimiert die kombinatorische Synthese von 10 Isocyaniden, 40 Aldehyden, 10 Aminen und 40 carbozyklischen Säuren. Die Kodierung erfolgt über den binärisierten Index des jeweiligen Fragments. Es werden die Standard-GA-Operatoren verwendet. Aus den 160 000 möglichen Kombinationen wurde bei einer Populationsgröße von 20 bereits nach 16 Generationen Thrombin-Inhibitoren mit IC_{50}-Werten im submikromolaren Bereich gefunden. In der Untersuchung von ILLGEN *et al.* [IEBW00] wurden sämtliche 15 360 Kombinationen einer 3-Komponenten-Reaktion synthetisiert und auf Trypsin-Inhibition getestet. Die Daten ermöglichten den Vergleich von verschiedenen Kodierungen (binär, integer) der Fragmente. Es wurde gezeigt, dass große Populationen für binärkodierte GAs von Vorteil sind, und das eine Integer-Kodierung eine höhere Mutationsrate benötigt und bessere Ergebnisse als eine Binärkodierung erbringt.

Kodierung sämtlicher Fragmente In dem von GOBBI & POPPINGER [GP98] beschriebenen Ansatz wird das Auftreten sämtlicher Fragmente in einem Molekül (bis zu einer gegebenen Länge) jeweils mit einem Bit im Bitstring kodiert. Da es sehr viele solcher Kombinationen geben kann, wird ein *folded fingerprint* verwendet. Es wird nur der Standard-GA-Operator zur Rekombination verwendet, die Diversität wird durch Hinzufügen zweier zufälliger Moleküle pro Generation erhalten. Das Ergebnis einer solchen Rekombination enthält in der Regel wiedersprüchliche Aussagen über vorkommende Fragmente (z. B. `CNC` aber nicht `N`), so dass mit einer Tabu-Search Methode jeweils das ähnlichste Molekül in einer virtuellen kombinatorischen Bibliothek gesucht wird. Der entstehende hohe Rechenaufwand ist dabei immer noch gering im Vergleich zur Dauer der Synthese des entstehenden Moleküls. Das Verfahren ist in der Lage, Moleküle mit hoher Bindungsstärke zu finden, wobei die Erfolgsquote mit der Größe der kombinatorischen Bibliothek ansteigt.

Weitere Ansätze aus dem Bereich der kombinatorischen Chemie sind in dem Übersichtsartikel von WEBER [Web98] zu finden.

2.4.8.3 Evolutionsstrategien

Das von SCHNEIDER *et al.* [SCH+00] vorgestellte Verfahren basiert auf einem Fragment-Ansatz. Die im WDI enthaltenen Substanzen werden mit der RECAP-Methode zerlegt, um möglichst „wirkstoffähnliche“ Fragmente zu erhalten. Diese können dann mit elf Reaktionsschemas wieder kombiniert werden. Die Indizes der Fragmente werden als reellwertige Zahlen kodiert und mit einer $(1, \lambda)$-Evolutionsstrategie optimiert. Als Fitnessfunktion wird die mit Hilfe eines topologischen Pharmakophordeskriptors (TOPAS) bestimmte Ähnlichkeit zu einem Templat-Molekül verwendet. Diese ist über die Paarhäufigkeiten von fünf Atomtypen (Donor, Akzeptor, positiv oder negativ geladen, lipophil) definiert. Moleküle, die zu groß sind oder zu viele Atome eines bestimmten Typs haben, werden mit zusätzlichen Straftermen belegt.

2.4.8.4 Simulated Annealing

Der von KVASNIČKA *et al.* [KP96] entwickelte Simulated Annealing-Ansatz arbeitet entweder auf azyklischen Graphen (die Kodierung erfolgt über eine von READ [Rea76] entwickelte lineare Notation) oder auf allgemeineren zyklischen Graphen (kodiert über eine Adjazenzmatrix). Es wird jeweils ein Mutationsoperator (bei SA als *pertubation* bezeichnet) definiert. Ziel ist die Optimierung von Alkanen hinsichtlich ihrer $^{13}\mathrm{C}$ NMR-Verschiebung (*chemical shift*), bei der ein vorgegebener Wert erreicht werden soll.

2.4.8.5 Genetische Programmierung

Bei dem von NACHBAR [Nac98, Nac00] entwickelten Ansatz werden Moleküle in Form von Bäumen kodiert. Dabei stehen in den Knoten die Verbindungstypen und in den Blättern die Atomtypen. Es sind einfach- bis dreifach-Bindungen und insgesamt 22 unterschiedliche Atomtypen vorgesehen. Wasserstoffatome werden explizit angegeben. Ein spezieller, mit einem Index versehener „Ringbindungs“-Atomtyp ermöglicht die Definition von Ringstrukturen. Vertreter dieses Sondertyps mit derselben Indexnummer müssen genau zweimal auftreten. Um dies sicherzustellen, werden spezielle Mutations- und Rekombinations-Operatoren verwendet. Als Fitnesskriterium dient die Ähnlichkeit zu einem oder mehreren Zielmolekülen. Berechnet wird diese mit Hilfe des Ähnlichkeitsmaßes von DICE [Dic45] auf *atom pair*-Deskriptoren. Die Leistungsfähigkeit des Verfahrens hängt laut NACHBAR [Nac00] stark von den als Templat vorgegebenen Molekülstrukturen ab: Je komplexer die Zielvorgabe, desto häufiger endet die Suche in einem lokalen Minimum.

2.5 Technische Standards

In diesem Abschnitt werden die zur Implementierung von SOLVES eingesetzten technischen Standards CORBA und XML beschrieben. Bei beiden handelt es sich um wohldefinierte, offene, Plattform-unabhängige und von zahlreichen Firmen unterstützte Standards. Die Einhaltung solcher Standards vereinfacht die Kommunikation zwischen Programmen verschiedener Hersteller, sorgt für eine längere Einsatzfähigkeit eines Programmes und erhöht die Zugriffssicherheit auf erstellte Daten.

2.5.1 Der CORBA-Standard

2.5.1.1 Geschichtliche Entwicklung

Eine Eigenschaft von fast allen größeren Computer-Netzwerken, seien sie nun im Internet oder in einem Firmen-internen Intranet angesiedelt, ist ihre Heterogenität. Es sind typischerweise Rechner von diversen Herstellern anzutreffen, auf denen Betriebssysteme wie Linux, Unix-Derivate und verschiedene Windows-Versionen laufen. Die eingesetzten Applikationen sind ebenfalls von unterschiedlichen Herstellern in Sprachen wie Java, C, C++, Perl, Python oder gar COBOL implementiert. Falls eine Kommunikation zwischen Applikationen über das Netzwerk vorgesehen ist, so finden sich auch hier eine Vielzahl von Möglichkeiten, wie TCP/IP, Sockets oder *Remote Procedure Calls* (RPC).

Ausgehend von dieser Situation wurde im Jahr 1989 die *Object Management Group* (OMG) gegründet, mit dem Ziel der Erstellung und Verbreitung von offenen, nichtproprietären Standards zur Erstellung von verteilten Applikationen in heterogenen Netzwerken. Die OMG (www.omg.org) hat sich seitdem mit mehr als 850 beteiligten Firmen zu einem der weltweit größten Software-Konsortien entwickelt.

2.5.1.2 Das OMA Referenz-Modell

Die *Object Management Architecture* (OMA) stellt eine abstrakte Beschreibung der verteilten Objektwelt aus der Sicht der OMG dar [Obj96]. Sie setzt sich aus den folgenden Komponenten zusammen:

- **Object Services** stellen grundlegende Schnittstellen bereit, welche die Entwicklung von CORBA-Anwendungen vereinfachen oder überhaupt erst ermöglichen. Als Beispiel sei ein *Naming Service* genannt, der ein zentrales Verzeichnis von Objektreferenzen verwaltet.
- **Common Facilities** stellen höherwertige Schnittstellen mit Ausrichtung auf die Endanwender zur Verfügung.

- **Domain Interfaces** stellen auf die Bedürfnisse von spezifischen Branchen ausgerichtete Schnittstellen bereit.
- **Application Interfaces** bezeichnen sämtliche nicht-standardisierten Schnittstellen, welche von einzelstehenden Anwendungen angeboten werden.

Als zentrale Komponente dient dabei der *Object Request Broker* (ORB), mit dessen Hilfe ein weitgehend transparenter Informationsaustausch zwischen auf unterschiedlichen Plattformen befindlichen Client- und Server-Objekten ermöglicht wird. Seine Eigenschaften werden im Rahmen des weit verbreiteten *Common Object Request Broker Architecture* (CORBA) Industriestandards spezifiziert [Obj95].

2.5.1.3 Die Interface-Beschreibungssprache IDL

In der *Interface Definition Language* (IDL) werden die von einem CORBA-Objekt bereitzustellenden Methoden deklariert. Mit Hilfe eines *language mappings* wird die so definierte Schnittstelle in eine konkrete Programmiersprache übertragen. Bisher existieren Abbildungen auf die Sprachen C, C++, Java, COBOL, Smalltalk, Ada, LISP und Python, sowie auf die eigens definierte Skriptsprache „IDLscript“.

Eine der umfangreichsten Open-Source-Implementierungen des CORBA 2.3 Standards für die Sprache C++ wurde von PUDER & RÖMER [PR00] unter der Bezeichnung MICO erstellt.

2.5.2 Der XML-Standard

Die *e**X**tensible **M**arkup **L**anguage* (XML) ist als Projekt des WORLD WIDE WEB CONSORTIUMS (W3C) enstanden und wurde von der XML WORKING GROUP spezifiziert. Die Spezifikation von XML 1.0 wurde 1998 als Empfehlung publiziert und im Jahr 2000 noch einmal überarbeitet (www.w3.org/TR/REC-xml) .

Bei XML handelt es sich um eine Meta-Sprache, die zur Definition von sogenannten Markup-Sprachen (wie beispielsweise XHTML) angewendet wird. XML stellt eine Vereinfachung und weitgehende Teilmenge von SGML dar (vgl. Abschnitt 2.5.2.1).

Auf dem XML-Standard basierende Dateiformate sind dabei, sich in vielen Anwendungsbereichen als bevorzugtes Ein- und Ausgabeformat von Applikationen durchzusetzen. Neben dem asynchronen Datenaustausch über Dateien gewinnt auch die direkte, synchrone Datenübermittlung über SOAP-kompatible Web-Services stetig an Bedeutung. Und auch im Bereich der internen Datenhaltung in der Persistenzschicht von Applikationen kommt zunehmend XML zum Einsatz, entweder über ein Objekt-XML-Mapping oder über ein Datenbank-Managementsystem (DBMS) mit XML unterstützender Funktionalität.

Ein wesentlicher Vorteil gegenüber den zuvor meist eingesetzten proprietären Binärformaten ist, dass die XML-Dateien auch für Menschen lesbar und editierbar sind. Ein Informationsverlust, wie er bei nicht mehr unterstützten Binärformaten auftreten kann, wird so verhindert. Ein bedeutender Nachteil liegt in der hohen Redundanz von XML-Dateien, welche durch die sich ständig wiederholenden öffnenden und schließenden *tags* erzeugt wird. Durch einfache Lauflängen-basierte Komprimierungsalgorithmen kann der Platzbedarf meist bereits auf ein Zehntel des ursprünglichen reduziert werden, noch besser gelingt dies mit speziell für XML-Dateien entwickelten Komprimierungsverfahren.

2.5.2.1 Geschichtliche Entwicklung

Die vorwiegend im Verlagswesen eingesetzte ***S**tandard **G**eneralized **M**arkup **L**anguage* (SGML) kann als Vorgänger von XML angesehen werden. Die Entwicklung von SGML begann bereits in den 60-70er Jahren des vorigen Jahrhunderts [Gol97]. Seit 1986 ist SGML als ISO-Standard (ISO 8879:1986) definiert.

Auch SGML wurde bereits mit dem Ziel der langfristigen, plattformunabhängigen Speicherung von großen Datenmengen in textueller Form entwickelt. Langfristig bedeutet hier, dass die Daten auch auf Rechnern nachfolgender Generationen noch lesbar sein sollen, auch wenn die Applikation, mit denen die Dateien ursprünglich erstellt wurden, bereits nicht mehr verfügbar sind (vgl. http://xml.coverpages.org/sgml.html).

Die hohe Komplexität von SGML verhinderte allerdings eine Ausbreitung auf weitere Anwendungsbereiche, diese konnte erst mit dem vereinfachten XML-Standard erreicht werden.

2.5.2.2 Die Struktur von XML-Dokumenten

XML-Dokumente sind hierarchisch aufgebaut, wobei die Schachtelung mit Hilfe von *tags* erfolgt. Der Name und die Attribute eines Tags werden im Rahmen der *document type definition* (DTD) festgelegt. Ein Tag kann entweder sofort wieder geschlossen werden, oder er kann Text (und evtl. auch weitere Tags) umschließen und mit dem End-Tag beendet werden.

2.5.2.3 Eigenschaften von XML

Im folgenden sind die wesentlichen Eigenschaften von XML aufgeführt. Dabei werden auch verwandte Standards wie XSL mit einbezogen.

Trennung von Struktur und Darstellung Während viele der in HTML vorhandenen Tags die Darstellung (Formatierung und Layout) des Textes beschreiben, dienen

die in XML-Dokumenten vorkommenden Tags ausschließlich der Strukturierung des Dokuments. Da XML als Metasprache dem Benutzer die Definition und damit die Sinngebung von Tags ermöglicht, können natürlich auch nur auf die Darstellung bezogene Tags definiert werden (wie z. B. in XHTML). Insofern ermöglicht XML die Trennung von Struktur und Darstellung, erzwingt sie aber nicht. Eine Beschreibung der gewünschten Darstellung der Inhalte kann dann in einem zweiten Schritt mit Hilfe von ***C**ascading **S**tyle **S**heets* (CSS) oder einer XSL-Transformation erfolgen. Wahlweise (z. B. bei sehr komplexen Daten) kann für die Darstellung auch ein spezialisierter Viewer oder die erzeugende Applikation selber eingesetzt werden.

Lesbarkeit XML-Dokumente sind sowohl für den Menschen als auch für Applikationen prinzipiell les- und auswertbar, sofern sie wenigstens wohlgeformt sind. Somit wird sichergestellt, dass einmal erstellte Daten auch nach einem Verlust der erstellenden Applikation noch weiterverarbeitet und in andere Formate konvertiert werden können.

Plattformunabhängigkeit Diese ist durch die explizite Angabe der verwendeten Zeichenkodierung bei XML-Dokumenten gewährleistet. Dadurch ist der Einsatz auch nicht auf eine bestimmten Sprache oder Kulturkreis beschränkt. Die Verarbeitung von Inhalten in z. B. japanischer oder chinesischer Sprache ist durch Wahl einer geeigneten Kodierung problemlos möglich. Voraussetzung dafür ist natürlich, dass die verarbeitende Applikation die gewählte Zeichenkodierung unterstützt, also z. B. Unicode-fähig ist. Seit XML 2.0 können auch die Tags selber in beliebigen Kodierungen definiert werden. Allerdings stehen z. B. in japanischen Kanji definierte Tags einer universellen Lesbarkeit im Weg, so dass hiervon meist abzuraten ist.

2.5.2.4 Die Formatierungssprache XSL

Die *e**X**tensible **S**tylesheet **L**anguage* (XSL) ist eine XML-Applikation, die zur Transformation von XML-Dokumenten in andere Formate eingesetzt wird. Die Beschreibung dieser Transformationen erfolgt dabei mit Dokumenten im XSLT-Format. Mögliche Ziele dieser Transformation sind:

Transformation nach XHTML Das Ergebnis dieser Transformation kann direkt in einem Web-Browser dargestellt werden.[25] Neben der Textformatierung kann bei dieser Transformation auch nicht benötigte Information ausgelassen oder anders angeordnet werden. So könnte z. B. für verschiedene Benutzergruppen jeweils ein eigenes XSLT-Dokument zur Verfügung gestellt werden.

[25] Da die Ausgabe einer XSL-Transformation nicht unbedingt selber im XML-Format sein muss, kann wahlweise auch ein auf einen bestimmten Web-Browser zugeschnittenes HTML-Dokument erstellt werden.

Transformation in ein anderes XML-Format Ermöglicht den Austausch von Daten zwischen Applikationen, die verschiedene XML-Formate verwenden. Zur Konvertierung der XML-Dokumente reicht im Idealfall die Definition eines XSLT-Dokuments aus.[26] Dies ist ein Fortschritt gegenüber der Konvertierung von proprietären Binärformaten, für die in jedem Fall ein spezialisiertes Konvertierungsprogramm benötigt wurde.

Transformation in ein proprietäres Textformat Ermöglicht den Austausch von Daten zwischen einer XML-basierten Applikationen und Applikationen, die eigene Textformate verwenden. Ob eine Konvertierung mit XSLT möglich ist, hängt hierbei stark von der Struktur des Zielformates ab.

2.5.2.5 Ein XML-Standard für modellbildende Verfahren

Mit der *Predictive Model Markup Language* (PMML) lassen sich die Ergebnisse der Berechnungen von modellbildenen Verfahren zwischen Applikationen verschiedener Hersteller austauschen. Dabei werden die folgenden Verfahren unterstützt:

- Neuronale Netze (MLP und RBF)
- Cluster-Verfahren (zentrums- und verteilungsbasiert; auch Kohonenkarten)
- Regression (linear, polynomial und logistisch; auch multivariate Regression)
- Naive Bayes-Verfahren
- Entscheidungsbaum-Verfahren
- Assoziationsregel-Verfahren
- Sequenzbasierte Verfahren

Der Standard wurde 1997 an der UNIVERSITY OF ILLINOIS entworfen und wird seit 1998 in einem prominent besetzten Industriekonsortium namens DATA MINING GROUP (www.dmg.org) weiterentwickelt. Seit der aktuellen Version 2.1 erfolgt die Spezifikation des Standards über XML Schemas. Da die Schema-Definitionen für die vollständige Beschreibung eines gültigen PMML-Dokuments nicht ausreichen, werden in einer zusätzlichen formlosen Spezifikation weitere Anforderungen aufgeführt. Der Standard lässt sich an zahlreichen Stellen mit eigenen Tags erweitern. Neue Attribute können zu bestehenden Tags hinzugefügt werden, solange eine Namenskonvention eingehalten wird. Nur eine Untermenge des Standards (*core features*) für eines der oben genannten Verfahren wird als Konformitätskriterium verlangt.

[26] Dieser Idealfall ist nicht gegeben, wenn z. B. im einem XML-Dokument numerische Daten enthalten sind, die in irgendeiner Form umgerechnet werden müssen.

An dem an sich sehr sinnvollen PMML-Standard lassen sich die folgenden Schwachpunkte bzw. potentiellen Probleme identifizieren:

- Es wird nur ein Teil der bekannten modellbildenden Verfahren unterstützt (es fehlen bespielsweise die *Support Vector Machines*).
- Die unterstützten Verfahren werden teilweise recht willkürlich zusammengefasst (so werden Kohonenkarten als reine Cluster-Verfahren behandelt).
- Nur fertig trainierte Modelle können im Rahmen des Standards beschrieben werden, nicht aber die verwendeten Eingabeparameter oder Besonderheiten des jeweiligen Lernverfahrens.
- Der Standard ist relativ weich, da eine Implementierung selbst dann noch konform ist, wenn sie nur die minimal geforderten *core features* umsetzt, diese aber mit eigenen Attributen und Tags erweitert.

Letztendlich führen diese Schwächen von PMML die Tatsache vor Augen, dass die zahlreichen und sehr diversen modellbildenden Verfahren nur schwer und nicht ohne Reibungsverluste in einem umfassenden Standard zusammengefasst werden können.

Kapitel 3

Das SOLVES Programmsystem

In diesem Kapitel wird das im Rahmen dieser Arbeit implementierte Programmsystem SOLVES beschrieben. Die sich aus der Ausgangslage ergebenden Anforderungen werden im folgenden aufgezählt. Hieraus ergeben sich eine Anzahl von Design-Entscheidungen, die den detailliert beschriebenen Aufbau des Systems beeinflussen. Schließlich wird diskutiert, welche der Eigenschaften von SOLVES sich gegenüber vergleichbaren Systemen hervorheben und daher als Alleinstellungskriterium angesehen werden können.

3.1 Ausgangslage

Für die Auswertung von Datensätzen, beispielsweise mit dem Ziel der Klassifizierung oder der numerischen Vorhersage von Messgrößen, steht heutzutage eine große Auswahl von potentiell erfolgreichen Methoden zur Verfügung. Neben den klassischen, auf eher einfachen statistischen Zusammenhängen beruhenden Verfahren haben sich in den letzten Jahren besonders Methoden aus dem Bereich des maschinellen Lernens und der evolutionären Optimierung etabliert.

Dabei ist jedoch zu beobachten, dass bisher kein einzelnes Verfahren sämtliche zuvor bereits existierenden Methoden ersetzen oder gar verdrängen konnte. Für eine bisher unbekannte Problemstellung muss vielmehr stets von neuem geprüft werden, welches der zur Verfügung stehenden Verfahren am besten geeignet ist. Je nach eingesetztem Verfahren müssen dann noch weitere nachgeordnete Probleme gelöst werden.[1]

Im Zunehmen begriffen ist auch die Anzahl der Kombinationen von verschiedenen Verfahren. So kann die erwähnte Netz-Topologie mit einem Genetischen Algorithmus optimiert werden. Dieses Verfahren kann wiederum mit einer Selektion der Eingabevariablen kombiniert werden [YH01]. Dieses Beispiel soll die vielfältigen sich ergebenden Möglichkeiten andeuten.

[1] Zum Beispiel, welche Topologie für ein Neuronales Netz gewählt werden soll.

Das Ziel der Entwicklung von SOLVES ist es, dem in den Anwendungsgebieten der oben genannten Verfahren arbeitenden Praktiker ein Werkzeug in die Hand zu geben, welches die einfache und flexible Zusammenstellung solcher Kombinationen ermöglicht. In diesem sollen möglichst viele unterschiedliche Methoden aus dem Bereich des maschinellen Lernens und der evolutionären Optimierung, aber auch andere klassische oder anwendungsbezogene Methoden als Module integrierbar sein. Ein wichtiger Punkt ist hierbei die gemeinsame, einheitliche Benutzungsoberfläche.

3.2 Anforderungen und Entwurf

In diesem Abschnitt werden die Anforderungen an ein verteiltes, auf dem Workflow-Paradigma basierendes System im Anwendungsbereich der pharmakologischen Forschung dargelegt. Diese Anforderungen wurden gemeinsam mit den im SOL-Projekt beteiligten Industriepartnern herausgearbeitet.

(A1) Integrierte Benutzerschnittstelle Die Vielzahl der im Wirkstoffdesign eingesetzten Methoden soll unter einer einzelnen, alle Methoden integrierenden Benutzungsoberfläche zusammengefasst werden. Diese Vereinheitlichung soll die Einarbeitungszeit verkürzen und die Anwendung der Methoden erleichtern.

(A2) Modularer Aufbau Jede Methode soll in einem Modul zur Verfügung stehen. Diese Module sollen einzeln oder in Kombination aufgerufen werden können. Als Ein- und Ausgabe werden sowohl einfache Datentypen, als auch komplexe, zusammengesetzte Datentypen benötigt.[2]

(A3) Flexibilität Die Abfolge, in denen die Module aufgerufen werden (im folgenden als Prozesskette oder *workflow* bezeichnet), soll nicht fest vorgegeben sein. Änderungen das Workflows sollen möglichst wenig Aufwand erfordern.

(A4) Erweiterbarkeit Das Gebiet der Wirkstoffsuche entwickelt sich in kurzen Innovationszyklen weiter, so dass bereits nach wenigen Monaten neue Methoden hinzukommen und andere nicht mehr gefragt sein können. Das Hinzufügen neuer Module soll daher einfach und auch für einen Pharmaforscher möglich sein. Es soll keine Einschränkungen bezüglich der möglichen Funktionalität neuer Module geben.

(A5) Plattformunabhängigkeit Das entwickelte Programm soll auf allen in der Pharmaforschung eingesetzten Plattformen lauffähig sein. Zu den verwendeten Plattformen zählen Microsoft Windows, SUN Solaris, zunehmend Linux und in abnehmenden Maße auch SGI IRIX.

[2] Ein einfacher Datentyp kann beispielsweise aus einem ganzzahligen Wert bestehen, ein komplexer Datentyp kann z. B. einen kompletten Klassifizierungs-Datensatz umfassen.

(A6) Verarbeitung großer Datenmengen Die Anzahl der in Pharmafirmen vorliegenden chemischen Substanzen bewegt sich im Bereich von 10^6. Ergänzt werden diese durch die kombinatorisch zugänglichen „virtuellen Substanzen“. Zu jeder Substanz müssen Tausende von berechneten oder experimentell ermittelten Zahlenwerten verwaltet werden können.

(A7) Verteilung der Berechnungen Neben den typischerweise für die Aufgabenstellungen des Wirkstoffdesigns eingesetzten Multiprozessor-Rechnern stehen den Mitarbeitern auch zunehmend leistungsfähige Arbeitsplatzrechner zur Verfügung. Berechnungen sollen daher auf einem Arbeitsplatzrechner zusammengestellt und gestartet werden können und dann auf den Multiprozessor-Rechnern ausgeführt werden, wobei weitere freie Rechenkapazitäten je nach Bedarf einbezogen werden sollen. Die Notwendigkeit der Verteilung ergibt sich auch aus der Tatsache, dass einige der eingesetzten Module nur auf ganz bestimmten Plattformen zur Verfügung stehen.[3]

(A8) Nachvollziehbarkeit der Berechnungen Gerade im Bereich der pharmazeutischen Forschung ist es wichtig, dass sich alle für eine Berechnung maßgeblichen Parameter auch noch Monate oder Jahre später nachvollziehen lassen. Daher soll eine Möglichkeit zur permanenten und langfristigen Datenspeicherung vorhanden sein.

(A9) Anbindung an bestehende Datenbanken Typischerweise liegen in Pharmafirmen bereits umfangreiche Datenbanksysteme vor, in denen Informationen über chemische Substanzen, HTS-Läufe, kombinatorische Bibliotheken usw. abgelegt sind. Ein Zugriff auf diese Datenbanken soll gewährleistet sein, wofür die Fähigkeit zur Anpassung an unterschiedliche Datenbankschemata eine Voraussetzung ist.

Bei der Auswertung der Anforderungen fällt auf, dass sich diese gegenseitig beeinflussen und teilweise auch widersprechen. Im folgenden werden einige der sich ergebenden Gegensatzpaare aufgeführt:

Bedienbarkeit vs. Flexibilität: Eine große Flexibilität führt zu einer vergleichsweise hohen Komplexität der Benutzungsoberfläche. Mit der Anzahl der Kombinationsmöglichkeiten der Module erhöht sich auch fast zwangsläufig die Anzahl möglicher Fehlerquellen.

Große Datenmengen vs. Verteilung: Die Verteilung der Berechnungen erfordert den Transport der benötigten Daten auf den Zielrechner. Bei großen Datenmengen stellt sich die Frage, wie der zwangsläufig entstehende Overhead reduziert werden kann.

[3] Beispielsweise ist eines der Programme zur Berechnung von Deskriptoren (DRAGON PLUS von TODESCHINI (www.disat.unimib.it/chm)) nur für das WINDOWS-Betriebssystem erhältlich.

Integration vs. Erweiterbarkeit: Die geforderte Integration der vorhandenen Methoden (A1) ist nur durch Definition von gemeinsamen Schnittstellen möglich. Diese können die zukünftige Erweiterbarkeit (A4) einschränken, da sie auch für später hinzuzufügende, bisher unbekannte Methoden verbindlich sind.

Im folgenden werden einige der grundsätzlichen Design-Entscheidungen aufgeführt und begründet, die im Vorfeld der Implementierung von SOLVES getroffenen wurden. Die in Klammern gesetzten Zahlen stellen dabei jeweils Verweise auf die entsprechenden Anforderungen dar.

Java als Programmiersprache Der geforderte modulare Aufbau (A2) legt die Wahl einer objektorientierten Programmiersprache nahe. Im Vergleich zwischen Java und C++ besticht Java durch die höhere Plattformunabhängigkeit (A5), besonders in Bezug auf die Benutzungsoberfläche (A1).

CORBA als Middleware Aus der zweiten Anforderung folgt auch hier die Beschränkung auf objektorientierte Methoden der Rechnerkommunikation. Im Vergleich von CORBA und DCOM wird wegen (A5) der offenere Standard CORBA bevorzugt. Gegenüber der Java-spezifischen Lösung RMI hat CORBA den Vorteil der Unabhängigkeit von der Programmiersprache, was die Erweiterbarkeit (A4) auf der Modulebene erhöht. Zudem lässt sich so bereits vorhandener, in unterschiedlichen Programmiersprachen vorliegender *legacy code* leichter einbinden.

XML als Datenablageformat Die in (A8) geforderte langfristige Datenspeicherung wird durch die Verwendung des XML-Standards ermöglicht. Beim Einsatz von XML mit großen Datenmengen (A6) lassen sich Probleme in Bezug auf Haupt- und Festplattenspeicherplatz voraussehen, so dass zusätzlich Strategien zur Datenreduktion erforderlich werden.

Verwendung des Workflow-Paradigmas Aus den Anforderungen (A3) und (A4) ergibt sich, dass dem Benutzer die Zusammenstellung der Abfolge der Berechnungen selbst zu überlassen ist. Daher werden die Module in der Benutzungsoberfläche als graphische Elemente dargestellt, deren Ein- und Ausgabeparameter der Benutzer interaktiv miteinander verbinden kann.

3.3 Die Software-Architektur von SOLVES

In diesem Abschnitt wird der programmiertechnische Aufbau von SOLVES beschrieben. Auf die graphische Benutzungsoberfläche wird in Abschnitt 3.5 eingegangen.

3.3.1 Übersicht

Aus Sicht der Software-Architektur wurde bei SOLVES ein sogenannter *n-tier* Aufbau gewählt. Während bei dem weit verbreiteten 3-Schichten-Aufbau zwischen Präsentations-, Geschäftslogik- und Persistenzschicht unterschieden wird, sind bei SOLVES neben einer Präsentationsschicht (der Benutzungsoberfläche) mehrere Arten von Servern vorhanden. Diese werden mit Hilfe eines Server-Managers verwaltet und greifen jeweils auf eigene Datenquellen zu. Neben den spezialisierten Servern (wie dem Archivserver), die meist nur einmal vorhanden sind, kann letztlich auch jedes einzelne Modul des Workflows als eigener Server betrachtet werden. Dies ermöglicht zum Beispiel, dass die Berechnungen innerhalb einer Prozesskette auf unterschiedlichen Rechnern ablaufen können, ohne dass der Benutzer etwas von dieser Verteilung der Berechnungen bemerkt. Andererseits erhöhen sich dadurch auch die Anforderungen an den nachfolgend vorgestellten Workflow-Manager, da mit zunehmender Verteilung der Overhead der Datenübermittlung zunimmt. Dieser kann bei großen Datensätzen zum geschwindigkeitsbestimmenden Element der Gesamtlaufzeit der Prozesskette werden. Durch entsprechende Caching-Mechanismen kann dieses Verhalten reduziert und in manchen Fällen ganz vermieden werden.

3.3.2 Server und Module

Die nachfolgend aufgeführten spezialisierten Server-Arten werden im folgenden oft auch nur kurz als „Server“ bezeichnet. Demgegenüber stehen die „Module“ eines Workflows, welche auch eine gewisse Server-Charakteristik aufweisen. Beide werden vom Server-Manager verwaltet, wobei die Module mehrfach vorhanden sein können.

3.3.3 Der Server-Manager

Neben der Verwaltung von Servern und Modulen übernimmt dieser auch einige grundlegende Funktionen eines Naming-Service. Ein solcher häufig bei CORBA-Applikationen zum Einsatz kommender Service stellt die Verbindung zwischen den verteilten Objekten einer Applikation her. So kann der Server-Manager beispielsweise die Verbindung zwischen einer als Client ablaufenden SOLVES-Instanz zu einer der aktuell im Server-Modus ablaufenden und frei zur Verfügung stehenden Instanz herstellen. Der hierfür notwendige Austausch des sogenannten IORs (*interoperable reference strings*) erfolgt über eine beiden Seiten bekannte Datenquelle, genauer gesagt über eine Tabelle in einer Datenbank. Diese Vorgehensweise hat sich als flexibler erwiesen als der Austauch der IORs über ein Dateisystem oder eine Internetadresse. Zudem kann ein konkurrierender Zugriff mit Hilfe von Datenbank-Transaktionen geregelt werden. Beiden Seiten wird hierbei ein schreibender Zugriff gewährt, wodurch eine Pflege der eventuell bereits wieder veralteten IORs einfach implementiert werden kann: Versucht

ein Client vergeblich, sich mit einem der aufgelisteten Server zu verbinden, so kann dieser den Eintrag dieses möglicherweise abgestürzten Servers entfernen. Umgekehrt müssen die im Server-Mode ablaufenden Instanzen prüfen, ob der von ihnen hinterlegte Eintrag in die IOR-Liste noch vorhanden ist.

3.3.4 Der Archiv-Manager

Während sich der Server-Manager um die Verwaltung der Funktionalität (also der dynamischen Bestandteile) kümmert, stehen beim Archiv-Manager die Daten (und somit die statischen Bestandteile) im Vordergrund. Alle Daten werden dabei grundsätzlich in Form von CORBA-Objekten verwaltet. Diese stellen lediglich eine gewisse Zugriffslogik zur Verfügung (ähnlich wie *beans* in Java). Der statische Aspekt dieser Daten liegt darin begründet, dass die Datenobjekte nach der Erstellung[4] nicht mehr abgeändert werden können. Dies ist notwendig, damit die (aus diesem Grund recht grob granularen) Berechnungen einer Prozesskette jederzeit wieder nachvollzogen werden können. Die einzige erlaubte Änderung besteht darin, ein nicht mehr benötigtes Zwischenergebnis vollständig zu löschen (z. B. in einer Optimierungsschleife). Diese Herangehensweise hat einerseits Nachteile: Bei einer naiven Umsetzung würden große Datenmengen akkumuliert werden, die irgendwann auch den Rahmen der Datenablagen sprengen würden. Die nachfolgend beschriebene Lösung dieses Problems erhöht wiederum den Verwaltungsaufwand. Andererseits lassen sich auch Vorteile identifizieren: Lokale Caching-Strategien können eingesetzt werden, bei denen ein Abgleich mit dem zentralen Datenspeicher (z. B. über einen Broadcast oder einen Listener-Mechanismus) nicht notwendig ist, da einmal angelegte Datenobjekte im Cache stets ihre Aktualität behalten.

Die Datenobjekte werden im folgenden auch als *datasets* bezeichnet, da sie jeweils eine Ansammlung von Daten beinhalten, die in einem gewissen Sinnzusammenhang stehen (z. B. einen Klassifizierungsdatensatz oder eine Kohonen-Karte). Diese bestehen wiederum aus sogenannten Datenelementen (DE). Hierbei handelt es sich um mehr oder wenige primitive Datentypen, die für sich genommen meist keine Bedeutung mehr aufweisen. Daten-Elemente besitzen einen Datentyp (string, double, int, ...) und eine Dimension (Skalar, Matrix, ...). Sie werden in den Datasets in Form von Referenzen verwaltet, wobei hier als weiteres, die Struktur des Datensatzes beschreibendes Merkmal noch die Kardinalität (einfach, mehrfach oder mindestens einmal vorhanden) hinzukommt. Neben diesen primitiven Datenelementen können auch beliebig komplexe oder auf gewisse Anwendungen spezialisierte DE erstellt werden. So wird ein SOLVES-Workflow beispielweise in einem eigenen DE verwaltet.[5] Für solche spezialisierten DE muss jeweils eine separate XML-Formatierung festgelegt und imple-

[4] Eine erfolgreiche und vollständig abgeschlossene Erstellung eines Datenobjektes wird durch ein Flag markiert.

[5] Ein interessanter Nebenaspekt hiervon ist, dass dadurch auch Prozessketten denkbar werden, die wiederum Prozessketten verarbeiten, z. B. um sie zu optimieren.

mentiert werden, sie besitzen also jeweils ein eigenes, wenn auch meist eher kleines XML-Format.

Der Vorteil dieses zweistufigen Aufbaus von Datenobjekten liegt nun darin, dass eine „Wiederverwertung“ von bereits bestehenden Datenelementen möglich ist. In der Regel geschieht dies bei der Zusammenstellung der Ergebnis-Datensätze eines Moduls. Wenn manche Bestandteile der Eingabe-Datensätze unverändert in die Ausgabe fließen, so müssen hierfür keine neuen DE erstellt werden. Dies ist z. B. der Fall bei der Vorverarbeitung von Datensätzen, bei der Aufteilung in Trainings- und Testdaten oder bei der Selektion von Deskriptoren. Theoretisch wäre es auch möglich, zu einem früheren Zeitpunkt erstellte DE zu reaktivieren, wenn diese genau denselben Inhalt aufweisen (z. B. über die Berechnung von Hash-Codes). Allerdings dürfte dies bei den typischerweise bearbeiteten großen, meist reellwertigen DE den Aufwand nur selten lohnen.

3.3.5 Der Workflow-Server

Der Workflow-Server dient zur Verwaltung von einer oder mehreren Prozessketten. Diese sind wiederum aus Modulen, Datenobjekten, den dazwischen bestehenden Verbindungen und weiteren sogenannten Workflow-Objekten (wie z. B. Kommentarfeldern) aufgebaut. Es können mehrere Prozessketten gleichzeitig ablaufen, wobei für jedes Modul der aktuelle Zustand mitverfolgt wird. Beispiele für mögliche Zustände sind „bereit“, „fehlende Eingabedaten“, „fehlerhafter Abbruch“ oder „erfolgreich beendet“. Die Ausführung der Prozessketten erfolgt nach einem einfachen, rundenbasierten Schema. In jeder Runde werden alle Module gestartet, die im Zustand „bereit“ sind. Die nächste Runde beginnt, sobald sich der Zustand eines der laufenden Module ändert. Das Ende der Ausführung kommt, wenn das letzte laufende Modul seinen Zustand geändert hat (auf „beendet“ oder „Fehler“), aber kein Modul dadurch in einen startbereiten Zustand gewechselt ist (z. B. weil inzwischen fehlende Eingabedaten bereitgestellt wurden). Anzumerken ist, dass innerhalb der Standard-Module keine Schleifen auftreten dürfen. Zwar wäre eine Einführung von Schleifen prinzipiell denkbar (z. B. über Zähler in den Modulen, die nur eine bestimmte Anzahl von Starts zulassen), die Ausführung über das beschriebene rundenbasierte System wäre jedoch viel zu ineffizient (man denke nur an die tausendfach durchlaufenen Optimierungsschleifen von Evolutionären Algorithmen). Aufgrund dieser Überlegungen wurde ein spezieller Modultyp eingeführt (das sogenannte Kontroll-Modul), welcher eine effiziente Ausführung von Schleifen ermöglicht.

3.3.6 Der Application Domain-Server

Der Application Domain-Server stellt Daten aus dem jeweiligen Anwendungsgebiet von SOLVES zur Verfügung. Im Gegensatz zu den im Archiv verwalteten Datensätzen

handelt es sich hierbei um Eigenschaften der untersuchten Studienobjekte (hier auch als *entities* bezeichnet), auf welche jeweils über eine eindeutige Nummer (*identifier*) zugegriffen werden kann. Im Falle des pharmakologisch-chemischen Einsatzgebietes handelt es sich bei den untersuchten Objekten um Molekülstrukturen,[6] bei den Eigenschaften der Moleküle um bereits im Vorfeld berechnete oder experimentell bestimmte Deskriptorwerte. Im Allgemeinen kann davon ausgegangen werden, dass solche Informationen über das Anwendungsgebiet bereits vorliegen; schließlich werden zumindest von Firmen selten völlig neue Betätigungsfelder in Angriff genommen. Hierbei kann es sich dann entweder um Sammlungen von mehr oder weniger strukturierten Dateien (z. B. um manuell gepflegte Excel-Dateien) handeln, welche dann aufbereitet und in ein von SOLVES vorgegebenes Datenbankschema importiert werden müssen. Häufig liegen diese Daten aber auch bereits konsolidiert in Form von Datenbanken vor. In diesem Fall muss ein Adapter bereitgestellt werden, welcher auf die jeweilige Quelldatenbank zugreifen kann. Im Rahmen dieser Arbeit wurde ein konfigurierbarer „universeller“ Adapter erstellt, der bei einfach gehaltenen, flachen Tabellenstrukturen der Quelldatenbank zum Einsatz kommen kann. In den meisten Fällen wird der Adapter jedoch eigens implementiert werden müssen.

Der Vorteil einer solchen Anbindung von anwendungsspezifischen Datenbanken an SOLVES liegt darin begründet, dass Datensätze nun direkt zusammengestellt werden können. Über Eigenschaftsfilter lassen sich Untermengen von Studienobjekten definieren,[7] welche dann z. B. zur Erstellung eines Klassifizierungsdatensatzes verwendet werden können. Auch eine Ergänzung von bereits bestehenden Datensätzen um weitere Eigenschaften wäre denkbar. Je nach Anwendungsgebiet kann die anwendungsspezifische Datenbank auch zur Generierung von graphischen Abbildungen der Studienobjekte eingesetzt werden. Die Struktur der untersuchten Moleküle lässt sich beispielsweise zwei- oder dreidimensional visualisieren, wenn diese in einem bekannten Format in den Datenbanken vorliegt.

3.3.7 Definition von Parametern

Parameter stellen das Bindeglied zwischen den Modulen dar. Jedes Modul besitzt einen oder mehrere Ein- und Ausgabeparameter, die wegen der potentiellen Verteilung der Module auf mehrere Rechner über das CORBA-Protokoll übertragen werden. Parameter werden in IDL als Struktur (`struct`) definiert, deren Elemente in Tab. 3.1 aufgelistet sind. Das wichtigste Element ist dabei das „value“-Feld vom Typ `any`. Dieser Typ zeichnet sich dadurch aus, dass er beliebige andere IDL-Datentypen enthalten

[6] Genauer gesagt um sogenannte „kleine Moleküle“ (*small molecules*, SMOL), welche als Liganden eines Rezepters dienen können. Ergänzend dazu könnten auch Listen von größeren Molekülstrukturen (Rezeptoren bzw. Proteine) verwaltet werden.

[7] An Stelle der übergreifenden Bezeichnung „Studienobjekt“ wird im folgenden auch öfter konkret von „Molekülen“ gesprochen, wobei stets auch andere Arten von Studienobjekten mit eingeschlossen sein sollen.

Element-Name	IDL-Typ	Beschreibung
name	`string`	Innerhalb eines Moduls eindeutiger Name, nur serverseitig verwendet.
description	`string`	Kurze Beschreibung des Parameters, wird dem Benutzer angezeigt.
isInputType	`boolean`	Gibt an, ob der Parameter zur Ein- oder zur Ausgabe verwendet wird.
isControlled	`boolean`	Gibt an, ob es sich um einen sog. „kontrollierten“ Parameter handelt.
isLinkable	`boolean`	Gibt an, ob der Parameter gerade als „verlinkbar“ exponiert ist.
relationName	`string`	Wird belegt, wenn der Typ dieses Parameters mit dem eines anderen in Beziehung steht.
value	`any`	Bestimmt anfangs über den Default-Wert den Datentyp, enthält danach den jeweils aktuellen Wert.

Tabelle 3.1: Die Elemente einer Parameter-Struktur

kann. Im „value“-Feld wird somit der eigentliche Wert des Parameters transportiert. Es dient gleichzeitig zur Festlegung des Parametertyps. Daher muss bei der Erstellung eines neuen Parameters für dieses Feld immer ein Default-Wert angegebenen werden. Das Feld „isControlled“ gibt an, ob ein Parameter für den Aufruf eines kontrollierten Moduls verwendet wird. Die Bedeutung des Feldes „relationName“ wird in Abschnitt 3.3.8 erläutert.

Die folgende Aufzählung beschreibt die bisher vorhandenen Parametertypen. Bis auf die primitiven, bereits in IDL vorhandenen Datentypen und dem Datensatz-Typ sind alle über jeweils eine IDL-`struct` mit entsprechenden Elementen definiert worden.

Primitive Datentypen: Dazu zählen die bereits in IDL zur Verfügung stehenden Datentypen `boolean`, `float`, `double`, `short`, `long` und `long long`.[8]

Beschränkte Datentypen: Die Datentypen `ConstrainedLong` und `ConstrainedDouble` enthalten jeweils obere und untere Grenzwerte für den gültigen Wertebereich. Die Verwendung der beschränkten Datentypen ist gegenüber den primitiven vorzuziehen, da die Grenzwerte automatisch als *constraints* in eine Optimierung übernommen werden können.

[8] Die in IDL definierten Typen `long` und `long long` werden in Java auf die Datentypen `int` und `long` abgebildet.

Einfache Listen: Dabei handelt es sich um ein Array eines vorgegebenen Datentyps mit variabler Länge. Definiert wird es beispielsweise über `sequence<double>`.[9]

Auswahllisten: Der Datentyp `SelectionList` enthält eine Liste von Optionen, aus der genau eine ausgewählt werden kann. Die `MultiSelectionList` lässt dagegen die Auswahl von mehreren Optionen zu. Bei der `ParameterSelectionList` ist jeder Option zusätzlich eine Liste von Parametern zugeordnet. Dies bewirkt, dass in der Benutzerschnittstelle jeweils nur die der aktuellen Auswahl zugeordneten Parameter sicht- und editierbar sind.

Bezeichnerlisten: Bei dem Datentyp `NamedDoubleList` wird für jeden enthaltenen `double`-Wert zusätzlich ein Bezeichner angegeben. Die Typen `SingleSelectDoubleList` und `MultiSelectDoubleList` lassen die Auswahl von einem oder mehreren Werten zu. Sie werden z. B. bei der Übertragung von Fitnesswerten zu einem Optimierungsmodul verwendet.

Ordner: Der `Folder`-Datentyp enthält eine Liste der in diesem Ordner befindlichen Parameter. Es handelt sich um ein reines Strukturierungselement, welches z. B. bei der Definition von Makros zum Einsatz kommt.

In der obigen Aufzählung fällt auf, dass sich die Parametertypen `MultiSelectionList` und `MultiSelectDoubleList` recht ähnlich sind (letzterer enthält zusätzlich ein Array von `double`-Werten). Dies ist durchaus so gewollt, da sie zu verschiedenen Zwecken eingesetzt werden. Selbst zwei verschiedene Parameter mit genau gleichem Inhalt können sinnvoll sein, wenn diese auf unterschiedliche Arten eingesetzt werden sollen oder eine unterschiedliche Bedeutung tragen.

Die Erweiterung der obigen Aufzählung um einen neuen Parametertyp erfordert die folgenden Schritte:

- **Erweiterung der IDL-Schnittstelle:** Damit der Wert eines Parameters in einer `any`-Variable abgelegt werden kann, muss er als Teil des CORBA-Interfaces in Form einer `struct` deklariert werden.
- **Erstellung von Hilfsklassen:** Diese sind notwendig, um einen Zugriff auf den Wert des Parameters innerhalb von Modulen zu vereinfachen.
- **Optional Erstellung von Konvertern:** Da Parameter innerhalb eines Workflows nur mit gleichen oder über eine Konvertierung erreichbaren anderen Parametern verbunden werden können, empfiehlt sich es sich in vielen Fällen, mindestens eine Möglichkeit zur Umrechnung in einen bereits häufig eingesetzten Parametertyp anzubieten.

[9] IDL kennt auch einen `array`-Datentyp mit fester Länge, der für sich allein genommen allerdings zu unflexibel ist, um als Parameter eingesetzt zu werden.

Das Hinzufügen von neuen Parametern ist also aufwändig und „teuer“, vor allem da die grundlegende IDL-Schnittstelle erweitert werden muss. Andererseits ist dieser Schritt auch nur selten notwendig, da alle anwendungsspezifischen Daten in Form von (flexibel definierbaren) Datensätzen innerhalb eines Standard-Parameters übermittelt werden können.

3.3.8 Konvertierung von Parametern und Datensätzen

Eines der Entwurfsziele von SOLVES liegt in der einfachen Anwendbarkeit für den Praktiker. Die Prozessketten sollen daher möglichst einfach und übersichtlich gehalten werden. Zu diesem Zweck wurde ein Mechanismus zur automatischen Konvertierung zwischen den unterschiedlichen Parametertypen implementiert, der im folgenden beschrieben wird.

Bei der Erstellung von Verbindungen zwischen zwei Modulen in einer Prozesskette wird stets ein Ausgabe- mit einem Eingabeparameter kombiniert.[10] Dies ist in erster Näherung nur möglich, wenn beide genau denselben Datentyp besitzen. Ist in dem Parameter ein Datensatz enthalten, muss zusätzlich noch der Typ des Datensatzes verglichen werden. Dieser Ansatz der exakten Zuordnung hat sich in der Praxis allerdings als zu starr erwiesen. Eine flexiblere Zuordnung mit implizierter Konvertierung der Datentypen ist wünschenswert, da sonst in zu vielen Fällen Module nicht „zueinander passen“. Hierbei lassen sich zum einen eine verlustfreie Konvertierung von einer verlustbehafteten unterscheiden. Im ersten Fall geht keine Information verloren, z. B. wenn ein `short`-Wert nach `integer` oder ein `integer`-Wert nach `double` konvertiert werden. Im zweiten Fall wird der Verlust von Informationen bewusst in Kauf genommen, z. B. bei einer Konvertierung von `float` nach `integer`. Je nach Datentyp sind verschiedene Arten der Konvertierung denkbar (z. B. Rundung oder Abschneiden der Nachkommastellen). Zum anderen kann auch eine „Datenextraktion“ sinnvoll sein, bei der z. B. aus einem Ergebnisdatensatz ein einzelner `double`-Wert herausgenommen wird. Dieser Schritt erfordert eine Interaktion mit dem Benutzer, da oft mehrere gleichartige Werte im Datensatz vorliegen.

Der Nachteil der oben beschriebenen Vorgehensweise bei der Konvertierung von Parametern liegt darin, dass für jede sinnvolle Kombination von Parameter-Datentypen eine eigene Konvertierungsroutine erstellt werden muss. Deutlicht erleichtert werden kann dies durch die Einführung von „seriellen Konvertern“, bei denen mehrere Konverter nacheinander ausgeführt werden. Anstelle einen eigenen Konverter von `short` nach `double` zu erstellen, reicht es nun aus, den Konvertierungsweg mit Hilfe von vorhandenen Konvertierern zu beschreiben (z. B. von `short` über `integer` zu `double`).

Einen Sonderfall stellt noch die Konvertierung von Datensätzen dar. Hier ist zu beobachten, dass ein Datensatz eine Übermenge eines anderen darstellen kann. So ent-

[10] Eine Ausnahme bilden hier die Datenobjekte, welche direkt mit einem Eingabeparameter verbunden werden können.

hält ein Klassifizierungsdatensatz im Vergleich zu einem Deskriptordatensatz lediglich einen weiteren Vektor, welcher die Klassenzugehörigkeit angibt. Nun wäre es sehr umständlich, beispielsweise ein Modul zur Normalisierung von Deskriptorwerten mehrfach für alle Arten von Datensätzen anzubieten, die Deskriptoren enthalten. Eine Lösung des Problems ist möglich, wenn den einzelnen Datenelementen eines Datensatzes eine Bedeutung zugewiesen wird. Dadurch wird ein Vergleich zwischen zwei Datensätzen möglich. Es wird geprüft, ob die enthaltenen Datenelemente sowohl den gleichen Typ, Dimension und Kardinalität aufweisen, und ob zusätzlich auch die Bedeutung dieselbe ist. In diesem Fall lässt sich die Beziehung „A ist Untermenge von B“ überprüfen. Somit kann ein Klassifizierungsdatensatz gefahrlos in einen Deskriptordatensatz umgewandelt werden, in dem einfach der Vektor mit den Klassenzugehörigkeiten entfernt wird.

3.4 Spezialisierte Modultypen

Neben dem in Abschnitt 3.3 vorgestellten Standard-Modultyp sind in SOLVES noch zwei weitere, spezialisierte Modultypen vorhanden:

- Ein „Kontroll-Modul“ kann die Kontrolle über ein anderes Modul übernehmen und dessen Aufruf steuern.
- Ein „Makro-Modul“ enthält eine beliebige lineare Prozesskette und sorgt für den sequentiellen Aufruf der enthaltenen Module.

Besonders durch das Zusammenspiel dieser beiden Modultypen ergeben sich eine Reihe von neuen Möglichkeiten.

3.4.1 Das Kontroll-Modul

Der Name dieses Modultyps hätte auch „Meta-Modul“ lauten können, da es sich besonders gut zur Umsetzung aller Arten von Meta-Verfahren eignet.[11] Im folgenden werden die in SOLVES realisierten Einsatzmöglichkeiten aufgezählt, je nach Anforderungen des Anwendungsgebiets sind weitere Varianten denkbar.

Optimierung Das kontrollierte Modul gibt in diesem Fall die zu optimierende Fitnessfunktion vor. Da viele Optimierungsverfahren nur auf bestimmten Parametertypen arbeiten (wie reellen oder binärwertigen Zahlen), kommt der Kon-

[11] Als Meta-Verfahren werden hier Verfahren bezeichnet, die auf anderen Methoden arbeiten. Als anschauliches Beispiel sei die Meta-Optimierung genannt, die zur Optimierung der Parameter eines Optimierungsverfahrens dient.

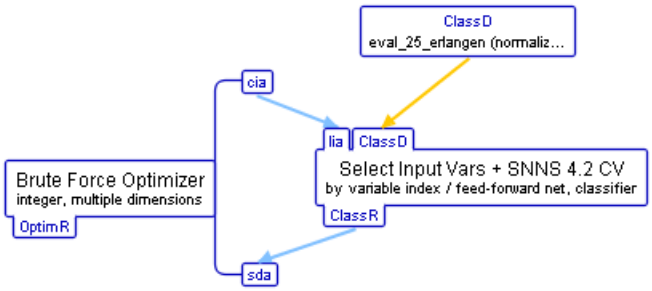

Abbildung 3.1: Darstellung eines Kontroll-Moduls im Workflow-Editor. Die Parametertypen werden abgekürzt angezeigt, über *tool tips* sind interaktiv weitere Informationen verfügbar.

vertierung der Ein- und Ausgabeparametern des kontrollierten Moduls hier eine besondere Bedeutung zu (siehe Abschnitt 3.3.8). Die Definition von Meta-Optimierungsverfahren ist problemlos möglich durch die Verwendung eines Optimierungsverfahrens als kontrolliertes Modul.

Validierung Für den Zweck der Validierung wird ein modellbildendes Verfahren (zur Klassifizierung oder Regression) als kontrolliertes Modul verwendet. Das Validierungsverfahren teilt den Eingabedatensatz nach einer festgelegten Vorgehensweise wiederholt in Trainings- und Testdaten auf. Beispiele sind die Kreuzvalidierung und das Bootstrap-Verfahren.

Modell-Verbesserung Die Leistung von modellbildenden Verfahren kann oft durch die Kombination von mehreren Modellen verbessert werden. Dies kann über ein Komitee (Abstimmung), über Boosting oder Bagging erfolgen.

Cache-Strategie Zwischen einem Optimierungsverfahren und der Berechnung der Fitnessfunktion kann ein Modul zur Speicherung bereits berechneter Werte eingefügt werden. Dies ist besonders bei aufwändigen Fitnessfunktionen von Vorteil. Allerdings ist die Verwendung eines Caches nur sinnvoll, wenn keine „echten“ (also nicht diskretisierte) reellwertigen Zahlen für die Kodierung des Individuums verwendet werden.

Datenbank-Pflege Über ein Kontrollmodul lässt sich die Pflege der Anwendungs-Datenbank vereinfachen. Dabei gibt das kontrollierte Modul eine Methode zur Berechnung einer Eigenschaft aus einer anderen an (z. B. die Berechnung von Deskriptorsätzen aus Molekülstrukturen). Bei Aufruf geht das Modul sämtliche in der Datenbank gespeicheren Objekte durch und aktualisiert gegebenenfalls die vorgegebene Zieleigenschaft.

Bearbeitung großer Dateien Die sequentielle Bearbeitung sehr großer Dateien, die

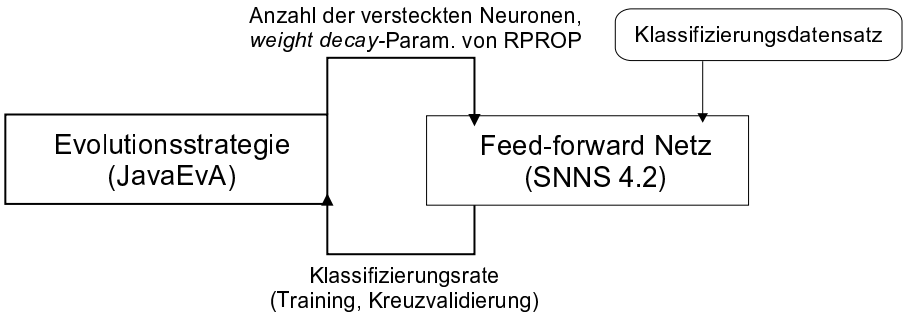

Abbildung 3.2: Schematische Darstellung eines Kontroll-Moduls. Die Verbindungspfeile zwischen den Modulen sind mit den Namen der übergebenen Parameter beschriftet.

nicht mehr als ganzes in den Hauptspeicher geladen werden können, lässt sich über ein Kontroll-Modul erledigen.

Erkennbar ist ein Kontroll-Modul an den in Form einer Klammer abgesetzten Parametern, welche die Verbindung zum kontrollierten Modul herstellen. In Abb. 3.1 ist die Darstellung eines Kontroll-Moduls im Rahmen der Benutzerschnittstelle gezeigt. Weitere Informationen, z. B. zu den zwischen den Modulen übergebenen Parametern, sind für den Benutzer interaktiv über *tool tips* erhältlich. Für nicht-interaktive Medien (z. B. in einer statischen Hilfefunktion oder in Handbüchern) ist dagegen die in Abb. 3.2 gezeigte schematische Art der Darstellung besser geeignet.

3.4.2 Das Makro-Modul

Bei der Ausführung eines Makro-Moduls spielt Geschwindigkeit eine größere Rolle als in der Haupt-Prozesskette. Wird ein Makro beispielsweise als Fitnessfunktion bei einer Optimierung eingesetzt, so wird es innerhalb kurzer Zeit sehr häufig aufgerufen werden. Daher verwenden Makro-Module einen anderen Mechanismus zur Steuerung der Abfolge der Module. Bei der Erzeugung eines Makros wird die gegebene Prozesskette analysiert, die Abfolge der Module festgelegt und die Verbindungen zwischen diesen „fest verdrahtet“.

3.5 Die graphische Benutzungsoberfläche

Die im Abschnitt 3.3 beschriebene Software-Architektur von SOLVES spiegelt sich in der graphischen Benutzerschnittstelle wieder. Sie besteht aus vier Dialogfenstern:

- Der Workflow-Editor dient zur Zusammenstellung von Prozessketten. Die Berechnungen können gestartet und das aktuelle Ergebnis betrachtet werden.
  → `WorkflowServer`

- Der Archivierungsteil erlaubt die Betrachtung bisher erzielter Ergebnisse. Diese können dauerhaft archiviert oder aus dem Archiv geholt werden.
  → `ArchiveServer`

- Die Eigenschafts-Datenbank ermöglicht die Verwaltung von Eigenschaften der untersuchten Objekte (z. B. Moleküle). Auf diesen können Filter definiert werden.
  → `ApplicDomainServer`

- Der Modul-Manager gibt einen Überblick über die aktuell vorhandenen Module und zeigt automatisch generierte Dokumentationsseiten an.
  → `ServerManager`

Hinter dem Pfeil ist jeweils der dem Dialogfenster zugeordnete spezialisierte Server angegeben. Zudem gibt es noch ein kleineres Hauptfenster, in dem übergreifende Einstellungen (wie Projekt, Benutzername) vorgenommen werden, und in dem Mitteilungen des Systems dargestellt werden. Da die Benutzerschnittstelle für ein auf den Praktiker als Anwender gerichtetes Programm eine große Rolle spielt, werden die einzelnen Bestandteile im folgenden noch detaillierter beschrieben.

3.5.1 Der Workflow-Editor

In der oberen Hälfte des Dialogfensters (vgl. Abb. 3.3) werden Prozessketten editiert. Das Einfügen von Modulen ist über die Menüleiste, über ein Popup-Menü oder über *drag-and-drop* aus dem Modul-Manager-Dialog möglich. Module können mit der Maus selektiert und ihre Position verschoben werden. Datensätze können entsprechend über das Menü eingefügt oder aus dem Archiv-Dialog herübergezogen werden. Zur Erstellung einer Verbindung wird auf einen Parameter geklickt und über dem Verbindungspartner (anderer Parameter oder Datensatz) die Maus wieder losgelassen. Es können Kommentare im HTML-Format eingefügt werden.

Wird eines der Bestandteile einer Prozesskette ausgewählt, so erscheinen im unteren Teil des Dialogfensters jeweils passende Eingabe- oder Abfragedialogelemente. Bei einem Modul werden die aktuellen Werte der Parameter angezeigt. Wird ein Parameter ausgewählt, so kann er vom Benutzer editiert werden. Zudem wird die automatisch generierte Dokumentation angezeigt. Bei einem Datensatz wird entweder ein spezialisierter Viewer oder die Liste der enthaltenen Datenelemente dargestellt. Wird ein HTML-Kommentar angewählt, so kann dessen Quelltext editiert werden.

Beim Start einer Prozesskette gibt es die folgenden Möglichkeiten:

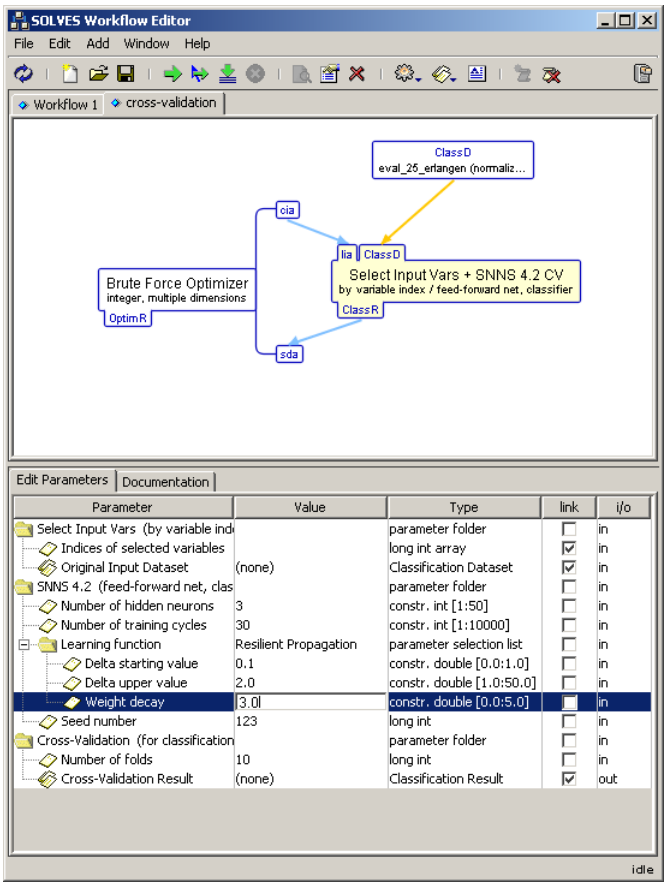

Abbildung 3.3: Das Dialogfenster des Workflow-Editors. Der obere Teil dient zur Zusammenstellung von Prozessketten. Unten können Parameter editiert oder das aktuelle Ergebnis betrachtet werden.

- Sofortiger Start der gesamten Prozesskette.
- Start der ausgewählten Module (also eines Teiles der Prozesskette).
- Ablage der Prozesskette in der Datenbank zur späteren, automatisierten Ausführung.

3.5.2 Der Archivierungsteil

Im Archiv werden die Ergebnisse früherer Berechnungen abgespeichert. Pro Berechnung wird dabei ein neuer Container generiert. Dieser wird nach Projektzugehörigkeit einsortiert. Innerhalb eines Projekts erfolgt eine weitere Unterteilung in Sektionen und

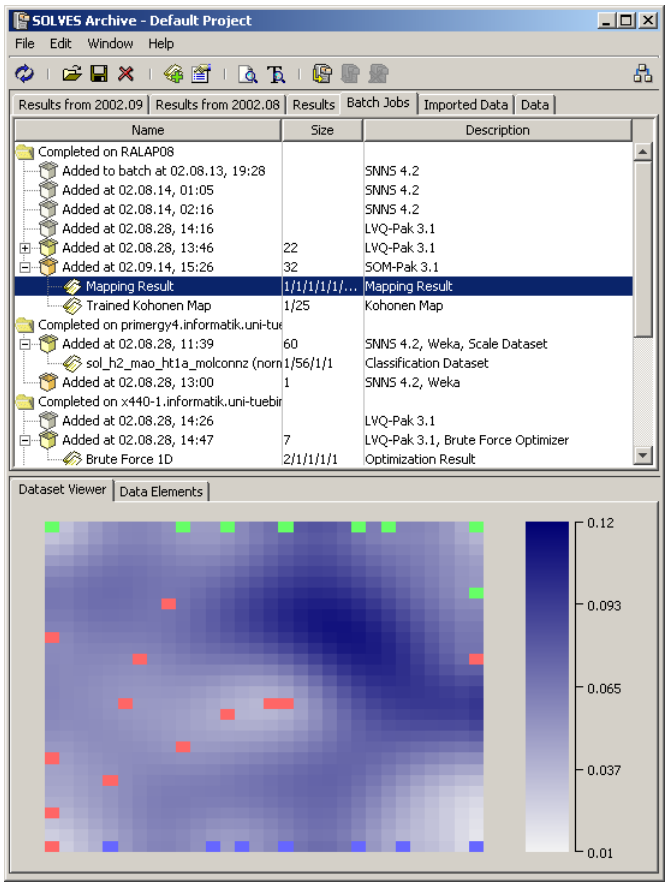

Abbildung 3.4: Das Dialogfenster des Archivierungsteils.

Ordner. Diese Zuordnung kann vom Benutzer geändert werden, so dass z. B. gute und schlechte Ergebnisse getrennt einsortiert werden können.

Im oberen Teil des Archiv-Dialogfensters (Abb. 3.4) sind die in einer Sektion vorhandenen Container zusammen mit den darin enthaltenen Datensätzen sichtbar (die anderen im Projekt vorhandenen Sektionen sind über die Kartenreiter zugänglich). Wird ein Container oder ein Datensatz ausgewählt, so erscheint im unteren Teil entweder die Liste der Datenelemente oder (sofern vorhanden) ein spezialisierter Viewer.

Der Archivierungszustand eines Containers wird über seine Farbe angezeigt. Dabei steht orange für einen Container, der noch nicht dauerhaft archiviert wurde. Ein archivierter und noch im Hauptspeicher vorhandener Container wird gelb dargestellt, ein nur in der Datenbank vorhandener Container dagegen grau.

Über die Import-Funktion können Daten aus unterschiedlichen Dateiformaten eingelesen werden. Zu den unterstützten Dateiformaten zählen allgemeine (wie das flache Textformat) und spezialisierte Formate (wie das *structured data file* (SDF) Molekülformat). Je nach Datenformat werden beim Import auch gleich Datensätze angelegt.

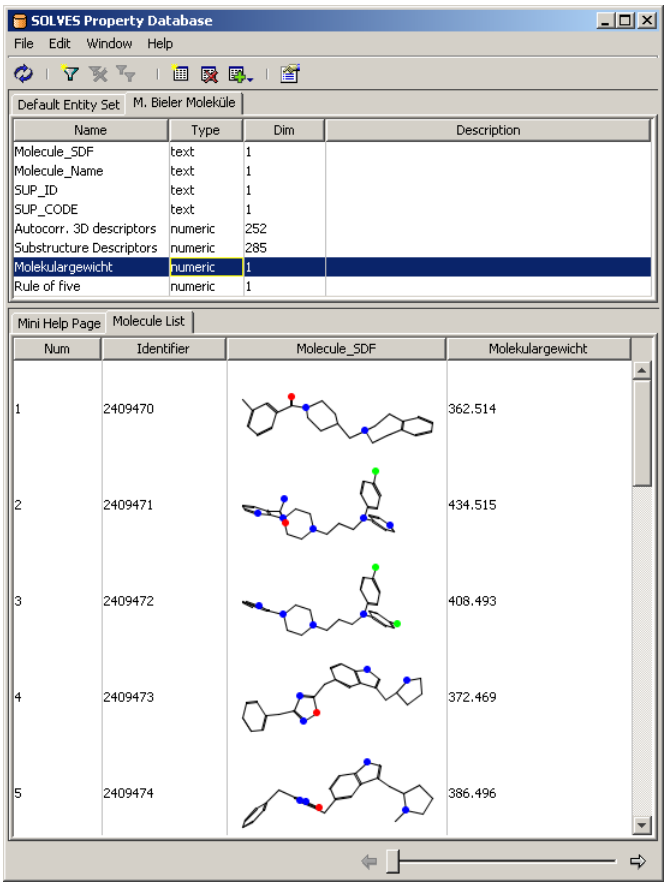

Abbildung 3.5: Das Dialogfenster der Eigenschafts-Datenbank.

3.5.3 Die Eigenschafts-Datenbank

Je nach Anwendungsgebiet ist ein separates Abspeichern der zu den untersuchten Objekten gehörigen Eigenschaften sinnvoll. Dabei kann es sich um folgende Eigenschaften handeln:

- Informationen zur graphischen Darstellung der untersuchten Objekte (z. B. die Molekülstruktur im SDF-Format).
- Angaben zu der Herkunft oder der Entstehung der Objekte (z. B. der Name der Zulieferfirma).
- Aus den Objekten berechnete numerische Eigenschaften (z. B. rechenzeitaufwändige Arten von Deskriptoren).
- Aus den Objekten gewonnene Messwerte (z. B. mit Hilfe von HTS-Läufen gemessene IC_{50}-Werte).

Die Angaben in Klammern beziehen sich jeweils auf den Fall, dass es sich bei den untersuchten Objekten um Moleküle handelt. Besonders bei großen Moleküldatenbanken ist es nicht sinnvoll, die zahlreichen Deskriptoren jedesmal neu zu berechnen. Eine Ablage jedes einzelnen Deskriptors in einer eigenen Tabellenspalte würde den Zugriff aber zu langsam machen. Daher können neben einzelnen numerischen Werten auch ganze Vektoren abgelegt werden, wobei eine Kompression des benötigten Platzes optional möglich ist.

Im oberen Teil der Eigenschafts-Datenbank (Abb. 3.5) sind die zur Verfügung stehenden Eigenschaften mit Datentyp (`Text`, `Numeric` oder `Date`) und Dimension aufgelistet. Wird eine Eigenschaft ausgewählt, so wird diese im unteren Teil in der letzten Spalte angezeigt. Daneben werden die Identifikationsnummer und (sofern vorhanden) eine graphische Repräsentation des untersuchten Objekts gezeigt.

Auf der ausgewählten Eigenschaft können abhängig vom Datentyp Filter definiert werden. Beim Datentyp `Text` wird eine Auswahlliste gezeigt, bei numerischen Eigenschaften kann ein Bereich mit den Vergleichsoperatoren $<, \leq, >$ und $\geq$ definiert werden. Filter lassen sich mit den logischen Operatoren `AND`, `OR` und `NOR` kombinieren. Aus mehreren Filtern kann schließlich ein neuer Klassifizierungsdatensatz generiert werden.

3.5.4 Der Modul-Manager

Informationen über die in SOLVES vorhandenen Module werden dem Benutzer über das Dialogfenster des Modul-Managers zur Verfügung gestellt (Abb. 3.6). Der obere Teil des Dialogfensters bietet einen nach Anwendungsgebieten geordneten Überblick an. Ein Modul kann dabei mehreren Anwendungsgebieten zugeordnet sein, wenn es entsprechende Funktionalitäten (in Form sogenannter *usage modes*) aufweist. Die Einordnung wird dabei vom Modul selber vorgenommen. Die Art und Anzahl der dargestellten Anwendungsgebiete wird somit nicht von SOLVES festgelegt, sondern ist dynamisch erweiterbar. Der Status eines Moduls ist über das zugehörige Icon erkennbar. Dies ist insbesondere dann nützlich, wenn das Modul aus irgendeinem Grund nicht aktiv (also nicht einsetzbar) ist. In einem verteilten Programmsystem kann ein Modul aus mehreren Gründen lang- oder kurzfristig inaktiv sein (z. B. fehlgeschlagene Initialisierung, maximale Auslastung des Moduls erreicht, Störung der Verbindung zum Host-Rechner des Moduls).

Im unteren Teil des Dialogfensters wird zum einen die automatisch generierte Dokumentation des oben ausgewählten Moduls angezeigt. Jedes Modul stellt eine Reihe von Information zur Verfügung (z. B. Autor, Versionsnummer, Hinweise, Internet-Links), die vom Modul-Manager in ein einheitliches Layout transformiert werden. Im Gegensatz zu statischen Hilfeseiten kann durch diesen Mechanismus die Aktualität der zu einem Modul angezeigten Information gewährleistet werden. Zum anderen werden in

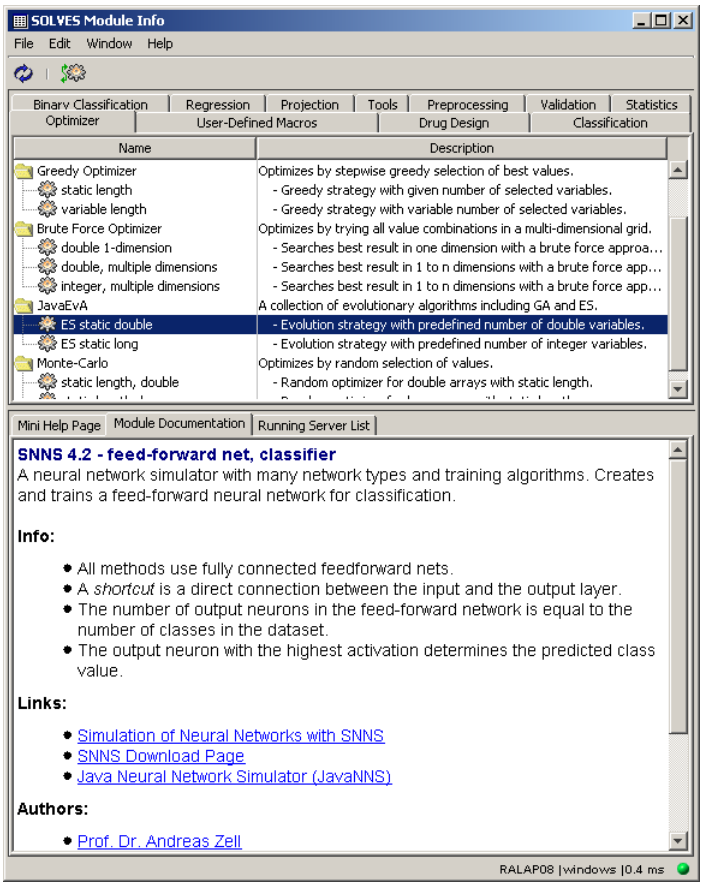

Abbildung 3.6: Das Dialogfenster des Modul-Managers.

einer tabellarischen Darstellung detailliertere Informationen zu den aktuell verfügbaren Modulen gezeigt. Dabei handelt es sich beispielsweise um die Anzahl der derzeit allokierten Instanzen, die Art der Implementierung und die Fähigkeit des Moduls zum Multithreading.

Zusätzlich werden in der Statusleiste noch Informationen über die Verbindung zum SOLVES-Server angezeigt.[12] Es wird der Name und das Betriebssystem des Servers, die Geschwindigkeit der Verbindung und der allgemeine Status (als grüner oder roter Punkt) angezeigt, über *tool tips* sind weitere Details erhältlich.

[12] Im Standalone-Modus von SOLVES laufen der Client- und der Serverteil von SOLVES in derselben Programminstanz ab.

3.6 In SOLVES eingebundene Module

Die Funktionalität des vorgestellten Programmsystems SOLVES hängt wesentlich von der Art und der Anzahl der eingebundenen Module ab. Zur Dokumentation des zum Zeitpunkt der Erstellung dieser Ausarbeitung dem Benutzer zur Verfügung stehenden Funktionsumfanges werden die Module im folgenden aufgelistet und beschrieben.

Neben einigen vollständig selbstgeschriebenen Modulen[13] handelt es sich dabei um eine Anzahl von frei verfügbaren[14] und kommerziellen Programmen. Dadurch soll der Anspruch von SOLVES, eine breite Palette von Programmen einbinden zu können, untermauert werden.

3.6.1 Einteilung der Module

Eine direkte Einteilung der Module z. B. nach ihrem Anwendungsgebiet ist in den meisten Fällen schwierig, da jedes Modul unterschiedliche Funktionalitäten in Form von sogenannten *usage modes* (Benutzungsarten bzw. Services) anbieten kann. Daher werden in Tabelle 3.2 eine Reihe von Kurzbezeichnungen definiert, welche im folgenden jeweils mit angegeben werden. Diese wurden bewusst so ausgewählt, dass sie sich teilweise überlappen (z. B. „Optimierung“ und „Evolutionäre Algorithmen“), um eine feinere Unterteilung zu ermöglichen.

Kürzel	Bedeutung	Kürzel	Bedeutung
OPT	Optimierung	EA	Evolutionäre Algorithmen
CLA	Klassifizierung	NN	Neuronale Netze
BCLA	Binäre Klassifizierung	SVM	Support Vector Machines
REG	Regression	SL	Statistische Lernverfahren
PRO	Projektion	STAT	Statistik
VAL	Validierung	CI	Chemoinformatik
VOR	Vorverarbeitung	MOL	Verarbeitung von Molekülen

Tabelle 3.2: Bedeutung der Kurzbezeichnungen, welche zur Einteilung der nachfolgend aufgelisteten Module eingesetzt werden.

[13] Diese sind im folgenden jeweils mit einem Stern ($*$) markiert.

[14] Wobei zu beachten ist, dass bei „freier Software“ der Grad der Freiheit von den jeweiligen Lizenzbestimmungen abhängt. Neben der häufig anzutreffenden *GNU General Public License* (GPL) sind eine Vielzahl oft nur graduell unterschiedlicher Bestimmungen im Umlauf.

3.6.2 Beschreibung der Module

Im folgenden sind die Module in alphabetischer Reihenfolge angeordnet. Eine nach Anwendungsgebieten geordnete Auflistung ist im nächsten Abschnitt zu finden.

Evolutionäre Algorithmen in Java (JavaEvA) EA OPT

Das an der Uni Tübingen im Arbeitskreis von Prof. Zell entwickelte Programmpaket enthält folgende Arten von Evolutionären Algorithmen (vgl. Abs. 2.2):

- **Evolutionsstrategien (ES):** Diese sind besonders zur Lösung von reellwertigen Optimierungsproblemen geeignet, wobei Strategieparameter wie die Mutationsrichtung und -schrittweite ebenfalls einer evolutionären Anpassung unterworfen sind.
- **Genetische Algorithmen (GA):** Die potentiellen Lösungen eines Optimierungsproblems werden in eine binäre Kodierung (*bitstring*) transformiert, das Verfahren arbeitet also auf einer genotypischen Darstellung des Individuums.[15]
- **Genetische Optimierung (GO):** Bei diesem Verfahren kann sich der Gencode eines Individuums aus unterschiedlichen Typen zusammensetzen, für die jeweils passende Mutations- und Crossover-Operatoren gewählt werden.[16]

Evolutionary Computation in Java (ECJ) EA OPT

Dieses Programmpaket enthält eine Umsetzung des streng-typisierten Genetischen Programmierens (STGP). Die beteiligten Java-Klassen werden erst zur Laufzeit geladen, wodurch die Flexibilität erhöht wird. Daneben sind in ECJ auch eher einfach gehaltene Implementierungen von GA und ES enthalten. ECJ wurde von Sean Luke an der George Mason-Universität (Virginia/Washington DC) entwickelt.

Grundlegende Optimierungsverfahren OPT $*$

Einige einfache, aber grundlegende Optimierungsverfahren wurden in SOLVES implementiert. Dazu zählen Hill-Climbing und Monte-Carlo. Diese beiden Verfahren ermöglichen einen Vergleich mit komplexeren Verfahren (wie GA oder ES). Daneben steht ein Brute Force-Ansatz zur Verfügung. Für kleine Datensätze kann hiermit über vollständige Enumeration eine obere Schranke für den Vergleich von Optimierungsverfahren ermittelt werden.

LIBSVM SVM BCLA CLA

Diese *Support Vector Machines*-Bibliothek wurde von Chih-Chung Chang und

[15] GAs werden in JavaEvA als eine auf den Typ BitString beschränkte Variante von GO umgesetzt.
[16] Das GO-Verfahren wurde von Felix Streichert an der Uni Tübingen entwickelt.

Chih-Jen Lin an der National Taiwan University in Taipei entwickelt. Sie enthält verschiedene Typen von SVM und lineare, polynomielle, radial-basische und sigmoide Kernelarten [CL02]. Eine Klassifizierung von mehr als zwei Klassen wird über eine „1-gegen-1“-Strategie ermöglicht.

LVQ_PAK [CLA]

Eine Implementierung des *Learning Vector Quantization* (LVQ) Verfahrens. Das Programm wurde von Teuvo Kohonen an der Helsinki University of Technology in Finnland entwickelt [KHK+96].

Principal Component Analysis (PCA) [PRO]

Dieses Modul berechnet eine Hauptkomponentenanalyse auf den Eingabevariablen eines Datensatzes. Über den Anteil der erklärten Varianz lassen sich die ersten n Hauptkomponenten auswählen und in einen neuen Regressions- oder Klassifizierungsdatensatz überführen. Die Berechnungen werden mit Hilfe der JAMA Java-Bibliothek durchgeführt.[17]

SOM_PAK [PRO]

Eine Implementierung des *Self-Organizing Map* (SOM) Verfahrens, welches auch als „Kohonenkarte“ bezeichnet wird [KHKL96]. Das Programm wurde von Teuvo Kohonen an der Helsinki University of Technology in Finnland entwickelt.

Statistische Deskriptorbewertung [STAT]

Eine statistische Bewertung von Eingabevariablen wird mit dem Statistik-Modul ermöglicht. Es wird eine Kovarianzmatrix, der Mittelwert und die Varianz von Deskriptoren sowie ein Maß für das Auftreten von gleichen Werten berechnet.[18]

STEEDS [CI] $*$

Das Akronym STEEDS steht für *SMARTS Tree Encoding for the Evolution of Descriptor Sets*. Das Verfahren dient zur automatischen Generierung und Optimierung von Substruktur-Deskriptoren unter Einsatz des streng-typisierten Genetischen Programmierens.[19]

Stuttgarter Neuronale Netze Simulator (SNNS) [NN] [REG] [CLA] [BCLA]

Das SNNS-Programmpaket wurde an der Uni Stuttgart unter Anleitung von Prof. Zell entwickelt. Es enthält über 30 verschiedene Trainingsverfahren, die auf unterschiedlichen Netztopologien arbeiten. In dem SNNS-Modul werden dem Benutzer folgende Verfahren zur Verfügung gestellt (vgl. Abschnitt 2.1):

[17] Die erste Version des PCA-Moduls wurde von Jörg Wegner an der Uni Tübingen implementiert.

[18] Das Statistik-Modul wurde von Boris Hollas an der Universität Tübingen in C++ implementiert.

[19] Die in SOLVES enthaltene Implementierung von STEEDS basiert auf den in diesem Abschnitt beschriebenen Java-Bibliotheken ECJ und JOELib.

- **Multi Layer Perceptrons (MLP):** Der in der Praxis am häufigsten eingesetzte Netztyp ist auch unter der Bezeichnung *feed-forward* Netze bekannt. Als Lernverfahren können Standard-Backpropagation, Resilient Propagation (RProp), Scaled Conjugate Gradient (SCG), Quickprop, Weight-Decay-Backprop, Backprop with Momentum und Batch-Backprogation eingesetzt werden.
- **Radiale Basis-Funktionen (RBF):** Der Vorteil der radialen Basis-Funktionen liegt darin, dass sie außerhalb des trainierten Bereiches ein besser vorhersagbares Verhalten haben als MLP (die Ausgabe fällt langsam auf Null ab).

Vorverarbeitung und Validierung VOR VAL $*$

Einen wichtigen Aspekt für den Einsatz und Vergleich von Verfahren zur Datenauswertung stellen die Vorverarbeitung der Daten (durch Skalierung oder Normierung) und die Validierung der Ergebnisse (z. B. über einen Testdatensatz oder eine Kreuzvalidierung) dar. Beides wird in SOLVES durch entsprechende Module und Methoden unterstützt.

Weka SL SVM NN REG CLA BCLA

Das *Waikato Environment for Knowledge Analysis* umfasst eine Vielzahl von Data Mining-Verfahren. Es wurde an der Universität von Waikato in Neuseeland entwickelt. In Weka sind unter anderem Support Vector Machines, das Entscheidungsbaum-Verfahren C4.5, Naive Bayes, k-Nearest Neighbors, PART, KStar und einfache Klassifizierungsverfahren wie Majority Predictor und OneRule enthalten. Weka ist unter die GNU General Public License (GPL) gestellt.

Anwendungsgebiet	Modul	Methoden
Regression	Weka	Lineare Regression
	ECJ	Symbolische Regression
	SNNS 4.2	Feedforward-Netz (MLP)
	LIBSVM	Support Vector Regression (SVR)
Projektion	SOM_PAK	Training von Kohonenkarten
		Abfrage von Kohonenkarten
	PCA	Hauptkomponentenanalyse
		Extraktion der Hauptkomponenten

Tabelle 3.3: Eine Einordnung der in SOLVES verfügbaren Module nach Anwendungsgebiet (erster Teil).

3.6.3 Tabellarische Einordnung der Module

In den Tabellen 3.3, 3.4 und 3.5 sind die in SOLVES eingebundenen Module noch einmal nach Anwendungsgebieten geordnet dargestellt. Die „binäre Klassifizierung“ stellt einen Sonderfall dar, bei dem genau zwei Klassen klassifiziert werden sollen. Die Kohonenkarten (SOM) werden in SOLVES den Projektionsverfahren zugeordnet, da die auf einer solchen Karte vorhandenen Cluster meist nicht automatisch, sondern durch visuelle Inspektion durch den Benutzer eingeteilt und bewertet werden.

Anwendungsgebiet	Modul	Methoden
Klassifizierung	SNNS 4.2	Feedforward-Netz (MLP) mit n Ausgabeneuronen
		Radiale Basis-Funktionen (RBF)
	Weka	Entscheidungsbäume (C4.5)
		Naive Bayes
		K-Nearest Neighbors
		Majority Predictor, 1-Rule only
		KStar, PART, Decision Table
	LVQ_PAK	Training der Codebook-Vektoren
		Abfrage von trainierten CBVs
	LIBSVM	Support Vector Machine (SVM) mit „1-gegen-1“-Strategie
Binäre Klassifizierung	SNNS 4.2	Feedforward-Netz (MLP) mit einem Ausgabeneuron
	Weka	Support Vector Machine (SVM)
Optimierung	JavaEvA	Evolutionsstrategie (ES)
		Genetischer Algorithmus (GA)
	Hill-Climbing	Genetische Optimierung (GO)
		für reellwertige Zahlen
		für ganze Zahlen (mit Cache)
	Monte-Carlo	für reellwertige Zahlen
		für ganze Zahlen (mit Cache)
	Brute Force	für reellwertige Zahlen
		für ganze Zahlen (mit Cache)
	Greedy-Auswahl	statisch vorgegebene Länge
		variabel angepasste Länge
Statistik	für Deskriptoren	Berechnet Korrelationsmatrix, Mittelwert und Varianz

Tabelle 3.4: Eine Einordnung der in SOLVES verfügbaren Module nach Anwendungsgebiet (zweiter Teil).

Anwendungsgebiet	Modul	Methoden
Vorverarbeitung	Normierung	Skaliert die Eingabevariablen
	Selektion	Selektiert Eingabevariablen
Validierung	Aufteilung	in Trainings- und Testdaten
	Kreuzvalidierung	Interne Validierungsläufe
	Testfunktionen	Teilweise Implementierung der EA-Testfunktionen F1 – F24
Hilfsmittel	für Datensätze	Vereinigt separate Trainings- und Testdatensätze

Tabelle 3.5: Eine Einordnung der in SOLVES verfügbaren Module nach Anwendungsgebiet (dritter Teil).

3.7 Das XML-Format `SolvesML`

Zur Speicherung von Eingabedaten, Ergebnissen und Prozessketten verwendet SOLVES ein eigenes XML-Format namens `SolvesML`. Dieses soll im folgenden erläutert und anhand seiner *Document Type Definition* (DTD) spezifiziert werden.

Wie in Abschnitt 3.3 dargelegt, werden in SOLVES sämtliche Daten mit Hilfe von Containern verwaltet. Diese enthalten Datenelemente, die über Datensätze gruppiert und in einen Sinnzusammenhang gebracht werden. Dies spiegelt sich auch in der in Abbildung 3.7 gezeigten Haupt-DTD von SOLVES wieder. Innerhalb eines `Solves`-Tags stehen ein oder mehrere `DataContainer`-Tags. In jeder `SolvesML`-Datei muss also mindestens ein Container vorhanden sein, wobei jeder Container eine beliebige Anzahl von Datenelementen enthalten kann. Jedes Datenelement hat einen festgelegten Datentyp (z. B. `double`) und eine Datenstruktur (z. B. `vector` oder `matrix`). Es enthält zudem eine eindeutige Identifikationsnummer (`uid`), welche eine verwechslungsfreie Zuordnung erlaubt. Der Inhalt eines Datenelements ist unabhängig von der Datenstruktur stets als Liste von `Item`-Tags angeordnet.[20] Für Datenstrukturen, die sich nicht ohne weiteres auf eine Listenstruktur abbilden lassen (z. B. ein `graph`), muss jeweils eine Unter-DTD definiert werden, welche den Inhalt der `Item`-Tags in diesem Fall spezifiziert (vgl. Abb. 3.8).

Neben Datenelementen enthalten Container auch Datensätze, welche mit dem `DatasetObject`-Tag spezifiziert werden. Jeder Datensatz enthält eine oder mehrere Listen, die neben dem Datentyp und -struktur auch eine Kardinalität festlegen (z. B. `single` oder `multiple`). Diese Listen enthalten wiederum Referenzen auf Datenelemente. Die referenzierten Datenelemente können im selben oder auch in einem anderen Datencontainer enthalten sein. Zwischen internen und externen Referenzen wird also nicht

[20] Aus Gründen der Speicherplatz- und Rechenzeitersparnis bei großen Datensätzen können einfache Datentypen (wie `double` oder `integer`) auch direkt als Leerzeichen-getrennte Zeichenfolge unter Umgehung der `Item`-Tags abgelegt werden.

```
<!ELEMENT Solves ( DataContainer+ )>
<!ELEMENT DataContainer ( DataElement*, DatasetObject* )>
<!ATTLIST DataContainer
  uid            CDATA #REQUIRED <!-- long integer -->
>
<!ELEMENT DataElement ( Item* | #PCDATA )>
<!ATTLIST DataElement
  name           CDATA #REQUIRED <!-- string -->
  type dtype     ( double | boolean | short int | long int
                 | string | workflow ) #REQUIRED
  structure str  ( scalar | vector
                 | matrix | graph ) #REQUIRED
  uid            CDATA #REQUIRED <!-- long integer -->
  size           CDATA #IMPLIED  <!-- integer -->
>
<!ELEMENT Item ( ANY )>
<!ELEMENT DatasetObject ( DatasetList+, DatasetProperty* )>
<!ATTLIST DatasetObject
  type           CDATA #REQUIRED <!-- string -->
  description    CDATA #REQUIRED <!-- string -->
  uid            CDATA #REQUIRED <!-- long integer -->
  size           CDATA #IMPLIED  <!-- integer -->
>
<!ELEMENT DatasetList ( DataElementRef* )>
<!ATTLIST DatasetList
  purpose        CDATA #REQUIRED <!-- string -->
  group          CDATA #REQUIRED <!-- string -->
  type dtype     ( double | boolean | short int | long int
                 | string | workflow ) #REQUIRED
  structure str  ( scalar | vector
                 | matrix | graph ) #REQUIRED
  cardinality ca ( single | multiple
                 | optional | undefined ) "undefined"
  size           CDATA #IMPLIED  <!-- integer -->
>
<!ELEMENT DataElementRef EMPTY>
<!ATTLIST DataElementRef
  uid            CDATA #REQUIRED <!-- long integer -->
<!ELEMENT DatasetProperty ( #PCDATA )>
<!ATTLIST DatasetProperty
  name           CDATA #REQUIRED <!-- string -->
>
```

Abbildung 3.7: Die Haupt-DTD von `SolvesML`. Für spezielle Datentypen (wie z. B. Prozessketten) können innerhalb des `Item`-Tags weitere Unter-DTDs definiert werden.

```
<!ELEMENT Item ( Node*, Connection* )>
<!ELEMENT Node ( DatasetRef | Module | #PCDATA )?>
<!ATTLIST Node
  type content ( data | module | text ) #REQUIRED
  posX           CDATA #REQUIRED <!-- integer -->
  posY           CDATA #REQUIRED <!-- integer -->
  handle         CDATA #REQUIRED <!-- integer -->
>
<!ELEMENT DatasetRef EMPTY>
<!ATTLIST DatasetRef
  type           CDATA #REQUIRED <!-- string -->
  uid            CDATA #REQUIRED <!-- long integer -->
  projectName    CDATA #REQUIRED <!-- string -->
  containerUID   CDATA #REQUIRED <!-- long integer -->
>
<!ELEMENT Module ( Parameter+ )>
<!ATTLIST Module
  serverName     CDATA #REQUIRED <!-- string -->
  implementation CDATA #REQUIRED <!-- string -->
  version        CDATA #REQUIRED <!-- integer -->
  usageMode      CDATA #REQUIRED <!-- string -->
>
<!ELEMENT Parameter ( #PCDATA )>
<!ATTLIST Parameter
  name           CDATA #REQUIRED <!-- string -->
  linkable flag ( true | false ) "false"
  relationName   CDATA #IMPLIED  <!-- string -->
>
<!ELEMENT Connection EMPTY>
<!ATTLIST Connection
  handle1        CDATA #REQUIRED <!-- integer -->
  handle2        CDATA #REQUIRED <!-- integer -->
  info1          CDATA #REQUIRED <!-- string -->
  info2          CDATA #REQUIRED <!-- string -->
  convertInfo    CDATA #IMPLIED  <!-- string -->
>
```

Abbildung 3.8: Die DTD einer Prozesskette innerhalb des `Item`-Tags von `SolvesML`.

explizit unterschieden.[21] Schließlich können Datensätze optional noch eine Menge von Properties enthalten, die aus einfachen Name-Wert-Paaren bestehen. Der Name wird dabei als Attribut angegeben, den Wert stellen die von dem `DatasetProperty`-Tag umschlossenen Zeichen dar.

Die in Abb. 3.8 dargestellte DTD spezifiziert eine Prozesskette, welche innerhalb der `Item`-Tags eines Daten-Elementes mit der Struktur `graph` und dem Typ `workflow` auftreten kann. Sie besteht aus den Tags `Node` und `Connection`, welche die Knoten bzw. Kanten des Graphen definieren. In den Knoten einer Prozesskette können Module, Datensätze und Kommentare enthalten sein. Ist ein Modul enthalten, so werden mit einem oder mehreren `Parameter`-Tags entsprechend die Parameter definiert. Auf Datensätze wird mit Hilfe des `DatasetRef`-Tags referenziert. Kommentare werden direkt über den von dem `Node`-Tag umschlossenen Text definiert, es müssen in diesem Fall also keine weiteren Tags angegeben werden.

3.8 Verteilung von Berechnungen

Durch den Einsatz eines verteilten Objektmodells und die Aufteilung der eingesetzten Methoden in voneinander unabhängige Module wird eine Verteilung der bei einer Berechnung auftretenden Rechenlast möglich. In diesem Abschnitt werden die verschiedenen in SOLVES eingebauten Verteilungsmodelle beschrieben.

Die kleinste Einheit der Verteilung liegt dabei auf der Ebene der „einfachen“ Module, also der Module, die keine anderen Module kontrollieren. Ist diese erreicht, kann vom Server-Manager keine weitere Aufteilung erreicht werden. Vom Modul selber kann mit Hilfe von Threads eine weitere Verteilung erzielt werden, welche allerdings nur auf einem Multiprozessorsystem Auswirkungen zeigt.

3.8.1 Verteilung auf zwei Rechner (Client / Server-Modus)

Wird SOLVES im Client-Modus gestartet, so erfolgt automatisch eine Suche nach geeigneten Server-Instanzen. Diese werden dem Benutzer zur Auswahl angeboten. Entscheidet der Benutzer sich für einen der Server, so kann er wie gewohnt mit der Benutzungsoberfläche arbeiten und auch den aktuellen Stand der Berechnungen mitverfolgen. Alle Berechnungen werden jedoch für ihn transparent auf dem Server durchgeführt. Dies ist insbesondere bei großen Datensätzen vorteilhaft, da der Server über mehr Rechenleistung oder über einen größeren Hauptspeicher verfügen kann.

[21] Daher können Container beispielsweise auch nur Datensätze, aber keine eigenen Datenelemente enthalten.

3.8.2 Verteilung auf viele Rechner (Server Pool-Modus)

Bei besonders komplexen Berechnungen kann eine gleichzeitige Verteilung auf mehrere Server-Instanzen einen Vorteil erbringen. Der Server-Manager stellt dabei absichtlich Module zusammen, die auf unterschiedlichen Rechnern laufen. Allerdings ist hierbei der durch die Übertragung der Datensätze entstehende Overhead zu berücksichtigen, so dass der tatsächliche Geschwindigkeitsgewinn geringer als erwartet ausfallen kann.

3.8.3 Keine Verteilung (Standalone-Modus)

Startet man SOLVES im Client-Modus, ohne dass eine Verbindung zu einer im Server-Modus laufenden Instanz aufgebaut wird, so übernimmt SOLVES auch die Server-Rolle. In diesem Fall stehen dann nur die lokal installierten Module zur Verfügung, und sämtliche Berechnungen werden auf dem lokalen Rechner durchgeführt. Ein Abspeichern der Ergebnisse in der Datenbank ist nach wie vor möglich.

Im Sinne der einfachen Anwendbarkeit von SOLVES ist ein solcher Standalone-Modus sinnvoll. Will ein Benutzer nur eine kleinere Berechnung durchführen oder die im Batch-Modus (siehe unten) erzielten Ergebnisse betrachten, dann ist eine Aufteilung in separate Client- und Server-Teile nicht notwendig.

3.8.4 Verteilung der spezialisierten Server

Es ist nicht notwendig, dass sich die spezialisierten Server (wie das Archiv oder der anwendungsbezogene Server) auf dem gleichen Rechner wie die jeweilige SOLVES-Instanz befinden. Prinzipiell können zudem auch mehrere SOLVES-Instanzen einen dieser Server gemeinsam benutzen. Dies kann z. B. beim Archiv von Vorteil sein, wenn mehrere Rechner mit geringem Hauptspeicherplatz über eine schnelle Datenleitung verbunden sind.

3.8.5 Synchrone vs. asynchrone Verteilung (Batch-Modus)

Bisher wurde implizit von einer synchronen Verteilung der Berechnungen ausgegangen, d. h. der Benutzer startet eine Berechnung, wartet bis sie abgeschlossen ist und betrachtet dann sofort das Ergebnis. Gerade bei langwierigen Berechnungen ist jedoch ein Ablauf der Berechnungen im Hintergrund wünschenswert. Der Benutzer definiert dabei einen oder mehrere Workflows, legt diese in der Datenbank ab und betrachtet zu einem späteren Zeitpunkt die berechneten Ergebnisse. In SOLVES prüfen im Server-Modus gestartete Instanzen, sofern sie gerade nicht mit einem Client verbunden sind, in regelmässigen Abständen, ob gerade ein freier (sich also noch nicht in Bearbeitung

befindlicher) Batch-Job vorhanden ist. Wenn dies der Fall ist, so wird er übernommen und abgearbeitet, das Ergebnis wird im Archiv abgelegt.

3.9 Vergleich mit existierenden Systemen

Um die Eigenständigkeit des in dieser Arbeit verfolgten Ansatzes zu belegen, werden im folgenden Systeme[22] mit im weitesten Sinne ähnlichen Eigenschaften aufgeführt und beschrieben. Eine Gegenüberstellung mit den Eigenschaften des SOLVES Programmsystems erfolgt in Form einer Tabelle, wodurch ein direkter Vergleich ermöglicht wird. Ein vorläufiges Fazit für diesen Vergleich ist in Abschnitt 3.10 zu finden.

Da ein Vergleich mit sämtlichen in Frage kommenden Systemen mit einem hohen Aufwand bei nur geringem Informationsgewinn verbunden wäre, wurden aus verschiedenen Bereichen jeweils einige repräsentative Kandidaten ausgewählt. Die dabei berücksichtigten Kriterien sind im folgenden aufgelistet:

- **Interaktion mit dem Benutzer:** Eine vergleichbare, auf Blockdiagrammen beruhende Interaktion erlauben die Programme ECANSE, LABVIEW, SIMULINK und PIPELINE PILOT.
- **Anwendungsgebiet:** Ebenfalls im Anwendungsgebiet der Chemoinformatik angesiedelt sind die Programme JOELIB, MOE, MARVIN und PIPELINE PILOT.
- **Verteilte Berechnungen:** Die Mehrzahl der hier erwähnten Systeme erlaubt eine verteilte Ausführung der Berechnungen, wobei unterschiedliche Methoden mit einem Schwerpunkt auf der COM-Technologie von MICROSOFT eingesetzt werden.
- **Eingesetzte Technologien:** Die beiden für SOLVES eingesetzten Technologien XML und CORBA kommen bei keinem der in diesem Abschnitt aufgelisteten Systeme in dieser Kombination zum Einsatz.

Da es sich bei den aufgeführten Systemen überwiegend um kommerzielle Produkte handelt, war nur in wenigen Fällen eine Installation und damit ein Test vor Ort möglich. Die Angaben zu den Systemen beruhen in diesen Fällen auf den von den jeweiligen Firmen selbst zur Verfügung gestellten Informationen. Eine typische Quelle für solche Informationen stellen heutzutage die jeweils aufgeführten Internet-Seiten der Hersteller dar. Die dort erhältlichen Informationen wurden einer gewissenhaften

[22] Die Bezeichnung „System“ wird hier als Überbegriff verwendet und kann wahlweise für ein einzelnes Programm, ein Programmpaket, ein Framework oder ein Programmsystem stehen.

Auswertung unterzogen. Produkte, über die auf diesem Wege keine oder nur wenige Informationen erhältlich waren, konnten nicht in den Vergleich mit aufgenommen werden.[23]

3.9.1 Beschreibung der Systeme

Die folgenden Beschreibungen sind alphabetisch angeordnet. Es werden jeweils kurz die wichtigsten Eigenschaften und Einsatzgebiete benannt.

ECANSE In dem *Environment for Computer Aided Neural Software Engineering* werden dem Benutzer verschiedene Methoden aus den Bereichen der Neuronalen Netze, der Genetischen Algorithmen und der Fuzzy-Logik zur Verfügung gestellt. Es wurde von der SIEMENS AG AUSTRIA [Sie98b] für die Windows-Plattform in der Sprache C++ entwickelt. Als Einsatzgebiete werden die Vorhersage der Auslastung von Kraftwerken und entstehender Umweltbelastungen, die Vorhersage von Börsenkursen sowie Forschung und Lehre aufgeführt. Mit Hilfe des *Component Object Models* (COM) von MICROSOFT können vom Benutzer editierbare Skripte auf weiteren Windows-Rechnern verteilt ausgeführt werden.

JOELIB Die von WEGNER (joelib.sourceforge.org) erstellte Java-Bibliothek basiert auf einer von der Firma OPENEYE (www.eyesopen.com) in der Sprache C++ erstellten Bibliothek namens OELIB. Letztere baut wiederum auf BABEL auf, einem weit verbreiteten Konvertierungsprogramm für chemische Molekülformate. JOELIB enthält zahlreiche Algorithmen zur Deskriptorberechnung und ermöglicht die Zusammenstellung von linearen und verzweigten Berechnungsabläufen mit Hilfe von textbasierten Konfigurationsdateien.

LABVIEW Dient zur Aquisition, Auswertung und Visualisierung von Messdaten. Die zahlreich vorhandenen Funktionen werden graphisch als Bestandteil von Blockdiagrammen dargestellt. Die jeweiligen Ein- und Ausgabeparameter können miteinander verbunden werden. Die erste Version von LABVIEW wurde von der Firma NATIONAL INSTRUMENTS bereits im Jahre 1986 auf den Markt gebracht. Seitdem hat das Programm eine weite Verbreitung im Bereich Messtechnik und Messgeräteherstellung gefunden.

MARVIN Das Programmsystem MARVIN wurde von DOMINIK [Dom96] für die Automatisierung von Aufgaben aus dem Bereich des Molecular Modeling entworfen und implementiert. Ausgangspunkt der Berechnungen sind Molekülstrukturen, für welche unter anderem verschiedene Konformationen und Potentiale

[23] Dies betrifft beispielsweise die Produkte der Firmen INFORSENSE (www.inforsense.com) und TURBOWORX (www.turboworx.com).

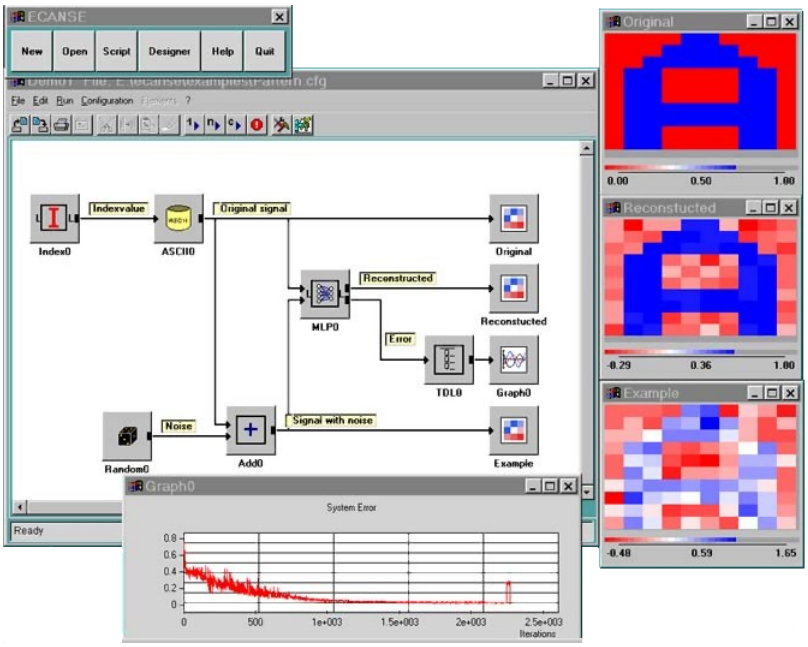

Abbildung 3.9: Screenshot der Benutzungsoberfläche von ECANSE.

berechnet werden. In Kombination mit Neuronalen Netzen, Non-Linear Mapping und Cluster-Analysen werden beispielsweise Pharmakophore oder Zuordnungen der Strukturen zu Arzneistoffklassen ermittelt. Der Ablauf der Berechnungen wird über eine Steuerdatei in einem textbasierten Format festgelegt. Die verwendeten Tools müssen als von der Konsole ausführbare Programme vorliegen. Eine Verteilung der Berechnungen auf mehrere Rechner ist möglich. Die Implementierung erfolgte in den Sprachen Shellscript und Perl.

MOE Das *Molecular Operating Environment* der CHEMICAL COMPUTING GROUP (www.chemcomp.com) stellt umfangreiche Funktionen aus dem Bereich des *Molecular Modeling* über eine graphische Benutzerschnittstelle zur Verfügung, in der Moleküle und Molekül datenbanken betrachtet und bearbeitet werden können. Mit Hilfe der vektorbasierten Skriptsprache SVL (*scientific vector language*) können neue Funktionen hinzugefügt und im Batchbetrieb berechnet werden. Eine graphische Erstellung von Workflows ist nicht vorgesehen, könnte aber vergleichsweise einfach hinzugefügt werden (ähnlich wie bei MATLAB und SIMULINK).

NEMO Das in der Scriptsprache Tcl/Tk implementierte Programm NEMO (***ne**ural **mo**deling*) wurde vom Autor im Rahmen einer Diplomarbeit erstellt und kann daher als Vorläufer von SOLVES gesehen werden. Ziel war bereits die Vereinfachung des praktischen Einsatzes von Neuronalen Netzen durch Bereitstellung von Methoden zur Daten-Aufbereitung, Validierung, Batch-Verarbeitung und

Archivierung. Allerdings werden neben Feed Forward-Netzen nur wenige Meta-Verfahren zur Optimierung der Netztopologien (Pruning, Greedy) bereitgestellt. Zur Durchführung der Berechnungen wird auf den Neuronale Netze-Simulator SNNS von ZELL *et al.* [ZMV+95] zurückgegriffen. Eine Erweiterung um andere Meta-Verfahren ist eingeschränkt möglich, eine Verteilung der Berechnungen nicht vorgesehen.

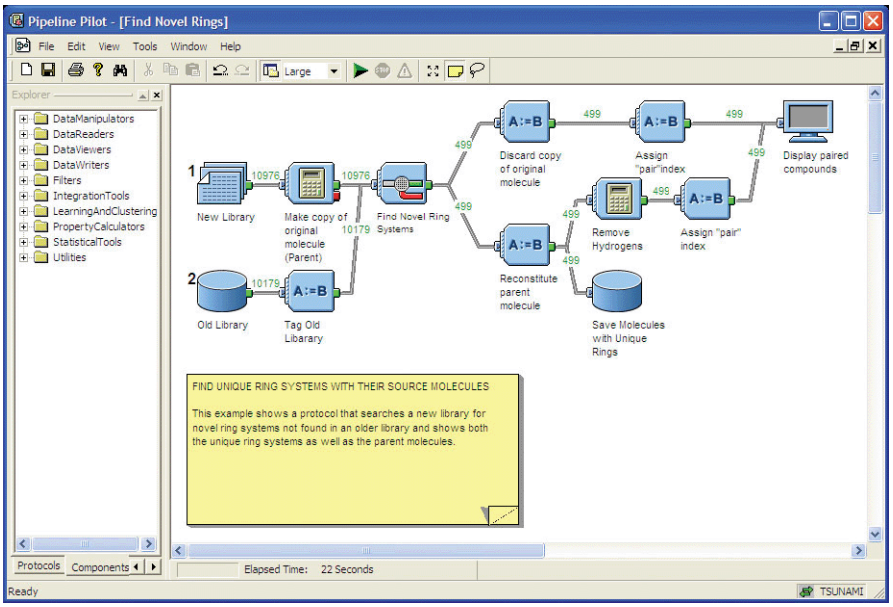

Abbildung 3.10: Screenshot des PIPELINE PILOTS von der Firma SCITEGIC.

PIPELINE PILOT Der von der Firma SCITEGIC (www.scitegic.com) entwickelte PIPELINE PILOT beruht auf dem sogenannten Prinzip des *Data Pipelining*. Bei diesem werden in einem flachen Tabellenformat vorliegende Daten über mehrere Stationen weitergereicht, wobei jede Zeile einzeln bearbeitet und zur nächsten Station geschickt wird. Auf diese Weise können im Laufe der Berechnungen die ersten Zeilen einer Tabelle schon mehrere Stationen weiter als der Rest gekommen sein. In den Zwischenstationen können sowohl Zeilen als auch Spalten hinzugefügt, editiert oder ausgefiltert werden, wobei eine Zeile meist ein Molekül (mit jeweils zugehörigen Informationen) repräsentiert. Eine Spalte kann beispielsweise Deskriptorwerte oder auch 3D-Molekülstrukturen enthalten. Die Weiterleitung der Daten erfolgt mit Hilfe des *Component Object Models* (COM) von MICROSOFT. Eine graphische Erstellung der Workflows ist möglich, wobei auf eine umfangreiche Funktionsbibliothek aus dem Bereich Chemie / Pharmazie zurückgegriffen werden kann. Die Workflows können verzweigen und in mehrere nicht-zusammenhängende Teile unterteilt sein, die dann nacheinander bearbeitet werden. Die Erweiterung des Datenformats auf hierarchische Struk-

turen und die Unterstützung von Web-Services sind für kommende Versionen geplant (Stand Anfang 2004).

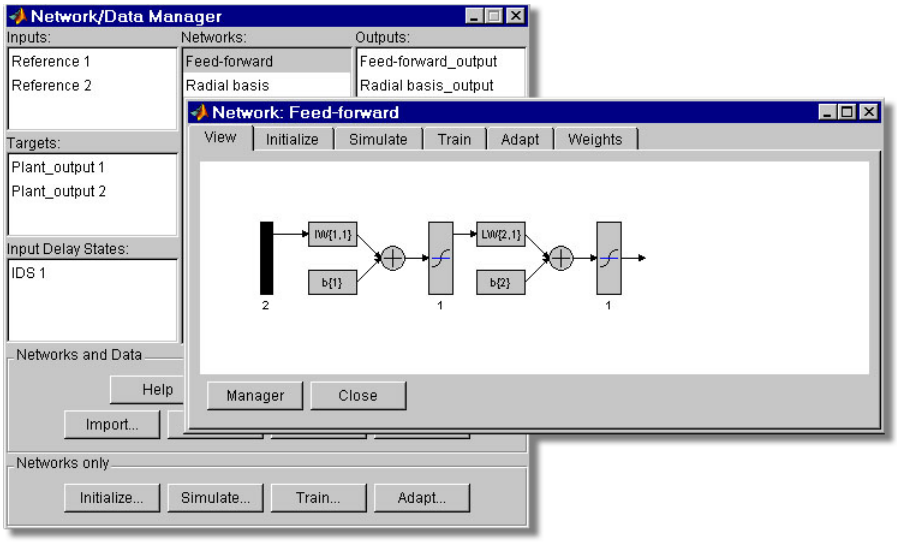

Abbildung 3.11: Erstellung eines Neuronalen Netzes in SIMULINK (unter Verwendung der Neuronale Netze-Toolbox von MATLAB).

SIMULINK Dient zur Modellierung, Simulation und Analyse zeitkontinuierlicher und zeitdiskreter dynamischer Systeme in Form von Blockdiagrammen. Es baut auf der technischen Programmiersprache MATLAB von der Firma MATHWORKS (www.mathworks.com) auf. Als Datentypen werden Gleitkomma- und Integer-Zahlen verwendet, die zu Vektoren und Matrizen zusammengefasst werden können. Eine Matrix repräsentiert dabei den Wert eines Signals zu verschiedenen Zeitpunkten. Insgesamt liegt der Schwerpunkt auf der Signalverarbeitung, wobei auch Neuronale Netze erstellt und trainiert werden können (vgl. Abb. 3.11). Die Definition von sogenannten „Subsystem-Blöcken“, die wiederum ein Blockdiagramm enthalten, ist möglich.

3.9.2 Gegenüberstellung der Eigenschaften

In der Tabelle 3.6 sind die Eigenschaften der betrachteten Systeme noch einmal in einer Übersicht aufgeführt. Die dabei verwendeten Symbole werden in Tabelle 3.7 aufgeschlüsselt. In der folgenden Aufzählung werden die als Grundlage der Bewertung dienenden Eigenschaften näher erläutert und konkretisiert.

Erläuterung der in der Tabelle 3.6 verwendeten Eigenschaften:

Definition von Prozessketten Gibt an, ob Prozessketten prinzipiell definierbar sind. Dies kann in graphischer Form (++) oder über textbasierte Steuerdateien (+)

Name des Systems	Definition von Prozessketten	Darstellung als Block-Diagramm	Erstellung von Makros	Verteilte Berechnung	Anbindung an Datenbanken	Eigene Programmiersprache	Maschinelles Lernen	Evolutionäre Optimierung	Anwendung im Wirkstoffdesign
ECANSE	++	++	++	○	−	+	+	−	−
JOELIB	+	−	−	−	+	○	++	+	++
LABVIEW	++	++	++	−	−	?	−	−	−
MARVIN	○	−	−	+	−	−	○	−	++
MOE	−	−	−	++	○	++	○	−	++
NEMO	−	−	−	−	−	−	+	−	+
PIPELINE PILOT	++	++	++	○	+	+	○	−	++
SIMULINK	++	++	++	−	−	++	++	?	○
SOLVES	++	++	++	++	++	○	++	++	++

Tabelle 3.6: Die Eigenschaften der betrachteten Systeme im Überblick.

erfolgen. Sind keine Verzweigungen, sondern nur rein lineare Prozessketten (○) vorgesehen, so wird dies eingeschränkt gewertet.

Darstellung als Block-Diagramm Bewertet die Fähigkeit, Prozessketten graphisch darzustellen (+) und über *drag-and-drop* editieren (++) zu können. Reine Darstellungen ohne Editierungsmöglichkeit (○) werden eingeschränkt gewertet.

Erstellung von Makros Bewertet die Fähigkeit, Teil-Prozessketten zu einem einzigen Modul zusammenziehen zu können. Dabei kann eine einfache (+) oder beliebige (++) Schachtelungstiefe erlaubt sein. Wenn die Parameter der zusammengefassten Module nicht mehr direkt editiert werden können (○), so wird dies eingeschränkt bewertet.

Verteilte Berechnung Gibt an, ob die Berechnungen auf mehrere Rechner (+) in einem heterogenen Umfeld (++) verteilt werden können. Wenn die Verteilung

Symbol	Beschreibung
$++$	Eigenschaft ist im vollen Umfang vorhanden.
$+$	Eigenschaft ist teilweise vorhanden.
$\circ$	Eigenschaft ist nur eingeschränkt vorhanden.
$-$	Eigenschaft ist in der betrachteten Version nicht vorhanden.
?	Über das Vorhandensein der Eigenschaft liegen keine oder nur unzureichende Informationen vor.

Tabelle 3.7: Erläuterung der in der Tabelle 3.6 verwendeten Symbole.

nur über einen speziellen Modultyp ($\circ$) möglich ist, so wird dies eingeschränkt gewertet.

Anbindung an Datenbanken Gibt an, ob die Anbindung an Datenbanken direkt auf der Systemebene ($++$) oder über spezielle Module ($+$) erfolgt. Dabei kann es sich um eine Anbindung an kommerzielle Datenbank-Managementsysteme wie Oracle oder DB2 handeln, oder um Open-Source DBMS wie MySQL. Einfache, auf flachen Dateiformaten basierende Datenbanksysteme ($\circ$) werden eingeschränkt gewertet.

Eigene Programmiersprache Bewertet die Bereitstellung einer eigenen, möglichst vollwertigen Programmiersprache zur Steuerung von Berechnungsabläufen. Dabei kann die Steuerungsmöglichkeit das ganze System ($++$) oder nur einen einzelnen Modultyp ($+$) umfassen. Die Möglichkeit der serverseitigen Erstellung von Modulen ($\circ$) in Skriptsprachen wie z. B. Python, Perl oder Tcl wird eingeschränkt gewertet.

Maschinelles Lernen Bewertet das Vorhandensein von Methoden aus dem Bereich des maschinellen Lernens. Dabei wird unterschieden, ob einfache Verfahren ($\circ$) (z. B. eine nichtlineare Projektion), oder ob wenige ($+$) oder viele ($++$) komplexere Verfahren (wie z. B. Neuronale Netze oder SVMs) vorhanden sind.

Evolutionäre Optimierung Auch hier wird zwischen wenigen, einfachen ($\circ$) und vielen, komplexeren ($+$) Verfahren unterschieden. Zusätzlich wird bei Systemen, die die Darstellung von Blockdiagrammen unterstützen, eine über die Anzeige eines einzelnen Moduls hinausgehende Einbindung in das Blockdiagramm ($++$) gefordert.

Anwendung im Wirkstoffdesign Gibt an, ob die Anwendung des Systems im Bereich Wirkstoffdesign bzw. Chemoinformatik mit einer Vielzahl ($++$) oder einer geringen Zahl ($+$) von Funktionen direkt unterstützt wird, oder ob ein Einsatz in diesem Bereich prinzipiell möglich ($\circ$) ist.

3.10 Alleinstellungskriterien

Wie im Abschnitt 3.1 bereits dargelegt, standen in dem Anwendungsbereich der pharmazeutischen Forschung bis vor kurzem keine zu SOLVES vergleichbaren Systeme zur Verfügung.[24] In anderen Anwendungsbereichen sind Systeme mit ähnlichen Eigenschaften allerdings bereits seit vielen Jahren etabliert. Im vorigen Abschnitt 3.9 wurden einige ausgewählte Systeme beschrieben und deren Eigenschaften tabellarisch dargestellt. Im folgenden wird jeweils erläutert, welche Eigenschaften des betrachteten Systems mit denen von SOLVES vergleichbar und welche unterschiedlich sind. In Klammern wird jeweils angegeben, auf welchen Spalten der Tabelle 3.6 die getroffenen Aussagen basieren.

Block-Diagramm-Systeme (Spalte 1-3) Die Darstellung von Prozessketten in Form von Block-Diagrammen ist bei LABVIEW, SIMULINK, PIPELINE PILOT und ECANSE möglich. Die beiden erstgenannten stammen aus dem Bereich der Signalverarbeitung, bei ihnen können nur relativ einfache Datentypen (Skalar, Vektor oder Matrix) zwischen den Blöcken ausgetauscht werden. PIPELINE PILOT unterstützt in der betrachteten Version nur ein flaches Tabellenformat. Bei ECANSE können zusätzlich vordefinierte Objekte übermittelt werden. Der Vorteil von SOLVES besteht in der Möglichkeit, zum einen vordefinierte Objekte (Parameter) als auch zur Laufzeit definierbare Kombinationen von Datentypen (Datensätze) zu übertragen.

Verteilte Berechnung und Datenhaltung (Spalte 4-5) Eine Verteilung der Berechnungen oder der Daten (über Datenbanken) ist in der Mehrzahl der betrachteten Systeme nicht oder nur eingeschränkt vorgesehen. ECANSE bietet einen speziellen Modultyp an, mit dessen Hilfe die zu diesem Modul gehörigen Berechnungen auf einem anderen Rechner ausgeführt werden können. MOE realisiert Moleküldatenbanken in Form von flachen, lokal abgelegten Dateien, wodurch zwar ein schneller und effizienter Zugriff, aber noch keine verteilte Datenhaltung ermöglicht wird.

Einfache Erstellung von Modulen (Spalte 6) Mit ihren eingebauten Programiersprachen ermöglichen MATLAB und MOE eine besonders große Flexibilität bei der Erstellung von neuen Modulen. Während die in MATLAB eingesetzte Skriptsprache einen deutlich spürbaren Overhead erzeugt, wurde bei der vektorbasierten Sprache SVL von MOE auf eine besonders performante Ausführung geachtet. SOLVES bietet in der vorgestellten Version keine direkt in das System eingebaute, eigene Skriptsprache an. Da es auf dem sprachübergreifenden CORBA-Standard aufbaut, ist jedoch die (serverseitige) Erstellung von neuen Modulen in Skriptsprachen wie z. B. Python oder Tcl möglich.

[24] Inzwischen sind z. B. mit dem Produkt PIPELINE PILOT von SCITEGIC im weiteren Sinne vergleichbare Systeme kommerziell verfügbar.

Methoden des Softcomputings (Spalte 7-8) Während zumindest einfache Varianten von Neuronalen Netzen in den meisten der verglichenen Systeme verfügbar sind, haben die Evolutionären Algorithmen eine deutlich geringere Verbreitung gefunden. Dies ist vermutlich auf Schwierigkeiten bei der Abbildung von Optimierungsverfahren in eine Block-Diagramm-Struktur begründet. SOLVES bietet einen speziellen Modultyp an, der eine einfache Definition des Fitnesskriteriums, welches bei Evolutionären Algorithmen eine zentrale Rolle spielt, erlaubt.

Anwendungbereich Chemie/Pharmazie (Spalte 9) Eine direkte Verarbeitung von Moleküldaten (ohne Zugriff auf externe Programmbibliotheken) ist in JOELIB und MOE möglich. Beide stellen zudem vielfältige Methoden zur Berechnung von Moleküldeskriptoren zur Verfügung. MARVIN verwendet BABEL als *chemical engine*, der PIPELINE PILOT besitzt eine Anbindung an die leistungsfähigen Produkte der Firma MDL (www.mdl.com). In SOLVES sind ausgewählte Funktionen von JOELIB und MOE in Form von Modulen angebunden.

Als vorläufiges Fazit lässt sich die Kombination von graphisch-interaktiver Zusammenstellung von Prozessketten und verteilter Ausführung der Berechnungen als Alleinstellungskriterium von SOLVES nennen. Zusammen mit den als Basis verwendeten Technologien Java, XML und CORBA, die bei den betrachteten Vergleichssystemen nicht oder nur teilweise zum Einsatz kommen, ergibt sich eine Eigenständigkeit des gewählten Ansatzes. Im Vergleich mit anderen Systemen besticht SOLVES durch den Umfang und die Qualität der eingebundenen Verfahren aus dem Bereich des maschinellen Lernens und der Vielzahl unterschiedlicher Optimierungsverfahren. Durch die einfache Erweiterbarkeit von SOLVES kann dieser bereits beachtliche Funktionsumfang problemlos an die Anforderungen eines neuen Anwendungsgebiets angepasst werden.

Kapitel 4

Anwendungen

In diesem Kapitel wird beschrieben, wie mit Hilfe des Programmsystems SOLVES typische Problemstellungen aus dem Bereich des Wirkstoffdesigns bearbeitet werden können. Hierbei beschränken sich die Ausführungen auf das Liganden-basierte Wirkstoffdesign, bei dem die 3D-Struktur des Rezeptors nicht als bekannt vorausgesetzt wird.

Die folgenden Abschnitte sind anhand der jeweils betrachteten Datensätze gegliedert. Begonnen wird mit einer einfachen Klassifizierungsaufgabe, bei der eine chemische Eigenschaft vorhergesagt werden soll (Abschnitt 4.1). Bereits schwieriger gestaltet sich dann die Vorhersage einer pharmakologischen Eigenschaft, nämlich der Bindung von Liganden an einen von vier Rezeptortypen, in Abschnitt 4.2. Die danach betrachtete Fragestellung, ob eine Substanz Ähnlichkeit zu den bisher bekannten Wirkstoffen besitzt, lässt sich nur näherungsweise beantworten. Eine gute Abschätzung stellt jedoch bereits ein wertvolles Hilfsmittel für die Pharmaforschung dar (Abschnitt 4.3). Zuletzt wird noch versucht, die Wasserlöslichkeit ($\log S$) einer Substanz vorherzusagen. Diese spielt für die orale Applizierbarkeit eines künftigen Medikaments eine wichtige Rolle und wird zunehmend bereits frühzeitig in die Suche nach neuen Wirkstoffen einbezogen (Abschnitt 4.4).

4.1 Klassifizierung einer chemischen Eigenschaft

In diesem Abschnitt soll bewusst zunächst ein kleiner, wenig komplexer Datensatz betrachtet werden, welcher die meisten Verfahren vor keine großen Schwierigkeiten stellt.[1] Das Ziel ist die Vorhersage einer chemischen Eigenschaft, nämlich ob eine Substanz aliphatisch, aromatisch oder ein Alkohol ist. Der Zusammenhang zwischen den verschiedenen eingesetzten Deskriptoren und dem Ergebnis der Klassifizierung

[1] Der Datensatz wurde freundlicherweise von Dr. Lothar Terfloth (Computer Chemie Centrum, Universität Erlangen) zur Verfügung gestellt.

ist hier viel leichter zu erkennen als bei der komplexeren Vorhersage von pharmakologischen Eigenschaften, wodurch sich die Stärken und Schwächen der eingesetzten Verfahren gut verdeutlichen lassen.

4.1.1 Beschreibung des Datensatzes

Der Datensatz besteht aus insgesamt 36 Molekülstrukturen. Eine Auflistung der Namen der Moleküle findet sich in Tabelle C.1 im Anhang. Die Strukturen sind in die drei Klassen `aliphatic`, `aromatic` und `alcohol` eingeteilt.

Deskriptoren: Es wird eine Menge von 25 Deskriptoren eingesetzt. Dabei handelt es sich um die Autokorrelationen der Atompolarisation (`autoAPolar`) und der Partialladungen (`autoQTot`), jeweils im Abstand von 1 bis 12 Einheiten, sowie um das Dipolmoment (`dipol`).[2]

Vorverarbeitung: Alle Deskriptoren wurden mit $x' = (x - x_{\mathrm{mean}})/x_{\mathrm{sdev}}$ zentriert und auf eine einheitliche Standardabweichung normiert.

Validierung: Da dieser Datensatz nur wenige Patterns enthält, wurde auf eine Aufteilung in Trainings- und Testdaten verzichtet (vgl. Abschnitt 2.3). Dafür wird für jedes Ergebnis eine 10-fache Kreuzvalidierung mit gleichmässiger Aufteilung der Patterns berechnet.

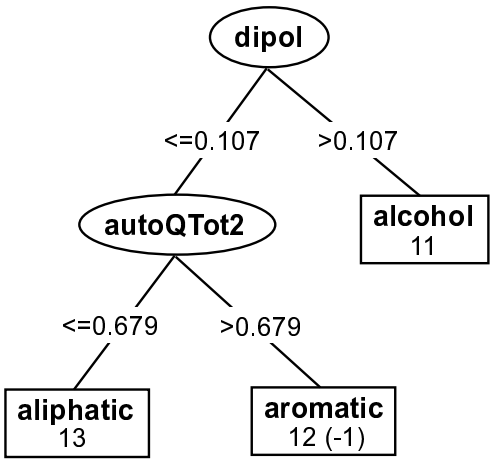

Abbildung 4.1: Der vom C4.5-Verfahren generierte Entscheidungsbaum. Die Zahlen unter den Klassennamen geben an, wie viele Patterns diesem Knoten insgesamt zugeordnet sind. Falsche Zuordnungen werden in Klammern angegeben.

[2]Zur Berechnung der Deskriptoren wurde die Programme PETRA und AUTOCORR verwendet, beide vom Computer Chemie Centrum, Universität Erlangen.

Verfahren	Modul	Klassifizierungsrate [%] Training	Kreuzval.
One Rule	Weka	72.22	72.22
C4.5	Weka	97.22	91.67
Naive Bayes	Weka	91.67	88.89
LVQ, 3 CBV	LVQ_PAK	72.22	66.67
LVQ, 6 CBV	LVQ_PAK	100.00	97.22
LVQ, 9 CBV	LVQ_PAK	100.00	**100.00**
KStar	Weka	100.00	86.11
PART	Weka	97.22	88.89
Decision Table	Weka	100.00	94.44
MLP, RProp, 2 vN	SNNS 4.2	100.00	97.22
MLP, RProp, 3 vN	SNNS 4.2	100.00	97.22
MLP, RProp, 4 vN	SNNS 4.2	100.00	91.67
MLP, SCG, 2 vN	SNNS 4.2	100.00	94.44
MLP, SCG, 3 vN	SNNS 4.2	100.00	97.22
MLP, SCG, 4 vN	SNNS 4.2	100.00	94.44
kNN, $k=1$	Weka	100.00	**100.00**
C-SVM, RBF, $C=1$	LIBSVM	100.00	97.22
C-SVM, linear, $C=1$	LIBSVM	100.00	**100.00**

Tabelle 4.1: Auf dem Datensatz erzielte Klassifizierungsraten. Bei dem MLP wurde das Lernverfahren und die Anzahl der versteckten Neuronen (vN) variiert, das Training umfasste jeweils 500 Lernzyklen.

4.1.2 Berechnungen mit allen Deskriptoren

In Tabelle 4.1 sind die mit allen 25 Deskriptoren erzielten Ergebnisse dargestellt. Eine optimale Klassifizierungsrate bei der Kreuzvalidierung (CV) wurde von den Verfahren LVQ mit neun Codebook-Vektoren, kNN mit einem Nachbarn ($k = 1$) und C-SVM mit linearem Kernel bei $C = 1$ erreicht.[3] Die Abhängigkeit der Vorhersage des LVQ-Verfahrens von der Anzahl der Codebook-Vektoren ist in Abb. 4.2 dargestellt. Ebenfalls gute Vorhersagen werden mit den Neuronalen Netzen vom feed-forward-Typ (MLP) getroffen, gefolgt von den Verfahren Decision Table und C4.5. Relativ schlecht ist dagegen die CV-Klassifizierungsrate der Verfahren KStar und PART, obwohl sie auf den Trainingsdaten jeweils ein gutes Ergebnis liefern. Dies unterstreicht einmal mehr die Notwendigkeit von internen (und wenn möglich auch von externen) Validierungsverfahren.

[3] Durch Variation des Kostenfaktors C ließ sich für den RBF-Kernel in diesem Fall keine Verbesserung erreichen. Für die polynomen und sigmoiden Kernelarten ergaben sich deutlich schlechtere Klassifizierungsraten (nicht gezeigt).

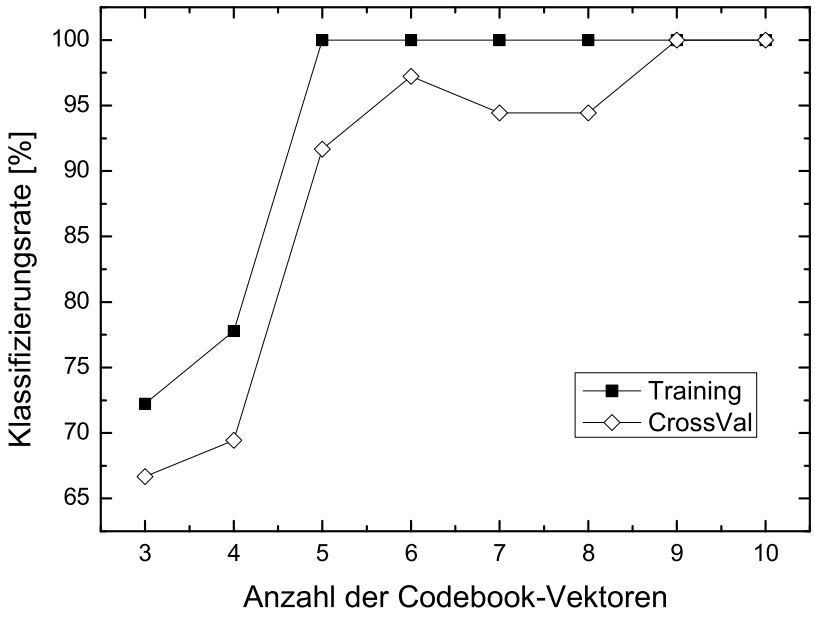

Abbildung 4.2: Abhängigkeit der Klassifizierungsrate von der Anzahl der Codebook-Vektoren beim LVQ-Verfahren.

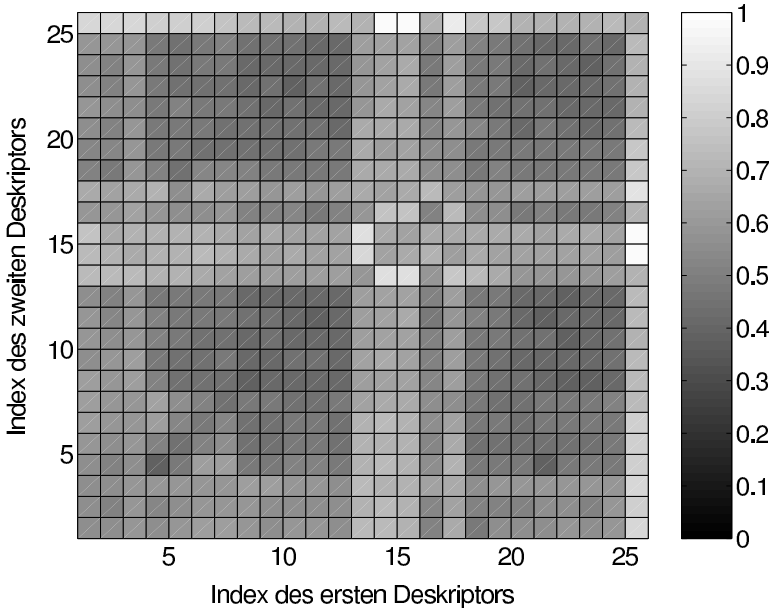

Abbildung 4.3: Klassifizierungsraten aller Kombinationen von je zwei Deskriptoren (Training).

Verfahren	Deskriptoren	Klassifizierungsrate [%] Training	Kreuzval.
C4.5	25	100.00	88.89
Naive Bayes	25, 14, 17	100.00	94.44
LVQ, 6 CBV	25, 14	97.22	97.22
Decision Table	14, 25	100.00	97.22
MLP, RProp, 2 vN	15, 25, 1	100.00	100.00
kNN, $k = 1$	1	100.00	66.67
C-SVM, RBF, $C = 1$	13, 14, 25, 1	100.00	100.00
C-SVM, RBF, $C = 1000$	13, 14	100.00	83.33
C-SVM, linear, $C = 1$	13, 14, 1, 25	100.00	100.00
C-SVM, linear, $C = 1000$	25, 15	100.00	94.44

Tabelle 4.2: Ergebnis der Greedy-Selektion, wobei als Auswahlkriterium die Klassifizierungsrate auf dem Trainingsdatensatz verwendet wurde. Die Nummern der selektierten Deskriptoren sind angegeben, wobei sich 1-12 auf die Autokorrelationen der Atompolarisation beziehen, 13-24 auf die Autokorrelationen der Partialladungen, und 25 auf das Dipolmoment.

4.1.3 Selektion von Deskriptoren

Im folgenden werden verschiedene Vorgehensweisen zur Variablenselektion diskutiert. Wegen der geringen Größe des untersuchten Datensatzes lassen sich auch aufwändigere „Meta-Verfahren“ anwenden.

Selektion über das Modell: Betrachtet man die im vorigen Abschnitt berechneten Modelle, so lässt sich am einfachsten aus dem C4.5-Entscheidungsbaum die Relevanz der Deskriptoren ablesen (Abb. 4.1): Nur die beiden Deskriptoren `autoQTOT2` und `dipol` gehen in das Modell ein, eine Selektion wird also bereits während des Trainings vorgenommen. Aus dem trainierten Neuronalen Netz lässt sich wegen des *black box*-Charakters ohne weiteres keine Information gewinnen, dasselbe gilt für die Codebook-Vektoren des LVQ-Verfahrens.

Selektion über einen Brute-Force-Ansatz: In Abb. 4.3 sind die Klassifizierungsraten sämtlicher Kombinationen von je 2 Deskriptoren gezeigt. Als beste Kombinationen ergeben sich `dipol` (25) zusammen mit `autoQTOT2` (14) oder zusammen mit `autoQTOT3` (15) (in beiden Fällen liegt die Klassifizierungsrate bei 97.22%).

Selektion mit einem Greedy-Verfahren: Hierbei wird von einer leeren Menge von Deskriptoren ausgegangen. Schrittweise wird ein weiterer Deskriptor hinzugefügt, wobei jeweils sämtliche Deskriptoren auf ihre Eignung getestet werden. Als Kriterium kann entweder die Klassifizierungsrate auf dem Trainingsdatensatz, oder die bei der Kreuzvalidierung erreichte Vorhersage verwendet werden.

Verfahren	Deskriptoren	Klassifizierungsrate [%] Training	Kreuzval.
C4.5	25, 14	97.22	94.44
Naive Bayes	25, 15, 1	100.00	100.00
LVQ, 6 CBV	25, 14	97.22	97.22
Decision Table	14, 25	100.00	97.22
MLP, RProp, 2 vN	25, 14, 1	100.00	100.00
kNN, $k = 1$	25	100.00	94.44
C-SVM, RBF, $C = 1$	25, 14, 1	100.00	100.00
C-SVM, RBF, $C = 1000$	25, 14, 1	100.00	100.00
C-SVM, linear, $C = 1$	25, 14, 1	100.00	100.00
C-SVM, linear, $C = 1000$	25, 14, 1	100.00	100.00

Tabelle 4.3: Ergebnis der Greedy-Selektion, wobei als Auswahlkriterium die Klassifizierungsrate der Kreuzvalidierung verwendet wurde. Die Nummern der selektierten Deskriptoren sind angegeben.

In Tabelle 4.2 sind die Ergebnisse einer Deskriptorselektion gezeigt, bei der die Auswahl anhand der auf dem Trainingsdatensatz erzielten Klassifizierungsrate getroffen wurde. Es fällt auf, dass die Verfahren C4.5 und kNN mit einem Nachbar ($k = 1$) schon nach der Auswahl eines einzelnen Deskriptors abbrechen, da sie auf dem Trainingsdatensatz bereits eine optimale Vorhersage erreicht haben. Der Kreuzvalidierungsfehler ist bei diesen Verfahren jeweils vergleichsweise hoch. Bei dem C-SVM-Verfahren führt ein ungünstig gewählter Kostenfaktor $C = 1000$ zu einer zu kleinen Auswahl an Deskriptoren. Die Tabelle 4.3 enthält die entsprechenden Ergebnisse, bei denen anhand der Vorhersage bei der Kreuzvalidierung selektiert wurde. Hier erreichen alle Verfahren relativ gute Ergebnisse, wobei kNN mit einem Nachbar nach wie vor nur einen Deskriptor auswählt. Auffällig ist, dass bei C-SVM der Einfluss des Kostenfaktors C nicht mehr erkennbar ist, für $C = 1$ und $C = 1000$ ergibt sich jeweils dieselbe Auswahl an Deskriptoren. Einige Verfahren, wie MLP, LVQ und Decision Table, kommen mit beiden Auswahlkriterien auf gute Vorhersagen und sinnvolle Deskriptormengen. Trotzdem lässt sich die Schlussfolgerung ziehen, dass bei einer Selektion von Deskriptoren bevorzugt der Kreuzvalidierungsfehler als Auswahlkriterium zum Einsatz kommen sollte.

4.1.4 Ergebnis

Anhand eines wenig komplexen Datensatzes wurden Möglichkeiten zur Variablenselektion aufgezeigt. Der C4.5-Entscheidungsbaum lieferte eine gute, aber etwas zu kleine Auswahl. Der Brute-Force-Ansatz lieferte einen rechenzeitaufwändigen Überblick über sämtliche Kombinationen von zwei Deskriptoren. Mit der Greedy-Methode

wurde mit vertretbarem Aufwand eine gute Auswahl erzielt. Es sei angemerkt, dass durchaus noch komplexere Meta-Verfahren denkbar sind, welche mit SOLVES auch mit vergleichweise geringem Aufwand umsetzbar wären. So könnte zur Deskriptorselektion auch eine Evolutionsstrategie eingesetzt werden, die neben der Menge der eingesetzten Deskriptoren gleichzeitig auch noch weitere Parameter, wie die Größe eines Neuronalen Netzes oder die *weight decay*-Rate des zugehörigen Lernverfahrens optimiert. Dieser Aufwand scheint angesichts des betrachteten einfachen Datensatzes jedoch in diesem Fall nicht notwendig zu sein, da auch mit einfacheren Mitteln bereits eine optimale Vorhersage erzielt werden kann.

4.2 Klassifizierung nach Indikationsgebiet

Wie in Abschnitt 2.4.2 erläutert wurde, besteht das Ziel des Liganden-basierten Wirkstoffdesigns darin, aus einer Menge von bereits bekannten, mehr oder weniger stark bindenden Liganden die für eine erwünschte Wirkung wesentlichen Merkmale herauszufiltern. Sind diese Merkmale bekannt, lassen sich neue Leitstrukturen finden, die wiederum zu neuen Arzneistoffen optimiert werden können.

Bei der Klassifizierung von Wirkstoffen nach Indikationsgebieten werden daher zwei Ziele verfolgt:

- Ein Klassifikator von akzeptabler Güte kann zur Suche nach neuen potentiellen Liganden in den großen Wirkstoffdatenbanken der Pharmafirmen eingesetzt werden. Wichtig hierbei ist, dass die Menge an Trainingspatterns eine ausreichende Diversität aufweist. Wird nur mit strukturell ähnlichen Substanzen trainiert, so kann für strukturell stark abweichende Substanzen keine zuverlässige Vorhersage getroffen werden.

- Eine erfolgreiche Klassifizierung lässt aber auch den Rückschluss zu, dass in den Eingabevariablen (Deskriptoren) relevante Informationen enthalten sind. Es kann versucht werden, aus der Art der benötigten Deskriptoren Rückschlüsse auf Eigenschaften zu ziehen, die sich auf die Wirkstärke einer Substanz positiv auswirken. Ob dies gelingt, hängt wesentlich von der Art der Berechnung bzw. von dem Abstraktionsgrad der Deskriptoren ab.

4.2.1 Beschreibung des Datensatzes

Der untersuchte Datensatz besteht aus vier Substanzklassen. Dabei handelt es sich um 5HT1A-Agonisten, H2-Antagonisten, Thrombin- und MAO-A-Inhibitoren.[4] Von den ersten drei Klassen liegen jeweils 75 Moleküle vor, die letzte Klasse umfasst 74 Moleküle.[5]

Deskriptoren: Mit Hilfe von MOE (Version 2004.3) wurden insgesamt 146 Deskriptoren berechnet. Es handelt sich dabei um alle verfügbaren 2D-Deskriptoren. Die dreidimensionale Struktur der Liganden ging somit bei keinem der eingesetzten Deskriptoren in die Berechnung ein. Die Namen der Deskriptoren sind in Tabelle C.2 im Anhang zu finden.

[4] Der Datensatz wurde dankenswerterweise von Michael Bieler (Universität Marburg) im Rahmen des SOL-Projekts zusammengestellt.

[5] In der MAO-A-Klasse waren ursprünglich zwei Moleküle enthalten, die sich nur in ihrer Chiralität unterschieden haben. Da die meisten Deskriptoren für diese die gleichen Werte liefern würden, wurde eine der beiden Strukturen aus dem Datensatz entfernt.

Vorverarbeitung: Alle Deskriptoren wurden mit $x' = (x - x_{\mathrm{mean}})/x_{\mathrm{sdev}}$ zentriert und auf eine einheitliche Standardabweichung normiert.

Validierung: Von den insgesamt 299 Strukturen werden 240 für das Training verwendet und 59 einem unabhängigen Testdatensatz zugeordnet (rund 20%). Die Auswahl der Testdaten erfolgte in gleichmässigen Abständen. Zusätzlich wird für jedes Ergebnis eine 10-fache Kreuzvalidierung mit gleichmässiger Aufteilung der Patterns berechnet.

		Klassifizierungsrate [%]		
Verfahren	Modul	Training	Test	Kreuzval.
C4.5	Weka	97.08	76.27	86.25
Naive Bayes, ohne DA	Weka	83.75	74.58	79.17
Naive Bayes, mit DA	Weka	87.50	77.97	82.92
LVQ, 10 CBV	LVQ_PAK	86.67	81.36	82.08
LVQ, 15 CBV	LVQ_PAK	90.83	84.75	83.75
LVQ, 25 CBV	LVQ_PAK	91.25	86.44	84.58
KStar	Weka	100.00	89.83	95.42
PART	Weka	97.92	86.44	87.50
Decision Table	Weka	95.42	79.66	77.08
MLP, RProp, 3 vN	SNNS 4.2	99.58	93.22	95.83
MLP, RProp, 4 vN	SNNS 4.2	100.00	93.22	96.25
MLP, RProp, 5 vN	SNNS 4.2	100.00	89.83	94.58
MLP, RProp, 8 vN	SNNS 4.2	100.00	84.75	92.50
MLP, SCG, 3 vN	SNNS 4.2	97.08	88.14	95.00
MLP, SCG, 4 vN	SNNS 4.2	100.00	89.83	94.58
MLP, SCG, 5 vN	SNNS 4.2	99.58	91.53	**96.67**
MLP, SCG, 8 vN	SNNS 4.2	100.00	89.83	93.33
kNN, $k = 1$	Weka	100.00	96.61	95.83
kNN, $k = 2$	Weka	97.50	91.53	93.75
kNN, $k = 3$	Weka	97.92	94.92	92.92
ν-SVM, linear, $\nu = 0.1$	LIBSVM	100.00	96.61	96.25
ν-SVM, RBF, $\nu = 0.08$	LIBSVM	100.00	96.61	95.83
ν-SVM, polynom, $d = 2$, $\nu = 0.1$	LIBSVM	100.00	91.53	93.75
ν-SVM, polynom, $d = 3$, $\nu = 0.02$	LIBSVM	100.00	**98.31**	93.75
ν-SVM, polynom, $d = 4$, $\nu = 0.02$	LIBSVM	100.00	91.53	91.25

Tabelle 4.4: Auf dem Datensatz erzielte Klassifizierungsraten. Das Naive Bayes-Verfahren wurde mit und ohne Dichteabschätzung (DA) berechnet. Bei den MLP wurde die Anzahl der versteckten Neuronen (vN) variiert, bei ν-SVM der Grad d des polynomiellen Kernels. Die besten Klassifizierungsraten sind hervorgehoben.

4.2.2 Berechnungen mit allen Deskriptoren

In Tabelle 4.4 sind die Ergebnisse der Berechnungen mit allen 146 2D-Deskriptoren aufgeführt. Die Verfahren C4.5 und Naive Bayes erzielen relativ schlechte Vorhersagen auf dem Testdatensatz. Eine Abschätzung der Dichteverteilung der Deskriptorwerte führt bei Naive Bayes zu einer leichten Verbesserung. Die Vorhersage des LVQ-Verfahrens liegt im mittleren Bereich, durch Erhöhung der Anzahl der Codebook-Vektoren (CBV) lassen sich die erzielten Klassifizierungsraten noch leicht verbessern. Ebenfalls im Mittelfeld liegen die Verfahren KStar und PART.[6] Mit Feedforward-Netzen (MLP) lassen sich gute Vorhersagen bei der Kreuzvalidierung erreichen, die Ergebnisse auf den Testdaten sind jedoch gemischt. Das instanzbasierte kNN-Verfahren erreicht sehr gute Vorhersagen sowohl auf den Testdaten, als auch bei der Kreuzvalidierung. Dabei ist es wiederum am besten, nur einen der umliegenden Nachbarn zu berücksichtigen ($k = 1$). Das Ergebnis bei mehreren Nachbarn kann noch leicht verbessert werden, wenn diese anhand ihrer Ähnlichkeit gewichtet werden (nicht gezeigt). Auf den Testdaten schneidet das ν-SVM-Verfahren mit polynomiellen Kernel vom Grad (*degree*) $d=3$ am besten ab, wobei der Fehler der Kreuzvalidierung deutlich höher ist. Mit dem linearen und dem RBF-Kernel konnten auch sehr gute Vorhersagen erzielt werden. Die ν-Wert wurden durch Variation im Intervall $(0, 1)$ mit einer Schrittweite von 0.02 bestimmt. Es wurde jeweils der ν-Wert ausgewählt, welcher die beste Vorhersage bei der Kreuzvalidierung liefert.

Um den Ursachen von Klassifizierungsfehlern nachzuspüren ist ein Blick auf die jeweilige Verwechslungsmatrix (*confusion matrix*) hilfreich. Betrachtet man die bisher beste, vom ν-SVM-Verfahren mit polynomiellen Kernel dritten Grades berechnete Vorhersage auf den Testdaten in Tabelle 4.5, so fällt auf, dass nur zwischen den Klassen MAO-A und 5HT1A Verwechslungen auftreten. Die beiden anderen Klassen, H2-ANT und Thrombin, werden korrekt vorhergesagt.

Ein Beispiel für ein eher mittelmässiges Ergebnis ist in Tabelle 4.6 dargestellt. Es wurde mit dem Naive Bayes-Verfahren berechnet, wobei die Dichteverteilung der Deskriptoren abgeschätzt wurde. Obwohl es zwischen den ersten drei Klassen 5HT1A, H2-ANT und MAO-A viele Verwechslungen gibt, wird selbst hier noch Thrombin vollständig richtig vorhergesagt. Dies lässt sich zum einen dadurch erklären, dass der Thrombin-Rezeptor relativ starr ist, also eine geringe Flexibilität aufweist. Zum anderen weisen die Thrombin-Inhibitoren im verwendeten Datensatz eine hohe strukturelle Ähnlichkeit auf. Das größte Problem bei der Vorhersage der Klassenzugehörigkeiten scheint es somit zu sein, die Klassen MAO-A und 5HT1A auseinanderzuhalten.

[6] Eine Beschreibung der beiden Verfahren KStar und PART findet sich in [WF00]. Da diese Verfahren sehr rechenzeitaufwändig sind, eignen sie sich in der Regel nur für kleinere Datensätze.

	5HT1A	H2-ANT	MAO-A	Thrombin
5HT1A	100		7.14	
H2-ANT		100		
MAO-A			92.86	
Thrombin				100

Tabelle 4.5: Die Verwechslungsmatrix des ν-SVM-Verfahrens mit polynomiellen Kernel dritten Grades für den Testdatensatz. Die Zeilen geben die „wahren“, die Spalten die vorhergesagten Klassen an. Fehlklassifizierungen werden abseits der Hauptdiagonalen aufgetragen. Sie treten in diesem Fall nur zwischen den Klassen MAO-A und 5HT1A auf.

	5HT1A	H2-ANT	MAO-A	Thrombin
5HT1A	60		21.43	
H2-ANT	13.33	80	7.14	
MAO-A	26.67	20	71.43	
Thrombin				100

Tabelle 4.6: Die Verwechslungsmatrix des Naive Bayes-Verfahrens mit Dichteabschätzung (Testdatensatz). Trotz der vielen Fehlklassifizierungen zwischen den ersten drei Klassen wird Thrombin korrekt vorhergesagt.

4.2.3 Berechnungen mit abgeleiteten Variablen

Um die Anzahl der Eingabevariablen zu reduzieren, wird eine Hauptkomponentenanalyse (PCA) auf den 146 Deskriptoren berechnet. Dabei wird bestimmt, wieviele der Hauptkomponenten notwendig sind, um einen vorgegebenen Anteil der Varianz im Datensatz zu erklären. Mit Hilfe von Abb. 4.4 kann ermittelt werden, wie hoch der benötigte Anteil der Varianz für den betrachteten Datensatz ist. Werden weniger als 98% der Varianz im Datensatz erhalten, dann sinkt die Klassifizierungsrate bei der Kreuzvaliderung für das bisher beste Verfahren, ν-SVM mit polynomiellen Kernel dritten Grades. Setzt man die zu erklärende Varianz aus diesem Grund auf 98%, so werden im konkreten Beispiel die ersten 32 Hauptkomponenten benötigt. Diese abgeleiteten Variablen werden im folgenden als Eingabe für die bereits im letzten Abschnitt eingesetzten Verfahren verwendet.

In Tabelle 4.7 sind die erreichten Klassifizierungsraten dargestellt. Für C4.5 und Naive Bayes ergeben sich leichte Verbesserungen der Vorhersage auf dem Testdatensatz. Verschlechtert hat sich dagegen das Ergebnis für die Verfahren LVQ, PART und MLP. Nach wie vor am besten ist die Vorhersage des ν-SVM-Verfahrens, hier hat sich nur das Ergebnis für den linearen Kernel leicht verschlechtert. Ebenso unverändert ist die Klassifizierungsrate des kNN-Verfahrens mit $k=1$ auf dem Testdatensatz.

Verfahren	Modul	Klassifizierungsrate [%] Training	Test	Kreuzval.
C4.5	Weka	97.50	86.44	76.25
Naive Bayes, mit DA	Weka	94.58	89.83	86.67
LVQ, 10 CBV	LVQ_PAK	86.67	81.36	83.33
KStar	Weka	100.00	89.83	86.67
PART	Weka	98.33	77.97	80.00
MLP, RProp, 3 vN	SNNS 4.2	99.58	89.83	90.42
MLP, RProp, 4 vN	SNNS 4.2	99.58	89.83	92.50
kNN, $k = 1$	Weka	100.00	96.61	92.92
ν-SVM, linear, $\nu = 0.1$	LIBSVM	99.58	94.92	93.33
ν-SVM, RBF, $\nu = 0.08$	LIBSVM	100.00	96.61	92.50
ν-SVM, polynom, $d = 3$, $\nu = 0.02$	LIBSVM	100.00	**98.31**	**93.75**

Tabelle 4.7: Auf den mit PCA abgeleiteten Variablen erzielte Klassifizierungsraten (32 Hauptkomponenten, erklärte Varianz 98%). Die besten Klassifizierungsraten sind auch hier hervorgehoben.

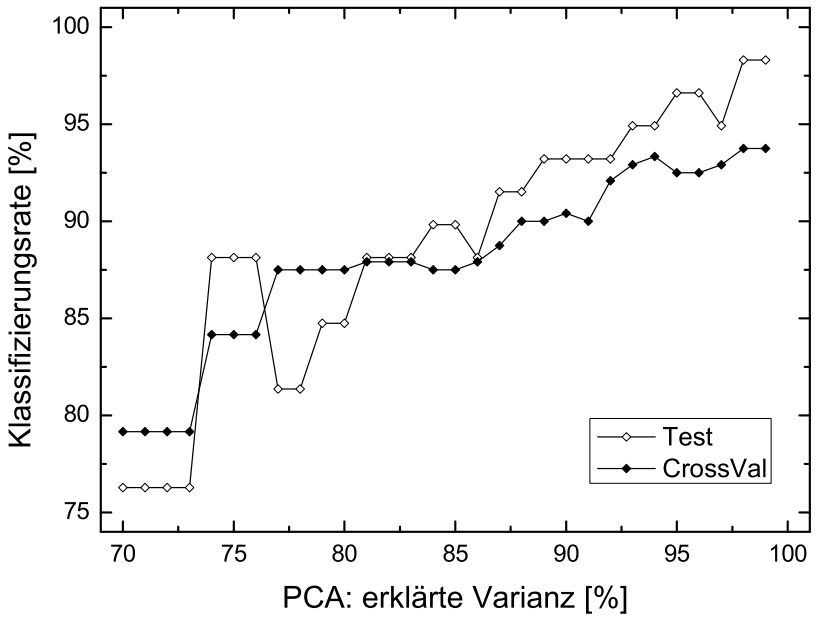

Abbildung 4.4: Abhängigkeit der mit ν-SVM (polynomieller Kernel, $d = 3$, $\nu = 0.02$) erzielten Klassifizierungsraten von der erklärten Varianz einer PCA auf den Eingabemustern. Der kleinste Kreuzvalidierungsfehler wird ab 98% erklärter Varianz erzielt.

Hieraus lässt sich schließen, dass die Anzahl der Eingabevariablen prinzipiell deutlich verringert werden kann, in diesem Fall von 146 auf 32. Die Vorhersagefähigkeit der Verfahren ist durch diese Verringerung insgesamt gesehen nicht wesentlich beeinträchtigt worden, die beste Klassifizierungsrate auf dem Testdatensatz liegt nach wie vor bei 98.31%. Allerdings konnten nur 2% der im Datensatz vorliegenden Varianz entfernt werden. Dieser relativ geringe Wert lässt sich dadurch erklären, dass die Werte der Eingabevariablen in diesem Fall nicht mit einem Messfehler behaftet sind, da sie durchweg aus der Molekülstruktur *in silico* berechnet wurden. Die Eigenschaft von PCA, unwichtiges „Rauschen“ im Datensatz reduzieren zu können, kommt hier also nur begrenzt zum Tragen. Zudem müssen sich die für die Vorhersage der Klassenzugehörigkeiten notwendigen Informationen nicht unbedingt durch eine hohe Varianz auszeichnen. Dies wäre besonders dann der Fall, wenn einige wenige der Deskriptoren besonders viel an wichtigen Informationen enthalten, wenn diese also nicht gleichmässig über alle Deskriptoren verteilt sind. Ob es möglich ist, eine entsprechend geringe Anzahl von einzelnen Deskriptoren zu selektieren, wird im folgenden Abschnitt untersucht.

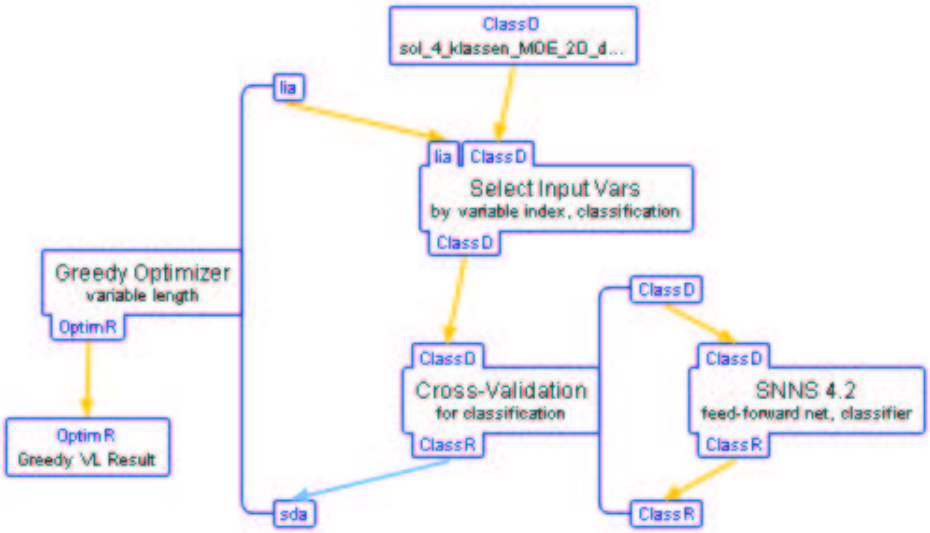

Abbildung 4.5: Abbildung der in SOLVES definierten Prozesskette, welche für die Selektion der Deskriptoren eingesetzt wird.

4.2.4 Selektion von Deskriptoren

In diesem Abschnitt soll eine möglichst kleine Untermenge der 146 2D-Deskriptoren ermittelt werden, mit der trotzdem noch eine gute Vorhersage erzielt werden kann. Hierfür wird die in Abb. 4.5 dargestellte Prozesskette eingesetzt. Ein Klassifizierungsverfahren wird dabei mit einer Kreuzvalidierung verbunden, welche wiederum zu-

Verfahren	Modul	#d	Klassifizierungsrate [%] Training	Test	Kreuzval.
C4.5	Weka	5	97.08	88.14	91.25
Naive Bayes, mit DA	Weka	6	92.08	84.75	92.08
LVQ, 10 CBV	LVQ_PAK	7	91.25	81.36	92.08
PART	Weka	5	97.08	86.44	93.33
MLP, RProp, 3 vN	SNNS 4.2	6	90.00	81.36	85.83
MLP, RProp, 4 vN	SNNS 4.2	6	91.67	76.27	90.42
kNN, $k = 1$	Weka	8	100.00	88.14	97.92
ν-SVM, RBF, $\nu = 0.08$	LIBSVM	8	99.17	**94.92**	**98.33**
C-SVM, linear, $C = 10$	LIBSVM	7	92.50	83.05	92.08
C-SVM, RBF, $C = 100$	LIBSVM	8	100.00	**94.92**	**98.33**
C-SVM, poly., $d = 3$, $C = 10$	LIBSVM	6	96.67	84.75	92.50

Tabelle 4.8: Auf den selektierten Deskriptoren erzielte Klassifizierungsraten. Die Kreuzvalidierung wurde jeweils mit fünf *folds* berechnet. Die Anzahl der ausgewählten Deskriptoren (#d) ist angegeben. Das MLP wurde mit 500 Lernzyklen trainiert.

sammen mit einer Variablenselektion einem Greedy-Optimierungsprozess unterzogen wird. Ein wichtiger Punkt ist dabei, dass die Optimierung nicht direkt auf den im Training erzielten Klassifizierungsraten erfolgt, sondern dass über eine zwischengeschaltete interne Validierung ein Übertrainieren bzw. reines „Auswendiglernen“ der Trainingsdaten verhindert wird.[7] Dabei wird im folgenden stets eine 5-fache Kreuzvalidierung verwendet, d. h. bei jeder Auswertung einer Kombination von selektierten Deskriptoren wird das jeweils eingesetzte Verfahren wiederholt aufgerufen, wobei reihum 20% der 240 Trainingspatterns ausgespart werden. Die Selektion erfolgt mit Hilfe des Kreuzvalidierungsfehlers. Die minimale Verbesserungsrate (*minimal improvement rate*, MIR) pro Selektionsschritt wurde auf 1% gesetzt.

In Tabelle 4.8 sind die erzielten Klassifizierungsraten dargestellt. Es fällt auf, dass jeweils eine relativ geringe Anzahl an Deskriptoren ausgewählt wird (5 bis 8). Mehrere Verfahren erzielen gute Vorhersagen bei der Kreuzvalidierung. Die Klassifizierungsrate auf den Testdaten liegt jedoch teilweise deutlich niedriger. Solche großen Abweichungen dieser beiden Fehlermaße sind ungewöhnlich, lassen sich in diesem Fall aber wie folgt erklären: Die Auswahl der Deskriptoren erfolgte anhand der erzielten Kreuzvalidierungsrate, es setzten sich also jeweils diejenigen Deskriptoren durch, welche die in den Trainingsdaten auftretenden Zusammenhänge erklären. Dabei kann nicht ausgeschlossen werden, dass zufällige Korrelationen (Artefakte) mit echten Korrelationen verwechselt werden. Bei der Auswahl von zufällig auf den Trainingsdaten passenden Korrelationen verschlechtert sich jedoch die externe Validierung, also die Vorhersage

[7] Ein solcher Übertrainierungseffekt tritt nicht nur bei Neuronalen Netzen auf, auch modellfreie Verfahren wie kNN sind davon betroffen.

auf dem Testdatensatz. Dieser Effekt verstärkt sich noch, wenn eine große Anzahl von Deskriptoren (hier 146) zur Auswahl angeboten wird. Besonders stark scheint dieser Effekt bei dem instanzbasierten kNN-Verfahren aufzutreten.

Das beste Ergebnis auf dem Testdatensatz wird von dem SVM-Verfahren mit RBF-Kernel erzielt. Dabei erreichen die beiden Varianten ν-SVM und C-SVM gleich gute Vorhersagen. Interessanterweise unterscheiden sich die Mengen der selektierten Deskriptoren relativ stark, von acht Deskriptoren treten nur drei in beiden Auswahlmengen auf (vgl. Tab. 4.9). Hierdurch wird die hohe Redundanz, welche in der Gesamtmenge der 146 Deskriptoren vorhanden ist, noch einmal verdeutlicht.

Verfahren	Selektierte Deskriptoren							
C4.5	57	66	20	92	15			
Naive Bayes, mit DA	136	15	54	120	34	122		
LVQ, 10 CBV	136	20	54	56	113	128	76	
PART	57	79	15	134	20			
MLP, RProp, 3 vN	142	136	66	20	57	50		
MLP, RProp, 4 vN	136	20	54	142	120	59		
kNN, $k = 1$	124	136	20	54	44	22	49	39
ν-SVM, RBF, $\nu = 0.08$	136	20	140	135	77	49	36	52
C-SVM, linear, $C = 10$	142	136	20	54	56	74	15	
C-SVM, RBF, $C = 100$	142	136	20	54	26	85	43	77
C-SVM, poly., $d = 3$, $C = 10$	57	136	20	126	124	18		

Tabelle 4.9: Auflistung der von den jeweiligen Verfahren selektierten Deskriptoren. Eine Zuordnung dieser Nummern zu den Deskriptornamen ist in Tabelle C.2 zu finden.

Betrachtet man nun anhand von Tabelle 4.9, welche Deskriptoren von den einzelnen Verfahren ausgewählt werden, so wird deutlich, dass keine zwei Verfahren genau die gleichen Deskriptoren ausgewählt haben. Zu den häufiger selektierten Deskriptoren zählen der Anteil der rotierbaren Bindungen im Molekül (Deskriptor b_rotR, Nr. 20), der Anteil der Moleküloberfläche, für den die molare Refraktivität R_i im Intervall $(0.26, 0.35]$ liegt (Deskriptor SMR_VSA2, Nr. 136) und der Anteil der Moleküloberfläche mit einer geringen Partialladung (Deskriptor PEOE_VSA+0, Nr. 54). Da die Deskriptoren b_rotR (Nr. 20) und b_1rotR (Nr. 15) sehr ähnlich sind (letzterer steht für den Anteil der rotierbaren Einfach-Bindungen im Molekül), lässt sich feststellen, dass die Verfahren Naive Bayes, LVQ und MLP mit vier versteckten Neuronen die ersten drei Deskriptoren übereinstimmend ausgewählt haben (Nr. 136, 15/20, 54). Diese Deskriptoren werden auch von kNN ausgewählt, allerdings erst nach dem Deskriptor SlogP_VSA1 (Nr. 124).

Da von den betrachteten Verfahren jeweils andere Deskriptoren ausgewählt wurden, wird in einem nächsten Schritt der Versuch unternommen, die selektierten Deskriptoren zusammenzufassen. Dabei sind zwei unterschiedliche Vorgehensweisen denkbar:

Verfahren	Modul	Klassifizierungsrate [%] Training	Test	Kreuzval.
C4.5	Weka	97.50	86.44	84.58
Naive Bayes, mit DA	Weka	94.58	83.05	90.42
LVQ, 10 CBV	LVQ_PAK	87.92	77.97	84.58
LVQ, 25 CBV	LVQ_PAK	95.42	89.83	90.00
KStar	Weka	100.00	89.83	95.83
PART	Weka	97.50	84.75	87.92
MLP, RProp, 3 vN	SNNS 4.2	99.58	**94.92**	92.50
MLP, RProp, 4 vN	SNNS 4.2	98.75	91.53	94.58
kNN, $k = 1$	Weka	100.00	93.22	**96.67**
ν-SVM, linear, $\nu = 0.1$	LIBSVM	99.17	**94.92**	94.17
ν-SVM, RBF, $\nu = 0.08$	LIBSVM	100.00	**94.92**	**96.67**
ν-SVM, polynom, $d = 3$, $\nu = 0.02$	LIBSVM	100.00	**94.92**	95.00

Tabelle 4.10: Auf dem gepoolten Satz von 35 Deskriptoren erzielte Klassifizierungsraten. Die Angaben beziehen sich auf eine zehnfache Kreuzvalidierung. Das MLP wurde jeweils mit 1000 Lernzyklen trainiert.

Es können entweder alle in Tabelle 4.9 aufgeführten Deskriptoren zusammengefasst werden. In diesem Fall ergibt sich eine Menge von 35 Deskriptoren. Oder es werden nur die Deskriptoren gepoolt, welche mindestens zweimal selektiert wurden. Dieser kleinere Deskriptorensatz umfasst 12 Deskriptoren.[8]

Die auf diesen beiden gepoolten Mengen von Deskriptoren erzielten Vorhersagen sind in den Tabellen 4.10 und 4.11 dargestellt. Die Verfahren C4.5 und Naive Bayes mit Dichteabschätzung erreichen auf beiden gepoolten Deskriptormengen eine bessere Vorhersage auf dem Testdatensatz als mit allen 146 Deskriptoren (vgl. Tab. 4.4). Die beste Vorhersagen auf dem größeren Deskriptorenpool (94.92%) werden vom MLP mit drei versteckten Neuronen und von ν-SVM mit den linearen, polynomiellen und RBF-Kerneln erzielt. Auf dem kleineren Deskriptorsatz wird von ν-SVM mit den polynomiellen und RBF-Kerneln mit jeweils 93.22% eine fast ebenso gute Vorhersage auf den Testdaten erreicht.

4.2.5 Ergebnis

Die beste Klassifizierungsrate auf dem unabhängigen Testdatensatz (98.31%) wurde von ν-SVM mit polynomiellen Kernel ($d = 3$, $\nu = 0.02$) auf allen 146 Deskriptoren erzielt. Das zweitbeste Ergebnis wurde von ν-SVM mit den linearen und RBF-

[8] Die kleinere gepoolte Deskriptorauswahl enthält die Deskriptoren mit den Nummern 15, 20, 49, 54, 56, 57, 66, 77, 120, 124, 136 und 142 (vgl. Tabelle C.2).

Verfahren	Modul	Klassifizierungsrate [%] Training	Test	Kreuzval.
C4.5	Weka	97.08	84.75	85.00
Naive Bayes, mit DA	Weka	90.42	86.44	87.08
LVQ, 10 CBV	LVQ_PAK	92.08	84.75	90.42
LVQ, 25 CBV	LVQ_PAK	95.00	84.75	89.58
KStar	Weka	100.00	89.83	95.42
PART	Weka	97.92	88.14	88.75
MLP, RProp, 3 vN	SNNS 4.2	95.83	84.75	87.50
MLP, RProp, 4 vN	SNNS 4.2	96.67	84.75	89.17
kNN, $k = 1$	Weka	100.00	91.53	**98.33**
ν-SVM, linear, $\nu = 0.1$	LIBSVM	93.75	83.05	91.67
ν-SVM, RBF, $\nu = 0.08$	LIBSVM	100.00	**93.22**	97.08
ν-SVM, polynom, $d = 3$, $\nu = 0.02$	LIBSVM	100.00	**93.22**	95.42

Tabelle 4.11: Auf dem kleineren gepoolten Satz von 12 Deskriptoren erzielte Klassifizierungsraten. Das MLP wurde auch hier jeweils mit 1000 Lernzyklen trainiert.

Kerneln, sowie von kNN mit einem Nachbarn ($k = 1$) erreicht. Die Verwendung von mit PCA berechneten Hauptkomponenten (erklärte Varianz 98%) führte lediglich bei ν-SVM mit dem linearen Kernel zu einer leichten Verschlechterung der Vorhersage. Die Anzahl der Eingabevariablen konnte somit praktisch verlustfrei von 146 auf 32 reduziert werden. Eine direkte Selektion von Deskriptoren mit dem Greedy-Verfahren war besonders bei ν-SVM und C-SVM, jeweils mit RBF-Kernel erfolgreich. Es konnten unterschiedliche Mengen von acht Deskriptoren ausgewählt werden, welche auf dem Testdatensatz eine Vorhersage von 94.92% erbrachten. Es wurde der Versuch unternommen, die in mehreren Selektionsläufen erhalten Deskriptormengen zusammenzufassen. Auf der größeren, alle zuvor selektierten Deskriptoren enthaltenden Menge konnte eine Vorhersage von 94.92%, auf der kleineren Menge der mehrfach selektierten Deskriptoren konnte eine nur wenig schlechtere Vorhersage von 93.22% auf dem Testdatensatz erzielt werden.

Es hat sich gezeigt, dass SVM bereits auf einem vollen Satz von teilweise redundanten Deskriptoren sehr gute Vorhersagen liefern können. Aus Gründen der benötigten Rechenzeit und des Speicherplatzes kann es sinnvoll sein, die Anzahl der Eingabevariablen über eine PCA zu verringern. Wird die erklärte Varianz dabei auf einen hohen Wert gesetzt (im Beispiel 98%), so ist eine Verschlechterung der Klassifizierungsrate nicht zu erwarten. Ferner hat sich gezeigt, das sich SVM mit RBF-Kernel gut für eine Deskriptorselektion mit dem Greedy-Verfahren eignen. Bei dem betrachteten Datensatz konnte die Menge an Deskriptoren mit einer solchen Selektion von 146 auf acht reduziert werden, bei nur geringer Verschlechterung der Klassifizierungsrate (von 98.31% auf 94.92%). Ein Nachteil der Deskriptorselektion liegt darin, dass von

verschiedenen Verfahren meist unterschiedliche Deskriptoren ausgewählt werden. Je mehr Deskriptoren im Ausgangsdatensatz vorhanden sind (je größer damit die Redundanz im Datensatz ist), desto stärker macht sich dieser Effekt bemerkbar. Eine leichte Stabilisierung der Auswahl kann erreicht werden, wenn nur die von mehr als einem Verfahren ausgewählten Deskriptoren berücksichtigt werden.

4.3 Klassifizierung nach Wirkstoffähnlichkeit

In diesem Abschnitt soll versucht werden, potentielle Wirkstoffe von nicht-wirksamen chemischen Substanzen zu unterscheiden. Dass eine solche Unterscheidung überhaupt möglich ist, mag auf den ersten Blick überraschend wirken. Schließlich ist das Hauptmerkmal eines Wirkstoffes die Bindung an einen von vielen im menschlichen Körper vorkommenden Rezeptoren (vgl. Abschnitt 2.4). Die Struktur der meisten dieser Rezeptoren ist noch nicht aufgeklärt, so dass eine generelle Aussage hinsichtlich der Bindungsfähigkeit schon aus diesem Grund nicht möglich ist.

Neben der Rezeptorbindung muss ein Wirkstoff allerdings noch weitere Kriterien erfüllen, die unter dem Begriff ADME/Tox zusammengefasst sind. So muss der Wirkstoff unter anderem erst einmal den Ort der Wirkung erreichen können. Hier spielt die orale Verfügbarkeit und eventuell die Fähigkeit zur Durchdringung der Blut-Hirn-Schranke eine Rolle.

Es ist anzunehmen, dass bei einer Klassifikation zwischen *drugs* und *non-drugs* eher diese sekundären, jedoch auch sehr wichtigen Kriterien erfasst werden. Der Begriff der Wirkstoffähnlichkeit (*drug-likeness*) wird von WALTERS & MURCKO [WM02] wie folgt definiert:

> *The phrase 'drug-like' (...) generally means 'molecules which contain functional groups and/or have physical properties consistent with the majority of known drugs.'*

Hierbei wird schon deutlich, dass sich eine Definition dieses Begriffes stets nur an den bereits bekannten Arzneistoffen orientieren kann. Deren Anzahl wird von LIPINSKI *et al.* [LLDF97] im Jahr 1997 auf rund 10 000 geschätzt. Im Vergleich zu den im organischen chemischen Raum vorhandenen über 10^{18} Substanzen [CP00] handelt es sich dabei um einen verschwindend geringen Wert.

Eine Möglichkeit zum Erkennen von potentiellen Wirkstoffen ist ein nützliches und an mehreren Stellen des *drug design cycles* einsetzbares Hilfsmittel. Eine hundertprozentige Genauigkeit kann hierbei jedoch nicht erwartet werden, da das Konzept der „Ähnlichkeit“ zu einem Wirkstoff notwendigerweise nur schwammig und ungenau definiert werden kann. Zum einen ist bisher nur ein kleiner Teil der Menge der potentiellen Wirkstoffe bekannt. Jede Vorhersage muss auf diesen bekannten Wirkstoffen basieren, völlig neuartige, bislang unentdeckte Wirkprinzipien bleiben dabei unberücksichtigt. Zum anderen ist auch die Zugehörigkeit einer Substanz zur Gruppe der Wirkstoffe nicht eindeutig definierbar. Substanzen können mehr oder weniger stark an verschiedenen Rezeptoren binden und dabei (positive) Heilwirkungen oder (negative) Nebenwirkungen entfalten.

Trotz der genannten Einschränkungen ist nach CLARK & PICKETT [CP00] ein Drug-non-Drug-Klassifikator an folgenden Stellen des *drug design cycles* einsetzbar:

Einkauf von Substanzbibliotheken Aus den Katalogen der von Zulieferfirmen angebotenen Substanzen sollen bevorzugt diejenigen ausgewählt werden, welche eine gewisse Ähnlichkeit mit Wirkstoffen aufweisen (*cherry picking*). Das Abgrenzungskriterium darf hierbei allerdings nicht zu eng gefasst werden, um neue Substanzklassen und Indikationsgebiete nicht von vorneherein auszuschließen.

Priorisierung von Substanzen für HTS-Läufe Aus einer sehr großen, historisch gewachsenen Substanzbibliothek soll eine verkleinerte Teilbibliothek ausgewählt werden. Bei den durchgeführten HTS-Läufen wird aus Kostengründen dann nur noch die Teilbibliothek routinemässig eingesetzt.

Optimierung von kombinatorischen Bibliotheken Oft ist es aus Kosten- oder Zeitgründen nicht möglich, alle der über einen bestimmten kombinatorischen Ansatz zugänglichen Substanzen zu synthetisieren. Bei der Auswahl einer optimierten Teilbibliothek kann die Wirkstoffähnlichkeit als eines von mehreren Kriterien mit eingehen.

Eine grobe Klassifizierung wäre bereits mit den in der *rule of five* von Lipinski enthaltenen Aussagen bezüglich der Molekülgröße und der Anzahl der Donoren und Akzeptoren möglich (vgl. Abschnitt 2.4.3). Allerdings werden nur relativ wenige der in Bibliotheken angebotenen Substanzen tatsächlich aussortiert, so dass es sich um ein recht schwaches Kriterium handelt. Ein Überblick über bereits bekannte Ansätze zur Vorhersage der Wirkstoffähnlichkeit wird in Abschnitt 2.4.6 gegeben.

In den folgenden Abschnitten werden der verwendete Datensatz beschrieben, die generelle Vorgehensweise erklärt und schließlich die erzielten Klassifizierungsraten diskutiert.

4.3.1 Beschreibung des Datensatzes

Der untersuchte Datensatz besteht aus insgesamt 9 034 Molekülstrukturen. Von diesen sind 3 347 Substanzen der Klasse „Drug“ zugeordnet, und entsprechend 5 687 Substanzen der Klasse „NonDrug“. Die den Wirkstoffen zugeordneten Substanzen wurden dabei einer *in-house* Patent-Datenbank entnommen (Stand 1998), die als nicht-wirkstoffähnlich angesehenen Substanzen wurden zufällig aus dem ACD ausgewählt.[9]

Deskriptoren: Es wird eine Menge von 285 Substruktur-Deskriptoren verwendet. Jedem dieser Deskriptoren liegt eine bestimmte Substruktur zugrunde, deren Auftrittshäufigkeit in den untersuchten Strukturen als ganzzahliger Wert bestimmt wird. Es

[9]Der Datensatz wurde dankenswerterweise von Dr. Andreas Dominik (ALTANA Pharma AG) im Rahmen eines SOL-Vorprojekts zusammengestellt.

Verfahren	Modul	Klassifizierungsrate [%] Training	Test	Kreuzval.
One Rule	Weka	76.77	77.51	76.77
C4.5	Weka	93.06	86.85	85.86
Naive Bayes, ohne DA	Weka	75.88	76.27	75.66
Naive Bayes, mit DA	Weka	82.14	81.81	81.67
LVQ, 5 CBV	LVQ_PAK	75.42	76.18	75.93
LVQ, 10 CBV	LVQ_PAK	80.52	80.52	80.12
LVQ, 20 CBV	LVQ_PAK	80.53	80.30	80.37
MLP, RProp, 2 vN	SNNS 4.2	89.37	86.45	85.18
MLP, RProp, 3 vN	SNNS 4.2	87.69	84.73	85.08
MLP, SCG, 2 vN	SNNS 4.2	89.14	86.59	85.59
MLP, SCG, 3 vN	SNNS 4.2	89.08	87.12	85.82
kNN, $k = 1$	Weka	98.80	85.13	84.83
C-SVM, RBF, $C = 100$	LIBSVM	93.55	90.00	**89.09**
C-SVM, RBF, $C = 140$	LIBSVM	93.92	**90.35**	89.00

Tabelle 4.12: Auf dem Datensatz erzielte Klassifizierungsraten. Bei den Neuronalen Netzen (MLP) wurde die Anzahl der versteckten Neuronen (vN) variiert, es wurde jeweils mit 1000 Lernzyklen trainiert.

handelt sich dabei um 106 von Ghose & Crippen für die Vorhersage von $\log P$ verwendete Substrukturen, bei denen jeweils die Umgebung eines einzelnen Atoms beschrieben wird. Zusätzlich wurden 116 Ringstrukturen, 14 lineare Atomketten und 49 sonstige Molekülgruppen definiert.[10]

Vorverarbeitung: Die Deskriptoren wurden keiner Skalierung oder Normierung unterworfen, da sie bereits in einem wohldefinierten Wertebereich liegen und eine vergleichbare Varianz aufweisen.

Validierung: Es werden jeweils 6 775 Strukturen für das Training verwendet, 2 259 Strukturen bilden einen unabhängigen Testdatensatz (rund 25%). Die Auswahl der Testdaten erfolgte in gleichmässigen Abständen. Zusätzlich wird für jedes Ergebnis eine 10-fache Kreuzvalidierung mit gleichmässiger Aufteilung der Patterns berechnet.

4.3.2 Berechnungen mit allen Deskriptoren

In Tabelle 4.12 sind die mit allen 285 Deskriptoren erzielten Ergebnisse dargestellt. Dabei ist zu beachten, dass durch einen einfachen *majority predictor*, der stets nur

[10] Es wurde ursprünglich von einer größeren Anzahl von Substrukturen ausgegangen. In einem Filterschritt wurden alle Substrukturen verworfen, die nicht oder nur einmal im gesamten Datensatz auftraten.

die Klasse mit den meisten Strukturen vorhersagt (in diesem Fall die Klasse Non-Drug), bereits ein Klassifizierungsrate von 62.95% erreicht werden kann (aufgrund der gleichmässigen Aufteilung gilt dieser Wert sowohl für die Trainings-, als auch für die Testdaten). Dieser Wert stellt also eine triviale untere Grenze für die folgenden Vorhersagen dar. Eine etwas komplexere untere Grenze kann mit dem *one rule only*-Verfahren (1Rule) berechnet werden. Hierbei wird nur ein einzelner Deskriptor für die Vorhersage verwendet.[11] Die Vorhersage von 1Rule liegt mit 77.51% auf dem Testdatensatz relativ hoch und übertrifft bereits andere Verfahren wie Naive Bayes ohne Dichteabschätzung und LVQ mit einer geringen Anzahl von Codebook-Vektoren.

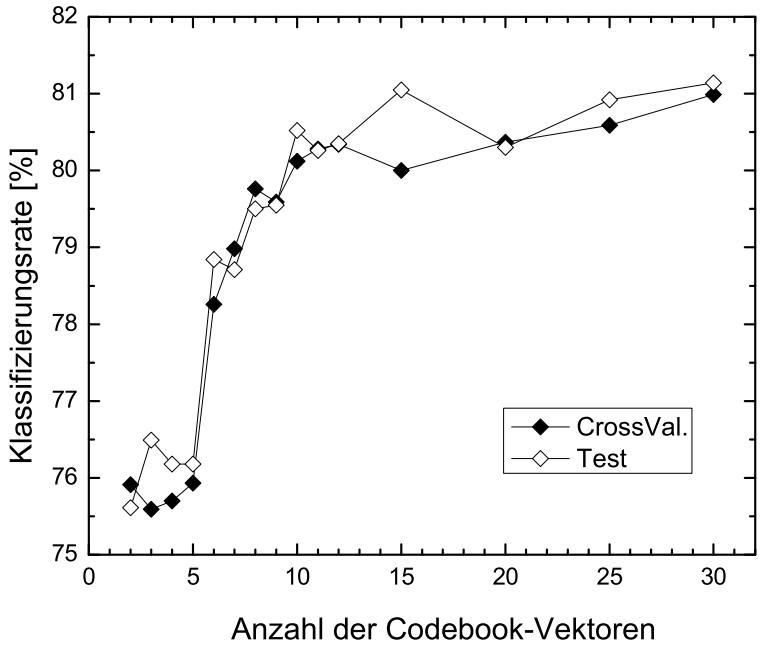

Abbildung 4.6: Abhängigkeit der Klassifizierungsrate von der Anzahl der Codebook-Vektoren beim LVQ-Verfahren. Die Anzahl der Trainingsschritte betrug für das initiale Training 20 000, für das verfeinernde Training 100 000.

Die beste Vorhersage auf dem Testdatensatz (90.35%) wird von dem C-SVM-Verfahren mit RBF-Kernel und einem Kostenfaktor $C = 140$ erreicht. Etwas schlechtere Klassifizierungsraten werden von den Verfahren C4.5, kNN mit einem Nachbar und den Feedforward-Netzen (MLP) erreicht. Mit Naive Bayes und LVQ können dagegen nur mittelmässige Vorhersagen getroffen werden. Die mit LVQ erzielte Klassifizierungsrate hängt dabei von der Anzahl der eingesetzten Codebook-Vektoren ab (vgl.

[11] Das 1Rule-Verfahren entspricht einem Entscheidungsbaum der Tiefe 1, siehe [WF00].

Abb. 4.6), wobei sich die Klassifizierungsrate ab 10 Codebook-Vektoren langsam stabilisiert und nicht mehr wesentlich verbessert.

	Klassifizierungsrate [%]					
	Training		Test		Kreuzval.	
Verfahren	NonD.	Drug	NonD.	Drug	NonD.	Drug
1Rule	80.87	69.80	81.65	70.49	80.87	69.80
C4.5	96.06	87.97	92.05	78.02	91.21	76.77
Naive Bayes, mit DA	87.08	73.75	87.34	72.40	86.68	73.15
LVQ, 10 CBV	81.45	78.92	81.36	79.09	81.92	77.05
MLP, RProp, 2 vN	93.11	83.03	90.93	78.85	90.57	76.02
MLP, RProp, 6 vN	94.72	87.41	91.42	81.36	90.01	79.64
MLP, SCG, 2 vN	92.71	83.07	90.65	79.69	90.01	78.09
MLP, SCG, 3 vN	93.08	82.27	91.49	79.69	90.41	78.01
kNN, $k = 1$	98.66	99.04	91.63	74.07	90.50	75.18
C-SVM, RBF, $C = 100$	96.32	88.84	93.95	83.27	93.60	81.43
C-SVM, RBF, $C = 140$	96.58	89.40	94.16	**83.87**	93.41	**81.51**

Tabelle 4.13: Auf dem Datensatz erzielte Klassifizierungsraten. Bei den Neuronalen Netzen (MLP) wurde die Anzahl der versteckten Neuronen (vN) variiert, es wurde jeweils mit 1000 Lernzyklen trainiert.

Eine Betrachtung der unterschiedlichen Fehlerarten (Fehler erster oder zweiter Art bzw. falsch positiv oder falsch negativ) wird durch die Tabelle 4.13 ermöglicht. Hier ist für die bislang erfolgreichsten Verfahren die Klassifizierungsrate jeweils für die beiden Klassen NonDrug und Drug aufgeschlüsselt. Es fällt auf, dass die NonDrug-Klasse von allen Verfahren deutlich besser vorhergesagt wird. Dies kann zum einen daran liegen, dass mehr nicht-wirkstoffähnliche Strukturen im Datensatz vorliegen. Zum anderen ist die Klasse der NonDrugs möglicherweise leichter abzugrenzen als die Klasse der Drugs. Betrachtet man nun die Vorhersage der Drug-Klasse für den Testdatensatz, so wird das beste Ergebnis von C-SVM (RBF-Kernel, $C = 140$) erzielt, dicht gefolgt von MLP (RProp, 6 versteckte Neuronen). Das kNN-Verfahren liefert eine relativ schlechte Vorhersage für die Klasse Drug, es kann somit für die betrachtete Problemstellung nicht zu den erfolgreichen Verfahren gezählt werden.

4.3.3 Berechnungen mit abgeleiteten Variablen

Um die Anzahl der Eingabevariablen zu reduzieren, wird eine Hauptkomponentenanalyse (PCA) auf den 285 Substruktur-Deskriptoren berechnet. Es wird bestimmt, wieviele der Hauptkomponenten notwendig sind, um einen vorgegebenen Anteil der Varianz im Datensatz zu erklären. Setzt man die zu erklärende Varianz auf 98%, so

werden im konkreten Beispiel die ersten 36 Hauptkomponenten benötigt. Diese abgeleiteten Variablen werden im folgenden als Eingabe für die bereits im letzten Abschnitt eingesetzten Verfahren verwendet.

		Klassifizierungsrate [%]		
Verfahren	Modul	Training	Test	Kreuzval.
One Rule	Weka	79.54	72.91	71.69
C4.5	Weka	96.22	83.13	81.70
Naive Bayes, ohne DA	Weka	71.70	70.56	71.59
Naive Bayes, mit DA	Weka	76.40	75.34	75.38
LVQ, 5 CBV	LVQ_PAK	75.44	76.27	76.06
LVQ, 10 CBV	LVQ_PAK	80.56	80.70	79.99
LVQ, 20 CBV	LVQ_PAK	81.15	80.83	80.72
PART	Weka	87.73	81.85	80.49
MLP, RProp, 2 vN	SNNS 4.2	84.13	83.80	83.07
MLP, RProp, 3 vN	SNNS 4.2	85.58	85.08	84.04
MLP, RProp, 4 vN	SNNS 4.2	85.51	84.55	83.26
MLP, SCG, 2 vN	SNNS 4.2	82.72	82.47	82.60
MLP, SCG, 3 vN	SNNS 4.2	85.15	85.66	84.07
MLP, SCG, 4 vN	SNNS 4.2	86.97	85.79	85.40
kNN, $k = 1$	Weka	98.80	87.16	85.06
C-SVM, RBF, $C = 1$	LIBSVM	91.39	88.89	87.87
C-SVM, RBF, $C = 8$	LIBSVM	95.68	**89.82**	**88.24**
C-SVM, RBF, $C = 10$	LIBSVM	95.99	89.69	88.13

Tabelle 4.14: Auf den mit PCA abgeleiteten Variablen erzielte Klassifizierungsraten. Die MLP wurden jeweils für 1000 Lernzyklen mit einem Ausgabeneuron trainiert, die Klassenzugehörigkeit wurde mit einem Schwellwert von 0.5 bestimmt.

In Tabelle 4.14 sind die auf den abgeleiteten Variablen erzielten Klassifizierungsraten dargestellt. Die besten Vorhersagen auf dem Testdatensatz werden von C-SVM mit RBF-Kernel erreicht, ebenfalls gute Vorhersagen liefert das MLP mit 3-4 versteckten Neuronen und kNN mit einem Nachbarn.

Betrachtet man für diese Verfahren nun wieder die Vorhersage der einzelnen Klassen Drug und NonDrug anhand von Tabelle 4.15, so fällt auf, dass das Verfahren kNN mit einem Nachbar eine relativ schlechte Vorhersage für die Klasse Drug auf den Testdaten liefert. Die besten Vorhersagen liefert wiederum das C-SVM-Verfahren mit RBF-Kernel, gefolgt von dem mit SCG trainierten MLP mit 3 versteckten Neuronen.

	Klassifizierungsrate [%]					
	Training		Test		Kreuzval.	
Verfahren	NonD.	Drug	NonD.	Drug	NonD.	Drug
1Rule	84.69	70.80	79.82	61.17	79.48	58.45
C4.5	97.23	94.50	88.26	74.43	87.08	72.55
MLP, RProp, 3 vN	90.69	76.89	90.37	76.11	88.89	75.82
MLP, SCG, 3 vN	88.51	79.44	88.26	81.24	88.44	76.65
kNN, $k = 1$	98.66	99.04	93.18	76.94	91.21	74.62
C-SVM, RBF, $C = 1$	94.61	85.94	92.62	82.56	92.17	80.56
C-SVM, RBF, $C = 8$	97.87	91.95	93.74	**83.15**	92.71	**80.64**
C-SVM, RBF, $C = 10$	98.19	92.23	93.67	82.92	92.64	80.48

Tabelle 4.15: Auf den mit PCA abgeleiteten Variablen erzielte Klassifizierungsraten, aufgeschlüsselt nach den beiden Klassen Drug und NonDrug.

4.3.4 Ergebnis

Bei der Vorhersage der Wirkstoffähnlichkeit mit den eingesetzten Substruktur-Deskriptoren ist darauf zu achten, dass die Klassifizierungsraten auf den Klassen Drug und NonDrug nicht zu stark voneinander abweichen. Einige Verfahren (wie kNN mit einem Nachbar) liefern zwar insgesamt gesehen eine gute Vorhersage, jedoch auf der Klasse Drug oft ein deutlich schwächeres Ergebnis. Als das beste Verfahren in diesem Vergleich hat sich das C-SVM mit RBF-Kernel erwiesen. Mit allen 285 Eingabedeskriptoren konnte auf dem unabhängigen Testdatensatz eine Klassifizierungsrate von 90.35% erreicht werden, wobei die Drug-Klasse zu 83.87% korrekt vorhergesagt wurde. Werden als Eingabe die mit Hilfe einer Hauptkomponentenanalyse abgeleiteten Variablen verwendet, so verschlechtert sich die Gesamt-Klassifizierungsrate leicht auf 89.82%, und entsprechend die Vorhersage der Drug-Klasse auf 83.15%.

4.4 Vorhersage der Wasserlöslichkeit

Die Wasserlöslichkeit (*aqueous solubility*, $\log S$) spielt bei der Suche nach neuen, potentiellen Arzneistoffen eine bedeutende Rolle. Sie beeinflusst die Aufnahme einer Substanz z. B. nach einer oralen Verabreichung, deren Verteilung im Körper und den Transport in Organe wie Leber und Niere. Mit der Wasserlöslichkeit einer Substanz lässt sich also in gewissem Umfang auch ihre Bioverfügbarkeit vorhersagen.

Die Wasserlöslichkeit $\log S$ weist eine hohe Korrelation mit dem Octanol-Wasser Partitionskoeffizienten $\log P$ auf (vgl. Abb. 4.7). Daher geht in vielen Verfahren zur Vorhersage der Wasserlöslichkeit ein auf unterschiedliche Arten berechneter $\log P$-Wert mit ein. Mit den zusätzlich gewählten Deskriptoren wird in diesem Fall hauptsächlich ein Korrekturterm berechnet, der $\log P$ in $\log S$ transformiert. Der Vorteil dieses Ansatzes ist darin zu sehen, dass es bereits zahlreiche erprobte Methoden zur Vorhersage von $\log P$ gibt. Der Nachteil liegt darin, dass eine schlechte Vorhersage von $\log P$ fast zwangsläufig ebenfalls eine schlechte Vorhersage von $\log S$ zur Folge hat. Aus diesem Grunde wird bei den folgenden Vergleichen von Deskriptorarten jeweils darauf hingewiesen, ob berechnete $\log P$-Werte als Eingabe verwendet werden oder ob auf diese verzichtet wird.

4.4.1 Beschreibung des Datensatzes

Es wird der in der Literatur weit verbreitete Datensatz von HUUSKONEN [Huu00] verwendet. Er besteht aus 1297 Substanzen, welche aus zwei Moleküldatenbanken (AQUASOL und PHYSPROP) zusammengestellt wurden. Die Löslichkeit S wurde experimentell bei $20-25°C$ bestimmt. Die gemessenen Werte haben die Einheit $\mathrm{mol/l}$, sie liegen in transformierter Form als $\log S$ vor.

Der ursprüngliche Datensatz enthält einige Substanzen, die mehrfach auftreten oder aus anderen Gründen herausfallen. Daher werden in den meisten Arbeiten über diesen Datensatz eine geringe Anzahl von Substanzen ausgeschlossen. Hier wird eine Teilmenge von 1269 Substanzen verwendet. Von diesen werden 1016 zum Training und 253 zum Testen verwendet.[12] Die Werte der Deskriptoren wurden jeweils mit $x' = (x - x_{\mathrm{mean}})/x_{\mathrm{sdev}}$ zentriert und auf eine einheitliche Standardabweichung normiert.

Als zweiter Testdatensatz wird eine von YALKOWSKI [YB92] vorgeschlagene Menge von 21 Substanzen verwendet, welche u. a. aus Wirkstoffen und Pestiziden besteht. Dieser Datensatz wird in praktisch allen Studien zum Thema als Gütekriterium herangezogen. Leider sind darin für einen statistisch signifikanten Vergleich zu wenige Testmuster enthalten (siehe Abschnitt 2.3). Ein qualitativer Wertevergleich zwischen verschiedenen Ansätzen kann jedoch als sinnvoll angesehen werden.

[12] Die Aufbereitung des Datensatzes und die Berechnung von 199 Deskriptoren wurde dankenswerterweise von Jörg Wegner (Uni Tübingen) durchgeführt.

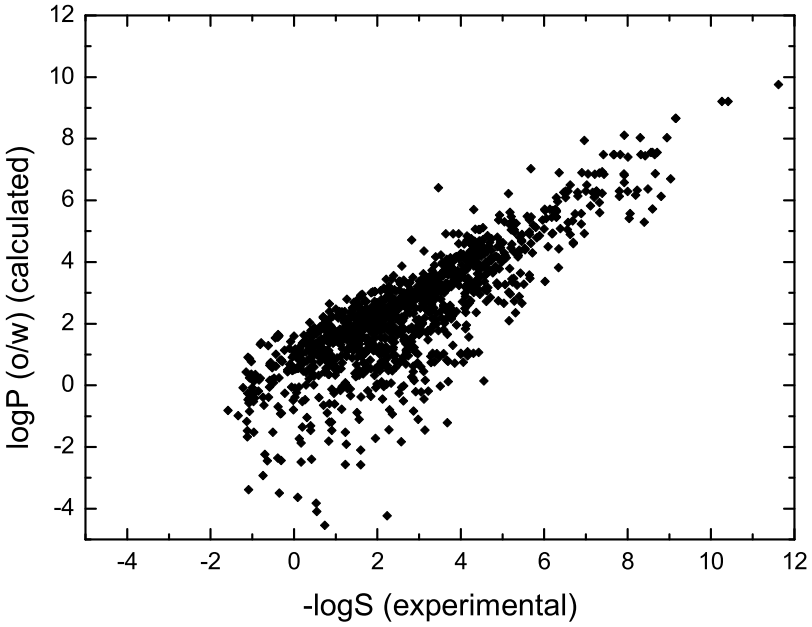

Abbildung 4.7: Die (negierte) experimentelle Wasserlöslichkeit $-\log S$ aufgetragen gegen den mit MOE berechneten Octanol-Wasser Partitionskoeffizienten $\log P$. Eine Korrelation zwischen den beiden Variablen ist erkennbar, der Korrelationskoeffizient r^2 beträgt 0.626.

4.4.2 Berechnungen mit allen Deskriptoren

Zuerst soll der volle Deskriptorsatz (199 Deskriptoren) als Eingabe verwendet werden. Im nächsten Abschnitt werden Methoden zur Selektion von Deskriptoren betrachtet.

In Tabelle 4.16 sind die Ergebnisse von Berechnungen mit Feedforward-Netzen (MLP) und dem ν-SVM-Verfahren mit RBF-Kernel dargestellt. Dabei wurde das MLP jeweils 10 000 Epochen lang trainiert. Als Lernverfahren kamen RProp und SCG zum Einsatz. Als Netztopologie wurde ein vollverdrahtetes Netz ohne Shortcuts mit einer versteckten Schicht eingesetzt, die Anzahl der Neuronen in der versteckten Schicht wurde variiert. Bei den ν-SVM wurden geeignete Werte für C und ν über eine systematische Veränderung von ν im Bereich $(0, 1)$ mit Schrittweite 0.05 bei $C = 1$,$C = 2$ und $C = 5$ ermittelt. In Abb. 4.8 ist die mit dem ν-SVM-Verfahren mit RBF-Kernel ($C = 5$, $\nu = 0.65$) erzielte Vorhersage in einem True-Predicted-Plot dargestellt. Unter den Testdatenpunkten ist eine geringe Anzahl von „Ausreissern“ erkennbar, deren vorhergesagter Wert vergleichsweise stark vom „wahren“ Wert abweicht. Hierbei könnte es sich jedoch auch um falsch überlieferte Werte oder Messfehler im Testdatensatz

Verfahren	RMSE Training	RMSE Test	Korrelationskoeff. r^2 Training	Korrelationskoeff. r^2 Test
MLP, RProp, 2 vN	0.4422	0.6317	0.9530	0.9054
MLP, RProp, 3 vN	0.3599	0.7574	0.9689	0.8649
MLP, RProp, 4 vN	0.3449	0.8380	0.9714	0.8398
MLP, SCG, 2 vN	0.4270	0.6527	0.9562	0.8968
MLP, SCG, 3 vN	0.3298	0.7265	0.9739	0.8750
MLP, SCG, 4 vN	0.3389	0.8988	0.9724	0.8239
ν-SVC, RBF, $C=5$, $\nu=0.65$	0.3157	0.5660	0.9761	**0.9219**
ν-SVC, RBF, $C=2$, $\nu=0.9$	0.4159	0.5674	0.9588	0.9215

Tabelle 4.16: Ergebnisse von Berechnungen mit dem vollen Deskriptorsatz (199 Deskriptoren) mit Feedforward-Netzen (MLP) und ν-SVM mit RBF-Kernel. Bei den MLP wurde das Lernverfahren und die Anzahl der versteckten Neuronen (#vN) variiert.

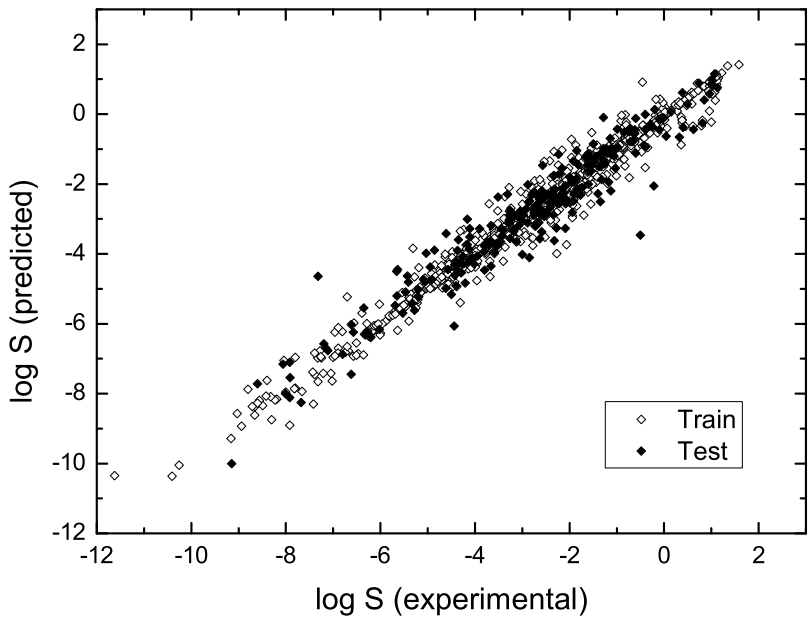

Abbildung 4.8: True-Predicted-Plot der Vorhersage mit dem ν-SVM-Verfahren mit RBF-Kernel, mit $C=5$ und $\nu=0.65$. Als Eingabedaten wurden alle 199 Deskriptoren verwendet.

handeln. Insgesamt gesehen weisen die Testdaten in Abb. 4.8 eine sehr gute Übereinstimmung ohne erkennbare systematische Abweichungen auf.

4.4.3 Selektion von Deskriptoren

Betrachtet man anhand von Tabelle 4.17 die Ergebnisse, die mit einer Teilmenge der vorhandenen Deskriptoren erzielt werden, so fällt zum einen auf, dass die Anzahl der ausgewählten Deskriptoren innerhalb einer großen Bandbreite variiert. So werden von der M5-Selektionsmethode 96 von 199 Deskriptoren ausgewählt, die Greedy-Methode verwendet dagegen nur 11-13 Deskriptoren. Als Abbruchkriterium für das Greedy-Verfahren wurde jeweils eine mindestens erforderliche Verbesserung (*minimal improvement rate*) von 1% des RMSE-Fehlers auf der Trainingsmenge verwendet. Bei der linearen Regression ohne expliziter Variablenselektion ist die Anzahl der Variablen angegeben, die mit einem Gewichtungsterm $\neq 0$ in die Gleichung eingehen. Das „Greedy/W"-Verfahren ist direkt im Weka-Programmpaket verfügbar und unterscheidet sich vor allem in Hinsicht auf das Abbruchkriterium von den anderen Greedy-Läufen. Gute Vorhersagen auf dem Testdatensatz können sowohl mit einer einfachen linearen Regression, als auch mit den wesentlich komplexeren Neuronalen Netzen vom feed-forward-Typ (MLP) erzielt werden. Es fällt jedoch auf, dass für das MLP-Verfahren deutlich weniger Deskriptoren benötigt werden, als für die lineare Regression.

			RMSE		Korrelationskoeff. r^2	
Verfahren	Selektion	#d	Training	Test	Training	Test
Lin. Regression	None	126	0.5310	0.6328	0.9323	0.9037
Lin. Regression	M5	96	0.5329	0.6239	0.9318	0.9060
Lin. Regression	Greedy/W	60	0.5389	0.6262	0.9303	0.9056
Lin. Regression	Greedy	13	0.6303	0.7001	0.9046	0.8824
MLP, RProp, 2 vN	Greedy	11	0.5781	0.6214	0.9198	0.9058
MLP, SCG, 2 vN	Greedy	12	0.5625	0.6369	0.9240	0.9011

Tabelle 4.17: Ergebnisse der Berechnungen mit einer Teilmenge der Deskriptoren. Angegeben sind jeweils das Regressionsverfahren, die Selektionsmethode und die Anzahl der ausgewählten Deskriptoren (#d).

4.4.4 Symbolische Regression

Mit Hilfe des Genetischen Programmierens (vgl. Abschnitt 2.2.3) kann eine sogenannte „symbolische Regression" durchgeführt werden. Hierbei werden die Funktions- und Terminalmengen so definiert, dass der Syntaxbaum einen arithmetischen Ausdruck

repräsentiert. Das Ziel ist nun, die Wasserlöslichkeit $\log S$ direkt über diese mit GP evolvierte Formel vorherzusagen.

Vorgehensweise: Die vollständige Menge der in diesem Abschnitt eingesetzten Funktionen lautet $F_{\max} = \{\mathrm{add}, \mathrm{sub}, \mathrm{mul}, \mathrm{logistic}, \tanh, \mathrm{gauss}\}$.[13] Dabei deutet die Bezeichnung $F_{\max}$ bereits an, dass in den nachfolgend beschriebenen GP-Läufen häufig Teilmengen $F' \subset F_{\max}$ eingesetzt werden. Als Grund für diese Vorgehensweise lässt sich anführen, dass durch die Entfernung von nicht benötigten Funktionen aus $F_{\max}$ der Suchraum verkleinert wird, und die Suche somit schneller konvergiert.

In den vorigen Abschnitten konnten mit einer linearen Regression, mit Neuronalen Netzen und SVM mit RBF-Kernel jeweils recht gute Vorhersagen erzielt werden. Daher werden die folgenden Teilmengen von $F_{\max}$ definiert:

- $F_{\text{linear}} = \{\mathrm{add}, \mathrm{sub}, \mathrm{mul}\}$ entspricht in Bezug auf den Funktionsumfang einer linearen Regression.
- $F_{\text{sigmoid}} = \{\mathrm{add}, \mathrm{sub}, \mathrm{mul}, \mathrm{logistic}, \tanh\}$ enthält zusätzlich die beiden sigmoiden Funktionen $\mathrm{logistic}$ und $\tanh$, welche auch bei Feedforward-Netzen (MLP) zum Einsatz kommen.
- $F_{\text{gauss}} = \{\mathrm{add}, \mathrm{sub}, \mathrm{mul}, \mathrm{gauss}\}$ enthält neben den linearen Funktionen die radialsymmetrische Gaußfunktion, welche ein Bestandteil von RBF-Netzen und von SVM mit RBF-Kernel ist.

Als Menge der Terminalsymbole T wird jeweils $T = \{\mathrm{desc}1, \ldots, \mathrm{desc}199, \mathrm{value}\}$ verwendet, wobei „$\mathrm{desc}\, n$“ den n-ten Deskriptorwert und „value“ eine jeweils zufällig gewählte, konstante reelle Zahl bezeichnet. Um das übermässige Wachstum der GP-Bäume zu verhindern, wird ein Bestrafungsterm (*node penalty*, NP) eingesetzt (vgl. Abschnitt 2.2.3.2). Dieser verschlechtert die Fitness eines Individuums pro Knoten im Baum um einen vorgegebenen Prozentsatz. Typische Werte für NP liegen bei 0.1 bis 0.5%.[14]

Als Selektionsmethode wird stets eine Tournament-Selektion verwendet. Die Generierung von neuen Individuen erfolgt mit jeweils einem von vier zur Verfügung stehenden Operatoren (vgl. Abschnitt 2.2.3.1). Es handelt sich dabei um Standard-Crossover, Standard-Mutation, Ein-Knoten-Mutation[15] und (unveränderte) Reproduktion. Als eine erfolgreiche Verteilung der Wahrscheinlichkeiten, mit denen die Operatoren angewendet werden, hat sich $P_{\text{crossover}} = 0.1$, $P_{\text{mutation}} = 0.3$, $P_{\text{one-node-mut}} = 0.5$ und

[13] Nicht verwendet werden die Funktionen div, exp und log, da sie die Bildung von nicht-funktionalen Teilbäumen begünstigen können (vgl. Abschnitt 2.2.3.2). Auch sin und cos werden ausgeklammert.

[14] Dieser Wertebereich für NP wurde empirisch bestimmt, hat sich jedoch für mehrere untersuchte Problemstellungen und Datensätze als sinnvoll erwiesen.

[15] Bei der Ein-Knoten-Mutation wird jeweils genau einer der Knoten im Baum durch einen passenden anderen Knoten ersetzt, der Rest des Baumes bleibt unverändert.

$P_{\text{reproduction}} = 0.1$ erwiesen. Der Crossover-Operator ist in dieser Verteilung relativ gering gewichtet, da sich bei der vorliegenden Aufgabe (Anpassung einer Kurve an Messwerte) nur selten sinnvolle Teilbäume zwischen Individuen austauschen lassen.

In Tabelle 4.18 sind die mit der symbolischen Regression erzielten Ergebnisse aufgeführt. Deutlich erkennbar ist der Einfluss des Betrafungsterms (NP) auf die Gesamtzahl der Knoten im Baum (#N), in geringerem Umfang auch auf die Anzahl der eingesetzten Deskriptoren (#d). Ein Zusammenhang mit dem Korrelationskoeffizienten r^2 auf der Testmenge ist dagegen nicht zu erkennen. Daraus lässt sich folgern, dass der Bestrafungsterm zu insgesamt kleineren Bäumen, aber nicht notwendigerweise zu einer schlechteren Vorhersage führt.

					RMSE		Korrelationskoeff. r^2	
Funkt.	NP [%]	Seed	#d	#N	Training	Test	Training	Test
F_{linear}	0.1	123	11	65	0.6832	0.7332	0.8894	0.8705
F_{linear}	0.1	100	12	49	0.6939	0.7878	0.8858	0.8574
F_{linear}	0.2	123	9	31	0.7367	0.7558	0.8697	0.8634
F_{linear}	0.2	100	11	37	0.6694	0.7395	0.8924	0.8675
F_{sigmoid}	0.1	123	16	50	0.6470	0.6848	0.8999	0.8909
F_{sigmoid}	0.1	100	13	51	0.6458	0.6736	0.8999	0.8894
F_{sigmoid}	0.2	123	8	24	0.6819	0.6765	0.8896	0.8886
F_{sigmoid}	0.2	100	13	48	0.6431	0.6722	0.9010	0.8898
F_{gauss}	0.2	123	7	25	0.7025	0.6966	0.8826	0.8818

Tabelle 4.18: Ergebnisse der symbolischen Regression für eine Populationsgröße von 1000 Individuen, jeweils 1000 Generationen. Variiert wurde die Funktionsmenge, der Bestrafungsterm pro Knoten im Baum (*node penalty*, NP) und der Startwert des Zufallszahlengenerators (*seed*). Angegeben sind die Anzahl der eingesetzten Deskriptoren (#d) und die Gesamtzahl der Knoten im Baum (#N).

4.4.5 Ergebnis

Auf der vollen Menge von Deskriptoren konnte mit dem ν-SVM-Verfahren mit RBF-Kernel ($C = 5$, $\nu = 0.65$) die beste Vorhersage erzielt werden. Es wurde ein Korrelationskoeffizient r^2 von 0.9219 auf den Testdaten erreicht. Aus der vollen Deskriptormenge konnten über verschiedene Arten der Deskriptorselektion Teilmengen von 11 bis 13 Deskriptoren ausgewählt werden. Die Vorhersage von $\log S$ auf dem Testdatensatz bleibt dabei vergleichbar gut (Korrelationskoeffizient $r^2 \approx 0.906$), wenn Neuronale Netze vom feed-forward-Typ (MLP) als Lernverfahren eingesetzt werden. Die direkte Ableitung von formelhaften Ausdrücken ist über eine GP-basierte, symbolische Regression möglich. Die hierbei erzielten Vorhersagen erwiesen sich jedoch im Vergleich mit MLP als etwas schlechter (Korrelationskoeffizient $r^2 \approx 0.891$).

Kapitel 5

Optimierung von Substruktur-Deskriptoren

In diesem Kapitel wird eine neuartige Methode zur Optimierung von Substruktur-Deskriptoren vorgestellt (vgl. Abschnitt 2.4.5.3). Das Verfahren trägt die Bezeichnung STEEDS (***S**MARTS **T**ree **E**ncoding for the **E**volution of **D**escriptor **S**ets*). Dabei werden Substrukturen in Form von Bäumen kodiert und mit Hilfe des Genetischen Programmierens (GP) für die jeweils vorgegebene Zielsetzung optimiert. Im Unterschied zu bisher bekannten Verfahren wird dabei eine streng-typisierte (*strongly typed*) GP-Variante eingesetzt (STGP). Sie erlaubt es, Molekülstrukturen je nach Baumtiefe auf verschiedenen Komplexitätsebenen zu repräsentieren. Damit wird es möglich, sowohl kleine Details (wie die Eigenschaften eines einzelnen Atoms) als auch große Fragmente (wie *backbones* oder lange Seitenketten) innerhalb einzelner Baumknoten darzustellen. Durch diese Eigenschaft ist das vorgestellte Verfahren besonders gut zur Optimierung von Substruktur-Mustern geeignet.[1]

Nach einer kurzen Motivation folgt eine ausführliche Beschreibung der Eigenschaften des STEEDS-Verfahrens. Im Abschnitt 5.3 werden Anwendungen im Wirkstoffdesign vorgestellt und die erzielten Ergebnisse diskutiert. Im letzten Abschnitt werden die Stärken und Schwächen des neuen Ansatzes im Vergleich zu bisher bekannten Ansätzen beschrieben.

5.1 Motivation

Bei den in Kapitel 4 vorgestellten Anwendungen von SOLVES stellte sich wiederholt die Frage, welche der zahlreich vorhandenen Deskriptoren sich am besten für die jeweilige Klassifizierung oder Regression eignet. Es wurde bereits deutlich, dass manche

[1] Unter einem Substruktur-Muster wird hier eine um *wildcards*, Auswahllisten und logische Operatoren erweiterte Substruktur verstanden.

Klassen von Deskriptoren anderen vorzuziehen sind, z. B. aus Gründen der benötigten Rechenzeit oder des zu hohen Abstraktionsniveaus. Die Klasse der Substruktur-Deskriptoren hat dabei gut abgeschnitten: Ihre Vertreter können mit geringem Aufwand berechnet werden und haben einen direkten Bezug zur Molekülstruktur. Ein weiterer, wesentlicher Vorteil dieser Art von Deskriptoren liegt in der nahezu unbeschränkten Variabilität der zugrunde liegenden Substruktur-Muster. Diese weitreichende Veränderbarkeit ist eine notwendige Voraussetzung für Ansätze zur Optimierung von Deskriptoren.

Eine direkte Optimierung von Deskriptoren bietet im Vergleich zu Selektionsverfahren die folgenden Vorteile:

- Es entfällt die Notwendigkeit, große Mengen von Deskriptorwerten im voraus zu berechnen (von denen die meisten nach der Selektion nicht mehr benötigt und daher in der Regel verworfen werden).
- Eine Selektion arbeitet stets auf einer fest vorgegebenen, endlichen Anzahl von Deskriptoren. Eine Vorauswahl dieser Grundmenge von Deskriptoren nach objektiven Kriterien ist schwierig, da praktische Überlegungen bezüglich Verfügbarkeit oder früheren Erfahrungen stets eine Rolle spielen. Bei einer Optimierung der Deskriptoren ist eine Vorauswahl dagegen nicht notwendig, es können auch völlig neue und unerwartete Substruktur-Muster entstehen.

Substruktur-Deskriptoren haben den Nachteil, dass die für die Bindung zwischen Ligand und Rezeptor wichtige dreidimensionale Struktur des Liganden unberücksichtigt bleibt. Sie haben dafür jedoch den aus der Sicht eines Chemikers wesentlichen Vorteil, noch „durchschaubar“ zu sein. D. h. wenn als Ergebnis einer QSAR-Studie optimale Werte für einen Satz von Substruktur-Deskriptoren ermittelt werden, so lassen sich diese vergleichsweise einfach in mögliche Molekülstrukturen übersetzen. Bei anderen Deskriptorarten (wie z. B. bei topologischen Deskriptoren) ist dies sehr viel schwieriger und führt zum sogenannten „inversen QSAR-Problem“.

Die Anzahl der für einen Deskriptor in Frage kommenden Substrukturen ist sehr groß. Sie umfasst zum einen den für pharmazeutische Wirkstoffe geeigneten Teil des „chemischen Raumes“.[2] Verwendet man dazu sogenannte *wildcards*, also Platzhalter, die für ein beliebiges Atom oder für eine Liste von Atomen stehen können (vgl. Abschnitt 2.4.7), so potenziert sich diese Anzahl noch einmal. Andererseits sind nur relativ kleine Molekülstrukturen als Deskriptoren verwendbar, da nur sie überhaupt in ausreichender Anzahl in einer Moleküldatenbank auftauchen. Um sinnvoll einsetzbar zu sein, sollte eine untere Schranke der Auftrittshäufigkeit nicht unterschritten werden. Diese kann z. B. so gewählt werden, dass eine Substruktur in mindestens 5% aller Substanzen vorkommen muss.

[2] Die Größe dieses chemischen Teilraumes (beschränkt auf relativ kleine, organische Substanzen) wird zwischen 10^{18} [CP00] und 10^{200} geschätzt [GE03, S. 602].

5.1.1 Zielsetzungen

Mit der vorgestellten Methode lassen sich verschiedene Zielsetzungen verfolgen. Zum einen kann nach einer größeren Anzahl von einfachen Substruktur-Mustern gesucht werden, mit dem Ziel der Optimierung von Deskriptorsätzen. Zum anderen können aber auch wenige Muster mit höherer Komplexität von Interesse sein, wobei das Ziel die Erlangung von strukturellen Informationen über eine Molekülklasse ist.

Optimierung von Deskriptorsätzen: Eine notwendige Voraussetzung für die Optimierbarkeit eines Deskriptors ist, dass mindestens einer der in die Berechnung einfließenden Parameter innerhalb eines weiten Wertebereichs variabel ist. Viele Arten von Deskriptoren haben eine feste Rechenvorschrift und somit keine adaptierbaren Eingabeparameter (vgl. Abschnitt 2.4.5). Bei den Deskriptoren mit einem oder mehreren Parametern ist Adaption meist nur innerhalb eines geringen Wertebereichs möglich (z. B. die Korrelationsweite bei Autokorrelations-Deskriptoren). Zu den wenigen Deskriptorarten mit einem großen adaptierbaren Wertebereich zählen aus oben genannten Gründen die Substruktur-Deskriptoren. Da diese einzeln gesehen nur wenig aussagekräftig sind, ist für die meisten Problemstellungen ein Zusammenwirken von mehreren Deskriptoren notwendig.[3] Eine mögliche Anforderung an einen Deskriptorsatz ist dabei die Orthogonalität bzw. völlige Unkorreliertheit der Deskriptoren untereinander. Da viele maschinelle Lernverfahren (wie Neuronale Netze und SVM) gut mit redundanten Informationen umgehen können, handelt es sich dabei um eine wünschenswerte, nicht aber um eine zwingend notwendige Anforderung.[4]

Suche nach Mustern mit höherer Komplexität: Unterschiedliche Molekülklassen können sich bezüglich der Diversität der zugehörigen Molekülstrukturen stark unterscheiden. Ein Beispiel für eine „eng gefasste“ Molekülklasse sind die Thrombin-Inhibitoren (vgl. Abschnitt 4.2). Eine solche strukturelle Homogenität kann beispielsweise in der Starrheit der Bindetasche des zugehörigen Rezeptors begründet sein. Im Extremfall kann eine einzige Substruktur für die Abgrenzung einer homogenen Molekülklasse von anderen Klassen ausreichend sein. Bei dieser Substruktur muss es sich nicht notwendigerweise um das komplette Grundgerüst (*scaffold*) der Molekülklasse handeln. Oft ist ein kleiner, aber charakteristischer Teil des *scaffolds* ausreichend. Auch bei weniger homogenen Molekülklassen, wenn also kein allen Strukturen gemeinsames Grundgerüst vorhanden ist, kann die vorgestellte Methode Vorteile bieten, da weder die Anzahl noch die Größe der gesuchten Fragmente von vorneherein festgelegt ist. Ein möglicher Nachteil liegt darin, dass bei Auftreten von zu vielen *wildcards* in den gefundenen Mustern der praktische Nutzen (im Sinne der oben erwähnten „Durchschaubarkeit“) vermindert wird.

Eine entscheidend wichtige Frage bei dieser zweiten Zielsetzung ist, ob die gefunde-

[3] Beispielsweise werden von HOU *et al.* [HXZX04] insgesamt 76 Substruktur-Deskriptoren und zwei Korrekturterme zur Vorhersage der Wasserlöslichkeit ($\log S$) eingesetzt.

[4] Auf Unkorreliertheit der Deskriptoren muss dagegen geachtet werden, wenn eine einfache lineare Regression eingesetzt wird.

nen Fragmente verallgemeinerbar sind, ob sie also tatsächlich ein wesentliches Merkmal der jeweiligen Molekülklasse beschreiben. Liegen in der Optimierungsphase nur wenige Moleküle für jede Klasse vor, so besteht die Gefahr, dass ein nur zufällig in allen Molekülen auftretendes, aber für die Wirkstärke nebensächliches Fragment ermittelt wird. Werden dagegen ausreichend viele Moleküle eingesetzt, so reduziert sich dieser Unsicherheitsfaktor. Es kann erwartet werden, dass der Erfolg dieses letztendlich auf die Suche nach „2D-Pharmakophoren“ hinauslaufenden Ansatzes stark vom jeweils untersuchten Datensatz abhängt. Einerseits reagiert die im dreidimensionalen Raum stattfindende Wechselwirkung zwischen Ligand und Rezeptor sehr empfindlich auf geringe strukturelle Veränderungen im Bereich der Bindetasche, welche das Verfahren nicht erfassen kann. Andererseits können, je nach vorliegender Substanzklasse, auch nicht-offensichtliche Gemeinsamkeiten in den Strukturen vorliegen, die von dem Verfahren erkannt werden können.

5.2 Beschreibung des Verfahrens

Mit Hilfe des STEEDS-Verfahrens werden Substruktur-Muster evolviert, welche für die Berechnung von Substruktur-basierten Deskriptoren eingesetzt werden. Die Substruktur-Muster werden mit Hilfe der SMARTS-Notation spezifiziert. Die Optimierung der SMARTS-Patterns erfolgt über das Verfahren der streng-typisierten Genetischen Programmierung (STGP).[5] Im Unterschied zu einem Standard-GP, bei dem alle Kombinationen von Eltern- und Kindknoten erlaubt sind, sind bei einem STGP nur bestimmte Kombinationen möglich. Diese erlaubten Kombinationen werden in einer Menge von Regeln definiert.

Im vorliegenden Fall ist dies notwendig, da die durch den Baum repräsentierten Substrukturen hierarchisch aufgebaut sind: Knoten im unteren Bereich enthalten einzelne Atomeigenschaften, darüber stehen Knoten, die komplette Atome enthalten, und im oberen Bereich werden Seitenketten oder Ringstrukturen abgebildet. Der Vorteil des hierarchischen Aufbaus liegt darin, dass einerseits eine Substruktur komplett aus einzelnen Atomeigenschaften aufgebaut werden kann. Damit ist sichergestellt, dass prinzipiell jede chemisch sinnvolle Substruktur dargestellt werden kann. Andererseits können aber auch Knoten definiert werden, die bereits ganze Atome oder Ringsysteme enthalten.[6] Somit kann chemisches Wissen über „sinnvolle“ Bestandteile einer Substruktur gezielt mit eingebracht werden (z. B. Akzeptoren oder Donoren). Das Verfahren muss diese grundlegenden Bestandteile nicht bei jedem Lauf wieder neu „erfinden“ und kann sich daher auf die problemspezifischen Anforderungen konzentrieren.

[5] Die Umsetzung des STEEDS-Verfahren basiert auf der in dem ECJ-Programmpaket enthaltenen STGP-Implementierung (von Sean Luke, George Mason-Universität, Virginia) und auf der SMARTS-Pattern-Suchfunktion von JOELib (Jörg K. Wegner, Uni Tübingen).

[6] Bei diesen vordefinierten Knoten handelt es sich in der Regel um Blattknoten. Prinzipiell können aber auch innere Knoten mit zusätzlichen Informationen angereichert werden.

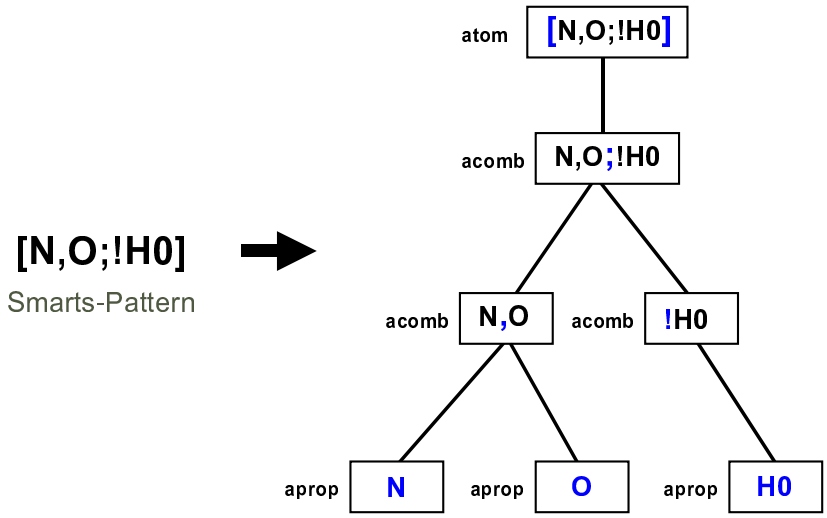

Abbildung 5.1: Das links dargestellte SMARTS-Pattern definiert ein einzelnes Atom, auf der rechten Seite ist die zugehörige Baumstruktur angezeigt.

Als Fitnessfunktion wird meist die Klassifizierungsrate verwendet. In der Regel reicht ein einzelner Substruktur-Deskriptor nicht aus, da als Eingabe für ein Klassifizierungsverfahren stets mehrere Deskriptoren benötigt werden (von trivialen Fällen einmal abgesehen). Bei der Erzeugung von mehreren Deskriptoren lassen sich die folgenden beiden Vorgehensweisen unterscheiden.

Bei einer „seriellen“ Vorgehensweise wird immer nur ein Deskriptor gleichzeitig evolviert. Wenn das Abbruchkriterium erreicht ist, wird der neue Deskriptor in die Menge der erzeugten Deskriptoren aufgenommen. Die Fitnessfunktion wird stets mit dem aktuell evolvierten und allen bereits vorhandenen Deskriptoren gemeinsam berechnet. Dadurch wird das Ergebnis schrittweise verfeinert. Diese Vorgehensweise entspricht einer Greedy-Strategie bei einer Deskriptorselektion (vgl. Abschnitt 4.1). Eine solche Vorgehensweise ist vergleichsweise effizient und die Berechnung der Fitnessfunktion ist einfach. Allerdings besteht die Gefahr, im Laufe des Verfahrens in ein lokales Optimum zu gelangen, da einmal getroffene Entscheidungen nicht mehr rückgängig gemacht werden können.

Bei einer „parallelen“ Vorgehensweise wird dagegen versucht, eine Menge von Deskriptoren gleichzeitig zu optimieren. In diesem Fall müssen entweder mehrere Bäume von dem Verfahren verwaltet werden, oder es werden mehrere SMARTS-Patterns innerhalb eines Baumes kombiniert. Dies kann beispielsweise mit logischen Funktionen geschehen, wenn das Vorhandensein eines SMARTS-Patterns im Molekül als Wahrheitswert interpretiert wird. Die Anzahl der Deskriptoren kann entweder von vorneherein festgelegt werden, oder sie kann variabel gehalten werden. Letzteres hat den Vorteil, dass sich die Komplexität der Lösung der inhärenten Komplexität der Pro-

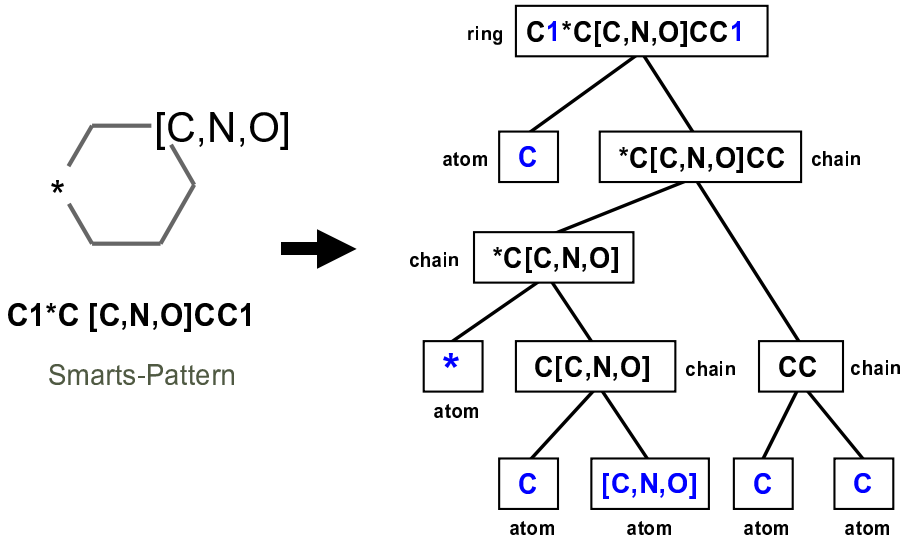

Abbildung 5.2: Die Baumstruktur für das links gezeigte SMARTS-Pattern, welches eine Ringstruktur darstellt. Der oberste Knoten im Baum bewirkt den Ringschluss.

blemstellung anpassen kann.

Werden mehrere Bäume gleichzeitig evolviert, so reicht ein einzelner Wert als Fitnessfunktion allerdings nicht mehr aus, da dieser die gemeinsame Leistung aller Deskriptoren, nicht aber den Beitrag des jeweils betrachteten einzelnen Deskriptors bewerten würde. Um den einzelnen Deskriptoren einen eigenen Fitnesswert zuzuweisen, kann die Klassifizierung mehrmals durchgeführt werden, wobei jeweils einer der Deskriptoren ausgelassen wird. Zuletzt wird noch einmal mit allen Deskriptoren klassifiziert. Der Beitrag eines Deskriptors besteht aus der Differenz des gemeinsamen Ergebnisses und des ohne seine Beteiligung erzielten Wertes. Ein Vorteil dieses Bewertungsschemas liegt darin, dass die Entstehung von sehr ähnlichen Deskriptoren vermieden wird: Je ähnlicher sich zwei Deskriptoren sind, desto kleiner wird ihr jeweiliger Beitrag zum Gesamtergebnis.

5.2.1 Evolution eines einzelnen Patterns

In diesem Abschnitt soll die für die Evolution eines einzelnes SMARTS-Patterns angewandte Vorgehensweise beschrieben werden. Wie in Abschnitt 5.3.1 gezeigt wird, können mit solchen einzelnen Patterns bereits einfache Aufgabenstellungen gelöst werden. Für die meisten praktischen Aufgaben sind jedoch Kombinationen von mehreren SMARTS-Patterns erforderlich, welche im nächsten Abschnitt beschrieben werden.

Die Festlegung der Beziehungen zwischen Eltern- und Kindknoten für das STGP-Verfahren besteht aus mehreren Teilen. Als erstes müssen die grundlegenden Knoten-

typen definiert werden (Tab. 5.1). Diese können zu Gruppen zusammengefasst werden (Tab. 5.2). Als nächstes werden die Einschränkungen (*node constraints*) für die Knoten und Gruppen spezifiziert (Tab. 5.3). Schließlich werden die Knoten, welche die eigentliche Funktionalität enthalten, erzeugt (Tab. 5.4). Jedem der in dieser Tabelle aufgeführten Knoten wird eine Einschränkung zugewiesen, wodurch auch der Typ des Knotens definiert wird.

Name	Beschreibung	Beispiel
`aprop`	Einzelne Atomeigenschaft	H0
`acomb`	Kombination von Atomeigenschaften	O,N
`atom`	Vollständiges Atom	[O,N;!H0]
`chain`	Kette von Atomen (mit Verzweigungen)	ccc(C)[H2]
`ring`	Ring aus Atomen	c1ccccc1

Tabelle 5.1: Die für die Repräsentierung von SMARTS-Patterns verwendeten Knotentypen.

In Tabelle 5.1 sind die grundlegenden Knotentypen aufgelistet. Für jeden Knotentyp ist ein Beispiel für ein SMARTS-Pattern, das in dem jeweiligen Knoten enthalten sein kann, angegeben.

Name	Beschreibung	Knotentypen
`APropComb`	Enthält einzelne Atomeigenschaften und Kombinationen von Eigenschaften	`aprop` `acomb`

`AtomRingChain`	Enthält einzelne Atome, lineare oder verzweigte Atomketten und Ringe	`atom`
		`chain`
		`ring`

Tabelle 5.2: Die Definition der beiden Gruppen, in denen jeweils unterschiedliche Knotentypen zusammengefasst sind.

In Tabelle 5.2 sind die beiden in STEEDS definierten Gruppen von Atomtypen aufgeführt. Die Gruppe `APropComb` enthält alle „subatomaren“ Eigenschaften, die Gruppe `AtomRingChain` enthält alle Kombinationen von Atomen.

In Tabelle 5.3 sind die Einschränkungen aufgelistet, welche die Anzahl und den Typ der Kinder eines Knotens festlegen. Es werden nicht alle möglichen Kombinationen von Eltern- und Kindknoten definiert, sondern nur diejenigen, welche auch tatsächlich von vorhandenen Knoten verwendet werden.

Schließlich werden in Tabelle 5.4 die Einschränkungen der vorhandenen Knoten definiert. Im folgenden wird die jeweilige Funktion dieser Knoten kurz erläutert. Der `AtomPropertyNode` enthält einzelne Atomeigenschaften. Dazu zählen Atomnamen

Name	Typ	erstes Kind	zweites Kind	drittes Kind
nc0-aprop	aprop			
nc1-acomb	acomb	aprop		
nc2-acomb	acomb	APropComb	APropComb	
nc0-atom	atom			
nc1-atom	atom	APropComb		
nc0-chain	chain			
nc2-chain	chain	AtomRingChain	AtomRingChain	
nc3-chain	chain	AtomRingChain	AtomRingChain	AtomRingChain
nc1-aprop	aprop	chain		
nc1-ring	ring	atom		
nc2-ring	ring	atom	chain	

Tabelle 5.3: Die Einschränkungen, die für die Eltern-Kind-Beziehungen zwischen den einzelnen Knotentypen und Gruppen gelten.

Knotenname	Einschränkung	Knotenname	Einschränkung
AtomPropertyNode	nc0-aprop	NotNode	nc1-acomb
AtomCountPropNode	nc0-aprop	AndOrNode	nc2-acomb
PredefAtomNode	nc0-atom	PredefChainNode	nc0-chain
AtomNode	nc1-atom	BondNode	nc2-chain
RingNode	nc2-ring	ChainNode	nc3-chain
HeteroRingNode	nc1-ring	EnvironmentNode	nc1-aprop

Tabelle 5.4: Die Namen aller vorhanden STEEDS-Knoten und deren Einschränkungen.

wie z. B. Sauerstoff (O), aber auch abstraktere Eigenschaften wie Ringatom (R). In AtomCountPropNode sind zählbare Atomeigenschaften enthalten, wie z. B. die Anzahl der impliziten Wasserstoffatome (Hx) oder die Anzahl der expliziten Verbindungen (Dx). Ein NotNode negiert die in dem Kindknoten enthaltene Atomeigenschaft, z. B. „nicht Sauerstoff“ (!O). Der AndOrNode kombiniert die in den beiden Kindern enthaltenen Atomeigenschaften, z. B. „Sauerstoff oder Stickstoff“ (O,N). Die bisher genannten Knoten enthalten Atomeigenschaften, die folgenden enthalten ein oder mehrere Atome. Der AtomNode erzeugt ein Atom mit den in dem Kindknoten enthaltenen Eigenschaften. Der PredefAtomNode enthält eine Reihe von vordefinierten Eigenschaften, wie z. B. Donor ([O,N;!H0]). Der PredefChainNode enthält eine Reihe von vordefinierten Atomketten und Ringen. Dazu zählt auch der aus zwei Atomen bestehende Akzeptor (O=[C,N]). Der BondNode verbindet zwei Atome oder Atomketten, z. B. mit einer Doppelbindung. Der ChainNode erzeugt eine Verzweigung aus den in seinen drei Kindknoten enthaltenen Atomen oder Atomketten. Der RingNode besitzt

zwei unterschiedliche Arten von Kindknoten, ein einzelnes Atom und eine Atomkette, aus denen er einen Ring bildet. Der `HeteroRingNode` hat nur ein einzelnes Atom als Kindknoten, welches in einen vordefinierten Ring variabler Größe eingefügt wird. Eine besondere Eigenschaft besitzt der `EnvironmentNode`: Er bildet aus einer Atomkette wieder eine Atomeigenschaft, indem er die Atomkette als Atomumgebung (*atom environment*) definiert.

Neben diesen Definitionen von Knoten und Einschränkungen spielen noch weitere Faktoren eine Rolle. Eine große Bedeutung kommt den Belohnungs- oder Bestrafungstermen zu, mit denen erwünschtes Verhalten der Verfahrens verstärkt und unerwünschtes Verhalten unterdrückt werden kann. In Abschnitt 2.2.3.2 wird die generelle Neigung von GP-Verfahren zum übermässigen Wachstum der Individuen beschrieben. Als eine Form von *parsimony pressure* wird in STEEDS daher für jeden im Baum auftretenden Knoten der Fitnesswert um einen einstellbaren Prozentsatz verringert (oft wird 0.01% verwendet). Zusätzlich wird für jeden `NOT`-Operator im Baum der Fitnesswert um ein Zehntel der *node penalty* verringert. Dadurch soll die positive Darstellung eines Sachverhalts (Eigenschaft ist vorhanden) gegenüber einer negativen leicht bevorzugt werden.

Für das hier beschriebene Verfahren werden keine eigens definierten Mutations- oder Rekombinationsoperatoren benötigt, es können die jeweiligen Standard-Operatoren eingesetzt werden. Zusätzlich zu den in Abschnitt 2.2.3.1 beschriebenen Operatoren wird eine sogenannte Einknotenmutation (*one node mutation*) eingesetzt, welche einen Knoten im Baum zufällig auswählt und ihn durch einen anderen Knoten mit passenden Eltern- und Kindknoten ersetzt.

5.2.2 Binäre Klassifizierung

Eine binäre Klassifizierung (also eine Unterscheidung zwischen zwei Klassen von Molekülen) kann bereits mit einem einzelnen SMARTS-Pattern erfolgen, wenn das Vorhandensein des Patterns in einem Molekül als Entscheidungskriterium verwendet wird. Da diese Vorgehensweise nur für einfache Aufgabenstellungen erfolgreich ist, sind jedoch meistens Kombinationen von mehreren SMARTS-Patterns erforderlich.

Im folgenden wird das Auftreten eines SMARTS-Patterns in einem Molekül als Wahrheitswert (`boolean`) interpretiert, wobei die Anwesenheit des Patterns als `true` und entsprechend die Abwesenheit als `false` definiert wird. Diese Wahrheitswerte lassen sich mit den logischen Operatoren `AND`, `OR` und `NOT` verknüpfen, weitere Operatoren (wie z. B. `XOR`) sind denkbar. Die Menge der bei STGP eingesetzten Funktionen wird entsprechend erweitert, die Beziehungsregeln zwischen den Funktionen sind im folgenden dargestellt.

Es wird ein neuer Knotentyp names `bool` definiert, welcher von Knoten verwendet wird, die einen Wahrheitswert enthalten. Die Tabelle 5.5 erweitert die im letzten Ab-

Name	Knotentyp	erstes Kind	zweites Kind
nc0-bool	bool		
nc1-bool	bool	bool	
nc2-bool	bool	bool	bool
nc1-smarts-bool	bool	AtomRingChain	

Tabelle 5.5: Die Einschränkungen, die für die Eltern-Kind-Beziehungen zwischen Knoten vom Typ bool gelten.

Knotenname	Einschränkung	Knotenname	Einschränkung
BinaryOpNode	nc2-bool	BoolValueNode	nc0-bool
UnaryOpNode	nc1-bool	SmartsExistsNode	SmartsExistsNode

Tabelle 5.6: Die für die binäre Klassifizierung hinzukommenden Knoten und deren Einschränkungen.

schnitt in Tabelle 5.3 beschriebenen Einschränkungen. Die Namen der neu hinzukommenden Knoten sind in Tabelle 5.6 aufgeführt. Der BoolValueNode enthält einen konstanten Wahrheitswert. Der BinaryOpNode verknüpft zwei in den Kindern enthaltene Wahrheitswerte mit einem der Operatoren AND oder OR. Der UnaryOpNode besitzt ein Kind, dessen Wahrheitswert er mit dem NOT-Operator negiert. Der SmartsExistsNode stellt schließlich die Verbindung zwischen den SMARTS-Patterns und den boolschen Werten her. Er prüft, ob das in dem Kindknoten enthaltene Pattern in dem aktuell ausgewerteten Molekül vorhanden ist, und liefert entsprechend den Wert true oder false.

Diese relativ geringen Änderungen sind ausreichend, um anstelle von einem einzelnen SMARTS-Pattern eine beliebige Anzahl von Patterns gleichzeitig evolvieren zu können. Die Verbindung zwischen den Patterns wird allerdings auf eine recht einfache Weise hergestellt. Ob sich mit diesem Ansatz verwertbare Ergebnisse erzielen lassen, werden die Anwendungsbeispiele im nächsten Abschnitt zeigen.

5.3 Anwendungen

Im folgenden wird das vorgestellte STEEDS-Verfahren auf die in Kapitel 4 beschriebenen Datensätze angewendet. Ein wichtiges Ziel ist dabei die Validierung des Verfahrens anhand von realen Datensätzen. Daneben soll der bereits im vorhergehenden Kapitel begonnene Vergleich zwischen Substruktur-Deskriptoren und anderen Arten von Deskriptoren weiter verfolgt und vertieft werden.

Wenn nicht anders angegeben, werden die folgenden Parametereinstellungen verwen-

det. Der Wert des Knoten-Bestrafungsterms (*node penalty*) beträgt 0.01%, für die Einsatzwahrscheinlichkeit der Rekombinationsoperatoren gilt $P_{\text{crossover}} = 0.3$, $P_{\text{mutation}} = 0.3$, $P_{\text{one-node-mut}} = 0.3$ und $P_{\text{reproduction}} = 0.1$.

5.3.1 Klassifizierung einer chemischen Eigenschaft

Die erstmals in Abschnitt 4.1 vorgestellte einfache Aufgabe der Unterscheidung zwischen aromatischen, aliphatischen Molekülen und Alkoholen soll als erster Testfall dienen. Es kann erwartet werden, dass das Verfahren ohne große Probleme passende Substruktur-Deskriptoren findet. Interessant ist hierbei die Frage, wie schnell und vor allem wie reproduzierbar Ergebnisse gefunden werden.

Es wird die in Abschnitt 5.2.2 beschriebene binäre Klassifizierung eingesetzt, wobei jeweils eine Klasse gegen die beiden anderen abgegrenzt wird. Da es hier nur um eine erste Einschätzung des Verfahrens geht, werden im folgenden nur die Klassifizierungsraten auf dem Trainingsdatensatz miteinander verglichen.

Zuerst soll die Klasse `alcohol` betrachtet werden. Es kann vermutet werden, dass diese einfach zu klassifizieren ist, da nur in den Molekülen dieser Klasse Sauerstoffatome vorkommen (vgl. Tabelle C.1 im Anhang).

Seed	Klass. [%]	Ausdruck
100	100.00	[O,N;!H0]
150	100.00	O
200	100.00	[O,N;!H0]
250	100.00	[O,N;!H0]

Tabelle 5.7: Klassifizierung von `alcohol` gegen die beiden Klassen `aliphatic` und `aromatic`. Angegeben ist der *seed*, die Klassifizierungsrate auf den Trainingsdaten und der evolvierte Ausdruck. Populationsgröße 100, Anzahl Generationen 100, Operatoren `AND`, `OR` und `NOT`.

In Tabelle 5.7 sind die Ergebnisse von vier Läufen dargestellt. In allen Fällen wurde bereits nach wenigen Generationen eine korrekte Klassifizierung erreicht. Dabei wird jeweils nur ein einzelnes SMARTS-Pattern verwendet, eine Kombination von mehreren Pattern mit den zur Verfügung stehenden Operatoren ist also nicht notwendig. Auffällig ist, dass die Unterscheidung nur einmal direkt über ein Sauerstoffatom erfolgt, in drei Fällen dagegen über ein vordefiniertes Donor-Atom. Der Grund liegt darin, dass in STEEDS die Auftrittswahrscheinlichkeit von vordefinierten Gruppen etwas höher eingestellt ist als die von einzelnen Atomtypen. Für die Bewertungsfunktion des STEEDS-Verfahrens sind dagegen beide gleichwertig, daher wird das jeweils zuerst

auftretende Pattern als das beste angesehen.[7] Hierbei wird deutlich, dass das Verfahren aus mehreren möglichen Unterscheidungskriterien zufällig eines auswählt, solange keine Steuerung der Auswahl über Belohnungs- oder Betrafungsterme erfolgt.

	mit NOT-Operator		ohne NOT-Operator	
Seed	Klass. [%]	Ausdruck	Klass. [%]	Ausdruck
100	100.00	¬ [H1]	83.33	[H2]
150	100.00	¬ [H1]	86.11	[H2] ∨ [D0]
200	100.00	¬ [H1]	83.33	[H2]
250	100.00	¬ [H1]	86.11	[H2,D0]

Tabelle 5.8: Klassifizierung von aliphatic gegen die beiden zusammengefassten Klassen aromatic und alcohol. Für die links gezeigten Ergebnisse werden die Operatoren AND (∧), OR (∨) und NOT (¬) eingesetzt, rechts nur AND und OR. Populationsgröße 100, Anzahl Generationen 100.

In Tabelle 5.8 sind die für die Klasse aliphatic evolvierten Ausdrücke gezeigt. Die Klassifizierung erfolgt korrekt und reproduzierbar, wenn neben AND und OR auch der NOT-Operator zur Verfügung steht. Als Kriterium wird die Anzahl der Wasserstoffatome verwendet, da kein Molekül in der aliphatic-Klasse ein Atom mit einem einzelnen Wasserstoffatom besitzt, die Moleküle in den anderen Klassen dagegen schon. Wird der NOT-Operator ausgeschlossen, dann ist eine vollständig korrekte Klassifizierung nicht mehr möglich.[8] Hier kommt das Verfahren auf verschiedene Lösungen, wobei die besseren Lösungen aus Lauf zwei und vier denselben Sachverhalt unterschiedlich ausdrücken (einmal wird der OR-Operator auf Patternebene und einmal die Oder-Verknüpfung von SMARTS auf Atomebene verwendet).[9]

Im folgenden soll geprüft werden, ob das Ergebnis bei ausgeschlossenen NOT-Operator noch verbessert werden kann. Dazu wird in Tabelle 5.9 zum einen die Anzahl der Generationen, und zum anderen die Populationsgröße verzehnfacht. Es zeigt sich, dass eine vermehrte Anzahl von Generationen zu leichten Verbesserungen in Lauf eins und drei führen, alle vier Läufe bringen jetzt dasselbe Ergebnis. Bei einer Erhöhung der Populationsgröße dagegen ergeben sich auch „neue Ideen“, der beste Lauf liegt hier bei 91.67%.

In Tabelle 5.10 sind die für die Klasse aromatic generierten Ausdrücke gezeigt. Es fällt auf, dass das SMARTS-Pattern „c“ alleine nicht ausreicht, da auch bei den Alkoholen aromatische Ringe vorkommen. Eine korrekte Klassifizierung ergibt sich, wenn

[7] Dieses Verhalten kann bei Bedarf einfach über eine Änderung der Belohnungs- und Bestrafungsterme angepasst werden.

[8] Aufgrund der Beschränkungen von SMARTS gibt es keinen „positiven“ Ausdruck, mit dem z. B. Methan von den Molekülen der anderen Klassen unterschieden werden könnte.

[9] Im Gegensatz dazu haben die Ausdrücke „¬ [H1]“ und „[!H1]“ jedoch eine unterschiedliche Bedeutung.

	$\mu = 100$, $\lambda = 1000$		$\mu = 1000$, $\lambda = 100$	
Seed	Klass. [%]	Ausdruck	Klass. [%]	Ausdruck
100	86.11	[H2,D0]	88.89	[H2][H2] $\vee$ [D0]
150	86.11	[H2] $\vee$ [D0]	86.11	[D0,H2]
200	86.11	[H2,D0]	91.67	CCCCC $\vee$ C@A $\vee$ [D0]
250	86.11	[H2,D0]	86.11	[H2,D0]

Tabelle 5.9: Klassifizierung von `aliphatic` gegen die beiden zusammengefassten Klassen `aromatic` und `alcohol` mit den Operatoren `AND` und `OR`. Im linken und rechten Teil werden die Populationsgröße μ und die Anzahl der Generationen λ wie angegeben variiert.

	mit `NOT`-Operator		ohne `NOT`-Operator	
Seed	Klass. [%]	Ausdruck	Klass. [%]	Ausdruck
100	100.00	c1ccccc1 $\wedge \neg$ [O,N;!H0]	88.89	c1ccccc1
150	88.89	c	88.89	c1ccccc1
200	88.89	c	88.89	cccccc
250	100.00	$\neg$ [O,N;!H0] $\wedge$ cc	88.89	cc

Tabelle 5.10: Klassifizierung von `aromatic` gegen die beiden zusammengefassten Klassen `aliphatic` und `alcohol`. Der `NOT`-Operator wird nur auf der linken Seite verwendet, `AND` und `OR` auf beiden Seiten. Populationsgröße 100, Anzahl Generationen 100.

zusätzlich noch Sauerstoffatome ausgeschlossen werden (wie in Lauf 1 und 4 links in Tab. 5.10). Ohne den `NOT`-Operator ist dieser Ausschluss jedoch nicht möglich, daher kann auf der rechten Seite von Tabelle 5.10 keine vollständig richtige Klassifizierung erreicht werden. Wird die Anzahl der Individuen pro Generation wieder auf 1000 verzehnfacht, dann kommen mit `NOT`-Operator alle vier Läufe zum richtigen Ergebnis, ohne `NOT`-Operator ergibt sich dagegen keine Verbesserung (nicht gezeigt).

Ergebnis Bei dem in diesem Abschnitt betrachteten, einfachen Datensatz konnten mit dem STEEDS-Verfahren gute Ergebnisse erzielt werden. Es wurde unter vier Läufen jeweils mindestens eine vollständig richtige Klassifizierung gefunden, sofern diese mit den eingesetzten Operatoren und SMARTS-Pattern realisierbar war. Durch Erhöhung der Populationsgröße von 100 auf das 1000 ließ sich das durchschnittliche Ergebnis verbessern. Durch den in Abschnitt 5.2.1 beschriebenen Bestrafungsterm wurde das bei GP-Verfahren häufig auftretende übermässige Wachstum der Individuen (*bloat*) vermieden. Anhand der zuerst betrachteten Klasse `alcohol` wurde deutlich, dass das Verfahren bei mehreren, in Bezug auf den Bestrafungsterm gleichwertigen Lösungen eine zufällige Auswahl trifft. Das bedeutet, dass je nach Aufgabenstellung eine An-

passung bzw. Verfeinerung des Bestrafungsterms notwendig sein kann. Die Anzahl der gleichwertigen Lösungen hängt aber auch von der Größe des Datensatzes ab. Je mehr strukturell unterschiedliche Moleküle pro Klasse vorliegen, desto genauer kann das Verfahren zwischen ähnlichen Lösungen unterscheiden.

5.3.2 Klassifizierung nach Indikationsgebiet

Die folgenden Ausführungen beziehen sich auf den in Abschnitt 4.2 beschriebenen, von Michael Bieler zusammengestellten Datensatz. Er enthält insgesamt 299 Moleküle, welche sich in die vier Substanzklassen 5HT1A-Agonisten, H2-Antagonisten, Thrombin- und MAO-A-Inhibitoren aufteilen. Der Datensatz ist unterteilt in 240 Trainings- und 59 Teststrukturen. Es wird wieder die in Abschnitt 5.2.2 beschriebene binäre Klassifizierung eingesetzt, wobei jeweils eine Klasse gegen die drei anderen abgegrenzt wird.

Klasse	Seed	Ausdruck	Schematischer Ausdruck
Thrombin	100	[O,N;!H0]=Cc	DON =Cc
Thrombin	150	[O,N;!H0]=C(c1ccccc1)[O,N;!H0]	DON =C(∘C6) DON
Thrombin	200	C(=[O,N;!H0])cccc	C(= DON)cccc
Thrombin	250	O=[C,N] ∧ C(cc)[H1][O,N;!H0]	ACC ∧ C(cc)[H1] DON

Tabelle 5.11: Die für die Klasse Thrombin generierten Ausdrücke. Rechts ist jeweils eine schmematisierte Fassung des Ausdrucks gezeigt, bei der ACC für Akzeptor, DON für Donor und ∘C6 für einen aromatischen Kohlenstoff-Sechs-Ring stehen.

Zuerst soll die Klasse Thrombin betrachtet werden. Wie sich bereits in Abschnitt 4.2 gezeigt hat, ist diese Klasse besonders einfach von den anderen abgrenzbar. In Tabelle 5.11 sind die in vier Läufen mit unterschiedlichen *seed*-Werten evolvierten Ausdrücke gezeigt. In allen vier Läufen wurde eine Klassifizierungsrate von 100% auf den Trainings- und auf den Testmolekülen erreicht. In den ersten drei Läufen wird nur ein einzelnes SMARTS-Pattern generiert, im vierten Ausdruck sind zwei Patterns über den `AND`-Operator verknüpft. Da besonders längere SMARTS-Patterns leicht unübersichtlich werden können, wird im rechten Teil der Tabelle 5.11 eine schematisierte, vereinfachte Darstellung von SMARTS-Patterns eingeführt. Häufig auftretende Gruppen (wie Donor, Akzeptor oder Ringe) werden mit Hilfe von kleinen Kästchen dargestellt.[10]

Betrachtet man nun die generierten SMARTS-Patterns, so fällt auf, dass das zweite SMARTS-Pattern eine erweiterte Form des ersten darstellt. Auch das dritte Pattern

[10] Dabei ist zu beachten, dass es zwei leicht unterschiedliche Donor-Varianten gibt. Das Symbol DON steht für „[O,N;!H0]“, das Symbol DON′ (mit Strich) steht für „[O,N;!H0;!R]“.

Klasse	Seed	Klass.-Rate [%] Training	Test
H2-ANT	100	96.25	93.22
H2-ANT	150	96.25	98.31
H2-ANT	200	97.50	98.31
H2-ANT	250	98.33	98.31

Tabelle 5.12: Die für die Klasse H2-ANT erzielten Vorhersagen. Der *seed* wird variiert, die *node penalty* beträgt jeweils $p = 0.01\%$, die Populationsgröße 1000, als logische Operatoren werden `AND` und `OR` verwendet.

Klasse	Seed	Schematischer Ausdruck
H2-ANT	100	$(((\text{S} \vee \boxed{\circ\,\text{A5}}\ \boxed{\text{DON}}) \wedge \boxed{\circ\,\text{A5}}) \vee [\text{C\&X3}]\ \boxed{\text{DON}'}\ \text{CC})$ $\wedge \text{cc}[\text{X2}] \wedge [\text{H2}]$
H2-ANT	150	$\text{C}[\text{D2}]\ \boxed{\text{DON}}\ \text{C=A} \vee \boxed{\circ\,\text{A5}}\ \boxed{\text{DON}} \vee (\text{S} \wedge \boxed{\circ\,\text{A5}})$
H2-ANT	200	$((\boxed{\circ\,\text{A5}} \vee \boxed{\text{ACC}}\ \boxed{\text{DON}}) \wedge ([\text{a;X2}] \vee [\text{D2}]\ \boxed{\circ\,\text{A6}}))$ $\wedge (\text{S} \vee (\boxed{\text{DON}}\,\text{c} \wedge [\text{H2}]) \vee \boxed{\text{ACC}}\,[\text{H1}])$
H2-ANT	250	$(\text{C}[\text{H2}]\ \boxed{\text{DON}'}\ [\text{C\&X3}] \vee (\boxed{\circ\,\text{A5}} \wedge \text{S})) \vee ([\text{H2}] \wedge \boxed{\text{ACC}}\ \boxed{\circ\,\text{A5}}))$

Tabelle 5.13: Die für die Klasse H2-ANT generierten Ausdrücke. Gezeigt ist die schmematisierte Fassung des Ausdrucks, bei der $\boxed{\text{ACC}}$ für Akzeptor, $\boxed{\text{DON}}$ für Donor und $\boxed{\circ\,\text{A5}}$ für einen aromatischen Fünf-Ring stehen.

Klasse	Seed	Klass.-Rate [%] Training	Test
MAO-A	100	96.25	96.61
MAO-A	150	89.58	94.92
MAO-A	200	95.83	94.92
MAO-A	250	94.58	91.53

Tabelle 5.14: Die für die Klasse MAO-A erzielten Vorhersagen. Der *seed* wird variiert, die *node penalty* beträgt jeweils $p = 0.01\%$, die Populationsgröße 1000, als logische Operatoren werden `AND` und `OR` verwendet.

Klasse	Seed	Schematischer Ausdruck
MAO-A	100	([v2] ∧ [!C](c(cc)*)ccccc ∧ [!c](cc)c) ∨ ○ A5 ccccc
MAO-A	150	[!X2](○ C6)[R;H0] ∧ [R;H0&n,O] ∧ [!C](○ C6)[R;H0]
MAO-A	200	[D3](@[c,O])A cc ∨ [R]([H3]) ○ C6
MAO-A	250	[X2] [X2] ∨ (CC ∧ [D2](aaaaaa) ○ C6) ∨ ([X2] ∧ ○ A5)

Tabelle 5.15: Die für die Klasse MAO-A generierten Ausdrücke in der schematisierten Fassung.

stellt eine leicht erweiterte Version des ersten dar, bei dem zusätzlich die Verzweigung anders angeordnet ist. Das vierte Pattern fordert im Gegensatz zu den anderen eine Akzeptor-Gruppe, zudem wird die Doppelbindung am Kohlenstoff durch „ein Atom mit einem impliziten Wasserstoff“ ([H1]) umschrieben. Wie im letzten Abschnitt unterscheiden sich die mit unterschiedlichem *seed* generierten Patterns durch den Grad ihrer Generalität bzw. ihrer Spezifizität. Dies liegt auch hier wiederum in der relativen geringen Anzahl von Molekülen begründet, welche eine genauere Eingrenzung nicht zulässt. Allerdings stellt diese Variabilität der evolvierten Patterns auch einen Vorteil dar, da daraus Hinweise auf mögliche strukturelle Eigenschaften einer Substanzklasse abgeleitet werden können.

In der Tabelle 5.12 sind die Klassifizierungsraten für die Klasse H2-ANT aufgeführt. Es wird überwiegend eine sehr gute Vorhersage auf den Testdaten von 98.31% erreicht. Von den vier generierten Ausdrücken ist der im zweiten Lauf erzeugte am kürzesten und damit am nachvollziehbarsten (vgl. Tabelle 5.13). Hier zeigt sich, dass vergleichbare Klassifizierungsraten auf den Testdaten durch komplizierte oder auch durch relative einfache Ausdrücke erreicht werden können. Im Falle der komplizierten Terme hat sich der *bloat* bestrafende Term nicht durchsetzen können.

Auch für die Klasse MAO-A werden insgesamt gute Klassifizierungsraten erreicht (vgl. Tabelle 5.14). Betrachtet man anhand von Tabelle 5.15 die erzeugten Ausdrücke, so fällt auf, dass diese überwiegend erstaunlich kurz sind. Der im dritten Lauf generierte Ausdruck verbindet beispielsweise zwei einzelne SMARTS-Patterns mittlerer Komplexität mit dem `OR`-Operator. Da sich bei den in Abschnitt 4.2 durchgeführten Klassifizierungsversuchen vor allem in Bezug auf die Klassen MAO-A und 5HT1A Zuordnungsprobleme ergeben haben, wären auch komplexere Ausdrücke plausibel gewesen.

Während für die drei bisher betrachteten Klassen jeweils der `NOT`-Operator nicht zum Einsatz kam, sollen für die letzte Klasse 5HT1A noch einmal die sich durch den Einsatz des `NOT`-Operators ergebenden Unterschiede betrachtet werden. Von Tabelle 5.16 kann abgelesen werden, dass sich die Klassifikationsrate auf den Testdaten in allen vier Läufen verbessert. Allerdings ist diese Verbesserung mit einem Nachteil verbunden: In drei der vier Läufe mit NOT-Operator ergaben sich lange und komplexe Ausdrücke (mit jeweils mehr als 30 Knoten im Baum). Nur für den *seed* 150 ergab sich ein

Klasse	Seed	mit NOT-Operator Klass.-Rate [%] Training	Test	ohne NOT-Operator Klass.-Rate [%] Training	Test
5HT1A	100	96.67	94.92	97.50	91.53
5HT1A	150	92.92	89.83	91.25	84.75
5HT1A	200	97.50	88.14	94.58	86.44
5HT1A	250	98.33	94.92	90.83	84.75

Tabelle 5.16: Die für die Klassen 5HT1A erzielten Vorhersagen. Der *seed* wird variiert, die *node penalty* beträgt jeweils $p = 0.01\%$, die Populationsgröße 1000, als logische Operatoren werden AND und OR und auf der linken Seite zusätzlich NOT verwendet.

Klasse	Seed	Schematischer Ausdruck
5HT1A	100	aaaaaaaaaaaa ∨ NCCCC ○C6 ∨ ○A6 c ∨ N[R]*-aaaaaa ∨ nccN
5HT1A	150	[C&R] ∧ ¬(DON′ [!H2] ∧ [$(DON)])
5HT1A	150	a ○A6 ∨ (C([C;R])a ∧ [R] ○C6)
5HT1A	200	C@ DON ∨ ((C(@C) ○C6 ∧ CCCC) ∨ *-A@*-*-*@*-aaa)
5HT1A	250	DON @CC ∨ c ○A6 ∨ Fccc ∨ cccc@CC

Tabelle 5.17: Die für die Klasse 5HT1A generierten Ausdrücke in der schematisierten Fassung. Bei dem ersten der beiden Läufe mit *seed* 150 war der NOT-Operator zugelassen, bei allen anderen Läufen dagegen nicht.

kürzerer Ausdruck (vgl. Tabelle 5.17), wobei dieser Lauf eine etwas schlechtere Klassifizierungsrate als zwei der anderen Läufe mit NOT-Operator aufweist. Daraus lässt sich folgern, dass bei manchen Molekülklassen zwischen der reinen Klassifizierungsrate und der Verständlichkeit des generierten Ausdrucks abgewogen werden muss.

5.4 Ergebnis

Es wurde gezeigt, dass eine Evolvierung von SMARTS-Patterns mit dem STGP-Verfahren und entsprechenden Regelsätzen prinzipiell möglich ist. Ein Vorteil des Verfahrens liegt darin, dass Vorwissen (z. B. über häufig auftretende Gruppen wie Akzeptoren und Donoren) einfach und nachvollziehbar eingebracht werden kann. Durch eine Erweiterung der Funktions- und Regelmenge des STGP können komplexere Aufgabenstellung wie z. B. eine binäre Klassifizierung bearbeitet werden. Neben der einfachen Vorhersage von chemischen Eigenschaften konnte auch die anspruchsvollere Aufgabe der Unterscheidung zwischen vier Wirkstoff-Klassen gut gelöst werden. Aus den generierten Ausdrücken können zusätzlich wertvolle Informationen über die für

eine Molekülklasse typischen Substrukturen gewonnen werden. Zudem sind die vorgenommenen Klassifizierungen „durchschaubar“ und können von fachlichen Experten auf ihre Sinnhaftigkeit überprüft werden.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Erzielte Ergebnisse

Mit dem im Rahmen dieser Arbeit implementierten SOLVES-Programmsystem konnten die im Vorfeld formulierten Anforderungen umgesetzt werden (vgl. Abschnitt 3.2). Es wurde ein Programmsystem entwickelt, das die geforderten Eigenschaften der Modularität und Flexibilität erfüllt, das unterschiedlichste Methoden unter einer einheitlichen Benutzungsoberfläche integrieren kann, das Berechnungen verteilt und plattformunabhängig ausführen kann, und das große Datenmengen verarbeiten und die Ergebnisse dauerhaft archivieren kann. Bei den sich aus den Anforderungen ergebenden Spannungsfeldern konnte jeweils ein gangbarer Kompromiss gefunden werden (vgl. Abschnitt 3.3). Die wichtigsten Eigenschaften von SOLVES sind:

- Die Integration der innerhalb eines Anwendungsbereiches eingesetzten Methoden unter einer gemeinsamen Benutzungsoberfläche wird ermöglicht. Im Rahmen dieser Arbeit wurde dies beispielhaft für den Bereich des Liganden-basierten Wirkstoffdesigns demonstriert. Eine Erschließung von anderen Anwendungsbereichen erfordert lediglich die Einbindung von weiteren Modulen, nicht jedoch Änderungen am Hauptprogramm.
- Die verfügbaren Methoden können flexibel über eine graphisch-interaktive Benutzungsoberfläche zu Prozessketten zusammengestellt werden. Häufig benötigte Teil-Prozessketten können zu Makros zusammengefasst werden.
- Die automatisierte Verteilung der Berechnungen ist innerhalb einer heterogenen Rechnerumgebung möglich. Dabei werden ein Standalone-, ein Client-Server- und ein verteilter Modus unterstützt.
- Die Ergebnisse aller Berechnungen können automatisch archiviert werden und bleiben auch langfristig noch eindeutig und vollständig nachvollziehbar.

SOLVES wurde in verschiedenen Bereichen des Wirkstoffdesigns erfolgreich eingesetzt. Daraus konnten folgende Erkenntnisse gewonnen werden:

- Im Vergleich von verschiedenen Arten von Molekülдeskriptoren lassen sich mit den Substruktur-Deskriptoren gute Ergebnisse erzielen. Dies gilt besonders für die Vorhersage der Wirkstoffähnlichkeit (*drug likeness*) von Molekülen.
- Über eine Selektion der wichtigsten Deskriptoren lassen sich Erkenntnisse bezüglich des untersuchten Problems gewinnen. Dieser Erkenntnisgewinn ist bei Substruktur-Deskriptoren allerdings deutlich höher als bei anderen, abstrakteren Deskriptorenarten.

Zudem wurde mit dem STEEDS-Verfahren eine neue Methode zur Evolution von Molekülsubstrukturen vorgestellt, welche durch ihren hierarchischen Aufbau die Vorteile bisher bekannter Verfahren kombiniert und neue Möglichkeiten eröffnet. Das Verfahren konnte erfolgreich für eine binäre Klassifizierung von chemischen Eigenschaften eingesetzt werden. Zudem konnten Substanzen aus vier Indikationsgebieten mit dem Verfahren unterschieden werden. Es hat sich gezeigt, dass die von STEEDS generierten Substrukturmuster wertvolle Hinweise auf relevante strukturelle Eigenschaften der untersuchten Molekülklassen geben können.

Insgesamt gesehen konnten mit Hilfe von SOLVES mehrere für die aktuelle Pharmaforschung relevante Problemstellungen erfolgreich bearbeitet werden. Die Lösung von vergleichbaren als auch von andersgearteten Aufgaben wird für den Praktiker durch die Bereitstellung des ausgereiften Programmsystems SOLVES erleichtert.

6.2 Weiterführende Arbeiten

Allerdings kann SOLVES im Moment noch nicht den Bedürfnissen aller potentiellen Anwender entgegenkommen. Ohne ein grundlegendes Verständnis der Vor- und Nachteile der eingebundenen Methoden ist eine zielgerichtete Vorgehensweise nur schwer möglich, was jedoch als ein generelles Problem beim Einsatz von komplexen Auswerteverfahren angesehen werden kann.

Da die für die Lösung eines Problems benötigten Prozessketten nur einmal zusammengestellt werden müssen und abgespeichert werden können, reicht es aus, wenn eine Person mit entsprechenden Kenntnissen innerhalb oder außerhalb einer Arbeitsgruppe zur Verfügung steht. Trotzdem ist eine Verbesserung und Vereinheitlichung der Dokumentation der eingebundenen Methoden wünschenswert.

Das Programmsystem SOLVES wurde mit dem Ziel erstellt, den Einsatz von Neuronalen Netzen und Evolutionären Algorithmen in praxisrelevanten Anwendungsgebieten wie der Pharmaforschung zu erleichtern.

- Für den anwendungsbezogenen Forscher steht der einfache und flexible Zugriff auf unterschiedliche Verfahren zur Datenauswertung in Kombination mit der stets notwendigen Validierung im Vordergrund. Mit SOLVES können neue Probleme schneller bearbeitet und die Bearbeitung von bekannten Problemstellungen automatisiert werden.

- Für den Entwickler von neuen Verfahren ist ein direkter Vergleich mit anderen, bereits existierenden Methoden von Interesse. Zudem erleichtert SOLVES die Bewertung neuer Methoden unter statistischen Gesichtspunkten, da Prozessketten für umfangreiche Testläufe erstellt werden können.

SOLVES schlägt also eine Brücke zwischen dem Methodenentwickler und dem Anwender, im Rahmen dieser Arbeit zwischen der angewandten Informatik und der Chemie bzw. Pharmazie.

Bedingt durch den allgemein gehaltenen Ansatz ist eine Fortführung der Entwicklung von SOLVES in viele Richtungen denkbar. Besonders hervorzuheben sind die im folgenden aufgeführten Punkte:

Umfassendere Dokumentation Gerade der praktische Anwender benötigt eine umfangreiche und leicht verständliche Dokumentation. SOLVES bietet bereits die Möglichkeit, bei der Modulerstellung die Dokumentation gleich mitzuliefern. Dies beschränkt sich bisher allerdings meist auf kurze Anmerkungen mit dem Hinweis auf Online-Manuals, Literaturstellen und Bücher. Wünschenswert wäre dagegen eine einheitliche, umfangreiche Dokumentation sämtlicher Module.

Weitere Modularten Neben dem Standard-Modul sind in SOLVES bisher nur zwei erweiterte Modultypen vorhanden: Das Kontroll-Modul, welches die Kontrolle über genau ein anderes Modul übernehmen kann, und das Makro-Modul, welches einen kompletten linearen Workflow enthält. Die sich aus dem Zusammenspiel dieser beiden Typen entstehenen Möglichkeiten haben sich im Rahmen der Arbeit als ausreichend erwiesen. Weitere Modultypen sind jedoch leicht vorstellbar:

1. Ein erweitertes Kontroll-Modul, welches zwei oder mehr abhängige Module steuern kann.
2. Ein Modul zur Unterstützung von sequentiellen Methoden des Datenimports und -exports. Bisher müssen sämtliche Daten vor der Berechnung ins Archiv importiert werden. Die Verarbeitung von Datensätzen, die sich in der Größenordnung des zur Verfügung stehenden Hauptspeichers bewegen, wird dadurch erschwert oder ganz unmöglich. Ein Modul zum sequentiellen Import von Datenblöcken kann hier Abhilfe schaffen, sofern eine schrittweise Abarbeitung der Daten sinnvoll ist.

Neue Anwendungsgebiete Die Funktionalität von SOLVES wird wesentlich von der Art und Anzahl der dem Benutzer zur Verfügung stehenden Module bestimmt. Durch das Einbinden von weiteren Modulen können zum einen die Einsatzmöglichkeiten auf dem in dieser Arbeit betrachteten Gebiet der Chemoinformatik noch erweitert werden, zum anderen können auch völlig neue Einsatzgebiete (wie z. B. die Bioinformatik) erschlossen werden.

Seit Anfang des Jahres 2005 steht SOLVES Anwendern aus dem akademischen Bereich kostenfrei zur Verfügung. Es kann unter der Internet-Adresse http://www.solves.org heruntergeladen werden. Zum gegenwärtigen Zeitpunkt muss es sich erst noch erweisen, ob sich der allgemein gehaltene Ansatz von SOLVES auszahlt und sich auch Anwender außerhalb des Bereichs Chemie / Pharmazie finden werden.

Literaturverzeichnis

[ABC+01] S. Anzali, G. Barnickel, B. Cezanne, M. Krug, D. Filimonov, and V. Poroikov. Discriminating between drugs and nondrugs by prediction of activity spectra for substances (PASS). *J. Med. Chem.*, 44:2432–2437, 2001.

[AWM98] Ajay, W. P. Walters, and M. A. Murcko. Can we learn to distinguish between 'drug-like' and 'nondrug-like' molecules? *J. Med. Chem.*, 41:3314–3324, 1998.

[BBR+04] N. Baurin, R. Baker, C. Richardson, I. Chen, N. Foloppe, A. Potter, A. Jordan, S. Roughley, M. Parratt, P. Greaney, D. Morley, and R. E. Hubbard. Drug-like annotation and duplicate analysis of a 23-supplier chemical database totalling 2.7 million compounds. *J. Chem. Inf. Comput. Sci.*, 44:643–651, 2004.

[BBS+02] M. Brüstle, B. Beck, T. Schindler, W. King, T. Mitchell, and T. Clark. Descriptors, physical properties, and drug-likeness. *J. Med. Chem.*, 45:3345–3355, 2002.

[Bäc96] T. Bäck. *Evolutionary Algorithms in Theory and Practise*. Oxford University Press, New York, Oxford, 1996.

[BH03] M. Berthold and D. J. Hand, editors. *Intelligent Data Analysis*. Springer-Verlag, Berlin, second edition, 2003.

[BKK96] H.-J. Böhm, G. Klebe, and H. Kubinyi. *Wirkstoffdesign*. Spektrum Akademischer Verlag, Heidelberg, 1996.

[BL02] W. Banzhaf and W. B. Langdon. Some considerations on the reason for bloat. *Genetic Programming and Evolvable Machines*, 3:81–91, 2002.

[BM97] R. D. Brown and Y. C. Martin. Designing combinatorial library mixtures using a genetic algorithm. *J. Med. Chem.*, 40:2304–2313, 1997.

[BNKF98] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, editors. *Genetic Programming - An Introduction*. Morgan Kaufmann, San Francisco, CA, 1998.

[BVM97] T. Bäck, D. B. Vogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. IOP Publishing Ltd and Oxford University Press, Bristol, Oxford, 1997.

[BZB+96] H. Bauknecht, A. Zell, H. Bayer, P. Levi, M. Wagener, J. Sadowski, and J. Gasteiger. Locating biologically active compounds in medium-sized heterogeneous datasets by topological autocorrelation vectors: Dopamine and benzodiazepine agonists. *J. Chem. Inf. Comput. Sci.*, 36:1205–1213, 1996.

[Cec98] A. Cechin. *The Extraction of Fuzzy Rules from Neural Networks*. Shaker Verlag, Aachen, 1998. Dissertation, Eberhard-Karls-Universität Tübingen.

[CL02] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. Technical report, Department of Computer and Information Engineering, National Taiwan University, 2002.

[CP00] D. E. Clark and S. D. Pickett. Computational methods for the prediction of 'drug-likeness'. *Drug Discovery Today*, 5(2):49–58, 2000.

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, 2000.

[Dar59] C. Darwin. *The Origin of Species by Means of Natural Selection or the Preservation of Favored Races in the Struggle for Life*. Murray, London, 1859.

[De 75] K. A. De Jong. *An analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[Dic45] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26:297–302, 1945.

[DM85] G. C. Derringer and R. L. Markham. A computer-based methodology for matching polymer structure with required properties. *Journal of Applied Polymer Science*, 30:4609–4617, 1985.

[Dom96] A. Dominik. *Implementierung und Anwendung eines Konzepts zur Automatisierung des Computer Aided Drug Design*. Dissertation, Eberhard-Karls-Universität Tübingen, 1996.

[DTG00] D. Douguet, E. Thoreau, and G. Grassy. A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm. *Journal of Computer-Aided Molecular Design*, 14:449–466, 2000.

[ES03] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, 2003.

[FBN+00] T. M. Frimurer, R. Bywater, L. Nærum, L. N. Lauritsen, and S. Brunak. Improving the odds in discriminating 'drug-like' from 'non drug-like' compounds. *J. Chem. Inf. Comput. Sci.*, 40:1315–1324, 2000.

[FOW66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley, Chichester, UK, 1966.

[FPBG99] D. Filimonov, V. Poroikov, Y. Borodina, and T. Gloriozova. Chemical similarity through multilevel neighborhoods of atoms: Definition and comparison with the other descriptors. *J. Chem. Inf. Comput. Sci.*, 39:666–670, 1999.

[FR99] J. Falbe and M. Regitz, editors. *Römpp Lexikon Chemie*. Georg Thieme Verlag, Stuttgart, New York, electronic edition, 1999. Version 2.0.

[GE03] J. Gasteiger and T. Engel, editors. *Chemoinformatics*. Wiley-VCH, Weinheim, 2003.

[GF00] G. K.-M. Goh and J. A. Foster. Evolving molecules for drug design using genetic algorithms via molecular trees. In *In Proceedings of the Genetic and Evolutionary Computation Conference*, pages 27–33, Las Vegas, Nevada, 2000. Morgan Kaufmann.

[GKK04] I. Gerdes, F. Klawonn, and R. Kruse. *Evolutionäre Algorithmen*. Vieweg-Verlag, Wiesbaden, 2004.

[GKW+02] V. J. Gillet, W. Khatib, P. Willet, P. J. Fleming, and D. V. S. Green. Combinatorial library design using a multiobjective genetic algorithm. *J. Chem. Inf. Comput. Sci.*, 42:375–385, 2002.

[GLW99] A. Globus, J. Lawton, and T. Wipke. Automatic molecular design using evolutionary techniques. *Nanotechnology*, 10:290–299, 1999.

[GLW01] A. Globus, J. Lawton, and T. Wipke. Graph crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, page 761ff, San Francisco, CA, 2001. Morgan Kaufmann.

[GNM+94] V. J. Gillet, W. Newell, P. Mata, G. Myatt, S. Sike, Z. Zsoldos, and A. P. Johnson. SPROUT: Recent developments in the de novo design of molecules. *J. Chem. Inf. Comput. Sci.*, 34:207–217, 1994.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[Gol97] C. F. Goldfarb. SGML: The reason why and the first published hint. *Journal of the American Society for Information Science*, 48(7), 1997.

[GP95] R. C. Glen and A. W. R. Payne. A genetic algorithm for the automated generation of molecules within contraints. *Journal of Computer-Aided Molecular Design*, 9:181–202, 1995.

[GP98] A. Gobbi and D. Poppinger. Genetic optimization of combinatorial libraries. *Biotechnology & Bioengineering (Combinatorial Chemistry)*, 61(1):47–54, 1998.

[GWBG99] V. J. Gillet, P. Willet, J. Bradshaw, and D. V. S. Green. Selecting combinatorial libraries to optimize diversity and physical properties. *J. Chem. Inf. Comput. Sci.*, 39:169–177, 1999.

[Haw04] D. M. Hawkins. The problem of overfitting. *J. Chem. Inf. Comput. Sci.*, 44:1–12, 2004.

[HBM03] D. M. Hawkins, S. C. Basak, and D. Mills. Assessing model fit by cross-validation. *J. Chem. Inf. Comput. Sci.*, 43:579–586, 2003.

[Heb49] D. Hebb. *The Organization of Behaviour*. John Wiley & Sons, New York, 1949.

[HN87] R. Hecht-Nielsen. Counterpropagation networks. *Applied Optics*, 26(23):4979–4984, 1987.

[HO96] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaption. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 312–317, Nagoya, Japan, 1996. IEEE.

[Hol62] J. H. Holland. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, 3:297–314, 1962.

[Hol75] J. H. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[Hol93] R. C. Holte. Very simple classification rules perform well on most commonly used data sets. *Machine Learning*, 11:63–91, 1993.

[Huu00] J. Huuskonen. Estimation of aqueous solubility for a diverse set of organic compounds based on molecular topology. *J. Chem. Inf. Comput. Sci.*, 40:773–777, 2000.

[HXZX04] T. J. Hou, K. Xia, W. Zhang, and X. J. Xu. ADME evaluation in drug discovery. 4. prediction of aqueous solubility based on atom contribution approach. *J. Chem. Inf. Comput. Sci.*, 44:266–275, 2004.

[IEBW00] K. Illgen, T. Enderle, C. Brogner, and L. Weber. Simulated molecular evolution in a full combinatorial library. *Chemistry & Biology*, 7:433–441, 2000.

[IG94] W. D. Ihlenfeldt and J. Gasteiger. Hash codes for the identification and classification of molecular structure elements. *J. Comput. Chem.*, 15:793–813, 1994.

[KGFR96] J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors. *Proceedings of the 1st Annual Conference on Genetic Programming*, Cambridge, MA, 1996. MIT Press.

[KGV83] S. Kirkpatrick Jr., C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[KH77] L. B. Kier and L. H. Hall. The nature of structure-activity relationships and their relation to molecular connectivity. *Eur. J. Med. Chem.*, 12:307–312, 1977.

[KH86] L. B. Kier and L. H. Hall. *Molecular Connectivity in Structure-Activity Analysis*. Research Studies Press – Wiley, Chichester (UK), 1986.

[KHK+96] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola. LVQ_PAK: The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

[KHKL96] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. SOM_PAK: The self-organizing map program package. Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Koh90] T. Kohonen. Improved versions of learning vector quantization. In *Proceedings of the International Joint Conference on Neural Networks*, pages 545–550, San Diego, 1990.

[Koh95] T. Kohonen. *Self-Organizing Maps*. Springer Verlag, Berlin, 1995.

[Koz92] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[Koz94] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994.

[KP96] V. Kvasnička and J. Pospíchal. Simulated annealing construction of molecular graphs with required properties. *J. Chem. Inf. Comput. Sci.*, 36:516–526, 1996.

[LDS90] Y. LeCun, J. Denker, and S. Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Processing Systems 2*, pages 598–605. Morgan-Kaufmann, 1990.

[LLDF97] C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, 23:3–25, 1997.

[LLS99] T. Lim, W. Loh, and Y. Shih. A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms. In *Machine Learning*, Boston, 1999. Kluwer Academic Publishers.

[LP02] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, Berlin, 2002.

[LRGB99] K. Länge, F.-R. Rapp, G. Gauglitz, and A. Brecht. Flow injection immunotitration: Extended working range for inhibition type immunodetection. *Anal. Chim. Acta*, 399:275–286, 1999.

[MB80a] G. Moreau and P. Broto. The autocorrelation of a topological structure: A new molecular descriptor. *Nouv. J. Chim.*, 4:359–360, 1980.

[MB80b] G. Moreau and P. Broto. Autocorrelation of molecular structures: Application to SAR studies. *Nouv. J. Chim.*, 4:757–764, 1980.

[MBI+00] D. Maclean, J. J. Baldwin, V. T. Ivanov, Y. Kato, A. Shaw, P. Schneider, and E. M. Gordon. Glossary of terms used in combinatorical chemistry. *Journal of Combinatorial Chemistry*, 2:562–578, 2000.

[MD89] J. Moody and C. J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[MHB01] I. Muegge, S. L. Heald, and D. Brittelli. Simple selection criteria for drug-like chemical matter. *J. Med. Chem.*, 44(12):1841–1846, 2001.

[Mic96] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, third edition, 1996.

[Mit96] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, 1996.

[Mit97] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[Møl93] M. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[Mon93] D. J. Montana. Strongly typed genetic programming. Technical Report 7866, Bolt Beranek and Newman, Cambridge, MA, 1993.

[Mon95] D. J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2):199–230, 1995.

[Moo90] W. J. Moore. *Grundlagen der physikalischen Chemie*. de Gruyter Verlag, Berlin, New York, 1990.

[MP43] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[MP69] M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[MPG+03] M. Murcia-Soler, F. Pérez-Giménez, F. J. García-March, M. T. Salabert-Salvador, W. Díaz-Villanueva, and M. J. Castro-Bleda. Drugs and non-drugs: An effictive discrimination with topological methods and artificial neural networks. *J. Chem. Inf. Comput. Sci.*, 43:1688–1702, 2003.

[Nac98] R. B. Nachbar. Molecular evolution: A hierarchical representation for chemical topology and its automated manipulation. In *In Proceedings of Third Annual Genetic Programming Conference*, pages 246–253, Madison, WI, 1998.

[Nac00] R. B. Nachbar. Molecular evolution: Automated manipulation of hierarchical chemical topology and its application to avarage molecular structures. *Genetic Programming and Evolvable Machines*, 1:57–94, 2000.

[NBV91] R. Nilakantan, N. Bauman, and R. Venkataraghavan. A method for automatic generation of novel chemical structures and its potential applications to drug discovery. *J. Chem. Inf. Comput. Sci.*, 31:527–530, 1991.

[Obj95] Object Management Group. *The Common Object Request Broker: Architecture and Specification*, 2.0 edition, July 1995.

[Obj96] Object Management Group. *Description of New OMA Reference Model, Draft 1*, May 1996. OMG Document ab/96-05-02.

[ODTL01] T. I. Oprea, A. M. Davis, S. J. Teague, and P. D. Leeson. Is there a difference between leads and drugs? a historical perspective. *J. Chem. Inf. Comput. Sci.*, 41:1308–1315, 2001.

[PFB+00] V. V. Poroikov, D. A. Filimonov, Y. V. Borodina, A. A. Lagulin, and A. Kos. Robustness of biological activity spectra predicting by computer program PASS for noncongeneric sets of chemical compounds. *J. Chem. Inf. Comput. Sci.*, 40:1349–1355, 2000.

[PG89] T. Poggio and F. Girosi. A theory of networks for approximation and learning. MIT, 1989. A. I. Memo No. 1140.

[PR00] A. Puder and K. Römer. *MICO: An Open Source CORBA Implementation*. Morgan Kaufmann, San Francisco, CA, 3rd edition, 2000.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, Santa Mateo, CA, 1993.

[Ran75] M. Randic. On characterization of molecular branching. *J. Am. Chem. Soc.*, 97:6609–6615, 1975.

[Rap96] F.-R. Rapp. *Kalibrierung und Validierung von Sensorarrays mit neuronalen Netzen und genetischen Algorithmen*. Diplomarbeit, Eberhard-Karls-Universität Tübingen, 1996.

[RB93] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

[Rea76] R. C. Read. The enumeration of acyclic chemical compounds. In A. T. Balaban, editor, *In Chemical Application of Graph Theory*, pages 25–61. Academic Press, London, UK, 1976.

[Rec65] I. Rechenberg. *Cybernetic solution path of an experimental problem*. Royal Aircraft Establishment Library Translation No 1122, Farnborough, UK, 1965.

[Rec73] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann Holzboog Verlag, Stuttgart, 1973.

[Rec94] I. Rechenberg. *Evolutionsstrategie '94*. Frommann-Holzboog, Stuttgart, 1994.

[RHW86] D. Rummelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In D. Rummelhart and J. McClelland, editors, *Parallel Distributed Processing*, pages 318–362, Cambridge, MA, 1986. MIT Press.

[Ros58] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[RRA+99] K. Rocksloh, F.-R. Rapp, S. Abu Abed, W. Müller, M. Reher, and G. Gauglitz. Optimization of crushing strength and disintegration time of a high-dose plant extract tablet by neural networks. *Drug. Develop. Ind. Pharm.*, 25:1015–1025, 1999.

[RSZG96] F.-R. Rapp, J. Seemann, A. Zell, and G. Gauglitz. Optimierung neuronaler Netze zur Kalibrierung von Sensorarrays. 5. Diskussionstagung zum Rechnereinsatz in der Spektroskopie, Gemen, 1996. Poster.

[RZ00a] F.-R. Rapp and A. Zell. SOLVES - Ein verteiltes Programmsystem zur Suche und Optimierung von Leitstrukturen. Molecular Modelling Workshop, Darmstadt, 2000. Poster.

[RZ00b] F.-R. Rapp and A. Zell. SOLVES - Ein verteiltes Programmsystem zur Suche und Optimierung von Leitstrukturen. Workshop Informationssysteme in den Biowissenschaften, Magdeburg, 2000. Vortrag.

[RZ01] F.-R. Rapp and A. Zell. Daten- und Methodenmanagement bei der Suche und Optimierung von Leitstrukturen. Molecular Modelling Workshop, Darmstadt, 2001. Vortrag.

[SAJ+96] J. Singh, M. A. Ator, E. P. Jaeger, M. P. Allen, D. A. Whipple, J. E. Soloweij, S. Chowdhary, and A. M. Treasurywala. Application of genetic algorithms to combinatorial synthesis: A computational approach to lead identification and lead optimization. *J. Am. Chem. Soc.*, 118:1669–1676, 1996.

[SBSW99] B. Schölkopf, P. Bartlett, A. J. Smola, and R. Williamson. Shrinking the tube: A new support vector regression algorithm. In *Advances in Neural Information Processing Systems*, volume 11, Cambridge, MA, 1999. MIT Press.

[Sch65] H.-P. Schwefel. *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*. Diplomarbeit, Technische Universität Berlin, 1965.

[Sch81] H.-P. Schwefel. *Numerical Optimization for Computer Models*. John Wiley, Chichester, UK, 1981.

[Sch95] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[SCH+00] G. Schneider, O. Clément - Chomienne, L. Hilfiger, P. Schneider, S. Kirsch, H.-J. Böhm, and W. Neidhart. Evolutionäres De-novo-Design bioaktiver Moleküle: ein Ansatz zum virtuellen Screening. *Angewandte Chemie*, 112:4305–4309, 2000.

[SFD96] T. Soule, J. A. Foster, and J. Dickinson. Code growth in genetic programming. In *[KGFR96]*, pages 215–223, 1996.

[Sie98a] H. Siemens. *Datenanalyse mit neuronalen Netzen am Beispiel der Arzneimittelforschung*. Dissertation, Eberhard-Karls-Universität Tübingen, 1998.

[Sie98b] Siemens AG Austria. Environment for Computer Aided Neural Software Engineering (ECANSE) User Manual, Release 2.0, 1998.

[SK95] R. P. Sheridan and S. K. Kearsley. Using a genetic algorithm to suggest combinatorial libraries. *J. Chem. Inf. Comput. Sci.*, 35:310–320, 1995.

[SK98] J. Sadowski and H. Kubinyi. A scoring scheme for discriminating between drugs and nondrugs. *J. Med. Chem.*, 41:3325–3329, 1998.

[SOW04] J. J. Sutherland, L. A. O'Brien, and D. F. Weaver. A comparison of methods for modeling quantitative structure–activity relationships. *J. Med. Chem.*, 47:5541–5554, 2004.

[Spa04] P. V. Spade. William of Ockham. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Stanford University, Sept. 2004.

[SRZG97] J. Seemann, F.-R. Rapp, A. Zell, and G. Gauglitz. Classical and modern algorithms for the evaluation of data from sensor arrays. *Fresenius J. Anal. Chem.*, 359:100–106, 1997.

[TC00] R. Todeschini and V. Consonni. *Handbook of Molecular Descriptors*. Wiley-VCH, Weinheim, 2000.

[UN96] M. A. Upal and E. Neufeld. Comparison of unsupervised classifiers. In *Proceedings of the First International Conference of Information, Statistics and Induction in Science*, volume 65, pages 342–353, Melbourne, 1996. World Scientific.

[Vap95] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[Vap98] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[VCC94] V. Venkatasubramanian, K. Chan, and J. M. Caruthers. Computer-aided molecular design using genetic algorithms. *Computers & Chemical Engineering*, 18(9):833–844, 1994.

[VCC95] V. Venkatasubramanian, K. Chan, and J. M. Caruthers. Evolutionary design of molecules with desired properties using the genetic algorithm. *J. Chem. Inf. Comput. Sci.*, 35:188–195, 1995.

[VJC+02] D. F. Veber, S. R. Johnson, H.-Y. Cheng, B. R. Smith, K. W. Ward, and K. D. Kopple. Molecular properties that influence the oral bioavailability of drug candidates. *J. Med. Chem.*, 45:2615–2623, 2002.

[Web98] L. Weber. Applications of genetic algorithms in molecular diversity. *Current Opinion in Chemical Biology*, 2:381–385, 1998.

[Wei88] D. Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28:31–36, 1988.

[WF00] I. H. Witten and E. Frank. *Data Mining*. Morgan Kaufmann, San Francisco, 2000.

[Wie47] H. Wiener. Structural determination of paraffin boiling points. *J. Am. Chem. Soc.*, 69:17–20, 1947.

[WM97] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[WM02] W. P. Walters and M. A. Murcko. Prediction of 'drug-likeness'. *Advanced Drug Delivery Reviews*, 54:255–271, 2002.

[WvG00] M. Wagener and V. J. van Geerestein. Potential drugs and nondrugs: Prediction and identification of important structural features. *J. Chem. Inf. Comput. Sci.*, 40:280–292, 2000.

[WWBG95] L. Weber, S. Wallbaum, C. Broger, and K. Gubernator. Optimization of the biological activity of combinatorial compound libraries by a genetic algorithm. *Angewandte Chemie, International Edition*, 34(20):2280–2282, 1995.

[WWW89] D. Weininger, A. Weininger, and J. L. Weininger. SMILES. 2. algorithm for generation of unique SMILES notation. *J. Chem. Inf. Comput. Sci.*, 29:97–101, 1989.

[XS00] J. Xu and J. Stevenson. Drug-like index: A new approach to measure drug-like compounds and their diversity. *J. Chem. Inf. Comput. Sci.*, 40:1177–1187, 2000.

[YB92] S. H. Yalkowski and S. Banerjee. *Aqueous Solubility. Methods of Estimation for Organic Compounds*. Marcel Dekker, New York, 1992.

[YH01] A. Yasri and D. Hartsough. Toward an optimal procedure for variable selection and QSAR model building. *J. Chem. Inf. Comput. Sci.*, 41:1218–1227, 2001.

[Zel94] A. Zell. *Simulation Neuronaler Netze*. Addison-Wesley, Bonn, 1994.

[ZMV+95] A. Zell, G. Mamier, M. Vogt, N. Mache, and R. Hübner. SNNS Stuttgart Neural Network Simulator – User Manual. Technical Report 6, Universität Stuttgart, 1995.

Anhang A

Symbole und Abkürzungen

Die im Text verwendeten physikalisch-chemischen und mathematischen Symbole werden in der Tabelle A.1 erläutert. In Tabelle A.2 wird die Bedeutung von Synonymen und Abkürzungen aufgeschlüsselt.

Symbol	**Bedeutung**
ΔE	Energiedifferenz
ΔG	Änderung der freien Bindungsenthalpie
G^0	freie Standard-Bindungsenthalpie
IC_{50}	die Proteinaktivität halbierende Ligandkonzentration
k_B	Boltzmann-Konstante
K_i	Bindungskonstante
P	n-Octanol/Wasser-Verteilungskoeffizient
$\log P$	Lipophilie
q	atomare Ladung
q_{tot}	Gesamtladung
R	Gaskonstante
T_U	Umgebungstemperatur
$D_{\max}$	Maximale Baumtiefe (GP)
η	Lernrate (NN)
γ	Anzahl der Generationswechsel (ES)
μ	Populationsgröße bzw. Anzahl der Eltern (ES)
λ	Anzahl der generierten Nachkommen (ES)

Tabelle A.1: Physikalisch-chemische (oben) und mathematische (unten) Symbole

Abkürzung	**Bedeutung**	**Gebiet**
ACD	*Available Chemicals Directory*	Substanzdatenbank
ADME	*Absorption, Distribution, Metabolism, Excretion*	Wirkstoffdesign
C4.5	Ein Entscheidungsbaumverfahren	Maschinelles Lernen
CADD	*Computer Aided Drug Design*	Wirkstoffdesign
CMC	*Comprehensive Medicinal Chemistry*	Substanzdatenbank
CoMFA	*Comparative Molecular Field Analysis*	Wirkstoffdesign
CORINA	*COoRdINAtes*	Programmpaket
EA	Evolutionärer Algorithmus	Evol. Optimierung
ES	Evolutionsstrategie	Evol. Optimierung
GA	Genetischer Algorithmus	Evol. Optimierung
GP	Genetisches Programmieren	Evol. Optimierung
HTS	*High Throughput Screening*	Messverfahren
kNN	*k-Nearest-Neighbour Classification*	Maschinelles Lernen
LISP	*LISt Processing*	Programmiersprache
LVQ	*Learning Vector Quantization*	Maschinelles Lernen
MDDR	*MACCS-II Drug Data Report*	Substanzdatenbank
MLP	*Multi-Layer Perceptron*	Neuronale Netze
MLR	*Multiple Linear Regression*	Lineare Regression
NN	Neuronales Netz	Maschinelles Lernen
PCA	*Principal Component Analysis*	Projektion
PCR	*Principal Component Regression*	Lineare Regression
PETRA	*Parameter Estimation for the Treatment of Reactivity Applications*	Programmpaket
QSAR	*Quant. Structure-Activity Relationship*	Wirkstoffdesign
QSPR	*Quant. Structure-Property Relationship*	Wirkstoffdesign
RBF	Radiale Basis-Funktionen	Neuronale Netze
RMSE	*Root Mean Square Error*	Fehlermaß
SNNS	Stuttgarter Neuronale Netze Simulator	Programmpaket
SOM	*Self-Organizing Map*	Neuronale Netze
STEEDS	*SMARTS Tree Encoding for the Evolution of Descriptor Sets*	Evol. Optimierung
STGP	Streng Typisiertes Genetisches Programmieren	Evol. Optimierung
SVM	*Support Vector Machines*	Maschinelles Lernen
WDI	*World Drug Index*	Substanzdatenbank

Tabelle A.2: Synonyme und Abkürzungen (in alphabetischer Reihenfolge)

Anhang B

Glossar

Im Rahmen des Glossars wird die Bedeutung von Schlagwörtern und Fachbegriffen dargelegt, sofern dies nicht bereits beim jeweils ersten Auftreten im Text geschehen ist. Ausführlichere Definitionen finden sich im RÖMPP LEXIKON CHEMIE [FR99], kurze Definitionen aus pharmazeutischer Sicht liefert BÖHM *et al.* [BKK96, S. 74]. Ein Glossar von Begriffen aus der kombinatorischen Chemie wurde von MACLEAN *et al.* [MBI+00] zusammengestellt.

Agonist Als Agonisten werden in der Pharmakologie physiologische Bindungspartner (Liganden) eines Rezeptors, welche eine Wirkung (Rezeptorantwort) erzeugen, bezeichnet.

Akzeptor Als Akzeptoren werden Atome oder Moleküle bezeichnet, die Elementarteilchen (Elektronen, Protonen), Atome, Ionen oder Moleküle anlagern oder Energie aufnehmen können. Ein für das Wirkstoffdesign wichtiges Beispiel ist der Elektronen-Donor-Akzeptor-Komplex, in dem der Akzeptor vom Donor einsame Elektronenpaare aufnimmt.

Antagonist Ein Antagonist erzeugt eine zum Agonisten entgegengesetzte Wirkung. In der Pharmakologie werden Substanzen, die Rezeptoren für ihre physiolog. Bindungspartner (Liganden) blockieren oder anderweitig die Rezeptorantwort unterbinden, als Antagonisten bezeichnet. Kompetetive Antagonisten konkurrieren mit dem Agonisten um dieselbe Bindetasche am Rezeptor, lösen aber nach der Anlagerung keine Wirkung aus. Nicht-kompetetive Antagonisten verhindern die Agonisten-Wirkung durch Anlagerung an einer anderen Stelle des Rezeptors.

Chemoinformatik Anwendung von Methoden der Datenverarbeitung auf chemische Strukturdatenbanken und biologische Messdaten. Ein Haupteinsatzgebiet ist die Suche nach neuen Leitstrukturen im Bereich des rationalen Wirkstoffdesigns.

Chemometrie Beschäftigt sich mit der statistischen und multivariaten Auswertung von chemischen Messdaten. Eines der Haupteinsatzgebiete ist die Kalibrierung von Messapparaturen in der analytischen Chemie.

Donor Als Donor (bzw. Donator) werden Atome oder Moleküle bezeichnet, die Elementarteilchen (Elektronen, Protonen), Atome, Ionen oder Energie abgeben können, wenn ein geeigneter Akzeptor zugegen ist. In der Pharmakologie sind meist Elektronenpaar-Donoren gemeint, welche sich mit geeigneten Akzeptoren zu Elektronen-Donor-Akzeptor-Komplexen zusammenlagern können.

Entropie Im Bereich des maschinellen Lernens wird die Entropie als Maß für den Informationsgehalt einer Datenmenge definiert. Sie gibt an, mit wieviel *bits* ein Element der Menge eindeutig definiert werden kann (vgl. Kap. 2.1).

Enzym Proteine, die im Organismus bzw. nach Ausscheidung durch denselben als Biokatalysatoren durch Beeinflussung der Aktivierungsenergie die Reaktionsgeschwindigkeit chemischer Prozesse spezifisch zu erhöhen vermögen und die Umsetzungen des Stoffwechsels katalytisch steuern.

High Throughput Screening Ein automatisiertes Messverfahren, bei dem pro Durchlauf $10^5 - 10^6$ Substanzen auf ihre Affinität zu einem Rezeptor getestet werden. Diese wird meist als IC_{50}-Wert angegeben und ist mit einem hohen Messfehler (oft $\approx 20\%$) behaftet.

Hit Bezeichnet Substanzen, deren Affinität zu einem Rezeptor in einem HTS-Lauf über einem vorgegebenen Schwellwert liegt.

Hydrophobe Wechselwirkung Eine Wechselwirkung, die z. B. zwischen apolaren (hydrophoben) Molekülen oder Seitenketten von Proteinen in wässriger Lösung auftritt. Sie kann beispielsweise zur Stabilität der Protein-Raumstruktur oder von Ligand-Rezeptor-Komplexen beitragen und beruht im wesentlichen auf dem Effekt einer Entropie-Änderung.

Inhibitor Im Wirkstoffdesign meist als Kurzbezeichnung für Enzym-Inhibitor (Antienzym, Antimetabolit) verwendet, dessen Wirkung auf kompetitiver Hemmung, auf Allosterie-Effekten oder auf irreversiblen chemischen Reaktionen beruhen kann.

Kombinatorische Chemie Die kombinatorische Chemie ermöglicht eine automatisierte Generierung großer Substanzdatenbanken. Dabei wird ein molekulares Grundgerüst (*scaffold*) an mehreren Reaktionspunkten systematisch um chemische Gruppen erweitert. Da die Herstellung sämtlicher Kombinationsmöglichkeiten in vielen Fällen zu aufwändig ist, wird oft mit zuvor optimierten Teilbibliotheken oder mit rein virtuellen Bibliotheken gearbeitet.

Leitstruktur Ein Ligand für einen vorgegebenen Rezeptor, der einerseits ein neuartiges Strukturmotiv aufweist, andererseits aber meist noch eine Reihe von unerwünschten Eigenschaften besitzt, wie z. B. zu geringe Bindungsstärke, schlechte Bioverfügbarkeit oder Toxizität. Im Vergleich mit den späteren Arzneistoffen weisen Leitstrukturen ein kleineres Molekulargewicht, sowie weniger Ringe und rotierbare Bindungen auf [ODTL01]. Eine zu hohe Komplexität wäre hinderlich bei der Optimierung, was zu der Forderung nach *lead-like leads* geführt hat.

Leitstruktur-Optimierung Durchführung von wiederholten, gezielten Änderungen an einer Leitstruktur, beispielweise durch den Austausch von bioisosteren Gruppen. Ziel ist die Eliminierung von unerwünschten Eigenschaften und die gleichzeitige Erhöhung der Bindungsstärke möglichst in den sub-nanomolaren Bereich.

Ligand In der Pharmakologie versteht man unter Liganden vergleichsweise kleine Moleküle, die an spezifischen Stellen von Makromolekülen (Rezeptoren) gebunden werden (z. B. Substrate an ein Enzym).

Pharmakophor Beschreibt die zur Erzielung einer bestimmten biologischen Wirkung notwendige räumliche Anordung von funktionellen Gruppen.

Rezeptor In der Pharmakologie werden als physiolog. Sensoren dienende Makromoleküle, welche auf spezifische physiolog. Signale reagieren, als Rezeptoren bezeichnet. Die häufiger auftretenden Chemorezeptoren reagieren auf die Konzentrations-Änderung bestimmter Stoffe. Andere Arten von Rezeptoren reagieren beispielsweise auf thermische, mechanische oder Lichtreize. Im engeren Sinne handelt es sich dabei um ein lösliches oder an einer Membran gebundenes Protein, bei dem die Bindung eines Agonisten eine Wirkung auslösen kann.

Schlüssel-Schloss-Prinzip Nach dem von Emil Fischer im Jahre 1894 postulierten Schlüssel-Schloss-Prinzip beruht die Wirkung eines Arzneistoffes stets auf der Bindung eines relativ kleinen Liganden (des Schlüssels) an einem größeren Rezeptor (dem Schloss). Die hier implizierte Starrheit des Schlosses hat sich jedoch als irreführend erwiesen, da der Rezeptor sich in vielen Fällen der Form des Liganden anpassen kann (*induced fit*). Zudem sind heute auch andere Arten der Wechselwirkung bekannt (z. B. Protein-Protein), die durch dieses Prinzip nicht abgedeckt werden.

Substrat In der Pharmakologie wird eine Substanz, die von einem spezifischem Enzym als Katalysator zu einem Produkt umgewandelt wird (oft als Teil einer Reaktionskette), als Substrat bezeichnet.

Anhang C

Datensätze und Deskriptoren

In diesem Abschnitt werden ins Detail gehende Informationen zu den verwendeten Deskriptoren und den untersuchten Datensätzen gegeben. In Tabelle C.2 finden sich die Namen von allen in MOE (Version 2004.3) verfügbaren 2D-Deskriptoren.

ID	Klasse `aliphatic`	ID	Klasse `aromatic`	ID	Klasse `alcohol`
1	Methan	15	Benzol	26	Methanol
2	Ethan	16	Naphthalin	27	Ethanol
3	Propan	17	Anthracen	28	Propanol
4	Butan	18	Tetracen	29	Isopropanol
5	Pentan	19	Pentacen	30	n-Butanol
6	Hexan	20	Phenantren	31	s-Butanol
7	Heptan	21	Pyren	32	t-Butanol
8	Octan	22	Toluol	33	Phenol
9	Decan	23	o-Xylol	34	m-Kresol
10	Cyclopropan	24	m-Xylol	35	o-Kresol
11	Cyclobutan	25	p-Xylol	36	p-Kresol
12	Cyclopentan				
13	Cyclohexan				
14	Cycloheptan				

Tabelle C.1: Die Namen der Strukturen des im Abschnitt 4.1 betrachteten einfachen Datensatzes. Die Strukturen sind in die drei Klassen `aliphatic`, `aromatic` und `alcohol` eingeteilt.

1	diameter	50	PEOE_PC+	99	RPC-
2	petitjean	51	PEOE_PC-	100	Kier1
3	petitjeanSC	52	PEOE_RPC+	101	Kier2
4	radius	53	PEOE_RPC-	102	Kier3
5	VDistEq	54	PEOE_VSA+0	103	KierA1
6	VDistMa	55	PEOE_VSA+1	104	KierA2
7	weinerPath	56	PEOE_VSA+2	105	KierA3
8	weinerPol	57	PEOE_VSA+3	106	KierFlex
9	a_aro	58	PEOE_VSA+4	107	apol
10	a_count	59	PEOE_VSA+5	108	bpol
11	a_IC	60	PEOE_VSA+6	109	mr
12	a_ICM	61	PEOE_VSA-0	110	a_acc
13	a_nH	62	PEOE_VSA-1	111	a_acid
14	b_1rotN	63	PEOE_VSA-2	112	a_base
15	b_1rotR	64	PEOE_VSA-3	113	a_don
16	b_ar	65	PEOE_VSA-4	114	a_hyd
17	b_count	66	PEOE_VSA-5	115	vsa_acc
18	b_double	67	PEOE_VSA-6	116	vsa_acid
19	b_rotN	68	PEOE_VSA_FHYD	117	vsa_base
20	b_rotR	69	PEOE_VSA_FNEG	118	vsa_don
21	b_single	70	PEOE_VSA_FPNEG	119	vsa_hyd
22	b_triple	71	PEOE_VSA_FPOL	120	vsa_other
23	chi0v	72	PEOE_VSA_FPOS	121	vsa_pol
24	chi0v_C	73	PEOE_VSA_FPPOS	122	SlogP
25	chi1v	74	PEOE_VSA_HYD	123	SlogP_VSA0
26	chi1v_C	75	PEOE_VSA_NEG	124	SlogP_VSA1
27	reactive	76	PEOE_VSA_PNEG	125	SlogP_VSA2
28	Weight	77	PEOE_VSA_POL	126	SlogP_VSA3
29	a_heavy	78	PEOE_VSA_POS	127	SlogP_VSA4
30	a_nB	79	PEOE_VSA_PPOS	128	SlogP_VSA5
31	a_nBr	80	PC+	129	SlogP_VSA6
32	a_nC	81	PC-	130	SlogP_VSA7
33	a_nCl	82	Q_PC+	131	SlogP_VSA8
34	a_nF	83	Q_PC-	132	SlogP_VSA9
35	a_nI	84	Q_RPC+	133	SMR
36	a_nN	85	Q_RPC-	134	SMR_VSA0
37	a_nO	86	Q_VSA_FHYD	135	SMR_VSA1
38	a_nP	87	Q_VSA_FNEG	136	SMR_VSA2
39	a_nS	88	Q_VSA_FPNEG	137	SMR_VSA3
40	b_heavy	89	Q_VSA_FPOL	138	SMR_VSA4
41	chi0	90	Q_VSA_FPOS	139	SMR_VSA5
42	chi0_C	91	Q_VSA_FPPOS	140	SMR_VSA6
43	chi1	92	Q_VSA_HYD	141	SMR_VSA7
44	chi1_C	93	Q_VSA_NEG	142	TPSA
45	FCharge	94	Q_VSA_PNEG	143	density
46	VAdjEq	95	Q_VSA_POL	144	vdw_area
47	VAdjMa	96	Q_VSA_POS	145	vdw_vol
48	zagreb	97	Q_VSA_PPOS	146	logP(o/w)
49	balabanJ	98	RPC+		

Tabelle C.2: Auflistung der in MOE (Version 2004.3) verfügbaren 2D-Deskriptoren.

Lebens- und Bildungsgang

seit Oktober 2002	**Angestellt als Chemoinformatiker** ALTANA Pharma AG Byk-Gulden-Str. 2 78467 Konstanz
Dezember 1998 bis September 2002	**Wissenschaftlicher Mitarbeiter, Universität Tübingen** Wilhelm-Schickard-Institut für Informatik Lehrstuhl Rechnerarchitektur Prof. Dr. Andreas Zell
November 1996 bis Dezember 1998	**Wissenschaftlicher Mitarbeiter, Universität Tübingen** Institut für Physikalische Chemie Arbeitskreis Optische Spektroskopie Prof. Dr. Günter Gauglitz
April 1996 bis November 1996	**Diplomarbeit** an der Universität Tübingen zum Thema "Kalibrierung und Validierung von Sensorarrays mit neuronalen Netzen und genetischen Algorithmen"
Oktober 1990 bis März 1996	**Studium der Informatik** an der Universität Tübingen mit Nebenfach Chemie