Anwendung multivariater Methoden und künstlicher neuronaler Netze zur Klassifizierung von Spirituosen mittels Headspace-GC/MS-Kopplung

Diplomarbeit

vorgelegt von

Patrick Kursawe

Ruhr-Universität Bochum 1998

Diese Diplomarbeit wurde zwischen November 1997 und April 1998 am Lehrstuhl für Analytische Chemie der Ruhr-Universität Bochum erstellt.

Ich danke Herrn Prof. Dr. H.-J. Götze für die Betreuung dieser Diplomarbeit.

Ich danke Herrn Prof. Dr. W. S. Sheldrick für die Bereitstellung des Arbeitsplatzes.

Ich danke ganz besonders Herrn Dr. P. Zinn und Herrn Dipl.-Chem. T. Blenkers für ihre Unterstützung im gesamten Verlauf dieser Arbeit.

Inhaltsverzeichnis

1 Einleitung - Mustererkennung und Klassifizierung

Das Erkennen von Mustern ist eine für Menschen und Tiere überlebenswichtige Fähigkeit. Äußere Reize müssen mit Bedeutungen assoziiert werden, um eine angemessene Reaktion auslösen zu können. Daher ist es für uns selbstverständlich, daß wir in Sekundenschnelle die komplexen Wahrnehmungen unserer Sinne bewerten können und zu Schlußfolgerungen wie harmlos, gefährlich, eßbar, ungenießbar, zur korrekten Einschätzung von Bewegungen anderer, zum Erkennen von Gesichtern, Stimmen, Buchstaben, Zahlen, Automarken oder Geschmacksrichtungen fähig sind, um nur wenige Beispiele zu nennen. Dies geschieht meist ohne bewußtes Nachdenken. Wie wichtig diese Fähigkeiten sind, fällt normalerweise erst dann auf, wenn sie verloren gehen. Untersuchungen an Personen mit Schlaganfällen oder Schädelverletzungen führten zu ersten Einblicken in die Funktionsweise natürlicher neuronaler Netze [5][6].

Die unerreichte Leistungsfähigkeit des Gehirns im Bereich der Mustererkennung wird deutlich, wenn Muster ohne menschliche Mitwirkung erkannt werden sollen. Erst nach jahrelanger Forschung sind Computerprogramme entwickelt worden, die auf Teilgebiete der Mustererkennung wie Schrift- und Spracherkennung oder Identifizierung von Gesichtern spezialisiert sind. Zwei verschiedene Ansätze sind hierbei von Bedeutung: Zum einen können die Muster statistischen Betrachtungen unterzogen werden, wonach eine von Experten programmierte Entscheidungslogik die eigentliche Klassifizierung vornimmt. Zum anderen wird versucht, den besten bekannten Klassifizierer – das Gehirn – zu simulieren und somit lernfähige Systeme zu erhalten, die ohne explizite Eingabe von Expertenwissen in der Lage sind, Mustererkennung durchzuführen.

Die Einsatzmöglichkeiten für computergestützte Mustererkennungssysteme sind vielfältig. Sie können den Menschen ersetzen, wie in autonom fahrenden Fahrzeugen, oder entlasten, beispielsweise in der Qualitätskontrolle, wenn ein Klassifizierungssystem Produkte untersucht, bewertet und nur die Zweifelsfälle dem menschlichen Experten überläßt.

In der Chemie kann die Mustererkennung überall da eingesetzt werden, wo Entscheidungen auf der Basis komplexer Meßdaten getroffen werden müssen. Ist diese Probe im normalen Rahmen oder muß sie näher untersucht werden? Auf welche funktionalen Gruppen deutet dieser Bereich des Spektrums hin? Bei der Analyse von Bodenproben können leicht bis zu 1000 Einzelbestandteile erfaßt werden. Um eine Reihe von Proben miteinander zu vergleichen, ist der Einsatz von chemometrischen Methoden unerläßlich [8].

Moderne Analysenmethoden wie die GC/MS-Kopplung produzieren schnell schwer überschaubare Datenmengen. Automatische Mustererkennung und Klassifizierung können in einigen Teilbereichen schneller und verläßlicher als der Mensch arbeiten – sie ermüden nicht und sind objektiv –, in anderen Bereichen sind sie zumindest eine wertvolle Unterstützung bei der Sichtung der gesammelten Informationen [9].

In den letzten Jahren werden verstärkt künstliche neuronale Netze zur Klassifizierung eingesetzt, wo bisher klassische multivariate Methoden zum Einsatz kamen. Gegenstand dieser Arbeit ist die exemplarische Anwendung verschiedener Arten neuronaler Netze auf ein chemisches Problem und der Vergleich mit einer multivariaten Methode. Damit dieser Vergleich aussagekräftig sein kann, sollte er an einem Beispiel durchgeführt werden, das folgende Forderungen erfüllt:

- Es muß eine große Anzahl von Proben verfügbar sein, die über einen längeren Zeitraum stabil sind, damit wiederholte Messungen unter unterschiedlichen Bedingungen möglich sind.

- Die Proben dürfen sich nicht zu ähnlich sein, um eine Klassifizierung überhaupt durchführbar zu machen. Der Datensatz muß sich in eine Reihe von Klassen mit jeweils mehreren Mitgliedern einteilen lassen.
- Die Klassen dürfen nicht extrem unterschiedlich sein, da eine zu einfache Aufgabe von allen Klassifizierern lösbar wäre und somit keinen Vergleich der Leistungsfähigkeit erlaubte.

Diesen Bedingungen genügen die Spirituosen: Luftdicht verschlossen sind die meisten gut haltbar. Sie lassen sich nach dem Ausgangsmaterial in eine überschaubare Anzahl von Klassen einteilen (Obstwasser, Kräuterschnaps...). Jede dieser Klassen hat viele Mitglieder (verschiedene Hersteller, verschiedene Obst- oder Getreidearten). Alle Spirituosen sind das Ergebnis der Destillation von vergorenen, stärke- oder zuckerhaltigen pflanzlichen Stoffen, was zu einer begrenzten Auswahl in größerer Menge vorhandener Komponenten führt. Daher können auch zu verschiedenen Klassen gehörende Schnäpse eine große Ähnlichkeit zeigen.

Die Headspace-GC/MS-Kopplung bietet die Möglichkeit, Proben dieser Art ohne viel Aufwand zu messen und gleichzeitig eine enorme Menge an Informationen zu erhalten, da sie jede flüchtige Komponente qualitativ und quantitativ erfaßt.

Die enorme Datenmenge in der Größenordnung von über 10^5 Informationen je Probe (hier jeweils 550 bis 2400 Massenspektren) stellt ein Problem für Mustererkennungssysteme dar, ist aber für moderne chemische Analysenmethoden nicht ungewöhnlich. In dieser Arbeit soll daher gezeigt werden, wie diese Datenflut sinnvoll reduziert und zur Klassifizierung von Proben durch autonom lernende neuronale Netze benutzt werden kann.

2 Ausgangspunkt

2.1 Literaturübersicht

Es gibt zahlreiche Veröffentlichungen zur Anwendung multivariater Methoden und künstlicher neuronaler Netze in der Chemie, allerdings keine, die sich mit der Klassifizierung von Spirituosen im selben Sinne wie diese Arbeit befassen. Direkte Vergleiche sind also nicht möglich, daher sollen in diesem Abschnitt einige Arbeiten kurz vorgestellt werden, die sich mit vom technischen Standpunkt aus ähnlichen Problemen befassen.

Horimoto, Lee und Shuryo [1] untersuchten mit Headspace-Gaschromatographie den Grad des Mikrobenbefalls von Milchproben. Ziel der Arbeit war die Entwicklung eines preiswerten Systems zur Qualitätskontrolle. H-Milch-Proben wurden mit drei verschiedenen Mikroorganismen geimpft; die Software sollte dies anhand des Gaschromatogramms erkennen. Zum Einsatz kamen künstliche neuronale Netze und die multivariaten Methoden PLS und PCR (siehe Abschnitt 3), wobei die neuronalen Netze eine geringere Fehlerquote ergaben als die klassischen Regressionsverfahren.

M. Lipp [2] ermittelte mit PLS und neuronalen Netzen aus den Gaschromatogrammen von Butter, ob Verunreinigungen durch andere Fette vorlagen. Die Klassifizierung verlief nicht sehr erfolgreich, durch PLS dagegen gelang eine quantitative Bestimmung fremder Fette mit einer Nachweisgrenze von 1-2%.

Montanarella, Bassani und Breas [4] setzten verschiedene multivariate Methoden sowie Backpropagation-Netze zur Klassifizierung von Weinen bezüglich ihres Herkunftslandes anhand von Pyrolyse-Massenspektren ein. Eine feinere Unterscheidung zwischen verschiedenen Regionen mißlang mit allen verwendeten Methoden; es standen aber auch nur 33 Proben zur Verfügung, was zu Problemen führt (siehe Abschnitt 3.4).

Cao, Lin, B. Wang, K. Wang und Yu [3] wendeten Sensor-Arrays auf Basis von adsorbensbeschichteten Piezokristallen zur Klassifizierung von Spirituosen und anderen Getränkeproben an. Zur Auswertung wurden PCA und Regressionsverfahren benutzt.

P. Schreier und L. Reiner [7] versuchten bereits 1977 auf der Basis gaschromatographischer Daten eine Klassifizierung von 368 Spirituosen in die Klassen Weinbrand, Obstschnaps und Korn durchzuführen. Sie waren damals allerdings darauf angewiesen, verschiedene Komponenten quantitativ zu bestimmen und mit diesen Daten eine Diskriminanzanalyse durchzuführen. Sie stellten fest, daß die Klassen sich überlappen und daher keine perfekte Klassifizierung durchführbar ist.

2.2 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung eines Systems zur Klassifizierung von Spirituosen. Dieses System soll soweit wie möglich automatisch oder zumindest automatisierbar sein. Daher soll die Methode der Headspace-GC/MS-Kopplung eingesetzt werden, da dieses Verfahren ohne aufwendige Probenvorbereitung auskommt und die Messung der Proben vollständig automatisch ablaufen kann.

Die Klassifizierung soll durch künstliche neuronale Netze durchgeführt werden. Zum Vergleich dieser relativ neuen Methode mit den klassischen multivariaten Verfahren soll exemplarisch die Hauptkomponentenanalyse verwendet werden. Die Verwendung neuronaler Netze nach den Verfahren DLVQ (Dynamic Learning Vector Quantization), Backpropagation und RBF-DDA (Radial Basis Functions - Dynamic Decay Adjustment) ist vorgesehen. Der erste dieser Algorithmen ist vergleichsweise alt und ähnelt in gewisser Hinsicht der Clusteranalyse, der zweite ist der wohl bekannteste und am häufigsten verwendete Typ neuronaler Netze, der dritte steht stellvertretend für die neueren Entwicklungen auf diesem Gebiet, die einige Vorteile gegenüber den etablierten Verfahren bieten.

Die Fähigkeit des Systems zur korrekten Bewertung unbekannter Proben soll herausgearbeitet werden.

Hierzu sind folgende Teilaufgaben zu lösen:

1. Entwicklung einer Meßmethode, die zur Klassifizierung hinreichend charakteristische Daten aus den Proben gewinnt
2. Programmierung einer Reihe von Werkzeugen, die die Auswertung der gewonnenen Daten mit bereits vorhandener Software ermöglichen und verschiedene Vorbearbeitungen der Rohdaten erlauben
3. Begutachtung der Daten mit multivariaten Methoden (Hauptkomponentenanalyse)
4. Anwendung und Optimierung verschiedener zur Klassifizierung verwendbarer künstlicher neuronaler Netze
5. Vergleich und Bewertung der Klassifizierer mit Hilfe eines zu entwickelnden Validierungssystems

3 Verwendete Klassifizierungsmethoden

3.1 Überwachtes und unüberwachtes Lernen

Die Verfahren zum maschinellen Lernen werden in zwei große Kategorien unterteilt: Überwachtes Lernen und unüberwachtes Lernen.

Beim überwachten Lernen ist für jeden Eingabevektor des Trainingsdatensatzes der zugehörige Ausgabevektor bekannt. Das Ziel des Trainings ist, das Klassifizierungssystem in die Lage zu versetzen, selbständig die korrekten Zuordnungen zu treffen (Reklassifizierung) und den richtigen Ausgabevektor zu einem bisher

unbekannten, aber den Trainingsdaten ähnlichen Eingabevektor zu finden (Generalisierung). Für Klassifizierungssysteme heißt das, daß die freien Parameter des Klassifizierers durch das Lernverfahren so eingestellt werden müssen, daß eine möglichst gute Abbildung vom Datenraum der Messungen auf den Datenraum der Klassen stattfindet. In die Kategorie der überwachten Lernmethoden fallen die hier verwendeten künstlichen neuronalen Netze mit den Algorithmen DLVQ, RBF-DDA und Backpropagation.

Das Konzept des unüberwachten Lernens ist deutlich anders. Die Idee einer korrekten Zuordnung existiert hier nicht. Das System soll stattdessen ähnliche Datenpunkte zu Gruppen zusammmenfassen. Zu dieser Kategorie zählt die Hauptkomponentenanalyse, bei der es dem Anwender dieser Methode überlassen bleibt, die Grenzen zwischen den verschiedenen Klassen in einer vereinfachenden Projektion zu ziehen.

So unterschiedlich die Ansätze auch sind, beide Arten des Lernens können bei der Entwicklung eines Klassifizierungssystems angewendet werden. Die unüberwachten Lernmethoden sind gut dazu geeignet, sich einen Überblick über die Struktur des Datensatzes zu verschaffen. Wenn mit unüberwachtem Lernen bereits eine deutliche Gruppierung der Datenpunkte im Raum erkennbar ist und diese Gruppen auch noch den tatsächlichen Klassen entsprechen, ist der Einsatz überwachter Lernmethoden zumindest erfolgversprechend. Der Umkehrschluß ist allerdings unzulässig: Sollten Cluster erkennbar sein, die aus den Angehörigen verschiedener Klassen bestehen, so ist der Versuch einer Klassifizierung nicht unbedingt aussichtslos. Es ist dann auch möglich, daß die überwachte Lernmethode genug signifikante Unterschiede zwischen Mitgliedern desselben Clusters findet, um eine korrekte Klassifizierung durchführen zu können, die unüberwachte Methode dagegen die Gruppierung nach einem ausgeprägteren, für die gewünschte Klassifizierung aber unerheblichen Merkmal durchführt. Falls allerdings überhaupt keine Cluster erkennbar sind, obwohl verschiedene Arten der Vorbearbei-

tung und Transformation versucht wurden, ist auch überwachtes Lernen sehr wahrscheinlich erfolglos.

3.2 Multivariate Methoden

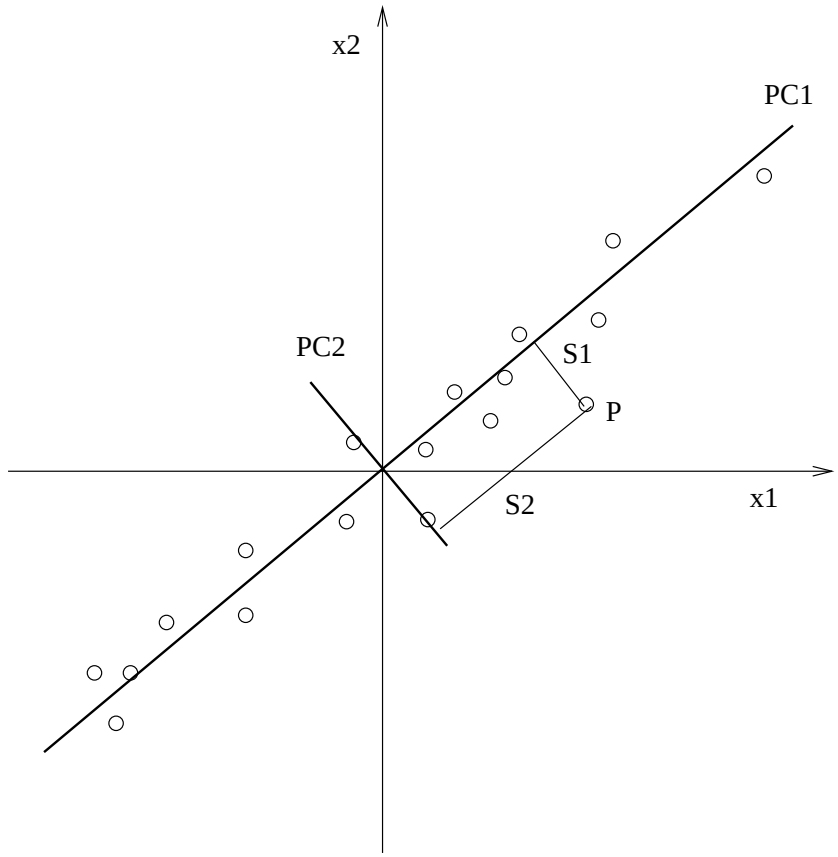
Unter multivariaten Methoden versteht man Auswerteverfahren, die mehr als nur ein Meßsignal derselben Probe verwenden, um zu einem Analysenergebnis zu kommen. In diese Kategorie fallen also Methoden wie multilineare Regression (MLR), Hauptkomponentenanalyse (PCA=Principal Components Analysis), Hauptkomponentenregression (PCR=Principal Components Regression), die PLS-Verfahren (Partial Least Squares oder Projection to Latent Structures) und Clustermethoden, um nur einige zu nennen [8].

Von dieser weit gefaßten Definition her betrachtet, sind auch die neuronalen Netze dazuzurechnen. Da sie historisch gesehen aus einem anderen Ansatz stammen und über sehr zahlreiche und stark unterschiedliche Varianten verfügen, sollen sie hier getrennt von den klassischen multivariaten Methoden betrachtet werden.

Aus diesem Bereich sind zur Klassifizierung vor allem die Hauptkomponentenanalyse (PCA) und die Clustermethoden geeignet. Da die letzteren sich funktional nicht wesentlich von den in dieser Arbeit benutzten neuronalen Netzen nach dem DLVQ-Algorithmus (Abschnitt 3.3.3) und RBF-DDA-Verfahren (Abschnitt 3.3.4) unterscheiden, sind sie im Rahmen dieser Arbeit nicht angewendet worden. Es bleibt die Hauptkomponentenanalyse, die im folgenden erläutert werden soll.

Die PCA führt eine Aufteilung der ursprünglichen Datenmatrix in zwei Matrizen durch, die Faktorenwerte (Scores) und Ladungen (Loadings) genannt werden. Im ursprünglichen Datenraum wird ein Vektor so gewählt, daß bei einer Projektion der Daten auf ihn der größtmögliche Teil der Varianz abgebildet wird. Dieser Vektor ist die erste Hauptkomponente. Die Koordinaten der Daten entlang einer

Hauptkomponente sind die Scores. Eine Multiplikation der Scores mit den zu dieser Hauptkomponente gehörenden Loadings führt wieder zu den ursprünglichen Koordinaten. Orthogonal zur ersten Hauptkomponente wird eine zweite gesucht, die möglichst viel der durch die erste Hauptkomponente noch nicht beschriebenen Varianz abbilden soll (siehe Abbildung 1).



x1,x2	Ursprüngliche Koordinatenachsen
PC1,PC2	Hauptkomponenten
S1,S2	Scores des Punktes P auf PC1 und PC2

Die Koordinaten entlang der ersten Hauptkomponente enthalten die wesentliche Information der Daten, die zweite Hauptkomponente gibt hauptsächlich die Streuung wieder.

Abbildung 1: Ermittlung der Hauptkomponenten – zweidimensionales Beispiel

Dieser Vorgang wird wiederholt bis entweder die Zahl der Hauptkomponenten der der Dimensionen der Ausgangsdaten entspricht oder bis ein vom Benutzer zu bestimmendes Abbruchkriterium erreicht ist.

Die so erhaltenen Hauptkomponenten

- sind Linearkombinationen der ursprünglichen Dimensionen
- linear unabhängig voneinander, wodurch eine gewisse Anzahl an Hauptkomponenten weniger redundante Information enthält als die gleiche Anzahl der Ausgangsvariablen
- beschreiben jeweils möglichst viel der Varianz der Ausgangsdaten, die von den vorhergehenden Hauptkomponenten noch nicht beschrieben wurde, was dazu führt, daß häufig die ersten drei bis fünf Hauptkomponenten den wesentlichen Anteil der Informationen im Datensatz wiedergeben.

Mathematisch gesehen ist die Ermittlung der Hauptkomponenten ein Eigenwertproblem. Zur Lösung gibt es mehrere Möglichkeiten, die einfachste davon ist der NIPALS (Nonlinear Iterative Partial Least Squares) - Algorithmus. Eine Beschreibung anhand eines Beispiels findet sich in [8].

Da die Hauptkomponenten nach absoluter Varianz ausgewählt werden, sollten die Daten vor der PCA autoskaliert werden. Bei NIPALS ist dies ein Teil des Algorithmus; Statistikprogramme wie der Unscrambler [11] bieten die Autoskalierung als Option, die explizit angewählt werden muß.

Hierzu wird jede Variable nach Subtraktion ihres Mittelwertes durch ihre Standardabweichung geteilt, was dazu führt, daß vor der Hauptkomponentenanalyse alle Variablen die Standardabweichung Eins und den Mittelwert Null haben.

Die Anzahl der Hauptkomponenten, die den „wesentlichen Anteil“ der Information des Datensatzes wiedergeben, ist nicht einfach zu wählen. Der wichtige Teil

soll vom Rauschanteil getrennt werden. Was jedoch wichtig ist und was Rauschen, ist schwer zu objektivieren. Eine Möglichkeit besteht darin, die Anzahl der Hauptkomponenten zu wählen, die einen vorher festgelegten Anteil der Varianz der Daten beschreibt. Ein anderer Ansatz besteht in der Betrachtung des zur Hauptkomponente gehörenden Eigenwertes; liegt er über dem Durchschnitt, ist die Hauptkomponente als wichtig anzusehen. Die Methode der Wahl für den Praktiker ist die Validierung nach dem *leave one out* - Verfahren (siehe Abschnitt 3.4), die allerdings sehr rechenintensiv ist. Die PCA wird hierbei als Mittel zur Modellierung der Originaldaten verwendet. Die Anzahl an Hauptkomponenten, die einen minimalen Validierungsfehler ergibt, wird zur weiteren Verwendung empfohlen.

Der Anwender hat als wesentliches Werkzeug zur Auswertung der PCA – neben einigen Plot-Typen, die die Ausreißersuche erleichtern – Score- und Loadingplots zur Verfügung. Score-Plots stellen die Verteilung der Meßpunkte im durch die Hauptkomponenten aufgespannten Raum dar, wogegen Loading-Plots eine Beurteilung der Wichtigkeit einzelner Variablen für das PCA-Modell erlauben. Näheres hierzu ist dem konkreten Beispiel bei der Auswertung in Abschnitt 6.3 zu entnehmen.

Das Resultat der Hauptkomponentenanalyse ist also eine Transformation des N-dimensionalen Datenraums, die zufolge hat, daß die ersten Dimensionen die wesentlichen, stark zur Gesamtvarianz beitragenden Datenanteile enthalten und die letzten Dimensionen praktisch nur noch den Rauschanteil wiedergeben. Auf diese Weise ist die Struktur der Daten durch Auftragungen der ersten Hauptkomponenten gegeneinander darstellbar. Als zwei- oder dreidimensionale Abbildung stehen sie dann zur visuellen Auswertung durch den Anwender zur Verfügung, dem es überlassen bleibt, geeignete Darstellungen auszuwählen, in denen eine Klassifizierung unbekannter Proben möglich ist.

Ein wesentlicher Nachteil dabei ist, daß Menschen nicht objektiv sind. Bei der Wahl der Grenzen zwischen den Clustern ist dem Anwender meist schon bekannt, welche Werte zur selben Klasse gehören. Die Klassifizierungsergebnisse hängen davon ab, wie gründlich der Benutzer verschiedene Auftragungen der einzelnen Hauptkomponenten gegeneinander betrachtet hat und welche Auswahl er zur Klassifizierung verwendet. Hinzu kommt, daß ein Mensch ein Gedächtnis hat und damit nicht unbeeinflußt nach der *leave one out* - Methode (siehe Abschnitt 3.4) die Leistungsfähigkeit der Klassifizierung validieren kann.

3.3 Künstliche neuronale Netze

3.3.1 Funktionsprinzip

Die Funktionsweise künstlicher neuronaler Netze ist an das biologische Vorbild Nervensystem angelehnt. Ein natürliches Neuron (siehe Abbildung 2) enthält über seine zahlreichen Dendriten Impulse von mit ihm verbundenen anderen Neuronen. Diese Signale können – je nach Art der verbindenden Synapse – anregend oder abregend auf das Neuron wirken. Überschreitet die Anregung ein gewisses Maß, gibt das Neuron über sein Axon einen Impuls an seine mit ihm vernetzten Nachbarn weiter.

Für sich betrachtet leistet ein Neuron also recht wenig. Die komplexe Verschaltung im Nervensystem und die gemeinsame, parallele Informationsverarbeitung innerhalb von Gruppen von Neuronen führen jedoch zu einem insgesamt äußerst leistungsfähigen System, das, sobald nicht nur einfaches Rechnen gefragt ist, jeden Computer problemlos in den Schatten stellt.

Wie genau solche Systeme lernen können, ist noch nicht vollständig erforscht [5]. Es gibt jedoch Versuche, diese Fähigkeit lebender Organismen für wissenschaftliche Anwendungen zu erschließen, indem die Vorgänge innerhalb der neuronalen

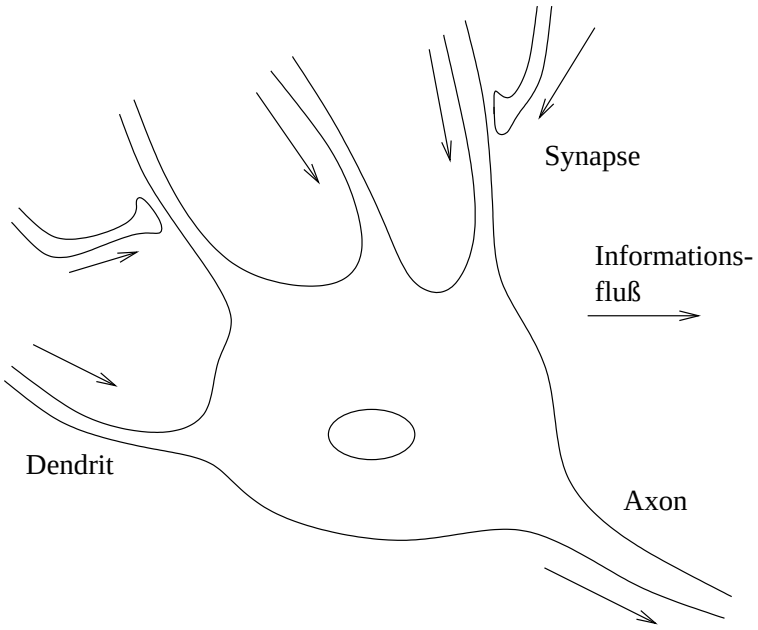


Abbildung 2: Schemazeichnung eines Neurons

Netze auf Computerprogramme übertragen werden. Hierzu ist die Verwendung stark vereinfachter Modelle unumgänglich, da die Simulation eines kompletten zentralen Nervensystems jeden Großrechner überfordern würde. Das menschliche Großhirn besteht aus ungefähr 10^{11} Neuronen, die jeweils mit 10^4 anderen verbunden sind [10]. Diese Größenordnungen sind zur Zeit nicht einmal ansatzweise erreichbar.

Das Ziel ist nicht, die Natur zu imitieren, sondern von ihr zu lernen und dadurch Systeme mit einigen Leistungsmerkmalen neuronaler Netze zu erhalten:

Autonomes Lernen ohne Vorgabe des Lösungsweges, der oft nicht oder nur unzureichend bekannt ist. Im Gegensatz zu klassischen Regressionsmodellen werden nicht nur die Parameter eines vorher festgelegten Modells angepaßt, sondern die Modellierungsfunktion selbst wird im Verlauf des Lernvorgangs entwickelt.

Schnelle Ergebnisse, wenn das Netz erst einmal trainiert ist. Wie beim natürli-

chen Vorbild ist der Lernvorgang ein langwieriger Prozeß, der Abruf der Ergebnisse dagegen sehr einfach und entsprechend rasch.

Parallele Informationsverarbeitung, die in Zukunft eine deutliche Leistungssteigerung auf Parallelrechnern erhoffen läßt und außerdem *hohe Fehlertoleranz* ermöglicht. Ausfall einzelner Neuronen verschlechtert das Ergebnis, macht es aber nicht vollständig falsch.

Dieses Ziel läßt sich mit relativ einfachen künstlichen Neuronen (siehe Abbildung 3) als Basisbausteinen erreichen. Die am häufigsten verwendete Variante besteht aus einem Vektor von *Gewichten,* die während des Lernvorgangs variiert werden. Dessen Skalarprodukt mit dem am Neuron anliegenden Eingabevektor wird durch eine – häufig sigmoide –*Aktivierungsfunktion* in den skalaren Ausgabewert transformiert. Wie beim natürlichen Vorbild ist es die Verschaltung einer Vielzahl von Neuronen, die die eigentliche Leistungsfähigkeit des Systems herstellt.

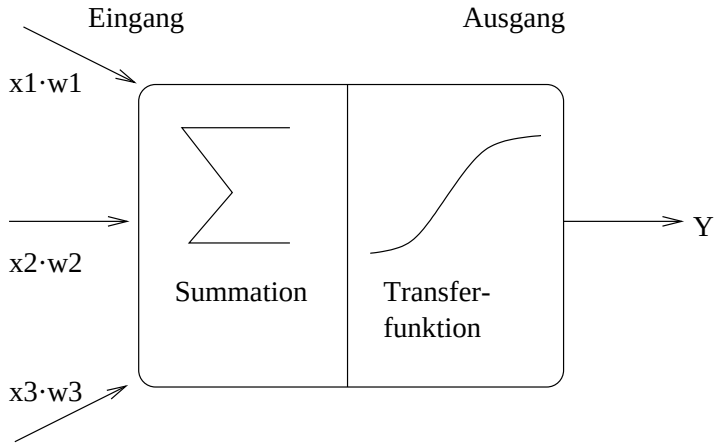


Abbildung 3: Künstliches Neuron

Die wohl bekannteste Topologie für ein neuronales Netz ist das „Multilayer Perceptron“. Hierzu werden die künstlichen Neuronen in drei Schichten angeordnet. Die erste Schicht (input layer) erhält ihre Eingaben direkt aus dem Eingabevektor. Die Ausgabewerte dieser ersten Schicht dienen der zweiten Schicht (hidden

layer) als Eingabe. Die Ausgaben der zweiten Schicht sind wiederum Eingaben für die dritte und letzte Schicht, deren Ausgabe die für den Benutzer sichtbare Antwort des Netzes auf den präsentierten Eingabevektor darstellt.

Während des Lernprozesses werden die Gewichtsvektoren, die die Verbindungen der Neuronen repräsentieren, dahingehend justiert, daß eine möglichst gute Übereinstimmung zwischen dem vorgegebenen und dem durch das Netz erzeugten Ausgabevektor erreicht wird. Hierzu gibt es eine ganze Reihe von Möglichkeiten; die folgenden drei wurden in dieser Arbeit verwendet.

3.3.2 Backpropagation

Der Backpropagation-Algorithmus ist wohl einer der am einfachsten zu implementierenden und am häufigsten verwendeten [8][1] Algorithmen für künstliche neuronale Netze. Ein typisches Backpropagation-Netz besteht aus drei (selten mehr) Schichten, wobei jedes Neuron einer Schicht mit allen Neuronen der darüber und darunterliegenden Schichten verbunden ist. Zwischen den Neuronen einer Schicht gibt es keine direkte Verbindung, die Informationen in der Abruf-Phase fließen nur in einer Richtung (*feedforward-Netzwerk*). Die Gewichte der Verbindungen werden mit Zufallszahlen initialisiert. Von diesem Startpunkt auf der Fehler-Hyperebene wird nach der Methode des steilsten Abstiegs ein Satz von Gewichten gesucht, der den Fehler des Netzes minimiert. Die Prozedur ist vergleichbar mit dem Versuch, mit verbundenen Augen den tiefsten Punkt in der Umgebung zu finden, indem man so lange auf dem steilstmöglichen Weg bergab geht, bis der Boden einer Mulde erreicht ist. Was dabei als Mulde aufgefaßt wird, hängt von der Schrittweite ab. Dabei wird offensichtlich nicht zwangsläufig das globale Optimum erreicht. Mulden mit relativ kleinem Durchmesser können verfehlt werden, obwohl sie auf dem Weg liegen. Der tatsächlich erreichte Endpunkt hängt von der Startposition

[1] $>$ 90% der Anwendungen in der Analytischen Chemie

ab. All das trifft auf den Backpropagation-Algorithmus zu.

Die Modifizierung der Gewichte beim Lernen (Δw_{ij}) erfolgt nach der Lernregel von Hebb, die in allgemeiner Form als

$$\Delta w_{ij} = g\left(a_j(t), t_j\right) h\left(o_i(t), w_{ij}\right)$$

geschrieben werden kann.

Dabei ist

w_{ij}	das Gewicht der Verbindung von Neuron i zu Neuron j,
$a_j(t)$	der Aktivierungszustand von Neuron j in Schritt t,
t_j	der gewünschte Ausgabewert für Neuron j,
$o_i(t)$	der tatsächliche Ausgabewert von Neuron i zum Zeitpunkt t,
$g(\ldots)$	eine vom Aktivierungszustand und gewünschten Ausgabewert abhängige Funktion und
$h(\ldots)$	eine vom Ausgabewert des vorgeschalteten Neurons und dem Gewicht der entsprechenden Verbindung abhängige Funktion.

Der Lernvorgang läuft in zwei Schritten ab, die als *forward propagation* und *backward propagation* bezeichnet werden. Im ersten Schritt wird dem Netz ein Eingabevektor präsentiert und Schicht für Schicht der Ausgabewert berechnet. Der so ermittelte Ausgabevektor wird mit dem gewünschten Ergebnis verglichen. Der Fehler, also die Abweichung δ_j der erhaltenen (o_j) und gewünschten (t_j) Ausgabe, wird errechnet und im zweiten Schritt zusammen mit dem Ausgabewert des Neurons der vorgeschalteten Schicht o_i verwendet, um die nötige Veränderung des Gewichts der Verbindung zwischen den Neuronen i und j, also Δw_{ij}, zu bestimmen. Dieser Vorgang verläuft nacheinander für alle Schichten des Netzes entgegengesetzt zur Richtung des normalen Informationsflusses – daher die Bezeichnung *backward propagation*.

Beim *online learning* werden die Gewichtsänderungen nach jedem einzelnen Eingabevektor auf das Netz angewendet, beim *offline* - oder *batch learning* werden

die Gewichtsänderungen für alle Muster des Trainingsdatensatzes kumuliert erst nachträglich ausgeführt. *Online learning* ist für große Datensätze deutlich schneller und wird daher auch häufiger verwendet, hat allerdings den Nachteil, daß das Lernergebnis abhängig von der Reihenfolge ist, in der die Muster dem Netz präsentiert werden.

Die Regel für die Gewichtsänderungen des klassischen Backpropagation-Algorithmus wird auch als verallgemeinerte Delta-Regel bezeichnet. Sie lautet:

$$\begin{aligned} \Delta w_{ij} &= \eta \delta_j o_i \\ \delta_j &= f_j'\left(\textstyle\sum_i w_i o_i\right) \cdot \begin{cases} (t_j - o_j) & \text{, falls } j \text{ ein Ausgabe-Neuron ist, sonst:} \\ \sum_k \delta_k w_{jk} \end{cases} \end{aligned}$$

Dabei ist

- η der Lernfaktor (eine Konstante),
- δ_i der Fehler (der Unterschied zwischen gewünschter und tatsächlicher Ausgabe) des Neurons j,
- t_j die gewünschte Ausgabe des Neurons j,
- o_i die Ausgabe des vorgeschalteten Neurons i,
- i der Index eines dem aktuellen Neuron j vorgeschalteten Neurons mit dem Gewicht w_{ij} von i nach j,
- j der Index des aktuellen Neurons,
- k der Index eines dem aktuellen Neuron j nachgeschalteten Neurons mit dem Gewicht w_{jk} von j nach k und
- $f'(x)$ die erste Ableitung der Aktivierungsfunktion des Neurons.

Die Aktivierungsfunktion bestimmt, wie der Zustand des Neurons aus den Ausgaben der vorgeschalteten Neuronen, den Gewichten der Verbindungen zu ihnen, dem vorherigen Zustand des Neurons und einem Schwellenwert (*bias*) berechnet wird. In verallgemeinerter Form kann sie als

$$a_j(t+1) = f_{\text{act}}(\text{net}_j(t), a_j(t), \Theta_j)$$

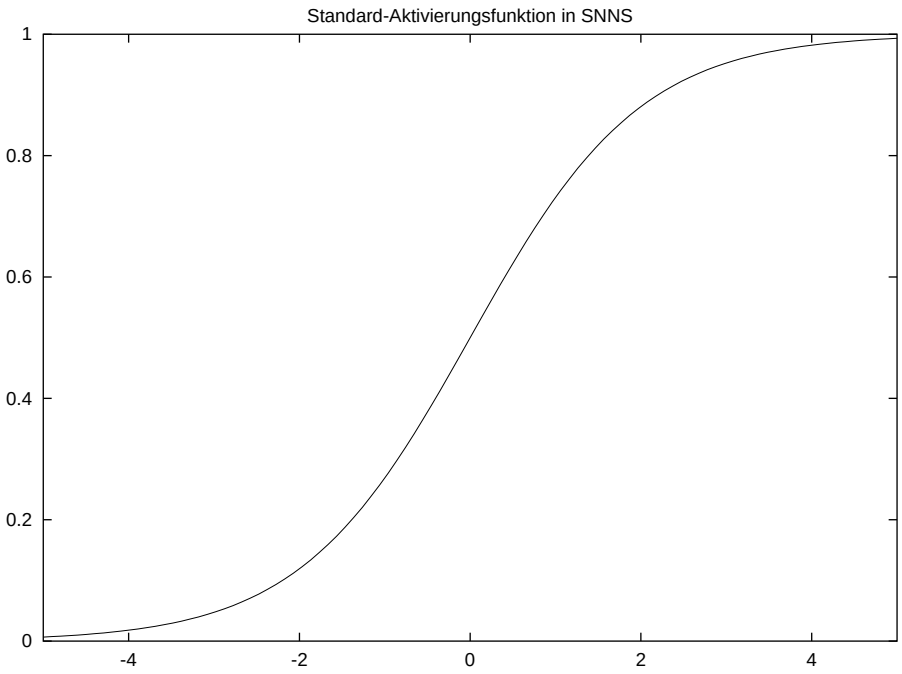


Abbildung 4: Sigmoide Aktivierungsfunktion - $1/(1+e^{-x})$

formuliert werden, wobei

$a_j(t)$	der Zustand des Neurons j im Lernzyklus t,
$\text{net}_j(t)$	die Eingabe aus der vorgeschalteten Schicht, $\sum_i w_{ij} o_i(t)$,
Θ_j	der Schwellenwert (*bias*) des Neurons j

ist und $f_{\text{act}}(x)$ bei SNNS [2] als Standard auf

$$f_{\text{act}}(x) = \frac{1}{1 + e^{-\text{net}_j(t) - \Theta_j}}$$

(siehe Abbildung 4) gesetzt ist.

Es existieren zahlreiche Abwandlungen des Backpropagation-Algorithmus, eine davon wird als *enhanced backpropagation* oder *momentum backpropagation* bezeichnet.

[2] siehe Abschnitt 6.1

Für die Berechnung der Gewichtsänderungen gilt hier:

$$\begin{aligned} \Delta w_{ij}(t+1) &= \eta \delta_j o_i + \mu \Delta w_{ij}(t) \\ \delta_j &= (f_j'(\mathrm{net}_j) + c) \cdot \begin{cases} (t_j - o_j) & \text{falls } j \text{ ein Ausgabe-Neuron ist} \\ \sum_k \delta_k w_{jk} & \text{andernfalls} \end{cases} \end{aligned}$$

Dabei ist

- η der Lernfaktor, er legt die Schrittweite des Gradientenabstiegs fest,
- μ das Moment, es legt den Anteil der letzten Gewichtsänderung fest, der zur aktuellen addiert werden soll und
- c eine Konstante, die zwecks *flat spot* Eliminierung zur Ableitung der Aktivierungsfunktion addiert wird, um ebene Flächen in der Fehlerebene überschreiten zu können.

Wenn im folgenden von Backpropagation die Rede sein wird, ist damit dieses verbesserte Verfahren gemeint. *Enhanced Backpropagation* lernt schneller und wird dank des Momentum-Terms nicht so leicht in kleinen, lokalen Minima gefangen wie der klassische Algorithmus.

Ein Backpropagation-Netz ist einfach zu implementieren, aber sehr zeitaufwendig zu trainieren. Die Ergebnisse hängen von der Wahl der Lernparameter, von der (zufälligen) Initialisierung und von der Reihenfolge der Präsentation der Eingabevektoren ab (weswegen in SNNS die Möglichkeit besteht, diese durch einen Zufallsgenerator bestimmen zu lassen).

3.3.3 DLVQ

Der Ansatz des DLVQ-Algorithmus (Dynamic Learning Vector Quantization) [13][14] ist, daß zur selben Klasse gehörende Punkte oft Cluster im Datenraum bilden. Für dieses Verfahren sollten die zu einer Klasse i gehörenden Vektoren $\vec{x}$ idealerweise um einen mittleren Vektor $\vec{\mu}_i$ normalverteilt sein. [3]

[3] DLVQ in SNNS erfordert zudem eine Normierung auf die Länge Eins.

Um einen Eingabevektor $\vec{x}$ zu klassifizieren, wird das Abstandsmaß $||\vec{\mu} - \vec{x}||^2$ zwischen $\vec{x}$ und allen mittleren Vektoren $\vec{\mu}$ ermittelt und $\vec{x}$ der Klasse mit dem nächstgelegenen mittleren Vektor zugeordnet. Die Ausgabe des Netzes besteht in der Klassennummer dieses Vektors. In der Lernphase wird geprüft, ob die Klassifizierung der vorgegebenen Klassenzugehörigkeit wirklich entspricht. Falls ein zur Klasse A gehörender Vektor als zur Klasse B gehörend eingestuft wird, werden die mittleren Vektoren $\vec{\mu}_A$ und $\vec{\mu}_B$ verschoben. Der nächstgelegene mittlere Vektor $\vec{\mu}_A$ wird etwas in Richtung des Eingabevektors $\vec{x}_A$ bewegt, der mittlere Vektor $\vec{\mu}_B$, dem $\vec{x}_A$ irrtümlich zugeordnet wurde, wird etwas von ihm fortbewegt.

Die mittleren Vektoren $\vec{\mu}$ werden durch die Gewichte zwischen den Eingabeneuronen und den Klassenneuronen der zweiten Schicht repräsentiert. Die Bewegungen dieser Vektoren werden nach folgender einfacher Regel durchgeführt:

$$w_{ij} = w_{ij} + \eta(o_i - w_{ij})$$

Hierbei ist w_{ij} das Gewicht zwischen dem Ausgabewert o_i des Neurons i der Eingabeschicht und dem Klassenneuron j. Über die Größe des Lernparameters η kann das Verhalten des Lernalgorithmus beeinflußt werden.

Dieser Trainingsschritt wird so oft durchgeführt, bis sich die Anzahl der korrekt klassifizierten Muster des Trainingsdatensatzes nicht mehr erhöht. Aus den fälschlich zur Klasse B zugeordneten Eingabevektoren der Klasse A wird ein neuer mittlerer Vektor $\vec{\mu}_A$ errechnet. Auf diese Weise werden mehrere neue mittlere Vektoren für jede Klasse erhalten. Einer dieser Vektoren wird nun zur Erzeugung je eines neuen Klasseneurons pro Klasse benutzt und der Lernvorgang erneut gestartet. Diese Prozedur wird so oft wiederholt, bis die vom Benutzer vorgegebene maximale Anzahl von Klassenneuronen pro Klasse erreicht ist.

Bemerkenswert an diesem Algorithmus ist, daß im Fall einer deutlichen, normalverteilten Clusterung der Lernprozeß bereits nach einem Zyklus beendet wäre.

3.3.4 Radial Basis Functions

Die neuronalen Netze, die unter dem Begriff *Radial Basis Functions* zusammengefaßt werden, funktionieren ähnlich wie DLVQ-Netze. Die Klassen werden durch je ein oder mehrere Neuronen der verborgenen Schicht repräsentiert. Für einen dem Netz präsentierten Eingabevektor wird unter diesen Neuronen dasjenige als Sieger ausgewählt, das die höchste Ausgabe aus diesem Eingabevektor erzeugt. Die Ausgabe des Gesamtnetzes ist dann die Nummer der Klasse, zu der dieses Siegerneuron gehört. Während bei DLVQ lediglich der kleinste euklidische Abstand ausschlaggebend für die Wahl des Siegers ist, ist das Auswahlverfahren ebenso wie der Lernvorgang bei RBF-Netzen deutlich differenzierter.

Ein mit dem DDA-Algorithmus (Dynamic Decay Adjustment)[15] trainiertes RBF-Netz ähnelt in seiner Topologie einem normalen Multilayer-Perceptron mit einer verborgenen Schicht ohne direkte Verbindungen zwischen Ein- und Ausgabeschicht. Jedes RBF-Neuron der verborgenen Schicht ist mit einem einzigen Neuron der Ausgabeschicht verbunden. Jedes der Ausgabeneuronen deckt eine Klasse ab. Alle RBF-Neuronen, die zur Klasse i gehören, sind also mit dem Ausgabeneuron i verbunden (siehe Abbildung 5).

Im Gegensatz zu einem Backpropagation-Netz ist die Aktivierungsfunktion eines RBF-Neurons lokalisiert. Es wird keine sigmoide Aktivierungsfunktion (siehe Abbildung 4) verwendet, sondern eine Gaußsche Glockenkurve. Als Eingabewert wird nicht das Skalarprodukt $\sum_i o_i w_{ij}$ verwendet, sondern der euklidische Abstand des Eingabevektors zu dem durch die Gewichte zwischen Eingabeschicht und dem RBF-Neuron repräsentierten Referenzvektor $\vec{r}_i$.

Für die Transferfunktion eines einzelnen RBF-Neurons ergibt sich also:

$$R_i(\vec{x}) = \exp\left(-\frac{||\vec{x} - \vec{r}_i||^2}{\sigma_i^2}\right),$$

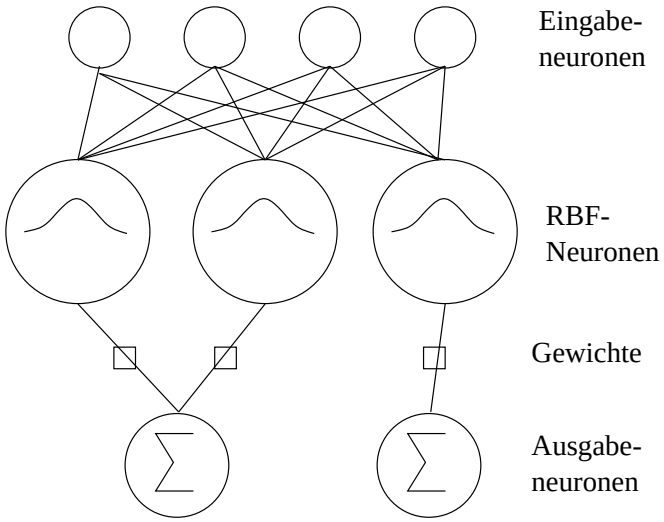


Abbildung 5: Topologie eines RBF-Netzes

wobei σ_i die Standardabweichung der Glockenkurve ist. Die Ausgabeschicht berechnet ihre Aktivierungswerte nach

$$f(\vec{x}) = \sum_{i=1}^{m} A_i \cdot R_i(\vec{x}),$$

wobei m die Anzahl an RBF-Neuronen ist, die zu der durch das Ausgabeneuron abgedeckten Klasse gehören; A_i ist das Gewicht der Verbindung zwischen dem RBF-Neuron und dem Ausgabe-Neuron.

Die verborgene Schicht eines RBF-Netzes stellt also eine Gruppe von Gaußkurven dar, deren Maxima durch die Referenzvektoren (also die Gewichte zwischen Eingabeschicht und verborgener Schicht) gegeben sind. Ein Eingabevektor löst bei jeder dieser Funktionen eine Ausgabe aus, die um so höher ist, je näher der Eingabevektor am jeweiligen Referenzvektor ist. Jedes Ausgabeneuron summiert die gewichteten Ausgaben der zur jeweiligen Klasse gehörenden RBF-Neuronen. Der Ausgabewert des Neurons mit der höchsten Aktivierung wird auf Eins gesetzt, der aller anderen Ausgabeneuronen auf Null. Das Ergebnis der Klassifizierung ist daher die Nummer des Ausgabeneurons, das als einziges den Wert Eins ausgibt.

Der DDA-Algorithmus unterscheidet zwischen sich widersprechenden und übereinstimmenden Nachbarn (zu verschiedenen / zur selben Klasse gehörend) in einer Konfliktzone, die durch die beiden Schwellwerte Θ^+ und Θ^- festgelegt wird. Diese Parameter liegen im Bereich um $f(\vec{x}) = 0,4$ und $f(\vec{x}) = 0,2$ für das jeweilige RBF-Neuron (das Maximum der Gaußkurven ist 1). Innerhalb dieser Konfliktzone darf kein Referenzvektor eines widersprechenden Nachbarn liegen. Zusätzlich soll jeder zum Training benutzte Eingabevektor im inneren Bereich ($f(\vec{x}) \geq \Theta^+$) mindestens eines RBF-Neurons der richtigen Klasse liegen.

Um dieses Ziel zu erreichen, wird, falls während des Trainings ein Muster falsch klassifiziert worden ist, entweder ein neues RBF-Neuron mit dem Gewicht Eins erzeugt oder aber das Gewicht eines existierenden Neurons, das das Muster in seinem Bereich hat ($f(\vec{x}) \geq \Theta^+$), wird erhöht. In beiden Fällen werden die Radien widersprechender Nachbarn soweit erniedrigt, daß ihre Ausgabe unter Θ^- sinkt.

Die wesentlichen Eigenschaften des RBF-DDA-Algorithmus gegenüber einem Backpropagation-Netz sind:

Konstruktives Training: Neue RBF-Neuronen werden, wenn nötig, hinzugefügt. Der Aufbau des Netzes beginnt praktisch bei Null. Es gibt daher keine überflüssigen freien Parameter.

Schnelles Training: Nach spätestens fünf Zyklen treten meist keine nennenswerten Veränderungen des Nezes mehr auf.

Garantierte Konvergenz: Es kann gezeigt werden, daß der Algorithmus definitiv einen stabilen Endpunkt erreicht, wogegen bei Backpropagation-Netzen eine Oszillation in einem Fehler-Minimum möglich ist.

Unkritische Parameterwahl: Nur die beiden Schwellwerte Θ^+ und Θ^- müssen vom Benutzer gewählt werden. Dazu kommt, daß die Auswahl recht unproblematisch ist.

Garantierte Eigenschaften des Netzes nach dem Training durch die Schwellwerte, wogegen bei Backpropagation-Netzen die tatsächliche Klassifizierungsfunktion immer auch von der zufälligen Initialisierung der Gewichte abhängt.

Darüber sollte nicht vergessen werden, daß RBF-DDA nur zur Klassifizierung geeignet ist, wogegen Backpropagation (und dessen Varianten) auch zur Modellierung benutzbar sind, was für diese Arbeit aber ohne Bedeutung ist.

3.4 Validierung

Bei der Überprüfung der Leistungsfähigkeit eines Klassifizierers sind zwei Aspekte zu berücksichtigen: Reklassifizierung und Generalisierung. Der Klassifizierer sollte in der Lage sein, die Eingaben, mit denen er trainiert wurde, richtig einzuordnen. Dies zu erreichen ist kein Problem, da durch Veränderung der Anzahl von Neuronen in den ersten beiden Schichten eine ausreichend hohe Anzahl von freien Parametern zur Verfügung gestellt werden kann, um jeden beliebigen Datensatz fehlerfrei zu reklassifizieren. Für die praktische Anwendung des Klassifizierers ist das jedoch uninteressant, es handelt sich um das Problem des *overfitting*. Ein Klassifizierer, der seinen Trainingsdatensatz abbildet, aber nicht generalisieren kann, ist vollkommen wertlos. Die interessante Größe zur Bewertung der Leistung des Systems ist die Fähigkeit, *nicht* im Training verwendete Muster korrekt zu bewerten.

Ein Klassifizierer darf also nicht mit allen zur Verfügung stehenden Daten trainiert werden, da sonst keine Validierung mehr möglich ist. Eine mögliche Lösung besteht darin, den Datensatz in zwei Teile einzuteilen, von denen einer zum Training, der andere zur Validierung benutzt wird. Je nachdem, wie diese Einteilung vorgenommen wird, werden dann allerdings unterschiedliche Ergebnisse erhalten, solange die Anzahl der Eingabevektoren nicht relativ groß im Vergleich zur

Anzahl der freien Parameter ist. Bedauerlicherweise ist die in der Praxis zur Verfügung stehende Datenmenge häufig recht gering. Daher bietet es sich an, den Datensatz in einen größeren Teil zum Training und einen kleineren Teil zur Validierung einzuteilen, beides durchzuführen und diesen Vorgang – mit einer anderen Aufteilung – so oft zu wiederholen, bis jeder Eingabevektor einmal zum Training und einmal zur Validierung benutzt worden ist. Konsequent weiterentwickelt gelangt man schließlich zum sogenannten *leave one out* - Verfahren, bei dem der Klassifizierer mit allen verfügbaren Eingabevektoren — außer einem — trainiert wird. Dieser eine Vekor wird nun klassifiziert und das Ergebnis registriert. Dies wird wiederholt, wobei ein anderer Vektor zur Validierung benutzt wird, bis jeder Vektor einmal beim Training ausgelassen und zur Validierung verwendet worden ist.

Diese Prozedur stellt das Optimum für die Bewertung des Klassifizierers dar, wenn nur ein relativ kleiner Datensatz zur Verfügung steht. Damit verbunden ist aber auch ein stark erhöhter Aufwand, da bei N Daten das neuronale Netz auch N-mal trainiert werden muß. Wie hoch dieser Aufwand tatsächlich ist, hängt stark von der Netztopologie, den Lernparametern und Algorithmen, der Größe des Datensatzes und der Leistung des Rechners ab. Bei dem in dieser Arbeit verwendeten System liegt die Zeit für eine Validierung nur im Bereich einiger Minuten.

4 Probenmaterial

Die bei dieser Arbeit verwendeten Spirituosenproben stammen größtenteils aus einem lehrstuhlinternen Spendenaufruf. Die Proben 19-57 [4] stammen aus einer privaten Sammlung von kleinen Flaschen mit jeweils um 20 ml Inhalt, [5] 28 davon

[4] Detaillierte Probenliste siehe Anhang A

[5] Ich danke Herrn Dr. Franke von der Firma Dr. Franke GmbH, Dortmund für die Bereitstellung dieser Proben

waren verwendbar. Der Spender hat sich anscheinend von einem älteren Teil seiner Sammlung getrennt, das Alter der Proben dürfte im Bereich einiger Jahre bis Jahrzehnte liegen, was bei den kleinen Flaschen ein Problem ist, da sie nicht zur längeren Lagerung ausgelegt sind. In einigen ungeöffneten Fläschchen fand sich daher auch nur die Hälfte des zu erwartenden Inhalts, andere Proben waren offensichtlich vergammelt. Zudem sind Flaschen dieser Art meist äußerst dürftig beschriftet, was die Klassifizierung erschwert. In Zweifelsfällen wurde die Einstufung wenn möglich durch eine Geruchsprobe vorgenommen. Viele der älteren Proben hatten einen stechenden bis gammeligen Geruch. Der menschliche Geruchssinn ist offensichtlich sehr empfindlich bei der Erkennung verdorbener Lebensmittel, da sich die Chromatogramme dieser Proben nicht erkennbar von denen frischer Proben unterschieden. Daher wurden sie, wenn keine anderen Gründe dagegen sprachen, trotzdem benutzt.

Für eine Klassifizierung sind natürlich nur prinzipiell klassifizierbare Proben verwendbar, weswegen ein Teil der Proben unbrauchbar war: Tequila beispielsweise läßt sich mit nichts vergleichen, außer mit anderem Tequila, der nicht verfügbar war.

Es wurden schließlich folgende Klassen verwendet:

Weinbrand und Cognac[6], aus Wein destillierter Schnaps. Er bedarf einer längeren Lagerung in Holzfässern, um trinkbar zu werden. Verschiedene Weinsorten und die Art und Dauer der Lagerung bieten eine Quelle für zahllose Variationen.

Grappa ist das Destillat aus vergorener Maische.

Sprit ist der in dieser Arbeit verwendete Oberbegriff für alle Schnäpse, die mit dem Ziel hoher Reinheit destilliert wurden und daher keinen nennenswerten

[6] Cognac bezeichnet Weinbrände aus einer bestimmten Region Frankreichs

eigenen Charakter mehr haben. In diese Klasse wurden Wodka, Korn und damit vergleichbare Schnäpse eingeordnet, die nur nach Alkohol riechen und deren Chromatogramm sich fast auf den Totzeit- und Ethanolpeak beschränkt.

Obstler sind alle Obstschnäpse. Diese Klasse hat eine sehr hohe Variationsbreite, da nicht nur die verwendeten Obstsorten unterschiedlich sind, sondern auch der Grad der Destillation. Einige Obstschnäpse haben einen Alkoholgehalt von fast 50% und sind kaum vom Sprit zu unterscheiden. Ein anderer Überlappungspunkt besteht darin, daß Weintrauben auch Obst sind, eine deutliche Trennung dieser Klasse von Grappa oder Weinbrand ist daher a priori wenig wahrscheinlich.

Kräuter ...schnäpse und Kräuterliköre haben nur eins gemeinsam: Bei ihrer Herstellung wurden irgendwelche Kräuter verwendet. Die genauen Rezepturen sind Firmengeheimnisse.

Rum wird auf Zuckerrohrbasis hergestellt. Weißer Rum wird so weit destilliert, daß davon aber nichts mehr zu merken ist und wurde deshalb unter Sprit eingeordnet.

Insgesamt standen 88 Proben zur Verfügung, von denen 68 verwendbar waren, also weder verdorben noch unklassifizierbar.

Diese 68 Proben verteilen sich wie folgt auf die sechs Klassen:

Klasse	Anzahl	Anteil
Weinbrand	6	9%
Grappa	8	12%
Sprit	18	26%
Obstler	20	29%
Kräuter	12	18%
Rum	4	6%

5 Datenerfassung

5.1 GC/MS-System

Zur Datenerfassung wurde ein GCQ[7] eingesetzt. Dabei handelt es sich um einen gewöhnlichen computerprogrammierbaren Gaschromatographen[8] mit einem Ion Trap Massenspektrometer als Detektor. Zur Injektion wurde ein CombiPAL-Autosampler benutzt, der mit einem Inkubator und einer beheizbaren gasdichten 2,5 ml Spritze ausgestattet war.

Die Steuerung des GCQ erfolgte durch einen PC mit 16 MB Arbeitsspeicher und 75 MHz Pentium-Prozessor. Die Steuerungssoftware lief unter Windows NT. Die bei der Messung erzeugten Dateien im GCQ-eigenen Datenformat wurden über das Netzwerk dem zur Auswertung benutzten Rechner zugänglich gemacht. Die Steuerung des Autosamplers über den PC ist zwar prinzipiell möglich, die GCQ-Software erlaubte jedoch in den verwendeten Versionen 2.0 und 2.2 nicht den Zugang zu headspacespezifischen Einstellungen. Daher mußten die Autosampler-Methoden direkt am Gerät erstellt werden.

[7] Firma Finnigan, Tochterfirma von ThermoQuest, deutscher Vertrieb Axel Semrau GmbH & Co (http://www.asonline.de)

[8] Finnigan GCQ GC, das System ist auch mit anderen GCs erhältlich

Die wesentlichen Gründe zur Verwendung eines Massenspektrometers als Detektor sind die erreichbare hohe Empfindlichkeit und Selektivität, besonders bei Verwendung von MS/MS[9]. Im Rahmen einer Klassifizierungsaufgabe kann das Gerät diese Vorteile kaum ausspielen, da die benötigte Information durch die Retentionszeiten der Komponenten bereits gegeben ist. Es zeigte sich allerdings bei der Auswertung, daß es unter bestimmten Umständen vorteilhaft sein kann, nur einen Teil des Massenbereiches zur Berechnung des Chromatogramms zu verwenden.

5.2 Chromatographie

Es ist mehr die Regel als die Ausnahme, daß eine Probe der Vorbereitung bedarf, bevor eine Messung möglich ist. Wäßrige Proben sind in der Gaschromatographie mit Kapillarsäulen besonders problematisch, da diese Säulen durch Wasser unter Umständen unbrauchbar gemacht werden können. Es ist dann ratsam, eine Extraktion durchzuführen. Auch wenn dies bei normalen Analysen häufig getan wird, ist es für Klassifizierungssysteme nicht akzeptabel. Diese Systeme werden für Qualitäts- oder Prozeßkontrolle sowie Screenings eingesetzt, wo jeder Arbeitsschritt, der schwer oder nicht automatisierbar ist, zu einer deutlichen Kostensteigerung führt.

Daher ist es erstrebenswert, daß die Probe möglichst ohne Vorbereitung gemessen werden kann. Hierzu gibt es verschiedene Ansätze, wie die Purge & Trap-Technik, bei der die flüchtigen Probenkomponenten durch ein Gas ausgetrieben (purge) und am Anfang der Säule ausgefroren (trap) werden. Mit weniger Aufwand realisierbar, wenn auch nicht ganz so leistungsfähig, ist die Headspace-Technik. Hier wird die Probe erwärmt, wodurch sich die leicht flüchtigen Bestandteile in der Gasphase anreichern. Aus dem Raum über der Probe wird mit einer speziellen temperierten und gasdichten Spritze eine gewisse Menge Gas entnommen, die

[9] Selektion von Ionen mit einem bestimmten m/z-Verhältnis, die dann weiter fragmentiert werden

dann in den Gaschromatographen injiziert wird. Für quantitative Analysen ist dieses Verfahren problematisch, da letztlich nur die Konzentration von Komponenten in der Gasphase über der Probe meßbar ist und Schlußfolgerungen auf die tatsächliche Zusammensetzung daher problematisch sind. Für eine Klassifizierung ist das bedeutungslos, da schließlich nicht mehr als eine Art „Fingerabdruck“ der Probe erforderlich ist.

Ohne spezielle Ausrüstung ist die Headspace-Technik nur schwer durchführbar. Der bereits angesprochene Autosampler mit Headspace-Ausrüstung stand am Anfang dieser Arbeit noch nicht zur Verfügung. Daher wurde versucht, den gewünschten Effekt zu erzielen, indem eine gasdichte Spritze und die Probe im Ofen eines anderen Gaschromatographen auf 60 °C temperiert wurden. Davon abgesehen, daß GC-Spritzen und Fläschchen bei dieser Temperatur schlecht handhabbar sind, ist die Reproduzierbarkeit schwer zu gewährleisten. Die zum Aufziehen der Spritze nötige Zeit genügt bereits, um zu sichtbarer Kondensation am Deckel des Fläschchens und der Spritze zu führen.

Der Autosampler mit Headspace-Ausrüstung löst diese Probleme. Die Probenfläschchen werden in einem genau passenden Inkubator geheizt und geschüttelt, wodurch die Wärmeübertragung auf die Probe und die Durchmischung optimal sind. Die Entnahme der Gasprobe erfolgt mit einer beheizten Spritze, wobei die Probe weiter im Ofen gelassen wird. Nach jeder Messung wird die Spritze für eine einstellbare Zeit mit einem Spülgas gereinigt, das durch ein Loch, das bei weit herausgezogenem Kolben zugänglich wird, die ganze Spritze durchströmen kann. Nach einer Minute Spülung mit Stickstoff waren keine Reste der vorherigen Probe mehr feststellbar. Möglicherweise hätte eine kürzere Zeit auch genügt, es wurde aber nicht versucht, den ohnehin geringen Gasverbrauch zu minimieren.

Der Vorzug dieser Anordnung gegenüber alternativen Headspace-Gaschromatographen, die mit speziell angepaßten Injektoren arbeiten, besteht darin, daß kei-

ne Veränderungen am eigentlichen Gerät erforderlich sind. Durch einen einfachen Spritzenwechsel ist mit derselben Apparatur auch normale Flüssiginjektion möglich.

Die Datenerfassung erfolgte in zwei Blöcken. Zu Beginn dieser Arbeit wurde zunächst eine Meßmethode entwickelt, die für eine exemplarisch ausgesuchte Probe mit vielen Komponenten (Probe 16, Zwetschgenwasser) gut aufgelöste Chromatogramme erlaubte. Während verschiedene Parameter wie die Inkubatortemperatur oder das Temperaturprofil variiert wurden, traten mehrfach Probleme mit der Reproduzierbarkeit der erzeugten Chromatogramme auf.

Anfänglich verwendete Probenfläschchen waren anscheinend aus sehr borhaltigem Glas gefertigt. Je länger eine Probe in einem derartigen Fläschchen stand, um so höher wurde ein Peak, der sich durch das Massenspektrum als Borsäuretriethylester identifizieren ließ.

Bei häufigeren Messungen derselben Probe erschienen allmählich drei Peaks, die bei weiteren Versuchen auch mit Wasser oder Methanol im Probengefäß reproduzierbar waren. Bei Headspace-Probennahme aus leeren Fläschchen waren sie schon bei der ersten Messung zu beobachten. Eine kleine Versuchsreihe und die Interpretation der Massenspektren führte schließlich zu der Schlußfolgerung, daß es sich hierbei um Substanzen aus dem Septum der Flaschen handeln mußte (siliciumhaltig). Diese Komponenten sind anscheinend wasserlöslich und daher bei Messungen wäßriger Proben erst dann feststellbar, wenn die Flüssigkeit ausreichend gesättigt ist. Bei nur zwei- oder dreimaliger Messung sind diese Peaks allerdings sehr klein und wurden daher für die zur Auswertung bestimmten Messungen als bedeutungslos bewertet.

Es folgen die für den ersten Block der Datenerfassung verwendeten GC-Bedingungen:

Säule: eine 30 m lange DB-XLB, Innendurchmesser 0,25 mm, Filmdicke 0,25 μm. Die Beschichtung besteht aus zu 12% phenyliertem, 88% methyliertem Polysiloxan.

Temperaturprogramm: Starttemperatur 35 °C, nach 5 min wird mit 10 °C/min auf 150 °C, dann mit 20 °C/min auf 200 °C erwärmt, diese Temperatur wird 1 min gehalten. Zwischen zwei Injektionen vergehen 29 Minuten.

Injektion: 500 μl in den Standardinjektor des GCQ-GC bei offenem Splitventil und einer konstanten Temperatur von 200 °C. Das Trägergas (He) ist vom GC auf eine konstante Flußgeschwindigkeit von 40 cm/s geregelt.

Autosampler: Der Inkubator auf 70 °C, die Spritze auf 90 °C temperiert. Die Verweilzeit im Inkubator beträgt 5 min. Die Inkubatortemperatur sollte möglichst hoch gewählt werden, damit sich die interessierenden Komponenten möglichst gut in der Gasphase anreichern können; sie ist jedoch nach oben durch den steigenden Dampfdruck des Alkohols begrenzt.

Die Bedingungen sind so gewählt, daß im Anfangsbereich eine möglichst gute Auflösung erreicht wird, denn hier treten besonders viele Peaks auf. Die Temperatursteigerung bis 150 °C ist recht klein gewählt, damit die Basislinie eine möglichst geringe Drift zeigt. Die letzte Temperaturrampe von 150 °C auf 200 °C dient nur dazu, störende hochsiedende Eluenten, wie sie beispielsweise durch ein neues Injektor-Septum entstehen, von der Säule zu entfernen. In diesem Endbereich, der auch nicht in die Auswertung eingeflossen ist, ist ein sehr breiter Peak zu beobachten (siehe Abbildung 7, Seite 40), der nach einem Septenwechsel recht groß ist und von Messung zu Messung immer kleiner wird.

Zu diesem ersten Block ist noch anzumerken, daß er sich über etwas mehr als einen Monat erstreckte, da einige der Proben erst später als erwartet verfügbar waren.

Zwischendurch mußten einige Einstellungen am Massenspektrometer geändert werden, deren Auswirkungen sich allerdings als nicht besonders störend für die Auswertung erwiesen.

Nachdem die Auswertung dieses ersten Blocks abgeschlossen war, wurde versucht, zu ähnlichen Ergebnissen bei kürzerer Meßzeit zu kommen. In der Zwischenzeit fanden auch einige Änderungen am GCQ statt. In diesem zweiten Block wurden daher folgende Einstellungen benutzt:

Säule: eine 25 m lange HT 8, Innendurchmesser 0,25 mm, Filmdicke 0,25 μm. Die Beschichtung besteht aus 8% phenyliertem, 92% methyliertem Polysiloxan.

Temperaturprogramm: Die ersten 2 min bei 45 °C, dann wird mit 25 °C/min auf 200 °C erwärmt. Zwischen zwei Injektionen liegen 16 Minuten.

Injektion: 500 μl in einen Optic 2 Injektor[10], bei konstant 200 °C bei offenem Splitventil. Der Trägergasdruck (He) wird während der Messung von 1 auf 1,5 bar erhöht.

Autosampler: Wie beim ersten Block. Der Inkubator ist auf 70 °C, die Spritze auf 90 °C erwärmt. Die Probe wird 5 min erhitzt.

Die mit diesen Einstellungen erhaltenen Chromatogramme sind deutlich schlechter als die des ersten Blocks (siehe Abbildung 8, Seite 41). Nur noch die größeren Peaks sind überhaupt sichtbar, die kleineren gehen im Grundrauschen fast unter, da sie viel breiter geworden sind. Es ist starkes Tailing erkennbar, was auf die veränderte Polarität der anderen Säule zurückzuführen sein dürfte. Durch das Tailing des Alkoholpeaks und das schnellere Temperaturprogramm ist die Basislinie weder gerade noch besonders gut zu erkennen, falls viele überlappende

[10] Der Optic 2 ist ein für *Large Volume Injection* entwickelter Injektor, der für eine andere Arbeit benötigt wurde.

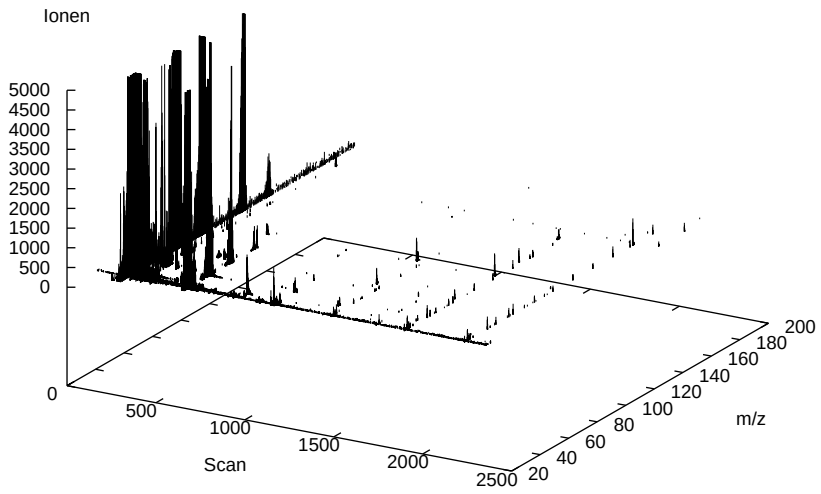
Peaks auftreten. Dies führt zu Problemen bei der Vorbearbeitung mit den Programmen `killbaseline` und `baseline` (siehe Abschnitt 6.2). Trotzdem wurde mit diesem Datensatz ein Teil der Auswertung wiederholt, um zu erfahren, wie gut die Klassifizierung mit einem unoptimierten (aber fast doppelt so schnellen) GC-System funktioniert. Eine weitere Modifikation dieser Methode erschien wenig aussichtsreich.

5.3 Detektion

Die Parameterwahl beim Massenspektrometer beeinflußt sehr stark die im Endeffekt zur Auswertung verwendeten Chromatogramme. Das Massenspektrometer des GCQ ist ein Ion-Trap-Gerät mit einer externen Ionenquelle, wie sie bei Quadrupol-Geräten verwendet wird. Die Säule des Gaschromatographen endet in der Ionenquelle. Die dort erzeugten Ionen werden in die Ionenfalle geleitet und dort zunächst gesammelt. Nach Abschluß dieser Sammelphase, die einige Millisekunden dauert, werden die Ionen nach und nach — die kleinsten Massen zuerst — aus der Ionenfalle in den Detektionsteil[11] geleitet. Dieser Scanvorgang benötigt um so mehr Zeit, je größer der Massenbereich gewählt ist.

Zur Verbesserung des Signal-Rausch-Verhältnisses für die Detektion von Ionen, die keine oder wenige schwere Fragmente bilden, bietet es sich daher an, den Scanbereich auf kleine Massen zu begrenzen. Dadurch können mehr Scans pro Datenpunkt aufsummiert werden, als wenn Zeit darauf verschwendet wird, einen Bereich abzusuchen, in dem keine Ionen vorhanden sind. Eine andere Möglichkeit besteht darin, weniger Punkte pro Zeiteinheit zu schreiben, damit mehr Scans pro Punkt vorliegen. Derselbe Effekt wird allerdings auch mit einer nachträglichen Mittelung erreicht.

[11] Der Detektor besteht aus einer Konversionsdynode und einem CDEM (Continuous Dynode Electron Multiplier). Auf technische Details soll hier verzichtet werden, da sie nicht in unmittelbarem Zusammenhang mit dem Klassifizierungsproblem stehen.



Dreidimensionales Chromatogramm der Probe 16, Zwetschgenwasser.
Nur Datenpunkte mit einer Intensität von mindestens 40 sind dargestellt, da ansonsten durch das Grundrauschen nichts zu erkennen wäre. Es ist leicht zu sehen, daß die Intensitäten bei höheren Massen deutlich kleiner werden.

Abbildung 6: 3D–Chromatogramm

Aus dreidimensionalen (Zeit, m/z, Intensität) Darstellungen der Daten (siehe Abbildung 6) läßt sich leicht ablesen, daß ein Scan über den technisch möglichen Bereich bis zur Masse 650 bei diesen Proben wenig sinnvoll wäre. Alle zur Auswertung herangezogenen Messungen beschränken sich daher auf den Massenbereich von 33 (höchste Störung durch Luft: $O_2^+ = 32\mathrm{m/z}$) bis 200. Pro Sekunde wurden zwei Datenpunkte in die Dateien geschrieben, jeder dieser Datenpunkte ist aus fünf Scans aufsummiert worden.

Eine nachträgliche Verkleinerung des Massenbereiches ist im Rahmen der Vorbearbeitung (siehe Abschnitt 6.2) problemlos möglich.

6 Auswertung

6.1 Vorstellung der verwendeten Software und Hardware

Die Auswertung wurde auf einem IBM-kompatiblen Rechner mit zwei Pentium 133 MHz Prozessoren und 32 MB Arbeitsspeicher durchgeführt. Da die Meßdaten vom GCQ in einem nur für das zugehörige Auswerteprogramm lesbaren Format gespeichert werden, mußten sie zunächst mit einem selbstgeschriebenen Programm in ein universell verwendbares Format (ASCII-Text) konvertiert werden, das den anderen Programmen leichter zugänglich ist.

Zur Hauptkomponentenanalyse wurde das kommerzielle Programm Unscrambler [11] verwendet, das unter Microsoft Windows 95 oder NT läuft. Die neuronalen Netze wurden mit dem Programmpaket SNNS [12] unter Linux [12] erstellt. Zur Konvertierung zwischen den verschiedenen Dateiformaten und zur Vorbearbeitung der Daten wurde eine Reihe von Hilfsprogrammen in C++ geschrieben, hierzu kam der Compiler egcs [13] zum Einsatz. Die Programme benutzen keine betriebssystemspezifischen Funktionen und sollten ohne größere Änderungen auf andere Systeme portierbar sein. Zur Automatisierung sich häufig wiederholender Arbeitsschritte wurden einige Hilfsprogramme wie `make, wc` und `awk` verwendet, die nicht auf allen Plattformen verfügbar sind.

6.2 Vorbearbeitung

Eine geeignete Vorbearbeitung der Daten ist der Schlüssel zum Erfolg oder Mißerfolg der eigentlichen Klassifizierung. Zu diesem Zweck wurde eine Reihe von Programmen geschrieben, die es möglich machen, recht komfortabel die Rohda-

[12] Linux ist ein frei verfügbares UNIX-artiges Betriebssystem. Für nähere Informationen siehe *http://www.linux.org*.

[13] egcs ist ein Paket aus mehreren Compilern, unter anderem für C und C++, das unter *http://www.cygnus.com/egcs* erhältlich ist.

ten zu modifizieren, um sie dem Klassifizierungssystem zuführen zu können.

Ausgangspunkt für die folgenden Schritte ist eine einfache ASCII-Tabelle, die in der ersten Spalte die Scan-Nummer[14] und in der zweiten Spalte die Intensität des chromatographischen Datenpunktes enthält. Die Programme lesen diese Tabelle über die Standardeingabe ein und geben das modifizierte Chromatogramm über die Standardausgabe aus. Dadurch können mehrere Arbeitsschritte durch *pipelining* in einer Kommandozeile durchgeführt werden, ohne daß jeweils eine Datei gelesen oder geschrieben werden muß. In der folgenden Aufstellung werden die Funktionen der einzelnen Programme kurz erläutert, um später durch Angabe des Programmnamens auf die Art der Vorbearbeitung Bezug nehmen zu können.

ms2chrom wandelt das GCQ-eigene Datenformat in einfache ASCII-Chromatogramme um[15]. Dazu werden die Ionenströme eines anzugebenden Massenbereiches aufsummiert und zusammen mit der Scan-Nummer an die Standardausgabe geschrieben. Da durch mehrfachen Aufruf mit anderen Parametern auch Chromatogramme mit mehreren verschiedenen Massenbereichen erzeugt werden können, ist dieses Programm in diesem Abschnitt „Vorbearbeitung“ mit aufgeführt.

baseline sorgt für eine einfache Basislinienkorrektur und eine erste Normierung des Chromatogramms. Die absoluten Intensitäten bei Messungen mit dem GCQ unterliegen über einen längeren Zeitraum, wie er in diesem Fall zur Erstellung von über 80 Chromatogrammen nötig war, starken Schwankungen. Das prinzipielle Erscheinungsbild ändert sich dadurch nicht unbedingt, die absoluten Werte sind allerdings - im Abstand mehrerer Tage - nicht reproduzierbar. Dieses Programm sucht – mit einer einstellbaren Fensterbreite – im Chromatogramm nach einem Bereich mit möglichst geringer Varianz.

[14] Die Scan-Nummer ist in die Retentionszeit konvertierbar.

[15] Ich danke der Firma Axel Semrau für die Bereitstellung der Dokumentation dieses Formates sowie Herrn Dipl.-Chem. T. Blenkers für seine Hilfe bei der Erstellung der Einleseroutine.

Der Mittelwert dieses Bereiches wird von jedem Punkt im Chromatogramm abgezogen. Zusätzlich wird jeder Punkt durch die Standardabweichung in diesem Intervall dividiert. Ein annähernd konstantes Rauschniveau des Massenspektrometers vorausgesetzt, werden hierdurch auch Chromatogramme mit sehr unterschiedlichen absoluten Intensitäten in den selben Bereich skaliert, wodurch ein Vergleich ermöglicht wird.

killbaseline arbeitet im wesentlichen wie `baseline`, setzt jedoch alle Punkte, die unterhalb des Wertes der N-fachen Standardabweichung liegen, auf Null. Bei geeignetem N bleiben die Peaks erhalten, das Grundrauschen wird ausgeblendet. Die resultierenden Veränderungen der Peakfläche sind recht gering und für eine Klassifizierung irrelevant.

chrom2chrom ist der erste Bearbeitungsschritt der basislinienkorrigierten Rohchromatogramme. Dieses Programm filtert alle Datenpunkte heraus, die nicht innerhalb eines anzugebenden Scan-Nummer-Bereiches liegen. Hiermit werden der Totzeitpeak und das Ende des Chromatogramms, in dem keine aus den Proben stammenden Stoffe mehr eluiert werden, vom probenspezifischen mittleren Teil des Chromatogramms getrennt. Optional kann das Chromatogramm so skaliert werden, daß der größte Peak eine vorgegebene Höhe erreicht. Eine weitere wichtige Funktion ist die Mittelwertbildung, wodurch die Chromatogramme um einen anzugebenden Faktor reduziert werden können (siehe Abbildungen 7 und 8), da mit sinkender Dimensionalität der Eingabevektoren die Rechenzeit abnimmt .

digitize erwartet ein Chromatogramm, das mit `killbaseline` bearbeitet worden ist. Werte ungleich Null werden auf Eins gesetzt.

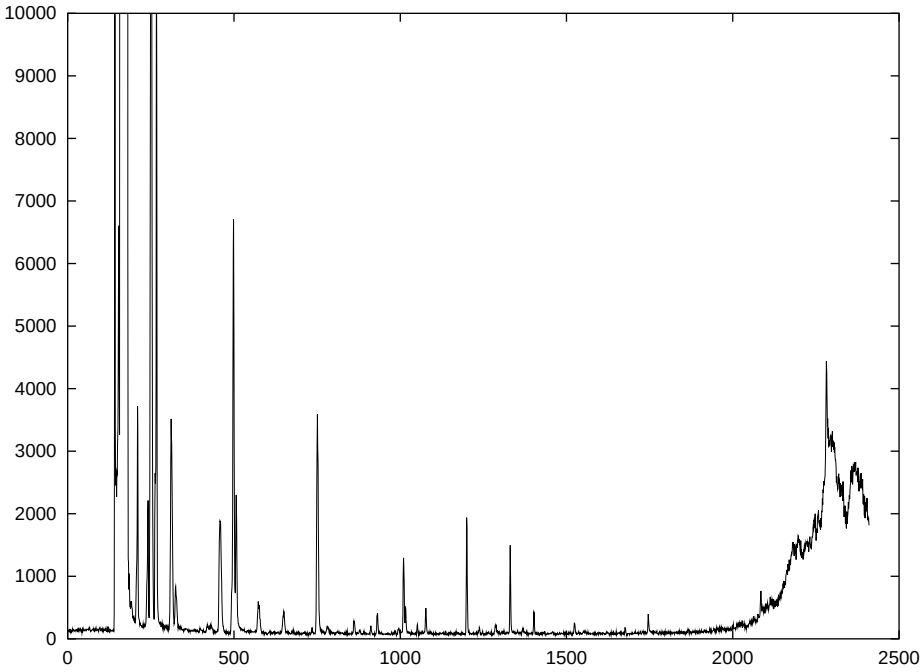
log logarithmiert die Eingabewerte. Werte, die nach Logarithmierung kleiner als Null oder nicht definiert sind, werden auf Null gesetzt.

normalize skaliert den Eingabevektor auf die Länge Eins wie beispielsweise von DLVQ benötigt.

Wie in Abbildung 7 zu erkennen ist, wurde in der ersten Meßreihe der Scanbereich 200 bis 1800 zur Auswertung verwendet. Bei der zweiten Meßreihe (siehe Abbildung 8) wurde der Bereich zwischen Scan 65 und 550 benutzt.

Zu diesen Programmen kommen noch einige weitere, die zur erleichterten Erstellung der Eingabedateien für den Unscrambler und SNNS dienen oder die Validierung mit SNNS erleichtern. Da sie aber nicht unmittelbaren Einfluß auf die Auswertung haben, sind sie hier nicht im einzelnen erläutert.

Originalchromatogramm (Probe 79, Grappa), 2410 Datenpunkte



Mit `killbaseline` und `chrom2chrom` bearbeitetes Chromatogramm, 107 Datenpunkte

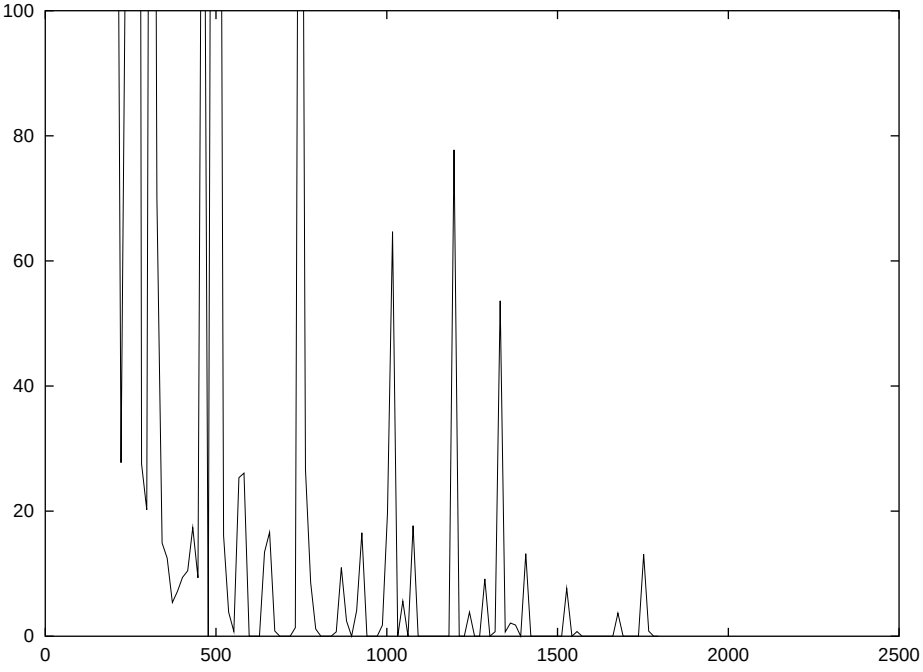
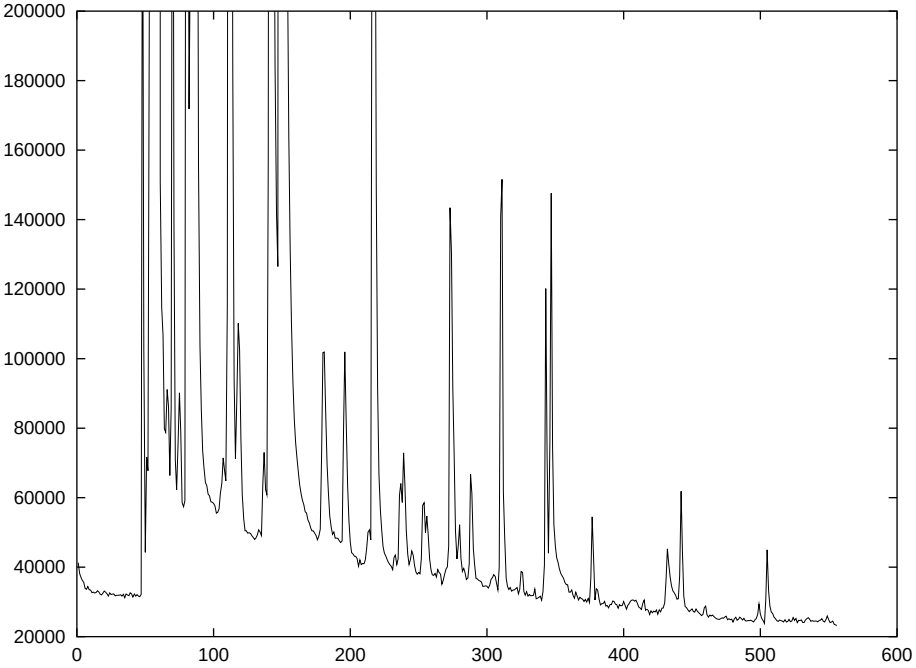


Abbildung 7: Beispiel für die Vorbearbeitung, erste Meßreihe

Originalchromatogramm (Probe 79, Grappa), 556 Datenpunkte



Mit `killbaseline` und `chrom2chrom` bearbeitet, 98 Datenpunkte

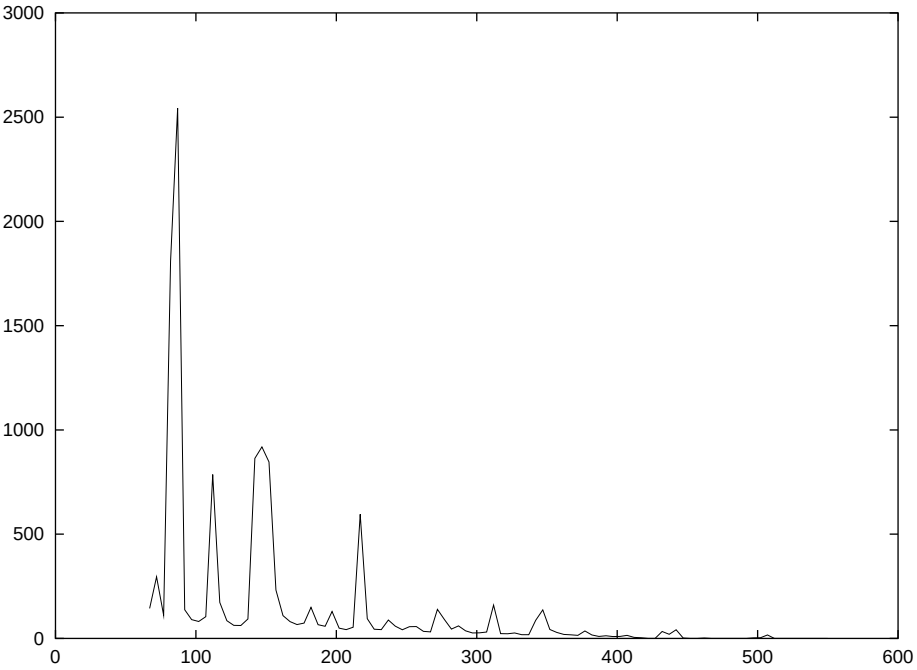


Abbildung 8: Beispiel für die Vorbearbeitung, zweite Meßreihe

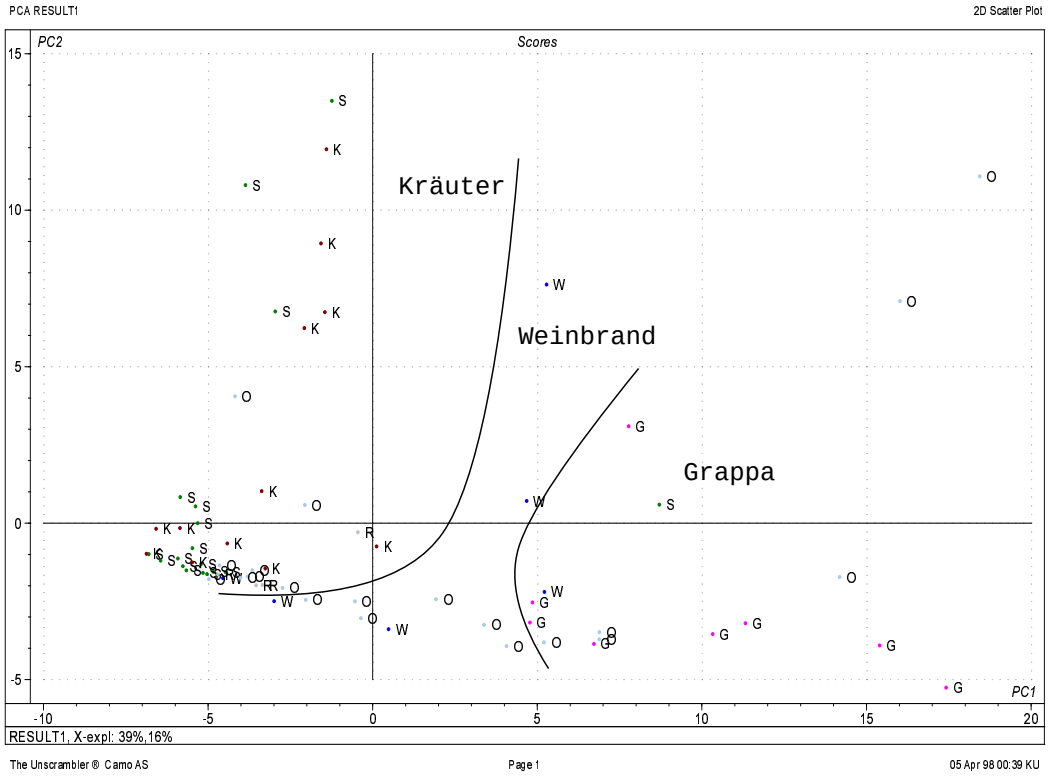
6.3 Multivariate Methoden

Wie bereits in Abschnitt 3.2 erwähnt, gehört die Hauptkomponentenanalyse zu den unüberwachten Lernmethoden, das heißt, zur Durchführung einer PCA ist es nicht notwendig, die Klassenzugehörigkeit der Proben zu kennen. In der Projektionsdarstellung der Score-Plots sind ähnliche Proben nah beieinander plaziert; das heißt allerdings nicht, daß diese Proben auch zur selben Klasse gehören müssen.

Ein konkretes Beispiel: Fast nur die zur Klasse „Kräuter“ gehörenden Proben enthalten Limonen (ein Terpen, $C_{10}H_{16}$). Der übrige Bereich des Chromatogramms zeigt allerdings keine auffälligen Gemeinsamkeiten innerhalb dieser Klasse. Daher werden sie nicht unbedingt im selben Bereich des Score-Plot erscheinen, wogegen ein solches besonderes Merkmal bei überwachten Lernmethoden durchaus ausgenutzt werden kann.

In Abschnitt 4 wurde bereits angesprochen, daß keine vollständige Unterscheidbarkeit der Klassen zu erwarten ist. Das macht die Klassifizierung durch Hauptkomponentenanalyse zu einem Problem. Im Idealfall, wenn die Ähnlichkeit innerhalb einer Klasse größer wäre als die Ähnlichkeit zwischen Mitgliedern verschiedener Klassen, ergäbe sich in den Score-Plots der ersten Hauptkomponenten eine klare Gruppierung der Proben. Bereits ein flüchtiger Blick auf den Score-Plot, der theoretisch die deutlichsten Gruppierungen liefern sollte (Abbildung 9), zeigt, daß nur eine Darstellung bei weitem nicht ausreicht, um alle sechs Klassen unterscheiden zu können.

Immerhin erlaubt diese Projektion einen ersten Schritt zur Klassifizierung nach dem Ausschlußverfahren: Eine Probe, deren Chromatogramm durch PCA in den rechten Teil dieser Abbildung projiziert würde, wäre höchstwahrscheinlich ein Grappa oder ein Obstschnaps, im linken oberen Bereich dagegen vermutlich ein Kräuterschnaps oder „Sprit“. Die in der Abbildung markierten Zonen sind ein notwendiges, aber nicht hinreichendes Kriterium für eine Klassenzugehörigkeit.



Erste Meßreihe, mit `baseline`, `chrom2chrom` und `log` bearbeiteter Datensatz, 107 Variablen.

Abbildung 9: PCA – Score-Plot, erste und zweite Hauptkomponente

Ein Punkt im Bereich *Kräuter* kann, aber muß nicht unbedingt ein Kräuterschnaps sein. Andererseits ist es recht unwahrscheinlich, daß ein Punkt außerhalb dieses Bereiches zu einem Kräuterschnaps gehört.

Auf diese Art zu einer eindeutigen Klassifizierung zu kommen, ist, falls es überhaupt gelingt, unter Umständen eine sehr aufwendige und zeitraubende Prozedur. In vielen Fällen genügt es, die Plots erste gegen zweite, erste gegen dritte, zweite gegen dritte, erste gegen vierte... bis zur sechsten oder siebten Hauptkomponente zu betrachten und eine Zuordnung nach dem Prinzip der nächsten Nachbarn zu treffen. Wenn der Punkt in den meisten betrachteten Darstellungen von Mitglie-

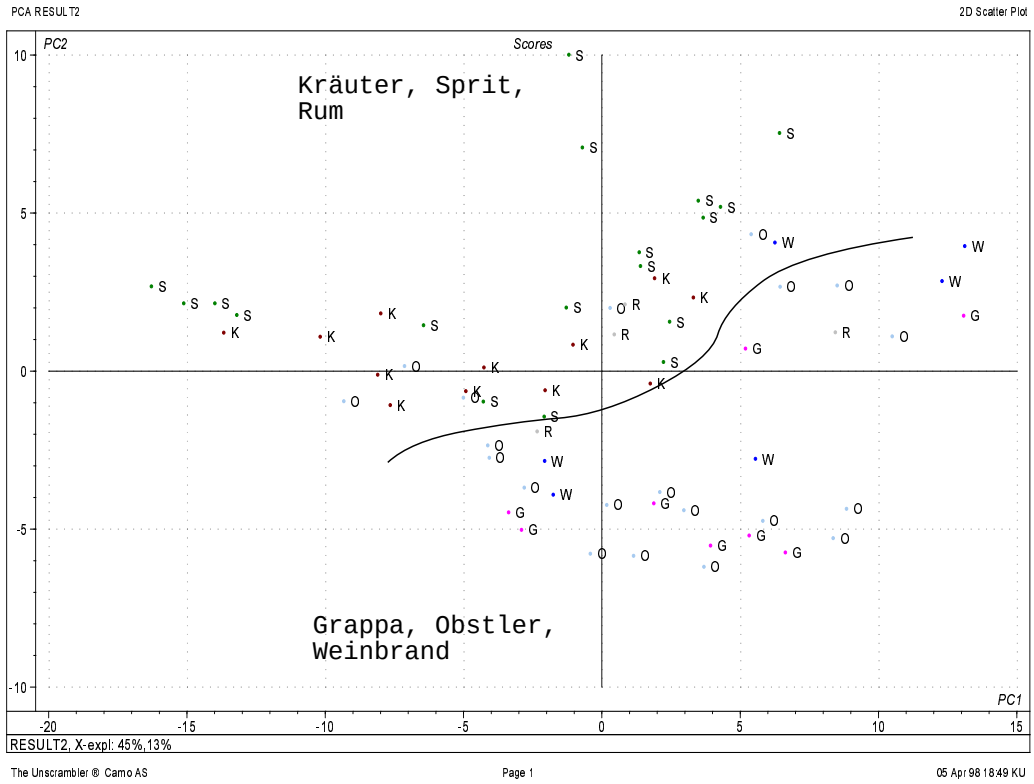
dern einer bestimmten Klasse umgeben ist, wird er höchstwahrscheinlich zu eben dieser Klasse gehören.

Bei anderen Proben reicht diese Methode nicht aus, die Punkte bilden in der Umgebung des zu Klassifizierenden keine deutliche Gruppierung. Hier kann versucht werden, weit von diesem Punkt entfernte Proben aus dem Datensatz zu eliminieren und erneut eine Hauptkomponentenanalyse mit den verbleibenden Daten durchzuführen. Möglicherweise ergibt sich im interessierenden Bereich eine deutlichere Trennung der Cluster.

In Abbildung 10 (Daten aus der zweiten Meßreihe) ist zu sehen, wie stark die Gestalt des Plots vom konkreten Datensatz abhängt. Noch stärker als die unterschiedlichen chromatographischen Bedingungen wirkt sich allerdings die Art der Vorbearbeitung auf die Interpretierbarkeit der Score-Plots aus. Logarithmierung erwies sich hier als die Methode der Wahl. An den Daten der ersten Meßreihe wurde intensiv versucht, geeignete Darstellungen zur Klassifizierung auszuwählen. Dabei gelang es, mit Hilfe von 15 verschiedenen Plots (verschiedene Hauptkomponenten, verschiedene Vorbearbeitungen), für fast alle Klassen geeignete Projektionen zu finden. Die kleine Klasse Rum (vier Mitglieder) wurde dabei nicht berücksichtigt – die Punkte lagen immer inmitten eines Clusters anderer Proben.

Es blieben also noch zehn mögliche Unterscheidungen zwischen den restlichen fünf Klassen. Vier davon konnten mit nicht logarithmierten Daten durchgeführt werden, fünf nur mit logarithmierten. Zwischen Obstschnaps und Weinbrand konnten keine zur Klassifizierung hinreichenden Unterschiede festgestellt werden.

Es wäre nicht zweckmäßig, hier detailliert aufzulisten, mit welchen Plots welche Klassifizierungen durchgeführt werden konnten. Bereits kleine Veränderungen am Datenmaterial oder der Art der Vorbearbeitung können dazu führen, daß die Hauptkomponenten anders gewählt werden und der Plot nicht wiederzuerkennen ist.



Zweite Meßreihe, mit `baseline`, `chrom2chrom` und `log` bearbeitet, 98 Variablen.

Abbildung 10: PCA – Score-Plot, erste und zweite Hauptkomponente

Eine präzise Auskunft über den Anteil korrekt klassifizierbarer Proben läßt sich prinzipiell nicht machen, da die Auswertung der PCA nicht automatisch erfolgt. Die Hauptkomponentenanalyse ist in diesem Kontext als Datenerkundungsmethode einsetzbar, die erlaubt, einen Überblick über die Zusammensetzung des Datensatzes und die Ähnlichkeit der Proben zu gewinnen und darauf basierend die Erfolgsaussichten der automatisierbaren Klassifizierungsmethoden abschätzen zu können.

Als Resümee der PCA für die in dieser Arbeit verwendeten Daten läßt sich aussagen, daß eine Klassifizierung (mit Einschränkungen bezüglich der zweiten Meßreihe) als prinzipiell erfolgversprechend angesehen werden kann.

6.4 Neuronale Netze

6.4.1 Backpropagation

Das wesentliche Problem für den Anwender von Backpropagation-Netzen ist, daß sehr viele Möglichkeiten zur Variation der Netzparameter zur Verfügung stehen. Es gibt drei stufenlos einstellbare Lernparameter; zusätzlich ist die Anzahl der Trainingszyklen und die Anzahl der Neuronen in der verborgenen Schicht prinzipiell frei wählbar. Vom Netz selbst einmal abgesehen, ermöglichen schon die verschiedenen Arten der Vorbearbeitung, die teilweise auch über optimierbare Parameter verfügen, eine große Auswahl an Ausgangspunkten für die eigentliche Auswertung. Zusammengenommen erhält man praktisch beliebig viele Klassifizierer, unter denen ein Optimum gefunden werden soll.

Wie schon in Abschnitt 3.3.1 erwähnt wurde, ist es problemlos möglich, eine Reklassifizierungsquote von über 95% zu erreichen. Daher muß, sollen aussagekräftige Ergebnisse erhalten werden, für jeden Parametersatz eine komplette Validierung durchgeführt werden. Aufgrund der geringen Größe des Datensatzes scheiden die von SNNS bereitgestellten Optionen aus, die erlauben, automatisch ein Netz bis zur Minimierung des Validierungsfehlers zu trainieren, da SNNS nur mit Trainings- und Testdatensatz arbeiten kann und die *leave one out* - Methode nicht unterstützt.

Daher war es nötig, das Hilfsprogramm `validate` zur Validierung zu entwickeln, was aus einem Datensatz von N Proben zwei SNNS-Musterdateien erzeugt, von denen eine $N-1$ Muster und die andere ein einziges enthält. Mit Hilfe des zum SNNS-Paket gehörenden Programms `batchman` kann nun ein vorher in SNNS erzeugtes neuronales Netz automatisch geladen, initialisiert und trainiert werden. Das Ergebnis der Klassifizierung des einen, nicht zum Training verwendeten Musters wird von `validate` gespeichert. Dieser Vorgang wird, wie bereits beschrie-

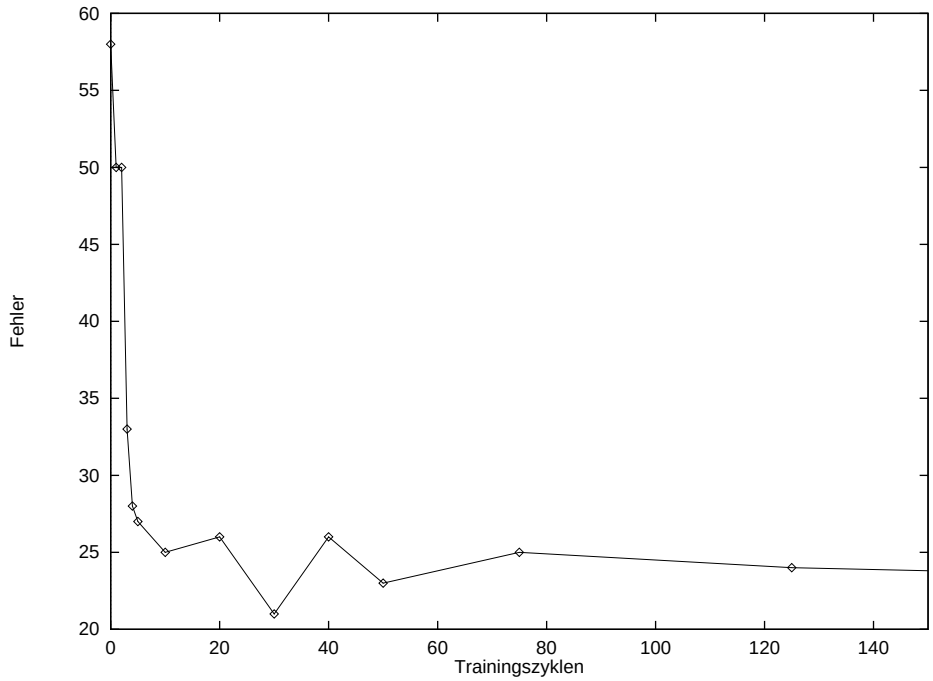
ben, N mal durchgeführt. Als Ergebnis wird der Anteil der bei der Validierung korrekt klassifizierten Proben erhalten. In diese Zahl geht *nicht* der Anteil der korrekt reklassifizierten Proben ein.

Je nach Größe der verborgenen Schicht, Größe des Eingabemusters und Anzahl der Trainingszyklen schwankt die zur Validierung nötige Rechenzeit zwischen fünf Minuten und über einer Stunde. Die Erzeugung der zum Training benutzten Chromatogramme im Fall einer Änderung der Vorbearbeitung nimmt auch einige Minuten in Anspruch. Zur Änderung der Netztopologie muß ein entsprechendes Netz in SNNS erstellt und gespeichert werden. Änderungen der Lernparameter sind in die Eingabedatei für `batchman` einzutragen. Insgesamt stellt die Erprobung jeder einzelnen der zahllosen Variationsmöglichkeiten einen nicht unerheblichen Aufwand dar. Eine wirklich gründliche und systematische manuelle Suche nach dem optimalen Klassifizierer ist daher aus Zeitgründen nicht möglich (siehe hierzu auch Abschnitt 8.3). Diese Aussagen treffen prinzipiell auch auf RBF-DDA und DLVQ-Netze zu.

Die Auswertung wurde zunächst nur mit den Daten der ersten Meßreihe durchgeführt.

Als erstes wurde nach der minimal nötigen Anzahl von Trainingszyklen gesucht, bei der von einem abgeschlossenen Training ausgegangen werden kann. Prinzipiell besteht die Möglichkeit, durch frühzeitigen Abbruch des Trainings ein Overfitting zu verhindern. Dieses Verfahren hat den Nachteil, daß es von der zufälligen Initialisierung des Netzes abhängt, wie gut es nach einer bestimmten Anzahl von Zyklen trainiert ist. Um diesen Effekt zu verhindern, wurde das Training erst beim Erreichen eines stabilen Zustandes abgebrochen und die Optimierung durch Änderung der Neuronenanzahl vorgenommen.

Abbildung 11 stellt die Abhängigkeit zwischen der Anzahl der Trainingszyklen und der Anzahl der Fehlklassifizierungen dar. Im folgenden wurden daher meist

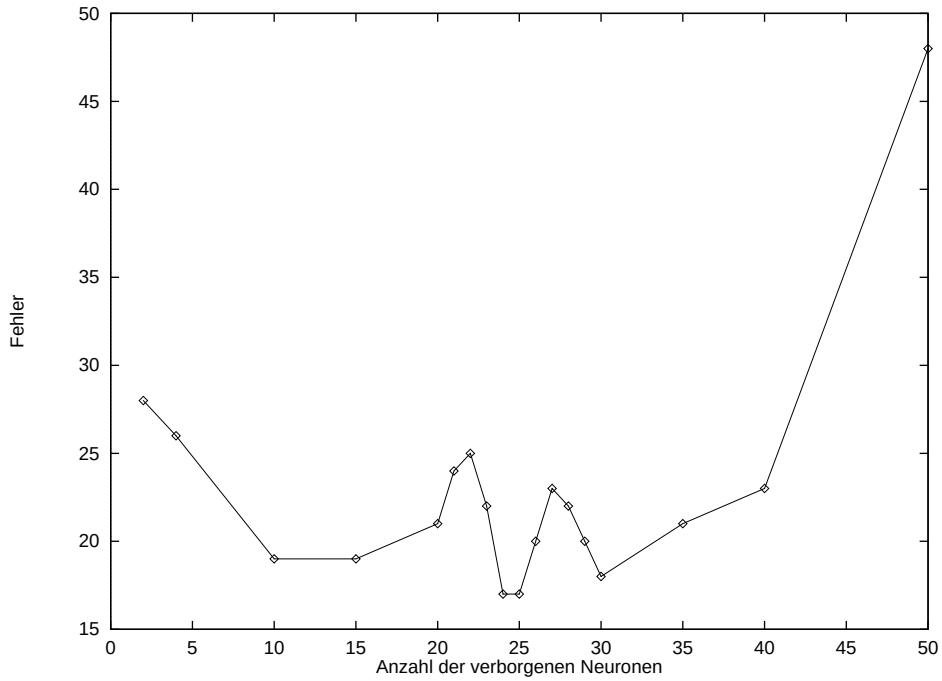


107 Eingabeneuronen, 5 versteckte Neuronen, mit `killbaseline` und `digitize` bearbeiteter Datensatz, 1. Meßreihe

Abbildung 11: Einfluß der Anzahl der Trainingszyklen auf den Klassifizierungsfehler

50 bis 100 Trainingszyklen benutzt. Auch wenn damit nicht absolut sicher ist, daß das Training abgeschlossen ist, ist zumindest der Bereich großer Veränderungen eindeutig verlassen.

Die Anzahl der verborgenen Neuronen ist ein wichtiger Faktor bei der Optimierung eines Backpropagation-Netzes. Jedes Neuron dieser Schicht hat eine Verbindung zu jedem einzelnen Neuron der Ein- und Ausgabeschicht, man kann also leicht durch Hinzufügen eines verborgenen Neurons in ein Netz mit 100 Eingabeneuronen über 100 (+ Anzahl der Verbindungen zur Ausgabeschicht) weitere anzupassende Variablen in das System einführen. Ein weiteres Eingabeneuron würde nur so viele Parameter einbringen, wie Verbindungen zur verborgenen Schicht nötig sind.



107 Eingabeneuronen, 50 Trainingszyklen, mit `killbaseline` und `digitize` bearbeiteter Datensatz, 1. Meßreihe

Abbildung 12: Einfluß der Anzahl der verborgenen Neuronen auf den Klassifizierungsfehler

Der Einfluß der Anzahl der verborgenen Neuronen ist in Abbildung 12 zu sehen. Es ist deutlich zu erkennen, daß eine zu geringe Zahl freier Parameter nicht genügt, den Datensatz angemessen zu beschreiben, andererseits eine zu hohe Anzahl zu starkem Overfitting führt. Die Gründe des V-förmigen Verlaufs des Graphen im Bereich um 25 verborgene Neuronen sind aus den vorliegenden Daten nicht ohne weiteres zu klären, eine genauere Untersuchung hätte aber höchstwahrscheinlich nicht zur Verbesserung des Klassifizierers beigetragen und wurde daher nicht durchgeführt.

Die Variation des Lernfaktors und des Momentum-Terms, die anfänglich auf den Standardwerten 0,3 und 0,9 gelassen worden waren, brachte die zu erwartenden Ergebnisse: In einem relativ großen Bereich ist die Wahl fast gleichgültig und be-

einflußt lediglich die Lerngeschwindigkeit. Werden diese Parameter allerdings zu groß gewählt, verfehlt die Methode des steilsten Abstiegs das bis dahin erreichte Minimum. Die Fehlerquote steigt dann sehr schnell an, ein Momentum-Term 0,95 führte bereits zu 50 Fehlklassifizierungen.

Der *Flat Spot* - Faktor hatte bei diesem konkreten Problem praktisch keinen Einfluß auf die Leistung des Klassifizierers.

Die Veränderung der Anzahl der Eingabeneuronen in einem Bereich zwischen 65 und 320 brachte nur geringe Unterschiede. Darüber allerdings sinkt die Leistung erkennbar: 800 Eingabeneuronen führen zu 30 Fehlklassifizierungen. Die Variation anderer Parameter kann dem in begrenztem Umfang entgegenwirken. So ergeben 107 Eingabeneuronen mit 25 verborgenen Neuronen 20 Fehler, 321 Eingabeneuronen mit 8 verborgenen Neuronen 21 Fehler. Über die Anzahl der Eingabeneuronen und die Anzahl der verborgenen Neuronen kann also einem Overfitting oder Underfitting entgegengewirkt werden. Aus den bereits aufgeführten Gründen wurde an diesem Punkt allerdings keine Optimierung versucht.

Bei einer Erhöhung der Anzahl der Eingabeneuronen steigt die Anzahl der Gewichte im Netz, es kann sich sehr gut an den Trainingsdatensatz anpassen, es kommt also zu Overfitting. Eine Reduzierung der Anzahl der verborgenen Neuronen hat genau den gegenteiligen Effekt. Dieses Wechselspiel funktioniert natürlich nur in einem begrenzten Bereich.

Die bisher angegebenen Ergebnisse wurden auf der Basis eines mit `killbaseline` und `digitize` bearbeiteten Datensatzes erzielt. Die Verwendung logarithmierter Chromatogramme, wie sie in der PCA zu den besten Ergebnissen führte, war bei Backpropagation-Netzen zunächst erfolglos. Es zeigte sich allerdings, daß die Verringerung des Momentum-Terms auf 0,3 auch mit logarithmierten Daten zu vergleichbar guten Ergebnissen (48 korrekte Klassifizierungen) führte. Ohne Digitalisierung oder Logarithmierung wurden allerdings maximal 36 Proben richtig

klassifiziert, die mögliche Erklärung hierfür folgt in Abschnitt 7.1.

Die Auswertung der zweiten Meßreihe führte wie erwartet zu schlechteren Resultaten. Erste Versuche ergaben nur knapp 30 korrekte Klassifizierungen. Es gelang im Verlauf der Optimierung, diesen Wert auf 39 zu steigern. Dabei erwies es sich als vorteilhaft, nur den Massenbereich ab 50 zu berücksichtigen[16].

Die Leistung erreichte also ungefähr den Wert eines nicht-optimierten Netzes mit den Daten der ersten Meßreihe, was beim Vergleich der Chromatogramme beider Meßreihen eher positiv überrascht.

6.4.2 DLVQ

Ein DLVQ-Netz bietet weniger Möglichkeiten zur Optimierung als Backpropagation. Die vom Benutzer vorzugebenden Parameter sind die Anzahl der maximal pro Klasse zu erzeugenden mittleren Vektoren, die Schrittweite für die Bewegung dieser Vektoren beim Training und die Anzahl an Schritten, nach denen das Training abgeschlossen werden soll. Hinzu kommen selbstverständlich wieder die zahlreichen Kombinationsmöglichkeiten unterschiedlicher Vorbearbeitungsverfahren. Da der Algorithmus nicht besonders flexibel ist (seine Stärke liegt eher in der Geschwindigkeit) und die Resultate bei der ersten Meßreihe generell schlechter als bei Backpropagation oder RBF waren, wurde dieses Verfahren weniger benutzt, um einen gut generalisierenden DLVQ-Klassifizierer zu entwickeln, sondern um mit möglichst kurzer Rechenzeit in der Anfangsphase der Auswertung verschiedene Vorbearbeitungsvarianten ausprobieren zu können. Weiterhin stellte sich schnell heraus, daß ein DLVQ-Netz relativ zu Backpropagation recht sensibel auf Fehler in den Eingabedaten reagiert, was rechtzeitig auf einen Logikfehler in `chrom2chrom` aufmerksam machte. Ein Backpropagation-Netz kann unsinnige Daten liefernde Eingabeneuronen durch Verringern der Gewichte praktisch

[16] Auf die Klassifizierungsleistung mit Daten der ersten Meßreihe hat dies fast keinen Einfluß.

deaktivieren. Diese Möglichkeit fehlt bei DLVQ.

Der kritische Parameter bei der DLVQ-Optimierung ist die Lernschrittweite. Wird dieser Parameter sehr klein gewählt, so kann dies problemlos durch eine ausreichend große Anzahl an Schritten kompensiert werden. Da der Algorithmus recht schnell ist, kann diese Zahl einfach so groß gewählt werden (einige 100, wo 20 oder 30 hinreichend wären), daß aus dieser Richtung keine Probleme zu erwarten sind. Die im SNNS-Handbuch empfohlene Schrittweite liegt bei 0,03. Hiermit wurden bei mit `digitize` bearbeiteten Daten 44 Proben korrekt klassifiziert. Bei logarithmierten Chromatogrammen war die Trefferquote nur halb so groß. Durch Änderung der Schrittweite auf 0,005 wurden mit beiden Datensätzen vergleichbare Ergebnisse erhalten.

Die Anzahl der maximal zu erzeugenden mittleren Vektoren erwies sich bei diesem Datensatz als völlig unproblematisch, da meist schon mit einem einzigen mittleren Vektor pro Klasse der Trainingsdatensatz hinreichend gut beschrieben wurde und der Algorithmus daher selbsttätig den Lernprozeß abbrach.

Die Ergebnisse von Backpropagation und RBF mit den Daten der zweiten Meßreihe machten wenig Hoffnung auf gute Resultate mit DLVQ-Netzen. Erstaunlicherweise litt deren Leistung kaum unter der schlechteren Qualität der Chromatogramme; bis zu 41 Proben wurden aus logarithmierten Daten korrekt klassifiziert.

6.4.3 Radial Basis Functions

RBF-DDA-Netze kombinieren eine durchaus mit Backpropagation vergleichbare Leistungsfähigkeit mit einem sehr schnellen Lernalgorithmus. Der größte Vorteil aus der Sicht des Anwenders ist aber die unproblematische Festlegung der Parameter. Wie in Abschnitt 3.3.4 erläutert, muß der Benutzer lediglich zwei Parameter (Θ^+ und Θ^-) auswählen und kann dabei, verglichen mit DLVQ oder Backpropagation, sehr wenig falsch machen.

Trotzdem bleiben noch etliche Variationsmöglichkeiten, die sich aus den vielen möglichen Kombinationen von Lernparametern, Anzahl der Eingabeneuronen und Arten der Vorbearbeitung ergeben.

Ein erster Versuch mit einem mit `killbaseline, chrom2chrom` und `digitize` bearbeitetem Datensatz aus der ersten Meßreihe führte mit den Standard-Parametern 0,4 und 0,2 bereits zu 39 korrekten Klassifizierungen bei 107 Eingabeneuronen. Es stellte sich heraus, daß sich sich diese Zahl durch Senken von Θ^- auf 0,1 auf immerhin 45 steigern ließ. Eine weitere Verbesserung ergab sich durch Variation der Vorbearbeitung; mit `log` und `normalize` anstelle von `digitize` ließen sich 50 von 68 Proben im leave one out - Verfahren korrekt klassifizieren.

Bei diesem Ergebnis gab es zwei Auffälligkeiten: Zum einen wurde es bereits nach einem Trainingszyklus erreicht. Der größte Anteil am Zeitbedarf der Validierung entstand nicht aus dem eigentlichen Training, sondern aus der Erzeugung der Musterdateien und dem Laden von SNNS. Zum anderen ist auffällig, daß keine einzige Probe fälschlicherweise zu einer der kleineren Klassen (Weinbrand, Grappa, Rum) zugeordnet wurde. Zur Verdeutlichung ist in der folgenden Tabelle (von `validate` erzeugt) eine Übersicht über die Fehlklassifizierungen gegeben.

```
0 0 0 0 0 0 Weinbrand
0 0 0 0 0 0 Grappa
1 0 0 2 3 1 Sprit
2 2 0 0 0 1 Obstler
1 0 2 3 0 0 Kräuter
0 0 0 0 0 0 Rum
```

Diese Matrix ist so zu verstehen, daß die Spaltenposition die tatsächliche Klasse einer Probe angibt und die Zeilenposition die vom Klassifizierer vermutete. Obige Matrix bedeutet also, daß ein Weinbrand für Sprit, zwei für Obstler und einer

für einen Kräuterschnaps gehalten worden ist. Zwei Grappa wurden als Obstler klassifiziert, drei Kräuterschnäpse für Sprit und so weiter.

Kein einziges Mal wurde eine Probe den Klassen Rum, Grappa oder Weinbrand zugeordnet, obwohl sie zu einer anderen Klasse gehörte. Bei Backpropagation und DLVQ kam es pro Klasse zwei bis sieben mal zu derartigen Fehlern.

Gerade die kleineren Klassen sind bei sich überlappenden Gebieten naturgemäß prozentual gesehen viel stärker von Fehlklassifizierungen betroffen als große Klassen. So sind in diesem Beispiel nur 50% der Weinbrände und Rums korrekt erkannt worden, dagegen 88% der Klasse Sprit. Vor diesem Hintergrund ist es interessant zu wissen, daß RBF anscheinend sehr wenige falsch-positive Aussagen bei diesen kleinen Klassen trifft.

Bei dem Parametersatz (Lernparameter 0,4 und 0,1, `killbaseline, chrom2chrom, log, normalize`) wurde der Einfluß der Anzahl der Eingabeneuronen auf die Fehlerquote untersucht. Es fand sich der in Abbildung 13 dargestellte Zusammenhang.

Dieses Bild erfordert eine etwas genauere Erläuterung. Prinzipiell sollte in diesem Graphen eine Kurve mit mindestens einem lokalen Minimum zu erwarten sein, für sehr große oder sehr kleine Zahlen von Eingabeneuronen sollte der Fehler steigen. Dies liegt daran, daß bei einer zu kleinen Anzahl von Eingabeneuronen wichtige Informationen verloren gehen, bei einer großen Zahl andererseits der Rauschanteil zu hoch ist. Die Mittelwertbildung verkleinert die Chromatogramme nicht nur, sondern glättet sie auch.

Dieser Trend ist in der Abbildung durchaus erkennbar; die recht groben Sprünge im Bereich bis 160 Eingabeneuronen verwundern aber zunächst. Vermutlich ergibt sich dieses Verhalten aus der Verschiebung der Intervallgrenzen bei Veränderung des Bereichs für die Mittelwertbildung. Wenn ein Peak zufällig genau auf der Grenze zwischen zwei Intervallen liegt, geht offensichtlich mehr Information

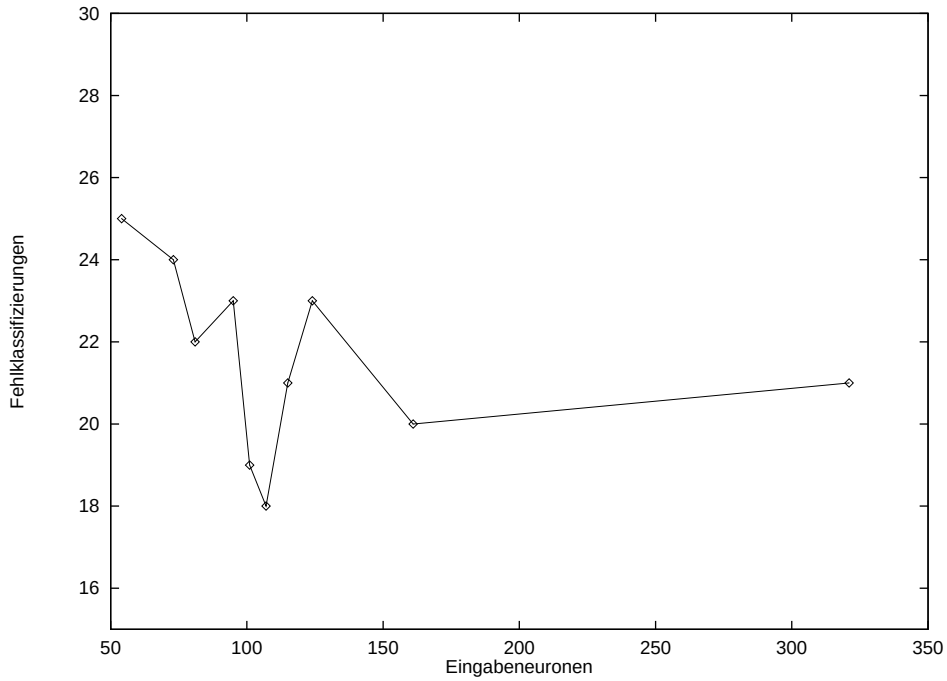


Abbildung 13: Einfluß der Anzahl der Eingabeneuronen, RBF-Netz

bei der Mittelung verloren, als wenn ein Peak in der Intervallmitte liegt. Falls dies einen oder mehrere für die Klassifizierung wichtige Peaks betrifft, so ist offensichtlich, daß der Fehler steigen kann, obwohl eine Veränderung der Neuronenzahl in Richtung des Optimums erfolgt ist.

Bei der Auswertung der zweiten Meßreihe brach die Leistung ähnlich stark ein, wie es bei Backpropagation zu beobachten war. Es gelang maximal, 38 Proben korrekt zu klassifizieren (unter Verwendung des Massenbereiches ab 50, mit 98 Eingabeneuronen, `killbaseline` und `log`).

7 Ergebnisse im Vergleich

7.1 Bedeutung der Vorbearbeitung

Bei allen vorgestellten Methoden wurden die besten Ergebnisse mit logarithmierten oder digitalisierten Daten erzielt. Dies ist zunächst überraschend, da auf diese Art bearbeitete Chromatogramme für den Betrachter praktisch nicht mehr als solche erkennbar sind. Ein visueller Vergleich gelingt am besten bei lediglich mit `baseline` oder `killbaseline` behandelten Daten.

Diese Daten sind nicht optimal für die Hauptkomponentenanalyse oder neuronale Netze, da die Chromatogramme ähnlicher Proben in unbearbeiteter Form nur von komplexeren Auswertesystemen korrekt bewertet werden, nicht aber von einfachen mathematischen Verfahren. Ein Mensch, der Chromatogramme vergleicht, führt dabei automatisch eine Peakerkennung durch. Wenn zwei Proben bei derselben Retentionszeit einen kleinen und einen großen Peak zeigen, so werden sie als untereinander ähnlicher angesehen als die Probe mit dem kleinen Peak und eine Probe, bei der kein Peak zu sehen ist. Für den Betrachter ist das einfach; wenn aber bei einem neuronalen Netz beispielsweise der Abstand zwischen zwei Datenvektoren als Ähnlichkeitsmaß genommen wird, so sind sich „kein Peak“ und „kleiner Peak“ ähnlicher als „kleiner Peak“ und „großer Peak“. Eine Logarithmierung löst dieses Problem zumindest teilweise.

Ein deutlich radikalerer Schritt ist die Digitalisierung. Dabei wird der Informationsgehalt des Chromatogramms auf „hier ist ein Peak“ und „hier ist kein Peak“ reduziert. Die ursprüngliche Größe des Peaks findet sich praktisch nur noch in der Breite wieder, wobei diese durch die Mittelwertbildung von `chrom2chrom` etwas nivelliert wird.

7.2 Erfolg der Klassifizierung durch neuronale Netze

Mit allen verwendeten Arten von neuronalen Netzen war eine praktisch fehlerfreie (> 97%) Reklassifizierung erreichbar. Im Rahmen der Validierung nach dem leave one out - Verfahren lieferten RBF-DDA und Backpropagation-Netze fast gleich gute Ergebnisse. Mit beiden Algorithmen konnten bei der ersten Meßreihe 51 der 68 Proben, also 75%, korrekt klassifiziert werden. Für die zweite Meßreihe betrug dieser Anteil bei Backpropagation 39 Proben, bei RBF-DDA 38, also 57% und 56%.

Die Ergebnisse sind quantitativ praktisch identisch. In der Wahl der Klassengrenzen gibt es zwar Unterschiede, diese allein würden aber nicht unbedingt zur Auswahl eines der beiden Verfahren genügen. Aus praktischen Gesichtspunkten allerdings ist den RBF-Netzen eindeutig der Vorzug zu geben. Das Training ist wesentlich schneller, die Einstellung der Lernparameter ist viel unproblematischer. Die Größe der verborgenen Schicht wird nach Bedarf automatisch ermittelt und muß nicht vom Anwender optimiert werden.

Der DLVQ-Algorithmus erlaubte mit den Daten der ersten Meßreihe 44 korrekte Klassifizierungen (65%), mit den Daten des zweiten Blocks immerhin noch 41 (60%). Verglichen mit RBF/Backpropagation ist die Leistung bei der ersten Reihe deutlich schlechter, nimmt dafür aber bei den weniger guten Chromatogrammen der zweiten Reihe nicht so stark ab. Für diesen Datensatz ist DLVQ sogar besser geeignet als die anderen Verfahren.

7.3 Vergleich der Ergebnisse der Hauptkomponentenanalyse mit denen neuronaler Netze

Wie schon in Abschnitt 3.2 erwähnt, liefert die Hauptkomponentenanalyse keine einfach durch eine Zahl erfaßbaren Ergebnisse. Der Sinn liegt vielmehr in

der Datenerkundung und in der Möglichkeit, manuell einzelne Klassifizierungen durchführen zu können, ohne ein richtiges Klassifizierungssystem dafür entwickeln zu müssen.

Diese Ansprüche wurden im wesentlichen erfüllt. Die manuelle Klassifizierung ist zwar schwierig, aber durchaus möglich. Die Vermutung aufgrund der Score Plots, daß die automatische Klassifizierung nicht problemlos verlaufen würde, aber prinzipiell möglich sei, hat sich bestätigt. Die häufigsten Klassifizierungsfehler durch die neuronalen Netze traten da auf, wo auch in der PCA starke Überlappungen zwischen den Klassen erkennbar waren.

Ein Vorteil der manuellen gegenüber der automatischen Klassifizierung ist die Möglichkeit einer – wenn auch subjektiven – Bewertung der Sicherheit der Klassifizierung. Dies ist bei den verwendeten neuronalen Netzen nur im Fall der Backpropagation-Netze möglich. RBF-DDA und DLVQ erzeugen diskrete Ausgabewerte, die keinen Schluß darauf zulassen, wie sicher oder wie unsicher das Ergebnis ist. Bei Backpropagation ist eine solche Einstufung prinzipiell machbar (siehe auch Abschnitt 8.3). Bei der PCA ist die Lage einfacher: Wenn der Auswerter Probleme bei der Klassifizierung hat, ist das Ergebnis nicht so sicher, als wenn er die Klassenzugehörigkeit schon mit einem Blick auf den Score-Plot der ersten beiden Hauptkomponenten sehen kann.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

Im Rahmen dieser Arbeit wurden mehrere, auf verschiedenen Arten von künstlichen neuronalen Netzen basierende Klassifizierungssysteme für Spirituosen entwickelt. Bei einer Meßzyklusdauer von 16 oder 29 Minuten gelang die korrekte

Klassifizierung von 56 bis 75% nicht zum Training des Systems benutzter Proben. Hierzu wurden folgende Arbeitsschritte durchgeführt:

Es wurden zwei Headspace-GC/MS-Meßmethoden für das GCQ entwickelt, die die reproduzierbare Gewinnung von „Fingerabdrücken“ der Proben in akzeptabler Zeit und praktisch ohne Probenvorbereitung erlauben.

Um die gestellten Aufgaben zu bearbeiten, wurden folgende Programme zur Datenreduktion und -filterung sowie zur Validierung entwickelt:

- `ms2txt, ms2chrom, msto3d:` Konvertierung des GCQ-Datenformates in ASCII-Tabellen
- Bearbeitung der so erzeugten Chromatogramme:
  - `baseline:` einfache Basislinienkorrektur und Normierung
  - `killbaseline:` Unterdrückung von Grundlinienschwankungen
  - `chrom2chrom:` Datenreduktion durch Mittelwertbildung und Auswahl eines Teilbereiches zur Weiterverarbeitung
  - `digitize, log:` Digitalisierung und Logarithmierung
- `chrom2list, chrom2pat:` Erzeugung von Eingabedateien für die Programme Unscrambler (Hauptkomponentenanalyse) und SNNS (neuronale Netze)
- `validate:` Validierung nach dem leave one out - Verfahren mit SNNS

Mit den gewonnenen Daten der Spirituosenproben wurden – exemplarisch für multivariate Methoden – Hauptkomponentenanalysen durchgeführt, die zeigten, daß eine automatische Klassifizierung durch neuronale Netze schwierig (da sich die sechs Klassen deutlich überschneiden), aber prinzipiell möglich ist.

Dazu wurden drei verschiedene Typen künstlicher neuronaler Netze untersucht, die jeweils in Hinsicht auf eine möglichst gute Klassifizierung nicht zum

Training eingesetzter Proben optimiert wurden. Es stellte sich heraus, daß Backpropagation- und RBF-DDA-Netze für diese Anwendung praktisch gleich leistungsfähig sind, wobei die letzteren sich als deutlich leichter zu optimieren und um ein Vielfaches schneller erwiesen. Mit der langsameren GC-Meßmethode konnten 75%, mit der schnelleren um 57% der Proben im leave one out - Verfahren korrekt klassifiziert werden. Der DLVQ-Algorithmus erreichte dagegen 65% und 60% korrekter Klassifizierungen, war also mit gut aufgelösten Chromatogrammen schlechter als die Konkurrenzverfahren, mit schlecht aufgelösten Chromatogrammen besser.

Diese Systeme können nach erfolgtem Training in Sekundenschnelle unbekannte Proben bewerten, wobei prinzipiell keine Interaktion mit dem Benutzer nötig ist. Eine Klassifizierung mittels Hauptkomponentenanalyse nimmt hingegen mehrere Minuten Arbeitszeit eines erfahrenen Anwenders in Anspruch.

Die entwickelten Klassifizierungssysteme sind modular aufgebaut, schnell, vollständig automatisierbar und — unter Berücksichtigung der sich überlappenden Klassen — sehr leistungsfähig.

8.2 Verbesserung der Vorbearbeitung

Im Bereich der Datenerfassung und Vorbearbeitung sind sicherlich noch Verbesserungen möglich. Relativ einfach zu realisieren wären weitere Filter für die Bearbeitung der Chromatogramme wie beispielsweise Glättungsfunktionen oder eine intelligentere Basislinienkorrektur, die auch in der Lage wäre, eine Basislinendrift zu kompensieren.

Bei der Entwicklung von verbesserten Klassifizierern könnte die Datenverarbeitung des Sehsystems bei Menschen und Tieren als Vorbild für eine leistungsfähige Mustererkennung dienen. Es gibt Gruppen von Neuronen, die darauf spezialisiert

sind, beispielsweise senkrechte oder waagerechte Linien zu erkennen. Diese reduzierten Daten erleichtern die Arbeit der nachgeschalteten Neuronen, die komplexere Muster erkennen sollen.

Analog hierzu könnten die Eingaben für ein die eigentliche Klassifizierung durchführendes Backpropagation-Netz nicht die einfach vorbearbeiteten Chromatogramme sein, sondern Vektoren, die die Gestalt des Chromatogramms kodiert wiedergeben wie durch Angabe der Zahl und Größe der Peaks in einem bestimmten Retentionszeitfenster. Ein derartiges vorgeschaltetes Peak-Erkennungssystem könnte vermutlich die Klassifizierungsleistung steigern, erhöht allerdings gleichzeitig die Komplexität des Systems.

8.3 Verbesserung des Auswerteverfahrens

Backpropagation-Netze liefern einen Ausgabevektor, der im Gegensatz zu denen der RBF-DDA und DLVQ-Netze keine diskreten Werte enthält, sondern mit jeder seiner Komponenten einen kontinuierlichen Wertebereich überstreicht. Als Ergebnis wird (bei einer binären Kodierung der Ergebnisse für das Training) die Nummer der Komponente mit dem höchsten Wert angegeben. Hier gibt es eine Möglichkeit für eine verbesserte Darstellung des Ergebnisses.

Je nach Abstand zur nächstkleineren Komponente könnte eine Schätzung der Sicherheit der Klassifizierung abgegeben werden. Im Rahmen dieser Arbeit ist diese Information unwichtig, da die Vergleichsklassifizierer solche Einstufungen nicht erlauben. Für eine eventuelle Anwendung eines derartigen Systems – beispielsweise in der Qualitätskontrolle – wäre sie aber eine wertvolle Ergänzung.

Für eine konkrete Verwendung ist es sicher interessanter zu wissen: „Diese Klassifizierung ist mit 43% Wahrscheinlichkeit falsch“ als nur die Information zu haben: „Im allgemeinen stimmen 75% der Klassifizierungen“.

Wie in Abschnitt 6.4.1 beschrieben, stellt die hohe Anzahl veränderlicher Parameter — wie Art der Vorbearbeitung (mit den zugehörigen Parametern), Anzahl der Eingabenneuronen oder der verborgenen Neuronen, Werte der Lernparameter — den Anwender vor das Problem, daß die Suche nach einem möglichst guten Klassifizierer sehr zeitraubend und vor allem nicht automatisch ist. Ein in den letzten Jahren populär gewordener Ansatz zur Lösung vergleichbarer Optimierungsprobleme sind genetische Algorithmen. Hier werden verschiedene Parametersätze (vergleichbar mit der DNA) „mutiert“, kombiniert und, im Falle einer relativ zu anderen „Individuen“ deutlich schlechteren Leistung, eliminiert. Das Ergebnis wäre eine „Population“ von vergleichbar guten Parametersätzen.

Es gibt bei diesem Verfahren keine Garantie, das Optimum zu finden, jedoch stellt es eine Möglichkeit dar, aus einer fast beliebig großen Anzahl von Alternativen einige akzeptable auszuwählen. Ein solches System zur vollautomatischen Optimierung eines auf neuronalen Netzen basierenden Klassifizierungssystems scheint von der theoretischen Seite durchaus erfolgversprechend, hätte jedoch den Rahmen dieser Arbeit gesprengt und mehr als die verfügbare Rechenleistung erfordert.

9 Literaturverzeichnis

[1] Y. Horimoto, K. Lee und S. Nakai: Classification of Microbial Defects in Milk Using A Dynamic Headspace Gas Chromatograph and Computer-Aided Data Processing, J. Agric. Food Chem. (1997), 45(3), 743-747

[2] M. Lipp: Determination of the adulteration of butter fat by its triglyceride composition obtained by GC, Food Chem. (1996), 55(4), 389-95

[3] Z. Cao, H. Lin, B. Wang, K. Wang und R. Yu: Piezoelectric crystal sensor array used as an organic vapor sensing system, Microchem. J. (1995), 52(2),

174-80

[4] L. Montanarella, M. R. Bassani, O. Breas: Chemometric classification of some European wines using pyrolysis mass spectrometry, Rapid Commun. Mass Spectrom. (1995), 9(15), 1589-93

[5] W. H. Calvin und K. Graubard: Styles of neuronal computation, The Neurosciences, Fourth Study Program, MIT Press 1979

[6] W. H. Calvin und G. A. Ojemann: Inside the Brain, New American Library, New York 1980

[7] P. Schreier, L. Reiner: Biometric evaluation of gas chromatographic data in the differentiation of spirits, Dtsch. Lebensm.-Rundsch. (1977), 73(12), 387-90

[8] M. Otto: Chemometrie - Statistik und Computereinsatz in der Analytik, VCH Weinheim, 1997

[9] O.Štrouf: Chemical Pattern Recognition, Research Studies Press Ltd., 1986

[10] J. Schürmann: Pattern Classification - a unified view of statistical and neural approaches, John Wiley & Sons, 1996

[11] The Unscrambler, Camo AS, Olav Tryggvasonsgt. 24, N-7011, Trondheim, Norway, http://www.camo.no

[12] SNNS - Der Stuttgart Neural Network Simulator, nähere Informationen unter http://www-ra.informatik.uni-tuebingen.de/SNNS/

[13] J. Schürmann und U. Kreßel: Mustererkennung mit statistischen Methoden. Daimler-Benz AG, Forschungszentrum Ulm, Institut für Informatik, 1992

[14] R. Duda und P. Hart: Pattern Classification and Scene Analysis, Wiley& Sons, 1973

[15] M. R. Berthold und J. Diamond: Boosting the performance of rbf networks with dynamic decay adjustment, Advances in Neural Information Processing Systems, Band 7, 1995

A Liste der Proben

Die Zahl vor dem Probennamen ist die Probennummer. Die Proben wurden in der Reihenfolge des Eingangs numeriert. Folgende Proben (nach Klassen geordnet) wurden verwendet:

Weinbrand: 9 Chantré, 10 Metaxa, 11 Asbach Uralt, 17 Scharlachberg Meisterbrand, 77 Hennessy Cognac, 85 Springer Urvater

Grappa: 2 Grappa die Barbaresco, 3 Grappa Costa Russi, 4 Grappa di Prosecco, 5 Grappa di Monovitigno, 23 Grappa Piave, 79 Inga Grappa di Pinot-Chardonay, 80 Euganea Grappa, 84 Grappa Mabruzzo

Sprit: 6 Ron Blanco Old Hopking (weißer Rum), 7 Wodka Gorbatschow, 12 Wodka Moskovskaya, 14 Hasebrink Doppelkorn, 26 Jobelius Korn, 30 Puschkin Vodka, 35 Schulte Weizenkorn, 37 Rogoschin Vodka, 40 Bommerlunder, 46 Münsterländer Bastkorn, 48 Ostfrisische Fischermelodie (Korn), 51 Corvit (Korn), 53 Nichts (Korn), 58 Strothmann Weizenkorn, 61 Auerhahn Weizenkorn, 72 Volinska Gorilka, 73 Kreml Extra Vodka, 82 Polmos Polish Wodka

Obstler: 8 Specht Bauernschnäpsle, 13 Manastirka Stara Sljivova Prepecenica, 15 Schwäbisches Obstwässerle, 16 Kranz Schwarzwälder Zwetschgenwasser, 21 Andreas Haas Obstwässerli, 28 Weis Schwarzwälder Zwetschgenwasser, 33 Flobs Apfelkorn, 41 Nordkirchener Apfelblüte, 45 Obstler

Goldmarke, 50 Scheibel Kirschwasser, 52 Aprikosengeist (Ungarn), 54 Mirabelle, 57 Schwarzwälder Himbeergeist, 59 Tiroler Zwetschkenbrand, 60 Osttiroler Vogelbeer, 62 Specht Kirschwasser, 65 Männle Schwarzwälder Zwetschgenwasser, 69 Golden Delicious, 83 Calvados Dauphine, 88 Specht Waldhimbeergeist

Kräuter: 18 Karlovarska Becherovka, 22 Alpengold Enzianlikör, 29 Karlsbader Becherovka, 31 Unicum Bitterlikör, 36 Sanddorn Edelbitter, 44 Stobbe würziger Machandel, 47 Stroh Enzian, 49 Mäckingerbach Wasser, 55 Hirtenfeuer, 56 Amaro Siciliano, 63 Kümmerling, 68 Jägermeister,

Rum: 64 Polar Rum, 66 Jamaika Rum Verschnitt, 70 Asmussen Jamaica Rum, 71 Steuerrad echter Übersee Rum