Entwicklung eines Systems zur dreidimensionalen Visualisierung von selbstorganisierenden Merkmalskarten

Diplomarbeit im Fach Computervisualistik

vorgelegt
von

Annekatrin Tretow

Geboren am 06.10.1979 in Rostock

Angefertigt am

Institut für Computervisualistik
Arbeitsgruppe Computergraphik
Universität Koblenz–Landau

Betreuer: Dipl. Inform. Ulrich Penndorf (Altran IT),
Dipl.Inform. Markus Ehl (Altran IT)
Dipl.Inform. Matthias Biedermann (Universität Koblenz-Landau)

Beginn der Arbeit: 28.11.2005

Abgabe der Arbeit: 28.05.2005

Danksagung

Direkt am Anfang möchte ich allen Personen danken, die durch ihre fachliche und persönliche Unterstützung direkt oder indirekt zum Gelingen dieser Arbeit beigetragen haben.

An erster Stelle danke ich Prof. Dr. Stefan Müller, der diese Arbeit erst ermöglicht hat.

Bei Dipl. Inform. Markus Ehl und Dipl. Inform. Ulrich Penndorf von der Firma *Altran IT* möchte ich mich zum einen für die Möglichkeit, in Zusammenarbeit mit *Altran IT* meine Diplomarbeit schreiben zu dürfen und zum anderen für die interessante Aufgabenstellung bedanken.

Für die Betreuungsarbeit bedanke ich mich bei Dipl. Inform. Matthias Biedermann und Dipl. Inform. Ulrich Penndorf. Mit zahlreichen Ratschlägen, Hilfestellungen, konstruktiver Kritik und einer genauen Korrektur meiner Texte haben sie zur Verbesserung dieser Arbeit beigetragen. Ein weiterer Dank geht an Dr. Volker Riediger der für Fragen aller Art immer ein offenes Ohr hatte – obwohl ich unbeirrbar seine beharrlichen Vorschläge Eclipse zu nutzen ignoriert habe.

Ebenfalls danke ich meinen Freunden für ihre Hilfsbereitschaft, Offenheit und das eifrige Korrekturlesen meiner Arbeit. Während des Studiums haben sie mich unterstützt, motiviert und mir über kurze Durststrecken hinweggeholfen.

Ein ganz besonderer Dank gilt meinen Eltern die mir das Studium ermöglicht und mich in jeder Hinsicht unterstützt haben.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Aufgabenstellung

Aufgrund der immer billiger und benutzerfreundlicher werdenden Computer- und Speichertechnologien, können heutzutage in Industrieunternehmen riesige Datenmengen erhoben und gespeichert werden. Diese Datenbestände werden aber erst dann wirklich interessant und zum Wettbewerbs- oder Effizienzvorteil, wenn es gelingt, die darin enthaltenen, nicht trivialen Informationen herauszufiltern.

Dieses ist die zentrale Aufgabe des Data Mining: Mit Hilfe mathematisch-statistischer Mittel werden Datenbestände analysiert und aufbereitet. Ziel ist das Auffinden von Auffälligkeiten und Zusammenhängen. So lassen sich Regelmäßigkeiten oder Veränderungen im Verhalten der Kunden aufspüren.
Eine der bedeutenden Anwendungen des Data Mining ist die Clusteranalyse, für welche die Selbstorganisierenden Merkmalskarten (engl. Self Organinzing Maps (SOM)) ein etabliertes Verfahren darstellen. Sie bezeichnen ein Segmentierungsverfahren, welches auf dem Prinzip der neuronalen Netze basiert und selbstständig eine Klassifizierung innerhalb eines Datensatzes bildet. Durch eine Visualisierung der analysierten Daten kann die Erkennung von Mustern oder Abweichungen stark vereinfacht werden.

Ziel der Diplomarbeit ist die Entwicklung eines Systems, in welchem verschiedene Visualisierungsansichten der SOM mit Ausführung externer Programme über eine Datenschnittstelle kombiniert werden. Mittels verschiedener Möglichkeiten zur Interaktion sollen dem Benutzer eine Vielfalt unterschiedlicher Informationen zugänglich gemacht werden.

Die Diplomarbeit wird bei der Eurospace IT in Koblenz durchgeführt.

Inhaltsverzeichnis

Abkürzungsverzeichnis

DM	Data Mining
ESOM	Emergente Selbst-Organisierende Merkmalskarten
KDD	Knowledge Discovery in Databases
KNN	Kunstliche Neuronale Netze
VDM	Visual Data Mining

Verzeichnis der Bilder

Kapitel 1

Einleitung

Da diese Diplomarbeit bei der Firma *Altran IT* geschrieben wird, werden in diesem einleitenden Kapitel zuerst die Firma *Altran IT* und ihre Tätigkeitsfelder vorgestellt. Danach wird die Problemstellung, die Motivation und der Aufbau der Arbeit erläutert.

1.1 Firmenprofil

Die Firma *Altran IT* ist ein Business- und IT-Consultingunternehmen und wurde im Frühjahr 2006 durch Zusammenschluss der drei deutschen ALTRAN-Unternehmen *IT-Bereich der BERATA GmbH*, *CHS Data Systems GmbH* und *EUROSPACE IT GmbH* gegründet. Der Hauptsitz von *Altran IT* befindet sich in Frankfurt am Main. Weitere Standorte der Firma sind in Hamburg, Wolfsburg, Koblenz, Böblingen und München.

Die Kernkompetenzen von Altran IT liegen in der Konzeption, Umsetzung und Management von IT-Lösungen. Dabei werden folgende Themenschwerpunkte gesetzt [1]:

- **Customer Relationship Management (CRM)** ist der Oberbegriff für eine Vielzahl von Systemen, welche einem Unternehmen helfen sollen, eine umfassende Sicht auf seine Kunden zu bekommen, um die Bedürfnisse der Kunden zu erkennen und ein zielgerichtetes Handeln des Unternehmens zu ermöglichen. So muss zum Beispiel die Einsicht in den Bestellvorgang oder der Rechnungsabruf im Call-Center möglich sein, wenn eine telefonische Kundenbeschwerde eintrifft.

- **Business Intelligence (BI)** ist der Sammelbegriff für ein breites Spektrum von analytischen Prozessen, Anwendungen und Technologien welche das Auswerten von Daten und das Bereitstellen von bisher unbekanntem Wissen ermöglichen. Mithilfe von BI können rohe Daten in nutzbares Wissen umgewandelt werden.

- **Enterprise Ressource Planning (ERP)** zielt darauf ab, die vorhandenen Ressourcen eines Unternehmens durch eine prozessorientierte Abbildung und Optimierung der betriebswirtschaftlichen Abläufe optimal einzusetzen. Dabei werden Kapital, Personal und der rein betriebliche Ablauf berücksichtigt. Nach der Abbildung erfolgt die Umsetzung und die Steuerung der optimierten Abläufe. Analysen und Auswertungen geben weitere Einsicht in die Verwaltung der Ressourcen.

- **Innovationsmanagement (IM)** beinhaltet die methodische Planung, Realisierung und Kontrolle von Ideen in Organisationen.

Zu den Kunden von *Altran IT* zählen große und mittelständische Unternehmen aus den Branchen Handel/Telekommunikation, Finanzdienstleistung, Telekommunikation, Automotive und Fertigungsindustrie.

1.2 Aufgabenstellung

Die Aufgabenstellung dieser Diplomarbeit ist dem Bereich des Data Mining[1] zuzuordnen. Ziel ist es ein Programm zu entwickeln, welches unter der in ESOM bereits vorhandenen Implementationen zur Datenanalyse, die Ergebnisse dieser Analyse dreidimensional visualisieren kann. Durch Kombination verschiedener Visualisierungsansichten und Interaktionsmöglichkeiten soll der Benutzer des Programmes sich einen Überblick über die Beziehungen zwischen den Daten machen können. Durch die Spezifikation einer offenen Schnittstelle soll es anderen Programmen ermöglicht werden, mit dem vorliegendem Programm zu kommunizieren.

[1] Data Mining ist der Oberbegriff einer Vielzahl verschiedener Anwendungen, um relevante Informationen aus einer Menge von Daten zu extrahieren.

1.3 Motivation

In jedem Unternehmen, sei es in der Produktion, im Vertrieb, in der Lagerhaltung oder im Personalwesen, fällt heutzutage eine riesige Menge von Daten an. Dank der rasanten technologischen Entwicklung in den letzten Jahren ist es den Unternehmen möglich, all diese Daten zu verarbeiten und auch zu speichern. So liegen den Unternehmen mitunter sehr detaillierte Informationen zum Beispiel über ihren Kunden, Interessenten oder auch Mitbewerbern vor. Neben der bloßen Adresse werden oftmals soziodemographische Daten, Kaufinformationen, Potentialdaten sowie Kommunikationsdaten gespeichert. Diese Informationen werden in der Regel genutzt, um direkt mit dem einzelnen Kunden zu kommunizieren. Auch einfache Managementfragen lassen sich mit Hilfe dieser Daten beantworten. So stellt es kein Problem dar, die Anzahl oder das Durchschnittsalter neuer Kunden oder Interessenten auszugeben.

Man könnte nun meinen, dass alles erreicht sei, wenn es den Unternehmen gelingt, Daten in ausreichendem Umfang zur Verfügung zu haben. Doch die Daten allein genügen häufig nicht. Die Einzelinformationen sind zwar für den Einzelfall sehr wichtig, doch allgemeinere Muster, Strukturen, Regelmäßigkeiten oder Ausnahmefälle bleiben häufig unbemerkt. Gerade diese Muster können aber für viele Probleme und Fragestellungen eines Unternehmens einen Lösungsansatz bieten und sich zum Beispiel für eine Umsatzsteigerung nutzen lassen. Findet man beispielsweise heraus, dass in einem Supermarkt bestimmte Produkte häufig zusammen verkauft werden, so kann die Verkaufszahl unter Umständen durch eine entsprechende Anordnung dieser Produkte gesteigert werden.

Trotzdem werden die Datenbanken heutzutage nur in wenigen Fällen für die Beantwortung solcher Fragen genutzt. Andere Fragestellungen, die eine allgemeine Sicht auf den Datenbestand für ihre Lösung benötigen, wären:

- Welchen Kunden sollte wann welches Angebot unterbreitet werden?
- Bei welchem Kundenprofil lohnt sich ein Außendienstbesuch?
- Welcher Umsatz wird im kommenden Jahr erzielt?

Aber warum bleiben gerade diese, für das Management so entscheidende, Fragen unbeantwortet?

Betrachtet man den Charakter der Fragestellungen, so liegt deren Antwort nicht in einem einzelnen Datenfeld oder einem Kundenmerkmal, sondern in der richtigen Kombination unterschiedlicher Informationen. So kann die Angebotsaffinität eines Kunden von einer Vielzahl von Merkmalen wie

dem Alter, Geschlecht, Familienstand, bisher gekauften Produkten, Zahlungsmoral und anderen Eigenschaften abhängen.

An dieser Stelle setzt das Data Mining an. Es ergänzt die statistischen Verfahren um neue Analysemethoden, die einen Großteil der Untersuchungsprozesse automatisieren und beschleunigen. Bildlich gesprochen durchforsten Data Mining Technologien eine Datensammlung nach Regelmäßigkeiten, Zusammenhängen und Ausnahmen selbständig.

In dieser Arbeit geht es darum, die Ergebnisse der Anwendung einer Technik des Data Mining zu visualisieren. Bei dieser Technik handelt es sich um die Selbstorganisierenden Merkmalskarten, welche ein etabliertes Verfahren der Clusteranalyse darstellen. Die Clusteranalyse wiederum ist nur eine von vielen Verfahren des Data Mining.

1.4 Aufbau der Arbeit

Das zweite Kapitel dieser Arbeit soll eine Einführung in die Thematik des Data Mining geben. Dazu wird zuerst das Data Mining als ein Schritt innerhalb eines Prozesses zur Wissensextraktion, des *Knowledge Discovery in Databases*, erläutert. Danach werden dann die wichtigsten Verfahren des Data Mining kurz erwähnt. Zuletzt wird auf die Data Mining-Technik der *Clusteranalyse* und dann auf *Selbstorganisierenden Merkmalskarten* als eine Technik der Clusteranalyse genauer eingegangen.

Im dritten Kapitel wird zuerst eine Anforderungsliste an das zu entwickelnde Programm erstellt. Im Anschluss daran werden Lösungsmöglichkeiten zur Erfüllung dieser Anforderungen, der gewählte Lösungsweg und dessen theoretische Umsetzung erläutert.

Im vierten Kapitel wird ausführlich auf die Implementation des Programmes eingegangen. Zuerst wird die Architektur des Programmes, die einzelnen Bausteine und deren Zusammenarbeit, beschrieben. Danach werden Probleme, welche sich während des Implementationsprozesses ergaben, und ihre Lösungen erläutert.

Im sechsten und letzten Kapitel wird eine Zusammenfassung gegeben. Dabei werden die Möglichkeiten und Grenzen des Programmes diskutiert und Weiterentwicklungsmöglichkeiten vorgestellt.

Kapitel 2

Hintergrund

Dieses Kapitel soll nun näher in die Thematik des Data Mining einführen und Grundbegriffe klären, welche für das Verständnis der folgenden Kapitel notwendig sind.

Dabei wird zunächst ein Überblick über die Thematik gegeben. Danach wird der Prozess der Wissensextraktion aus Datenbanken *(Knowledge Discovery in Databases (KDD))* beschrieben. Anschließend werden kurz die gebräuchlichsten Methoden des Data Mining erklärt. Um die Ideen des Data Mining zu veranschaulichen, werden im dritten Abschnitt dieses Kapitels einige praktische Anwendungsbeispiele des Data Mining genannt. Danach wird ein Überblick über die Data Mining-Methode der *Clusteranalyse* gegeben. Im letzten Abschnitt wird dann das Verfahren der *Selbstorganisierenden Merkmalskarten* als ein Verfahren der Clusteranalyse erläutert.

2.1 Überblick

Das Einkaufsverhalten im Supermarkt, die Strategie bei Geldgeschäften, das Verhalten im Internet oder auch das Reiseziel für den nächsten Urlaub – fast alle Entscheidungen, die im öffentlichen Leben getroffen werden, werden tagtäglich elektronisch aufgezeichnet und erzeugen einen weiteren Datensatz in einer Datenbank.

Schon vor Jahrzehnten haben Unternehmen damit begonnen, alle anfallenden Informationen, wie Mitbewerber- oder Kundendaten zu sammeln und zu archivieren [2]. Begünstigt durch preiswerte Hardware und leistungsfähige Datenbanksysteme, welche eine effiziente und zuverlässige Speiche-

rung und einen schnellen Zugriff auf sehr umfangreiche Datenbestände ermöglichen, nahm diese „Sammelleidenschaft“ in den letzten Jahren noch zu: In Wirtschaft, Wissenschaft und Verwaltung (Kundenverwaltung, Auftragsbearbeitung, Telekommunikationsprotokolle, und andere) werden große Mengen an unterschiedlichen Daten gesammelt, die bei Bedarf wieder abgerufen werden können. Doch das weitaus größere Potential dieser Daten liegt in den Informationen, welche sich aus den Relationen zwischen ihnen ergeben. Diese Informationen stellen häufig einen Schlüssel für viele Problemlösungen dar.

Das Ziel des Konzipierens und Aufstellens riesiger Datenbanken ist daher nicht nur die Anhäufung eines riesigen Datenbergs, sondern häufig die Extraktion des Wissens, welches sich aus den Relationen zwischen den Daten ergibt. Der dabei durchlaufene Prozess wird als Knowledge Discovery in Databases (Wissensextraktion aus Datenbanken) oder Data Mining (DM) bezeichnet. Diese zwei Begriffe werden in den zahlreichen auf diesem Gebiet erschienenen Veröffentlichungen, teilweise sehr unterschiedlich definiert. In der Informatik wird beispielsweise deutlich zwischen Data Mining und KDD unterschieden, während in anderen Gebieten, wie der Chemoinformatik diese beiden Begriffe häufig als Synonyme verwendet werden.

Die in dieser Arbeit verwendete Definition bezieht sich auf Fayyad, welcher den KDD-Prozess als einen Prozess zur Identifikation verborgenen Wissens aus Datenbanken beschreibt, in dem das Data Mining einen wichtigen Arbeitsschritt darstellt [3].

2.2 Knowledge Discovery in Databases (KDD)

Knowledge Discovery in Databases beschreibt den prozessorientierten Rahmen, um aus Rohdaten sinnvolles Wissen ableiten zu können. Der KDD-Prozess wurde wie folgt definiert:

> *„Knowledge Discovery in Databases (KDD, Wissensextraktion aus Datenbanken) bezeichnet den nicht trivialen, mehrstufigen Prozess der Identifikation valider, neuartiger, potentiell nützlicher und klar verständlicher Beziehungen und Regelmäßigkeiten zwischen Datensätzen oder den Daten innerhalb eines Datensatzes.“* [3]

Die grundsätzlichen Stufen des KDD-Prozesses sind in Abbildung 2.1 schematisch dargestellt.

Der KDD-Prozess ist ein interaktiver und iterativer Prozess der aus folgenden Teilschritten besteht [3]:

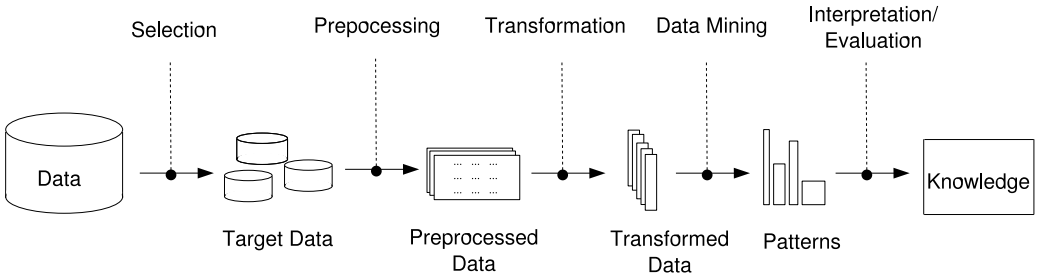


Bild 2.1: KDD-Prozessmodell (Fayyad, 1996)

- Selection
  Mit diesem ersten Schritt wird eine Auswahl der zu untersuchenden Daten getroffen. Hier werden Teildatenbestände gebildet, oder auch relevante Daten aus unterschiedlichen operationalen (eventuell auch externen) Datenquellen zusammengeführt.

- Preprocessing
  Dieser Teilschritt dient der Bereinigung und Aufbereitung der ausgewählten Daten. Hier werden unter anderem falsche Einträge korrigiert, Extremwerte (Ausreißer) eliminiert oder auch Ergänzungen vorgenommen.

- Transformation
  In diesem Schritt werden die Daten für die Analyse weiter vorbereitet. Sie können beispielsweise durch Reduktion von Dimensionen oder anderen Transformationsmethoden reduziert werden.

- Data Mining
  Der Schritt des Data Mining ist das Herzstück des KDD-Prozesses und bezeichnet ein Forschungs- und Anwendungsgebiet, welches sich mit der Entwicklung und Anwendung von Techniken zur Wissensextraktion aus großen Datensätzen befasst. In diesem Teilschritt werden die selektierten und vorbereiteten Daten durch Anwendung von Data-Mining-Techniken automatisch oder halbautomatisch nach Zusammenhängen und Auffälligkeiten durchsucht.

- Interpretation/Evaluation
  Im letzten Schritt des KDD-Prozesses werden die Ergebnisse interpretiert und evaluiert. Falls die Ergebnisse als unzureichend erkannt werden, könne die Schritte beliebig oft wiederholt werden.

Da das Data Mining ein wichtiger Teilschritt des KDD-Prozesses ist, werden die Begriffe Data Mining und KDD häufig synonym verwendet. Allerdings sind die anderen Teilschritte des KDD-Prozesses nicht weniger wichtig um zu guten Ergebnissen zu kommen und KDD erfolgreich einsetzen zu können.

2.3 Data Mining-Methoden

Der Begriff Data Mining steht für eine Reihe verschiedener Methoden und Techniken, um relevante Informationen aus einer Menge von Daten zu extrahieren. Die Data Mining Techniken erzeugen Modelle, mit deren Hilfe Daten analysiert und prognostiziert werden können. Einsatzgebiete dieser Techniken sind vor allem die klassische Markt- oder Risikoanalyse, aber auch etwas ausgefallenere Gebiete, wie die Astronomie, Chemie oder Biologie. In der Literatur werden eine Vielzahl von Data Mining-Methoden vorgestellt. In Bezug auf [2] kann aber zwischen fünf wichtigen Data Mining Methoden unterschieden werden, welche sich den verschieden Data Mining Techniken zur Lösung von Problemen bedienen. Als Techniken bezeichnet man spezielle Implementierungen der Data Mining-Methoden. In diesem Abschnitt werden wichtigsten Methoden kurz erläutert und gegebenenfalls jeweils einige angewandten Techniken[1] benannt.

- Abhängigkeitsanalyse
  Diese Data Mining Technik zielt auf eine Ermittlung von interessanten Beziehungen (Assoziationen) oder Abhängigkeiten zwischen einzelnen Daten oder Datensätzen. Mit Hilfe dieser Beziehungen können dann Aussagen über das Verhalten von Objekten in Abhängigkeit eines einzelnen Objektes getroffen werden. Assoziationsregeln werden heutzutage häufig im Bereich der Kreditentscheidungen und in der Analyse von Registerkassenbons genutzt.

- Klassifikationstechniken
  Bei der Klassifikation werden die einzelnen Daten einer Datensammlung anhand von Entscheidungsregeln in vordefinierte, disjunkte Klassen eingeordnet. Auf diese Weise können Vorhersagen zu bestimmten Fragestellungen gemacht werden. Um eine Klassifikation

[1] Ein großes Problem beim Data Mining besteht in der Zuordnung der verschiedenen Techniken zu den Methoden: „Es gibt keine Eins-zu-eins- Zuordnung. [...] Im Sinne der Datenanalyse heißt das, dass man zur Problemlösung grundsätzlich verschiedene Techniken heranziehen sollte, um die Ergebnisse miteinander vergleichen zu können.“[2]

durchzuführen, stehen verschiedene Techniken bereit. Einige der wichtigen Techniken um eine Klassifikation durchzuführen sind:

- Entscheidungsbäume
  Bei der Anwendung von Entscheidungsbäumen werden die Daten einer Datensammlung in einer baumähnlichen Struktur gegliedert. Die Daten werden dabei nach bestimmten Merkmalen segmentiert. Mit Hilfe von Wenn-Dann-Abfragen können dann geltende Regeln innerhalb dieses Datensatzes abgefragt werden.
- Bayesche Netze
  Bayesche Netze sind eine klassische Technik, die genutzt wird, um die Wahrscheinlichkeit eines Ereignisses zu bestimmen, wenn andere Ereignisse bekannt sind.
- Künstliche neuronale Netze
  Ein künstliches neuronales Netz ist ein Computerprogramm, welches der biologischen Informationsverarbeitung nachempfunden wurde. Es lernt Zusammenhänge aufgrund von Trainingsbeispielen selbstständig.

Ein typisches Anwendungsbeispiel der Klassifikation ist die Kundensegmentierung zur Zielgruppenermittlung im Direktmarketing. Andere Einsatzbereiche sind die Diagnosehilfe oder die Risikoabschätzung bei Versicherungen und Banken.

- Sequentielle Muster
  Bei dieser Technik handelt es sich meist um das Erkennen zeitlicher Muster. So können Trends und geordnete Reihenfolgen aufgedeckt werden. Anwendungen finden sich zum Beispiel im Bank- oder Finanzwesen.

- Clusteranalyse
  Durch eine Clusteranalyse werden heterogene Daten anhand ihrer Eigenschaften in Gruppen (Cluster genannt) zusammengefasst. Die Homogenität der Daten innerhalb eines Clusters und die Heterogenität der Daten eines Cluster zu den Daten eines anderen Clusters sollten möglichst groß sein. Gelingt eine solche Einteilung, so können die Daten einer Gruppe gleich behandelt werden. Der Unterschied der Clusteranalyse zur Klassifikation besteht darin, dass bei der Clusteranalyse keine vordefinierten Klassen verwendet werden. Die Klassen entstehen erst während der Analyse. Einige wichtige Techniken um eine Clusteranalyse durchzuführen sind:

  - Selbstorganisierende Merkmalskarten
    Ein häufig verwendetes Verfahren zur Clusteranalyse sind die Selbstorganisierenden Merkmalskarten (siehe Abschnitt 2.6). Diese Technik ist ein Segmentierungsverfahren, das auf den Prinzipien neuronaler Netze basiert und selbstständig Cluster innerhalb eines Datensatzes bildet.
  - Warenkorbanalyse
    Ziel der Warenkorbanalyse ist das Auffinden von Gruppen von Produkten, welche häufig gemeinsam in einer Transaktion vorkommen [2]. Da für diese Technik auch assoziative Regeln erstellt werden, kann sie auch der Abhängigkeitsanalyse zugeordnet werden.
  - Statistische Techniken

  In den Kapiteln 2.5 und 2.6 wird genauer auf die Methode der Clusteranalyse und deren Technik der Selbstorganisierenden Merkmalskarten eingegangen.

- Vorhersage
  Aufgrund von analytischen Modellen, die die Daten sequentiell beschreiben, wird bei dieser Technik eine im Allgemeinen zeitliche Vorhersage gemacht. Anwendungen hierzu sind Zeitreihenmodelle im Finanzwesen wie die Vorhersage von Börsenkursen.

Der Einsatz der unterschiedlichen Methoden und Techniken hängt stark vom Charakter der Fragestellung ab. Auch werden bestimmte Eigenschaften der Methoden/Techniken bei der Auswahl herangezogen:

- Eignung der Methoden/Techniken für bestimmte Eingabedatentypen
- Transparenz der Ergebnisse
- Toleranz gegenüber fehlender Variablenwerten
- Fähigkeit große Datenmengen handhaben zu können
- und andere

Keine Data Mining Technik ist der alleinige Lösungsansatz für auftretende KDD-Probleme [2]. So bieten sich neuronale Netze und Entscheidungsbäume besonders bei Fragestellungen mit Prognosecharakter an. Eine eindeutige Zuordnung der Methoden nach Aufgabenstellung ist allerdings nicht möglich. Oftmals werden mehrere Data Mining-Lösungen für dieselbe Aufgabenstellung entwickelt und gegeneinander ausgetestet. Auch die Kombination unterschiedlicher Methoden und Techniken innerhalb einer Lösung ist möglich.

Doch so unterschiedlich die verschiedenen Methoden und Techniken auch einsetzbar sind, eine Eigenschaft haben sie alle gemein: Sie sind darauf spezialisiert, Zusammenhänge und Auffälligkeiten in Datensammlungen zu erkennen.

2.4 Anwendungsgebiete für Data Mining-Methoden

Das Data Mining findet praktische Anwendung in vielen Branchen: Banken, Versicherungen, Medizin und Pharmaforschung, Handel und E-Commerce-Shops sind nur einige Beispiele von Nutzern von Data Mining-Verfahren. Generell können die Probleme, für welche Data Mining-Techniken eine Lösung bieten können, in zwei Bereiche eingeteilt werden [3]:

1. **Beschreibungsprobleme**
   Diese Probleme beziehen sich auf das Finden von interpretierbaren Mustern in Daten. Ein Beispiel für ein Beschreibungsproblem wäre das Finden von Zusammenhängen in Kundeneinkäufen: Es kann festgestellt werden, dass Kunden, welche sich häufig Computerspiele kaufen, auch für Computerfachbücher oder Computerzubehör interessieren. Durch die Anwendung von Data Mining-Techniken können solche Zusammenhänge gefunden werden, die auf den ersten Blick nicht ersichtlich sind.

   Zur Lösung solcher Probleme eignen sich Data Mining-Methoden, die den Anwender bei der Suche nach Strukturen und Besonderheiten in Daten oder Datensätzen unterstützen. Das Ziel der Abhängigkeitsanalyse und der Clusteranalyse ist die Ermittlung von interessanten Beziehungen zwischen Daten oder Datensätzen. Damit stellen diese Verfahren zwei Beispiele gebräuchlicher Verfahren für die Lösung von Beschreibungsproblemen dar.

2. **Prognoseprobleme**
   Diese Probleme versuchen, Aussagen aus bestehenden Datenmengen über die Entwicklung dieser Daten zu schließen. Ein Beispiel für ein Prognoseproblem kann im Bereich der Ban-

ken gefunden werden: Bevor eine Bank einen Kredit vergibt, sollte sie die Kreditwürdigkeit des Kunden analysieren. Anhand bestehender Daten über Kunden, die ähnliche Eigenschaften wie der aktuelle Kunde haben, kann eine Prognose über die Kreditwürdigkeit erstellt werden. Eigenschaften über welche Kunden miteinander verglichen werden, können beispielsweise der Arbeitsplatz, das Alter oder der Familienstand sein.

Um Prognoseprobleme zu lösen, eignen sich unter anderem Verfahren der Klassifikation. Mit diesen Verfahren können Modelle konstruiert werden, mit deren Hilfe Objekte anhand ihrer Eigenschaften vordefinierten Klassen zugeordnet werden können. Durch diese Zuordnung kann das Objekt mit den klassenspezifischen Eigenschaften in Verbindung gebracht werden.

2.4.1 Anwendungsbeispiele

Sinnvolles Angebot neuer Bankingdienste

Banken waren frühe Nutzer der Data Mining-Technologie [4]. Hier wird sie unter anderem genutzt, um das Abwandern von Kunden zu verhindern, indem Veränderungen in verschiedenen Bankinggewohnheiten, die einen Bankwechsel nach sich ziehen könnten, detektiert werden: Es könnte sich zeigen, dass eine Kundengruppe, die ihre Bankgeschäfte über das Telefon erledigt, unzufrieden ist, wenn die Antwortzeiten sehr lang sind [4]. Mittels Data Mining können aber auch Gruppen von Bankkunden erkannt werden, für die das Angebot eines neuen Dienstes sinnvoll wäre.

Anordnung der Sortimente im Supermarkt

Die Einkaufswagenanalyse ist die Anwendung von Assoziierungstechniken, um Artikelgruppen zu erkennen, die in Transaktionen kombiniert auftreten. Diese Vorgehensweise wird in Supermärkten angewendet: Eine automatisierte Analyse der Kassenbons kann darauf hinweisen, dass Kunden, die Bier kaufen, auch an Chips interessiert sind. Diese Information könnte unter anderem zur Planung der Ladenarchitektur genutzt werden.

Gezieltes Direktmarketing

Das Direktmarketing ist ein weiterer Bereich, in dem Data Mining erfolgreich eingesetzt wird. Gezielte Kampagnen sind kostengünstiger als Massenkampagnen, weil Unternehmen damit Geld sparen, wenn sie nur solchen Kunden ein Angebot unterbreiten, die das Produkt wahrscheinlich gerne kaufen würden. Data Mining hilft dabei, genau diese Kunden ausfindig zu machen.

Bildanalyse

Tagtäglich wird eine große Anzahl von Daten aus entfernten Quellen, wie Satelliten, Flugzeugen oder Radaren empfangen. Militär und Wissenschaft haben großes Interesse an diesen Daten. Mit ihrer Hilfe kann das Verhalten feindlicher Objekte oder Anomalien der Atmosphäre und der Erdoberfläche erkannt und erforscht werden.

Lastabschätzung der Energieunternehmen

Für die Energieversorgeunternehmen ist es wichtig, den zukünftigen Strombedarf möglichst weit im voraus bestimmen zu können. Durch möglichst genaue Abschätzungen können die Versorgungsunternehmen wesentliche Einsparungen in verschiedenen Bereichen erzielen: bei der Vorhaltung von Betriebsreserven, der Verwaltung von Einplanungen oder auch bei der Verwaltung des Treibstoffbedarfes.

2.5 Clusteranalyse

Bei der Clusteranalyse handelt es sich um eine bedeutende Anwendung des Data Mining. Sie wird eingesetzt, um Daten oder Datensätze auf der Grundlage ihrer Merkmalen in Gruppen, so genannte Cluster, einzuteilen.

Daten, beziehungsweise Datensätze sind gesammelte Informationen, die auf Computersystemen gespeichert sind und verarbeitet werden können. Als Merkmale werden in diesem Rahmen die einzelnen Attribute (Datenfelder) der Daten beziehungsweise Datensätze bezeichnet.

Die Ordnungsprinzipien für eine Segmentierung der Daten durch die Clusteranalyse können dabei folgendermaßen charakterisiert werden:

- Die Homogenität innerhalb der Cluster soll möglichst hoch sein, dass heißt das die Ähnlichkeit der Daten, gemessen mit einem Distanz- oder Ähnlichkeitsmaß, innerhalb eines Clusters möglichst groß sein soll.
- Die Heterogenität zwischen den unterschiedlichen Clustern soll ebenfalls möglichst hoch sein, dass heißt das die Ähnlichkeit der Daten, die verschiedenen Clustern angehören, möglichst klein sein soll.

Klassische Anwendungsgebiete der Clusteranalyse sind die Kunden- oder Produktsegmentierung im Marketing, die Typisierung von Verhaltensweisen in der Psychologie und die Taxonomierung von Lebewesen in der Biologie. Generell können Clusteranalysen überall dort eingesetzt werden, wo man eine Datensammlung auf wenige überschaubare Einheiten reduzieren möchte[6].

Dieser Abschnitt soll nun näher in die Thematik der Clusteranalyse einführen. Der erste Abschnitt erläutert die allgemeine Vorgehensweise der Clusteranalyse. Anschließend wird ein kurzer Überblick über eine Auswahl der bekanntesten Techniken zur Clusterbildung gegeben.

2.5.1 Vorgehensweise

Die Clusteranalyse wird eingesetzt um Daten anhand ihrer Merkmale in Cluster einzuteilen. Die Gruppierung soll dabei so erfolgen, dass sich die Daten innerhalb eines Clusters möglichst ähnlich sind, während sich gleichzeitig die Cluster untereinander möglichst unähnlich sind.

Dabei muss vor der Durchführung einer Clusteranalyse eine Vorstellung darüber bestehen, in welchem Sinne die Ähnlichkeit der vorliegenden Daten sinnvoll definiert und gemessen werden kann.

Methodisch besteht jede Clusteranalyse aus drei Schritten:

- Ähnlichkeitsmaß festlegen
  In der Literatur werden eine Vielzahl geeigneter Ähnlichkeitsmaße vorgestellt. Ähnlichkeitsmaße lassen sich sowohl für metrische (numerische Merkmalsvektoren) als auch für ordinale und nominale Merkmale definierten.

  Für metrische Merkmale werden häufig Distanzfunktionen auf Grundlage geometrischer Abstrandsfunktionen verwendet. Dabei werden m Merkmale als Punkte im m-dimensionalen Raum interpretiert.

  Eine der bekanntesten Distanzfunktionen ist die *Minkowski-Metrik*. Sie ist definiert als:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[c]{\sum_{i=1}^{m} |x_i - y_i|^c}$$

wobei d die Distanz zwischen den beiden Punkten $\boldsymbol{x}$ und $\boldsymbol{y}$ im m-dimensionalen Raum ist und $c \in \mathcal{R}^+$. Die aus der Minkowski-Metrik abgeleiteten Metriken werden häufig als $\mathcal{L}_c$-Norm bezeichnet. Die zwei bekanntesten Metriken sind:

1. für $c = 1$ die $\mathcal{L}_1$-Norm, auch Manhattan- oder City-Block-Distanz:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

2. für $c = 2$ die $\mathcal{L}_2$-Norm, auch euklidische Distanz:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Weitere Ähnlichkeitsmaße werden in [5] vorgestellt.

- Distanzen bestimmen
  In diesem Schritt werden die Distanzen der Daten zueinander bestimmt. Bei Bedarf müssen Merkmale, die unterschiedlichen Wertebereichen entstammen oder mit unterschiedlichen Maßstäben gemessen worden sind, normiert werden. Falls beispielsweise die zu vergleichenden Merkmale die Ausmaße von Objekten darstellen und einige Merkmale in km und andere Merkmale in cm angegeben werden, müssen diese auf eine einheitliche Maßeinheit umgerechnet werden.

- Cluster bilden
  Auf Grundlage dieser Distanzen soll nun die Bildung der Cluster vorgenommen werden. Dieser Schritt soll eine Zuordnung der einzelnen Daten zu in sich möglichst homogenen Cluster liefert. Um Cluster zu bilden, gibt es zahlreiche Algorithmen. Im folgenden Abschnitt 2.5.2 wird ein Überblick über die Techniken zur Clusterbildung gegeben.

2.5.2 Techniken zur Clusterbildung

Die Techniken zur Clusterbildung können in zwei Kategorien unterteilt werden:

- **Partitionierende Techniken**
  Partitionierende Clusterverfahren bauen auf einer gegebenen Clustereinteilung auf und tauschen so lange Daten zwischen den Clustern aus, bis eine optimale Einteilung gegeben ist.

- **Hierarchische Techniken**
  Bei Beginn dieses Verfahrens ist die Clusteranzahl und -einteilung noch nicht bekannt, sondern entwickelt sich erst Schritt für Schritt. Grundsätzlich lassen sich hier zwei Verfahren unterscheiden:

  – Teilende Techniken
    Ausgangspunkt dieses Verfahrens ist ein größtmögliches Cluster, welches alle Daten enthält. Dieses wird dann sukzessive in kleinere Cluster aufgeteilt.

  – Anhäufende Techniken
    Bei diesem Verfahren wird mit den kleinstmöglichen Clustern begonnen, nämlich mit den einzelnen Daten. So stellt jedes Datenobjekt ein Cluster dar. Schritt für Schritt werden jetzt die jeweils einander ähnlichsten Cluster zu größeren Clustern zusammengefasst.

  Hierarchische Techniken können mitunter beendet werden, wenn alle Cluster eine bestimmte Distanz zueinander überschreiten oder wenn eine genügende Anzahl von Clustern ermittelt worden ist.

Für die Clusterbildung werden statistische Methoden, wie auch künstliche neuronale Netze in Form von Selbstorganisierenden Merkmalskarten (engl. Self Organizing Maps (SOM)) verwendet. Selbstorganisierenden Merkmalskarten stellen ein partitionierendes Clusterverfahren dar. Auch sie berechnen eine Partitionierung der Daten auf der Grundlage eines Ähnlichkeitsmaßes. Zusätzlich wird bei dieser Technik jedoch noch eine topographische Nachbarschaftsstruktur zwischen den Repräsentanten angenommen. Im folgenden Kapitel wird genauer auf die SOM eingegangen.

2.6 Selbstorganisierende Merkmalskarten (SOM)

Zu den etablierten Vertretern der Clusteranlaysetechniken gehören die Selbstorganisierenden Merkmalskarten. Dieses Verfahren wurde 1982 von Professor Teuvo Kohonen an der Universität Helsinki entwickelt, weshalb es auch unter der Bezeichnung Kohonen-Karten bekannt ist.

Die SOM gehören zur Klasse der Künstlichen Neuronalen Netze[2] (KNN) und damit zu den Lernverfahren.

KNN sind informationsverarbeitende Systeme, welche auf sehr abstrakte Weise die Neuronengeflechte, wie sie in menschlichen und tierischen Gehirnen vorkommen, imitieren und so in der Lage sind, ein klassifizierendes Verhalten zu erlernen. SOM können verwendet werden, um beliebige Eingabedaten automatisch zu klassifizieren. Dies steht im Gegensatz zu vielen anderen neuronalen Netzen, denen man, damit sie lernen, mitteilen muss, was richtig und was falsch ist.

In den folgenden Abschnitten soll das Verfahren der SOM als ein Vertreter der KNN näher erläutern werden. Dabei wird zunächst auf den biologischen Hintergrund der KNN und die Besonderheiten der SOM eingegangen. Danach wird der Aufbau und die Arbeitsweise der SOM erklärt.

2.6.1 Das biologische Vorbild

Biologisches Vorbild der KNN sind die Neuronengeflechte, wie sie in menschlichen und tierischen Gehirnen vorkommen. KNN versuchen die Struktur und Leistungsmerkmale des Gehirns nachzubilden. In diesem Abschnitt wird kurz auf den Aufbau des Gehirnes eingegangen, bevor eine Verbindung vom Gehirn zu den KNN gezogen wird. Im Anschluss werden die Besonderheiten der SOM in Hinblick auf die Gehirnforschung erläutert.

Das Gehirn besitzt von Geburt an eine sehr komplexe Struktur. Es besteht aus Milliarden Neuronen. Biologische Neuronen sind Nervenzellen, welche Signale empfangen und weiterleiten können. Sie haben einen Zellkörper, von dem mehrere Dendriten ausgehen. Dies sind sozusagen die Eingangsleitungen des Neurons. An den Dendriten empfängt die Nervenzelle Signale von anderen Nervenzellen. Außerdem geht vom Zellkörper aus genau ein Axon ab, über welches das Neuron

[2] Der Begriff Künstliche Neuronale Netze stammt aus der Neuroinformatik. Die Neuroinformatik sucht nach Methoden und Anwendungen, um neuronale biologische Informationssysteme auf technische Informationssysteme abzubilden.

in der Lage ist, Signale zu senden. Ein Axon stellt somit die Ausgangsleitung eines Neurons dar. Das Axon mündet in mehrere Synapsen (Verbindungen zwischen den Neuronen), über welche die Neuronen Informationen in Form von Impulsen austauschen können.

Ein Lebewesen lernt, indem es Erfahrungen sammelt. Durch das Sammeln von Erfahrungen verändert sich die Struktur des Gehirnes: Der Neurophysiologe Donald Hebb fand heraus, das sich die Übertragungseigenschaften der Synapsen zwischen einzelnen Nervenzellen abhängig von der Dauer und der Häufigkeit des einlaufenden Signals verändern:

> *Sofern ein Axon der Zelle A einer Zelle B nahe genug ist, um sie immer wieder zu erregen bzw. dafür zu sorgen, das sie feuert, findet ein Wachstumsprozess oder eine metabolische Veränderung in einer der beiden Zellen oder in beiden statt, so dass die Effektivität der Zelle A, die Zelle B zu erregen, gesteigert wird* [8].

Lernvorgänge zeigen sich im Gehirn also dadurch, dass Synapsen zwischen einzelnen Nervenzellen ihre Übertragungseigenschaften verbessern, dass heißt, die Verbindung verstärken.

Ein KNN besteht, wie sein natürliches Vorbild, aus einer Vielzahl einfacher Recheneinheiten, den Neuronen, die untereinander verbunden sind und miteinander kommunizieren können. Die Arbeitsweise eines KNN orientiert sich an der Arbeitsweise eines biologisches Neuronennetzes: es lernt aus Erfahrungen. Dafür durchläuft es einen Lernprozess. In diesem Prozess lernen die Neuronen, indem sie ihre Verbindungen mit Gewichten versehen. Durch einen Lernalgorithmus werden die Gewichte modifiziert. Der allgemeine Lernprozess der SOM wird in Abschitt 2.6 genauer erläutert.

Besonderheiten der SOM

Eine wichtige Eigenschaft der SOM, welche aus der Gehirnforschung motiviert wurde, ist die Selbstorganisation. Das Gehirn ist in der Lage selbstständig Muster und Regelmäßigkeiten zu erkennen und neue Eindrücke bereits bekannten Mustern zuzuordnen. Auf dieselbe Weise sind auch SOM in der Lage Muster in den Eingaben automatisch zu identifizieren und neue Eingaben diesen Mustern zuzuordnen.

Eine andere wichtige Eigenschaft der SOM ist die Topologieerhaltung[3]. Das bedeutet, dass ähnliche Eingangssignale auf benachbarte Regionen auf der SOM abgebildet werden. Auch diese Eigenschaft wurde aus der Gehirnforschung motiviert: Das Gehirn ist topographisch geordnet, dass heißt,

[3] Topologieerhaltend meint, dass die Nachbarschaftsverhältnisse erhalten bleiben.

bestimmten sensorischen Signalen sind bestimmte räumlich voneinander abgegrenzte Bereiche des Gehirns zugeordnet.

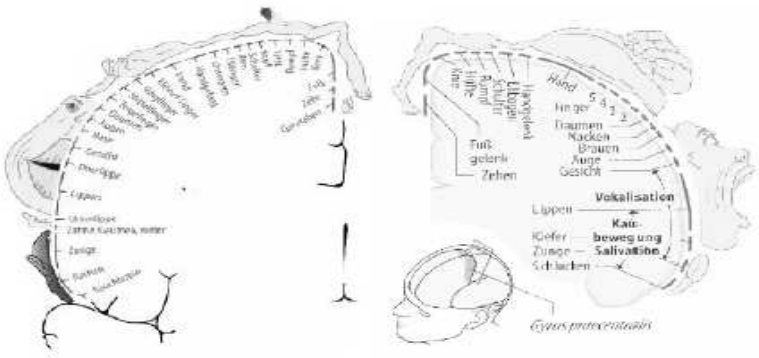


Bild 2.2: Topologieerhaltung des Gehirnes [10]

In Abbildung 2.2 ist auf der linken Seite des Gehirnes die somatosensorische Hirnrinde - zuständig für den Tastsinn - und auf der rechten Seite die motorische Hirnrinde zu erkennen. Die einzelnen Körperteile sind der jeweils zuständigen Hirnregion zugeordnet. Dabei kontrollieren benachbarte Hirnregionen auch benachbarte, beziehungsweise ähnliche Körperteile. Zum Beispiel liegt der Bereich für den Tastsinn der Arme direkt neben dem Bereich für die Hände. Wie das menschliche Gehirn die Sinneseindrücke wirklich verarbeitet und speichert, ist noch nicht genügend erforscht. Die SOM wurden unter anderem auch dazu entwickelt, mehr Einblick in die Organisationsweise unseres Gehirns zu geben.

2.6.2 Aufbau und Arbeitsweise der SOM

Wie bereits erwähnt, orientieren sich SOM in ihrem Aufbau und ihrer Arbeitsweise an ihrem natürlichen Vorbild, dem Gehirn von Lebewesen. Analog zu diesem Vorbild setzen sich SOM aus vielen einfachen Bausteinen, den Neuronen, zusammen, welche miteinander in Verbindung stehen. Wie Abbildung 2.3 zeigt, besteht eine SOM aus zwei Schichten: der Eingabeschicht und der Kartenschicht - auch Kohonenschicht genannt - welche die eigentliche selbstorganisierende Merkmalskarte darstellt.

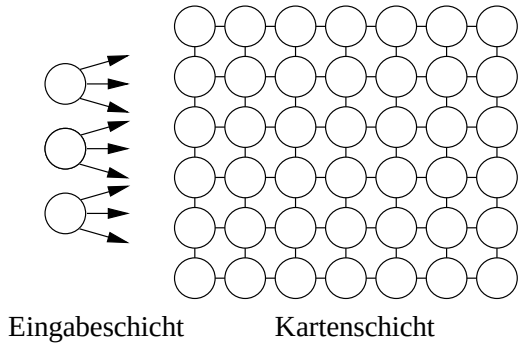


Bild 2.3: Schichten einer SOM

Die Neuronen der Eingabeschicht sind über Verbindungen, die jeweils einem Gewichtungsvektor entsprechen, mit den Neuronen der Kartenschicht voll vernetzt. Das heißt, jedes Eingabeneuron hat zu jedem Neuron der Kartenschicht eine Verbindung. Auch jedes Neuron der Kartenschicht steht mit seinen Nachbarn in Verbindung [12]. Dadurch erhält die Kartenschicht eine gitterförmige Struktur [4], in welcher jedes Neuron eindeutig durch seine Koordinaten bestimmt ist. Über die Verbindungen untereinander sind die Neuronen in der Lage, Informationen auszutauschen.

Die Arbeitsweise der SOM basiert darauf, verschiedene Eingabesignale zu klassifizieren. Im folgenden Abschnitt wird erläutert, wie es der SOM gelingt, ähnliche Eingangssignale auch als ähnlich zu erkennen.

2.6.3 Der SOM-Algorithmus

Zu Beginn eines jeden SOM-Algorithmus wird jedem Neuron der Eingabeschicht ein externes Eingabesignal zugewiesen. Werden alle Eingabesignale zu einem Zeitpunkt zusammengefasst, ergeben sie einen n-dimensionalen Eingabevektor:

$$x = (x_1, x_2, \ldots, x_n) \qquad n = \text{Anzahl der Eingabeneuronen}$$

[4] Für die Kartenschicht wird meist ein zweidimensionales Gitter gewählt, weil es sich sehr gut visualisieren lässt. Aber auch n-dimensionale Gitter sind möglich und werden in [11] behandelt.

Die Eingabevektoren können Bilder, Audiodaten wie Phoneme, Daten von künstlichen Sensoren (Druck, Temperatur, Helligkeit,...), geographische Positionen, oder auch volkswirtschaftliche Kennzahlen eines Landes sein.

Das Ziel des SOM-Algorithmus ist es nun diese hoch-dimensionalen Eingabevektoren so auf die niedrigdimensionale (meist zweidimensionale) Kartenschicht zu projizieren, dass die topologischen Beziehungen zwischen den Eingabevektoren erhalten bleiben. Es gibt verschiedene SOM-Algorithmen, an dieser Stelle soll nur die Grundversion dargestellt werden:

1. Initialisierung
   Zu Beginn des Algorithmus werden die Neuronen der Kartenschicht platziert und jedem einzelnen Neuron wird ein Referenzvektor m_i zugewiesen. Um m_i mit einem Wert zu belegen, können verschiedene *Initialisierungsfunktionen* verwendet werden. Ein häufig verwendete Funktion ist die Initialisierung durch Zufallsvektoren.

   Um die Neuronen zu platzieren wird häufig ein zweidimensionales rechteckiges oder hexagonales Raster verwendet.

2. Training - Siegerneuron festlegen
   Im nächsten Schritt wird ein Eingabevektor $\boldsymbol{x}$ über die Eingabeschicht herangezogen und mit allen Referenzvektoren der Neuronen der Kartenschicht verglichen. Das Neuron, dessen Referenzvektor $\boldsymbol{m_i}$ dem Eingabevektor $\boldsymbol{x}$ am ähnlichsten ist, wird als Siegerneuron c festgelegt. Für die Berechnung der Ähnlichkeit gibt es verschiedene Funktionen. Wie in Abschnitt 2.5.1 bereits erwähnt, sind die von der Minkowski-Metrik abgeleitete euklidische Distanz oder Manhattan-Distanz gebräuchliche Distanzfunktionen. In den verschiedenen Variationen des SOM-Algorithmus werden aber auch andere Ähnlichkeitsmaße verwendet.

3. Training - Nachbarschaft bestimmen
   Um die Topologieerhaltung zu gewährleisten, müssen auch die Neuronen der Nachbarschaft des Siegerneurons in den Lernprozess mit einbezogen werden. Mit Hilfe einer *Distanzfunktion* kann für jedes Siegerneuron sein Einflussbereich festgelegt werden. Abbildung 2.4 zeigt den möglichen Einflussbereich eines Siegereurons.

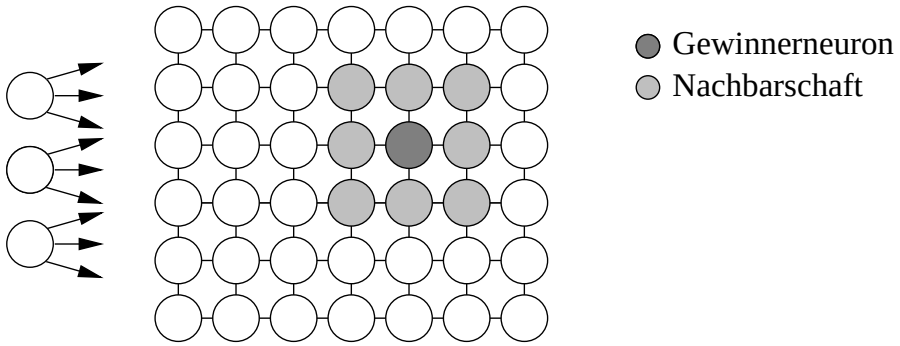


Bild 2.4: Gewinnerneuron mit seiner Nachbarschaft

Allerdings soll im Laufe des Trainings nach jedem Trainingsschritt der Einflussbereich des Siegerneurons abnehmen. Dies ist sinnvoll, da sich die Karte auf diese Weise besser entfalten kann und schneller konvergiert. Es wird also eine monoton fallende Distanzfunktion benötigt. Die Nachbarschaft ist dabei zu Beginn des Trainings größer und wird bei jedem Trainingsschritt verkleinert. Ein Beispiel einer Distanzfunktion ist die Lineare Distanzfunktion:

$$d_{lin}(t) = \begin{cases} c(1 - \frac{t}{t_{end}}) & f\ddot{u}r \;\; t < t_{end} \\ 0 \quad sonst \end{cases}$$

Wobei d den Einflussbereich des Siegerneurons c zur aktuellen Epoche t darstellt. t_{end} bezeichnet die Epoche, bei der das Training abgebrochen werden soll.

Weitere Beispiele werden in [17] vorgestellt.

4. Training - Nachbarschaft anpassen
   Jetzt werden die Referenzvektoren aller Neuronen, die sich in der räumlichen Nachbarschaft des Siegerneurons befinden (siehe Abbildung 2.4), so angepasst, dass sie dem Eingabevektor x ähnlicher werden.

   Dabei kann jedem Neuron der Nachbarschaft, abhängig von seiner Entfernung von dem Siegerneuron, der Grad der Veränderung zugeordnet werden.

   Für die Berechnung des Grades können verschiedene Funktionen verwendet werden. Diese Funktionen werden auch als *Nachbarschaftsfunktionen* bezeichnet.

Die simpelste Nachbarschaftsfunktion ist die Rechtecksfunktion:

$$f_{rechteck}(r,d) = \begin{cases} grad = 1 & für\ \ r < d \\ grad = 0 & sonst \end{cases}$$

Wobei d den maximalen Abstand der Neuronen der Nachbarschaft und r den Abstand des aktuellen Neurons bezeichnen. $grad$ gibt den Grad an, mit welchem die Neuronen lernen. Bei Anwendung der Rechtecksfunktion erhalten alle Neuronen im Einflussbereich des Siegerneurons den Nachbarschaftsgrad 1 – sie lernen also wie das Siegerneuron – alle anderen Neuronen erfahren keine Veränderung ihrer Gewichte.

Eine weitere Nachbarschaftsfunktion ist Gaussfunktion:

$$f_{gauss}(r,d) = e^{-(\frac{r}{d})}$$

Abbildung 2.5 zeigt den Nachbarschaftsgrad der Gaussfunktion, wobei das Siegerneuron im Ursprung des Koordinatensystemes liegt.

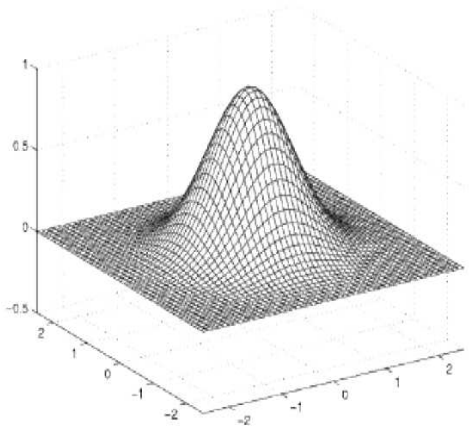


Bild 2.5: Nachbarschaftsgrad der Gaussfunktion [17]

5. Training - Iteration
   Schritt 2 und Schritt 3 werden für weitere Eingabevektoren wiederholt. Dieses geschieht

solange, bis ein bestimmtes Abbruchkriterium (zum Beispiel Anzahl der Trainingsepochen) erreicht wurde.

Ziel des SOM-Algorithmus ist es, jedes Neuron der Kartenschicht auf einen bestimmten Bereich des Eingaberaumes zu spezialisieren.

2.7 Visualisierung

Die Ergebnisse der oben beschriebenen Analysemethoden werden im Allgemeinen nicht in textueller oder numerischer Form, sondern mit Hilfe von Visualisierungsapplikationen dargestellt. Dies ist sinnvoll, da eine visuelle Darstellung beim Verständnis der Data Mining-Ergebnisse hilft und eine manuelle Analyse und Clusterung der Daten durch das menschliche Auge erlaubt. Daher hat sich in den letzten Jahren, aufbauend auf der Wissensextraktion durch automatische Data Mining-Methoden, ein neues Forschungsgebiet etabliert: Das Visuelle Data Mining (VDM) [14].

Die Grundidee des VDM ist die geeignete Darstellung der Daten in visueller Form, welche dem Benutzer ermöglicht, einen Überblick über die Struktur der Daten zu bekommen, Schlussfolgerungen zu ziehen, sowie direkt mit den Daten zu interagieren [13]. VDM bezeichnet also die Verknüpfung automatischer Analysemethoden mit Methoden zur visuellen Darstellung der Resultate. Diese Verknüpfung ist sinnvoll, da die klassischen, computergestützten Analysemethoden oft eine Art „Black Box“ darstellen, welche nur begrenzte oder gar keine Einflussnahme durch den Benutzer erlauben [14]. Bei vielen Problemstellungen des Data Mining können erst dann gute Ergebnisse gewährleistet werden, wenn die Exploration der Daten nicht nur automatisch durch die klassischen Data Mining-Methoden stattfindet, sondern auch die menschliche Intuition, Flexibilität, Kreativität und Fachwissen mit einfließen.

Das Ziel des VDM ist es, den Benutzer, sowie dessen visuelle Wahrnehmungsfähigkeit mit Hilfe von Visualisierungstechnologien in den Datenanalyseprozess zu integrieren: Zuerst soll der Benutzer durch die Visualisierung der Resultate einer automatischen Datenanalyse einen ersten Überblick über die Zusammenhänge in den Daten erhalten. Danach sollte es ihm möglich sein, diese Resultate anzupassen. Durch die Kombination dieser zwei Arbeitsschritte können sich viele Vorteile ergeben, einige wären:

- Die Resultate der automatischen Analyse können manuell berichtigt oder erweitert werden. Daraus folgt eine höhere Qualität der Resultate.

- Durch die Kombination des existierenden Expertenwissens mit den Speicherkapatizäten und Rechenleistungen moderner Computersysteme kann eine effektive Nutzung der Ressourcen gewährleistet werden.
- Durch die aktive Einbeziehung des Benutzers in den Analyseprozess wird sein Verständnis für die resultierenden Muster erhöht. Dies steigert das Vertrauen in die Resultate.

In diesem Abschnitt werden zuerst kurz die zu visualisierenden Daten beschrieben. Da das Ziel dieser Arbeit die dreidimensionale Visualisierung von SOM ist, werden im Anschluss daran verschiedene Visualisierungsmöglichkeiten der SOM vorgestellt und diskutiert.

2.7.1 Zu visualisierende Daten

Die Daten, welche einer Analyse und Visualisierung unterzogen werden sollen, besitzen häufig eine große Anzahl an Datensätzen. Jeder Datensatz entspricht dabei einer Beobachtung, wie zum Beispiel einer Messung bei einem physikalischen Experiment oder einer Transaktion in einem E-Commerce-System, und besitzen eine feste Anzahl von Attributen, auch Dimensionen genannt. Dabei kann die Anzahl der Attribute mitunter stark variieren – von einigen wenigen bis hin zu Tausenden von Attributen. Aus diesem Grund spricht man hier von mehrdimensionalen Daten [14] Mehrdimensionale Daten sind zum Beispiel Tabellen in relationalen Datenbanken, die häufig mehrere hundert oder tausend Attribute besitzen können.

2.7.2 Visualisierung der SOM

Um eine vom Menschen erfassbare Visualisierung zu erhalten, können verschiedene Topologien/Strukturen verwendet werden. Häufig werden zweidimensionale Gitter benutzt. Dabei können die Neuronen in einem Quad-Gitter oder Hex-Gitter angeordnet werden. Um Randeffekte zu vermeiden werden häufig auch randlose Gitter, bei welchen der obere mit dem unteren und der linke mit dem rechten Rand verbunden sind, verwendet. Randeffekte könne Gruppen auftauchen, bei denen die Daten nur auf den Kartenrand projiziert werden und die Kartenmitte größtenteils leer bleibt [21]. Abbildung 2.6 zeigt die verschiedenen Topologien.

Das Ergebnis einer SOM-Anwendung ist ein niedrig dimensionales Gitter (die Merkmalskarte), auf welchem die hochdimensionalen Datenvektoren (die Eingabevektoren) abgebildet sind. Dieses Gitter muss durch eine Visualisierung geeignet dargestellt werden.

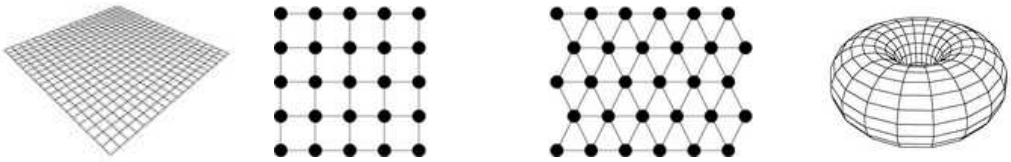


Bild 2.6: planare Topologie mit Rand, Quad-Gitter, Hex-Gitter, toroide randlose Topologie

In den meisten Anwendungen werden die SOM zweidimensional dargestellt. Die Strukturierung wird dabei häufig durch die Farbgebung veranschaulicht. Im Folgenden werden nun zwei Programme vorgestellt, die nach der Erzeugung einer SOM diese zweidimensional darstellen.

- **WEBSOM**

  WEBSOM wurde unter Professor Teuvo Kohonen vom *National Networks Research Centre* an der Technischen Universität in Helsinki entwickelt.

  Mithilfe des SOM-Algorithmus erzeugt WEBSOM eine zweidimensionale Abbildung einer Sammlung von Textdokumenten, zum Beispiel die Einträge einer Internet-Newsgroup. Die Struktur der Sammlung wird durch die Farbgebung dargestellt: Je heller eine Region auf der Abbildung erscheint, desto höher ist die Dokumentendichte. WEBSOM gibt dem Benutzer die Möglichkeit durch klicken auf eine Region in die Region hineinzuzoomen. Dazu stellt WEBSOM eine neue Abbildung mit der detaillierteren Ansicht der Region dar. [19]

  In Abbildung 2.7 ist die Visualisierung der Analyse mittels WEBSOM von über Millionen Dokumenten aus über 80 Internet-Newsgroups dargestellt.

- **ESOM** Die ESOM Tools wurden in der Databionics Forschungsgruppe der Universität in Marburg in der Programmiersprache Java entwickelt.

  Auch ESOM führt eine Datenanalyse mithilfe des SOM-Algorithmus durch und visualisiert die Resultate zweidimensional. Die Strukturierung der Daten wir ebenfalls durch die Farbgebung dargestellt. ESOM liefert eine Vielzahl von Funktionen und Interaktionsmöglichkeiten, welche unter [21] nachzulesen sind.

  Abbildung 2.8 zeigt ein Bildschirmfoto von ESOM mit einer von ESOM erzeugten und visualisierten SOM.

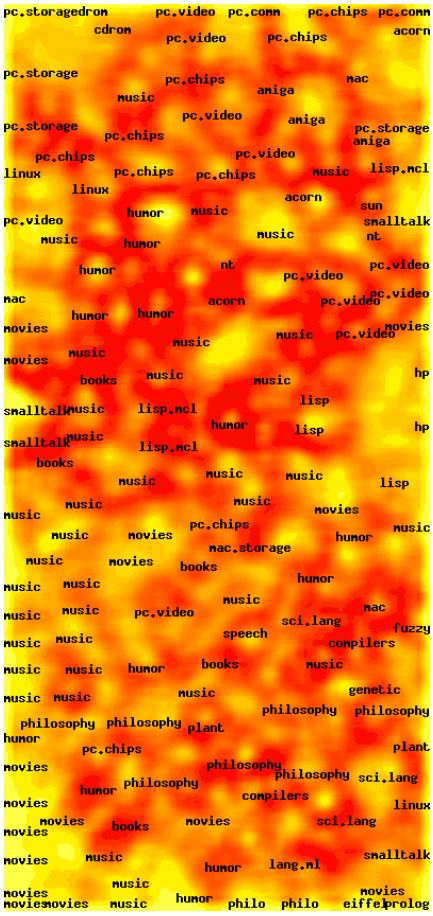


Bild 2.7: WEBSOM map – „Million documents“[19]

WEBSOM und ESOM stellen nur zwei ausgewählte Beispiele einer Vielzahl von Anwendungen dar, welche SOM zweidimensional visualisieren können. Es existieren aber auch Anwendungen, welche eine dreidimensionale Sicht auf die SOM erlauben. Dabei wird Strukturierung der Datenanalyseresultate durch ein Höhenprofil (ähnlich einer Landschaft) dargestellt.

Ziel dieser Diplomarbeit ist es, unter Verwendung der in ESOM bereits vorhandenen Implementationen zur Analyse von Datensätzen, ein Programm zu entwickeln, welches die Resultate der Analyse dreidimensional darstellen kann. Im Folgenden wird die zweidimensionale Ansicht mit der dreidimensionalen Ansicht gegenübergestellt.

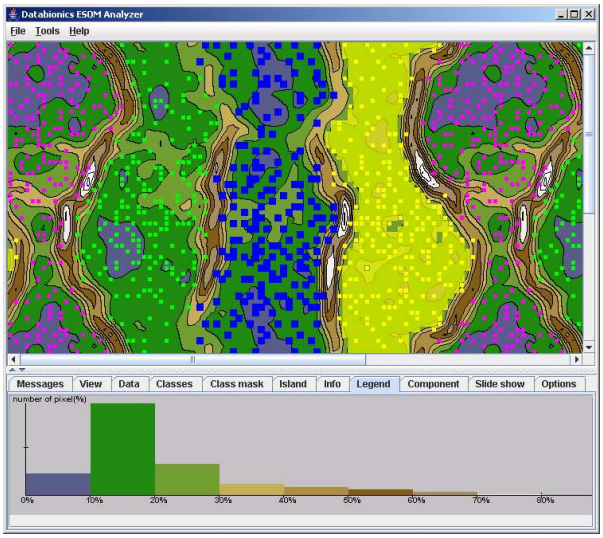


Bild 2.8: Bildschirmfoto des Programmes ESOM [21]

Zweidimensionale versus Dreidimensionale Visualisierung

WEBSOM und ESOM sind zwei Beispielanwendungen, welche in der Lage sind, die Resultate einer Datenanalyse mittels SOM zweidimensional zu visualisieren.

Die Strukturierung der Resultate wird dabei durch eine passende Farbgebung dargestellt: Bei WEBSOM werden hohe Dokumentendichten durch hellere Farben hervorgehoben. Bei ESOM hat der Nutzer die Möglichkeit die Farbverteilung selbst zu wählen.

Bei einer dreidimensionalen Ansicht wird die Strukturierung der Datenanalyseresultate dagegen durch ein Höhenprofil (ähnlich einer Landschaft) dargestellt. Durch eine zusätzliche Farbgebung kann die Darstellung optimiert werden und sogar eine abstrakte Landschaft simulieren. Die dreidimensionale Sicht liefert also eine zusätzliche Komponente zur Visualisierung der Struktur: die Darstellung der Strukturierung durch die Farbgebung wird durch das Höhenprofil noch unterstützt.

Bei der dreidimensionalen Visualisierung gibt es allerdings verschiedene Probleme, welche das Verständnis und die Anschaulichkeit der SOM negativ beeinflussen könnten:

- Die entstehenden Höhen („Berge“) könnten die dahinter liegenden Daten verdecken. Durch verschiedene Interaktionsmöglichkeiten des Benutzers, wie eine Rotation der Darstellung, könnte diesem Problem Abhilfe geschaffen werden.

- Daten könnten mitunter sehr große Differenzen zueinander haben, so dass der Überblick über die Karte durch zueinander stark differenzierende Höhen und Tiefen beeinträchtigt wird. Durch eine Normierung der Daten auf einen bestimmten Wertebereich kann dieses Problem gelöst werden.

- Dreidimensionale Visualisierungen sind auf dem Bildschirm nicht immer leicht navigierbar. Die Bewegungen im Navigationsraum müssen flüssig sein. Ruckelt es zu sehr, so muss sich der Nutzer nach jeder Bewegung neu orientieren. Es besteht die Gefahr der Orientierungslosigkeit.

- Zweidimensionale Ansichten haben den Vorteil, dass sie leicht zu überblicken sind. Mit Bewegungen in horizontaler und vertikaler Richtung kann schon ausreichend navigiert werden. Die Navigation kann durch einfache Bildlaufleisten umgesetzt werden. Die Kontrolle einer dreidimensionalen Ansicht dagegen erweist sich, durch die zusätzliche Dimension, als etwas komplizierter und ist mit einem höheren Aufwand zur Einarbeitung verbunden.

Seit Anfang der 1990er Jahre ist im Rahmen der computergestützten Visualisierung die Diskussion entbrannt, welche Dimensionalität als Darstellungsform besser geeignet ist [22]. Mehrere praktische Untersuchungen auf diesem Gebiet haben allerdings gezeigt, dass der Erfahrungsvorteil aus der natürlichen dreidimensionalen Umgebung des Menschen durch die Erschließung der dritten Dimension individuell von den räumlichen Fähigkeiten der Nutzer abhängt [22].

Doch gleichgültig ob für die Visualisierung die zwei- oder die dreidimensionale Ansicht gewählt wird, ein Problem stellt sich trotzdem:

Die reine Darstellung der Positionen der Vektoren auf der SOM reicht zur Erkennung von Strukturen oft nicht aus: Die Distanzen zwischen den Vektoren werden auf der Karte verzerrt dargestellt. So können Datenvektoren, welche auf der Karten direkt benachbart sind, trotzdem eine sehr große Distanz zueinander haben [23]. Für den Betrachter ist aber gerade dieses Erkennen der Distanzen sehr wichtig. Für eine gute und intuitive Visualisierung der Strukturen in einem Datensatz reicht es aus diesem Grund nicht aus, nur die Positionen der Vektoren auf der SOM wiederzugeben [23], es müssen andere Methoden hinzugezogen werden.

In [21] werden verschiedene Visualisierungsmethoden vorgestellt. Im Folgenden soll auf zwei dieser Methoden genauer eingegangen werden:

U-Matrix

Die U-Matrix wurde von Professor Alfred Ultsch in der Forschungsgruppe Datenbionik an der Philipps-Universität in Marburg entwickelt. Sie stellt die lokalen Distanzbeziehungen zwischen den Daten als dreidimensionale Landschaft dar. Die Struktur dieser Landschaft bietet einen Überblick über die Distanzbeziehungen der Daten: Berge verdeutlichen große Unterschiede zwischen den Daten, während Daten, welche in einem gemeinsamen Tal liegen, sich sehr ähnlich sind[23].

Die Höhe eines Datenpunktes ergibt sich aus der Summe der Distanzen zu allen direkten Nachbarn, normiert durch die maximal auftretende Höhe:

$$h_j = \frac{\sum_{i=0}^{n} h_i}{max\_h}$$

Dabei beschreibt $i \in \{0..n\}$ die direkten Nachbarn des aktuellen Neurons j und max_h die maximale Höhe auf der gesamten Karte.

Abbildung 2.9 stellt die U-Matrix-Ansicht des Datensatzes *hexa.lrn* dar. Es sind gut sechs klar voneinander abgegrenzte Datencluster zu erkennen. Der Datensatz wurde mithilfe des Programmes *ESOM* erzeugt und visualisiert.

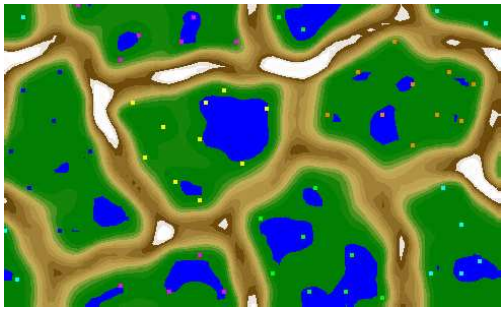


Bild 2.9: U-Matrix von hexa.lrn

Komponentenkarten

Bei der Ansicht der Komponentenkarten wird die Verteilung der Daten für jedes Attribut (Komponente) des Datensatzes einzeln dargestellt. Abbildung 2.10 zeigt die drei Komponentenkarten des Beispieldatensatzes *hexa.lrn*. Auch diese Abbildung wurde mit *ESOM* erzeugt und visualisiert.

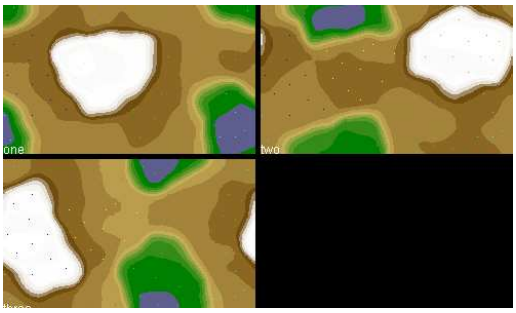


Bild 2.10: Komponentenkarten von hexa.lrn

Kapitel 3

Eigener Ansatz

Nach einer kurzen Systembeschreibung wird in diesem Kapitel eine strukturierte Anforderungsliste erstellt. Im Anschluss daran werden die genutzten Technologien vorgestellt. Danach wird auf die Realisierung der Anforderungen eingegangen. Zuletzt wird in diesem Kapitel eine umfassende Übersicht über die Systemarchitektur gegeben.

3.1 Systembeschreibung

Ziel dieser Diplomarbeit ist die Entwicklung eines Programmes, welches einen gegebenen Datensatz nach dem System der SOM automatisch klassifiziert und die Ergebnisse dieser Datenanalyse dreidimensional visualisiert. Desweiteren soll das Programm dem Benutzer verschiedene Möglichkeiten zur Interaktion bieten, so dass dieser sich einen guten Überblick über die erzeugte SOM machen kann.

Das zu entwickelnde Programm soll den Namen *SomVis3D* tragen. Dieser leitet sich aus der Hauptaufgabe des Programmes, der dreidimensionalen Visualisierung von SOM ab: „SomVis3D“ setzt sich zusammen aus „Som“ für Selbstorganisierende Merkmalskarten, „Vis“ für Visualisierung und „3D“ für die Dreidimensionalität.

3.2 Systemanforderungen

Ziel dieser Diplomarbeit ist die Erstellung eines Systems, welches die folgenden Anforderungen erfüllt:

1. **Funktionale Anforderungen**

   (a) Das Programm muss verschiedene Visualisierungsmöglichkeiten der SOM bereitstellen.

      i. Das Programm muss die SOMs dreidimensional darstellen können.

      ii. Das Programm muss die Darstellung der SOMs in U-Matrix ermöglichen.

      iii. Das Programm muss die Ansicht der Komponentenkarten ermöglichen.

   (b) Das Programm sollte unter Verwendung der Databionics ESOM Tools (siehe Abschnitt 2.7.2) komplexere Funktionen ermöglichen.

      i. Durch das ESOM-Training muss das Trainieren von Learn-Dateien möglich sein.

      ii. Durch die ESOM-Projection-Funktion sollte eine übergebene Learn-Datei auf eine bereits bestehende SOM projiziert werden können.

   (c) Das Programm muss mittels dynamisch verknüpfter Displays eine Vielzahl unterschiedlicher Informationen anzeigen können.

      i. Verschiedene Displays sollten unterschiedliche Visualisierungen (U-Matrix, Komponentenkarten) der SOM darstellen.

      ii. Informationen geladener Dateien sollten abrufbar sein.

      iii. Die Displays sollten miteinander kommunizieren.

      iv. Die Verwaltung der Displays muss effizient sein in Bezug auf Übersichtlichkeit.

   (d) Das Programm sollte in der Lage sein, verschiedene Dateitypen[1] zu verwalten.

      i. Verschiedene Dateitypen sollten geladen werden können.

      ii. Die geladenen Dateitypen müssen ihrem Sinn entsprechend verarbeitet werden können.

      iii. Verschiedene Dateitypen sollten gespeichert werden können.

[1] Datentypen, welche im Rahmen des ESOM-Projektes (siehe *http://databionic-esom.sourceforge.net*) zur Speicherung verschiedener SOM-Informationen verwendet werden.

(e) Durch die Spezifikation einer Schnittstelle sollte die Kommunikation mit externen Programmen möglich sein.

2. **Anforderungen an die Benutzerschnittstelle**

   (a) Die grafische Benutzerschnittstelle sollte intuitiv aufgebaut sein.

   (b) Der Benutzer sollte die Möglichkeit haben, sich die Arbeitsfläche möglichst individuell einzurichten.

   (c) Das Programm muss verschiedene Interaktionsmöglichkeiten für den Benutzer zur Verfügung stellen.

      i. Die Möglichkeit zur Rotation und Zoom der Darstellungen muss gegeben sein.

      ii. Das Programm sollte die interaktive Identifikation von Clustern ermöglichen.

      iii. Der Benutzer sollte die Möglichkeit haben, die selektierten Cluster in einer Datei zu speichern.

      iv. Der Benutzer sollte die Möglichkeit zur Veränderung des Erscheinungsbildes der SOM-Visualisierung haben.

         A. Die Einstellung alternativer Farbschemata sollte möglich sein.

         B. Die Bestmatches müssen in Größe und Farbe veränderbar sein.

3. **Begleitende Dokumente**

   (a) Eine Ausarbeitung sollte Einblick in die Umsetzung des Programmes geben.

   (b) Das Programm muss über eine Hilfsfunktion verfügen.

   (c) Es muss eine Installationsanleitung zur Verfügung stehen.

4. **Anforderungen an den Entwicklungsprozess**

   (a) Über den Entwicklungsprozess hinweg sollten alle 2 Wochen Statusberichte über den Entwicklungsstand per E-Mail an den Auftraggeber gesendet werden.

   (b) Der Entwicklungsprozess sollte den Zeitraum von 6 Monaten nicht überschreiten.

3.3 Technologien

Der erste Schritt in der Bearbeitung der oben genannten Anforderungen besteht in der Wahl einer geeigneten Programmiersprache und einer Grafikbibliothek. Im Folgenden werden die gewählten Werkzeuge kurz vorgestellt.

3.3.1 Die Programmiersprache

Heutzutage werden Programme vor allem objektorieniert entwickelt. Zu den bedeutendsten objektorientierten Sprachen zählen Java und C++.

- **C++**
  1989 wurde mit der Entwicklung von C++ von Bjarne Stroustrup bei den Bell Labs begonnen. C++ wurde aus der Programmiersprache C heraus entwickelt und enthält, bis auf einige Ausnahmen, C als Teilmenge. Den anderen Haupteinfluss auf C++ hatte die Sprache SIMULA67, welche das Klassenkonzept unterstützt. C++ ermöglicht eine schnelle und hardwarenahe Implementierung und stellt eine weit verbreitete und industriell bedeutende Programmiersprache dar [25].

- **Java**
  Java wurde von der Firma Sun Microsystems entwickelt und erstmals am 23. Mai 1995 als eine objektorientierte, plattformunabhängige und leicht zu handhabende Programmiersprache vorgestellt. Sie ist eine Komponente der Java-Technologie [24].

  Die Javasyntax ist der Syntax von C++ sehr ähnlich. Allerdings wurden bei der Entwicklung von Java auf komplexe Konstrukte, wie zum Beispiel der Mehrfachvererbung oder der, zwar mächtigen aber fehleranfälligen, Zeigerarithmetik verzichtet [26]. Die interne Speicherverwaltung wird bei Java durch eine automatische Speicherbereinigung erledigt. Diese und andere Java-Eigenschaften führen zu einheitlichen und pflegeleichten Programmen.

Für die Implementierung von SomVis3D wurde aus folgenden Gründen die Programmiersprache Java verwendet:

1. SomVis3D soll die bereits implementierten Funktionalitäten zum Erzeugen einer SOM aus dem Programm ESOM übernehmen. ESOM wurde komplett in Java implementiert. Aus

diesem Grund bietet sich an, Java als Programmiersprache für das neue Programm zu verwenden.

2. Da Java eine sehr umfangreiche Klassenbibliothek bietet, ist die Entwicklung von Anwendungen mit Java komfortabel.

SomVis3D wurde unter der Entwicklungsumgebung *JBuilder* der Firma *Borland* implementiert.

3.3.2 Die Grafikbibliothek

Bei der Wahl einer Grafikbibliothek[2] standen Direct3D, Java3D und OpenGL zur Auswahl.

- **Direct3D** Direct3D ist eine Programmbibliothek für Windows, welche Funktionen für die Entwicklung von dreidimensionalen Anwendungen, vor allem Computerspiele, bereitstellt. Die erste Version von Direct3D wurde 1996 von Mircrosoft als Bestandteil von DirectX vorgestellt und seitdem kontinuierlich weiterentwickelt.

  Weitere Informationen zu Direct3D sind unter *http://www.microsoft.com/windows/directx/* zu finden.

- **Java3D**
  Java3D ist eine Programmbibiliothek für Java, die Funktionen für die Programmierung von 3D Anwendungen bereitstellt. Sie wurde in Zusammenarbeit von Silicon Graphics Inc., Intel Corporation, Apple Computer Inc. und Sun Microsystems Inc. entwickelt. Die erste Implementierung von Java3D wurde 1998 veröffentlicht. Java3D ist szenegraphbasiert, das bedeutet, dass die darzustellende Szene mit Hilfe eines Szenegraphen[3] aufgebaut werden muss.

- **OpenGL**
  OpenGL ist die Abkürzung für Open Graphics Library. Es ist eine Bibliothek welche Grafikbefehle zur Programmierung von zwei- und dreidimensionalen Anwendungen bereitstellt. Erfinder von OpenGL ist die Firma Silicon Graphics Incorporations, kurz SGI, welche 1992

[2] Eine Grafikbibliothek ist eine Programmbibliothek, die einem Programm Funktionen zur Realisierung von dreidimensionalen Szenen bereitstellt.

[3] Ein Szenegraph ist ein gerichteter, azyklischer Graph zum Aufbau hierarchischer Baumstrukturen, welche der Organisation, Speicherung und Darstellung der Szenenobjekte dienen.

die die erste OpenGL Spezifikation veröffentlichte[4]. OpenGL funktioniert nach dem Prinzip einer Statemachine. Das bedeutet, durch spezielle Befehle OpenGL in einen neuen Status gebracht werden kann. Ein Beispiel ist der Farbstatus nach der Initialisierung (1,1,1) (=Weiß). Durch `glColor()` ist es möglich, einen neuen Farbstatus zu setzen. Dann werden alle nachfolgenden Vertices in der angegebenen Farbe gezeichnet. Ein Status ist also solange gültig, bis er durch neue Werte ersetzt wird.

Ein großer Vorteil von OpenGL gegenüber anderen Grafikbibliotheken ist die Plattformunabhängigkeit.

Aus folgenden Gründen wurde OpenGL als Grafikbibliothek für SomVis3D verwendet:

Direct3D von Microsoft beschränkt sich auf Windows-Systeme. Um SomVis3D die Möglichkeit offen zu halten, ohne größe Hindernisse auch für andere Plattformen weiterentwickelt werden zu können, kommt Direct3D für die Entwicklung von SomVis3D nicht in Frage.
Java3D ist szenegraphbasiert. Da die zu visualisierenden SOM aber einfach aufgebaut sind, ist eine Organisation der Szene mithilfe eines Szenegraphen nicht nötig. Ein Vorteil von OpenGL gegenüber Java3D ist die bessere Performanz [28].

3.3.3 Die OpenGL-Programmbibliothek für Java

Um OpenGL unter Java zu nutzen, wird eine OpenGL-Programmbibliothek, welche die OpenGL-Befehle für Java bereitstellt, benötigt. Die bekanntesten OpenGL-Programmbibliotheken sind JOGL, gl4java und LWJGL.

- **gl4java (OpenGL for Java Technology)**
  gl4Java ist eine OpenGL-Programmbibliothek unter Java, welche in der Lage ist, den Funktionsumfang der OpenGL1.3 und GLU1.2 API unterstützt. Seit 1997 wird gl4Java im Rahmen eines Open Source Projektes entwickelt und von Jausoft kostenlos im Internet zum Download angeboten[30]. Durch die hardwarenahe Implementierung von gl4java ist die Geschwindigkeit von mit Java und gl4java entwickelten 3D-Anwendungen durchaus mit C++/OpenGL Anwendungen vergleichbar. Ein Nachteil von gl4java ist, dass es OpenGL nur bis zur Version 1.3 unterstützt und die Benutzung neuer I/O-Geräte nur beschränkt ge-

[4] Weitere Informationen zu OpenGL unter *http://www.opengl.org/*.

währleistet. Eine umfangreiche API (engl. application programming interface) macht Programmierern zudem den Einstieg schwer.

- **LWJGL (Lightweight Java Game Library)**
  LWJGL ist eine OpenGL-Programmbibliothek mit dem Fokus auf die Spieleentwicklung. Mithilfe von LWJGL kann ein Java-Programmierer auf OpenGL-Funktionen zugreifen. Informationsquellen zu LWJGL enthalten keine Angaben über das Entstehungsdatum und die Entwickler von LWJGL. Die aktuelle Version von LWJGL ist 0.98 Alpha, die am 18. August 2005 veröffentlicht wurde. Einige wichtige Ziele bei der Entwicklung von LWJGL sind Geschwindigkeit, Einfachheit, Sicherheit und Robustheit [29]. Der Hauptnachteil von LWJGL liegt in der nicht zulässigen Einbindung von AWT oder Swing Komponenten und in dem stark eingeschränkten Funktionsumfang.

- **JOGL (Java Bindings for OpenGL)**
  JOGL wurde unter der Unterstützung von Sun (Java) und SGI (OpenGL) entwickelt. Die erste Version von JOGL wurde im August 2003 veröffentlicht. Die Weiterentwicklung erfolgt heute durch die Game Technology Group von Sun Microsystems [31]. JOGL ermöglicht die Einbindung von Java-Swing und Java-AWT Komponenten und ist vom Funktionsumfang größer als LWJGL. Die Funktionen der GLUT (OpenGL Utility Toolkit)[5] und GLU (OpenGL Utility Library)[6] sind in JOGL vollständig implementiert.

Bei der Entwicklung von JOGL wurde versucht eine OpenGL Anbindung zu erstellen, die die Nachteile von gl4java und LWJGL nicht besitzt. Für die Entwicklung von SomVis3D wurde aus diesem Grund JOGL als OpenGL Anbindung verwendet.

3.4 Realisierung

Nachdem die gewählten Werkzeuge vorgestellt wurden, wird in diesem Abschnitt die Realisierung der wichtigsten Anforderungen beschrieben. Dabei soll jeweils auf die verschiedenen Möglichkeiten zur Erfüllung einer Anforderung und auf die Umsetzung des gewählten Lösungsweges eingegangen werden. Die wichtigsten Anforderungen ergeben sich aus dem Anwendungsfalldiagramm

[5] Weitere Informationen unter *http://www.opengl.org/documentation/specs/glut/spec3/spec3.html*.
[6] Weitere Informationen unter *http://www.opengl.org/documentation/specs/glu/glu1_3.pdf*.

zu SomVis3D (vgl. Abbildung 3.1). Der letzte Teil dieses Abschnittes widmet sich der Umsetzung der übrigen Anforderungen an SomVis3D.

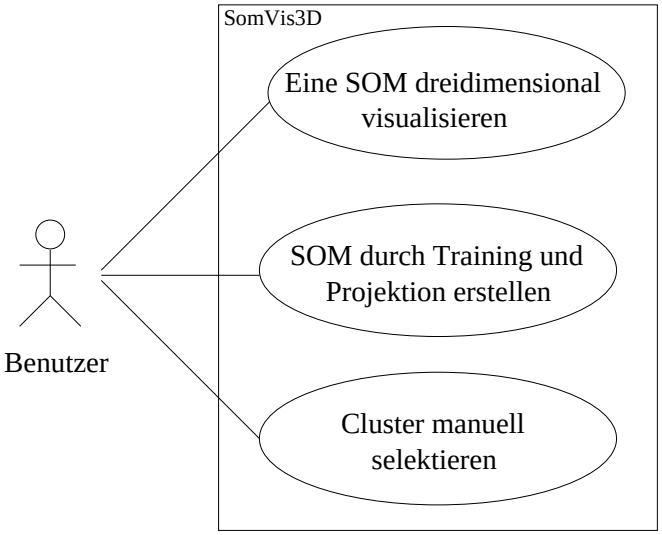


Bild 3.1: Anwendungsfalldiagramm zu SomVis3D

3.4.1 Die dreidimensionale Visualisierung einer SOM

Die dreidimensionale Visualisierung einer SOM bezieht sich auf ihre Darstellung als U-Matrix und die Darstellung der Komponentenkarten (vgl. Abschnitt 2.7.2). In diesem Anschnitt soll nun zuerst erläutert werden, wie die dreidimensionale Visualisierung einer SOM in SomVis3D realisiert werden kann. Danach wird auf die Umsetzung eingegangen.

Möglichkeiten

Um das Problem der Visualisierung zu lösen, ist es hilfreich, sich nochmals den Aufbau einer SOM zu veranschaulichen: Eine SOM bildet sich aus einem Netz von untereinander verbundenen Neuronen. Durch das Training werden die Neuronen gewichtet und das Netz gewinnt an Struktur (siehe Abschnitt 2.6.2), ähnlich einer Landschaft (siehe Abbildung 3.2).

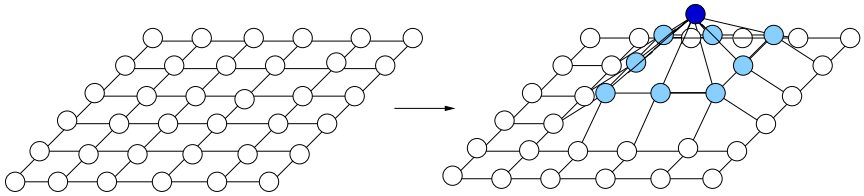


Bild 3.2: Strukturierung des Netzes

In der Computergrafik werden Oberflächen bevorzugt durch folgende Modellierungstechniken erzeugt [18]:

1. **polygonale Netze**
   Polygonale Netze sind stückweise lineare Approximationen für Flächen, die durch eine Menge von miteinander verbundenen Polygonen, meist Dreiecksflächen, definiert sind. Abbildung 3.3 zeigt ein polygonales Netz aus Dreiecksflächen.

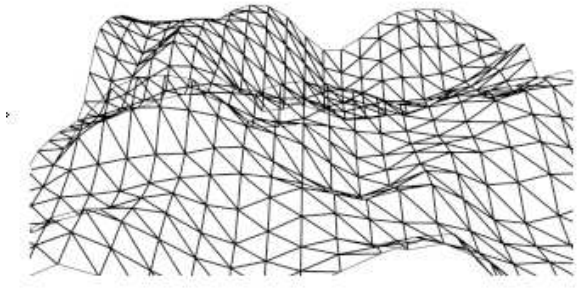


Bild 3.3: polygonales Netz [32]

   Die Modellierung von polygonalen Netzen ist nicht besonders rechenaufwendig und somit schnell. Die Möglichkeit, ohne viel Aufwand das Polygonnetz (zum Beispiel durch Veränderung der Höhe der Eckpunkte von aneinander liegenden Polygonen) zu verändern, macht dieses Modellierungsverfahren sehr flexibel. Ein weiterer Vorteil der polygonalen Netze ist, dass sie in der Umsetzung sehr einfach sind.
   Doch diesen Vorteilen stehen auch Nachteile gegenüber: Eine Oberfläche, welche mit Polygonen modelliert wurde, erscheint häufig sehr kantig. Dieses ist vor allem der Fall, wenn die Modellierung mit nur wenig Polygonen umgesetzt wurde. Abgerundete Oberflächen sind

nur mit einer hohen Anzahl von Polygonen realisierbar. Dieses beansprucht aber wiederum einen größeren Speicherplatz.

2. **Freiformflächen**
   Freiformflächen sind stückweise polynomiale Flächen, welche mathematisch durch Formeln definiert werden. Beispiele zu Freiformflächen sind NURBS-, Bézier- oder B-Splineflächen. In Abbildung 3.4 ist eine Bézierfläche dargestellt.

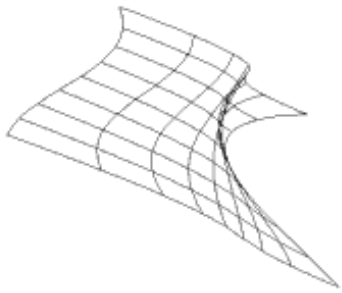


Bild 3.4: Bézierfläche [33]

Der Vorteil der Freiformflächen liegt in der Möglichkeit zur Berechnung von sehr glatten Oberflächen, die auch bei Vergrößerung nicht „kantig“ erscheinen und sich durch verhältnismäßig wenig Punkte beschreiben lassen. Allerdings sind die Freiformflächen dadurch, dass jeder Punkt durch eine mathematische Formel beschrieben wird, erheblich rechenaufwendiger als die polygonalen Netze.

Welche Technik kann für die Modellierung der SOM sinnvoll eingesetzt werden?

Die Ausdehnung der SOM kann der Benutzer frei wählen. Dabei sind sehr kleine, wie auch sehr große SOM denkbar. Um auch sehr große Darstellungen schnell berechnen zu können und aus den Gründen der einfacheren Umsetzung wurden hier die polygonalen Netze als Modellierungsverfahren für die dreidimensionale Oberfläche gewählt.

Eine SOM wird aber nicht nur durch eine strukturierte Oberfläche, sondern auch durch die Cluster repräsentiert. In ESOM werden die Cluster durch ihre Repräsentanten dargestellt: *die Bestmatches*. Ein Bestmatch bildet sich hier als der mittlere Merkmalsvektor aller einem bestimmten Cluster zugeordneten Datenpunkte. Die Bestmatches können als Punkte dargestellt werden. Diese Form der Repräsentation der Cluster soll auch in SomVis3D übernommen werden.

Umsetzung

Die Anforderung der Visualisierung einer SOM kann in zwei Teilanforderungen aufgeteilt werden (siehe Abbildung 3.5):

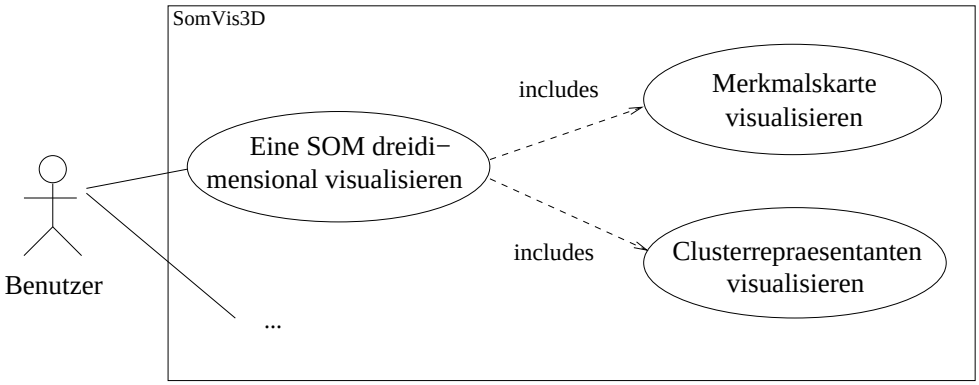


Bild 3.5: Teilanforderungen der Visualisierung

Diese Teilanforderungen sind in SomVis3d in 2 Methoden implementiert:

1. Visualisierung der SOM → `void drawBackground();`
2. Visualisierung der Clusterrepräsentanten → `void drawForeground();`

Wie diese Methoden ihre Aufgaben umsetzen, soll im Folgenden erläutert werden:

1. **Visualisierung der SOM – `drawBackground()`**
   Um eine Oberfläche erstellen zu können, werden die dreidimensionalen Koordinaten der Neuronen des Netzes benötigt. In einem noch nicht trainiertem Netz liegen die Neuronen gleichverteilt auf einer Ebene. Durch das Training gewinnt das Netz an Struktur. Diese Struktur lässt sich, wie in Abschnitt 2.7.2 bereits beschrieben, zum Beispiel durch die U-Matrix visualisieren.

   Für die U-Matrix-Visualisierung wird die Ausdehnung des Netzes sowie die Höheninformationen eines jeden Neurons benötigt. Die Höheninformationen der Neuronen entstehen während des Trainings und werden in einer Datei (U-Matrix-Datei/ *.umx) abgespeichert. Abbildung 3.6 zeigt einen Ausschnitt einer solchen Datei. Zeilen die mit dem Zeichen „%“

beginnen, gehören zum Header. In dem Header der U-Matrix-Dateien wird die Ausdehnung der SOM angegeben. Die SOM, deren Strukturierung in dieser Datei gespeichert ist, hat also eine Ausdehnung von 50x82 Neuronen. Das bedeutet, dass die Datei aus 50 Zeilen und 82 Spalten besteht. Jeder Eintrag in der Dateit steht für die Höhe eines Neurons. Zum Beispiel hat das erste Neuron (erste Zeile und erste Spalte nach dem Header) der SOM dieser Datei eine Höhe von 0.031.

```
%50 82
0.031, 0.063, 0.131, 0.209, 0.266, 0.272, 0.264, 0.266, 0.267,
0.015, 0.012, 0.012, 0.012, 0.013, 0.012, 0.025, 0.062, 0.134,
0.029, 0.027, 0.025, 0.028, 0.033, 0.041, 0.049, 0.049, 0.044,
0.295, 0.296, 0.248, 0.177, 0.118, 0.097, 0.083, 0.073, 0.059,
0.161, 0.19, 0.223, 0.277, 0.302, 0.298, 0.286, 0.215, 0.175,
0.026, 0.045, 0.089, 0.156, 0.237, 0.29, 0.302, 0.32, 0.329,
0.017, 0.01, 0.007, 0.008, 0.014, 0.019, 0.035, 0.073, 0.142,
0.027, 0.025, 0.026, 0.03, 0.037, 0.046, 0.053, 0.05, 0.043,
0.282, 0.3, 0.282, 0.254, 0.2, 0.172, 0.166, 0.155, 0.136,
0.207, 0.227, 0.241, 0.254, 0.241, 0.224, 0.183, 0.127, 0.087,
0.022, 0.033, 0.058, 0.106, 0.174, 0.25, 0.3, 0.334, 0.344,
0.027, 0.014, 0.01, 0.012, 0.023, 0.039, 0.063, 0.102, 0.167,
0.031, 0.028, 0.029, 0.032, 0.04, 0.049, 0.052, 0.048, 0.041,
0.271, 0.31, 0.319, 0.316, 0.287, 0.256, 0.254, 0.243, 0.229,
```

Bild 3.6: Ausschnitt einer U-Matrix Datei (hepta_zt_82x50e20.umx)

Jetzt müssen die Neuronen nur noch zu Polygonen (hier Dreiecken) und diese zu einer Fläche miteinander verbunden werden. Dabei sollen die Polygone so aneinander gefügt werden, das ein Polygon sich als Erweiterung des Vorgängerpolygons definiert. Abbildung 3.3 zeigt, wie eine Oberfläche aus Dreieckspolygonen erzeugt werden kann. Die Zahlen an den Ecken sollen dabei die Zeichenreihenfolge darstellen. Dieser Vorgang wird auch Triangulierung genannt und kann in OpenGL effizient durch den Befehl `GL_TRIANGLE_STRIP` umgesetzt werden.

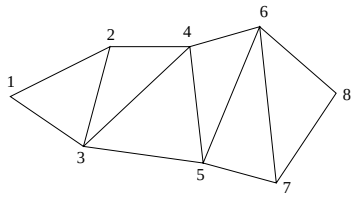


Bild 3.7: Triangle Strip

Der Pseudocode der Implementierung einer solchen Triangulierung sieht in SomVis3D wie folgt aus:

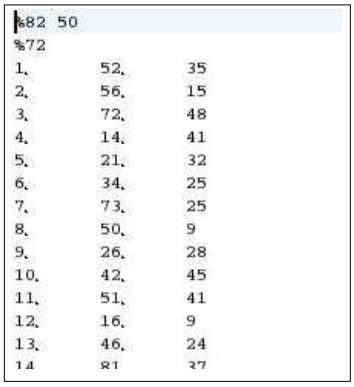
```
Für alle Neuronen in z-Richtung{
    gl.glBegin(gl.GL_TRIANGLE_STRIP)
     Für alle Neuronen in x-Richtung{
         h_1 = Höhe des aktuellen Neurons
         h_2 = Höhe des Nachfolgeneurons
         Zeichne das aktuelle Neuron an (x, h_1, z)
         Zeichne das Nachfolgeneuron an (x, h_2, z+1)
    }
     gl.glEnd()
}
```

Durch eine farbliche Gestaltung der Oberfläche kann die Strukturierung der SOM noch anschaulicher dargestellt werden. In SomVis3D sind verschiedene Farbgebungen in Form von Farbverläufen möglich. Dabei werden den Neuronen, abhängig von ihrer Höhe, Farben zugeordnet. Auf den Dreiecksflächen, welche durch die Neuronen als Eckpunkte gebildet werden, werden die Farben der Eckpunkte interpoliert.

2. **Visualisierung der Clusterrepräsentanten – `drawForeground()`**
   Die Clusterrepräsentanten, auch Bestmatches genannt, repräsentieren - wie der Name es schon sagt - die einzelnen Cluster auf der SOM. Die einfachste Möglichkeit diese zu visualisieren, ist sie als Voxel auf der SOM abzubilden.

   Die Bestmatches werden nach dem Training einer SOM in einer Datei (Bestmatch-Datei/ *.bm) abgespeichert. Eine Bestmatch-Datei enthält die Indizes und die Koordinaten (x, y) der Bestmatches. Dabei sind meistens mehrere Neuronen einem Bestmatch zugeordnet. Die Koordinaten (x, y) sind die Positionen der Bestmatches auf der SOM. Die Höhe der Bestmatches kann jeweils an die Höhe der SOM an der Stelle (x, y) angepasst werden. Abbildung 3.8 zeigt einen Ausschnitt einer Bestmatch-Datei.

   Wie bei der U-Matrix-Datei ist der Header der Bestmatch-Dateien durch „$\%$“ gekennzeichnet. Die Ausdehnung der SOM, zu welcher die Bestmatches dieser Datei gehören, ist in der ersten Zeile abzulesen. In diesem Fall ist die SOM also 82x50 Neuronen groß. In der zweiten Headerzeile ist die Anzahl der Bestmatches dieser SOM gegeben.



```
%82 50
%72
1.      52.     35
2.      56.     15
3.      72.     48
4.      14.     41
5.      21.     32
6.      34.     25
7.      73.     25
8.      50.     9
9.      26.     28
10.     42.     45
11.     51.     41
12.     16.     9
13.     46.     24
14      81      37
```

Bild 3.8: Ausschnitt einer Bestmatch-Datei (hepta_zt_82x50e20.bm)

Die Darstellung der Bestmatches ist wie folgt implementiert:

```
Für alle Bestmatches
{
    int row = Bestmatch.getRow();
    int column = Bestmatch.getColumn();
    int height = grid.getHeight(row,column); //wobei grid
    die aktuelle SOM darstellt

    //draw Bestmatch
    gl.glPointSize(pointSize);
    gl.glBegin(GL.GL_POINTS);
    gl.glVertex3f(row, height, column);
    gl.glEnd();
}
```

Die Methoden `void drawBackground()` und `void drawForeground()` werden in der Methode `void display()` aufgerufen. Ein Beispiel einer visualisierten SOM ist in Abbildung 3.9 dargestellt.

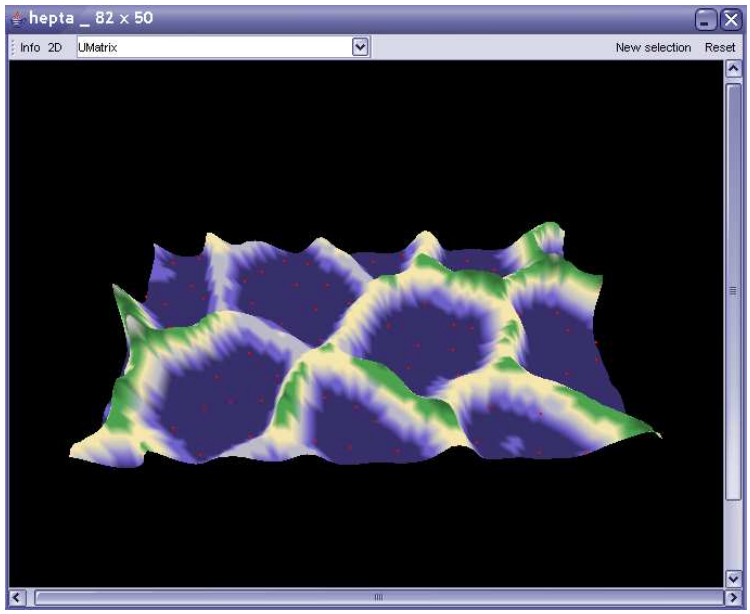


Bild 3.9: Beispiel einer SOM

Visualisierung der Komponentenkarten

Für die Darstellung der Komponentenkarten muss für jedes Attribut des Datensatzes eine Höhenkarte erstellt werden. Allerdings kann die Anzahl der Komponenten (Attribute) der zur visualisierenden Datensätze stark variieren – es kann vorkommen, dass ein Datensatz nur eine, zwei oder drei Komponenten besitzt, in der Praxis hat ein Datensatz allerdings häufig eine sehr große Anzahl von Komponenten. Falls der Benutzer auf die Komponentendarstellung umschaltet, werden ihm alle Komponentenkarten in einem Frame repräsentiert. Aus Gründen der Übersichtlichkeit und aus Performancegründen wurde für die Darstellung der Komponentenkarten hier die zweidimensionale Ansicht gewählt.

Die dreidimensionale Ansicht der Komponentenkarten wurde im Rahmen der Umsetzung des *InfoFrames* (siehe Abschnitt 3.4.4) implementiert. In dem InfoFrame hat der Benutzer die Möglichkeit sich die Namen aller Komponentenkarten in einer Liste zeigen zu lassen. Durch Mausklick auf eine oder mehrere Komponentennamen kann er die Komponentenkarten wählen, von welchen er die dreidimensionale Ansicht sehen möchte.

Da die Umsetzung der dreidimensionalen Ansicht einer Komponentenkarte der Umsetzung der

U-Matrix-Ansicht entspricht, soll an dieser Stelle nur auf die zweidimensionale Sicht auf die Komponentenkarten eingegangen werden:

Wie die Modellierung der U-Matrix-Ansicht, basiert die Modellierung der zweidimensionalen Komponentenkarten auf den polygonalen Netzen. Für jede Komponentenkarte wird eine Oberfläche aus Dreiecksflächen erstellt. Für die zweidimensionale Ansicht werden die Höhenwerte der Neuronen (Eckpunkte der Dreiecke) nicht benötigt. So entsteht für jede Komponentenkarte ein planares Netz. Diese Netze werden so verschoben, dass sie nebeneinander liegen. Folgender Ausschnitt aus der Implementierung zeigt die Umsetzung der Visualisierung der Komponentenkarten:

```
compCount = Anzahl der Komponentenkarten;
xSize = Größe der Komponentenkarten in x-Richtung;
ySize = Größe der Komponentenkarten in y-Richtung;

gl.glPushMatrix();
//alles in die Mitte translatieren
gl.glTranslated(-(compCount * (xSize + 3))/ 2, -ySize/2, 0);
//xSize + 3; für den Zwischenraum zwischen den Karten
Für alle Komponentenkarten
{
   gl.glPushMatrix();
   //Komponentenkarten nebeneinander anordenen
   gl.glTranslated(i*(xSize + 3),0,0);
   //mit i als aktuelle Komponentenkarte

   drawBackground(i);
   drawForeground(i);
   gl.glPopMatrix();
}
gl.glPopMatrix();
```

Um die Strukturen der Komponentenkarten farblich zu veranschaulichen, werden hier die Höhenwerte der Neuronen wieder benötigt: Wie bei der dreidimensionalen Ansicht der U-Matrix, werden den Neuronen, abhängig von ihrer Höhe, Farben zugeordnet. Auf den Dreiecksflächen des Netzes werden die Farben der Eckpunkte interpoliert.

Die Abbildung 3.10 zeigt die Komponentenkarten des Datensatzes hepta.lrn.

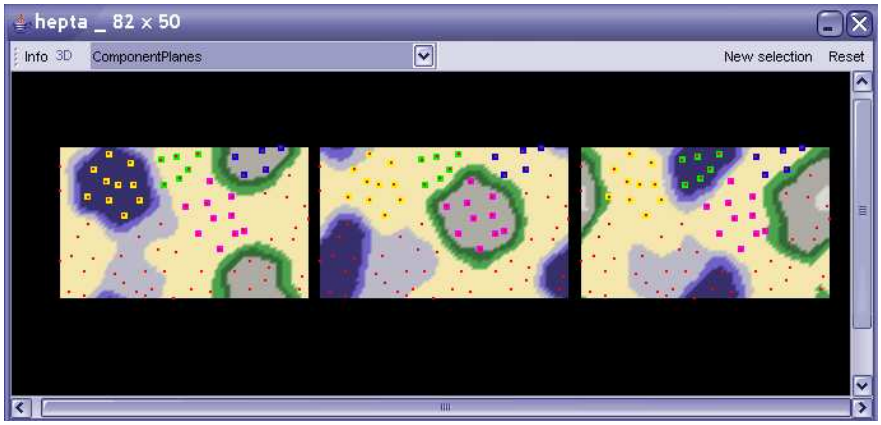


Bild 3.10: Komponentenkarten des Datensatzes hepta.lrn

3.4.2 Die Erstellung einer SOM

Die Erstellung einer SOM soll in SomVis3D durch das Training und die Projektion möglich sein (siehe Abbildung 3.11).

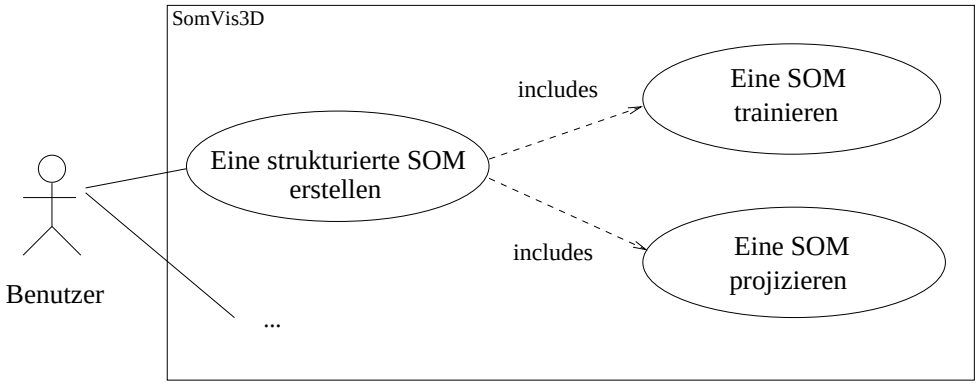


Bild 3.11: Teilanforderungen der Erstellung einer SOM

Durch ein Training wird aus einer Menge von Eingabedaten, welche aus einer Datei (Learn-Datei, siehe Abschnitt 3.4.4) ausgelesen werden, eine strukturierte SOM erstellt. Nach einem Training werden die Bestmatches in einer *Bestmatch-Datei*, die entstandenen Gewichte der Neuronen in eine *Weights-Datei*, die Höhen der Neuronen für die U-Matrix-Dartstellung in einer *U-Matrix-Datei* gespeichert. (Im Abschnitt 3.4.4 wird genauer auf die verschiedenen Dateitypen eingegangen.)

Durch eine Projektion werden die Daten aus einer eingelesenen Learn-Datei auf eine bereits strukturierte SOM abgebildet. Das heißt, es werden die Datenpunkte der Learn-Datei auf die Karte projiziert, indem das jeweils an besten passende Neuron der Karte ermittelt wird. Das Ergebnis der Projektion ist eine Datei, in welcher diese Neuronen als Bestmatches abgespeichert sind.

Um die Funktionen „Training“ und „Projektion“ umzusetzen, sollten die bereits implementierten Funktionalitäten des Programmes ESOM genutzt werden. Im Folgenden wird beschrieben, wie das Training und die Projektion in SomVis3D übernommen wurden.

1. Training
   Die Funktion des Trainings ist in ESOM als eine Methode der Klasse SOM (`somvis3d.tools.databionics.train`) implementiert und konnte nach minimalen Veränderungen in SomVis3D übernommen werden. Das Training benötigt eine Vielzahl an Parametern, deren Werte der Benutzer wählen kann. Einige Beispiele dieser Parameter sind *Trainingsart, Größe der SOM, Anzahl der Trainingsepochen oder Initialisierungsmethode der SOM*[7]. Aus diesem Grund muss SomVis3D eine Benutzerschnittstelle bereitstellen, die dem Benutzer die Möglichkeit gibt, diese Parameter zu setzen. In ESOM wurde die Benutzerschnittstelle durch einen Dialog realisiert. Auch SomVis3D soll dem Benutzer die Wahl der Parameter über einen Dialog ermöglichen. Der Dialog wurde in der Klasse `somvis3d.tools.Trainingdialog.java` implementiert. Außer dem Setzen der Trainingsparameter hat der Benutzer über den Dialog die Möglichkeit eine neue Trainingsdatei (Learn-Datei/ *.lrn) zu wählen, die Namen der entstehenden Dateien zu erfahren, den Fortschritt des Trainings zu beobachten und es auf Wunsch abzubrechen.

   Das Erscheinungsbild des Trainingsdialoges von SomVis3D ist sehr stark an das des Trainingsdialoges von ESOM angelehnt. Abbildung 3.12 zeigt den Trainingsdialog von SomVis3D.

[7] Unter *http://databionic-esom.sourceforge.net/user_de.html* sind weitere Informationen zum Training und den Trainingsparametern zu finden.

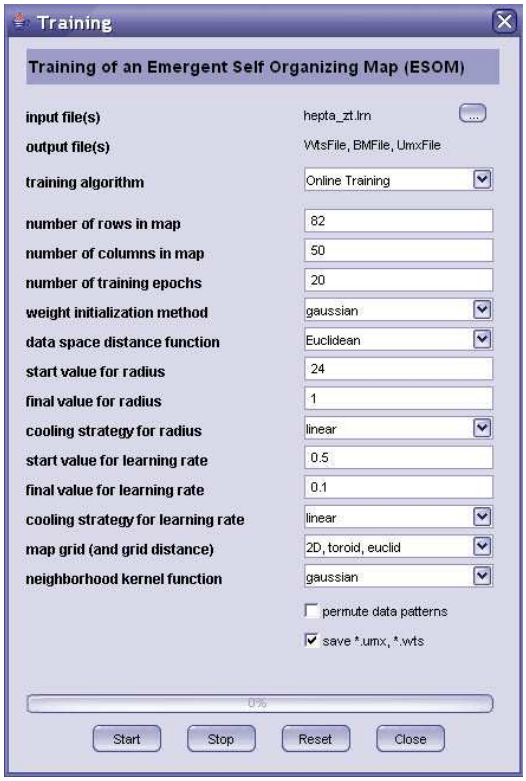


Bild 3.12: Der Trainingsdialog von SomVis3D

Weitere Informationen zum Training und den Trainingsparametern sind auch unter *http://databionic-esom.sourceforge.net/user_de.html* zu finden. In Abschnitt 3.5.2 wird genauer auf die Klasse `Trainingdialog.java` eingegangen.

2. Projektion
   Die Projektion wurde in ESOM in der Klasse `esom/tool/projection/ProjectionTool.java` implementiert. Zur Umsetzung der Projektion wurden nur die zur Projektion relevanten Methoden in SomVis3D übernommen und in eine neue Klasse `somvis3d.tools.Projection.java` übertragen.
   Um eine Projektion durchzuführen, muss Learn-Datei (welche die zu projizierenden Daten enthält) und eine Weights-Datei (in welcher die SOM gegeben ist, auf die die Daten der Learn-Datei projiziert werden soll) geladen werden. Über einen Dialog, welcher in der Klasse `somvis3d.tools.ProjectionDialog.java` implementiert ist, soll der Be-

nutzer die Möglichkeit haben, die zu projizierenden Dateien zu wählen, Auskunft über die erstellte Bestmatch Datei zu erhalten, den Fortschritt der Projektion zu beobachten oder sie, falls gewünscht, abzubrechen. Abbildung 3.13 zeigt den Projektionsdialog von SomVis3D. Hier sind `hepta_zt.lrn` und `hepta_zt_82x50e20.wts` die zu projizierenden Dateien. Die durch die Projektion entstehende Datei ist `hepta_zt_projected.bm`.

Bild 3.13: Der Projektionsdialog von SomVis3D

In Abschnitt 3.5.2 wird die Klasse `Projektion.java` genauer beschrieben.

3.4.3 Die manuelle Clusterselektion

Durch eine manuelle Clusterselektion soll dem Benutzer von SomVis3D die Möglichkeit gegeben werden, Bestmatches manuell zu Clustern zusammenzufassen. Für die Umsetzung dieser Anforderung sind verschiedene Lösungswege möglich, welche im Folgenden kurz vorgestellt werden sollen. Im Anschluss daran wird die Umsetzung der manuellen Clusterselektion erläutert.

Möglichkeiten

Für die Umsetzung der Clusterselektion gibt es verschiedene Lösungswege: Zum einen kann die Selektion auf der zwei- oder auf der dreidimensionalen Darstellung erfolgen. Zum anderen muss festgelegt werden, auf welche Weise die Objekte selektiert werden können.

Ein allgemeiner Vergleich der Selektion auf einer zweidimensionalen Ansicht mit der Selektion auf einer dreidimensionalen Ansicht ist schwierig, da die jeweiligen Vor- und Nachteile vor allem von der Art der Anwendung abhängen. In SomVis3D wurde aus folgenden Gründen die Selektion auf der zweidimensionalen Ansicht umgesetzt:

1. In der dreidimensionalen Ansicht der SOM können Bestmatches teilweise durch „Berge“ im Vordergrund überdeckt werden. Um eine Übersicht über alle Bestmatches der SOM zu bekommen, müsste der Benutzer die Landschaft so rotieren, dass er von oben drauf schauen kann. Diese Darstellung ist mit einer zweidimensionalen Ansicht gleichzusetzen.

2. In SomVis3D wird das Höhenprofil durch die Darstellung der Farbe unterstützt. Aus diesem Grund könnte der Anwender das Höhenprofil auch erkennen, wenn alle Daten auf einer Ebene liegen. Eine Umwandlung einer dreidimensionalen in eine zweidimensionale Ansicht würde somit einen für die Selektion minimalen Informationsverlust bedeuten.

Für die Art und Weise, wie Objekte selektiert werden, gibt es verschiedene Möglichkeiten:

- Indirekte Selektion
  Die indirekte Selektion umfasst Selektionsverfahren, bei denen die Objekte indirekt, zum Beispiel über eine Liste, über Spracherkennung oder über direkte Texteingabe angesprochen werden.

  Diese Selektionsart kommt ohne jeglichen Kontakt mit den Objekten aus und macht es somit auch möglich, nicht sichtbare Objekte zu wählen. Um eine eindeutige Objektauswahl zu ermöglichen, ist es allerdings Voraussetzung, dass alle Objekte mit einem bekannten Identifikationsnamen versehen sind [34].

- Direkte Selektion
  Eine direkte Selektion der Objekte kann auf verschiedene Arten realisiert werden: Objekte können zum einen durch einfaches Anklicken angesprochen werden, oder durch Umfassen der zu selektierenden Objekte mit Bounding Volumes[8], Bounding Boxes[9] oder frei definierbaren Körpern oder Formen.

[8]Ein Bounding Volume ist ein möglichst einfacher geometrischer Körper, welcher komplexere Körper im dreidimensionalen Raum umschließt.

[9]Eine Bounding Box ist eine möglichst einfache geometrische Form, um komplexere Formen im zweidimensionalen Raum zu umschließen.

Über diese Möglichkeiten hinaus gibt es noch weitere Überlegungen, welche im Vorfeld der Realisierung gemacht werden sollten: Für viele Anwendungen ist es wichtig, sowohl einfache als auch mehrfache Selektion zu ermöglichen. Desweiteren sollte in Erwägung gezogen werden, ob es für den vorliegenden Anwendungsfall nützlich sein kann, auch die Selektion von Gebieten, in welchen sich Objekte befinden, zu realisieren. Bei einer Selektion sollte dem Benutzer ausreichend Feedback gegeben werden: Es muss gut erkennbar sein, welche Objekte ausgewählt sind und welche potentiell ausgewählt würden (Mouse-Over-Effect). Dieses kann zum Beispiel durch eine farbliche Hervorhebung umgesetzt werden [34].

Ziel der manuellen Clusterselektion in SomVis3D ist das Zusammenfassen von Bestmatches zu Clustern. Die direkte Selektion durch Anklicken der Objekte ist zwar intuitiv und funktioniert gut, falls aus einer Menge von Objekten einzelne Objekte gewählt werden sollen, ist aber aus mehreren Gründen für die Clusterselektion von SomVis3D nicht geeignet:
Bestmatches werden als Voxel dargestellt und sind demnach sehr klein. Das Treffen eines einzelnen Voxels mit der Maus könnte sich als schwierig erweisen.Im Falle, dass nicht nur einzelne, sondern eine Vielzahl von Bestmatches zu einem Cluster zusammengefasst werden sollen, ist diese Selektionsmöglichkeit nicht besonders effizient.

Die Selektion durch Umfassen der zu selektierenden Bestmatches durch Bounding Boxes oder frei definierbaren Formen wäre hier die bessere Alternative. In SomVis3D wurde die direkte Selektion durch Bounding Boxes umgesetzt. Als Bounding Box wurde vorerst die einfachste Form, die Rechtecke implementiert. Da eine Selektion mithilfe von Rechtecken eine sehr „starre“ Methode darstellt, wäre als eine Weiterentwicklung von SomVis3D die direkte Selektion durch frei definierbare Formen denkbar.

Da auf einer dreidimensionalen SOM die Bestmatches durch im Vordergrund liegende „Berge“ verdeckt werden könnten, wäre es sinnvoll, wenn SomVis3D auch die indirekte Selektion unterstützen würde. Die indirekte Selektion wurde in SomVis3D nicht im Rahmen der Clusterselektion, sondern im Rahmen der Informationsanzeige implementiert (siehe Abschnitt 3.4.4).

Umsetzung

Die direkte Selektion mithilfe von Rechtecken als Bounding Boxes soll in SomVis3D wie folgt funktionieren:

Nachdem der Benutzer von der dreidimensionalen auf die zweidimensionale Darstellung der SOM gewechselt hat, ist eine Clusterselektion möglich. Durch Ziehen mit der Maus mit gedrückter linker Maustaste kann ein zweidimensionales Rechteck auf der SOM definiert werden. Alle Bestmatches, welche sich innerhalb dieses Rechteckes befinden, gelten als potentiell ausgewählt. Die potentiell selektierten Bestmatches sollen weiß markiert werden. Erst wenn der Benutzer die linke Maustaste losläßt, wird ein Cluster dieser Bestmatches erstellt. Die Bestmatches, welche zu einem Cluster gehören, sollen farblich markiert werden, wobei jedem Cluster eine andere Farbe zugeordnet wird. Auf diese Weise soll der Benutzer die Möglichkeit haben, nacheinander mehrere Cluster von Bestmatches zu selektieren. Außerdem soll ihm durch das parallele Betätigen der SHIFT-Taste die Gelegenheit gegeben werden, die aktuell selektierten Bestmatches dem zuletzt definierten Cluster hinzuzufügen.

Die Anforderung der Selektion in SomVis3D kann demnach in mehrere Teilanforderungen unterteilt werden (siehe Abbildung 3.14):

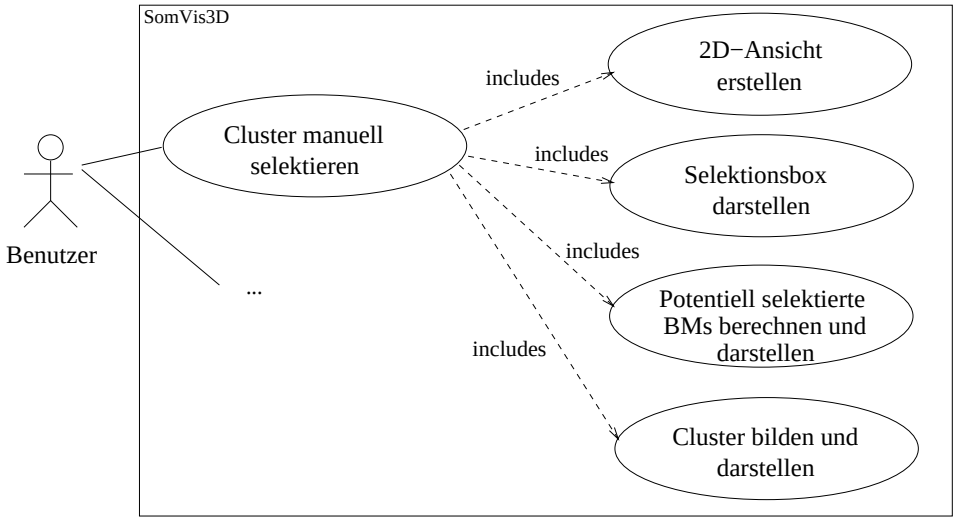


Bild 3.14: Teilanforderungen der Clusterselektion

Im Folgenden soll die Umsetzung dieser Teilanforderungen erläutert werden. Alle Funktionen, welche die Selektion realisieren, wurden in der Klasse `somvis3d.visualisation.SelectionView.java` implementiert.

- **Erstellung einer zweidimensionalen Ansicht der SOM**
  Die Vorgehensweise bei der Erstellung der zweidimensionalen Ansicht der SOM entspricht der Erstellung der zweidimensionalen Ansicht der Komponentenkarten (siehe Abschnitt 3.4.1):

  Die SOM wird ebenfalls als ein polygonales Netz aus Dreiecksflächen modelliert, bei welchem die Höhenwerte der Eckpunkte der Dreiecksflächen, also der Neuronen, keine Rolle spielen. Das Ergebnis ist ein planares Netz, welches die Oberfläche der SOM darstellt. Für die Farbgebung der Oberfläche werden die Höhenwerte der Neuronen wieder herangezogen. Abbildung 3.15 zeigt ein Beispiel einer zweidimensionalen Ansicht einer SOM.

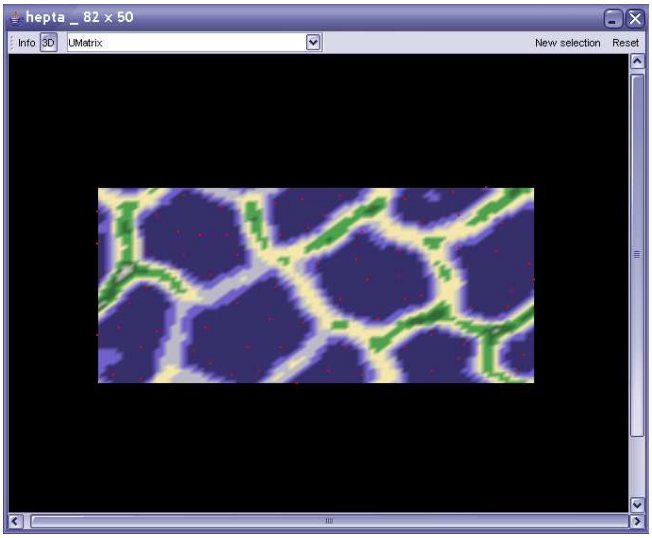


Bild 3.15: 2D-Ansicht einer SOM (hepta_82x50e20.umx)

  Da bei der zweidimensionalen Ansicht keine Daten verdeckt werden können, ist die Rotation hier nicht mehr erforderlich und wurde deaktiviert. Die Zoomfunktion wurde beibehalten.

- **Darstellung der Selektionsbox**
  Die Selektionsbox ist ein Rechteck beliebiger Größe und soll das Bounding Box für die Selektion darstellen. Der Benutzer hat durch Ziehen der Maus mit gedrückter linker Maustaste die Möglichkeit diese Selektionsbox zu definieren. Durch das Interface `java.awt.event.MouseListener` stellt Java bereits vordefinierte Funktionen bereit, um die Mauskoordinaten innerhalb eines Fensters zu bestimmen.

Eine Mausbewegung zur Definition der Selektionsbox besteht aus 3 Schritten: dem Ansetzen der Maus (linke Maustaste drücken), dem Bewegen der Maus (mit gedrückter linker Maustaste) und dem Absetzen der Maus (linke Maustaste loslassen). In jeder dieser Schritte werden folgende Anweisungen ausgeführt. Diese Anweisungen beziehen sich auf die Abbildung 3.16.

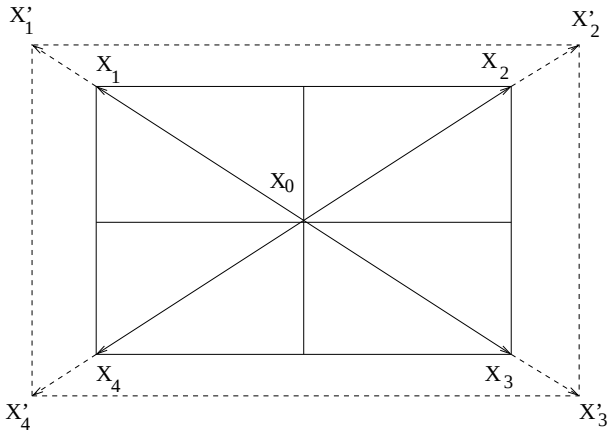


Bild 3.16: Defi nition der Selektionbox

(1) Ansetzen der Maus – Bestimmen der aktuellen Mausposition ergibt den Startpunkt der Selektionsbox, die Koordinaten des Punktes X_0. Vom Startpunkt der Mausbewegung aus, hat der Benutzer vier Möglichkeiten ein Rechteck zu bilden (siehe Abbildung 3.16): Entweder er führt die Maus nach links oben (X_1'), rechts oben X_2', rechts unten (X_3') oder links unten (X_4').

(2) Bewegen der Maus – Bestimmen der aktuellen Mausposition ergibt die Koordinaten des Punktes X_1', X_2', X_3' oder X_4' für den Endpunkt der temporären Selektionsbox. Bei jeder Bewegung der Maus werden diese Koordinaten neu bestimmt und die Selektionsbox wird neu gezeichnet. Auf diese Weise kann die Darstellung der Selektionsbox an die aktuelle Mausbewegung angepasst werden. Die dargestellte Selektionsbox ist nur temporär und soll dem Benutzer lediglich Feedback über die *potentiell selektierten Bestmatches* geben.

(3) Absetzen der Maus – Die letzte, durch die Bewegung der Maus, entstandene Koordinate ist die Koordinate für den Endpunkt der Mausbewegung und stellt somit den zweiten

Eckpunkt X_1, X_2, X_3 oder X_4 für die finale Selektionsbox dar. Diese Koordinate und X_0 werden nun für die Berechnung der *selektierten Bestmatches* herangezogen.

Die Selektionsbox wird mit OpenGL durch einfache, miteinander verbundene Linien gezeichnet. Abbildung 3.17 zeigt eine Selektionsbox auf einer zweidimensionalen Ansicht einer SOM. Die weiß markierten Bestmatches sind die potentiell gewählten Bestmatches. Die farbig unterlegten Bestmatches gehören bereits zuvor selektierten Clustern an.

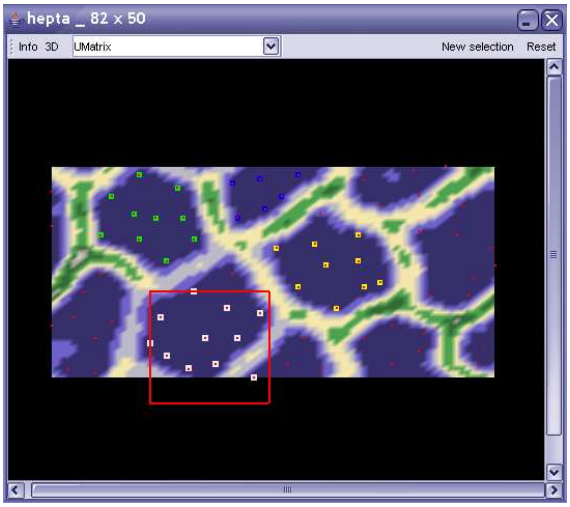


Bild 3.17: Beispiel einer Selektionbox

- **Berechnung und Markierung der potentiell selektierten Bestmatches**
  Als potentiell selektierte Bestmatches gelten die Bestmatches, welche sich innerhalb einer *temporären Selektionsbox* befinden. Eine Selektionsbox ist solange temporär, wie der Benutzer die Maus noch nicht abgesetzt hat, sondern sie mit gedrückter linker Maustaste bewegt (Schritt 2 bei der Definition der Selektionsbox). Zur Berechnung der potentiell selektierten Bestmatches wird X_0 mit den Koordinaten (x_{start}, y_{start}) und die aktuellen Koordinaten von X'_1, X'_2, X'_3 oder X'_4 mit den Koordinaten (x_{end}, y_{end}) verwendet. Ein Bestmatch mit den Koordinaten (x_{bm}, y_{bm}) gilt dann als potentiell selektiert, falls es eine der folgenden Bedingungen erfüllt:

  $$x_{bm} >= x_{start} \;\&\&\; x_{bm} < x_{end} \;\&\&\; y_{bm} <= y_{start} \;\&\&\; y_{bm} > y_{end}$$
  $$x_{bm} <= x_{start} \;\&\&\; x_{bm} > x_{end} \;\&\&\; y_{bm} <= y_{start} \;\&\&\; y_{bm} > y_{end}$$

$$x_{bm} <= x_{start} \;\&\&\; x_{bm} > x_{end} \;\&\&\; y_{bm} >= y_{start} \;\&\&\; y_{bm} < y_{end}$$
$$x_{bm} >= x_{start} \;\&\&\; x_{bm} < x_{end} \;\&\&\; y_{bm} >= y_{start} \;\&\&\; y_{bm} < y_{end}$$

Die potentiell selektierten Bestmatches werden in einem Java-`Vector, (java.util.Vector)`[10] gespeichert und der Zeichenfunktion übergeben, damit sie auf der Ansicht der SOM weiß markiert werden können.

```
Vector tempSel = new Vector();
Für alle Bestmatches
{
   if(Bestmatch innerhalb der temp Selektionsbox)
      tempSel.add(Bestmatch);
}
view.setTempSelection(tempSel);
//mit view = Ansicht der SOM
```

Jedesmal wenn die temporäre Selektionsbox verändert wird, wird ein neuer Vector `tempSel` mit neuen potentiell selektierte Bestmatches erstellt und der Zeichenfunktion übergeben.

- **Bilden des Clusters**
  Ein Cluster ist in SomVis3D als ein Java-`Vector` aus Bestmatches implementiert. Für die Verwaltung und Darstellung aller gebildeten Cluster werden diese wiederum in einen Java-`Vector` abgelegt.

  Die Erstellung eines neuen Clusters und die Berechnung der selektierten Bestmatches erfolgt nachdem der Benutzer die Selektionsbox definiert hat, die Maus also abgesetzt hat (Schritt 3 bei der Definition der Selektionsbox). Falls die Selektion mit gedrückte SHIFT-Taste durchgeführt wurde, wird kein neues Cluster gebildet. Dann wird das zuletzt erzeugte Cluster lediglich durch die selektierten Bestmatches erweitert.
  Der Test, ob ein Bestmatch selektiert ist, entspricht dem Test für die potentiell selektierten Bestmatches. Falls ein Bestmatch als selektiert gilt, wird es dem Cluster hinzugefügt. Dabei gilt, dass bereits selektierte Besmatches nicht erneut selektiert werden können. Nachdem der Selektionstest für alle Bestmatches durchgeführt wurde, wird das Cluster dem Clustervektor

[10] Ein Vektor ist ein Array mit variabler Länge.

zugefügt. Dieser wird dann an die Zeichenfunktion zur Markierung der Cluster übergeben.

```
Vector cluster = new Vector();
if(!newCluster) cluster = zuletzt erzeugtes Cluster;
//newCluster ist false, falls SHIFT-Taste gedrückt

  Für alle Bestmatches
  {
     if(Bestmatch innerhalb Selektionsbox)
        cluster.add(Bestmatch);
  }
  clusterVektor.add(cluster)
  view.setCluster(clusterVektor)
  //mit view = Ansicht der SOM
```

Die Cluster werden markiert, indem alle enthaltenen Bestmatches farbig unterlegt werden. Für jedes Cluster wird dabei eine andere Farbe verwendet. Abbildung 3.18 zeigt die Darstellung verschiedener manuell selektierter Cluster.

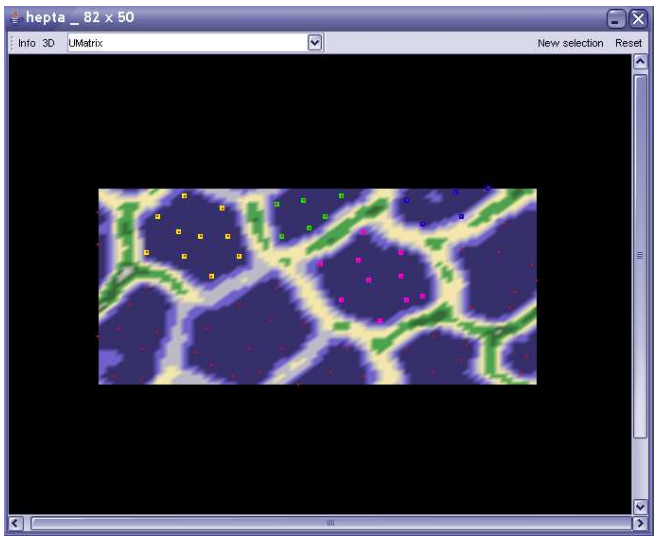


Bild 3.18: Manuell selektierte Cluster

3.4.4 Extrafunktionen

Bisher wurde lediglich auf die Realisierung der Visualisierung und der Erstellung der SOM und auf die manuelle Clusterselektion eingegangen. Abbildung 3.19 zeigt Anforderungen, welche noch nicht beschrieben wurden.

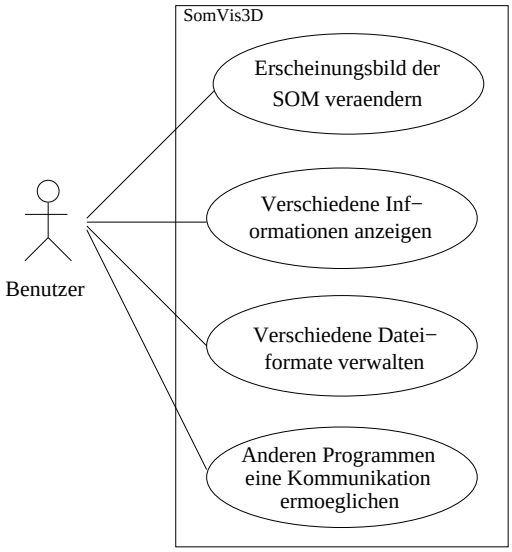


Bild 3.19: Weitere Anforderungen an SomVis3D

Im Folgenden soll nur kurz auf die Realisierung dieser Anforderungen eingegangen werden.

1. **Veränderung des Erscheinungsbildes der SOM**

   Zur Realisierung dieser Anforderung wurde eine Toolbox (`somvis3d.interaction.ToolBox.java`) implementiert. Die Toolbox von SomVis3D bietet folgende Möglichkeiten die Erscheinung einer SOM zu verändern:

   (a) Einstellen des Farbgradienten
   Hier hat der Benutzer die Möglichkeit ein anderes Farbschema zu wählen. Mögliche Farbschemen sind: „beach“, „ocean“, „desert“, „mountains“, „city“, „arctis“ und „jungle“. Die Namen der Farbschemen ist dabei ausschlaggebend für die Wahl der Farben

des Schemas. So zeigt das „desert“-Schema beispielsweise Abstufungen von Braun- bis Gelbtönen.

(b) Einstellen der Bestmatchgröße
Die Bestmatchgröße kann minimal auf 0 (dann wären keine Bestmatches sichtbar) und maximal auf 20 eingestellt werden.

(c) Einstellen der Farbe der Bestmatches
Hier hat der Benutzer die Möglichkeit eine Farbe aus einer Liste zu wählen oder sich auch selbst eine Farbe zu definieren.

(d) Umschalten auf „Tiled Display“
Für die Visualisierung einer SOM können auch toroide Topologien verwendet werden (vgl. Abschnitt 2.7.2). Zur Erkennung der Torusform der SOM hat der Benutzer die Möglichkeit die gekachelte Ansicht (Tiled-Display) zu aktivieren. Um die randlose Topologie darzustellen, erscheint die SOM im Tiled-Modus viermal in gekachelter Ansicht.

Die Toolbox kann von der Menüleiste des Hauptfensters aus geöffnet werden und während der Arbeit mit SomVis3D auch geöffnet bleiben. Falls eine neue SOM dargestellt werden soll, passt sich ihr Erscheinungsbild an die Einstellungen der Toolbox an und falls eine SOM mit einem anderen Erscheinungsbild, als in der Toolbox eingestellt ist, aktiviert wird, passt sich die Toolbox an die aktuelle SOM an.

2. **Anzeige verschiedener Informationen**

Der Benutzer muss die Möglichkeit haben, sich verschiedene Informationen zu der aktuellen SOM anzeigen zu lassen. Zu diesen Informationen zählen:

- Anzahl und Koordinaten der Bestmatches.
- Cluster (über eine Datei geladene oder manuell selektierte) und Anzahl der Bestmatches je Cluster.
- Karteninformationen, wie maximale, minimale und durchschnittliche Höhe der Neuronen.
- Komponenten (Attribute) des Datensatzes.

Zur Umsetzung dieser Anforderung wurde ein Informationsfenster, der InfoFrame (`somvis-3d.interaction.InfoFrame.java`) entwickelt. Der InfoFrame ist ein Fenster, welches als Thread implementiert wurde. Der InfoFrame enthält vier Register:

- In dem ersten Register (*Bestmatches*) findet sich eine Tabelle mit allen Bestmatches und ihren Koordinaten. An dieser Stelle wurde die indirekte Selektion umgesetzt: Der Benutzer hat die Möglichkeit Bestmatches in der Tabelle zu wählen. Diese werden dann in der zwei- oder dreidimensionalen Ansicht der SOM markiert.
- Im zweiten Register (*Classes*) befindet sich eine Auflistung der Cluster. Diese Cluster können manuell selektiert oder aus einer Datei heraus geladen worden sein. Jedes Cluster wird durch eine Farbe identifiziert und zu jedem Cluster wird die Anzahl der zugehörigen Bestmatches angezeigt. Werden die Cluster durch Hinzufügen von Bestmatches verändert, oder werden neue Cluster gebildet, wird dieses Register des InfoFrames aktualisiert. Verändern sich die Cluster, verändert sich auch dieses Register des InfoFrames. Abbildung 3.20 zeigt die Auflistung der Cluster aus Abbildung 3.18 im InfoFrame.

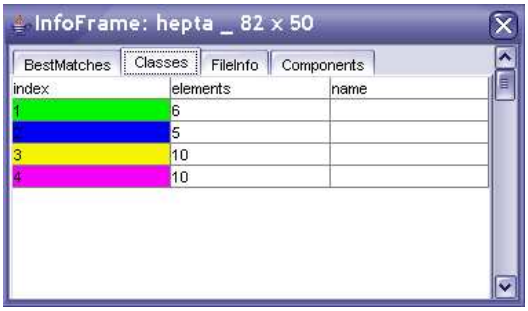


Bild 3.20: Liste von Clustern

- In dem dritten Register (*FileInfo*) des InfoFrames wird eine Auflistung der geladenen Dateien gegeben. Darüber hinaus kann der Benutzer hier die maximale, die minimale und die durchschnittliche Höhe aller Neuronen ablesen.
- Im vierten und letzen Register (*Components*) werden die Komponenten, also die Attribute des Datensatzes mit Namen genannt. Da die übliche Ansicht der Komponentenkarten nur in zweidimensionaler Ansicht gegeben wird (vgl. Abschnitt 3.4.1), hat der Benutzer hier die Möglichkeit sich durch Klicken auf eine Komponente, die dreidimensionale Ansicht anzuschauen.

Da der InfoFrame für jede SOM andere Werte enthält, wird auch für jede SOM ein eigener

InfoFrame geöffnet. Auf diese Weise hat der Benutzer die Möglichkeit die Informationen mehrerer SOM durch das Nebeneinanderstellen der jeweiligen InfoFrames zu vergleichen.

3. **Verwaltung verschiedener Dateitypen**

   Das Programm sollte in der Lage sein, verschiedene Dateitypen zu verwalten.
   SomVis3D ist in der Lage verschiedene Dateiformate zu lesen und zu speichern. In der folgenden Übersicht gibt eine kurze Erklärung zu den Formaten. Alle Dateiformate sind von ESOM übernommen worden.

   (a) **Learn-Dateien (*.lrn)** enthalten die Inputdaten für das Training. In den Spalten werden die Attribute und in den Zeilen die Datensätze beschrieben.

   (b) **Weights-Dateien (*.wts)** enthalten die Gewichtungsvektoren für alle Neuronen der SOM.

   (c) **U-Matrix-Dateien (*.umx)** enthalten die Werte für die U-Matrix-Darstellung: die Höhen der Neuronen.

   (d) **Bestmatch-Dateien (*.bm)** enthalten die Indizes der Neuronen und die Koordinaten ihrer Bestmatches.

   (e) **Class-Dateien (*.cls)** enthalten die Klassenzuordnung der Neuronen der SOM.

   Über den Menüpunkte „File“ des Hauptfensters, hat der Benutzer die Möglichkeit die genannten Dateitypen zu laden oder abzuspeichern.

4. **Kommunikation mit anderen Programmen ermöglichen**

   Diese Anforderung wurde umgesetzt, indem es anderen Programmen ermöglicht wird, ein Objekt von SomVis3D zu erstellen um dann auf verschiedene Methoden zuzugreifen. Als eine Beispielmethode wurde der Aufruf einer Visualisierungsfunktion, welche eine U-Matrix- und eine Bestmatch-Datei als Parameter erhält, implementiert: `public void render( UmxFile umx, BMFile bm)`. Durch das Aufrufen dieser Methoden wird ein Fenster geöffnet, in welchem die dreidimensionale Visualisierung der übergebenen Dateien zu sehen ist.

3.5 Architektur

Nachdem im vorangegangenen Abschnitt die Realisierung der Anforderungen erläutert wurde, soll in diesem Abschnitt näher auf die Systemarchitektur von SomVis3D eingegangen werden. Dieses beinhaltet die Beschreibung des Systems indem die grundlegenden Komponenten und ihre Beziehungen zueinander erläutert werden.
Nachdem im ersten Abschnitt dieses Kapitels ein Gesamtüberblick über die Komponenten von SomVis3D gegeben wird, werden die, für die grundlegenden Funktionalitäten von SomVis3D verantwortlichen Pakete vorgestellt. Diese Funktionalitäten ergeben sich wiederum durch das Anwendungsfalldiagramm (siehe Abbildung 3.21), welches bereits in Abschnitt 3.4 vorgestellt wurde.

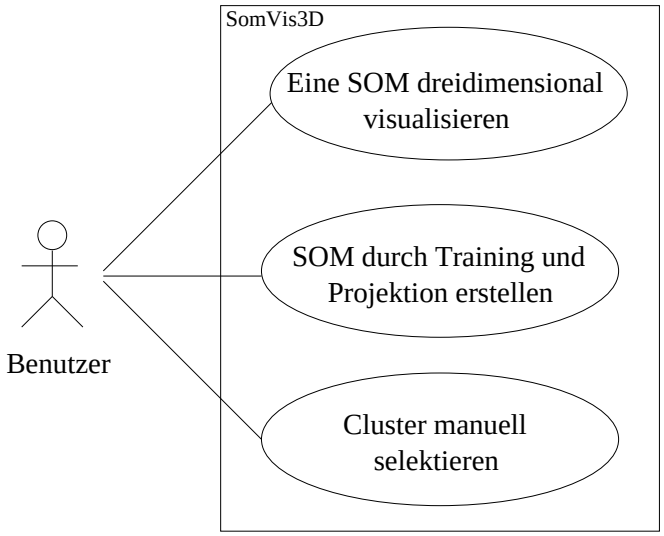


Bild 3.21: Grundlegende Funktionalitäten

Die Funktionen der Visualisierung einer SOM und der manuellen Clusterselektion wurden in dem Paket `somvis3d.visualisation` implementiert. Die Funktionen zur Erstellung einer SOM sind in dem Paket `somvis3d.tools` zu finden. Auf diese beiden Pakete soll nach dem Überblick über die Systemarchitektur von SomVis3D eingegangen werden.

3.5.1 Überblick

Aus Gründen der Wiederverwendbarkeit ist SomVis3D modular aufgebaut, das heißt, es ist in mehrere Klassenpakete untergliedert. Die wichtigsten Klassenpakete sind die Pakete zur Visualisierung (`somvis3d.visualisation`) und zur Berechnung (`somvis3d.tools`) der SOMs. Durch diese Aufteilung ist es möglich, dem Programm zukünftig auf einfache Weise weitere Visualisierungs- und Berechnungsfunktionen hinzuzufügen. Weitere Klassenpakete von SomVis3D sind `somvis3d.utils`, `somvis3d.interaction` und `somvis3d.help`.

Desweiteren sind in dem Paket `somvis3d` die Klassen `SomVis3D.java` und `SomVis3DApp.java` enthalten. Das Hauptfenster von SomVis3D ist in der Klasse `SomVis3DApp.java` implementiert und stellt den Arbeitsbereich von SomVis3D dar. Dieses wird durch einen entsprechenden Methodenaufruf von `SomVis3D.java` gestartet.
In Abbildung 3.22 ist die Architektur von SomVis3D als Paketdiagramm dargestellt. Aus Gründen der Übersichtlichkeit werden in dieser Abbildung lediglich die für die Visualisierung und Berechnung der SOM relevanten Beziehungen dargestellt.

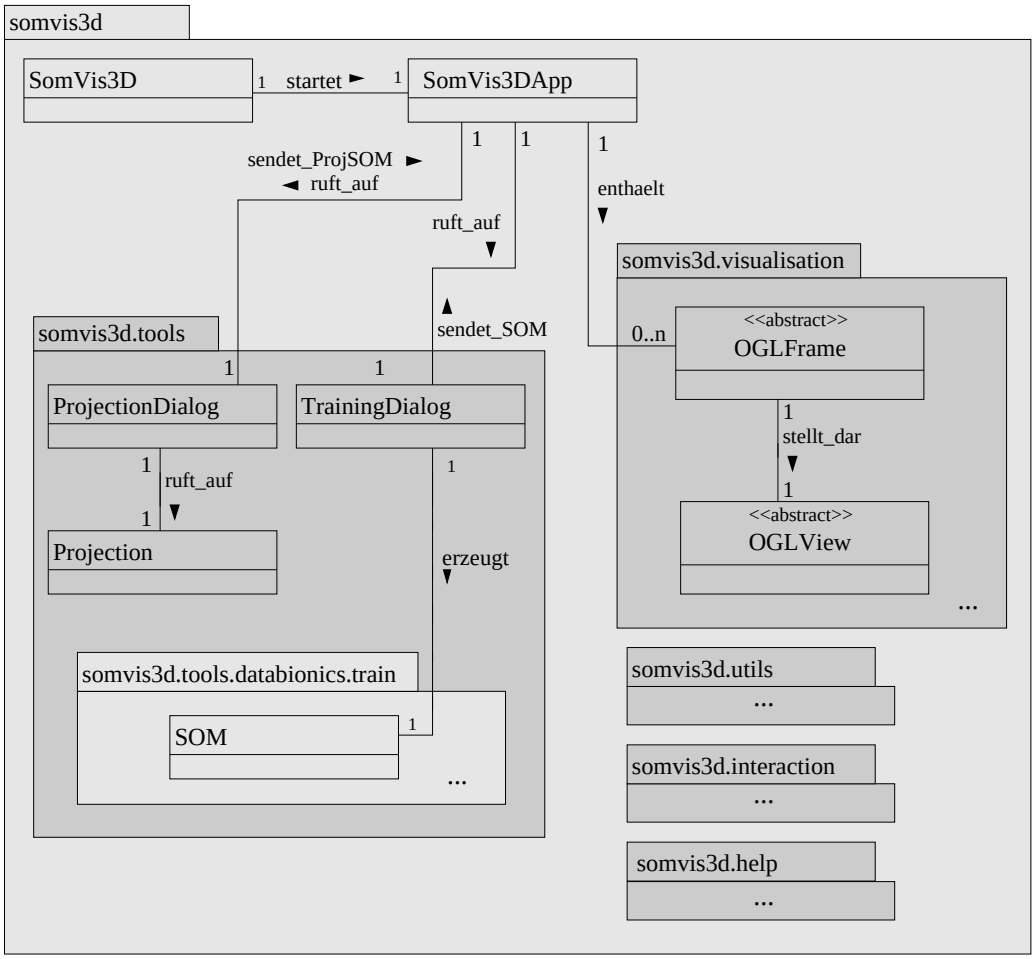


Bild 3.22: Klassenübersicht `somvis3d`

`SomVis3DApp.java` stellt die Verbindung zwischen den Funktionen „Projektion → Visualisierung“ und „Training → Visualisierung“ dar. Diese Verbindung wird durch die Methode `public void render(...)` von `SomVis3DApp. java` realisiert. Diese Methode wird nach einem Training oder einer Projektion aufgerufen. Sie ist zuständig für das Öffnen eines internen Fenster mit der SOM-Visualisierung[11], welche ihr als Parameter übergeben wird.

3.5.2 Die Tools (`somvis3d.tools`)

Überblick

Grob kann in diesem Paket zwischen den Klassen von SomVis3d und den Klassen, welche im Rahmen des ESOM-Projektes (siehe *http://databionic-esom.sourceforge.net*) implementiert worden sind, unterschieden werden. Zuletzt genannte Klassen stellen Funktionalitäten bereit, welche SomVis3D für das Training und die Projektion benötigt und befinden sich in dem Paket `somvis3d.tools.databionics` und. In dieser Arbeit soll nicht weiter auf die ESOM-Klassen eingegangen werden. Weitere Informationen dazu sind unter *http://databionic-esom.sourceforge.net* und *http://www.mathematik.uni-marburg.de/ databionics/de/* zu finden.

Weitere Klassen in `somvis3d.tools` sind `TrainingDialog.java`, `ProjectionDialog.java` und `Projection.java`[12]. Abbildung 3.23 stellt eine Übersicht der Klassen dieses Paketes und die wichtigsten Beziehungen dar.

Auf das Paket `somvis3d.tools.databionics` und die darin enthaltenen Pakete soll – da diese Implementierungen von ESOM enthalten – in dieser Arbeit nicht eingegangen werden. Weiterführende Informationen sind unter *http://databionic-esom.sourceforge.net/* zu finden.

[11] SOM-Visualisierung bezieht sich auf trainierte, projizierte oder aus Dateien geladene SOM. Dabei ist `render(...)` nicht für die Darstellung der SOM zuständig. Ihre Anweisungen beziehen sich lediglich auf das Öffnen eines internen Fensters mit den nötigen Parametern zur Visualisierung. Die Visualisierung selbst ist in `OGLView.java` (siehe Abschnitt 3.5.3) implementiert.

[12] Das Training einer SOM ist nicht in einer eigenen Klasse, sondern als Methode in der Klasse `somvis3d.tools.databionics.train.SOM.java` implementiert.

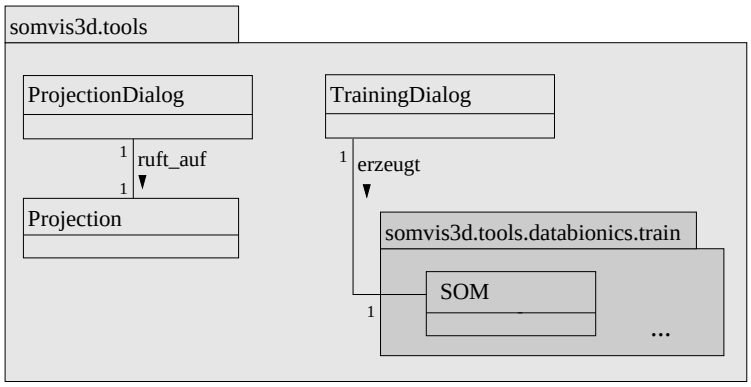


Bild 3.23: Klassenübersicht `somvis3d.tools`

TrainingDialog.java

Die Klasse `TrainingDialog.java` ist von der Klasse `JDialog (javax.swing)` abgeleitet und stellt den Eingabedialog für die benötigten Trainingswerte bereit. Sie ist als Thread implementiert, welcher durch Verwenden des Java Runnable Interfaces erzeugt wurde. `TrainingDia -log.java` übernimmt alle Aufgaben rund um das Training einer SOM:

1. Zuerst wird in dieser Klasse eine Instanz der Klasse `SOM.java`, welche die Merkmalskarte darstellt, erzeugt. Die Klasse `SOM.java` ist im Rahmen des ESOM-Projektes implementiert worden und in SomVis3D in dem Paket `somvis3d.tools. databionics.train` zu finden.

2. Die Trainingsparameter (vgl. Abschnitt 3.5.2) die eine SOM benötigt, um trainiert werden zu können, werden über den Dialog eingelesen und der SOM übergeben.

3. Dann wird die SOM durch eine entsprechende Funktion (zum Beispiel durch Zufallswerte) initialisiert.

4. Nun kann das Training aufgerufen werden. Das Training ist als Methode `public void train()` der Klasse `SOM.java` implementiert. An dieser Stelle soll das Training nicht weiter erläutert werden. Weitere Informationen zum Training sind unter *http:// databionic-esom.sourceforge.net/* zu finden. Die Methode `train()` verändert die Instanz der Klasse `SOM.java`. Diese stellt jetzt die strukturierte SOM dar. Diese, wie auch die Bestmatches werden in den entsprechenden Dateien gespeichert.

5. Abschließend wird die entstandene SOM der `render()`-Methode der Klasse `SomVis-3DApp.java` übergeben.

ProjectionDialog.java

`ProjectionDialog.java` ist, wie die Klasse `TrainingDialog.java` von `JDialog (javax.swing)` abgeleitet und als Thread durch Verwendung des Java Runnable Interfaces implementiert. Sie stellt den Dialog für die Projektion bereit.

Diese Klasse übernimmt die Aufgaben rund um die Projektion einer Learn-Datei auf eine bereits bestehende SOM. Die Projektion selbst wird allerdings durch die Klasse `Projection.java` ausgeführt.

1. Bevor eine Projektion durchgeführt werden kann, wird in dieser Klasse eine Instanz der Klasse `Projection.java` erzeugt. Dem Konstruktor von `Projection.java` werden dabei die zur Projektion nötigen Informationen in Form einer Learn-Datei und einer Weights-Datei übergeben.

2. Durch Aufruf der Methode `public BMFile convert()` der Klasse `Project .java` wird die Projektion gestartet. `convert()` liefert ein Bestmatch-File mit den projizierten Bestmatches zurück.

3. Jetzt kann die SOM, welche durch die Weights-Datei gegeben ist, und die entstandene Bestmatch-Datei an die entsprechende `render()`-Methode der Klasse `SomVis3DApp .java` übergeben werden.

Projection.java

Diese Klasse übernimmt die eigentliche Projektion der Daten einer eingelesenen Learn-Datei auf eine bereits strukturierte SOM. Die für die Projektion relevanten Methoden dieser Klasse wurden von ESOM übernommen. Weitere Informationen zu der Projektion sind unter *http://databionic-esom.sourceforge.net/* zu finden.

3.5.3 Die Visualisierung (`somvis3d.visualisation`)

Überblick

Die beiden Hauptklassen dieses Paketes sind `OGLFrame.java` und `OGLView.java`. `OGLFrame.java` ist die abstrakte Oberklasse für die internen Fenster, welche die Visualisierung der SOM enthalten. Die Visualisierung selbst ist in den Unterklassen der Klasse `OGLView.java` implementiert.

Abbildung 3.24 zeigt eine Übersicht der Klassen des Paketes `somvis3d.visualisation`. Von `OGLFrame.java` leiten sich die Klassen `DevelopmentFrame.java` und `AssemblyFrame.java` ab. Und von `OGLView.java` leiten sich die Klassen `UMatrixView.java`, `SingleComponentsView.java`, `MultiComponentsView.java`, `SelectionView.java` und `AssemblyView.java` ab.

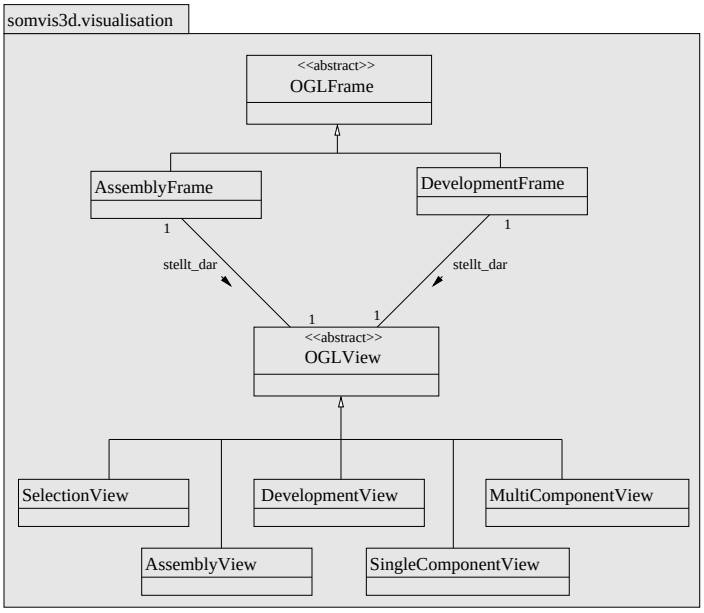


Bild 3.24: Klassenübersicht `somvis3d.visualisation`

OGLFrame

OGLFrame ist die abstrakte Oberklasse für die Erstellung der internen Fenster. Daher leitet sie sich auch von der Java-Klasse `JInternalFrame (javax.swing)` ab. Um auf Benutzeraktivitäten, zum Beispiel das Wechseln zwischen den Fenstern, oder um auf Eingaben durch die Maus (für die Navigation) reagieren zu können, implementiert diese Klasse die Interfaces `InternalFrameListener (javax.swing.event)` und `ComponentListener (java.awt.event)`. So kann zum Beispiel sichergestellt werden, dass die Toolbox (siehe Abschnitt 3.4.4) mit dem zur Zeit aktivierten Fenster referenziert ist.

Folgende Klassen leiten sich von OGLFrame ab:

- `DevelopmentFrame.java`
  Diese Klasse übernimmt die Darstellung einer gerade trainierten SOM, der Komponentenkarten oder einer einzelnen Komponentenkarte.
- `AssemblyFrame.java`
  Diese Klasse ist für die Visualisierung der SOM zuständig, welche aus Dateien heraus geladen wurden. Die Klasse `AssemblyFrame` verfügt über mehrere Konstruktoren, welche als Parameter jeweils die darzustellenden Dateien, zum Beispiel eine U-Matrix- und eine Bestmatch-Datei, übergeben bekommen.

Die wichtigsten Klassenbeziehungen der Klasse `OGLFrame.java` sind in Abbildung 3.25 zu sehen. Das Hauptfenster (`SomVis3DApp.java`) kann mehrere interne Fenster (`OGLFrame.java`) enthalten. In einem `OGLFrame.java` ist immer ein `OGLView.java` dargestellt und jeder `OGLFrame.java` kann bis zu einen InfoFrame (`InfoFrame.java`) besitzen, in welchem die Informationen der visualisierten SOM angezeigt werden.

OGLView

`OGLView.java` ist die abstrakte Hauptklasse für die Visualisierung. In dieser Klasse, sowie in ihren Unterklassen, sind alle OpenGL-Zeichenroutinen zu finden.

Die Unterklassen von `OGLView.java` sind:

- `UMatrixView.java`
  Stellt eine trainierten SOM in U-Matrix-Ansicht dar.

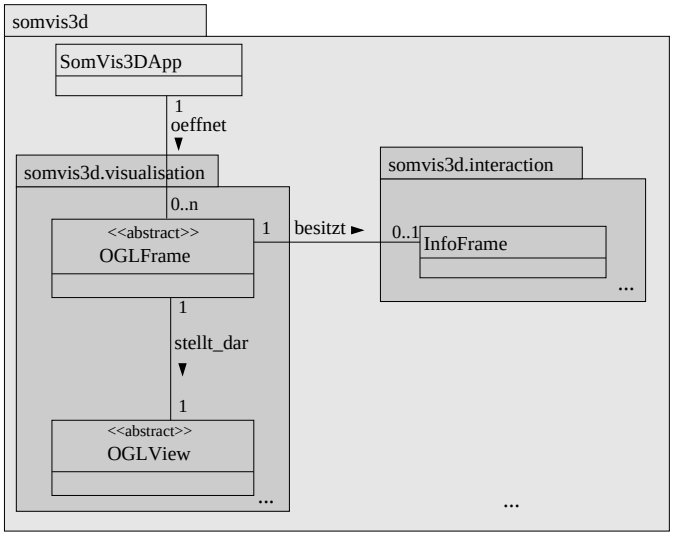


Bild 3.25: Übersicht der Beziehungen von `OGLFrame.java`

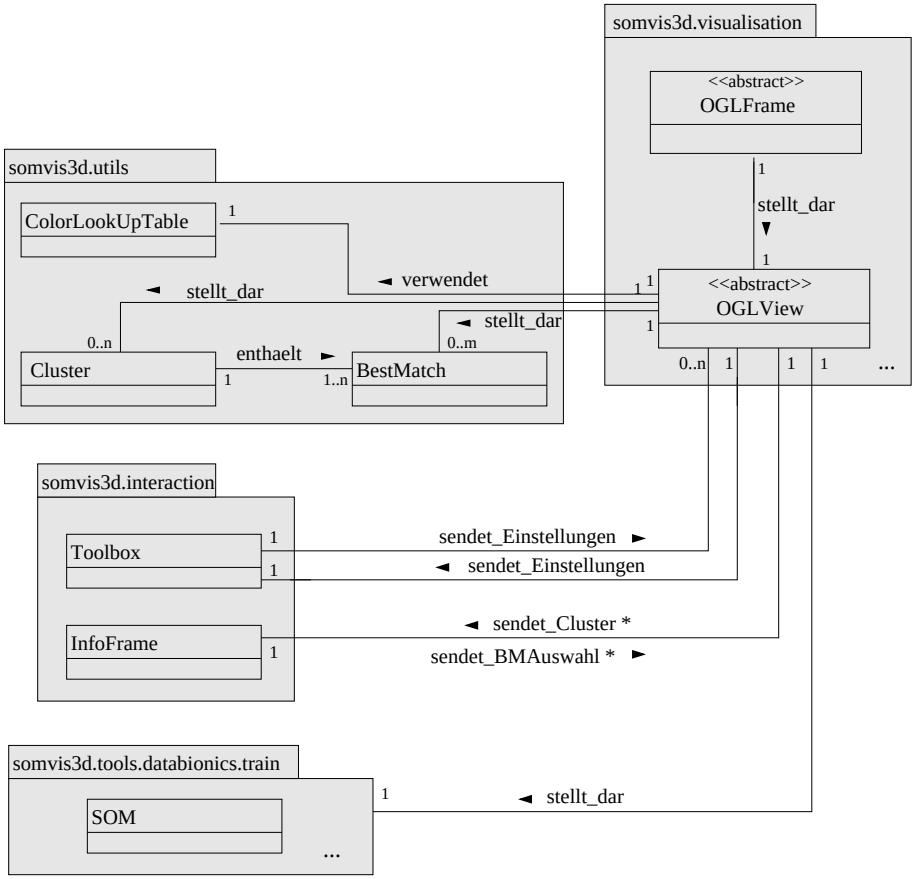
- `MultiComponentView.java`
  Übernimmt die zweidimensionalen Ansicht aller Komponentenkarten.
- `SingleComponentView.java`
  Stellt eine Komponentenkarte in dreidimensionaler Ansicht dar.
- `SelectionView.java`
  Übernimmt die Darstellung der zweidimensionalen Ansicht der SOM und implementiert alle Methoden, die für die Clusterselektion relevant sind.
- `AssemblyView.java`
  Stellt SOM dar, welche in Dateien gespeichert sind.

Alle Unterklassen von `OGLView.java` überschreiben die Methode `void display()`. In `display()` wird die Szene, welche OpenGL auf dem Bildschirm darstellen soll, definiert. Hier werden also die beiden Zeichenmethoden `void drawBackground()` und `void drawForeground()` aufgerufen. `drawBackground()` ist zuständig für die Darstellung der strukturierten SOM und in `drawForeground()` werden die Bestmatches auf der SOM positioniert und die selektierten Bestmatches markiert. (In Abschnitt 3.4.1 wird ausführlich auf diese beiden Methoden eingegangen.)

`SelectionView.java` verfügt zusätzlich über Zeichenmethoden, welche ausschließlich der Funktion der manuellen Clusterselektion dienen:

- `void drawSelectionBox()`
  Diese Methode wird aufgerufen um die Selektionsbox darzustellen.
- `void drawTempSelection()`
  Hier ist die Visualisierung der temporären (weißen) aktivierten Bestmatches implementiert.
- `void fillSelectedBMsVector()`
  In dieser Methode werden die selektierten Bestmatches berechnet und zur Visualisierung für alle Ansichten in einem Java-`Vector` gespeichert.

Abbildung 3.26 zeigt die wichtigsten Klassenbeziehungen von `OGLView.java`. Ein Instanz der Klasse `OGLView.java` wird von `OGLFrame.java` mit den Informationen der darzustellenden SOM erzeugt und dargestellt. `OGLView.java` kann eine SOM (`SOM.java`), Bestmatches (`BestMatch.java`) und Cluster (`Cluster.java`) darstellen. Für die Farbgebung der SOM benötigt `OGLView.java` die Klasse `ColorLookUp-Table.java`. Desweiteren kommuniziert `OGLView.java` mit der Toolbox (`ToolBox.java`), um das Erscheinungsbild der SOM zu verändern, und mit dem InfoFrame (`InfoFrame.java`), um die SOM-Informationen anzeigen zu lassen und im InfoFrame selektierte Bestmatches zu markieren.



* sendet_Cluster bezieht sich auf die manuell selektierten Cluster.
* sendet_BMAuswahl bezieht sich auf die temporaer selektierten Bestmatches im InfoFrame.

Bild 3.26: Übersicht der Beziehungen von `OGLView.java`

Kapitel 4

Experimente und Ergebnisse

In diesem Kapitel sollen die Ergebnisse der SOM-Visualisierung getestet und kurz bewertet werden.

Eine SOM wird in SomVis3D durch ein polygonales Netz aus Dreiecksflächen dargestellt. Die Farbgebung entsteht durch die Interpolation der Farben an den Eckpunkten der Dreiecksflächen. Diese Modellierungstechnik hat den Vorteil, dass sie nicht sehr rechenaufwendig und somit schnell ist. Auch die Darstellungen größerer Oberflächen sollten effizient möglich sein. Doch diesem Vorteil steht der Nachteil gegenüber, dass die Oberflächen, vor allem wenn sie mit weniger Polygonen modelliert wurden, sehr kantig erscheinen können.

Im Hinblick auf diese beiden Punkte, soll die Visualisierung für eine kleine SOM mit den Ausmaßen von 10x15 Neuronen, eine mittelgroße SOM mit den Ausmaßen von 82x50 Neuronen und eine größere SOM mit den Ausmaßen von 300x380 Neuronen untersucht werden.

Für die folgenden Testbilder wurde der Datensatz `hepta.lrn` für unterschiedliche Kartengrößen trainiert. Das Training und die Visualisierung wurden auf folgendem System durchgeführt:

Prozessor: Intel Pentium 4 mit 2,50 GHz
Speicher: 512 MB RAM
Betriebssystem: Windows XP

Eine SOM mit einer Kartengröße von 10x15 Neuronen, wie sie in Abbildung 4.1 zu sehen ist, kann mit relativ wenig Polygonen modelliert werden. Die Oberfläche zeigt deutliche Artefakte in Form

von Kanten zwischen den Polygonen. Die Struktur der SOM ist nur schwer zu erkennen. Aufgrund der sehr kleinen Anzahl an Polygonen ist die Navigation dieser Darstellung sehr flüssig.

Abbildung 4.2 zeigt eine SOM mit einer Kartengröße von 82x50 Neuronen. Im Gegensatz zu der SOM mit der Kartengröße von 15x20 Neuronen erscheint die Oberfläche nicht mehr ganz so kantig und die Struktur der SOM ist sehr gut zu erkennen. Auch in dieser Darstellung kann sehr flüssig navigiert werden.

Abbildung 4.3 zeigt eine im Vergleich zu Abbildung 4.1 und Abbildung 4.2 große SOM. Allein durch die Tatsache, dass der Benutzer die SOM wegzoomen muss, um die Strukturen erkennen zu können, sind die Kanten auf der Oberfläche nicht mehr sehr deutlich. Für die Modellierung dieser SOM ist allerdings eine hohe Anzahl von Polygonen nötig. Durch den dadurch verursachten erhöhten Rechenaufwand kann es bei der Navigation teilweise zu leichten Verzögerungen kommen.

Abschließend kann gesagt werden, dass die Visualisierung für den Datensatz `hepta.lrn` eine mittlere Kartengröße optimal ist. In Abbildung 4.2 ist die Struktur der SOM sehr gut erkennbar. Allerdings kann diese Aussage nicht generell für alle Datensätze gemacht werden. Je nach Datensatz muss entschieden werden, mit welcher Kartengröße die SOM-Struktur am besten erkennbar ist.

Im Allgemeinen ist die Visualisierung einer SOM mit SomVis3D – aufgrund der Kanten bei sehr kleinen SOM und der leicht beeinträchtigten Effizienz bei sehr großen SOM – für eine mittlere Kartengröße optimal.

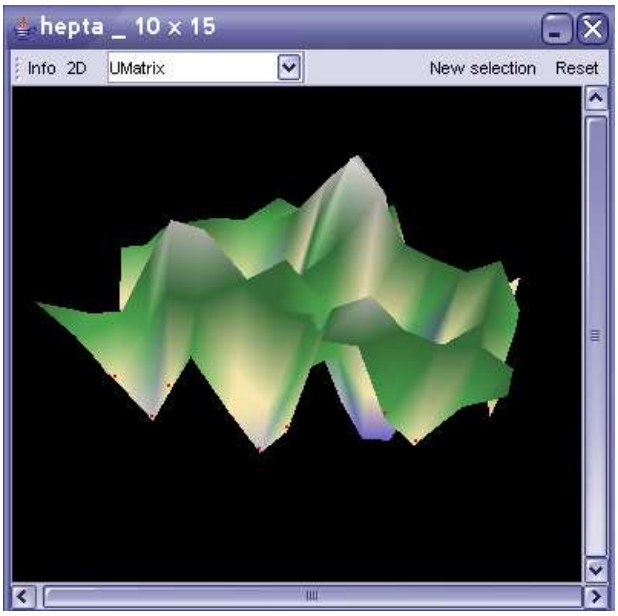


Bild 4.1: U-Matrix von `hepta.lrn` (15x20)

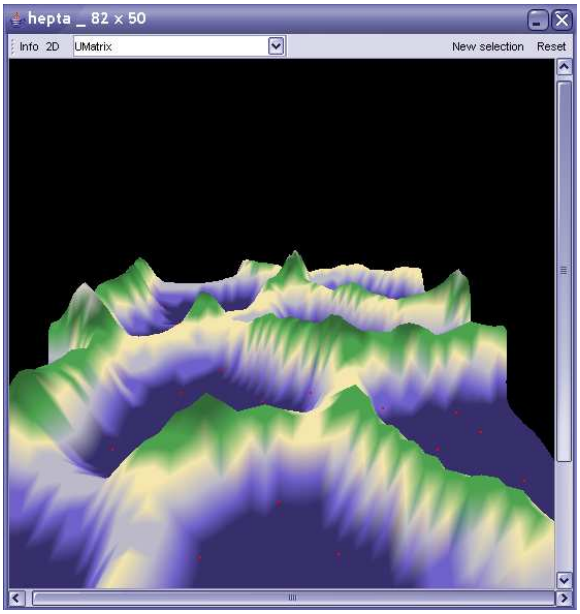


Bild 4.2: U-Matrix von `hepta.lrn` (82x50)

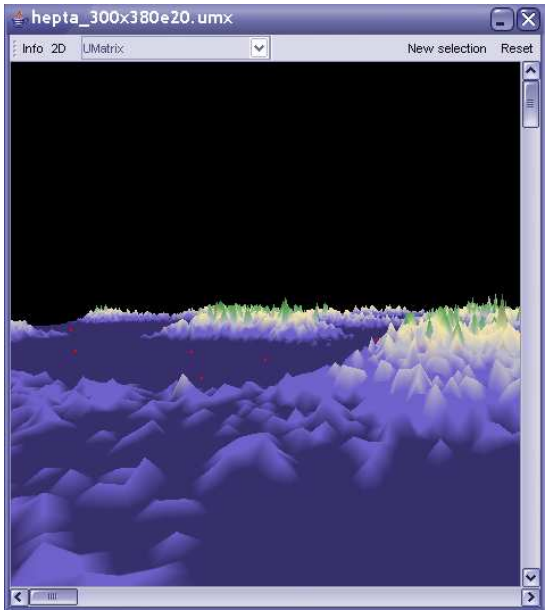


Bild 4.3: U-Matrix von `hepta.lrn` (300x380)

Kapitel 5

Zusammenfassung und Ausblick

Nachdem in diesem Kapitel eine kurze Zusammenfassung dieser Arbeit gegeben wird, werden Weiterentwicklungsmöglichkeiten für SomVis3D vorgeschlagen.

5.1 Zusammenfassung

In den Unternehmen fallen heutzutage täglich eine riesige Menge von Daten an. Die Exploration dieser Datenmengen ist ein sehr wichtiges, aber auch sehr schwieriges Problem, bei welchem Data Mining Methoden helfen können. Durch eine Analyse der Daten mithilfe von Data Mining Methoden können Strukturen zwischen den Daten ermittelt werden, welche für viele Probleme und Fragestellungen eines Unternehmens einen Lösungsansatz bieten oder sich zur Umsatzsteigerung nutzen lassen können. Für ein effektives Data Mining ist es allerdings wichtig, den Menschen in den Datenexplorationsprozess mit einzubinden. Diese Erkenntnis führte zur Entstehung des Visuellen Data Mining (VDM). Die Zahl der Anwendungen, die das VDM für eine verbesserte Datenexploration verwenden, steigt stetig an; das VDM gewinnt an immer größerer Bedeutung [13].

Nach einem einleitenden Kapitel wurde in dieser Arbeit in Kapitel 2 eine umfassende Einführung in die Thematik des Data Mining gegeben. Hier wurde geklärt, was Data Mining ist und warum es so wichtig ist. Dabei wurden die wichtigsten Data Mining Methoden vorgestellt. Zu den bekanntesten Methoden zählt die Clusteranalyse. Nachdem die Vorgehensweise und die Verfahren der Clusteranalyse vorgestellt wurden, wurde ausführlich auf die Selbstorganisierenden Karten (SOM) als ein Verfahren der Clusteranalyse eingegangen. Im Anschluss daran wurde eine Einführung in

das Visuelle Data Mining gegeben. Dabei wurde anhand von zwei Beispielen gezeigt, wie SOM in der Praxis visualisiert werden können.

Im Rahmen der vorliegenden Diplomarbeit wurde ein System entwickelt, welches eine SOM dreidimensional visualisieren kann. Neben der dreidimensionalen Ansicht der SOM als U-Matrix und den Komponentenkarten gibt SomVis3D dem Benutzer die Möglichkeit zur Navigation innerhalb der Szene, zur manuellen Clusterselektion, zur Anzeige verschiedener Informationen über die SOM in einem eigenen Informationsfenster (InfoFrame) und zur Veränderung des Erscheinungsbildes der SOM(Toolbox).
SomVis3D ist in der Lage, sehr kleine wie auch sehr große Datensätze zu trainieren und die entstandene SOM dreidimensional zu visualisieren.
Das System bietet dem Benutzer eine einfache Arbeitsfläche, welche er individuell organisieren kann. Das bedeutet, er kann beispielsweise die Toolbox in einer beliebigen Ecke verstauen und die Fenster so anordnen, wie er es für übersichtlich empfindet. Diese Möglichkeit zur individuellen Organisation der Arbeitsfläche wird durch die alternative Anzeige der SOM-Informationen in einem anderen Fenster, dem InfoFrame, unterstützt. Während der Entwicklung von SomVis3D wurde darauf Wert gelegt, dass das System so entworfen wird, das ein Benutzer sich ohne eine lange Einarbeitungszeit schnell zurechtfindet.

Die Entwicklung stützte sich auf die Anforderungsliste 3.2. In den Abschnitten 3.3 und 3.4 wurden verschiedene Möglichkeiten gezeigt, um die in der Liste beschriebenen Anforderungen umzusetzen. Aus diesen Möglichkeiten wurde jeweils ein Lösungsweg gewählt. Nach einer Beschreibung der Realisierung des gewählten Lösungsweges wird die Systemarchitektur von SomVis3D erläutert. In Kapitel 4 werden die Ergebnisse der Visualisierung bewertet.

5.2 Weiterentwicklungsmöglichkeiten

In SomVis3D wurden alle Anforderungen der Anforderungsliste 3.2 erfolgreich umgesetzt. Trotzdem gibt es Möglichkeiten SomVis3D weiterzuentwickeln und in vielen Bereichen noch zu verbessern. Folgende Vorschläge konnten aus zeitlichen Gründen im Rahmen der Diplomarbeit nicht umgesetzt werden.

5.2.1 Visualisierung der SOM durch Freiformflächen

Da vor allem bei kleinen SOM die Darstellung der Landschaft durch die Modellierung mithilfe eines polygonalen Netzes sehr kantig erscheint, wäre es eine Überlegung wert, die SOM durch Freiformflächen (vgl. Abschnitt 3.4.1) zu modellieren.

Bei der Modellierung der SOM durch Freiformflächen könnte es aber vor allem bei großen SOM zu erheblichen Performanzeinbußen kommen. Daher wäre es wahrscheinlich praktischer dem Benutzer die Modellierung der SOM durch Freiformflächen als eine Alternative zu den polygonalen Netzen anzubieten.

5.2.2 Implementierung weiterer Visualisierungsansichten

Neben den vorgestellten Visualisierungsansichten der U-Matrix und der Komponentenkarten werden in [23] eine Reihe weiterer Ansichten vorgestellt. Durch die Implementierung dieser Ansichten kann SomVis3D weiterentwickelt werden. Desweiteren können neue Ansichten erfunden und implementiert werden, die gerade bei der dreidimensionalen Visualisierung die Struktur einer SOM gut demonstieren könnten. Beispielsweise kann eine verzerrte Darstellung der Distanzen der Neuronen zueinander visualisiert werden, bei welcher große Distanzen sehr groß und kleine Distanzen sehr klein erscheinen.

5.2.3 Weiterentwicklung der manuellen Clusterselektion

In SomVis3D wurde die Clusterselektion durch Bounding Volumes lediglich mit Hilfe eines Rechteckes ermöglicht. Da diese Art der Selektion eine sehr „starre“ Methode darstellt, wäre es eine sehr nützliche und für die Zukunft von SomVis3D auch notwendige Weiterentwicklung, die Selektion auch für frei definierbare Formen zu implementieren.

ESOM verfügt über die Möglichkeit nicht nur Bestmatches, sondern auch Gebiete zu selektieren. Hier wird dann aus allen Bestmatches, welche in einem Gebiet liegen, ein Cluster gebildet. Diese Funktion nennt sich in ESOM „Classification“.
Diese Funktion kann in der Zukunft auch in SomVis3D umgesetzt werden.

Desweiteren kann eine Funktion implementiert werden, welche eine automatische Auswahl von Clustern ermöglicht.

5.2.4 Selektion in 3D durch Bounding Volumes

Eine weitere Weiterentwicklungsmöglichkeit von SomVis3D ist die Clusterselektion innerhalb der 3D-Ansicht durch Bounding Volumes. Diese Selektion wäre für den Benutzer wahrscheinlich ein wenig schwieriger und er bräuchte eine gewisse Übungszeit um gut mit ihr arbeiten zu können, aber er hätte dadurch beispielsweise die Möglichkeit alle Bestmatches einer Höhe auf einmal zu selektieren. Die Selektion auf der 2D-Ansicht könnte auf viele Benutzer ein wenig veraltet wirken.

5.2.5 Weiterentwickelter Zoom

In SomVis3D wir dem Benutzer die Möglichkeit zum zoomen gegeben. Diese Funktion hat hier nur die Bedeutung, dass die Karte grafisch vergrößert wird. Eine Weiterentwicklungsmöglichkeit wäre, dass das Zoomen eine Auswahl eines Ausschnittes der dargestellten SOM ermöglicht. So würde der gewählte Ausschnitt nicht nur vergrößert dargestellt werden, sondern durch zusätzliche Informationen, wie Beschriftungen oder Dokumentendetails bereichert werden.

Diese Veränderung brächte verschiedene Vorteile mit sich:

1. Mehr Informationsgehalt
2. Optimierung der Rechenleistung - dadurch, dass nach dem Zoom nur noch ein Ausschnitt der gesamten Karte dargestellt wird, könnte die benötigte Rechenleistung erheblich optimiert werden.

5.3 Fazit

SomVis3D stellt einen Prototypen eines Systemes zur dreidimensionalen Visualisierung einer SOM dar. Obwohl alle Anforderungen der Anforderungsliste 3.2 erfolgreich umgesetzt wurden, gibt es viele Weiterentwicklungsmöglichkeiten. Das Ziel ist es, dem Benutzer die Informationen so anschaulich wie möglich zu präsentieren und ihm überdies eine Vielzahl von Interaktionsmöglichkeiten zu bieten. Um solch ein System zu erstellen, stellt SomVis3D den ersten Schritt dar.

Literaturverzeichnis

[1] Altran IT. Online im Internet unter: http://www.altran-it.de/, 19.05.2006

[2] Martin, W.: Data Warehousing, DataMining - Olap. An International Thomson Publishing Company, 1998

[3] Fayyad, U.M.: Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. Advances in Knowledge Discovery and Data Mining. AAAI Press Series in Computer Science. A Bradford Book, The MIT Press, Cambridge Massachusetts, London England, 1996

[4] Witten I.H., Frank E.: Data Mining. Praktische Werkzeuge und Techniken für das maschinelle Lernen. Carl Hanser Verlag, München, Wien, 2001

[5] Hotho, Dipl.-Wirsch.-Inform. A.: Clustern mit Hintergrundwissen, 18.08.2004, Online im Internet: http://www.kde.cs.uni-kassel.de/hotho/pub/2004/dissAho.pdf, 20.05.2006

[6] Loss, D.: Data Mining: Klassifikations- und Clusteringverfahren, 15.04.2002

[7] Pfeffer, F.: Einführung in die Clusteranalyse. Online im Internet: http://www.metaanalyse.de/material/re040503.pdf, 23.05.2006

[8] Hebb, D.O.: The Organization of Behaviour. J.Willey & Sons, New York, 1949

[9] Krebs, S.: Selbstorganisierende Karten, Seminar Intelligente Systeme im Finance. Online im Internet: WWW: http://www.aifb.uni-karlsruhe.de /Lehre/Sommer2003/ISF/Seminararbeit_SOM.pdf, 08.09.2005

[10] Bessas, Y.: Selbstorganisierende Karten, Proseminar Neuronale Netze. Online im Internet: http://www.informatik.uni-ulm.de/ni/Lehre/WS04/ProSemNN/pdf/SOM.pdf, 08.09.2005

[11] Kohonen, T.: Self-Organizing Maps. Springer Series in Information Sciences, Berlin, 2001

[12] Prof. Dr.-Ing. Lämmel, U., Prof. Dr. rer. nat. Cleve, J.: Künstliche Intelligenz. Fachbuchverlag Leipzig im Carl Hanser Verlag München Wien, 2001

[13] Keim, A. D.: Datenvisualisierung und Data Mining. Online im Internet: http://fusion.cs.uni-magdeburg.de/pubs/spektrum.pdf, 20.04.2006

[14] Oellien, F.: Algorithmen und Applikationen zur interaktiven Visualisierung und Analyse chemiespezifischer Datensätze, 2003. Online im Internet: http://www2.chemie.uni-erlangen.de/people/Frank_Oellien diss/index.html, 23.04.2006

[15] Wiedenbeck, M., Züll, C.: An International Thomson Publishing Company, 1998

[16] Macho: Modelle des Lernens: Neuronale Netze. Online im Internet: http://www.unifr.ch/spc/UF/93mai/macho.html, 10.08.05

[17] Weiß, J.: Selbstorganisierende Karten, Online im Internet: http://www.informatik.uni-ulm.de/ni/Lehre/SS04/ProsemSC/ausarbeitungen/Weisz.pdf, 13.05.2006

[18] Prof. Mueller, S.: Vorlesung zur Computergrafik2, 2004/2005

[19] WEBSOM - Self-Organizing Maps for Internet Exploration. Online im Internet: http://websom.hut.fi/websom, 20.04.2006

[20] WEBSOM map - Million documents. Online im Internet: http://websom.hut.fi/websom/milliondemo/html/root.html, 20.04.2006

[21] Databionic ESOM Tools. Online im Internet: http://databionic-esom.sourceforge.net, 20.04.2006

[22] Kneips, T.-R.: Interaktive Visualisierung von Groupware-basierten Topic Maps - Konzepte und Prototypenentwicklung.
Online im Internet: http://gcc.upb.de/www/WI/WI2/wi2_lit.nsf/0/b93e32bff9402cfcc1256d20-005217df/FILE/Diplomarbeit_Torsten_Knieps.pdf, 20.04.2006

[23] Ultsch, A., Mörich, F.: ESOM-Maps: tools for clustering, visualisation and classification with Emergent SOM. Data Bionics Research Group, University of Marburg, 17.03.2005

[24] Sun Microsystems, Inc. Online im Internet: http://java.sun.com/, 05.05.2006

[25] Stroustrup, B.: Die C++ Programmiersprache - 2.überarb. Aufl Bonn; München; Paris[u.a]: Addison-Wesley (Deutschland) GmbH, 1992

[26] Minas, M.: Programmierung. Online im Internet: http://www2.informatik.uni-erlangen.de/Lehre/WS200001/AIWProg/PDF/vorl02.pdf, 05.05.2006

[27] Stritzinger, A. M.: Komponentenbasierte Softwareentwicklung. Online im Internet: http://www2.informatik.uni-erlangen.de/Lehre/WS200001/AIWProg/PDF/vorl02.pdf, 05.05.2006

[28] Obermeier, F.: Java3D. Online im Internet: http://graphics.cs.uni-sb.de/Courses/ws9900/cg-seminar/Ausarbeitung/Florian.Obermeier/index.html, 10.05.2006

[29] LWJGL-Lightweight Java Game Library. Online im Internet: http://lwjgl.org/about.php, 16.05.2006

[30] Java OpenGL Tutorial unter Verwendung von gl4Java. Computer Graphics Lab, FH-Stralsund. Online im Internet: http://www.micg.et.fh-stralsund.de/Java3D/gl4java.htm#Einleitung, 16.05.2004

[31] JOGL.info. Online im Internet: http://www.jogl.info, 23.11.2005

[32] Konyha, Z.: Aspects of developing a driving simulation game. Online im Internet: http://www.gris.uni-tuebingen.de/gris/GDV/java/doc/html/german/3.2.4.html, 06.05.2006

[33] Drawing a Bezier Surface Tutorial. Online im Internet: www.sulaco.co.za/tut4.htm, 06.05.2006

[34] Schwall, J.: 3D-Interaktion. Westfälische Wilhelms-Universität Münster, Institut für Informatik. Online im Internet: http://www.schwall.de/dl/20041004_3d-interaktion.pdf, 10.05.2006