Die pixelbasierte Clusterung von Luftaufnahmen im Rahmen von Erosionsuntersuchungen

Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften der Universität Dortmund

Dem Fachbereich Statistik
der Universität Dortmund
vorgelegt von

Matthias Zerbst
aus Hagen

Dortmund 2001

Erstgutachter: Prof. Dr. Wolfgang Urfer
Zweitgutachter: Prof. Dr. Walter Krämer

Tag der mündlichen Prüfung: 07. Dezember 2001

Danksagung

Ich danke Prof. Dr. Walter Krämer, der mir das Stipendium im Graduierten-Kolleg „Angewandte Statistik“ der Universität Dortmund ermöglicht hat, in dessen Rahmen diese Arbeit entstanden ist. Er hat die Forschungen zur Erstellung dieser Dissertation intensiv begleitet.

Ich danke Prof. Dr. Wolfgang Urfer, der mich während meiner Dissertation betreut und viele Impulse gegeben hat, um die Forschung voranzutreiben. Durch sein Engagement sind zahlreiche internationale Kontakte zu Stande gekommen, die mir bei meiner wissenschaftlichen Arbeit sehr geholfen haben.

Ich danke dem Graduierten-Kolleg „Angewandte Statistik“ und der Deutschen Forschungsgemeinschaft (DFG) für die finanzielle Unterstützung und die guten Rahmenbedingungen, die diese Arbeit erst ermöglicht haben.

Des Weiteren danke ich Marco Stolpe für seine Hilfe bei der Implementierung, der in dieser Arbeit vorgestellten Verfahren, in eine Rechnerumgebung.

Ich danke Claudia Reimpell, Olaf Schoffer und Simone Kaufmann für ihre hilfreichen Kommentare und Anmerkungen.

Insbesondere danke ich meinem Freund und Kollegen Lars Tschiersch, der mir in der wissenschaftlichen Arbeit ein guter Partner war und dessen eigene Forschungen auf dieser Arbeit aufbauen.

Inhaltsverzeichnis

Symbolverzeichnis

n	-	Anzahl Pixel des zu clusternden Bildes, bzw. Anzahl zu clusternder Objekte
q	-	Anzahl Klassen für Clusterung
r	-	Anzahl unterschiedlicher Pixel bzgl. ihres Farbwertes des zu Grunde liegenden Bildes
P	-	Wert eines Pixels, hier i. Allg. vektorieller RGB-Farbwert
p_i	-	Wert der i-ten Komponente des Pixels i
C	-	Menge der Klassen der Clusterung
c_i	-	i-ter Cluster der Clusterung
z_i	-	i-ter (Cluster-)Zentroid
W	-	Parameterraum, hier i. Allg. $[0,...,255]^3$
O	-	Menge der zu clusternden Elemente, z. B. $O = \{P_1,...,P_n\}$
n_i	-	Anzahl (unterschiedlicher) Pixel in der Klasse c_i
d	-	Distanz - Vektor beim Maximum Linkage
D	-	Distanz - Matrix beim Maximum Linkage
Q	-	Menge der Indizes der Pixel im Cluster der Zentroide
	-	Indexmenge der Pixel außerhalb des Clusters der Zentroide
m	-	Dimension der Komponenten eines Pixels P, hier i. Allg. drei wegen des RGB-Modells
s	-	Bei neuronalen Netzen: aktueller Lernschritt beim Training Bei Maximum Linkage: Anzahl wie oft dasselbe Maximum bei der algorithmischen Umsetzung der Auswahl der Zentroide bis zum aktuellen Iterationsschritt des jeweiligen Durchlaufs aufgetreten ist
E_B	-	theoretisch bestes Niveau der Clusterung, das durch Bedingung (3.1) impliziert wird
E_Z	-	Eingangsniveau, zu dem die Zentroide Z beim Maximum Linkage ausgewählt wird
d	-	Nachbarschaftsfunktion für das Training der neuronalen Netze
$h(s)$	-	monoton fallende Lernrate beim Lernschritt s
a	-	Faktor der Konvergenz-Geschwindigkeit beim Training der Neuronalen Netze
h_{ij}	-	Gewicht, berechnet aus der Summe der absoluten Häufigkeiten der (unterschiedlichen) Pixel P_i und P_j beim Maximum Linkage
$H(P_i)$	-	Funktion, die die Häufigkeit des Auftretens des Wertes des Pixels P_i im zu Grunde liegenden Bild berechnet
k	-	Varianzbasierter Parameter beim Maximum Linkage
b	-	Konstante aus dem Intervall (0,1), die die Lernrate beim Training neuronaler Netze definiert

Z	-	Menge der Zentroide beim Maximum Linkage
k	-	Index der Komponente eines Pixels
s_{max}	-	Bei neuronalen Netzen: Maximale Anzahl an Lernschritten beim Training Bei Maximum Linkage: Anzahl, wie oft dasselbe Maximum bei der algorithmischen Umsetzung der Auswahl der Zentroide beim aktuellen Durchlauf insgesamt auftritt
O	-	Menge zu clusternder Objekte
N_j	-	j-tes Neuron eines neuronalen Netzes
q_j	-	Schwellenwert des j-ten Neurons
net_j	-	Eingabewert des j-ten Neurons
f_i	-	Aktivierungsfunktion der i-ten Eingabe eines Neurons
w_{ij}	-	Gewicht der i-ten Eingabe für das j-te Neuron
q_{ROW}	-	Seitenbreite der Gitterstruktur eines SOMs
zk	-	Anzahl Zyklen für das Training neuronaler Netze
zks	-	Nummer des aktuellen Trainingszyklus, läuft von 1 bis zk
ar	-	Adaptionsradius für das Training neuronaler Netze
ah	-	Adaptionshöhe für das Training neuronaler Netze
$dist$	-	Je nach Methode (WTAN oder SOM) Abstand zwischen den Gewichten der Neuronen oder Abstand zwischen den Neuronen bzgl. ihrer Lage auf der Kohonenkarte
rad	-	Radius des Kreises, der das Hexagon umschließt, das bei der Herleitung des zweidimensionalen, regelmäßigen, hexagonalen Neuronengitters benötigt wird
h	-	Vertikaler Abstand zwischen den Zeilen eines zweidimensionalen, regelmäßigen, hexagonalen Neuronengitters
$init$	-	Initialisierungsschranke für die Initialisierung der Neuronen
$\lceil \cdot \rceil$	-	Obere Gaußklammer: Der enthaltene Wert wird auf die nächstgrößere ganze Zahl gerundet
$\lfloor \cdot \rfloor$	-	Untere Gaußklammer: Der enthaltene Wert wird auf die nächstkleinere ganze Zahl gerundet
G	-	Die Menge aller möglichen Clusterungen der Elemente $P_1,...,P_n \in \boldsymbol{R}^m$
S	-	Die Menge aller möglichen Zentroide zu allen möglichen Clusterungen G
P	-	Auswahlwahrscheinlichkeit eines mehrfach auftretenden Maximums beim Maximum-Linkage-Verfahren
mod	-	Operator, der bei ganzzahliger Division den ganzzahligen Anteil liefert
div	-	Operator, der bei ganzzahliger Division den Rest liefert
$\lvert \cdot \rvert$	-	Ist das Argument eine Menge, so wird durch $\lvert \cdot \rvert$ die Kardinalität der Menge bezeichnet
id	-	Identität: $f(x) = x$

Einleitung

In dieser Arbeit werden Verfahren zur Clusterung von Bilddaten betrachtet. Eine Clusterung teilt eine Menge von Elementen in möglichst homogene Gruppen ein. Bei den Bilddaten handelt es sich insbesondere um Luftaufnahmen aus ökologischen Anwendungsbereichen. Die Informationen solcher Aufnahmen sind zu umfangreich, um in weiteren Analysen sofort verwendet werden zu können. Daher werden Clusterverfahren entwickelt und angewendet, um die Bildinformationen auf ein handhabbares Maß zu reduzieren.

Die Idee zu dieser Arbeit entstand bei der Analyse und Erforschung von Erosionsphänomenen in Nordafrika, wie sie in Arbeiten von Tschiersch (1998) und Zerbst (1999) behandelt werden. In diesen Arbeiten wird die Entwicklung der Erosion anhand von Zeitreihen für Bodendaten untersucht. Bei den dort analysierten Daten handelt es sich um Beobachtungen über den Bodenverlust in Ausläufern des Atlasgebirges im Norden Marokkos. Diese Größen geben Aufschluss über die zeitliche Entwicklung der Erosion. Sowohl der geringe zeitliche als auch der kleine räumliche Umfang dieser Bodendaten führten bei der Analyse zu unlösbaren Schwierigkeiten.

Die Grundidee zur Lösung dieses Problems ist die Verwendung von Satellitenbildern oder Luftaufnahmen, um die Datensituation zu verbessern. Satellitenbilder sind heute leicht von nahezu jeder Region der Erde zu vielen Zeitpunkten zu bekommen. Außerdem lassen sich damit Gebiete von beliebiger Größe in den verschiedensten Auflösungen überwachen. Um solche Bilder in Analysen verwenden zu können, müssen diese zunächst aufbereitet werden. Die Informationen aus diesen Bildern müssen erst herausgearbeitet werden. Dazu wird aus einem Satellitenbild eine schematische Karte erstellt. Diese Karten bestehen aus vielen einzelnen Kategorien, die bzgl. der Erosionsproblematik Flächen mit unterschiedlichen Erosionsgraden darstellen. Die Bilddateien bestehen i. Allg. aus vielen Millionen Pixeln und tausenden von unterschiedlichen Farben. Jede einzelne Farbe steht hier für eine Klasse oder Kategorie. Diese Kategorien müssen stark reduziert werden. Dieses Vorgehen wird als Clusterung des Bildes bezeichnet. Im hier vorliegenden Fall werden einige besondere Anforderungen an die zu verwendenden Verfahren gestellt. Die mittels Clusterung bearbeiteten Bilder können dann als Basis dienen, um zu den ermittelten Klassen gezielt Bodendaten zu erheben. Diese Bodeninformationen sollen auf den ganzen Bereich der entsprechenden Klasse erweitert werden. Mit wenig Aufwand vor Ort stehen dadurch umfangreiche Daten für weiterführende Analysen zur Verfügung. Anhand der so entstandenen Daten ist es möglich, umfangreiche neue Erkenntnisse über Probleme der Erosion und Wüstenausbreitung zu erarbeiten. Weitere Anwendungsmöglichkeiten für die entwickelten Cluster-Methoden werden im Ausblick vorgestellt.

Wie beschrieben, enthalten die für eine Erosionsanalyse zur Verfügung stehenden Ausgangsbilder eine nicht interpretierbare Vielzahl an (Farb-)Informationen. Diese wird mit Hilfe der Cluster-Verfahren auf ein überschaubares Maß reduziert. Dazu

wird ein neues Verfahren vorgestellt, dem ich den Namen *Maximum Linkage* gebe. Des Weiteren werden alternative Clusterungsmethoden betrachtet, die auf neuronalen Netzen basieren. Dazu werden neue Funktionen entwickelt, die zur Anpassung der neuronalen Netze an die Clusterung von Bilddaten aus der Erosionsforschung nötig sind. Dadurch liefern beide Verfahrenstypen besonders gute Clusterungsergebnisse für die Weiterverwendung in der Erforschung der Wüstenausbreitung und Erosionsproblematik. Schließlich werden auch neue Methoden zur Ergebnisbeurteilung und Ergebnisdarstellung eingeführt.

Ein Clusterungsverfahren hat in diesem Zusammenhang das Problem zu lösen, die Pixel eines Bildes mit vielen tausend verschiedenen Farben in wenige Gruppen (Cluster) einzuteilen. Die Einteilung soll die Informationen des Urbildes durch wenige Klassen gut wiedergeben. Bei der Erosionsforschung müssen zusätzlich zur homogenen Aufteilung der Klassen auch seltene Elemente bzw. Objekte gefunden werden. Die seltenen Elemente sind in diesem Zusammenhang wichtig, da sie oft einen großen Einfluss auf die Erosion haben. Man denke zum Beispiel an eine schmale Hecke um ein Feld oder einen einzelnen Baum, Strauch oder Felsen in der Wüste (Banzhaf, 1989). Außerdem muss eine deutliche Trennung zwischen den Klassen bestehen. Die deutliche Trennung zwischen den Klassen ist aus zwei Gründen nötig. Zum einen sollen die identifizierten Klassen u. a. unterschiedlich stark erodierte Flächen anzeigen. Wenn eine deutliche Trennung zwischen den Klassen vorhanden ist, lassen sich die verschiedenen Stufen der Erosion besser erkennen. Außerdem sollen später Bodendaten zu den Klassen im Feld erhoben werden. Wenn sich die Klassen deutlich unterscheiden, ist es leichter, die verschiedenen Bereiche bei der Erhebung von Bodendaten im Feld zu identifizieren.

Die klassischen Cluster-Verfahren, wie z. B. das ISODATA-Programm von Tou und Gonzales (1974), beruhen im Wesentlichen auf dem k-Mittelwert-Algorithmus (k-means). Davon ausgehend wurden viele Weiterentwicklungen betrieben, um die Probleme und Nachteile dieses Algorithmus zu beheben. Darunter fallen auch Ansätze wie die Hyperklumpen-Strategie von Kelly und White (1993). Die zunächst letzte Entwicklung in der langen Reihe der Verbesserungen stellt der Pixel-Hypercluster-Approximating-Spatial-Ensembles-Ansatz (PHASE) von Myers, Patil und Taillie (1997a) dar.

Alle diese Ansätze weisen jedoch drei wesentliche Nachteile auf: lange Rechenzeiten bei der praktischen Umsetzung, Verlust seltener Elemente und keine scharfe Trennung zwischen den Klassen.

Die in dieser Arbeit neu vorgestellten Verfahren Maximum Linkage und die speziell angepassten neuronalen Netze beruhen nicht auf dem k-Mittelwert-Verfahren. Sie lösen die oben beschriebenen Probleme zum Teil ganz oder sind so abgestimmt, dass sie die Probleme deutlich reduzieren. Bei diesen Verfahren wird ausgenutzt, dass sich ein Clusterungsproblem auf die Ermittlung von Startwerten (Zentroiden) für eine Klassifizierung reduzieren lässt. Dadurch werden bei der praktischen Umsetzung besonders kurze Laufzeiten der Algorithmen erzielt.

Die Bilddaten-Formate werden in Kapitel 1 beschrieben. Diese liefern die Datenbasis für die Clusteralgorithmen. In Kapitel 2 werden einige grundlegende Prinzipien zur Clusterung und Klassifikation vorgestellt. Ferner wird ein Überblick über die klassischen Clusterverfahren gegeben. In Kapitel 3 und 4 wird die Theorie zum Maximum-Linkage-Verfahren und zu den verwendeten neuronalen Netzen präsentiert. Anschließend werden die für die konkrete Programmumsetzung notwendigen Details beschrieben. Methoden zur Beurteilung und zum Vergleich von Clusterungen werden in Kapitel 5 und 6 beschrieben. Anschließend folgt der Vergleich der Algorithmen auf empirischer Ebene (Kapitel 7). Als wichtiges Element bei der praktischen Analyse von Bilddateien werden außerdem Verfahren zur Visualisierung der hier vorliegenden Clusterungen beschrieben. Diese besondere Ergebnispräsentation ermöglicht dem Anwender einen tieferen Einblick in die Leistungsfähigkeit der Algorithmen und erleichtert dem Substanzwissenschaftler die Interpretation. Zum Abschluss folgen Beispiele und empirische Vergleiche sowie der Ausblick auf weiterführende Projekte, in deren Rahmen die vorgestellten Algorithmen Anwendung finden werden.

1 Die Formate der zu Grunde liegenden Daten

In diesem Kapitel werden die Grundlagen und Formate der Ausgangsdaten vorgestellt. Im ersten Teil wird auf die Herkunft und Gewinnung der Luftaufnahmen eingegangen. Der zweite Teil befasst sich mit der Umwandlung in ein Dateiformat, welches zur rechnerunterstützten Clusterung verwendet werden kann. Als geeignetes Dateiformat wird hier das Bitmap-Format (BMP) gewählt. Abrundend werden einige grundsätzliche Anmerkungen zu Farbdarstellungsmethoden und zur Farbmischung gemacht.

1.1 Luftaufnahmen und Satellitenbilder

Die Notwendigkeit der Verwendung von Luftaufnahmen und Satellitenbildern in der Erosionsuntersuchung wird, wie schon erwähnt, auch in Arbeiten von Tschiersch (1998) und Zerbst (1999) vorgestellt. In diesen Studien wurden neben der Zielgröße des Bodenverlustes zusätzlich weitere Daten erhoben, von denen ein Einfluss auf die Erosion vermutet wird. Diese beinhalten Informationen über den Bewuchs, die Bodenbeschaffenheit und weiterer Parameter der Untersuchungsgebiete. Bei der Erhebung dieser Daten treten drei große Probleme auf. Zum einen stehen nur Daten aus einem relativ begrenzten Gebiet zur Verfügung, da keine umfangreichen Installationen zur Messung der Daten in den Beobachtungsgebieten vorhanden sind. Zum Zweiten wurden die Daten von örtlich ansässigen Laien (z. B. Bauern und Hirten) erhoben, so dass eine gute Qualität der Daten nicht sichergestellt ist. Zum Dritten sind die Zeitreihen über den Bodenverlust relativ kurz. Dies führt bei der Anwendung statistischer Verfahren wie der Varianzanalyse und verschiedenen zeitreihenanalytischen Untersuchungen dazu, dass die statistische Signifikanz in vielen Fällen nicht nachgewiesen werden kann. Für weitere und komplexere Analysen reicht die Datenmenge und -qualität nicht aus. Die beschriebene Situation ist kein Einzelfall, wie Kötting et al (1998) zeigt, so dass es für die Erforschung und Untersuchung von Erosionsphänomenen einer erheblichen Verbesserung im Bereich der Datenmenge und -qualität bedarf. Die Lösungsidee besteht in der Verwendung von Luftaufnahmen und Satellitenbildern als Datenquelle.
In vielen Studien, wie sie z. B. von Urfer et al (1994), Talbi (1993) und Bürkert et al (1996) durchgeführt wurden, werden Luftaufnahmen bereits als Datenquelle verwendet. Luftaufnahmen werden mit Hilfe von Kameras aufgenommen, die an Drachen, Ballons oder Kleinflugzeugen montiert sind. Als Kamera werden entweder herkömmliche Fotokameras oder Digitalkameras verwendet. Bei der herkömmlichen Kamera muss der belichtete Film entwickelt und ggf. Bildabzüge erstellt werden. Anschließend sind die Negative oder Bildabzüge mittels eines Scanners zu digitalisieren. Erst dann können die Bilddaten vom Rechner verarbeitet werden. Die Daten der Digitalkamera können direkt in einen Rechner eingelesen werden.

Bei Satellitenbildern ist es deutlich komplizierter, die Bilddaten in den Rechner einzulesen. Satelliten nehmen mit Hilfe von speziellen Sensoren Teile des elektromagnetischen Spektrums auf. Dabei teilen sie den sichtbaren und infraroten Bereich des Spektrums in sogenannte Bänder auf. Jedes Band wird von einem eigenen Sensor aufgenommen. Aus diesen Sensordaten werden die Satellitenbilder durch Addition der Bänder generiert. Genaueres dazu siehe NASA (1997) und Talbi (1999).

1.2 Die RGB-Bilddarstellung

Bei der Betrachtung von Farben gibt es unterschiedliche Aspekte: den physiologischen, den psychologischen und den technischen Aspekt. Je nachdem, welchen Aspekt man betrachtet, gibt es unterschiedliche Farbdarstellungs-Modelle. Die folgenden Betrachtungen beschränken sich lediglich auf den technischen Aspekt der Farbdarstellung.
Basierend auf der Farbwahrnehmungsfähigkeit des menschlichen Auges, ist davon auszugehen, dass sich jede Farbe als Mischung der drei Grundfarben Rot, Grün und Blau darstellen lässt. Das menschliche Auge besitzt nur drei verschiedene Rezeptortypen auf der Netzhaut, die das Farbsehen ermöglichen. Je ein Rezeptortyp reagiert dabei genau auf eine der Grundfarben Rot, Grün oder Blau. Alle Farben, die wir als Menschen wahrnehmen können, werden dann vom Gehirn bei der Verarbeitung der an den drei Rezeptortypen im Auge anliegenden Reize „zusammengemischt".
Um diesen biologischen Aspekt durch ein technisches Modell zu realisieren, ist zunächst eine Standardisierung der Mischung der Farben nötig. Dazu legte die Commision Internationale de L'Eclairage (CIE) die Zahlenwerte X, Y und Z fest. Diese Grundwerte basieren auf der von den Farben angestrahlten Lichtenergie. Zum anderen wurden standardisierte Werte definiert, die Farbe, Helligkeit und Sättigung repräsentieren. Jede Farbe kann durch die standardisierten Werte x, y und z beschrieben werden, die auf den Bereich zwischen 0 und 1 beschränkt sind. Der Zusammenhang zwischen den lichtenergetischen Grundwerten und der Standardisierung ist wie folgt gegeben (Wyszecki & Stiles, 1982):

$$x = \frac{X}{X+Y+Z}\ ,\ y = \frac{Y}{X+Y+Z}\ ,\ z = \frac{Z}{X+Y+Z}\ . \qquad (1.1)$$

Man beachte, dass in dieser Form der Darstellung $z = 1 - x - y$ ist, und damit eigentlich nur die Koordinaten (x, y) zur Farbdefinition benötigt werden. Auf Grund dieser Definition der CIE kann jede Farbe als Kombination aus den drei Grundwerten bestimmt werden. Die Rücktransformationen zu (1.1) sind unter gewissen Bedingungen auch möglich, so z. B. wenn die Helligkeit bekannt ist, was in der CIE-Definition dem Y Wert entspricht (Wyszecki & Stiles, 1982):

$$X = x\frac{Y}{y}, \qquad Z = z\frac{Y}{y}.$$

Um das CIE-Modell zu komplettieren, muss noch eine spezifische Charakteristik ergänzt werden, die es ermöglicht, die Farbe Weiß festzulegen. Üblicherweise wird dazu der Punkt *x = 0,313, y = 0,329, z = 0,358* gewählt. Dieser Wert hat seinen Ursprung in der Physik und entspricht dem Weiß eines bis zur Weißglut bei 6500 Kelvin erhitzten schwarzen Körpers.
Für viele Anwendungen ist dieses Modell zu unhandlich. Ein Modell, das mehr dem biologischen Vorbild des menschlichen Auges entspricht, ist sinnvoller. Solch ein Modell ist das RGB-Modell. Der Name RGB leitet sich von den Farben Rot, Grün und Blau ab. Bei diesem Modell wird für jede Farbe der jeweilige Rot-, Grün- und Blauanteil Farbe angegeben. Mit der folgenden Umrechnung erhält man aus dem CIE-Modell das RGB-Modell (Wyszecki & Stiles, 1982):

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 2.741 & -1.147 & -0.426 \\ -1.118 & 2.028 & 0.034 \\ 0.137 & -0.332 & 1.105 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}. \qquad (1.2)$$

Auch die Rücktransformation dieser Umrechnung vom RGB- auf das CIE-Modell ist anhand der folgenden Umrechnung möglich:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.478 & 0.299 & 0.175 \\ 0.263 & 0.655 & 0.081 \\ 0.020 & 0.160 & 0.908 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}.$$

Mit der Transformation (1.2) lässt sich ein Farbraum definieren, der durch ein dreidimensionales, kartesisches Koordinatensystem repräsentiert wird. Die Einträge sind auf ein endliches Intervall beschränkt. Die Dimensionen *R*, *G*, *B* repräsentieren die drei Grundfarben, wobei *R* für den Rot-, *G* für den Grün- und *B* für den Blauanteil einer Farbe stehen.
Damit sind nun die Grundlagen erklärt, um das Bitmap-Bildformat vorzustellen, das als Datenbasis für die Clusteralgorithmen dient.

1.3 Das Bitmap-Format

Das Bitmap-Format ist ein einfaches Format, um Bilddaten in einer Rechnerumgebung zu speichern bzw. zu verwalten. Das Bild wird dabei als *Rasterbild* abgespeichert. Bei einem Rasterbild wird das Bild in einzelne Bildeinheiten (Pixel) zerlegt

und in einem orthogonalen, äquidistanten Koordinatensystem organisiert abgespeichert (Chou, 1997). Im Folgenden wird ein orthogonales, äquidistantes Koordinatensystem als regelmäßig bezeichnet. Pixel sind kleine quadratische Flächen, die in einem einzigen Farbton eingefärbt sind. Durch die Anordnung der Pixel in dem regelmäßigen Koordinatensystem ergibt sich das Gesamtbild. Die Kanten der Pixel ergeben das Raster des Rasterbildes. Jedes Pixel enthält Informationen über seine Farbe, bzw. seine Farbzusammensetzung auf Grund seiner Rot-, Grün- und Blauanteile nach dem RGB-Modell und seiner zweidimensionalen Lage in dem Bild. Die Lage des Pixels ist dabei nur indirekt über die Reihenfolge gespeichert. Eine Bitmap-Datei besteht typischerweise aus einem Kopf, der unter anderem Informationen über Breite und Höhe des Bildes, also Anzahl der Pixel in der Vertikalen und Horizontalen, enthält, gefolgt von einer Reihe binär codierter Farbinformationen der einzelnen Pixel. Es werden in diesem Format immer nur rechteckige Bilder bzw. Grafiken abgespeichert. Die Farbinformationen über die einzelnen Pixel werden einfach aneinander gereiht abgespeichert. Die Abb. 1.1 zeigt für einen kleinen Bildausschnitt, wie die Darstellung dieses Bereichs mittels Pixeln erfolgt. Zur besseren Verdeutlichung wird dabei in der Vergrößerung ein Raster für die einzelnen Pixel über den Bildausschnitt gelegt.

Abb. 1.1: Vergrößerung eines Ausschnitts eines typischen Bildes zur Verdeutlichung der Pixel-Darstellung des BMP-Formats

Die Anzahl darstellbarer Farben ist beim BMP-Format nicht grundsätzlich fest, sondern wird vor der Speicherung eines Bildes festgelegt. Durch diese Flexibilität bei der Anzahl darstellbarer Farben, kann die Qualität eines Originalbildes in vollem Umfang erhalten werden (Cass, 1999). Die Festlegung des Farbumfangs steht in engem Zusammenhang mit dem Speicherplatz, den man für die Farbinformation je Pixel bereitstellen kann. Je mehr Bits als Speicherplatz für die Farbinformation je Pixel zur Verfügung stehen, desto höher ist die darstellbare Farbtiefe, d. h. die Anzahl möglicher Farben in einem Bild. Die folgende Tab. 1.1 zeigt die maximal mögliche Farbtiefe in Abhängigkeit vom Speicherformat des Bitmap-Typs.

Bitmap-Typ	**Maximale Farbtiefe (-anzahl)**
Schwarz/Weiß (S/W); Monochrom	2
4 Bit pro Pixel	16
8 Bit pro Pixel	256
24 Bit pro Pixel	16777216

Tab. 1.1: Zusammenhang zwischen Speicherbedarf und Farbtiefe

2 Prinzipien des Clusterns

Dieses Kapitel gibt zunächst einige theoretische Grundlagen zu den Begriffen Clusterung und Klassifikation. Im Weiteren werden klassische Verfahren zur pixelbasierten Clusterung präsentiert. Dabei werden auch die Nachteile dieser Verfahren erläutert und die daraus resultierenden Anforderungen an verbesserte Verfahren vorgestellt.

2.1 Theoretische Grundlagen

Anhand der Grundbegriffe wie Clusterung, Klassifikation und überwachten bzw. unüberwachten Lernverfahren, wird in die Theorie des Clusterns eingeführt. Dabei ist es wichtig, zwischen Clusterung und Klassifikation zu unterscheiden. Es sei zunächst der Begriff der Clusterung in enger Anlehnung an Bock (1998) betrachtet. Wie schon erwähnt soll eine Clusterung eine Menge von Elementen in möglichst homogene Gruppen einteilen.
Sei $\Omega \subseteq \mathbb{R}^m$ der Merkmalsraum und $O = \{ P_1,...,P_n \}$, $P_i \in \Omega$, $i = 1,...,n$ eine Menge vektorieller Werte. Gesucht ist eine Clusterung $C(O)$ der Werte $P_1,...,P_n$ aus O. Im Falle der Bildclusterung im RGB-Format ist $\Omega = [0,1,...,255]^3$. Die Menge O umfaßt alle n Pixel des Bildes. Die $P_i, i = 1,...,n$ beschreiben die RGB-Vektoren der Pixel. Die Klassenanzahl q, in die die Daten aufgeteilt werden sollen, ist a priori festzulegen. Formal stellt sich eine Clusterung gemäß

$$C(O) = \{c_1,...,c_q\} \quad \text{mit} \quad c_i = \{P_{i1},...,P_{i\,n_i}\},\ i = 1,...,q \qquad (2.1)$$

dar. Damit eine Clusterung sinnvoll ist, sind zwei zusätzliche Anforderungen an die Parameter q und n_i zu stellen:

1. $n_i > 0$, $i = 1,...,q$, damit alle Klassen besetzt sind und
2. $q \ll n$, damit durch die Clusterung eine deutliche Informationsreduktion erreicht wird.

Zu den q Klassen aus (2.1) werden zusätzlich Klassenrepräsentanten gesucht. Die Klassenrepräsentanten $z_1,...,z_q \in R^m$ dienen zur Beschreibung der Klassen und werden bei weiterführenden Anwendungen benötigt.
Die algorithmische Umsetzung von Clusterverfahren erfolgt i. Allg. auf Grund von rekursiven Verfahren. Bei solchen rekursiven Berechnungen entstehen zwischenzeitlich vorläufige Klassenrepräsentanten. Solche vorläufigen Klassenrepräsentante werden als Zentroide bezeichnet. Am Ende dieser Clusteralgorithmen werden die Klassenrepräsentanten aus den Zentroiden berechnet.

Welchen Anforderungen muss die Clusterung C aus (2.1) genügen, damit sie als homogen bezeichnet werden kann? Wie Bock (1998) darstellt, sollte die Clusterung C dem Intra-Class-Varianz-Clusterungskriterium $g_n(C)$ genügen. Dieses Kriterium wird auch als SSW-Clusterungskriterium bezeichnet. Dabei steht SSW für „Sum of Squares Within“. Das deutet darauf hin, dass die Intra-Class-Varianz auf der Summe der mittleren quadratischen Abweichung der Werte innerhalb der Klassen basiert.
Sei G die Menge aller möglichen Clusterungen der n Elemente $P_1,...,P_n \in \mathbb{R}^m$. Eine homogene Clusterung genügt folgender Bedingung:

$$g_n^{\,1}(C) := \frac{1}{n}\sum_{i=1}^{q}\sum_{k \in c_i}\left\| P_k - \overline{P}_{c_i}\right\|_2 \rightarrow \min_{C \in \Gamma} \ . \qquad (2.2)$$

Die Intra-Class-Varianz ist eine Kenngröße für die Variabilität innerhalb der Klas–sen. Sie bestimmt sich, indem die Varianz in allen Klassen berechnet und über alle diese Größen gemittelt wird. Formal stellt sich das wie in Formel (2.2) ohne die Minimierungsbedingung dar. Für die späteren Anwendungen von (2.2) sind noch andere Darstellungen interessant. Bock (1974) zeigt die Äquivalenz von (2.2) bis (2.4). Sei dazu S die Menge aller möglichen Zentroide bezüglich aller möglichen Clusterungen G.

$$g_n^{\,2}(C\,,\,Z) := \frac{1}{n}\sum_{i=1}^{q}\sum_{k \in c_i}\left\| P_k - z_i \right\|_2 \rightarrow \min_{C \in \Gamma,\, Z \in \Sigma}\,, \qquad (2.3)$$

$$g_n^{\,3}(Z) := \frac{1}{n}\sum_{i=1}^{n}\min_{1 \le j \le q}\left\{ \left\| P_i - z_j \right\|_2 \right\} \rightarrow \min_{Z \in \Sigma} \ . \qquad (2.4)$$

Von den drei äquivalenten Darstellungen des SSW-Clusterungskriteriums ist beson-ders (2.4) interessant. Das liegt daran, dass sich Clusterungen einer Datenmenge in der Praxis dadurch berechnen lassen, dass zunächst aus den vorliegenden Daten Zen-troide bestimmt werden. Anschließend werden alle zur Clusterung anstehenden Werte gemäß einer vorgegebenen Norm zu diesen Zentroiden klassifiziert. Dadurch entsteht eine Clusterung.
Welche spezielle Norm dabei Anwendung findet, ist zunächst nebensächlich. Kaballo (1997) zeigt die Äquivalenz aller Normen auf $\mathbb{R}^m$ im topologischen Sinne. Dennoch können unterschiedliche Normen auch zu unterschiedlichen Ergebnissen führen. Jiang (1997) zeigt, wie Normen an die Verteilung der Ausgangsdaten angepasst werden können, falls deren Verteilung bekannt ist. Im hier vorliegenden Fall der Bildanalyse hängt die Verteilung der Farbwerte vom einzelnen Bild ab und auch eine allgemeine Verteilung ist nicht angebbar. Aus den Ausführungen von Jiang (1997) geht hervor, dass auch bei Unkenntnis der Verteilung die euklidische Norm

als Klassifikator geeignet ist, wenn ausreichend viele Daten vorliegen. Da die Pixelzahl eines Bildes üblicherweise bei mehreren 100.000 liegt, wird für die folgenden Anwendungen generell die euklidische Norm verwendet.
Eine Clusterung ist eine Gruppierung von Daten, ohne dass dazu irgendwelches Vorwissen existieren muss. Eine Klassifikation dagegen ist die Zuordnung von Daten zu bekannten Gruppen. Es werden bei einer Klassifikation keine neuen Klassen geschaffen, sondern nur neue Werte bekannten Klassen zugeordnet.
Bei dem Begriff Klassifikation ist zwischen überwachten (oder beaufsichtigten) und unüberwachten (oder unbeaufsichtigten) Verfahren zu unterscheiden. Bei den überwachten Verfahren wird für einen Teil der zu klassifizierenden Daten, für den die Klassenzugehörigkeit bekannt ist, geprüft, wie gut die Klassifikation ist. Das Verfahren kann dann anhand einer Fehlerfunktion verbessert werden. Bei den unüberwachten Verfahren gibt es solche Kontrollmechanismen nicht. Da bei Clusterverfahren die Gruppen oder Klassen zum Zeitpunkt der Clusterung nicht bekannt sind, gehört die Clusterung zu den unüberwachten Verfahren (Niemann, 1983).

2.2 Klassische Verfahren

Der folgende Abschnitt gibt einen Überblick über die klassischen Clusterverfahren. Diese Verfahren beruhen auf dem k-Mittelwert-Algorithmus (Hartigan, 1975). Im Folgenden werden der k-Mittelwert-Algorithmus sowie die drei wichtigen Clusterungsmethoden ISODATA, Hyperklumpen-Strategie und der PHASE-Ansatz vorgestellt.

2.2.1 Der k-Mittelwert-Algorithmus

Der k-Mittelwert-Algorithmus ermöglicht die Einteilung der Ausgangsdaten in eine a priori vorgegebene Anzahl q von Klassen. Der Name „k-Mittelwert“ leitet sich davon ab, dass die Ausgangsdaten durch das wiederholte Bilden von Mittelwerten in k Klassen eingeteilt werden. In dieser Arbeit wird die Klassenzahl mit q statt k bezeichnet und dies wird auch bei der Vorstellung des k-Mittelwert-Algorithmus beibehalten.
Sei $\Omega \subseteq I\!R^{m}$ der Merkmalsraum, der die zu clusternden Ausgangsvektoren P_j, $j = 1,...,n$ enthält. Die Clusterung sei durch C beschrieben. Dabei ist $C = \{c_1,...,c_q\}$ die Menge der Klassen. Die Anzahl zu bildender Cluster sei mit q bezeichnet, wobei $q \ll n$ und q a priori gegeben. Die Zentroide und späteren Klassenrepäsentanten werden mit $z_1,...,z_q \hat{I} W$ bezeichnet. Die Zentroide sind die gesuchten Parameter der Clusterung. Daher stimmen hier Merkmalsraum und Parameterraum überein.
Zu Beginn des Algorithmus werden Zentroide $z_i^{(0)}, i = 1,...,q$ zufällig aus dem Merkmals- und Parameterraum W gewählt. Der Superskript-Index der Zentroide bezieht sich auf den jeweiligen Iterationsschritt. Die Null steht für den Initia-

lisierungsschritt. Im ersten Schritt wird eine Klassifizierung der n Ausgangsdaten P_j, $j = 1,...,n$ zu den q Zentroiden $z_1,...,z_q$ entsprechend einer vorgegebenen Norm vorgenommen. Dabei werden q Klassen $c_1,...,c_q$ gebildet. Als Norm dient i. Allg. die euklidische Norm. Die Zuordnung der Werte zu den Klassen ergibt sich somit durch:

$$P \in c_{k^*}^{(0)} \quad \Leftrightarrow \quad k^* = \underset{k=1,...,q}{argmin} \left\| P - z_k^{(0)} \right\|_2 .$$

Nach dieser Klassifikation enthalten die Klassen $c_i^{(0)}$, $i = 1,...,q$ jeweils die n_i Werte $P_{i1},...,P_{in_i}$, $i = 1,...,q$. Im zweiten Schritt wird innerhalb der q Klassen $c_1^{(0)},...,c_q^{(0)}$ der Mittelwert über die n_i Werte in der Klasse gebildet. Die Klassenmittelwerte werden komponentenweise nach

$$\overline{P}_i = \begin{pmatrix} \overline{P}_{i1} \\ \vdots \\ \overline{P}_{im} \end{pmatrix} = \begin{pmatrix} \frac{1}{n_i} \sum_{j=1}^{n_i} P_{ij1} \\ \vdots \\ \frac{1}{n_i} \sum_{j=1}^{n_i} P_{ijm} \end{pmatrix}, \qquad i = 1,...,q.$$

gebildet. Diese Mittelwerte sind die Zentroide für den nächsten Iterationsschritt:

$$z_i^{(1)} := \overline{P}_i , \quad i = 1,...,q .$$

Mit diesen neuen Zentroidwerten wird wieder eine Klassifikation der Ausgangsdaten vorgenommen. Die Iteration endet, wenn sich die Folge der Zentroide nicht mehr verändert: $z_i^{(\nu+1)} = z_i^{(\nu)}$, $i = 1,...,q$, wobei ν den Iterationsschritt bezeichnet. In sehr seltenen Fällen tritt diese Bedingung nicht ein. Bock (1974) zeigt, welche Abbruchbedingungen in so einem Fall anzuwenden sind.

Die Werte für die Bedingungen $g_n^i(\cdot)$, $i = 1,..,3$ aus (2.2) bis (2.4) für die Folge der $c_i^{(\nu)}$, $i = 1,..,q$ bzw. $z_i^{(\nu)}$, $i = 1,..,q$ fallen monoton mit wachsendem ν (Bock, 1974). Da nur eine endliche Menge an Clusterungen c_i, $i = 1,..,q$ existiert, wird für die Bedingungen $g_n^i(\cdot)$, $i = 1,..,3$ nach einer Reihe von Iterationsschritten ein Minimum erreicht. Nach Bock (1974) handelt es sich dabei oft nur um ein lokales Minimum. Es hängt von der Initialisierung der $z_i^{(0)}$, $i = 1,...,q$ ab, ob ein lokales oder globales Minimum erreicht wird und wie viele Iterationsschritte dazu nötig sind.

Allerdings treten bei Anwendung dieser Methode zwei große Probleme auf. Zum einen ist die Laufzeit oft sehr lang. Die lange Laufzeit ist durch die große Anzahl der Rekursionsschritte ν bedingt, bis eine stationäre Folge der $z_i^{(\nu)}$, $i = 1,...,q$ erreicht ist (Bock, 1974).

Zum anderen werden Elemente mit geringer Häufigkeit, die auf Grund ihrer Lage im Parameterraum eine eigene Klasse bilden sollten, oft nicht in eigenen Klassen identifiziert. Auf Grund ihrer geringen Häufigkeit werden sie oft benachbarten, aber dennoch entfernt liegenden Klassen mit mehr Elementen zugeordnet. Bei der anschließenden Mittelwertbildung zur Bestimmung des neuen Zentroids oder Klassenrepräsentanten, spiegeln sich die Informationen der seltenen Elemente auf Grund ihrer geringen Häufigkeit im Klassenrepräsentanten kaum wider. Schon bei einmaliger Mittelwertbildung besteht die Gefahr, dass seltene Elemente durch die Clusterung nicht mehr repräsentiert werden. Je nach Anwendungsgebiet sind solche seltenen Elemente für die Interpretation aber sehr wichtig.

2.2.2 Das ISODATA-Programm

Bei dem 1974 von Tou und Gonzales vorgestellten ISODATA-Programm (ISODATA = Iterative Self Organizing Data Technique A) handelt es sich um einen auf dem k-Mittelwert-Verfahren basierenden Algorithmus. Auf Grund der geringen Rechenleistung der Rechner von 1974 wird dabei die Iteration nicht bis zum Erreichen stationärer Folgen $z_i^{(\nu)},\ i = 1,...,q$ fortgesetzt, sondern nach einer Reihe von Iterationsschritten abgebrochen. Zum einen ist die Iterationsanzahl a priori durch einen Parameter begrenzt. Zum anderen wird in jedem Iterationsschritt auf Grund weiterer a priori festgelegter Parameter eine Abbruchbedingung geprüft, die ggf. vor Erreichen der Maximalanzahl an Iterationen zum Abbruch des Verfahrens führt. Die bis dahin erreichte Clusterung wird als Endergebnis gewertet. Selbst mit heutigen Rechnern kann eine Clusterung mit dem k-Mittelwert-Verfahren in Abhängigkeit von Größe (Anzahl der Pixel) und Inhalt (Anzahl unterschiedlicher Pixel) viele Stunden, Tage oder Wochen dauern (Myers et al, 1997b). Das ISODATA-Programm stellte eine erste Verbesserung des k-Mittelwert-Verfahrens zur praktischen Umsetzung dar. Mit der Rechnerumsetzung dieses Verfahrens steht seit 1974 ein brauchbarer Algorithmus für eine Vielzahl von Clusterungsproblemen in der Praxis zur Verfügung. Die Identifizierung seltener Elemente in eigenen Klassen ist dadurch jedoch weiterhin nicht gewährleistet.

2.2.3 Die Hyperklumpen-Strategie

Die Hyperklumpen-Strategie von Kelly und White (1993) ist eine Weiterentwicklung des k-Mittelwert-Verfahrens. Die Weiterentwicklung soll folgendem Problem entgegenwirken:
Oftmals werden Werte, die in die gleiche Klasse gehören, dennoch in verschiedene Klassen klassifiziert, weil besondere Einflüsse die wirkliche Datenlage verfälschen. Angenommen, es liegt eine Aufnahme eines Waldstücks auf einem Bergrücken vor. Auf Grund der tiefstehenden Sonne und der Hanglage liegt ein Teil des Bergrückens

im Schatten; der andere Teil wird von der Sonne beschienen. Dadurch weisen die beiden Teilbereiche des Waldstücks eine deutlich unterschiedliche Farbgebung auf. Daher werden sie in verschiedene Klassen eingeteilt, obwohl sie eigentlich in eine Klasse gehören. Der k-Mittelwert-Clusteralgorithmus kann dieses Problem nicht lösen, deshalb kommt hier die Hyperklumpen-Strategie zur Anwendung:
Das Bild wird in wesentlich mehr Klassen unterteilt, als vom Analysten im Vorhinein festgelegt sind. „Wesentlich mehr" kann dabei drei- bis zehnmal mehr bedeuten. Kelly und White (1993) empfehlen jedoch aus zwei Gründen, nicht mehr als insgesamt 256 Klassen zu verwenden. Zum einen sind mehr als 256 Klassen vom menschlichen Auge nicht zu unterscheiden. Zum anderen basiert dieser Ansatz allein auf Graustufendarstellungen. Bei 256 Klassen werden im Rechner pro Datenpunkt nur 8 Bit an Speicherplatz benötigt. Das vereinfacht und verkürzt die Umsetzung im Rechner.
Die Klassen dieser ersten Clusterung werden als Hyperklumpenschicht bezeichnet, weil diese Clusterung deutlich mehr Klassen als letztlich gewollt enthält. Die Berechnung erfolgt mit dem üblichen k-Mittelwert-Algorithmus. Auf Grund der Klassenrepräsentanten der Hyperklumpenschicht wird eine weitere Clusterung durchgeführt, um die Klassenzahl auf die ursprünglich vom Analysten vorgegebene zu reduzieren. Daher hat dieses Verfahren den Namen „Hyperklumpen-Strategie", weil Cluster geclustert werden. Bei diesem weiteren Schritt wird die Clusterung nicht mehr unbeaufsichtigt durchgeführt, sondern nach den Vorgaben des Analysten. Durch diese Beaufsichtigung können z. B. bei obigem Beispiel die Waldstücke, die in unterschiedliche Klassen einsortiert wurden, an dieser Stelle nach Vorgabe des Analysten zusammengefügt werden. Sollten keine besonderen Erkenntnisse über die Hyperklumpenschicht vorliegen, kann der weitere Schritt auch wieder als unbeaufsichtigte Clusterung nach dem k-Mittelwert-Algorithmus durchgeführt werden.
Die Idee der Hyperclusterung besteht also darin, die unbeaufsichtigte Clusterung an geeigneter Stelle durch eine beaufsichtigte Klassifikation zu erweitern, um dadurch vorhandene Kenntnisse einbringen zu können und zu besseren Ergebnissen zu gelangen.

2.2.4 Der PHASE-Ansatz

Der PHASE-Ansatz (PHASE = Pixel Hypercluster Approximating Spatial Ensembles) von Myers et al (1997b) weist zusätzlich zur Hyperklumpen-Strategie eine Reihe technischer Neuerungen auf. Diese dienen dazu, die bisher präsentierten Methoden in eine Rechnerumgebung zu implementieren.
Bei der Umsetzung in ein Rechnerprogramm soll die Berechnungsdauer einer Clusterung möglichst kurz sein. Gerade bei auf dem k-Mittelwert-Algorithmus basierenden Verfahren, ist die praktische Berechnungszeit lang, weil eine unbekannte Zahl an Iterationen und eine Vielzahl an Zuordnungen und Mittelwertbildungen not-

wendig sind. Die Detaillösungen bei der Implementierung nach den Ideen von Myers et al (1997b) sollen diesem Problem entgegenwirken. Dazu kommen unter anderem zusätzliche Methoden der Vor- und Nachverarbeitung zum Einsatz.
Die erste Verbesserung besteht in einer speziellen Definition des Algorithmus, so dass alle Abläufe auf *relationalen Tabellen* abgewickelt werden können. Die Definition relationaler Tabellen ist bei Schmidt (1983) zu finden. Bei dieser Form der Datenspeicherung wird ein Datenpunkt zerlegt und in einer Haupttabelle und mehreren Nebentabellen gespeichert. Zwischen den Tabellen werden über ein eindeutiges Schlüsselelement Beziehungen hergestellt. Diese definieren, wie die gespeicherten Daten zusammengehören. Diese Form der Speicherung hat insbesondere dann Vorteile, wenn zu einem Hauptdatensatz mehrere Detaildatensätze vorhanden sind. Die relationale Datenspeicherung wird anhand der folgenden Rasterbildskizze dargestellt:

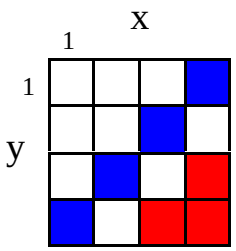


Abb. 2.1: Skizze eines einfachen Rasterbildes

Die Informationen des Rasterbildes aus Abb. 2.1 lassen sich in folgender Tabelle darstellen:

Koordinaten		RGB-Werte		
x	y	Rot-Anteil	Grün-Anteil	Blau-Anteil
1	1	255	255	255
2	1	225	225	225
3	1	255	255	255
4	1	0	0	255
1	2	255	255	255
2	2	255	255	255
3	2	0	0	255
4	2	255	255	255
1	3	255	255	255
2	3	0	0	255
3	3	255	255	255
4	3	255	0	0
1	4	0	0	255
2	4	255	255	255
3	4	255	0	0
4	4	255	0	0

Tab. 2.1: Informationen der Rasterbildskizze aus Abb. 2.1 anhand von Pixelkoordinaten und deren RGB-Farbwert

Im Falle der relationalen Datenspeicherung werden die Informationen über Koordinaten und Farbwerte der Pixel in zwei Tabellen getrennt abgespeichert. Die Tabellen werden über ein Schlüsselelement, hier die Farbnummer (Farbnr.), miteinander verbunden.

Koordinaten		
x	y	Farbnr.
1	1	1
2	1	1
3	1	1
4	1	2
1	2	1
2	2	1
3	2	2
4	2	1
1	3	1
2	3	2
3	3	1
4	3	3
1	4	2
2	4	1
3	4	3
4	4	3

Tab. 2.2.a: Koordinaten und Farbnummer der Rasterbildskizze aus Abb. 2.1

	RGB-Werte		
Farbnr.	Rot-Anteil	Grün-Anteil	Blau-Anteil
1	255	255	255
2	0	0	255
3	255	0	0

Tab. 2.2.b: Farbnummer und RGB-Werte der Rasterbildskizze aus Abb. 2.1

Wie das Beispiel zeigt, sind durch die Aufteilung der Informationen auf zwei Tabellen insgesamt weniger Informationen zu speichern. Es werden keine redundanten Informationen gespeichert, weil zu jedem Hauptdatensatz (Koordinaten der Pixel) nicht jedesmal alle Details (RGB-Werte des Pixels) gespeichert werden müssen. Diese werden separat in einer Detailtabelle (Tab. 2.2.b) gespeichert und über das Schlüsselelement „Farbnr." einander zugeordnet. Das spart bei der Implementierung Speicherplatz und verbessert die Laufzeit der Algorithmen.
Im Falle der Bildclusterung bietet diese Form der Datenverwaltung den zusätzlichen Vorteil, dass nur zu Beginn des Algorithmus auf die eigentliche Bilddatei zugegriffen werden muss und die folgenden Berechnungen programmintern durchgeführt werden können. Das verbessert die Laufzeit der Clusterung. Im Falle des PHASE-Ansatzes werden die Klassen der Clusterung in solchen relationalen Tabellen verwaltet. Als Schlüsselelement für die Beziehung zwischen den relationalen Tabellen wird die Nummerierung der Klassen c_i verwendet. Die Nummerierung erfolgt gemäß des Rangs der Zentroide z_i , $i = 1,...,q$ (Gibbons, 1992): $No(c_i) = Rank(\|z_i\|_2)$.

Die zweite Neuerung besteht in der Einführung sogenannter Startwertzerstreuungs-Algorithmen für die ersten Zentroidwerte $z_i^{(0)}$, $i = 1,...,q$. Dabei werden die Zentroide so angeordnet, dass sie sich für eine erste Klassifizierung eignen. Nach Myers, Patil und Taillie (1997a) sind Zentroide dann geeignet, wenn sich alle Zentroide unterscheiden und der euklidische Abstand zwischen den Zentroiden möglichst groß ist. Durch das Vorschalten eines Startwertzerstreuungs-Algorithmus vor den Clusterungsprozess wird die Laufzeit bis zum Erreichen einer statischen Clustermenge beim k-Mittelwert-Verfahren verkürzt.
Die dritte Neuerung besteht in einer Nachverarbeitung der Clusterung. Dabei sollen „zu kleine" Klassen vermieden werden. Diese Form der Nachverarbeitung ist für die Anwendung in der Erosionsforschung eher kritisch zu bewerten. Gerade die angeblich zu kleinen Klassen beinhalten oft wichtige Elemente. Diese Klassen werden schon mit dem einfachen k-Mittelwert-Verfahren oftmals gar nicht gefunden. In den Anwendungen von Myers, Patil und Taillie (1997b) bestand jedoch offenbar ein Problem mit „zu kleinen" Klassen. Daher ist der PHASE-Ansatz in der Lage, sehr kleine Klassen mit wenigen Elementen, die inhaltlich keine Bedeutung haben, wieder aufzulösen. Damit eine Mindestgröße der Klasse garantiert ist, wird der zugehörige Zentroid der „zu kleinen" Klassen gelöscht. Die Werte dieser Klasse werden, gemäß der noch vorhandenen Zentroide, neu klassifiziert. Der frei werdende Zentroidwert wird dazu verwendet, die Klasse mit der größten Intra-Class-Varianz aufzuspalten.

2.3 Probleme der Anwendung des k-Mittelwert-Verfahrens und Hinweise für verbesserte Clusterverfahren

Das k-Mittelwert-Verfahren weist einige Probleme auf. Insbesondere bei der Bildanalyse im ökologischen Anwendungsbereich hat dieses besonders starke Auswirkungen. Der Laufzeiteffekt tritt deshalb so stark auf, weil ein Bild zur Clusterung in die einzelnen Pixel zerlegt wird, aus denen es besteht. Die Anzahl dieser Pixel beträgt oft mehrere 100.000. Diese Pixel bestehen zusätzlich selbst aus einem Vektor, den drei Komponenten Rot, Grün und Blau, bei der RGB-Darstellung. Somit liegt dem Clusterungsproblem eine sehr große Datenmenge zu Grunde, die bei Anwendung des k-Mittelwert-Verfahrens in jedem Iterationsschritt klassifiziert werden muss. Um die Laufzeit zu verkürzen, müssen Algorithmen gefunden werden, die im Idealfall nur einmal auf die Ursprungsdaten zugreifen, um den Aufwand möglichst klein zu halten. Die später folgenden Algorithmen Maximum Linkage und die neuronalen Netze greifen jeweils genau zweimal auf die gesamte Datenmenge zu: Einmal, um die Zentroide zu finden, anhand derer die Klassifikation durchgeführt wird, und ein zweites Mal für die Klassifikation der einzelnen Werte. Damit wird die Laufzeit dieser Algorithmentypen gegenüber dem k-Mittelwert-Verfahren um ein Vielfaches kürzer.

Das zweite Problem, dass seltene Elemente bei der Mittelung verloren gehen, ist in der Erforschung der Erosion besonders kritisch. Gerade die seltenen Elemente haben oft besonders große Auswirkungen auf die Erosion. So ist z. B. ein Fahrweg, eine Mauer oder eine Hecke in der Wüste von großer Bedeutung, da dadurch Verwüstung und Erosion stark beeinflusst werden, auch wenn die Fläche auf dem Bild relativ zur Gesamtfläche klein ist. Angenommen, es wird eine Luftaufnahme einer Wüstengegend analysiert. In dieser ist ein Feld von einer kleinen Mauer eingegrenzt, so dass die Winderosion dort deutlich weniger Schaden anrichten kann als im übrigen Umfeld. Bei Anwendung des k-Mittelwert-Verfahrens findet man nach der Clusterung oft nur die Flächen innerhalb und außerhalb der Mauer. Die Ursache für die Trennung der Flächen, die Mauer, wird in keiner Klasse gefunden, da sie im Vergleich zu den Flächen durch zu wenig Pixel repräsentiert wird. Somit gehen ursächliche Elemente durch die Clusterung verloren. Eine weitergehende Analyse wird dadurch erschwert. Zu dieser Problematik ließen sich noch eine ganze Reihe von Beispielen finden, wie z. B. einzelne Sträucher in einer von Verwüstung bedrohten Landschaft, die bei einer Clusterung mittels k-Mittelwert-Algorithmus nicht in einer eigenen Klasse identifiziert werden. Auf Flächen mit solchem Bewuchs ist eine andere Erosionsentwicklung zu erwarten als auf Flächen ohne Bewuchs. Daher ist es von Nachteil, wenn die Ursachen für eine unterschiedliche Erosionsentwicklung durch die Clusterung aus dem Bild entfernt werden.
Somit besteht der Bedarf, bessere Algorithmen und Verfahren zu finden. Bei der Entwicklung verbesserter Algorithmen ist, auf Grund der obigen Überlegungen, auf drei Punkte besonders zu achten:

1. **Laufzeit:**
   Die verbesserten Algorithmen sollten kurze Laufzeiten aufweisen.

2. **Herausarbeitung seltener Elemente:**
   Die in ökologischen Anwendungen wichtigen seltenen Elemente einer Aufnahme dürfen durch die Clusterung nicht verloren gehen.

3. **Deutliche Trennung zwischen den Klassen:**
   Für die Weiterverwendung des Ergebnisbildes einer Clusterung ist eine deutliche Trennung zwischen den Klassen wichtig. Diese Trennung ist notwendig, damit z. B. die verschiedenen Bereiche eines Bodens im Feld wiedergefunden werden können. Damit können in diesen Bereichen zusätzliche Bodendaten für weitergehende Analysen erhoben werden.

Um diese drei Forderungen zu erfüllen, werden folgende Strategien verfolgt:

Zu 1. **Laufzeit:**

Die zur Clusterung verwendeten Verfahren dürfen keine unbestimmte Anzahl von Iterationsschritten beinhalten, so dass von vornherein die Laufzeit besser kalkuliert werden kann.

Der Aufwand in jedem Iterationsschritt sollte möglichst klein sein oder zumindestens im Verhältnis zur Anzahl der notwendigen Iterationsschritte stehen. Werden, wie beim Maximum Linkage, nur wenige Iterationen benötigt, kann der Rechenaufwand pro Schritt größer sein als wenn viele Iterationsschritte nötig sind, wie bei neuronalen Netzen.

Zu 2. **Herausarbeitung seltener Elemente**

Um seltene Elemente bei der Clusterung nicht zu verlieren, muss auf zwei Dinge geachtet werden. Zuerst darf das Clusterverfahren nur auf so wenig Mittelungen wie möglich beruhen. Bei jeder Durchschnittsbildung besteht die Gefahr, dass seltene Elemente in eine Klasse mit vielen anderen Werten klassifiziert wurden und bei der Mittelung auf Grund der geringen Häufigkeit durch den Klassenrepräsentanten nicht repräsentiert werden. Mit jeder weiteren Mittelung verstärkt sich dieser Effekt. Daher ist es am besten, wenn nur eine Mittelwertbildung stattfindet. Sowohl beim später verwendeten Maximum Linkage als auch bei den neuronalen Netzen wird erst nach der Ermittlung der Zentroide und der darauf beruhenden Klassifikation einmal in den einzelnen Klassen gemittelt.

Zum Zweiten müssen die Zentroide geeignet gewählt werden. Nur dadurch ist zu gewährleisten, dass überhaupt vernünftige Klassen für die seltenen Elemente zur Verfügung stehen. Bei der auf den Zentroiden beruhenden Klassifikation müssen die seltenen Werte dann nur den entsprechenden Klassen zugeordnet werden.

Zu 3. **Deutliche Trennung zwischen den Klassen:**

Spricht man von der Trennung zwischen den Klassen einer Clusterung, so muss dieses zunächst einmal mathematisch formuliert werden. Dazu führe ich das Kriterium des mittleren minimalen Abstands zwischen den Zentroiden der Clusterung ein. Dieses in Kapitel 5.3 genauer beschriebene Kriterium, wird im Späteren zur Beurteilung der Qualität einer Clusterung und als Entscheidungskriterium zwischen verschiedenen Clusterungen verwendet.

Je größer der Wert des mittleren minimalen Abstands zwischen den Zentroiden ist, desto besser ist eine Clusterung. Welche Werte für dieses Kriterium erreicht werden, wird im Wesentlichen durch die Grundstruktur des Algorithmus und die Auswahl der Zentroide bedingt.

3 Maximum Linkage

Im Rahmen dieser Arbeit wurde ein neues Verfahren zur Clusterung entwickelt. Diesem gebe ich den Namen Maximum-Linkage-Algorithmus. Das Verfahren soll die in Kapitel 2 aufgeführten Nachteile der klassischen Verfahren beheben.
Der Maximum-Linkage-Algorithmus dient dazu, aus einer zu clusternden Wertemenge geeignete Zentroide auszuwählen. Anhand dieser Zentroide wird eine Klassifikation der übrigen Werte durchgeführt. Durch diese beiden Schritte wird eine Clusterung erstellt. Maximum Linkage hat eine entfernte Verwandtschaft zum Single Linkage (Johnson und Wichern, 1992).

3.1 Single Linkage

Der Single-Linkage-Algorithmus ist ein hierarchisches Clusterverfahren. Dabei werden die zu gruppierenden Werte iterativ in Klassen zusammengefasst, bis schließlich alle Werte in einer einzigen Klasse Z zusammengefasst sind. Weil die Werte Schritt für Schritt gruppiert werden, wird dieses Verfahren als hierarchisch bezeichnet. Durch die schrittweise Klassenbildung wird eine Ordnungsrelation definiert. Diese Ordnungsrelation beschreibt die Struktur der zu gruppierenden Werte.

Der Ablauf des Single Linkage gestaltet sich wie folgt:
Sei $\Omega \subseteq \mathbb{R}^m$ der Merkmalsraum, der die verschiedenen Werte $P_1,...,P_r$ beinhaltet. Im ersten Schritt werden die Distanzen zwischen allen Punkten $P_1,...,P_r$ berechnet und in eine Distanzmatrix D eingetragen:

$$D = (d_{ij}),\ d_{ij} = \left\| P_i - P_j \right\|_2 ,\quad i, j = 1,...,r \ .$$

Die beiden Werte P_{i^*}, P_{j^*}, die zum minimalen d_{ij}, $i \neq j$ aus D gehören, werden in einer Klasse Z_1 aufgenommen:

$$P_{i^*}, P_{j^*} \in Z_1 \quad \Leftrightarrow \quad \{i^*, j^*\} = \underset{i,\, j \in \{1,...,r\},\, i \neq j}{arg\,min}\ d_{ij} \ .$$

Ist das Minimum nicht eindeutig, so wird zufällig einer der Minimalwerte verwendet. Die nicht ausgewählten Werte bilden einelementige Klassen:

$$Z_j := P_i \ ,\ \ j \in \{2,..., r-1\},\ \ i \in \{1,..., r\} \setminus \{i^*, j^*\} \ .$$

Im zweiten Schritt wird wieder eine Distanzmatrix D berechnet. Diese enthält alle Distanzen d_{ij} zwischen den Mengen Z_i, Z_j. Der Abstand d zwischen zwei Klassen definiert sich über:

$$d_{ij} := d\left(Z_i, Z_j\right) := \min_{P_i \in Z_i,\, P_i \in Z_i} \left\| P_i - P_j \right\|_2$$

Die zum minimalen d_{ij} gehörenden Werte werden vereinigt:

$$Z_{i^*} \cup Z_{j^*} \quad \Leftrightarrow \quad \left\{i^*, j^*\right\} = \arg\min_{i,j \in \{1,\ldots,r-1\}} d_{ij}$$

Der zweite Schritt wird nun so lange wiederholt, bis alle Klassen in einer einzigen Klasse Z zusammengefasst worden sind. Dazu sind genau $(r-1)$-Iterationen nötig, weil in jedem Iterationsschritt die Klassenanzahl um eins verringert wird.
Um die im Laufe des Verfahrens entstandene Ordnung auf der Menge Z graphisch darstellen zu können, wird ein *Dendrogramm* verwendet. Die Skizze eines Dendrogramms ist in Abb. 3.1 dargestellt. In einem Dendrogramm werden die vereinigten Klassen über einen Bügel verbunden. Die Höhe des Bügels richtet sich nach dem *Eingangsniveau* der Klasse, die durch den Bügel definiert wird. Das Eingangsniveau E der Klasse $Z = Z_{i^*} \cup Z_{j^*}$ wird durch die zugehörige Distanz $d_{i^*j^*}$ definiert.

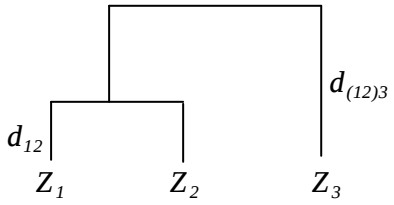


Abb. 3.1: Skizze eines Dendrogramms für 3 Klassen.

3.2 Beispiel zum Single Linkage

Das folgende zweidimensionale Beispiel verdeutlicht die Funktionsweise des Single Linkage. Die Werte (1,1) ; (1,2) ; (4,3) ; (5,5) und (7,5) sollen geclustert werden. Die Distanzmatrix des ersten Iterationsschritts ist in Tab. 3.1 dargestellt. Dabei wird die euklidische Norm als Abstandsmaß verwendet. Die Einträge der Matrix sind auf eine Dezimalstelle gerundet.

	(1,1)	(1,2)	(4,3)	(5,5)	(7,5)
(1,1)					
(1,2)	1				
(4,3)	3,6	3,2			
(5,5)	5,7	5,0	2,2		
(7,5)	7,2	6,7	3,6	2,0	

Tab. 3.1: Distanzmatrix des ersten Iterationsschritts. Die Einträge sind auf eine Dezimalstelle gerundet.

Auf Grund der Symmetrie der Norm wird nur die untere Dreiecksmatrix betrachtet. Auch die Hauptdiagonale darf nicht berücksichtigt werden, da es keinen Sinn macht, ein Element mit sich selbst zusammen zu fassen. Die grau hinterlegte Fläche kennzeichnet das Minimum. Im ersten Iterationsschritt entsteht damit die Clusterung:

$$C^{(1)} = \{\{(1,1);(1,2)\};\{(4,3)\};\{(5,5)\};\{(7,5)\}\} \ .$$

Das Eingangsniveau beträgt: $E^{(1)} = 1$.

Die Distanzmatrix des zweiten Iterationsschritts ergibt sich damit als:

{(1,1) ; (1,2)}	{(1,1) ; (1,2)}	(4,3)	(5,5)	(7,5)
(4,3)	Min{3,6 ; 3,2}			
(5,5)	Min{5,7 ; 5,0}	2,2		
(7,5)	Min{7,2 ; 6,7}	3,6	2,0	

Tab. 3.2: Distanzmatrix des zweiten Iterationsschritts. Die Einträge sind auf eine Dezimalstelle gerundet.

Das grau hinterlegte Feld weist auf das Minimum hin und damit ergibt sich die Clusterung

$$C^{(2)} = \{\{(1,1);(1,2)\};\{(4,3)\};\{(5,5);(7,5)\}\}$$

mit dem Eingangsniveau: $E^{(2)} = 2$.

In der dritten Iteration ergibt sich eine Distanzmatrix der folgenden Form:

{(1,1) ; (1,2)}	{(1,1) ; (1,2)}	(4,3)	{(5,5) ; (7,5)}
(4,3)	Min{3,6 ; 3,2}		
{(5,5) ; (7,5)}	Min{5,7 ; 7,2 ; 5,0 ; 6,7 }	Min{2,2 ; 3,6)	

Tab. 3.3: Distanzmatrix des dritten Iterationsschritts. Die Einträge sind auf eine Dezimalstelle gerundet.

Wie das grau hinterlegte Feld in Tab. 3.3 zeigt, entsteht zum Niveau $E^{(3)} = 2{,}2$ die Clusterung

$$C^{(3)} = \{\{(1,1);(1,2)\};\{(4,3);(5,5);(7,5)\}\} .$$

Im vierten und letzten Iterationsschritt werden die beiden Klassen des dritten Iterationsschritts zusammengefasst. Die Distanzmatrix ist nur noch nötig, um das Niveau zu bestimmen:

{(1,1) ; (1,2)}	{(1,1) ; (1,2)}	{(4,3) ; (5,5) ; (7,5)}
{(4,3) ; (5,5) ; (7,5)}	Min{3,6 ; 3,2 ; 5,7 ; 5,0 ; 7,2 ; 6,7 }	

Tab. 3.4: Distanzmatrix des vierten Iterationsschritts (Einträge auf eine Nachkommastelle gerundet)

Aus Tab. 3.4 ergibt sich das Niveau $E^{(3)} = 3{,}2$ aus dem grau hinterlegten Wert. Auf Grund dieser Informationen lässt sich das Dendrogramm der Clusterung der Werte (1,1) ; (1,2) ; (4,3) ; (5,5) und (7,5) nach Single Linkage darstellen. :

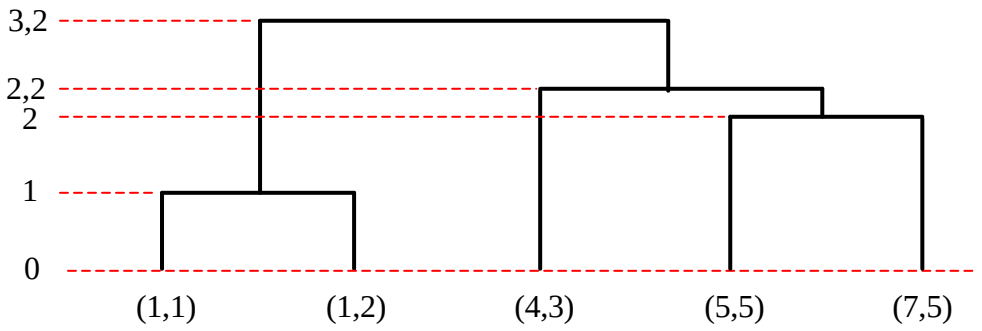


Abb. 3.2: Dendrogramm einer Clusterung mit Single Linkage. Die gestrichelten Linien dienen zur Angabe der jeweiligen Eintrittsniveaus.

3.3 Der Maximum-Linkage-Algorithmus

Der Maximum-Linkage-Algorithmus ist zwar algorithmisch ähnlich zum Single Linkage, wird allerdings in einer anderen Wirkungsweise eingesetzt. Während beim Single Linkage eine Ordnungsrelation der zu clusternden Werte entsteht, wird beim Maximum Linkage eine Aufteilung der zu clusternden Werte in q vorgegebene Klassen erreicht. Das Entstehen einer Ordnungsrelation ist die Eigenschaft eines hierarchischen Clusterverfahrens (Anderberg, 1973). Bei den klassischen Weiterentwicklungen von Single Linkage handelt es sich immer um eng verwandte hierarchische Verfahren (Everitt, 1980).

Bei der Clusterung mit Maximum Linkage wird die Äquivalenz zwischen dem Erstellen von q Klassen und dem Auffinden von q Zentroiden ausgenutzt (Bock, 1998). Maximum Linkage wählt Zentroide aus der Menge der zu clusternden Werte aus. Dabei wird der Abstand zwischen den Zentroiden möglichst groß gehalten. Der Abstand wird aus den selben Gründen wie in Abschnitt 2.1 beschrieben gemäß der

euklidischen Norm bestimmt. Die Funktionsweise des Maximum Linkage ist aber nicht auf diese Norm beschränkt, sondern funktioniert mit beliebiger Norm.
Die durch Maximum Linkage berechneten Zentroide dienen zur Klassifizierung der übrigen Werte. Zum Abschluss des Verfahrens werden in allen entstandenen Klassen Mittelwerte gebildet, die als Klassenrepräsentanten dienen. Formal bedeutet das:
Sei W der Merkmalsraum, der die verschiedenen Werte $P_1,...,P_r$ beinhaltet und q die Anzahl der gesuchten Cluster mit $q \ll r$. Sei $\{P_{1^*},...,P_{q^*}\}$, $\{1^*,...,q^*\} \subset \{1,...,r\}$ die Menge der ausgewählten Zentroide, die zur Klassifikation aller übrigen Werte dienen. Die ausgewählten Zentroide $\{P_{1^*},...,P_{q^*}\}$ sollen folgender Bedingung genügen:

$$\min_{i,j \in \{1^*,...,q^*\},\ i \neq j} \| P_i - P_j \|_2 \rightarrow \max_{\{1^*,...,q^*\} \subset \{1,...,r\}} . \qquad (3.1)$$

Diese hier neu eingeführte Clusterungsbedinung wird im Folgenden als *Maximum-Linkage-Bedingung* bezeichnet. Bedingung (3.1) fordert, dass der minimale Abstand zwischen den Zentroiden möglichst groß ist. Die Bestimmung, der durch (3.1) vorgegebenen Wertemenge $P_{1^*},..., P_{q^*}$, ist nicht trivial. Es ist daher eine algorithmische Umsetzung gesucht. In dieser werden Zentroide bestimmt, die der Maximum-Linkage-Bedingung möglichst nahe kommen.
Der Maximum-Linkage-Algorithmus ist ein solches Verfahren. Betrachtungen zur Abweichung zwischen den durch den Maximum-Linkage-Algorithmus bestimmten Zentroiden und den durch die Maximum-Linkage-Bedingung (3.1) geforderten, werden in Abschnitt 3.6 durchgeführt.

Der Ablauf des Maximum-Linkage-Algorithmus gestaltet sich wie folgt:

Der Algorithmus ist ein schrittweises Verfahren. Die Zentroide werden iterativ bestimmt, bis die a priori festgelegte Anzahl q erreicht ist. Dazu werden zunächst die Distanzen zwischen allen Punkten $P_1,...,P_r$ berechnet und in eine Distanzmatrix D eingetragen:

$$D = (d_{ij}),\ d_{ij} = \| P_i - P_j \|_2 ,\quad i,j = 1,...,r \ . \qquad (3.2)$$

Auf Grund der Symmetrieeigenschaft aller Normen ist es ausreichend, nur die obere oder untere Dreiecksmatrix zu betrachten. Die Betrachtung der Hauptdiagonalen von D ist ebenfalls unnötig. Alle Abstände auf der Hauptdiagonalen sind gleich null und haben damit auf die folgenden Betrachtungen keine Auswirkung.
Als nächstes wird das Maximum der Distanzen d_{ij} aus D bestimmt. Sollte das Maximum nicht eindeutig sein, treten dadurch keine großen Schwierigkeiten auf. Jeder

Maximalwert ist geeignet. Damit wird ein Wert zufällig ausgewählt und für die folgenden Operationen verwendet. Die beiden zum Maximum korrespondierenden Punkte werden in die Menge der gesuchten Zentroide Z aufgenommen:

$$P_{i^*}, P_{j^*} \in Z \quad \Leftrightarrow \quad \{i^*, j^*\} = \underset{i,j \in \{1,\ldots,r\}}{arg\,max} d_{ij} \; .$$

Anschließend werden alle Distanzen zwischen den Werten in Z und den übrigen Werten berechnet und in eine neue Distanzmatrix eingetragen. Diese hat die Dimension $(r - 2) \times 2$. Damit reduziert sich die Anzahl der zu betrachtenden Distanzen von $(r-1) \cdot r / 2$ auf $(r - |Z|) \cdot |Z|$. Zur formalen Darstellung dieser Distanzmatrix werden die Indizes der Werte P_{i^*}, $i^* = 1,\ldots,|Z|$, die bereits als Zentroide ausgewählt wurden, in der Indexmenge Q zusammengefasst. Weiter sei die Indexmenge R definiert durch $R = \{1,\ldots,r) \setminus Q$. Damit beinhaltet R die Indizes der Werte P_i, die nicht als Zentroide festgelegt wurden. Die Distanzen d_{iQ} sind damit gegeben durch:

$$d_{iQ} = \min_{k \in Q} \|P_i - P_k\|_2, \quad i \in R \; . \qquad (3.3)$$

Das Maximum der Distanzen d_{iQ} wird aufgesucht und der zugehörige Punkt in die Menge der Zentroide gemäß

$$Z_{neu} = P_{i^*} \cup Z_{alt} \quad \Leftrightarrow \quad i^* = \arg\max_{i \in R} d_{iQ} \qquad (3.4)$$

aufgenommen. Die Schritte, beginnend mit der Abstandsberechnung zwischen den Punkten in Z und den übrigen Werten gemäß Formel (3.3), werden solange wiederholt, bis die geforderte Anzahl q an Zentroiden erreicht ist, also $|Z| = q$.

3.4 Beispiel zum Maximum-Linkage-Algorithmus

Das folgende Beispiel verdeutlicht den Maximum-Linkage-Algorithmus. Aus einer Menge von fünfzehn Punkten im $\mathbb{N}^2$ seien vier Zentroide auszuwählen. Tab. 3.5 enthält die fünfzehn Punkte und die auf eine Dezimalstelle gerundeten Distanzen zwischen den Punkten gemäß Formel (3.2).

Index *i*		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
i	P_i	*(1,9)*	*(2,1)*	*(3,1)*	*(3,3)*	*(3,8)*	*(4,8)*	*(5,5)*	*(6,4)*	*(6,6)*	*(7,2)*	*(8,1)*	*(8,5)*	*(8,7)*	*(9,3)*	*(9,6)*
1	*(1,9)*	*0*														
2	*(2,1)*	*8,1*	*0*													
3	*(3,1)*	*8,3*	*1,0*	*0*												
4	*(3,3)*	*6,3*	*2,2*	*2,0*	*0*											
5	*(3,8)*	*2,2*	*7,1*	*7,0*	*5,0*	*0*										
6	*(4,8)*	*3,2*	*7,3*	*7,1*	*5,1*	*1,0*	*0*									
7	*(5,5)*	*5,7*	*5,0*	*4,5*	*2,8*	*3,6*	*3,2*	*0*								
8	*(6,4)*	*7,1*	*5,0*	*4,2*	*3,2*	*5,0*	*4,5*	*1,4*	*0*							
9	*(6,6)*	*5,8*	*6,4*	*5,8*	*4,2*	*3,6*	*2,8*	*1,4*	*2,0*	*0*						
10	*(7,2)*	*9,2*	*5,1*	*4,1*	*4,1*	*7,2*	*6,7*	*3,6*	*2,2*	*4,1*	*0*					
11	*(8,1)*	*10,6*	*6,0*	*5,0*	*5,4*	*8,6*	*8,1*	*5,0*	*3,6*	*5,4*	*1,4*	*0*				
12	*(8,5)*	*8,1*	*7,2*	*6,4*	*5,4*	*5,8*	*5,0*	*3,0*	*2,2*	*2,2*	*3,2*	*4,0*	*0*			
13	*(8,7)*	*7,3*	*8,5*	*7,8*	*6,4*	*5,1*	*4,1*	*3,6*	*3,6*	*2,2*	*5,1*	*6,0*	*2,0*	*0*		
14	*(9,3)*	*10,0*	*7,3*	*6,3*	*6,0*	*7,8*	*7,1*	*4,5*	*3,2*	*4,2*	*2,2*	*2,2*	*2,2*	*4,1*	*0*	
15	*(9,6)*	*8,5*	*8,6*	*7,8*	*6,7*	*6,3*	*5,4*	*4,1*	*3,6*	*3,0*	*4,5*	*5,1*	*1,4*	*1,4*	*3,0*	*0*

Tab. 3.5: Distanzmatrix gemäß Formel (3.2). Die Einträge sind auf eine Dezimalstelle gerundet.

Das Maximum der Abstände beträgt *10,6*. Dies wird in Tab. 3.5 durch die hervorgehobene Umrandung der Zelle angezeigt. Damit sind die ersten beiden Zentroide durch die entsprechend korrespondierenden Werte gegeben:

$$Z = \{\ P_1\ ;\ P_{11}\ \} = \{\ (1{,}9)\ ;\ (8{,}1)\ \}.$$

Um den dritten der gesuchten vier Werte zu finden, wird ein weiterer Rekursionsschritt begonnen. Dazu werden die Distanzen zwischen den beiden Punkten in Z und den übrigen Punkten gemäß (3.3) ermittelt. Diese müssen nicht unbedingt neu berechnet werden, sondern können auch direkt aus der Matrix D durch Herausschreiben der entsprechenden Zeilen und Spalten ermittelt werden.

Index	*Punkte*	*Z*	
i	P_i	*(1,9)*	*(8,1)*
Z	*{ (1,9) ; (8,1) }*	-	-
2	*(2,1)*	*8,1*	*6,0*
3	*(3,1)*	*8,3*	*5,0*
4	*(3,3)*	*6,3*	*5,4*
5	*(3,8)*	*2,2*	*5,0*
6	*(4,8)*	*3,2*	*8,1*
7	*(5,5)*	*5,7*	*5,0*
8	*(6,4)*	*7,1*	*3,6*
9	*(6,6)*	*5,8*	*5,4*
10	*(7,2)*	*9,2*	*1,4*
12	*(8,5)*	*8,1*	*4,0*
13	*(8,7)*	*7,3*	*6,0*
14	*(9,3)*	*10,0*	*2,2*
15	*(9,6)*	*8,5*	*5,1*

Tab. 3.6: Distanzen zur Bestimmung des dritten Zentroids. Die grau hinterlegten Zellen zeigen die Minima gemäß (3.3). Die dunkelgrau hinterlegten Zellen zeigen die Maxima gemäß (3.4). Da kein eindeutiges Maximum existiert, wird zufällig das zu Punkt *(2,1)* gehörige gewählt. Dies wird durch die stärkere Umrandung der Zelle angezeigt.

Tab. 3.6 zeigt die Abstände zwischen den Punkte aus der Menge der Zentroide und den übrigen Werten. Die grau hinterlegten Zellen zeigen die Minima gemäß (3.3). Das maximale Minimum ist hier gemäß (3.4) nicht eindeutig zu bestimmen. Sowohl der Punkt *(2,1)* als auch der Punkt *(8,7)* haben nach (3.4) einen maximalen Minimumabstand von *6,0* zu den bisherigen Zentroiden. In der Tabelle sind diese Werte durch die dunkelgrau hinterlegten Zellen gekennzeichnet. Solche Probleme sind

nicht auf dieses Beispiel beschränkt, sondern können auch bei der Clusterung von Luftaufnahmen auftreten. Das Problem tritt deshalb so häufig auf, weil in einem beschränkten aber vor allem diskreten Parameterraum gearbeitet wird. Nach der Definition des Algorithmus ist keiner der Werte mit dem gleichen Maximum eindeutig zu präferieren. Daher wird zur Lösung des Problems einer der Punkte zufällig ausgewählt. In Abschnitt 3.8 wird dieses Auswahlverfahren im Detail vorgestellt. Dort wird auch die iterative Programmumsetzung präsentiert. Hier wird der Punkt *(2,1)* als dritter Zentroid ausgewählt. Dies wird in Tab. 3.6 durch die hervorgehobene Umrandung der Zelle gekennzeichnet. Somit ergibt sich Z als:

$$Z = \{ (1,9) ; (8,1) ; (2,1) \}.$$

Um den letzten der vier gesuchten Werte zu identifizieren, wird der Abstandsvektor zwischen der dreielementigen Menge der Zentroide und den übrigen fünf Werten berechnet:

Index	*Punkte*	*Z*
i	P_i	*{ (1,9) ; (8,1) ; (2,1) }*
Z	*{ (1,9) ; (8,1) ; (2,1) }*	*0*
3	*(3,1)*	*1,0*
4	*(3,3)*	*2,2*
5	*(3,8)*	*2,2*
6	*(4,8)*	*3,2*
7	*(5,5)*	*5,0*
8	*(6,4)*	*3,6*
9	*(6,6)*	*5,4*
10	*(7,2)*	*1,4*
12	*(8,5)*	*4,0*
13	*(8,7)*	*6,0*
14	*(9,3)*	*2,2*
15	*(9,6)*	*5,1*

Tab. 3.7: Distanzvektor gemäß (3.3) zur Bestimmung des vierten Zentroids

Der letzte Zentroid ist nach Tab. 3.7 eindeutig gegeben durch *(8,7)*. Somit beinhaltet die Menge der Zentroide die Werte:

$$Z = \{ (1,9) ; (8,1) ; (2,1) ; (8,7) \}.$$

Dieses Beispiel ist in der folgenden Abb. 3.3 grafisch dargestellt. Die eingefärbten Punkte sind die ausgewählten Zentroide. Die mit z_1 und z_2 gekennzeichneten Punkte sind die im ersten Iterationsschritt berechneten Zentroide. Der im zweiten Schritt berechnete dritte Zentroid wird mit z_3 bezeichnet. Der vierte und letzte Zentroid hat die Bezeichnung z_4. Die weißen Punkte sind die übrigen Werte, die anhand der Zentroide zu klassifizieren sind. Die Ellipsoide um die jeweiligen Zentroide schließen die Werte ein, die ihnen durch die Klassifikation zugeordnet werden:

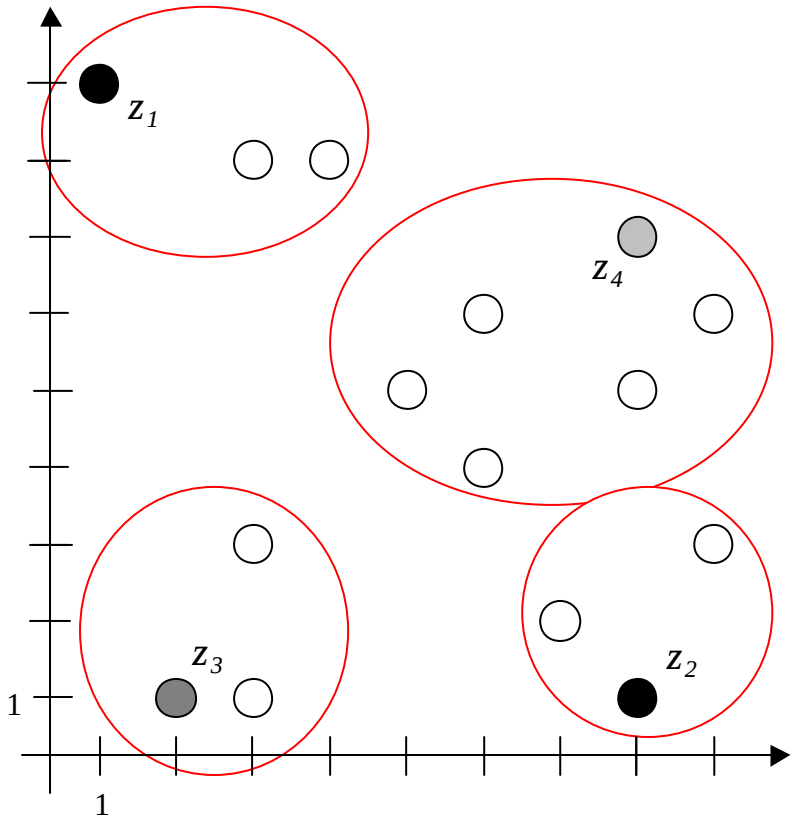


Abb. 3.3: Visualisierung der Funktionsweise von Maximum Linkage an einem Beispiel gemäß Tab. 3.5

3.5 Vorteile des Maximum-Linkage-Algorithmus

Der Maximum-Linkage-Algorithmus besitzt folgende Vorteile:

1. Zur Bestimmung der Zentroide für q Cluster werden genau $(q-1)$ Iterationen benötigt. In jedem Rekursionsschritt wird genau ein Zentroid bestimmt. Im ersten Schritt werden jedoch zwei Zentroide ausgewählt. Das führt zu einer enormen Laufzeitverbesserung gegenüber allen anderen Verfahren. Dieser Zeitaspekt wird in Abschnitt 5.1 behandelt.
2. Die q Zentroide werden aus der Menge der real vorliegenden Werte ausgewählt. Dadurch wird der Parameterraum stark eingeschränkt. Das vereinfacht das Verfahren in jedem einzelnen Schritt. Dieser eingeschränkte Aufwand wirkt sich positiv auf die Laufzeit des Algorithmus aus.
3. Die Bedingung, nach der die Zentroide bestimmt werden, entspricht fast dem Kriterium des minimalen mittleren Abstands zwischen den Zentroiden. Betrachtungen zu den Abweichungen zwischen den algorithmisch bestimmten Zentroiden und den durch das Kriterium des minimalen mittleren Abstands geforderten folgen in Abschnitt 3.6 .
4. Nach Bestimmung der Zentroide werden die übrigen Werte anhand der Zentroide klassifiziert. Durch Mittelwertbildung in den einzelnen Klassen entstehen Klassenrepräsentanten. Die Abweichung zwischen den Zentroiden und den Klassenrepräsentanten ist i. Allg. gering. Treten dennoch große Abweichungen auf, ist i. Allg. die Anzahl q der Klassen zu klein gewählt worden. Durch Betrachtung dieser Abweichungen lassen sich somit Rückschlüsse darauf ziehen, ob die Anzahl q der Klassen zu klein gewählt wurde.
5. Ein Nebeneffekt besteht darin, dass die Zentroide einer Clusterung mit $(q-1)$ Klassen auch in einer Clusterung mit q Klassen enthalten sind. Dieser Effekt entsteht durch die hierarchische Auswahl der Zentroide. Das hat den Vorteil, dass aus einem Satz mit q Zentroiden diverse Clusterungen mit 1 bis q Klassen erstellt werden können. Dazu muß lediglich die Reihenfolge bekannt sein, in der die Zentroide berechnet wurden. Durch Klassifizierung der Werte zu den jeweils ersten der 1 bis q Zentroide erhält man die entsprechenden Clusterungen mit 1 bis q Klassen. Diesen Nebeneffekt können Verfahren, die auf dem k-Mittelwert-Algorithmus oder auf neuronalen Netzen basieren, nicht bieten. Wie in Abschnitt 3.9 gezeigt wird, bietet sich damit die Möglichkeit, einen Mechanismus zu konstruieren, der eine Entscheidungshilfe für die gesuchte Anzahl der zur Clusterung notwendigen Klassen liefert.

3.6 Genauigkeit des Maximum-Linkage-Algorithmus

Der Maximum-Linkage-Algorithmus soll dazu dienen, eine Auswahl von Datenpunkten gemäß der Maximum-Linkage-Bedingung (3.1) zu bestimmen. Daher muss überprüft werden, ob das algorithmische Verfahren dies tatsächlich leistet.
Die Zentroide werden beim Maximum Linkage hierarchisch bestimmt. Deshalb genügen die durch Maximum Linkage ausgewählten Zentroide nicht in jedem Iterationsschritt der Bedingung (3.1). Um die Abweichung zwischen dem Maximum-Linkage-Algorithmus und der Maximum-Linkage-Bedingung zu untersuchen, wird ein Vergleich zwischen den durch die Maximin-Bedingungen beider Verfahren gegeben Größen angestrebt. Der durch die Maximum-Linkage-Bedingung (3.1) implizierte Wert E_B ist folgendermaßen gegeben:

$$E_B = \max_{\{1^*,\ldots,q^*\}\subset\{1,\ldots,r\}} \min_{i,j\in\{1^*,\ldots,q^*\}} \| P_i - P_j \|_2 . \qquad (3.5)$$

Um das Gegenstück zu E_B beim Maximum-Linkage-Algorithmus zu bestimmen, sind einige Vorüberlegungen nötig. Da es sich um eine hierarchische Auswahl der Zentroide handelt, entsteht in jedem Iterationsschritt eine zu (3.5) ähnliche Größe. Die Unterschiede bestehen lediglich in der Einschränkung der Indexmengen, über die das Maximum bzw. Minimum gebildet werden. Diese Größen werden im Folgenden in Anlehnung an Single Linkage als Eingangsniveau des Zentroids ($E_{Zentroid}$) bezeichnet. Das Eingangsniveau eines Zentroids ist der Wert der Maximin-Bedingung, zu dem er ausgewählt wird.
Die ersten beiden Zentroide werden in einem Schritt ausgewählt. Das Eingangsniveau dieser beiden Zentroide wird mit E_2 bezeichnet und ist gegeben durch:

$$E_2 = \max_{i,j\in\{1,\ldots,r\}} \| P_i - P_j \|_2 .$$

Das Eingangsniveau des *i*-ten Zentroids, $i = 3,\ldots,q$, ist gegeben durch

$$E_i = \max_{i\in\{1,\ldots,r\}\setminus\{1^*,\ldots,(i-1)^*\}} \min_{j\in\{1^*,\ldots,(i-1)^*\}} \| P_i - P_j \|_2 , \; i = 3,\ldots,q .$$

Aus dem Konstruktionsprinzip des Maximum-Linkage-Algorithmus ist folgende Bedingung abzuleiten:

$$E_q \le E_{q-1} \le \ldots \le E_2 . \qquad (3.6)$$

Zum Beweis dieser Behauptung ist zu zeigen: $\forall\, i \in \{3,\ldots,q\}:\; E_i \le E_{i-1}$.

Betrachte:

$$E_i = \max_{i \in \{1,\ldots,r\} \setminus \{1^*,\ldots,(i-1)^*\}} \min_{j \in \{1^*,\ldots,(i-1)^*\}} \| P_i - P_j \|_2 , i \in \{3,\ldots,q\}.$$

Indem das Minimum über einen Wert weniger gebildet wird, wird es höchstens größer. Wird das wirkliche Minimum weggelassen, wird E_i tatsächlich größer; andernfalls bleibt es gleich. Wird der Index $(i-1)^*$ bei der Minimumbildung weggelassen, ergibt sich:

$$\max_{i \in \{1,\ldots,r\} \setminus \{1^*,\ldots,(i-1)^*\}} \min_{j \in \{1^*,\ldots,(i-1)^*\}} \| P_i - P_j \|_2 \leq \max_{i \in \{1,\ldots,r\} \setminus \{1^*,\ldots,(i-1)^*\}} \min_{j \in \{1^*,\ldots,(i-2)^*\}} \| P_i - P_j \|_2 .$$

Die rechte Seite der Ungleichung lässt sich weiter abschätzen, indem das Maximum über einen Wert mehr gebildet wird. Führt der neue Wert zum realen Maximum, wird es echt größer; andernfalls bleibt es gleich. Als Index des zusätzlichen Wertes wird $(i-1)^*$ gewählt. Das Maximum wird dann über $\{1,\ldots,r\} \setminus \{1^*,\ldots,(i-2)^*\}$ anstatt über $\{1,\ldots,r\} \setminus \{1^*,\ldots,(i-1)^*\}$ gebildet. Damit ergibt sich:

$$\max_{i \in \{1,\ldots,r\} \setminus \{1^*,\ldots,(i-1)^*\}} \min_{j \in \{1^*,\ldots,(i-2)^*\}} \| P_i - P_j \|_2 \leq \max_{i \in \{1,\ldots,r\} \setminus \{1^*,\ldots,(i-2)^*\}} \min_{j \in \{1^*,\ldots,(i-2)^*\}} \| P_i - P_j \|_2 .$$

Eine Indextransformation von i zu $i-1$ ergibt:

$$\max_{i-1 \in \{1,\ldots,r\} \setminus \{1^*,\ldots,(i-2)^*\}} \min_{j \in \{1^*,\ldots,(i-2)^*\}} \| P_{i-1} - P_j \|_2 = E_{i-1}.$$

Damit ist Behauptung (3.6) gezeigt. ð

Nach (3.6) ist der minimale Abstand beim Maximum-Linkage-Algorithmus durch E_q gegeben. Daher ist zur Untersuchung der Abweichung zwischen der Maximum-Linkage-Bedingung (3.1) und dem Maximum-Linkage-Algorithmus ein Vergleich zwischen E_q und E_B notwendig. Es gilt:

$$\begin{aligned} E_B &= \max_{\{1^*,\ldots,q^*\} \subset \{1,\ldots,r\}} \min_{i,j \in \{1^*,\ldots,q^*\}} \| P_i - P_j \|_2 \\ &\geq \max_{q \in \{1,\ldots,r\} \setminus \{1^*,\ldots,(q-1)^*\}} \min_{j \in \{1^*,\ldots,(q-1)^*\}} \| P_i - P_j \|_2 = E_q \quad . \end{aligned} \qquad (3.7)$$

Ungleichung (3.7) gilt, weil auf der linken Seite die Maximin-Bedingung unter allen möglichen Alternativen gesucht wird, wohingegen auf der rechten Seite eine Einschränkung der Alternativen vorliegt.

Des Weiteren stellt sich die Frage, wie groß die maximale Abweichung ist. Um den Unterschied zwischen Bedingung (3.1) und dem Maximum-Linkage-Algorithmus zu quantifizieren, werden beide Seiten der Gleichung (3.7) abgeschätzt. Betrachte dazu den günstigsten Fall für Bedingung (3.1), d. h. das Auftreten des theoretisch größten Maximums. Dieser Fall tritt ein, wenn zum einen die zwei am weitesten entfernten Punkte als Zentroide ausgewählt sind und zum anderen alle Distanzen zwischen den Zentroiden gleich sind. Dann ist diese Distanz gegeben durch:

$$\frac{1}{q-1} \max_{i,j \in \{1,\ldots,r\}} \left\| P_i - P_j \right\|_2 . \qquad (3.8)$$

Der ungünstigste Fall beim Maximum Linkage tritt auf, wenn der letzte gefundene Zentroid dicht bei einem der schon gefundenen liegt. Theoretisch gibt es Konstellationen von Punktmengen $O = \{P_1,\ldots,P_r\}$ und Clusterungen mit Maximum Linkage $C_{MaxL}(O)$, bei denen der letzte Zentroid beliebig dicht bei einem zuvor ausgewählten liegt. Es gilt also:

$$\exists\ O = \{P_1,\ldots,P_r\}\ \exists\ C_{MaxL}(O)\ \exists\ e \in \mathbb{R}, e > 0\ :\ E_q = e . \qquad (3.9)$$

Mit den Überlegungen aus (3.8) und (3.9) lässt sich (3.7) folgendermaßen abschätzen:

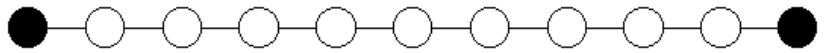
$$\frac{1}{q-1} \max_{i,j \in \{1,\ldots,r\}} \left\| P_i - P_j \right\|_2 \geq E_B \geq E_q > 0 .$$

Damit lässt sich nun die maximale Abweichung zwischen E_q und E_B folgendermaßen abschätzen:

$$E_B - E_q < \frac{1}{q-1} \max_{j_1, j_2 \in \{1,\ldots r\}} \left\| P_{j_1} - P_{j_2} \right\|_2 - 0$$
$$= \frac{1}{q-1} \max_{j_1, j_2 \in \{1,\ldots r\}} \left\| P_{j_1} - P_{j_2} \right\|_2$$

Obige Ungleichung zeigt die Beschränkung der Abweichung zwischen Bedingung und Algorithmus in Abhängigkeit von der Klassenanzahl q und dem maximalen Abstand der Ausgangswerte. Außerdem geht daraus hervor, dass die Abweichung mit zunehmender Anzahl an Klassen geringer wird.

Beispiel:
In einem eindimensionalen Merkmalsraum liegen elf äquidistanten Punkte mit Abstand eins. Aus dieser Menge sollen Zentroide mit dem Maximum-Linkage-Algorithmus ausgewählt werden. Die in der Darstellung abgebildeten schwarzen Punkte kennzeichnen die im ersten Schritt ermittelten Zentroide:



Die so entstandene erste Konstellation an Zentroiden genügt der Maximum-Linkage-Bedingung (3.1). Wird zusätzlich ein dritter Zentroid nach Maximum Linkage ausgewählt, so ergibt sich folgendes Bild:

Auch in diesem Schritt ist Bedingung (3.1) erfüllt. Der vierte Zentroid ist nicht mehr eindeutig bestimmbar. Die folgende Abbildung zeigt die Punkte, die als mögliche Kandidaten in Frage kommen. Diese Punkte sind hellgrau markiert.

Eine mögliche Konstellation nach der beschriebenen zufälligen Auswahl zeigt die folgende Abbildung. Jedoch ist dabei Bedingung (3.1) nicht mehr erfüllt.

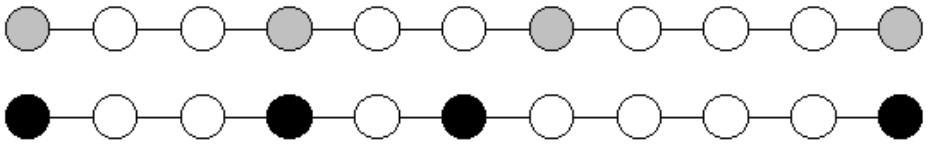
Betrachtet man die dem Kriterium (3.1) zu Grunde liegende Abstandsberechnung, so ergibt sich unter obiger Annahme der äquidistanten Abstände:

$$\min_{i,j \in \{1^*,\ldots,4^*\}} \| P_i - P_j \|_2 = 2 , \quad \{1^*,\ldots,4^*\} \text{ fest} .$$

Die Maximierung der Punkteauswahl über alle auftretenden Minima ergibt:

$$\max_{\{1^*,\ldots,4^*\}\subset\{1,\ldots,11\}} \min_{i,j\in\{1^*,\ldots,4^*\}} \left\| P_i - P_j \right\|_2 = 3$$

Es gibt drei möglich Konstellationen, die die Bedingung (3.1) erfüllen. Eine davon ist in der folgenden Abbildung durch die grau markierten Punkte dargestellt. Zum Vergleich steht darunter die von Maximum Linkage ausgewählte Reihe mit den schwarzen Punkten:



Punkteauswahlen nach dem Maximum-Linkage-Algorithmus genügen damit nicht in jedem Schritt der Bedingung (3.1). Wenn ein weiterer Zentroid hinzugezogen wird, ist die Bedingung (3.1) aber wieder erfüllt, egal welcher der beiden möglichen Kandidaten gewählt wird. Eine Wahl könnte z. B. so aussehen:

Das Beispiel zeigt die Möglichkeit der Abweichung zwischen Maximum-Linkage-Algorithmus und der Maximum-Linkage-Bedingung. Die Abweichung ist aber nicht sehr groß. Außerdem entstehen im Verlauf der Iteration immer wieder Konstellationen, die die Bedingung erfüllen. Wenn entsprechend viele Zentroide ausgewählt werden, wird die Abweichung immer geringer.

Der Maximum-Linkage-Algorithmus ist dennoch ein gutes Verfahren, um Zentroide in enger Anlehnung an (3.1) zu berechnen, weil die Abweichung beschränkt ist. Zum anderen wird diese Beschränkung bei steigender Klassenzahl monoton stärker.

3.7 Berücksichtigung der Häufigkeitsverteilung der Pixel

Das Hauptproblem bei der pixelbasierten Clusterung von Bildern ist die Wahl der „richtigen“ Zentroide. Der Maximum-Linkage-Algorithmus ist in der Lage, die Zentroide geeignet aus der Menge der vorhandenen Pixel auszuwählen.

Cressie (1993) zeigt, dass die Berücksichtigung der Häufigkeitsverteilung der Pixel bei Clusterungsproblemen wichtig ist. In vielen Ansätzen wird die empirische Häufigkeitsverteilung implizit berücksichtigt. Beim k-Mittelwert-Verfahren geschieht dies, indem bei der Mittelwertbildung in den Klassen die Häufigkeiten der einzelnen Pixel in den Klassen berücksichtigt wird. Bei neuronalen Netzen wird dies erreicht, indem ein Wert entsprechend seiner empirischen Häufigkeit als Lerneingabe geschaltet wird. Die begriffliche Erklärung zu neuronalen Netzen folgt im nächsten Kapitel.
Der Maximum-Linkage-Algorithmus berücksichtigt in seiner einfachsten Form die Häufigkeitsverteilung der verschiedenen Pixel in den Ausgangsdaten nicht. Dadurch kann es passieren, dass zu viele Zentroide am „Rand“ der Häufigkeitsverteilung ausgewählt werden. Die dicht besetzten Regionen bleiben unterrepräsentiert. Das hat zur Folge, dass sehr viele Pixel bei der anschließenden Klassifizierung in „falsche“ Klassen einsortiert werden. Um das Problem zu lösen, wird die Häufigkeitsverteilung der Pixel in die Auswahl der Zentroide mit einbezogen. Dazu werden die Abstände d_{ij} aus (3.2) und (3.3) mit der Häufigkeit der zugehörigen Pixel h_{ij} gewichtet. Dabei berechnet sich die Häufigkeit h_{ij} durch:

$$h_{ij} = H(P_i) + H(P_j) .$$

Die Funktion $H(P)$ gibt die Anzahl der Pixel an, die den selben (Farb-) Wert wie der Pixel P besitzen. Angenommen, im Beispiel aus Abschnitt 3.4, Tab. 3.5, würde der Punkt $(2,1)$ mit der Häufigkeit 10 und der Punkt $(3,8)$ mit der Häufigkeit 5 auftreten, so würde die Distanz $d_{ij} = 7{,}1$ zwischen den Punkten mit dem Faktor $h_{ij} = 10 + 5 = 15$ gewichtet. Dadurch ändert sich (3.2) zu

$$\Delta_h = (h_{ij} \cdot d_{ij}) , \; d_{ij} = \left\| P_i - P_j \right\|_2 , \quad i, j = 1,...,r , \qquad (3.10)$$

und (3.3) ergibt sich als:

$$d_{iQ_h} = h_i \cdot \min_{k \in Q} \left\| P_i - P_k \right\|_2 , \; h_i = H(P_i), \; i \in R . \qquad (3.11)$$

Die Verwendung von (3.10) und (3.11) statt (3.2) und (3.3) im Maximum-Linkage-Algorithmus führt dazu, dass nur Zentroide aus den dicht besetzten Bereichen der Verteilung gefunden werden. Dadurch wird die Forderung 2 aus Abschnitt 2.3 nicht mehr erfüllt. Ein Lösungsansatz besteht darin, zwei verschiedene Mengen von Zentroiden Z_A und Z_B zu berechnen. Die Menge Z_A wird unter Verwendung von (3.2) und (3.3) ungewichtet berechnet, Z_B mit Hilfe von (3.10) und (3.11) mit Gewichtung. Die Vereinigung beider Mengen ergibt die zur Klassifizierung nötigen Zentroide:

$$Z = Z_A \cup Z_B .$$

Die Menge Z_A enthält die Zentroide, anhand derer die seltenen Elemente heraus gearbeitet werden können. Die Zentroide der Menge Z_B repräsentieren die dicht besetzten Bereiche der Verteilung. Im Sinne des SSW-Clusterungskriteriums (2.2) entsteht so eine gute Clusterung. Es ist sicherzustellen, dass die Kardinalität der vereinigten Menge q entspricht:

$$|Z_A \cup Z_B| \stackrel{!}{=} q .$$

Dazu muss die Summe der Kardinalität von Z_A und Z_B gerade q ergeben, also $|Z_A| + |Z_B| \dot{=} q$. Daraus folgt die Disjunktheit der Mengen Z_A und Z_B.
Bei der praktischen Umsetzung ist diese Eigenschaft leicht sicherzustellen. Der Algorithmus wird dazu so angelegt, dass nicht zwei verschiedene Mengen von Zentroiden berechnet werden, sondern nach einer Reihe von Iterationsschritten die Distanzberechnung von (3.10) bzw. (3.11) auf (3.3) umgestellt wird.

Es bleibt zu klären, wie viele Zentroide in den jeweiligen Mengen Z_A und Z_B zu berechnen sind. Zur Beantwortung sei zunächst auf Abb. 3.4 hingewiesen. In dieser sind zwei mögliche Randdichten einer Komponente von Häufigkeitsverteilungen der RGB-Farbwerte eines Bildes skizziert.

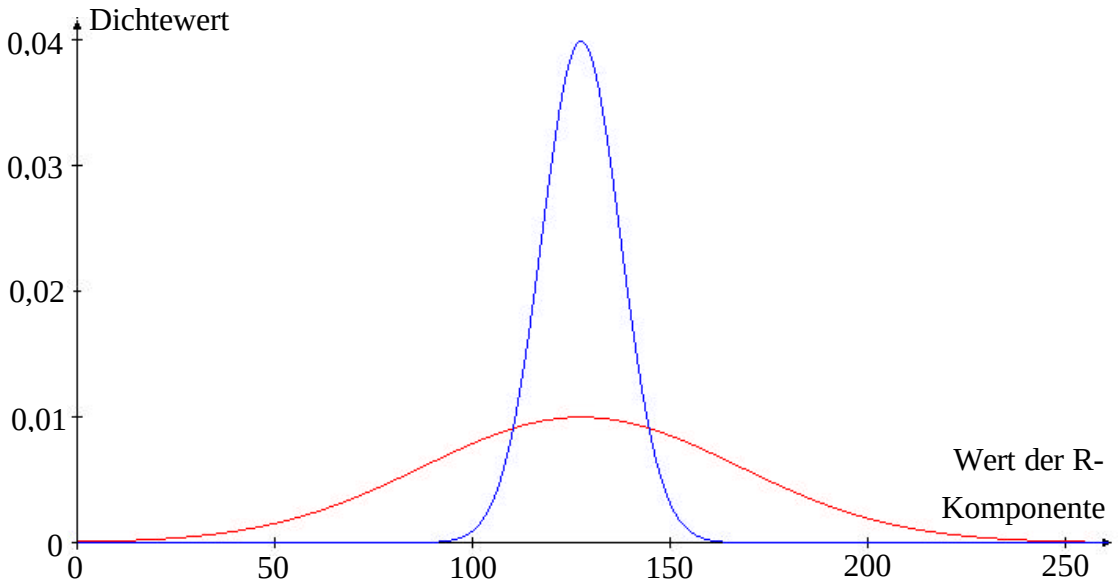


Abb. 3.4: Skizzierte Randdichten der RGB-Farbwerte zweier Bilder. Die flache Kurve entspricht einer Randdichte mit großer Varianz; die steile Kurve einer Randdichte mit kleiner Varianz.

Angenommen, es liegt eine Verteilung gemäß der „steileren“ Dichte vor. In diesem Fall sollten mehr Zentroide ungewichtet als gewichtet gewählt werden, weil es nur wenige, dicht zusammenliegende Werte mit großer Häufigkeit gibt. Daher repräsen-

tieren wenige Zentroide diesen Bereich des Merkmalsraumes hinreichend. Die meisten Zentroide können darauf verwandt werden, die übrigen Bereiche des Merkmalsraumes zu beschreiben. Würden viele Zentroide mit Gewichtung gewählt, könnte nur ein sehr kleiner Teil des Merkmalsraumes repräsentiert werden.
Im Falle einer Verteilung gemäß der „flachen“ Dichte, sind viele Zentroide gewichtet zu wählen. Die Differenz zwischen den Häufigkeiten der auftretenden Werte ist gering. Daher ist der Unterschied zwischen gewichteter und ungewichteter Auswahl auch gering. Durch Einbeziehung der Gewichtung lässt sich eine bessere Anpassung an die Ausgangsdaten erreichen. Schwach besetzte Bereiche des Merkmalsraumes können nicht unterrepräsentiert werden, da die Besetzung aller Bereiche sehr ähnlich ist. Die Clusterung kann sich unter Einbeziehung der Häufigkeiten nur verbessern. Auf Grund dieser Erkenntnis liegt es nahe, die Anzahl der gewichtet bzw. ungewichtet zu wählenden Zentroide auf Grund eines varianzbasierten Parameters festzulegen. Für die Definition einer solchen Größe k sind folgende Gegebenheiten zu berücksichtigen:
Der Parameterraum des zu Grunde liegenden RGB-Modells ist auf $\Omega = \{0,1,...,255\}^3$ beschränkt. Die Varianzbetrachtung der einzelnen Komponenten der RGB-Vektoren reicht für die folgenden Überlegungen aus. Durch Berücksichtigung der Kovarianz zwischen den Komponenten der RGB-Vektoren werden gegenüber der Varianzbetrachtung der einzelnen Komponenten keine zusätzlichen Erkenntnisse gewonnen. Des Weiteren sind die einzelnen Komponenten der RGB-Vektoren alle von der selben Größenordnung. Diese Überlegungen und die Annahme, dass die Varianz eine geeignete Größe für die Definition von k ist, führen zu folgender Definition:

$$k = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{3}\left(P_{ik} - \frac{1}{n}\sum_{j=1}^{n}P_{jk}\right)^2 . \qquad (3.12)$$

P_{i1} bezieht sich hier auf die R-Komponente, P_{i2} auf die G-Komponente und P_{i3} auf die B-Komponente des i-ten Wertes im RGB-Modell. Unter Berücksichtigung des beschränkten Parameterraumes ist auch k beschränkt:

$$k \in [\ 0\ ;\ 48768{,}75\ ].$$

Die untere Grenze wird erreicht, wenn alle (Farb-)Werte gleich sind. Die obere Grenze wird erreicht, wenn es genau zwei Vektoren gibt, deren Komponenten den maximal möglichen Abstand haben, z. B.:

$$P_1 = (\ 0\ ,\ 255\ ,\ 0\ ) \quad \text{und} \quad P_2 = (\ 255\ ,\ 0\ ,\ 255\ ).$$

Für die Berechnung der Anzahl gewichtet bzw. ungewichtet zu wählender Zentroide, ist eine Funktion gesucht. Dazu wird zunächst eine Funktion bestimmt, die die An-

zahl gewichtet zu wählender Zentroide berechnet. Daraus lässt sich über die Gesamtanzahl gesuchter Zentroide die Anzahl ungewichtet zu wählender Zentroide bestimmen. Diese Funktion hängt von k und der Gesamtanzahl q gesuchter Zentroide ab.
Die einfachste Form einer solchen Funktion ist linear. Die Steigung definiert sich über zwei Fixpunkte. Der eine ist $(0\ ;\ 0)$, da bei $k = 0$ keine Zentroide gewichtet bestimmt werden. Der andere ist $(48768{,}75\ ;\ q)$, weil bei maximalem k alle Zentroide gewichtet berechnet werden.
Für kleines k ist zu gewährleisten, dass nur eine kleine Anzahl an Zentroiden gewichtet bestimmt wird. Im Umkehrschluss ist es sinnvoll, wenn für großes k viele Zentroide gewichtet gewählt werden. Um die Anzahl von gewichteten Zentroiden für kleines k zu berechnen, muss die gesuchte Funktion für kleines k unterhalb der linearen Verbindung zwischen den Extrempunkten verlaufen. Analoges gilt für die Anzahl gewichteter Zentroide. Dort soll die Funktion oberhalb der Geraden zwischen den Extrempunkten liegen.
Als Basis für eine solche Funktion bietet sich ein Ausschnitt aus einer Sinusfunktion an. Diese wird auf die gewünschten Definitions- und Wertebereiche beschränkt. Die gesuchte Funktion sei definiert gemäß:

$$f_W(k,q) = \left\lfloor \left(sin\left(\frac{pk}{48768.75} - \frac{p}{2} \right) + 1 \right) \frac{q}{2} \right\rfloor . \qquad (3.13)$$

Damit bestimmt sich die Anzahl der ungewichtet zu wählenden Zentroide durch

$$f_{UW}(k,q) = q - f_W(k,q) . \qquad (3.14)$$

Abb. 3.5 zeigt die Funktion der gewichtet zu wählenden Zentroide für $q = 10$ Zentroide. Die Darstellung beinhaltet sowohl das Original, wie in (3.13), als auch eine Version ohne Gaußklammer. Es zeigt sich, dass diese Funktion den oben dargestellten Überlegungen genügt.

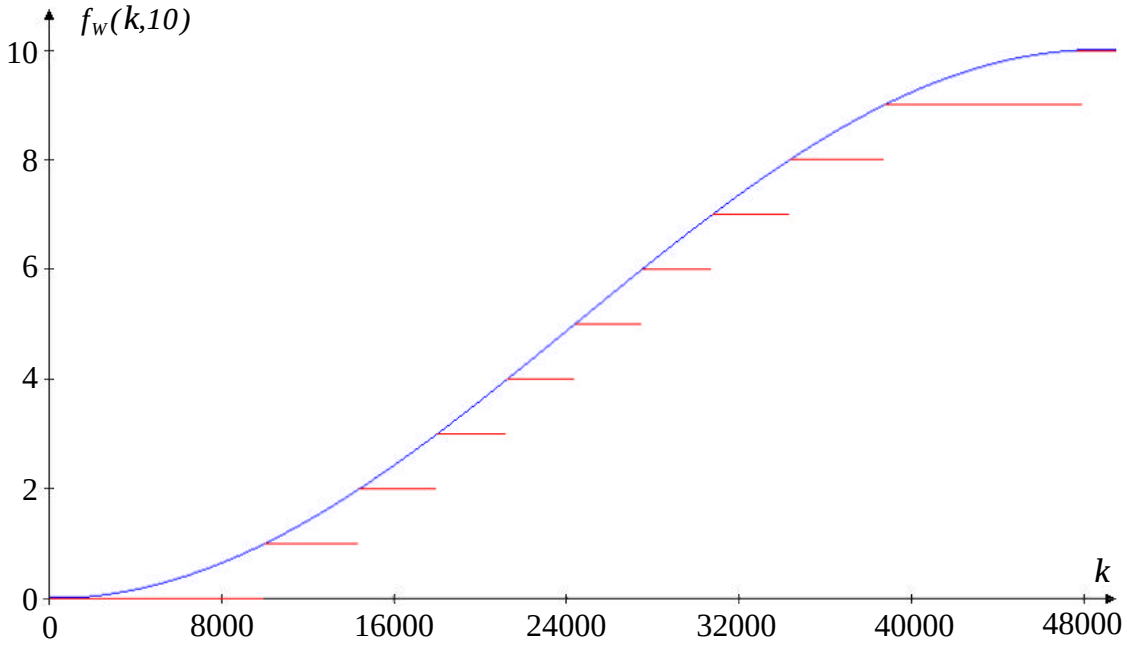


Abb. 3.5: Darstellung der Funktion gewichtet zu wählender Zentroide. Die Treppenfunktion stellt die eigentliche Funktion wie in (3.13) dar. Der stetige Graf zeigt (3.13) ohne die Gauß-klammer.

3.8 Umsetzung des Maximum-Linkage-Algorithmus

Der Maximum-Linkage-Algorithmus ist in einem C++-Programm implementiert. Dieses Programm wird genutzt, um in kurzer Zeit Clusterungen eines Bildes zu berechnen und die Ergebnisse der Clusterung in vielfältiger Weise darstellen zu können. Eine der Besonderheiten liegt in der farbigen Darstellung des Ergebnisses. Unter der Voraussetzung, dass die Clusterung „gut“ ist, wird eine visuelle inhaltliche Interpretation möglich. Wann eine Clusterung als „gut“ zu bezeichnen ist, wird im Rahmen des Vergleichs der Methoden in Kapitel 5 behandelt. Das Programm heißt PicAna (**Pic**ture **Ana**lysis = Bildanalyse). Die aktuelle Version läuft auf den meisten MS Windows™ basierten PC-Systemen. Das Programm benötigt nur wenig Festplattenplatz; es wird jedoch empfohlen, einen möglichst großen Hauptspeicher (RAM) zu haben (Minimalempfehlung: 64 MB RAM). Steht deutlich mehr Speicher als die Minimalempfehlung zur Verfügung, so erfolgt die Bildverarbeitung erheblich schneller.

Bei der Umsetzung des Maximum-Linkage-Algorithmus in ein Rechnerprogramm ist ein besonderes Problem zu lösen. Die zwei zentralen Berechnungsschritte des Algorithmus sind die Distanzberechnungen und die Auswahl des maximalen Minimalwertes.

Die Ergebnisdarstellung der Distanzberechnungen erfolgt gemäß der Formeln (3.2) und (3.3) bzw. (3.10) und (3.11) in Matrix- bzw. Vektorform. Dadurch müssen bei der Programmumsetzung große Datenmengen temporär verwaltet werden. Die Folge: Auslagerung der Daten auf die Festplatte. Solche Auslagerungsprozesse verlängern die Laufzeiten eines Programms erheblich. Wenn die Datenmenge innerhalb des Hauptspeichers verwaltet werden kann, reduziert sich die Laufzeit deutlich.
Dies kann durch eine entsprechende algorithmische Umsetzung gewährleistet werden. Bei der Umsetzung werden die Einträge der Distanzmatrix iterativ berechnet. Das bietet die Möglichkeit, die Distanzen nicht in einer Matrix zu speichern, sondern das Berechnen der Distanzen und das Auswählen des entsprechenden maximalen Minimalwertes gleichzeitig vorzunehmen. Dann muss keine Distanzmatrix, sondern nur die aktuell berechnete Distanz und der bis dahin aufgetretene maximale Minimalwert gespeichert werden. Des Weiteren wird der zum maximalen Minimalwert gehörige Punkt gespeichert, wobei im ersten Schritt des Maximum-Linkage-Algorithmus zwei Punkte zu speichern sind.
In jedem Iterationsschritt wird die aktuell berechnete Distanz mit dem bisher aufgetretenen maximalen Minimum verglichen. Ist die aktuelle Distanz größer als die bisherige, wird diese durch die aktuelle Distanz ersetzt. Zusätzlich wird der zum bisherigen maximalen Minimum gehörige festgehaltene Punkt durch den zur aktuellen Distanz gehörigen überschrieben. Somit wird während der Berechnung nur ein Minimum an temporären Daten verwaltet. Dadurch können auch bei Rechnern mit wenig Hauptspeicher gute Laufzeiten erzielt werden.
So effektiv dieses Vorgehen auch ist, wird dadurch ein neues Problem aufgeworfen. Es kommt vor, dass die aktuell berechnete Distanz und das momentan festgehaltene Maximum identisch sind. Dabei stellt sich die Frage, welcher von den zugehörigen Punkten mit der gleichen Maximaldistanz zu bevorzugen ist. Wie schon in Beispiel 3.4 erwähnt, soll beim Auftreten mehrerer Maxima einer der zugehörigen Punkte zufällig ausgewählt werden. Die Auswahlwahrscheinlichkeit soll einer Gleichverteilung genügen. Tritt dieser Fall zum ersten mal auf, ist einer der Punkte mit Wahrscheinlichkeit ½ zu wählen. Bei größeren Clusterproblemen kommt es durchaus vor, dass mehrmals hintereinander derselbe Maximalwert gefunden wird. Dabei sollen alle zugehörigen Punkte die gleiche Chance haben, als der letztlich verwendete Zentroid ausgewählt zu werden. Es kann also in den folgenden Fällen nicht mit Wahrscheinlichkeit ½ zwischen dem aktuellen und bisherigen Punkt ausgewählt werden.
Es sei dazu die folgende Skizze betrachtet, bei der *s* die Wiederholung kennzeichnet, in der dasselbe Maximum gefunden wird:

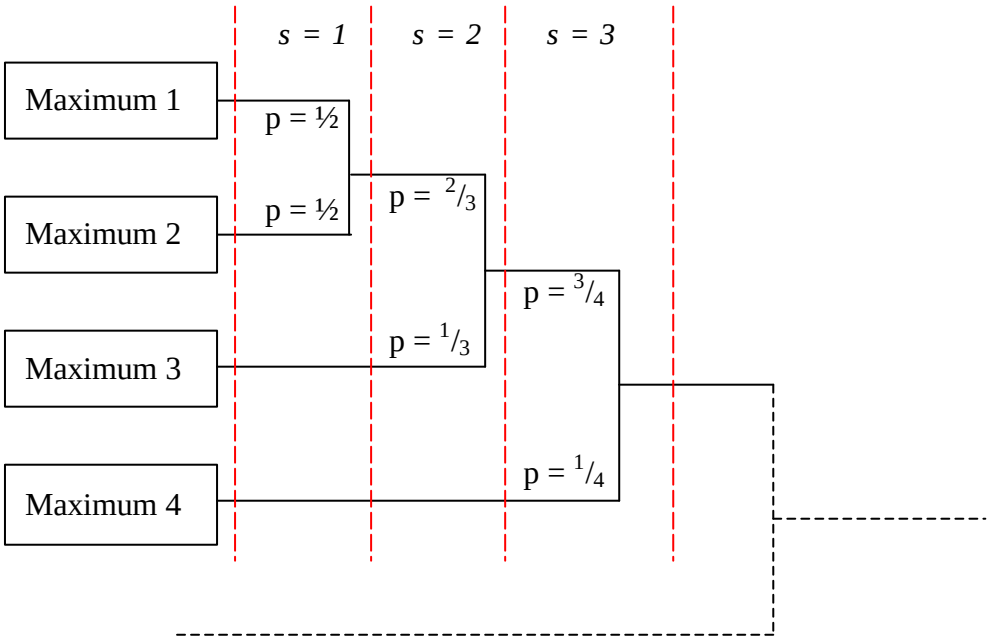


Abb. 3.6: Baumdiagramm zur Auswahlwahrscheinlichkeit p mehrfach auftretender Maxima

Um das Problem zu lösen, muss der Wert s festgehalten werden. Dieser zeigt an, wie oft die Maximaldistanz bisher gefunden wurde. Tritt der Maximalwert also zum zweiten Mal auf, beträgt $s = 1$. Abb. 3.6 zeigt, mit welcher Wahrscheinlichkeit in Abhängigkeit von s der bisher festgehaltene Punkt ersetzt wird bzw. zu behalten ist. Jeder Punkt wird letztlich mit Wahrscheinlichkeit $1/s_{max}$ ausgewählt. Dabei entspricht s_{max} der jeweiligen Gesamtanzahl der Maxima, die in einem Iterationsschritt auftreten.

Allgemein sind die Wahrscheinlichkeiten in Schritt s wie folgt definiert:

$$P^{(s)}(\text{„bisherige Werte beibehalten“}) = \frac{s}{s+1}$$

und

$$P^{(s)}(\text{„neu berechnete Werte verwenden“}) = \frac{1}{s+1}\ .$$

Damit ergeben sich für die vier Maxima aus Abb. 3.6 folgende Auswahlwahrscheinlichkeiten:

$$P(\text{„Wahl vom Maximum 1“}) = \frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4} = \frac{1}{4}$$

$$P(\text{„Wahl vom Maximum 2“}) = \frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4} = \frac{1}{4}$$

$$P(\text{„Wahl vom Maximum 3“}) = \frac{1}{3}\cdot\frac{3}{4} = \frac{1}{4}$$

$$P(\text{„Wahl vom Maximum 4“}) = \frac{1}{4}$$

Mit Hilfe der dargestellten Details war die algorithmische Umsetzung von Maximum Linkage in das Rechnerprogramm PicAna möglich.

3.9 Bestimmung der Klassenanzahl beim Maximum-Linkage-Algorithmus

Eine der zentralen Fragen im Rahmen einer Clusterung, ist die Anzahl der zu bildenden Klassen. Bei der bisher vorgestellten Art der Datenclusterung wird die Anzahl der Cluster vom Anwender vor Beginn der Clusterung immer ad hoc vorgegeben. Häufig wird dabei auf bestehende Erkenntnisse bzw. die Erfahrungen eines Analysten aus dem Anwendungsgebiet verwiesen, um die ad-hoc-Festlegung der Anzahl der benötigten Klassen zu rechtfertigen. Bei der hier vorgestellten unbeaufsichtigten Clusterung ist oft nicht klar, wie viele Klassen wirklich notwendig bzw. sinnvoll sind. Daher muss durch mehrmalige Durchführung von Clusterungen untersucht werden, welche Klassenanzahl das beste Ergebnis liefert.
Daher ist es wünschenswert, einen Mechanismus zu finden, der bereits vor der Clusterung die Anzahl der benötigten Klassen vorgibt. Insbesondere bei neuen Problemstellungen, zu denen nur wenig Vorwissen existiert, ist ein solcher Mechanismus sehr hilfreich.
Auf Grund der Überlegungen zur Beurteilung der Qualität einer Clusterung, wie das in Kapitel 5 ausführlich dargestellt wird, lässt sich ein Ansatz zur halbautomatischen Bestimmung der Klassenanzahl entwickeln.
Das wichtigste Kriterium zur Beurteilung der Qualität einer Clusterung ist die Intra-Class-Varianz. Diese basiert auf der Summe der quadratischen Abweichungen innerhalb einer Klasse (SSW = Sum of Squares Within). Sie steht in direktem Zusammenhang mit der Gesamtvarianz, basierend auf der Summe der quadratischen Abweichung aller Werte (SST = Sum of Squares Total). Die Inter-Class-Varianz wird aus der Summe der quadratischen Abweichung zwischen den Klassen (SSB = Sum of Squares Between) berechnet. Hierfür gilt:

$$SST = SSB + SSW .$$

Zum Beweis dieser Behauptung sind zunächst die folgenden Darstellungsmöglichkeiten von SST, SSB und SSW zu betrachten:

$$SST = \sum_{i=1}^{n}\left(P_i - \frac{1}{n}\sum_{j=1}^{m}P_j\right)^2 = \sum_{i=1}^{n}\left[P_i - \overline{P}\right]^2 = \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[P_{ij} - \overline{P}\right]^2$$

$$SSB = \sum_{i=1}^{q} n_i\left[\left(\frac{1}{n_i}\sum_{j=1}^{n_i}P_{ij}\right) - \left(\frac{1}{n}\sum_{i=1}^{n}P_i\right)\right]^2 = \sum_{i=1}^{q} n_i\left[\overline{P}_i - \overline{P}\right]^2 = \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[\overline{P}_i - \overline{P}\right]^2$$

$$SSW = \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[P_{ij} - \left(\frac{1}{n_i}\sum_{j=1}^{n_i}P_{ij}\right)\right]^2 = \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[P_{ij} - \overline{P}_i\right]^2$$

Der Beweis wird durch das Nachrechnen der Behauptung geführt:

$$SST + SSW$$

$$= \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[\overline{P}_i - \overline{P}\right]^2 + \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[P_{ij} - \overline{P}_i\right]^2$$

ergänze obige Gleichung geeignet um Null

$$= \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[\overline{P}_i - \overline{P}\right]^2 + \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[P_{ij} - \overline{P}_i\right]^2 + 2\sum_{i=1}^{q}\left[P_i - \overline{P}\right]\underbrace{\sum_{j=1}^{n_i}\left[P_{ij} - \overline{P}_i\right]}_{=0,\ da\ \sum_{j=1}^{n_i}\left[P_{ij}-\overline{P}_i\right]=\sum_{j=1}^{n_i}P_{ij}-n\overline{P}_i=n\overline{P}_i-n\overline{P}_i=0}$$

$$= \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[\overline{P}_i - \overline{P}\right]^2 + \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[P_{ij} - \overline{P}_i\right]^2 + \sum_{i=1}^{q}\sum_{j=1}^{n_i}2\left[P_i - \overline{P}\right]\left[P_{ij} - \overline{P}_i\right]$$

$$= \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[\overline{P}_i - \overline{P}\right]^2 + \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left[P_{ij} - \overline{P}_i\right]^2 + \sum_{i=1}^{q}\sum_{j=1}^{n_i}2\left[P_i - \overline{P}\right]\left[P_{ij} - \overline{P}_i\right]$$

$$= \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left(\left[\overline{P}_i - \overline{P}\right]^2 + 2\left[P_i - \overline{P}\right]\left[P_{ij} - \overline{P}_i\right] + \left[P_{ij} - \overline{P}_i\right]^2\right)$$

$$= \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left(\left[\overline{P}_i - \overline{P}\right] + \left[P_{ij} - \overline{P}_i\right]\right)^2$$

$$= \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left(\overline{P}_i - \overline{P} + P_{ij} - \overline{P}_i\right)^2$$

$$= \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left(P_{ij} - \overline{P} + \overline{P}_i - \overline{P}_i\right)^2$$

$$= \sum_{i=1}^{q}\sum_{j=1}^{n_i}\left(P_{ij} - \overline{P}\right)^2 = \sum_{i=1}^{n}\left(P_i - \overline{P}\right)^2 = SST$$

ð

Sei r die Anzahl der zu clusternden Datenpunkte und q die Klassenanzahl. Dann fällt die SSW monoton bei steigendem q. Bei $q := 1$ wird die Gesamtvarianz allein durch die SSW erklärt und die SSB ist null. Bei $q := r$ wird die Gesamtvarianz nur durch die SSB erklärt und die SSW ist null.
Das Ziel der Clusterung ist es, eine kleine SSW und gleichzeitig eine kleine Anzahl an Klassen q zu erreichen. Beide Forderungen sind aber nicht gleichzeitig zu erreichen, da sie konträr zueinander stehen. Es muss ein Kompromiss zwischen möglichst kleiner SSW und kleinem q gefunden werden.
In der Praxis wird so vorgegangen, dass eine Vielzahl von Clusterungen mit unterschiedlichen Werten für q durchgeführt werden. Zu allen Clusterungen wird die SSW berechnet. Damit liegen die notwendigen Daten vor, so dass auf Grund des Vergleichs von SST mit SSW eine Entscheidung getroffen werden kann. Außerdem liegen die Clusterungsergebnisse selbst vor und können mit berücksichtigt werden. Die Durchführung einer großen Anzahl verschiedener Clusterungen stellt aber im Wesentlichen ein Zeitproblem dar. Insbesondere bei der Anwendung neuronaler Netze, die in Kapitel 4 ausführlich vorgestellt werden, ist dieses Zeitproblem eklatant. Das liegt daran, dass die Laufzeit bei Clusterungen mit neuronalen Netzen deutlich über der des Maximum Linkage liegt. Bei der Implementierung des Maximum-Linkage-Algorithmus bietet sich eine Möglichkeit, dieses Problem durch eine Zwischenberechnung zu lösen.
Dabei wird der in Abschnitt 3.5 dargestellte Vorteil ausgenutzt, dass die Zentroide einer Clusterung mit weniger Klassen, in denen mit mehr Klassen enthalten sind. Dazu wird innerhalb der Berechnung der Zentroide ein zusätzlicher Zwischenschritt eingefügt. Jedesmal wenn ein weiterer Zentroid ausgewählt wurde, wird in diesem Zwischenschritt eine Klassifikation der übrigen Werte durchgeführt. Damit stehen die notwendigen Daten zur Verfügung, um die SSW zu berechnen. Diese wird zusammen mit der zugehörigen Anzahl der aktuellen Zentroide abgespeichert. So entsteht bei akzeptabler Laufzeitverlängerung eine Tabelle, die es ermöglicht, die am besten geeignete Clusterung zu bestimmen. Es bleibt dabei dem Anwender überlassen, die Klassenzahl festzulegen, je nachdem, welchen Wert für SSW er zu akzeptieren bereit ist. Zu Beginn einer Clusterung muss statt der Klassenzahl eine Anzahl festgelegt werden, bis zu welchem q die Tabelle der SSW berechnet werden soll. Für die programmtechnische Umsetzung ist es wichtig, die Reihenfolge, in der die Zentroide in das Zentroidcluster aufgenommen werden, festzuhalten. Nachdem der Anwender das q auf Grund der Tabelle der Intra-Class-Varianzen ausgewählt hat, werden alle Zentroide gelöscht, die nach dem q-ten Schritt aufgenommen wurden. Mittels der verbleibenden Zentroide wird die endgültige Klassifizierung der übrigen Werte durchgeführt und die Ergebnistabellen und -kenngrößen erzeugt.

4 Künstliche neuronale Netze

Die Beschäftigung mit künstlichen neuronalen Netzen verlangt zunächst zu verstehen, was biologische neuronale Netze sind. Erst mit dem Verständnis des allgemeinen Aufbaus biologischer neuronaler Netze können die daran angelehnten künstlichen neuronalen Netze behandelt werden.
Die intelligente Leistung des menschlichen Gehirns befindet sich in der Hirnrinde (Neokortex). Diese aus Nervenzellen bestehende Hirnrinde lässt sich in verschiedene Felder einteilen. So gibt es spezialisierte Gebiete, für die visuelle Wahrnehmung (visueller Koretx) oder Tastwahrnehmung (somatosensorischer Kortex). Unter jedem Quadratmillimeter des Neokortex befinden sich beim Menschen ca. 100.000 eng vernetzte Nervenzellen, sogenannte *Neuronen*. In diesen Neuronen findet die „Datenverarbeitung“ im Gehirn statt. An einem typischen Neuron kann man drei Strukturen herausarbeiten. Diese sind der Dendritenbaum, der Zellkörper und das Axon. Im übertragenen Sinne sind diese Strukturen für die Eingabe, Verarbeitung und Ausgabe von Informationen verantwortlich.

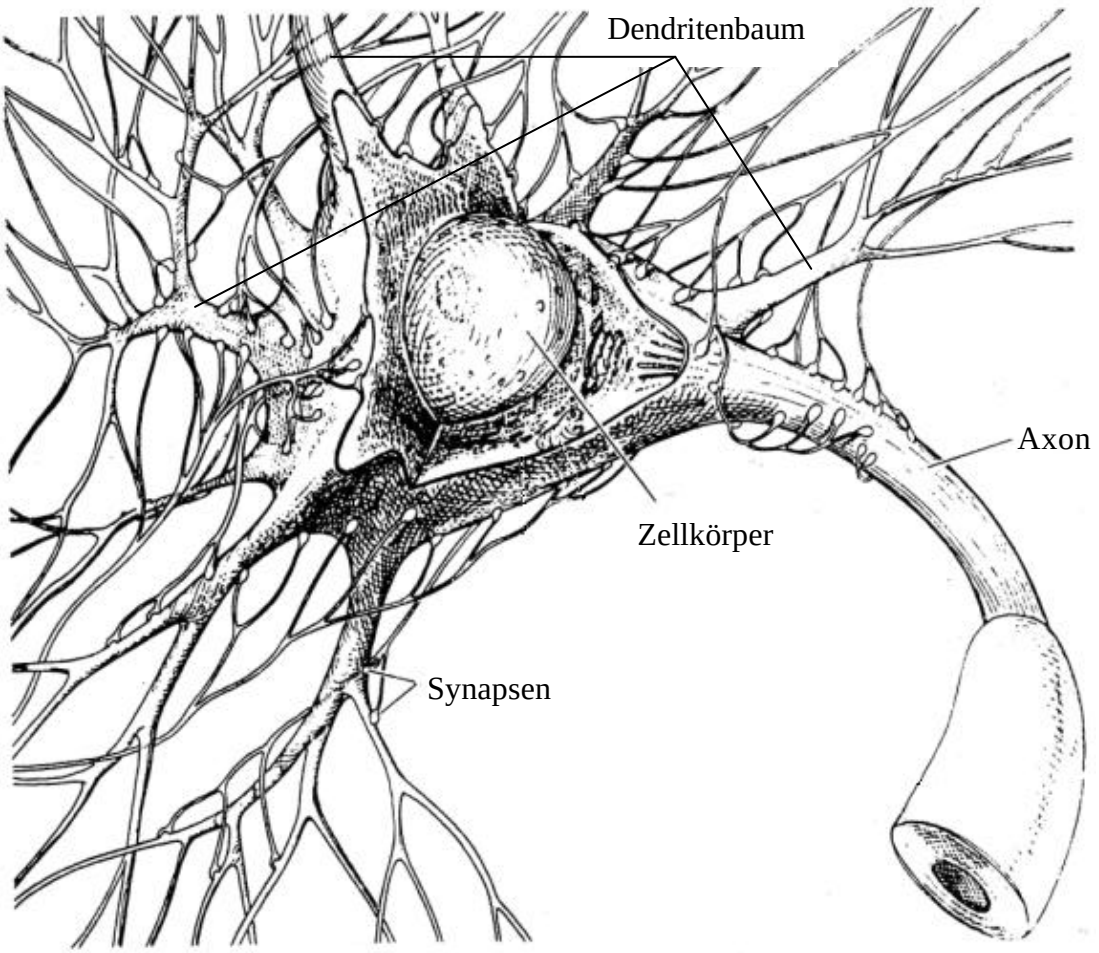


Abb. 4.1: Schematische Darstellung eines typischen Neurons aus Rojas (1993)

Der Dendritenbaum als Haupteingabeweg dehnt sich um ein Neuron herum aus und summiert die Ausgabesignale der umgebenden Neuronen in Form eines elektrischen Potentials. Überschreitet dieses Potential einen bestimmten Schwellenwert, so erzeugt der Zellkörper einen elektrischen Impuls, der vom Axon weitergeleitet wird. Dem Axon ist es vorbehalten, den Impuls an die Zielneuronen weiterzuleiten. Die Kontaktstellen des Axons mit dem Zellkörper des Zielneurons heißen Synapsen. Der elektrische Impuls des Axons bewirkt an der Synapse die Ausschüttung eines Überträgerstoffes, der wiederum zu einer Potentialänderung am Zellkörper des Zielneurons führt. Bedingt durch Art und Zustand der Synapse, kann ein ankommender Impuls mehr oder weniger starke Potentialerhöhungen oder –erniedrigungen auslösen.

Diese biologischen Vorgaben galt es in geeignete mathematische Modelle umzusetzen. Zwei Mathematiker, McCulloch und Pitts, entwickelten 1943 ein Modell, welches ein Neuron als ein logisches Schwellenwertelement mit zwei Zuständen beschreibt. Ein solches Schwellenwertelement weißt m Eingangsleitungen und eine Ausgangsleitung auf. Eine Eingangsleitung kann entweder aktiv (Zustand = 1) oder passiv (Zustand = 0) sein. Der Zustand des Schwellenwertelements ergibt sich als Summe der Eingangssignale P_i und Vergleich mit dem Schwellenwert q. Überschreitet die Summe den Schwellenwert, so ist das Neuron „erregt“, andernfalls nicht. Erregende Eingangssignale werden durch die „Synapsenstärke“ $W_i = +1$ beschrieben, hemmende mit $W_i = -1$. Das Ausgabesignal *net* eines Neurons ist somit gegeben durch

$$net = f\left(\sum_{i=1}^{m} W_i P_i - q \right), \qquad (4.1)$$

wobei gilt $f(x) = 1$ für $x \geq 0$ und $f(x) = 0$ für $x < 0$. Die Funktion aus (4.1) wird dabei als Aktivierungsfunktion bezeichnet. Mit diesem Prinzip lässt sich jede beliebige logische Funktion durch eine geeignete Kombination derartiger Elemente aufbauen (Ritter et al, 1991). Die Idee von McCulloch und Pitts, ein Neuron als Schwellenwertelement zu betrachten, ist bis heute Grundlage bei vielen entwickelten Modellen.

Leider weist diese Theorie zwei Schwachstellen auf. Zum einen ist nicht erkennbar, wie die Neuronen verschaltet werden können. Insbesondere wie dieses durch *„Lernen“* geschehen soll. Zum anderen müssen bei diesem Modell alle einzelnen Komponenten für einen reibungslosen Ablauf funktionieren. Die Beantwortung des ersten Problems durch den Psychologen Hebb (1949) ermöglichte es, Neuronen als lernfähige Verschaltung zu begreifen. Nach seinen Überlegungen ändert sich die durch die Synapse bewirkte Verschaltung zweier Neuronen proportional zur Aktivität vor und hinter der Synapse.

Mit der Entwicklung der elektronischen Rechner ist man erstmals in die Lage versetzt worden, die Lernfähigkeit von neuronalen Netzwerken unter Berücksichtigung

der obigen Überlegungen in Simulationen zu erproben. Um solche Simulationen vornehmen zu können, bedurfte es einer weiteren Neuerung bei den Ansätzen zu neuronalen Netzen. Diese Neuerung ist das auf Rosenblatt (1958) zurückgehende Perzeptron. Ein Perzeptron besteht aus einer festen Anzahl an Elementen N, die als Neuronen bezeichnet werden. Diesen Neuronen werden über m Leitungen so genannte Eingabemuster angelegt. Ein Eingabemuster ist somit ein m-dimensionaler Vektor und repräsentiert eine mögliche Eingabe aus der Menge aller möglichen Eingaben für das Netz. Ein weiterer Unterschied zum McCulloch/Pitts Modell ist, dass nur noch gewichtete Netze aufgebaut werden. In älteren Modellen kommen dagegen keine Gewichte vor. Die Gewichtung wird über Gewichte an den Verbindungen (Kanten) zwischen den Neuronen realisiert. Ein neuronales Netz, was auf dem Modell von Rosenblatt beruht, verändert mit Hilfe eines Algorithmus die Kantengewichte zwischen den Neuronen. Diese Gewichtsveränderung wird als Lernen bezeichnet. Im Laufe des Lernens (Lernphase) passt jedes Element seine Gewichte dergestalt an, dass es nur noch auf Eingabemuster reagiert, die aus „seiner" Klasse kommen. In der Theorie wird das Anlernen eines Netzes auch als Training bezeichnet. Wie Ritter et al (1991) beschreibt, entspricht jedes angelegte Muster einem Punkt P in einem mitunter hochdimensionalen Merkmalsraum. Die einzelnen Klassen lassen sich nun als Punktemenge in diesem Raum ansehen. Jedes Element muss seine Ausgabewerte so vergeben, dass der Raumbereich, der mit dem Ausgangswert 1 korrespondiert, nur Punkte der dem Element zugeordneten Klasse enthält (Rojas, 1993). Über die Jahre ist dieses Modell weiter verfeinert worden. Eine erste Verfeinerung zielte etwa darauf ab, das Modell auf das Notwendige zu reduzieren. Weitere Verbesserungen sind die Parallelisierung der Signalverarbeitung, wie sie beispielsweise bei Minsky (1985) nachzulesen sind.

Lernen in neuronalen Netzen

Das Herausarbeiten des korrekten neuronalen Netzes entspricht einer sukzessiven Gewichtsveränderung. In diesem Zusammenhang spricht man vom Lernen des Netzes. Fasst man ein neuronales Netz als Menge von Funktionen auf, so wird in der Lernphase eine vorgegebene Lernfunktion wieder und wieder aufgerufen. Das Netz organisiert sich selbst. Diese Organisation wird durch das Anlegen von Ein- und Ausgabebeispielen sowie einem Korrekturschritt erreicht. Im Korrekturschritt werden die Gewichte an die Eingabe angepasst. Ein guter Lernalgorithmus erkennt aus der „Erfahrung" notwendige Korrekturen, um die Gewichte so lange Schritt für Schritt zu verändern, bis eine Lösung des Problems gefunden ist.

Man unterscheidet zwei Arten des Lernens. Dieses sind überwachtes bzw. unüberwachtes Lernen. Dabei entspricht überwachtes Lernen dem folgenden Prinzip: Für ein anstehendes Problem sei eine Menge von Eingaben für das Netz vorgegeben. Diese Eingaben werden an das neuronale Netz angelegt, wobei zu beachten ist, dass zu Beginn der Lernphase die Gewichte zufällig initialisiert sind. Die Ausgabe des Netzes wird bzgl. jeder Eingabe beobachtet. Entspricht diese nicht der erwarteten

Ausgabe, so müssen die Gewichte verändert werden. Die Gewichtsveränderung geschieht entsprechend einer vorgegebenen Lernfunktion. Festzuhalten ist: Zu jeder Eingabe ist bereits zu Beginn die Ausgabe bekannt.
Ist zu einer vorgegebenen Eingabe deren Ausgabe nicht bekannt, spricht man von unüberwachtem Lernen. In diesem Fall muss die Lernfunktion die „korrekte“ Ausgabe selbst finden. Zu diesem Zweck wird in die Lernfunktion ein Korrekturterm eingebaut. Dieser Term „bewertet“ die momentane Anpassung an die gestellte Aufgabe. Ist die Abweichung unterhalb einer vorgegebenen Schranke, so ist das Lernen abgeschlossen. Man spricht auch davon, dass das Netz konvergiert ist. In Abwesenheit weiterer Informationen, erfordert diese Art von Lernen nach Ritter et al (1991) einen stochastischen Suchprozess im Raum aller möglichen Werte, dessen Ziel in der Maximierung der bei jedem Schritt erhaltenen Bewertung besteht.

Aufbau von neuronalen Netzen

Erst mit Einführung des Perzeptrons ist es möglich, mehrere Neuronen miteinander zu verknüpfen. Allerdings befinden sich alle Neuronen noch in einer Ebene, auch Schicht genannt. Mit einschichtigen neuronalen Netzen lassen sich jedoch nicht alle Berechnungsprobleme lösen. Um auch komplizierte Probleme mittels eines neuronalen Netzes zu berechnen, bedarf es der Erweiterung auf mehrere Schichten. Wobei ein eben solches Netz stets aus einer Eingabe-, einer Ausgabe- und ggf. ein oder mehreren verdeckten Schichten besteht.

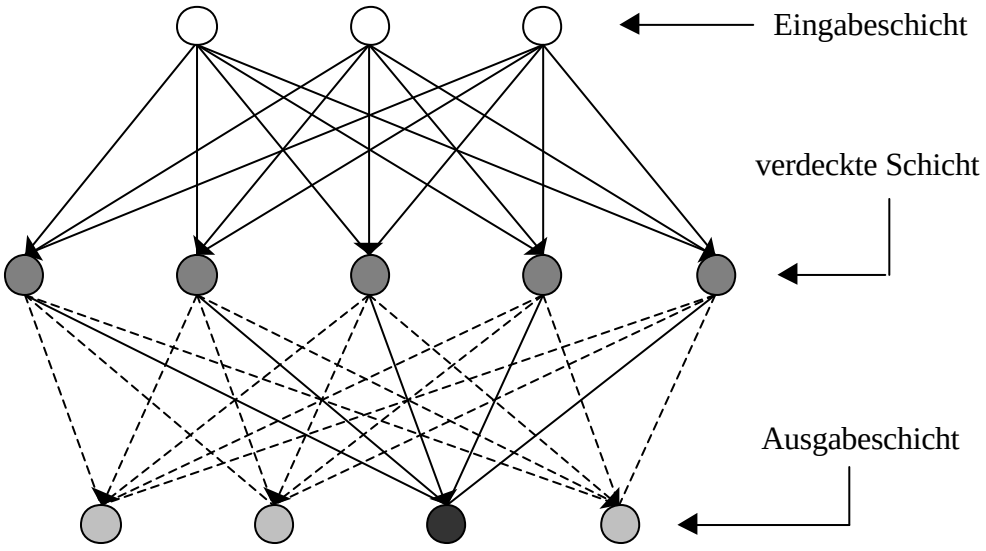


Abb. 4.2: Klassifikationsnetz mit einer verdeckten Schicht für 4 Klassen bei 3 Merkmalen

Im Laufe der Jahre, insbesondere in den 80er Jahren, wurde eine Vielzahl unterschiedlicher neuronaler Netze entwickelt. Insbesondere sind dabei zu erwähnen: Das

Hopfield-Netz (Hopfield, 1982), das Winner-takes-all-Netz und selbstorganisierende Karten (SOMs, Kohonen, 1982). Diese Netzformen eignen sich zur Lösung von Clusterungsproblemen (Bock, 1998). Insbesondere die von Kohonen (1982) vorgestellten selbstorganisierenden Karten bieten sich dafür an. Solche Clusterungsprobleme stehen in enger Verbindung mit klassischen Verfahren wie Projection Persuit (Friedman, 1987) oder Hauptkomponentenanalyse (Fahrmeir, 1996). Die Anwendung neuronaler Netze in diesem Bereich ist vielversprechend, weil damit in ähnlichen Bereichen beachtliche Erfolge erzielt werden konnten (Guimarães und Urfer, 2000). Die erwähnten Hopfield-Netze unterscheiden sich gegenüber früheren Modellen, z. B. dem Perzeptron, dadurch, dass das Eingabemuster jedes Neurons von den Zuständen aller übrigen Neuronen gebildet wird (Ritter et al, 1991). Dabei wird in der Aktivierungsfunktion (4.1) die Signumfunktion verwandt. Die Veränderungen werden „asynchron“ vorgenommen. Somit aktualisiert jedes Neuron seinen Zustand zu diskreten Zeitpunkten, deren Wahl zwischen den Neuronen unkorreliert erfolgt.
Auf die anderen erwähnten Netztypen, Winner-takes-all-Netz und selbstorganisierende Karten, wird im Weiteren detaillierter eingegangen, da diese Netze zur Behandlung des Ausgangsproblems eingesetzt werden.

4.1 Spezielle Netztypen

4.1.1 Das Winner-takes-all-Netz

Ein Winner-takes-all-Netz (WTAN) gehört zu den einfachsten, unüberwacht lernenden Netzen. Es besteht aus einer Eingabe- und einer Ausgabeschicht von Neuronen. Es existieren keine verdeckten Schichten. Die Schichten weisen keine Strukturen zwischen den Neuronen auf. Damit ein Datenwert mit einem WTAN klassifiziert werden kann, sind Anforderungen an die Anzahl der Neuronen in den einzelnen Schichten zu stellen. Die Anzahl der Neuronen in der Eingabeschicht entspricht der Anzahl m der Merkmale, der zu verarbeitenden Daten. Das ist notwendig, damit die zu verarbeitenden Daten überhaupt als Eingaben für das Netz geschaltet werden können. Die Aktivierungsfunktion der Neuronen der Eingabeschicht ist hier die Identität. Die Ausgabeschicht besteht aus einer q-elementigen Schicht von Neuronen, von denen jedes über einen m-dimensionalen Gewichtsvektor mit der Eingabeschicht verbunden ist. Die Ausgabeschicht besitzt genau q Neuronen, weil jedes Neuron genau eine der Klassen repräsentieren soll, welcher der Eingabewert anschließend zugeordnet wird. Schaltet man den zu klassifizierenden Wert als Netzeingabe, wird auf Grund der Netzstruktur genau ein Neuron der Ausgabeschicht aktiviert. Die Details der Aktivierung werden im weiteren Verlauf dieses Abschnitts genauer erläutert. Das aktivierte Neuron wird als Gewinner-Neuron bezeichnet. Der Eingabewert wird dann der Klasse zugeordnet, deren repräsentierendes Neuron aktiviert wurde.
Ein Winner-takes-all-Netz entspricht nicht explizit einem sogenannten Feed-Forward-Netz. Bei einem Feed-Forward-Netz sind die Neuronenverbindungen nur in

einer Richtung ausgeprägt. Die Richtung verläuft stets von den Eingabeneuronen zu den Ausgabeneuronen. Neuronen innerhalb einer Schicht sind nicht miteinander verbunden. Da bei einem Winner-takes-all-Netz in der Ausgabeschicht genau ein Neuron aktiviert werden soll, kann ein Feed-Forward-Netz nicht verwendet werden. Es bedarf in der Ausgabeschicht einer Rückkoppelung zwischen den Neuronen. Jedes Neuron enthält dann hemmende (inhibitorische) Verbindungen zu den anderen Neuronen seiner Schicht, sodass sich das Neuron mit der stärksten Aktivierung durchsetzt. Diese Eigenschaft wird als laterale Rückkoppelung bezeichnet (Zell, 2000). Die folgende Abbildung zeigt eine schematische Darstellung eines Winner-takes-all-Netzes bei aktiviertem Ausgabeneuron.

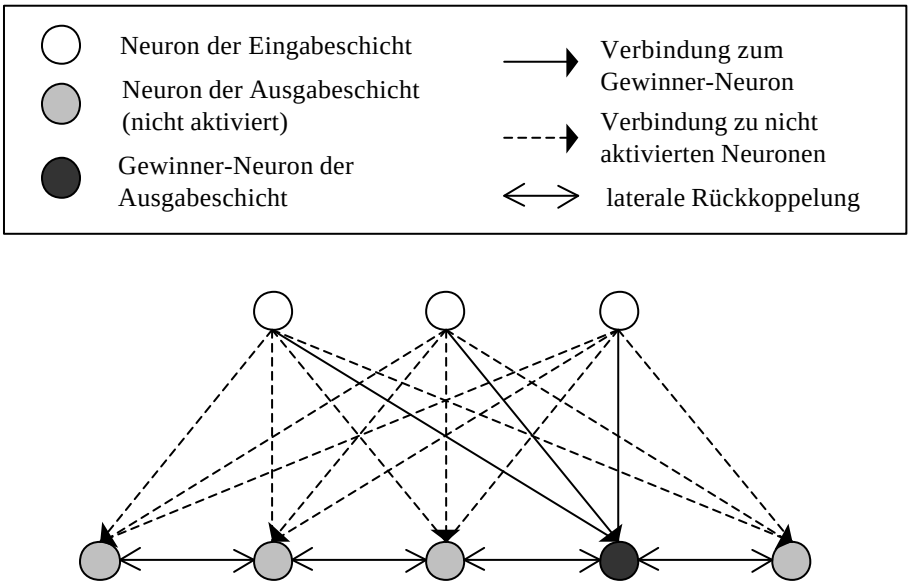


Abb. 4.3: Winner-takes-all-Netz mit 3 Eingabeneuronen und 5 Ausgabeneuronen

Die Klassifikation der Werte P erfolgt durch die spezielle Definition der Aktivierungsfunktion der Neuronen in der Ausgabeschicht. Sei N_j das Neuron dessen Aktivierung gesucht ist und seien W_i, $i=1,...,q$ die Gewichte der Ausgabeneuronen N_i, $i=1,...,q$. Die Aktivierungsfunktion der Ausgabeneuronen lässt sich wie folgt darstellen:

$$f_{WTAN}(N_j) = \begin{cases} 1 & , \quad \| P - W_j \|_2 = \min\limits_{i=1,\ldots,q} \left\{ \| P - W_j \|_2 \right\} \\ 0 & , \quad \| P - W_j \|_2 \neq \min\limits_{i=1,\ldots,q} \left\{ \| P - W_j \|_2 \right\} \end{cases}. \qquad (4.2)$$

Die Aktivierungsfunktion ist so geartet, dass bei Anlegen eines Eingabemusters, wie gefordert, genau ein Neuron aktiviert wird. Nach (4.2) ist dasjenige Ausgabeneuron

N_j aktiv, dessen Gewichtsvektor W_j den kleinsten Abstand (im Sinne einer vorgegebenen Norm) zum Eingabemuster aufweist. Es gibt also genau ein Gewinner-Neuron und alle anderen Neuronen sind nicht aktiviert. Daher hat diese Netzart ihren Namen „Winner-takes-all-Netze“. Das setzt voraus, dass die Gewichtsvektoren aller Neuronen verschieden sind. Diese Eigenschaft des Netzes muss bereits in der Lernphase ausgebildet werden.

4.1.2 Selbstorganisierende Karten (SOMs)

Die selbstorganisierenden Karten gehen auf Kohonen (1982) zurück. Eine klassische selbstorganisierende Karte (engl.: Self Organizing Map, kurz: SOM) ist ein unüberwacht lernendes, in der Regel zweischichtiges, Winner-takes-all-Netz. Es besitzt eine Eingabeschicht und eine im allgemeinen Fall k-dimensionale Ausgabeschicht. Die Neuronen der Ausgabeschicht sind dabei durch eine Nachbarschaftsstruktur verbunden. Die Nachbarschaftsstruktur wird durch die Lage der Neuronen innerhalb der Ausgabeschicht vorgegeben. Das macht den wesentlichen Unterschied zum einfachen WTAN aus, da dort keine Nachbarschaftsstruktur vorliegt. Durch das Anlernen eines solchen Netzes wird eine so genannte Kartierung des Merkmalsraumes durchgeführt. Nach dem Training repräsentiert jedes Ausgabeneuron einen zusammengehörenden Bereich im Merkmalsraum. Nebeneinander liegende Neuronen auf der Karte sollen nebeneinander liegende Bereiche im Merkmalsraum repräsentieren (Matecki, 1999).

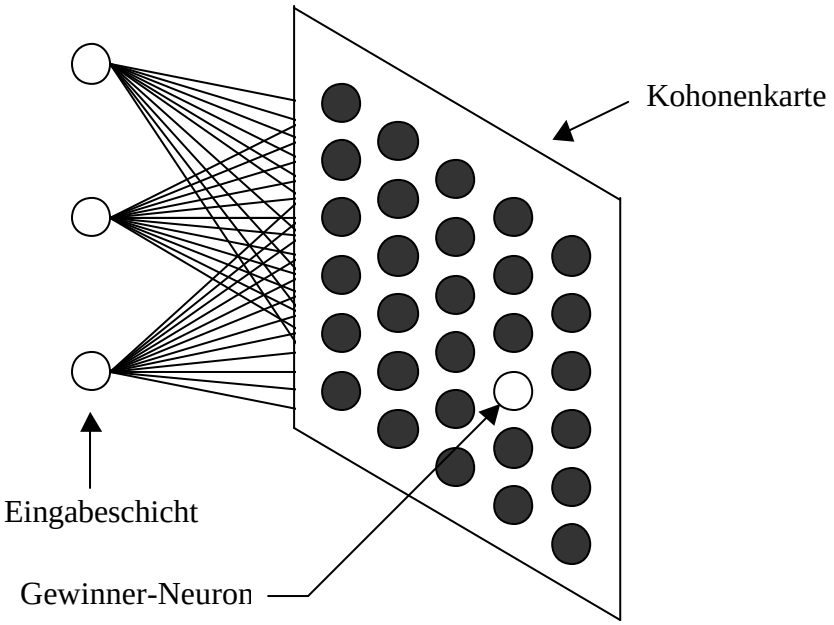


Abb. 4.4: Schematische Darstellung eines zweidimensionalen 5 x 6 SOM mit 5 Eingabeneuronen. Der Übersichtlichkeit halber ist ein Teil der Verbindungen zwischen Eingabe- und Ausgabeschicht nicht eingezeichnet.

Abb. 4.4 skizziert ein zweidimensionales 5 x 6 SOM mit 5 Eingabeneuronen. Der Übersichtlichkeit halber ist dabei ein Teil der Verbindungen zwischen Eingabe- und Ausgabeschicht nicht eingezeichnet. Die Dimension der Ausgabeschicht richtet sich nach der Vorgabe des Anwenders und kann von Fall zu Fall variieren. Auf Grund der Übersichtlichkeit und der späteren Anwendung wird hier nur auf zweidimensionale Strukturen eingegangen.
Die Datenklassifizierung beim SOM erfolgt analog zum WTAN. Die Aktivierungsfunktion ist somit ebenfalls analog zu wählen. Sie entspricht (4.2). Der wesentliche Unterschied zum WTAN liegt in der Neuronenanordnung der Ausgabeschicht. Erst die Visualisierung der herausgearbeiteten Nachbarschaftsstruktur der SOMs verdeutlichen diesen Unterschied. In dieser Visualisierung werden die berechneten Cluster durch unterschiedliche Grauwerte innerhalb einer Karte dargestellt. Die als U-Matrix bekannte Methode (Ultsch, 1989) stellt dabei geringe Abstände zwischen den Neuronen mit einem hellen Grauwerten und große Abstände mit dunklen Grauwerten dar. Die Visualisierung der U-Matrix wird im Folgenden Kohonenkarte genannt.
Die Anordnung der Ausgabeneuronen, also die Struktur der Kohonenkarte, ist nicht auf Grund der Daten zu ermitteln, sondern muss bei der Konstruktion des Netzes festgelegt werden. Sie wird als Nachbarschaftsstruktur der Neuronen bezeichnet. Die Lage der Neuronen innerhalb dieser Struktur hat einen entscheidenden Einfluss auf das Training des Netzes. Es gibt vielfältige Grundstrukturen. Am gebräuchlichsten sind ein-, zwei- oder dreidimensionale Strukturen (Ritter et al, 1991). Bei den späteren Anwendungen werden nur zweidimensionale Strukturen eingesetzt, sodass auf andere hier nicht weiter eingegangen wird. Die Anordnung der Neuronen erfolgt unabhängig von der Dimension in einer regelmäßigen Gitterstruktur. Für die Art des Gitters sind regelmäßige Vielecke von quadratisch bis oktagonal üblich. Da sich die späteren Theorien auf quadratische und hexagonale Gitter beschränken, wird im Folgenden nur auf diese Strukturen eingegangen.

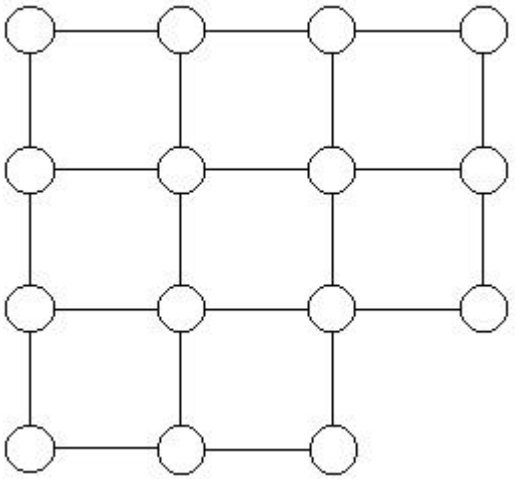


Abb. 4.5: Neuronengitter von 15 Neuronen mit zweidimensionalem, regelmäßigem, quadratischem Gitter als Nachbarschaftsstruktur.

Das in Abb. 4.5 dargestellte Neuronengitter zeigt $q = 15$ Neuronen. Diese sind in einem zweidimensionalen, regelmäßigen, quadratischen Gitter angeordnet. Die Gitterverbindungen repräsentieren die Nachbarschaftsstruktur der Neuronen untereinander. Die Gitterstruktur wird zum Anlernen von SOMs benötigt. Daher sind die Details dieses Gitteraufbaus entscheidend. Es soll ein Gitter mit q Neuronen konstruiert werden. Bei einem regelmäßigen (äquidistanten und orthogonalen), quadratischen Gitter soll zudem die Anzahl der Zeilen und Spalten möglichst gleich sein.
Die Gleichheit wird nicht notwendig verlangt, bietet aber die beste Alternative. Gerade im Hinblick auf das unüberwachte Lernverfahren, welches ohne Vorinformationen auszukommen hat, ist diese Vorgehensweise angebracht. Die Anzahl der Zeilen und Spalten ist aus der Anzahl q der geforderten Neuronen zu berechnen. Dabei entspricht die Anzahl der Zeilen und Spalten genau der Anzahl der Neuronen pro Spalte und Zeile. Die Anzahl der Neuronen pro Zeile und Spalte q_{ROW} berechnet sich bei vorgegebener Anzahl q an Neuronen gemäß:

$$q_{ROW} = \left\lceil \sqrt{q} \right\rceil .$$

Zum Anlernen des Netzes muss zunächst ein Abstandsmaß zwischen den Neuronen definiert werden. Auf Grund der Gitterstruktur und den Kanten zwischen den Neuronen, bietet sich ein Kantenverfolgungs-Algorithmus an. Bei einem Kantenverfolgungs-Algortihmus wird ein Weg zwischen zwei Neuronen definiert (Reade, 1991). Ein Weg definiert sich über die Kanten zwischen den Neuronen. Er muss nicht eindeutig sein. Der Abstand zwischen zwei Neuronen definiert sich über den kürzesten möglichen Weg zur Verbindung dieser Neuronen. Die Länge eines Wegs definiert sich über die Anzahl an Kanten, die er enthält. Auch der kürzeste Weg muss nicht eindeutig sein. Es stellt sich die Frage, ob der Fall von mehreren kürzesten Weg gesondert behandelt werden muss. Es ist sehr aufwendig, den Abstand zwischen den Neuronen über Kantenverfolgungs-Algortihmen zu behandeln. Deshalb wird eine einfachere Methode zur Abstandsbestimmung angewandt. Das Problem der mehrfachen Erreichbarkeit über den kürzesten Weg tritt dabei nicht auf. Bei dieser Methode wird die Länge der Kanten auf eins festgesetzt. Auf Grund dieser Festlegung ist es möglich, die Neuronen mit Koordinaten zu versehen, die zur Abstandsbestimmung nutzbar sind. Dazu werden die Neuronen zeilenweise von eins bis q durchnummeriert. Das linke, obere Neuron bekommt die Koordinaten (x,y) = (1,1) zugewiesen. Die Koordinaten aller anderen Neuronen ergeben sich kanonisch, anhand der Lage des Neurons im Gitter. Dazu sind Zeilen und Spalten des Neuronengitters von eins bis q_{ROW} nummeriert. Formal ergeben sich die Koordinaten x_{Koo} , y_{Koo} des i-ten Neurons N_i durch:

$$\left.\begin{aligned} x_{Koo}(N_i) &= i \bmod q_{ROW} \\ y_{Koo}(N_i) &= \lfloor\, i / q_{ROW} \,\rfloor \end{aligned}\right\} \; i = 1,\ldots,q\,. \qquad (4.3)$$

Auf Grund dieser Koordinaten lässt sich eine Abstandsbeziehung definieren, die auf dem euklidischen Abstand zwischen den Koordinaten der Neuronen beruht. Die folgende Abbildung soll diese Abstandsberechnung verdeutlichen:

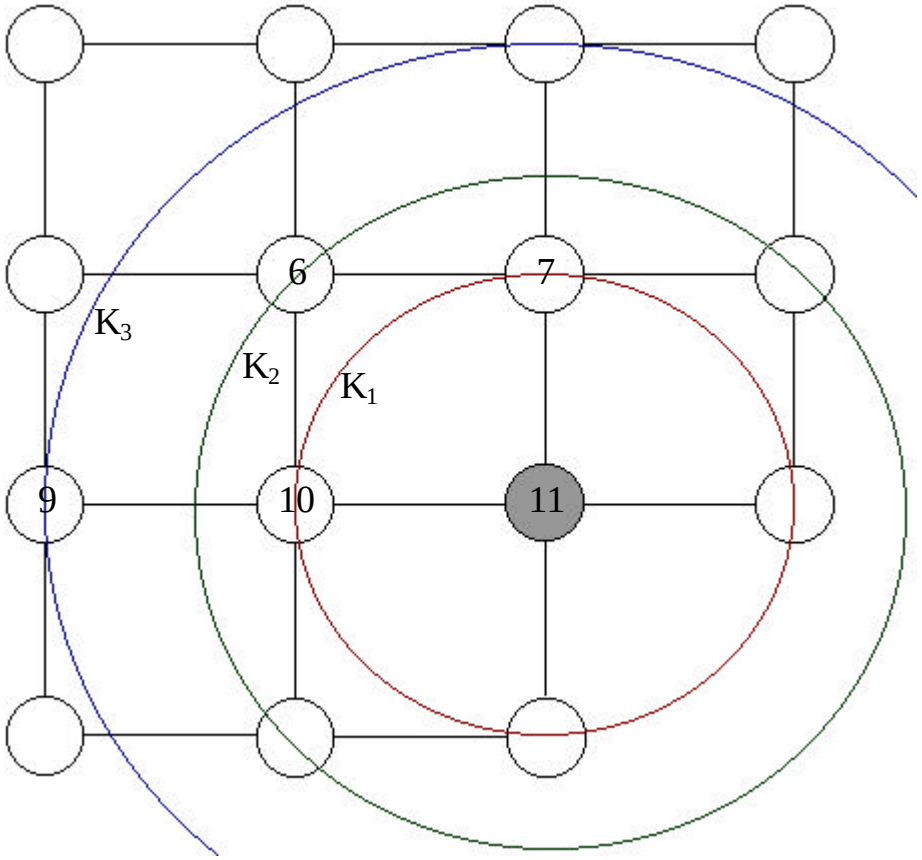


Abb. 4.6: Abstandsbestimmung auf dem Neuronengitter

Das grau markierte Neuron 11 in Abb. 4.6 stellt den Referenzpunkt dar, von dem aus die Abstandsberechnung zu den übrigen Neuronen durchgeführt werden soll. Die Abstände zwischen Neuron 11 und den übrigen Neuronen werden durch die Radien der Kreise K_1, K_2 und K_3 angegeben. Die Radien entsprechen gerade dem euklidischen Abstand zwischen dem Neuron 11 und den durch den Kreis geschnittenen Neuronen. Der Abstand zwischen den „Abstandskreisen“ ist nicht gleichmäßig. Dadurch ergibt sich aber kein negativer Effekt auf das Anlernen des Netzes. Es wird

sogar implizit die unterschiedliche Erreichbarkeit der Neuronen auf der Gitterebene berücksichtigt. So sind die Neuronen 9 und 10 nur auf einem Weg von 11 aus zu erreichen. Das Neuron 6 ist auf zwei gleichlangen Wegen sowohl über das Neuron 10 als auch über das Neuron 7 zu erreichen. Daher ist der Radius von K_2 näher bei dem von K_1 und weiter von dem von K_3 entfernt.

Der Abstand zwischen den Neuronen N_i und N_j ist unter Verwendung von (4.3) gegeben durch:

$$\left\| \begin{pmatrix} x_{Koo}(N_i) \\ y_{Koo}(N_i) \end{pmatrix} - \begin{pmatrix} x_{Koo}(N_j) \\ y_{Koo}(N_j) \end{pmatrix} \right\|_2 = \sqrt{(x_{Koo}(N_i) - x_{Koo}(N_j))^2 + (y_{Koo}(N_i) - y_{Koo}(N_j))^2} \,. \qquad (4.4)$$

Als Nächstes folgen die hexagonalen Gitter. Viele der oben gewonnenen Erkenntnisse sind auf diesen Bereich übertragbar. Das liegt daran, dass die hexagonale Struktur aus der quadratischen abgeleitet werden kann. Abb. 4.7 zeigt in regelmäßiger Hexagonstruktur angeordnete Neuronen. Zur Veranschaulichung des Konstruktionsprinzips liegen die Neuronen im Mittelpunkt regelmäßiger Hexagone.

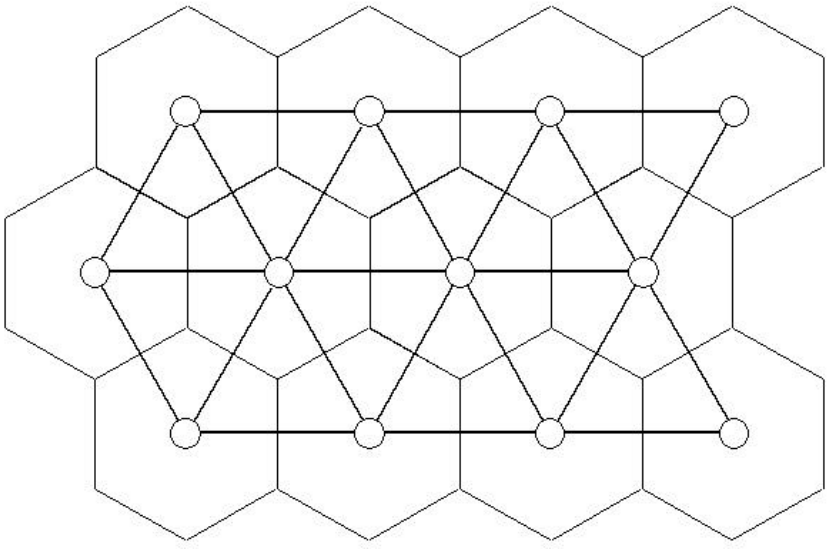


Abb. 4.7: Darstellung des Aufbaus eines zweidimensionalen, regelmäßigen hexagonalen Neuronengitters

Analog zu der quadratischen Variante sollen alle Kanten des Neuronengitters o.B.d.A. die Länge eins aufweisen. Daraus ergeben sich zum einen die Abmessungen der Hexagone. Zum anderen bedingt dies auch die Abstände und Verschiebungen zwischen den Zeilen des Neuronengitters.

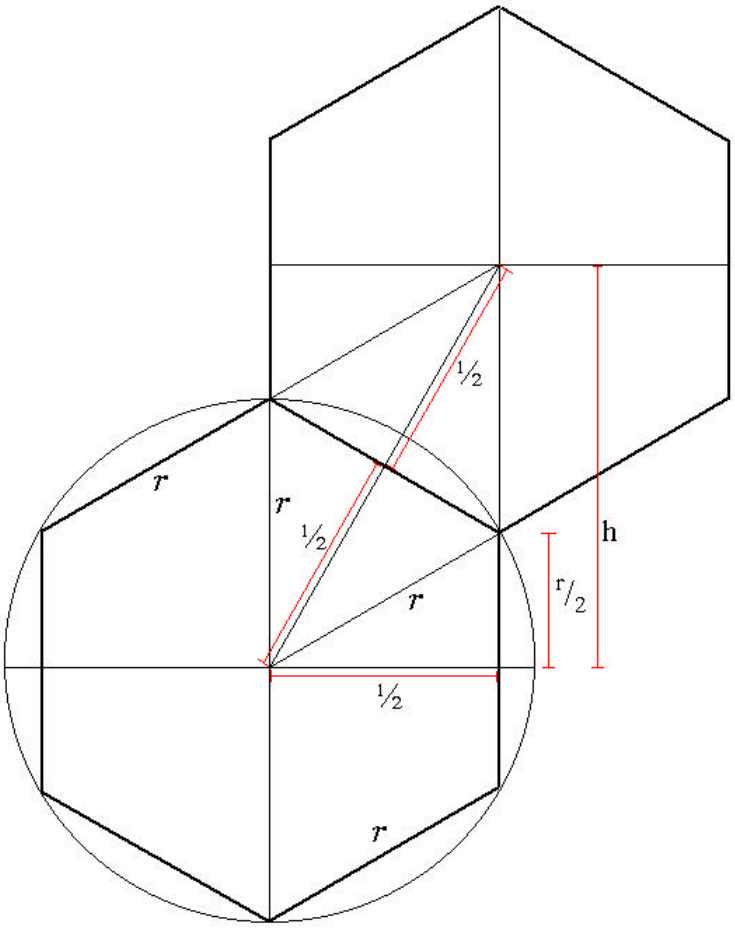


Abb. 4.8: Schema zweier Hexagone als Grundlage für die hexagonale Neuronengitterstruktur

Wie Abb. 4.8 verdeutlicht, gelten für die Hexagone, in deren Mittelpunkt die Neuronen liegen, und das Neuronengitter eine Reihe von Eigenschaften.

- Die Eckpunkte der Hexagone liegen alle auf einem Kreis mit Radius r.
- Die Seiten der Hexagone haben alle die Länge r.
- Die Innenwinkel der Hexagone weisen in jedem Eckpunkt 60° auf.
- Die Seitenhalbierenden der Hexagone müssen die Länge ½ aufweisen, damit der Abstand zwischen den Neuronen bei genau eins liegt.
- Zwei übereinander liegende Zeilen von Neuronen sind um die Länge ½ in der Horizontalen gegeneinander verschoben.
- Der vertikale Abstand h zwischen zwei übereinander liegenden Zeilen berechnet sich durch:

$$h^2 + \left(\frac{1}{2}\right)^2 = \left(\frac{1}{2} + \frac{1}{2}\right)^2 \quad \Leftrightarrow \quad h = \sqrt{\frac{3}{4}} \quad . \qquad (4.5)$$

Auf Grund dieser Erkenntnis ist die hexagonale Struktur leicht aus der quadratischen herzuleiten. Die Überführung der quadratischen in eine hexagonale Karte erfolgt in mehreren Schritten:

1. Jede zweite Zeile des quadratischen Neuronengitters mit Kantenlänge ist horizontal um ½ versetzt (siehe Abb. 4.9 und Abb. 4.10). Es ist unerheblich, ob die geraden oder ungeraden Zeilen verschoben werden. Auch die Richtung der Verschiebung nach rechts oder links hat keine Bedeutung.

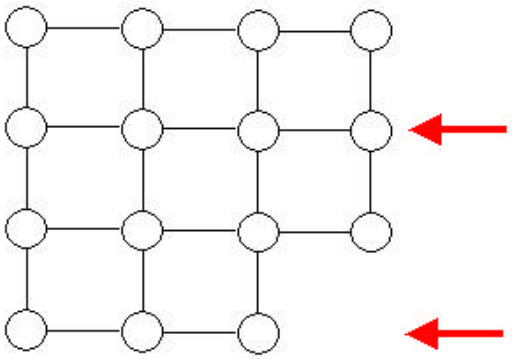


Abb. 4.9: Regelmäßiges, zweidimensionales Gitter, bei dem zur Überführung in die Hexagonalstruktur jede zweite Zeile ausgerückt werden soll

2. Der vertikale Abstand zwischen den Zeilen ist bei der Transformation von einem quadratischen zu einem Hexagonalen Gitter zu verkürzen (siehe Abb. 4.10 und Abb. 4.11). Der Abstand wird gemäß (4.5) von eins auf $\sqrt{3/4}$ reduziert.

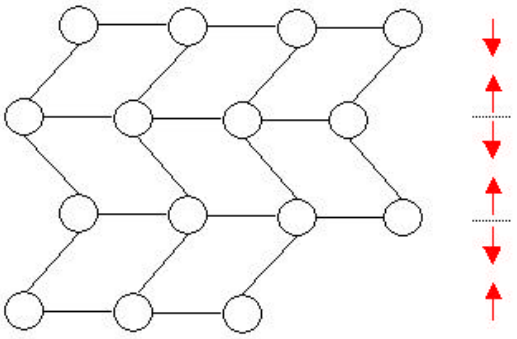


Abb. 4.10: Nach dem Ausrücken jeder Zeile gegenüber Abb. 4.9, muss der Abstand zwischen den Zeilen verkürzt werden, um eine Hexagonalstruktur zu erhalten

3. Zusätzlich zur Lageveränderung der Neuronen werden weitere Verbindungen zwischen den Neuronen geschaffen. Ausgehend von den nicht verschobenen Zeilen werden von jedem Neuron Verbindungen zu den Neuronen der darüber bzw. darunter liegenden Zeile gebildet. Diese Verbindungen stellen zusätzliche Kanten dar. Je nach Verschiebung der Zeile (nach rechts oder links) wird die Kante umgekehrt proportional (nach links oder rechts) eingefügt (siehe Abb. 4.11).

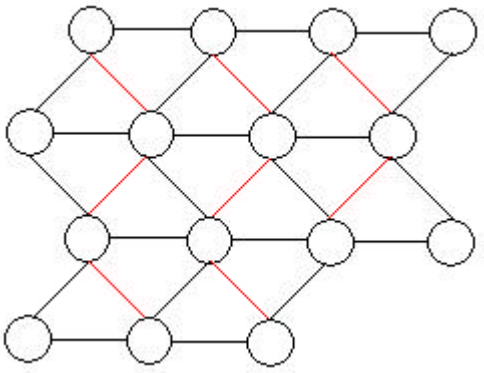


Abb. 4.11: Nach dem Verkürzen des Zeilenabstands wie in Abb. 4.10 skizziert, werden noch weitere Kanten eingezogen und es entsteht ein zweidimensionales, regelmäßiges hexagonales Gitter

Abb. 4.11 zeigt eine zweidimensionale, regelmäßige hexagonale Gitterstruktur. Die Überführung des quadratischen Gitters in ein hexagonales Gitter ist ohne weiteres möglich. Diese Erkenntnisse lassen sich zur Abstandsberechnung im hexagonalen Gitter nutzen. Die Abstandsberechnung wird zum Anlernen des Netzes benötigt. Wie bei den quadratischen Gittern beruht die Abstandsberechnung auf den Koordinaten der Neuronen im Neuronengitter. Die Überführung des quadratischen in ein hexagonales Schema lässt sich auf die Koordinatenberechnung der Neuronen übertragen.

Für eine formale Darstellung muss die zu verschiebende Zeile konkret angegeben werden. Ferner ist auch die Richtung der Verschiebung festzulegen. Seien dazu die Neuronen jeder zweiten (geraden) Zeile horizontal verschoben. Die Verschiebung erfolgt um ½ nach links. Die Koordinaten des Neurons N_i, eines zweidimensionalen, hexagonalen, regelmäßigen Gitters, ergeben sich als:

$$x_{KOO-hex}(N_i) = \begin{cases} i \bmod q_{ROW} & , \quad \lfloor i / q_{ROW} \rfloor \ div\ 2 = 0 \\ (i \bmod q_{ROW}) - 0{,}5 & , \quad \lfloor i / q_{ROW} \rfloor \ div\ 2 \neq 0 \end{cases} \qquad (4.6)$$

$$y_{KOO-hex}(N_i) = \sqrt{\frac{3}{4}} \cdot \lfloor i / q_{ROW} \rfloor$$

Die Abstandsbestimmung zwischen den nach (4.6) berechneten Koordinaten im hexagonalen Gitter erfolgt gemäß der euklidischen Distanz.
Alternativ ist es möglich, die Abstandsbestimmung im hexagonalen Gitter auf eine andere Weise vorzunehmen. Dabei werden die Koordinaten der Neuronen nicht auf eine hexagonale Struktur umgerechnet. Statt dessen ist die Abstandsberechnung zu modifizieren und es wird mit den Koordinaten der Neuronen eines quadratischen Gitters gerechnet. Die gewöhnliche euklidische Abstandsbestimmung ergibt sich formal als:

$$\left\| \begin{pmatrix} x_{Koo}(N_i) \\ y_{Koo}(N_i) \end{pmatrix} - \begin{pmatrix} x_{Koo}(N_{j*}) \\ y_{Koo}(N_{j*}) \end{pmatrix} \right\|_2 = \sqrt{\left(x_{Koo}(N_i) - x_{Koo}(N_{j*})\right)^2 + \left(y_{Koo}(N_i) - y_{Koo}(N_{j*})\right)^2} \quad .$$

In (4.7) ist die Transformation der obigen Abstandsbestimmung dargestellt. Diese ermöglicht es, mit den Koordinaten eines quadratischen Gitters die Abstände eines fiktiven, hexagonalen Gitters zu berechnen.

$$dist = \begin{cases} dist_1 & , \quad \left[y_{Koo}(N_i) - y_{Koo}(N_{j*})\right] div\ 2 \neq 0 \\ dist_2 & , \quad \left[y_{Koo}(N_i) - y_{Koo}(N_{j*})\right] div\ 2 = 0 \end{cases} , \qquad (4.7)$$

mit

$$dist_1 = \sqrt{\left(\left[x_{Koo}(N_i) - x_{Koo}(N_{j*})\right] - 0{,}5\right)^2 + \left(\sqrt{\frac{3}{4}} \cdot \left[y_{Koo}(N_i) - y_{Koo}(N_{j*})\right]\right)^2}$$

und

$$dist_2 = \sqrt{\left(x_{Koo}(N_i) - x_{Koo}(N_{j*})\right)^2 + \left(\sqrt{\frac{3}{4}} \cdot \left[y_{Koo}(N_i) - y_{Koo}(N_{j*})\right]\right)^2} \quad .$$

Damit liegen nun alle Informationen vor, um im Folgenden das Anlernen neuronaler Netze vorzustellen.

4.2 Das Training neuronaler Netze

Das Training eines neuronalen Netzes erfolgt, indem eine Reihe von Daten als Lerneingaben für das Netz geschaltet werden. Dazu werden ein Teil der Ausgangsdaten als Lerneingaben festgelegt. Dieser als Trainigsdatensatz bezeichnete Anteil kann alle zu verarbeitenden Daten oder nur einen Teil der Ausgangsdaten umfassen. In der Lernphase werden alle Werte des Trainingsdatensatzes mindestens einmal als Lerneingabe für das Netz geschaltet. Dieser Vorgang wird als ein Lernzyklus definiert.

Oft werden alle Daten des Trainingsdatensatzes mehrmals hintereinander als Lerneingabe für ein Netz geschaltet. Man spricht von einer Reihe von Lernzyklen. Je größer die Anzahl der Lernzyklen, desto besser ist i. Allg. die Anpassung des Netzes an die Ausgangsdaten. Ist eine Lerneingabe an das Netz angelegt, wird das Netz durch Gewichtsveränderung an die Lerneingabe angepasst. Diese Anpassung wird als Adaption bezeichnet. Bei Anliegen einer Lerneingabe wird jeweils das Gewinner-Neuron bestimmt. Der Gewichtsvektor dieses auch mit Best-Match bezeichneten Neurons hat die kürzeste Distanz zur Lerneingabe.
Sei $P = (p_1,...,p_m)$, $p_k \in \mathbb{R}$, ein Lerneingabevektor der Dimension m für ein neuronales Netz. Sei $W_i = (w_{j1},..., w_{jm})$, $j = 1,...,q$ eine Menge von Gewichtsvektoren der Dimension m, wobei W_i dem Neuron N_i, $i = 1,...,q$ zugeordnet ist. Die Anzahl q steht für die Anzahl der Neuronen der Ausgabeschicht. Der Gewinner N_{j^*} ermittelt sich durch:

$$N_{j^*} : j^* = \arg\min_{j=1,..,q} \| P - W_j \|_2 .$$

Die Gewichte aller Neuronen und insbesondere das des Gewinner-Neurons werden in Richtung der Lerneingabe verändert. Die Veränderung hängt von der Anzahl s der bis einschließlich zu diesem Schritt geschalteten Lerneingaben ab. Daher wird s als Lernschritt bezeichnet. Die Gewichtsanpassung vom Lernschritt s zu $s + 1$ erfolgt gemäß

$$W_i(s+1) = W_i(s) + h(s) \cdot d_{ij^*}(s) \cdot (P - W_i(s)), \quad \forall\, i = 1,..., q . \qquad (4.8)$$

Die Veränderung erfolgt in Abhängigkeit von den Funktion $h(s)$ und $d_{ij^*}(s)$. Die Funktion h wird als Lernrate bezeichnet und d_{ij^*} ist die Nachbarschaftsfunktion.

Die Gestaltung der Lernrate h hängt vom jeweiligen Netztyp ab. Allgemein sind zwei Bedingungen an eine Lernrate zu stellen:

$$\forall\, s : \eta(s) \in [0,1] \qquad \text{und} \qquad \forall\, s > \tilde{s} : h(s) \geq h(\tilde{s}) .$$

Nach Zell (2000) wird für die Lernrate oftmals eine rekursive Definition verwendet, wie die folgende Darstellung zeigt:

$$h(s+1) = a\, h(s), \text{ mit } a \in (0,1). \qquad (4.9)$$

Die Wahl des Faktors α aus (4.9) bestimmt folglich die Konvergenzgeschwindigkeit des Netzes.

Die Nachbarschaftsfunktion d_{ij^*} gibt an, wie stark die Gewichte W_i der Nicht-Gewinner-Neuronen $N_i, i = 1,...,q \setminus j^*$ in Abhängigkeit vom Gewinner-Neuron N_{j^*} angepasst werden. Die Definition dieser Funktion hängt sehr stark vom zu Grunde liegenden Netztyp und der Anwendung ab. Im Falle der WTANs basiert die Nachbarschaftsfunktion auf der Distanz zwischen den Gewichten des Gewinner-Neurons und dem jeweiligen Gewicht des zu modifizierenden Neurons. Bei den SOMs ist die Distanz zwischen den Neuronen in der Ausgabeschicht entscheidend.
Klassische Varianten der Funktion sind die Gaußfunktion (Ritter et al, 1991) oder die Mexican-Hat-Funktion (Kohonen, 1995). Im Falle der gaußschen Nachbarschaftsfunktion wird die euklidische Distanz gemäß der Gaußfunktion transformiert und durch eine geeignete Varianzfunktion $s^2(s)$ normiert. Formal gestalten sich diese Nachbarschaftsfunktionen wie folgt:

Im Falle eines WTAN:

$$d_{ij^*}(s) = e^{-\frac{\left\| W_i(s) - W_{j^*}(s) \right\|^2}{s^2(s)}}. \qquad (4.10)$$

Im Falle eines SOM:

$$d_{ij^*}(s) = Exp\left\{ -\frac{1}{\sigma^2(s)} \left\| \begin{pmatrix} x_{Koo}(N_i) \\ y_{Koo}(N_i) \end{pmatrix} - \begin{pmatrix} x_{Koo}(N_{j^*}) \\ y_{Koo}(N_{j^*}) \end{pmatrix} \right\|^2 \right\}. \qquad (4.11)$$

Durch die Modifikation mit der Exponentialfunktion wirkt sich die Veränderung auf Neuronen mit großem Abstand zum Gewinner-Neuron deutlich weniger aus als auf Neuronen mit geringem Abstand. Wird das Gewinner-Neuron in die Nachbarschaftsfunktion eingesetzt, so erhält man den Wert eins. Das führt auf einen weiteren Aspekt der Nachbarschaftsfunktion. Damit $d_{ij^*}(s)$ für die Modifikation aller Neuronen, also auch des Gewinners N_{j^*}, gilt, muss folgende Bedingung erfüllt sein:

$$\forall\, s \;\; \forall\, i = j^*, \;\; i, j^* \in \{1,...,q\} \;:\; d_{j^*j^*}(s) = \max_{i, j^* \in \{1,...,q\}} d_{ij^*}(s).$$

Damit wird das Gewinner-Neuron am stärksten adaptiert. Gemäß (4.8) werden alle Neuronen der Ausgabeschicht modifiziert. Es kann jedoch vorkommen, dass die Nachbarschaftsfunktion $d_{ij^*}(s)$ für „weit entfernte“ Neuronen den Wert null annehmen kann. Dann wird das entsprechende Neuron nicht verändert. In einigen speziellen Definitionen für die Nachbarschaftsfunktion (z. B. Mexican-Hat-Funktion, Zell, 2000) kann $d_{ij^*}(s)$ auch negative Werte annehmen. Dadurch erfolgt eine Abstoßung der Gewichte gegenüber der Lerneingabe.

Bock (1998) zeigt, dass sich klassische Verfahren, wie z. B. das k-Mittelwert-Verfahren, bei geeigneter Wahl der Nachbarschaftsfunktion $d_{ij*}(s)$, als neuronale Netze darstellen lassen. Durch solche Netze werden ebenso wie bei den klassischen Verfahren Clusterungen angestrebt, die möglichst nahe an Bedingung (2.2) herankommen.

4.3 Die verwendeten Netztypen

Die folgenden fünf Netztypen sind im Rahmen dieser Arbeit in die Rechnerumsetzung PicAna implementiert.

1. **Winner-takes-all-Netz**
   Die Lernrate und Nachbarschaftsfunktionen des Winner-takes-all-Netzes müssen für jede Anwendungsumgebung speziell angepasst werden. Daher wird hier eine spezielle Definition der Lernrate in enger Anlehnung an die klassischen Varianten gewählt. Für die Nachbarschaftsfunktion definiere ich eine eigene Variante, unter besonderer Berücksichtigung der speziellen Anwendung in der Erosionsforschung. Diese Funktion ergibt sich aus der Kombination der klassischen, auf der Gaußfunktion basierenden Variante, mit der Mexican-Hat-Funktion und eigenen Überlegungen.

2. **Standard-selbstorganisierende-Karte mit zweidimensionaler, quadratischer Nachbarschaftsstruktur**
   Dieser Netztyp orientiert sich an den von Kohonen (1982) vorgeschlagenen Varianten. Er dient als Basisversion der neuronalen Netze. Damit ist es möglich, das Verhalten der anderen Netze einzuordnen.

3. **Standard-selbstorganisierende-Karte mit zweidimensionaler, hexagonaler Nachbarschaftsstruktur**
   Auf Grund der geänderten Nachbarschaftsstruktur von quadratisch auf hexagonal ist dieser Netztyp eine Abwandlung des unter 2. beschriebenen Typs.

4. **Schnelle-selbstorganisierende-Karte mit zweidimensionaler, quadratischer Nachbarschaftsstruktur**
   Dieses Netz wurde von mir entwickelt, um die Lerngeschwindigkeit des Netzes aus 2. zu erhöhen. Die Lerngeschwindigkeit wird dabei um Faktoren im Bereich zwischen 100 und 1000 erhöht.

5. **Schnelle-selbstorganisierende-Karte mit zweidimensionaler, hexagonaler Nachbarschaftsstruktur**
   Hierbei handelt es sich in Analogie zu 4. um eine schnelle Variante des unter 3. aufgeführten Netztyps.

4.3.1 Datenstruktur

Für die Definition von Lernrate und Nachbarschaftsfunktion muss die Datenstruktur (Güting, 1992) der zu verarbeitenden Daten berücksichtigt werden. Genau wie beim Maximum Linkage basiert die Clusterung mit neuronalen Netzen auf den RGB-Werten der Pixel eines Bildes. Durch das Anlernen des Netzes werden Gewichte im Raum $\Omega = \{0,1,...,255\}^3$ gesucht. Diese Gewichte dienen als Ausgangsbasis für eine Klassifizierung der Pixel. Abb. 4.12 zeigt eine zu Abb. 4.3 analoge Darstellung für den hier vorliegenden Fall der pixelbasierten Bildklassifikation:

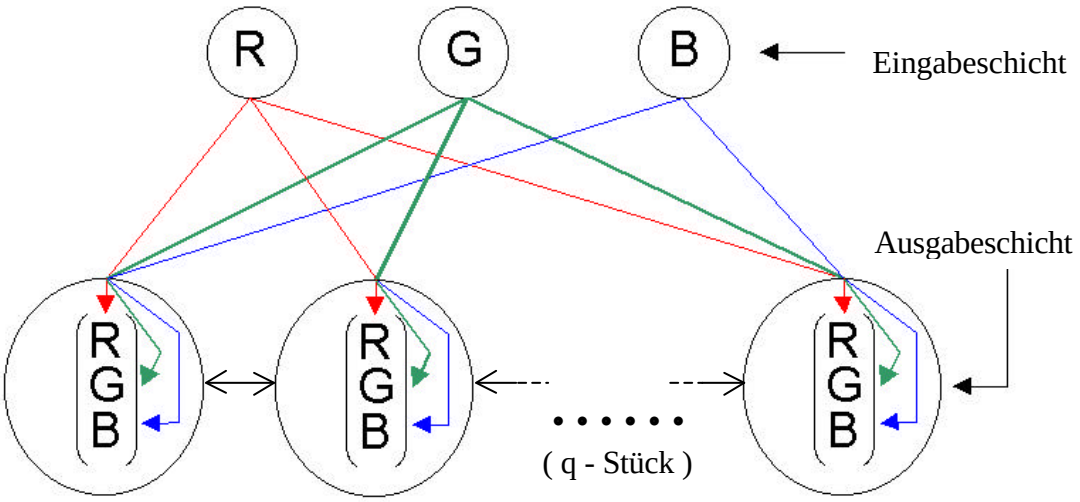


Abb. 4.12: Skizze eines neuronalen Netzes zur pixelbasierten Bildklassifikation. Dies gilt sowohl für WTANs als auch SOMs, da die Klassifikation ohne Berücksichtigung von Nachbarschaftsstrukturen erfolgt.

Die Aktivierungsfunktion bleibt die in (4.2) vorgestellte. Beim Anlernen des Netzes führt der Parameterraum, als Teilmenge der natürlichen Zahlen IN, zu folgendem Problem: Das Training erfolgt über das Schalten von mehreren tausend Lerneingaben. Durch jede Lerneingabe werden die Gewichte minimal geändert. Diese minimalen Veränderungen sind auf einer Teilmenge des IN^3 nicht durchführbar, weil die Zahlenwerte dort nicht „dicht“ genug liegen. Daher muss während der Lernphase der Parameterraum für die Gewichte vom Raum $\Omega = \{0,1,...,255\}^3 \subset IN^3$ auf den Raum $\Omega^* = [\,0\,,\,255\,]^3 \subset IR^3$ erweitert werden. Nach der Lernphase, wenn das Netz konvergiert ist, muss eine Rücktransformation $[\,0\,,\,255\,]^3 \rightarrow \{0,1,...,255\}^3$ durchgeführt werden. Die ermittelten Parameter liegen damit in dem ursprünglich zu Ggrunde gelegten Parameterraum. Die Rücktransformation besteht also darin, die Werte komponentenweise auf natürliche Zahlen zu runden.

4.3.2 Datenvorverarbeitung und Initialisierung

Bei der Bildclusterung mit neuronalen Netzen tritt, wie bereits beschrieben, oft das Problem auf, dass die seltenen Elemente nicht in eigene Klassen eingeordnet werden. Sie werden anderen Klassen zugeordnet und auf Grund ihrer geringen Häufigkeit durch den entsprechenden Klassenrepräsentanten nicht hinreichend gut repräsentiert. Bei der vorliegenden Anwendung werden alle Pixel eines Bildes einmal pro Lernzyklus als Lerneingabe geschaltet. Damit stehen wesentlich mehr Trainingsdaten zur Verfügung als bei typischen Anwendungen neuronaler Netze. Die Häufigkeitsverteilung der unterschiedlichen Werte dieser Lerneingaben variiert sehr stark. Insbesondere sind die seltenen Elemente nur durch wenige Werte repräsentiert. Weil neuronale Netze die Häufigkeitsverteilung der Lerneingaben sehr stark berücksichtigen, können die seltenen Elemente nur schlecht identifiziert werden. Aus diesem Grund empfiehlt Ritter et al (1991), seltene Werte überrepräsentativ häufig als Lerneingabe zu schalten. Dazu müssen die seltenen Elemente bereits vor der Clusterung bekannt sein. Da es sich hier um unüberwachte Verfahren handelt, stehen diese von vornherein nicht fest. Daher muss hier ein anderer Weg gefunden werden, um diesem Problem zu begegnen.
Unter dem Aspekt der ungleichen Häufigkeitsverteilung unterschiedlicher Werte, besteht die Gefahr, anstatt einer guten Repräsentation der Ausgangsdaten einschließlich der seltenen Elemente, nur eine Auswahl der am häufigsten auftretenden Werte zu erhalten. Daher muss die Häufigkeit der Lerneingaben mit großen Häufigkeiten reduziert werden. Dadurch werden die stark differierenden Häufigkeiten angeglichen. Das hat den angenehmen Seiteneffekt, dass bei weniger Lerneingaben pro Zyklus die Lernphase verkürzt wird. Um die Gesamtanzahl an Lerneingaben adäquat zu reduzieren, wird eine Häufigkeitstabelle der Pixel P_i , $i=1,...,r$ mit unterschiedlichen RGB-Werten erstellt. Im Rahmen dieser Arbeit wurde von mir die unter (4.12) dargestellte Funktion zur Häufigkeitsreduktion entwickelt.

$$H^*(P_i) = \left\lceil H(P_i)^{l(H(P_i))} \right\rceil \ , \ i = 1,...,r \ ,$$

mit (4.12)

$$l(H(P_i)) = \frac{arccot\left((1/40)(H(P_i)-80)\right)}{4p} + 0.75 \ .$$

In (4.12) gibt die Funktion $H(P_i)$ die Häufigkeit des Wertes P_i wieder. Die Werte dieser transformierten Häufigkeitstabelle werden Schritt für Schritt zufällig ausgewählt und als Lerneingabe für das neuronale Netz geschaltet. Dabei wird ein Wert insgesamt so häufig zufällig ausgewählt, wie es seine transformierte Häufigkeit $H^*(P_i)$ aus (4.12) vorschreibt.

Die Entwicklung der Funktion (4.12) ergibt sich aus heuristischer Sicht folgendermaßen: Es wird eine monoton wachsende Funktion gesucht, die bei größeren Werten immer flacher wird. Somit werden größere Werte stärker reduziert als kleinere. Eine grundsätzliche Transformation, die das leistet, ergibt sich als:

$$H(P_i)^{0,75}.$$

Neben den theoretischen Überlegungen hat sich diese Transformation auch in empirischen Untersuchungen mit der Programmumsetzung PicAna bewährt. Diese grundsätzliche Transformation muss noch modifiziert werden, weil kleine Häufigkeiten nicht oder nur wenig reduziert werden sollten. Die Werte mit diesen geringen Häufigkeiten sind ohnehin unterrepräsentiert. Weist $H(P_i)$ kleine Werte auf, so wäre die Transformation $H(P_i)^1$ gewünscht. Dazu muss der Exponent von $H(P_i)^{0,75}$ im Anfangsbereich um einen Term additiv ergänzt werden. Der additive Term liegt für kleines $H(P_i)$ bei 0,25. Bei größer werdenden Werten geht er gegen null. Damit läuft der gesamte Exponent von eins nach null. Der Verlauf von 0,25 nach null soll so erfolgen, dass die Transformationsfunktion dennoch monoton wachsend bleibt. Eine passende Funktion ist:

$$l^*(H(P_i)) = \frac{arccot\left((1/40)(H(P_i)-80)\right)}{4p}.$$

Der $arccot(x)$ ist eine stetige, monoton fallende Funktion zwischen p ($x \to -\infty$) und null ($x \to \infty$). Auf Grund der Division durch $4p$ weist die Funktion Werte zwischen $0{,}25$ und null auf, wie es für die Transformation benötigt wird. Das Argument des Arkus-Kotangens von

$$\frac{H(P_i)-80}{40}$$

sorgt dafür, dass der Verlauf von 0,25 nach null in einem für das Ausgangsproblem geeigneten Bereich erfolgt. Diese Werte wurden empirisch ermittelt. Damit gelten folgende Bedingungen für $H^*(P_i)$ aus (4.12):

1. $\forall H(P_i) \le 6 \;:\; H^*(P_i) = id$,
2. $\forall H(P_i) > 1000 \;:\; H^*(P_i) \le H(P_i)^{0,75} + 4$,
3. $H^*(P_i) \to H(P_i)^{0,75} \;,\; (H(P_i) \to \infty)$.

Abb. 4.13 zeigt (4.12) ohne Gaußklammer als flache Kurve und im Vergleich dazu die Identität.

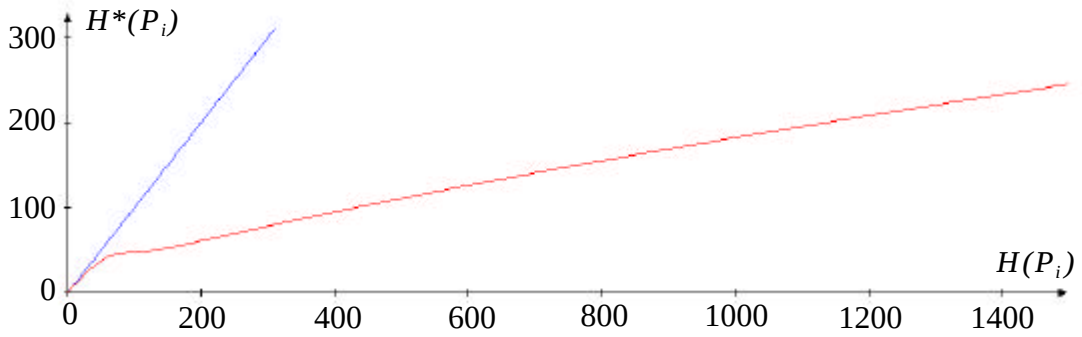


Abb. 4.13: Darstellung der Häufigkeitsreduzierung gemäß (4.12) ohne Gaußklammern im Vergleich mit der Identität.

In der Abbildung ist gut zu erkennen, dass es sich bei (4.12) um eine monoton wachsende Funktion handelt, so dass sich die Rangfolge zweier Häufigkeiten auch nach der Transformation nicht ändert:

$$\forall\, P_i, P_j :\; H(P_i) \le H(P_j) \;\Leftrightarrow\; H^*(P_i) \le H^*(P_j)\,. \qquad (4.13)$$

Außerdem entspricht die Transformationsfunktion für Werte ≤ 6 der Identität, d. h. kleine Häufigkeiten werden nicht verändert. Erst bei größeren Werten tritt eine Reduktion der Häufigkeit auf. Je größer dabei der Wert, desto stärker ist auch die Reduktion. Bei der Transformation wird Bedingung (4.13) nicht verletzt:

$$\forall\, P_i, P_j :\; H(P_i) > H(P_j) \;\Leftrightarrow\; \left(H(P_i) - H^*(P_i)\right) > \left(H(P_j) - H^*(P_j)\right)\,. \qquad (4.14)$$

Zusammenfassend ergeben sich für die Funktion (4.12) folgende Eigenschaften:

- Kleine Werte werden nicht transformiert.
- Je größer der Wert, desto stärker ist die Reduktion.
- Die Rangfolge der Werte wird durch die Transformation nicht verändert.

Durch die Häufigkeitstransformation (4.12) der Lerneingaben wird die Clusterung durch das neuronale Netz verbessert. Die Verbesserung besteht in einer besseren Identifikation seltener Elemente. Durch die starke Reduzierung großer Häufigkeiten wird eine Angleichung zwischen selten und häufig auftretenden Werten erreicht. Dadurch können seltene Elemente leichter gefunden werden. Diese werden dann in eigenen Klassen repräsentiert. Die wesentlichen Eigenschaften der Ausgangsdaten bleiben bei der Datenreduktion jedoch erhalten, sodass beim anschließenden Anlernen des Netzes die Zentroide als Basis für eine Clusterung ermittelt werden können.

Zu Beginn einer Lernphase müssen die Gewicht der Neuronen initialisiert werden. Diese Initialisierung wird in der Literatur meist nur spärlich behandelt, obwohl sie je nach Netztyp eine entscheidende Rolle spielt.
Die Grundidee ist, die Anfangsgewichte der Neuronen zufällig gemäß einer Gleichverteilung aus dem Parameterraum zu wählen. Obwohl eine derartige Initialisierung bei den Standard-SOMs grundsätzlich möglich ist, wird dadurch das Training i. Allg. verlängert. Die Verlängerung tritt auf, wenn Gewichte in Bereichen initialisiert wurden, die Weit entfernt von den Lerneingaben liegen. Dann ist eine Vielzahl von Lernschritten bzw. -zyklen nötig, bis diese Gewichte in repräsentative Bereiche verschoben sind. Der nächste Abschnitt zeigt, wie die Initialisierung für alle Netztypen in der Programmumsetzung PicAna gewählt ist.

4.3.3 Die Lernparameter des Winner-takes-all-Netzes

Damit ein Netz trainiert werden kann, müssen Lernrate und Nachbarschaftsfunktion unter Berücksichtigung des Ausgangsproblems definiert werden. Die Lernrate $h(s)$ wird für das WTAN in Analogie zu Kohonen (1996) als

$$\eta_b(s) = b^s, \qquad b \in (0,1), \tag{4.15}$$

definiert. Die Wahl der Konstante b aus (4.15) muss unter Berücksichtigung der Gesamtanzahl an Lernschritten s_{max} getroffen werden. Die Konstante b ermittelt sich aus der Pixelanzahl des zu clusternden Bildes bzw. der Anzahl an Trainingsdaten, die zur Verfügung stehen. Ferner muss die Anzahl an Lernzyklen, die vorgenommen werden sollen, berücksichtigt werden. Durch Multiplikation beider Größen wird der Wert s_{max} bestimmt.

$$s_{max} = \#Lerneingaben\ pro\ Lernzyklus \cdot \#Lernzyklen .$$

Angenommen, der letzte Schritt soll einer Lernrate von 0,005 genügen. Damit bestimmt sich b gemäß:

$$b = (0{,}005)^{1/s_{max}} .$$

Die in Abb. 4.14 dargestellte Struktur der Lernrate unterstützt das Konvergenzverhalten des neuronalen Netzes. Dabei wurde ein s_{max} von 500.000 und ein b wie oben angegeben gewählt. Der Wert von 0,005 hat sich in empirischen Studien mit der Rechnerumsetzung PicAna bewährt.

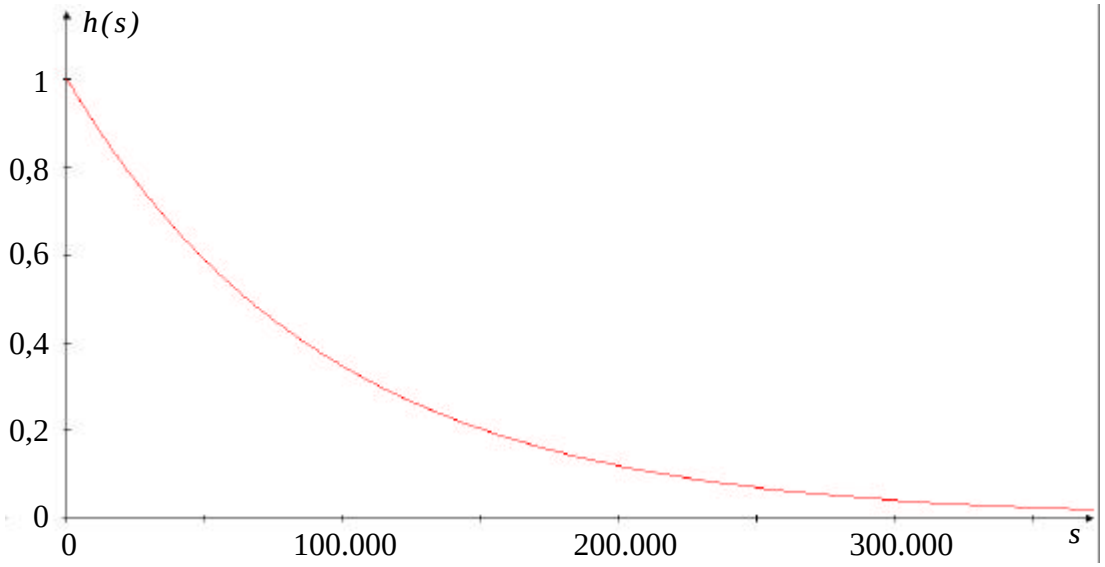


Abb. 4.14: Beispiel einer Lernrate $h(s)$ gemäß (4.15) mit $s_{max} = 500.000$

Die Abstandsfunktion d_{ij} beruht hier nur auf den Gewichten der Neuronen. Des Weiteren wird die Funktion unabhängig vom Lernschritt s definiert, weil die Größe s bereits in der Lernrate enthalten ist. Da sich die Adaption jedes Lernschritts durch Multiplikation von Lernrate und Abstandsfunktion ergibt, hat s einen indirekten Einfluss auf den durch die Nachbarschaftsfunktion definierten Lernradius. Der Lernradius gibt den Bereich um den Gewinner an, in dem die Neuronengewichte modifiziert werden. Weil der Parameterraum in der vorliegenden Anwendung Teilmenge des $\mathbb{N}^3$ und nicht des $\mathbb{R}^3$ ist, wird eine andere Funktion für d_{ij} als in (4.10) benötigt. Das liegt daran, dass beim WTAN die Adaption der Neuronen im Wesentlichen auf Grund ihrer Gewichte vorgenommen wird. Auf Grund der Eigenschaften des Parameterraumes können die Gewichtsvektoren zweier oder mehrerer Neuronen durch die Modifikation des Lernprozesses den selben Wert annehmen. Diese Neuronen sind sozusagen „übereinander geschoben“ worden. Die Gewichte übereinandergeschobener Neuronen können nicht wieder voneinander getrennt werden. Sie werden im Weiteren Verlauf der Lernphase immer gleich verändert. Dadurch geht für die anschließende Klassifikation, und damit auch für die Clusterung, eine Klasse verloren. Dieser unerwünschte Effekt ist unbedingt zu vermeiden. Durch spezielle Abstimmung der Nachbarschaftsfunktion ist es möglich, diesen Effekt auszuschalten. Dazu ist die Nachbarschaftsfunktion so geartet, dass die Gewichtsvektoren von Neuronen, deren Gewichtsvektor nahe am Gewichtsvektor des Gewinner-Neurons liegen, entgegengesetzt zur Lerneingabe verschoben werden. Dazu weist die Nachbarschaftsfunktion für kleine Abstände zwischen dem Gewicht des Gewinner-Neurons und dem des zu modifizierenden Neurons negative Werte auf. Das führt zu der gewünschten Abstoßung. Dadurch wird zum einen das Aufeinanderschieben von Gewichtsvektoren verhindert. Zum Anderen ist dieser Bereich des Parameterraumes

hinreichend gut durch das Gewinner-Neuron repräsentiert. Dadurch wird eine hohe Trennschärfe zwischen den Neuronen erreicht. Das ist wichtig, damit deutlich getrennte Klassen herausgearbeitet werden können.
Für die übrigen Bereiche soll d_{ij} ähnlich zu den von Kohonen etablierten Varianten gewählt werden. Dort treten nur noch positive Werte auf, die mit zunehmendem Abstand allmählich abfallen. Der Maximalwert für d_{ij} soll erreicht werden, wenn beide Indizes von d_{ij} dem Gewinner-Neuron zuzuordnen sind. Je größer die Unterschiede zwischen dem Gewicht des zu modifizierenden Neurons und dem des Gewinners sind, desto kleiner soll d_{ij} sein. Eine Funktion mit diesen Eigenschaften, die zudem noch ein gutes Konvergenzverhalten für die Netzausbildung gewährleistet, lässt sich wie folgt definieren. Dabei ist zu beachten, dass die konkrete Abstimmung von Parametern nur empirisch möglich ist (Kohonen, 1982). Die Werte der folgenden Nachbarschaftsfunktion und insbesondere die Größe des Abstoßungsbereichs sind anhand der Rechnerumsetzung PicAna ermittelt worden. Dazu wurde der Vergleich mit dem Maximum-Linkage-Algorithmus und den Standard-selbstorganisierenden-Karten herangezogen.

$$d_{ij^*}(s) = \begin{cases} 0{,}5 & , \quad \left\| W_i(s) - W_{j^*}(s) \right\|_2 = 0 \\ -0.1 & , \quad \left\| W_i(s) - W_{j^*}(s) \right\|_2 < 15 \\ \dfrac{1}{\left(\left\| W_i(s) - W_{j^*}(s) \right\|_2 \right)^{3/2} + 1} & , \quad \text{sonst} \end{cases} . \qquad (4.16)$$

Zur Visualisierung von (4.16) sei folgende Transformation im Definitionsbereich betrachtet. Dabei wird der ursprüngliche Abstand des Neurons vom Gewinner-Neuron berücksichtigt. Es können negative Werte auftreten, die eine Abweichung in eine andere Richtung aufzeigen:

$$d_{ij^*}(s) = \begin{cases} 0{,}5 & , \quad dist = 0 \\ -0.1 & , \quad 0 < |dist| < 15 \\ \dfrac{1}{|dist|^{3/2} + 1} & , \quad \text{sonst} \end{cases}, \qquad (4.17)$$

mit

$$dist = sign\left(W_i(s) - W_{j^*}(s)\right) \cdot \left\| W_i(s) - W_{j^*}(s) \right\|_2 .$$

Mit Hilfe dieser Transformation lässt sich eine anschauliche Darstellung der Funktion $d_{ij}(s)$ unabhängig von s erzeugen, wie sie in Abb. 4.15 vorliegt.

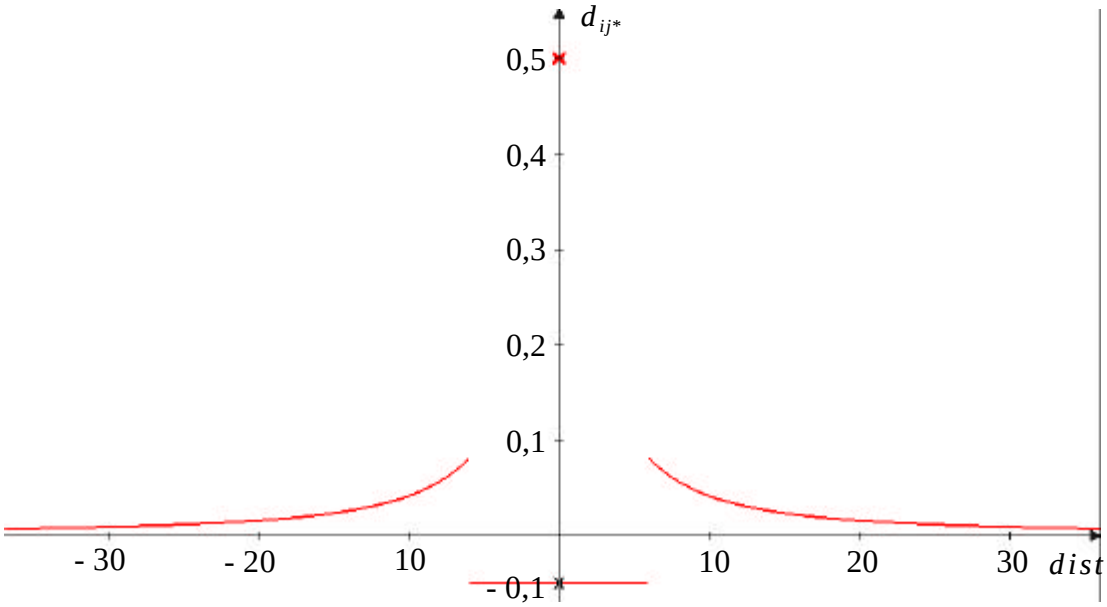


Abb. 4.15: Visualisierung der Abstandsfunktion d_{ij} in der Darstellung aus (4.17).

Eigene empirische Studien mit der Rechnerumsetzung PicAna haben gezeigt, dass eine bessere Anpassung des Netzes an die Ausgangsdaten möglich ist, wenn die Lernrate h der abschnittsweisen Funktion d_{ij} aus (4.16) angepasst wird. Definiere h^*:

$$h_b^*(s) = \begin{cases} h_b(s) & , \left\| W_i(s) - W_{j^*}(s) \right\|_2 < 15 \\ h_b(s)^2 & , \left\| W_i(s) - W_{j^*}(s) \right\|_2 \geq 15 \end{cases} . \qquad (4.18)$$

Damit ist der Gesamtmodifikator jedes Lernschrittes, der sich aus dem Produkt der Lernrate aus (4.18) und der Nachbarschaftsfunktion aus (4.16) ergibt, folgendermaßen definiert:

$$h^{*}_{0{,}005^{s/s_{max}}}(s)\cdot d_{ij^{*}}(s)=\begin{cases} 0{,}05^{s/s_{max}}\cdot 0{,}5 & , \quad \left\| W_i(s)-W_{j^{*}}(s)\right\|_2 = 0 \\ 0{,}05^{s/s_{max}}\cdot(-0.1) & , \quad 0 < \left\| W_i(s)-W_{j^{*}}(s)\right\|_2 < 15 \\ 0{,}05^{2s/s_{max}}\cdot\dfrac{1}{\left(\left\| W_i(s)-W_{j^{*}}(s)\right\|_2\right)^{3/2}+1} & , \quad \text{sonst} \end{cases} . \qquad (4.19)$$

Sei (4.19) analog zu (4.17) transformiert. Dann ergibt sich:

$$h^{*}_{0{,}005^{s/s_{max}}}(s)\cdot d_{ij^{*}}(s)=\begin{cases} 0{,}05^{s/s_{max}}\cdot 0{,}5 & , \quad dist = 0 \\ 0{,}05^{s/s_{max}}\cdot(-0.1) & , \quad 0 < \left| dist\right| < 15 \\ 0{,}05^{2s/s_{max}}\cdot\dfrac{1}{\left(\left| dist\right|\right)^{3/2}+1} & , \quad sonst \end{cases} , \qquad (4.20)$$

mit

$$dist = sign\left(W_i(s)-W_{j^{*}}(s)\right)\cdot\left\| W_i(s)-W_{j^{*}}(s)\right\|_2 .$$

Abb. 4.16 veranschaulicht die in (4.20) zusammengefasste Funktion:

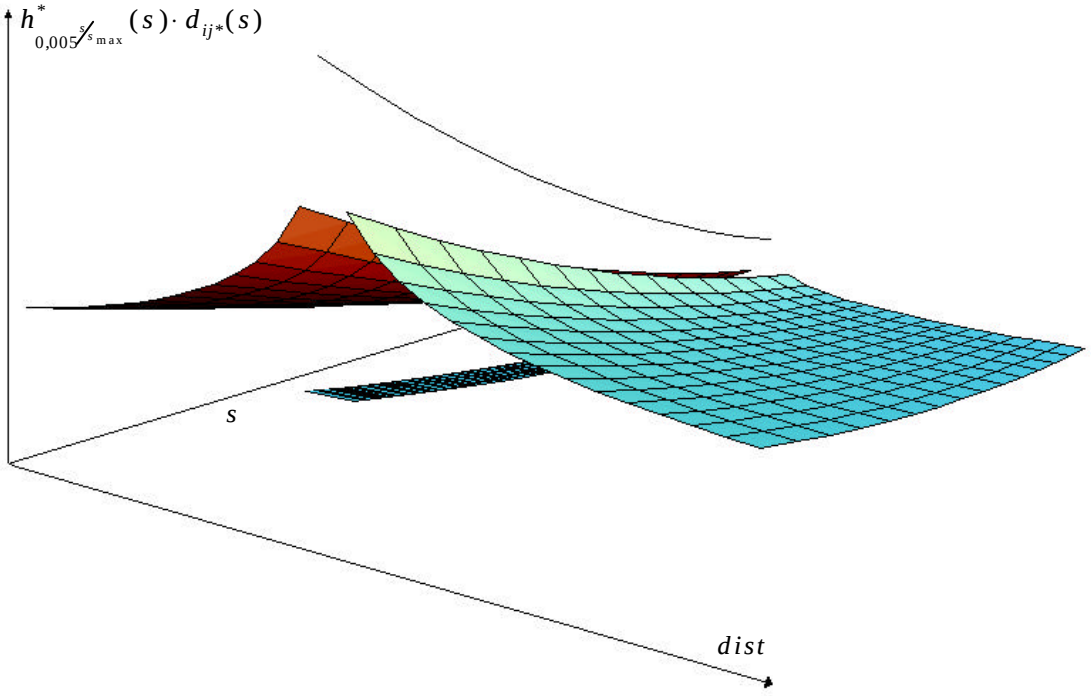


Abb. 4.16: Produkt aus Lernrate und Abstandsfunktion

Bei den WTANs führt eine Initialisierung der Gewichte mit völlig zufälligen Werten zu größeren Problemen. Wie bereits dargelegt, treten diese bei Gewichtsgleichheit von Neuronen auf. Bei einer zufälligen Initialisierung können aber identische Gewichte auftreten. Deshalb wird an die Gewichte des Initialisierungsschritts $W_i(0)$ für alle q Neuronen folgende Bedingung gestellt:

$$W_i(0) \neq W_j(0), \quad \forall\, i \neq j, \quad i, j \in \{1,...,q\}.$$

Diese Bedingung lässt sich auch algorithmisch leicht umsetzen und wird in der Programmumsetzung PicAna für alle Netztypen angewandt. Auch die Initialisierung mit Werten, die weit von den zu verarbeitenden Ausgangswerten liegen, hat bei den WTANs und den schnell lernenden SOMs größere Auswirkungen als nur eine Laufzeitverlängerung des Algorithmus. Wie die Definition der Nachbarschaftsfunktionen aus (4.16) zeigt, werden Neuronen, die vom Gewinner weit entfernt liegen, nur wenig adaptiert. Angenommen, es ist ein Gewicht initialisiert worden, das einen Bereich des Merkmalsraumes repräsentiert, in dem keine der zu verarbeitenden Ausgangsdaten vorliegen. In der Lernphase kann ein solches Gewicht nicht in repräsentative Bereiche überführt werden, weil es selber nicht zum Gewinner wird und daher nur wenig adaptiert wird. Bei der Klassifizierung der Daten nach der Lernphase werden einem solchen Wert keine Datenwerte zugeordnet. Dadurch geht eine Klasse verloren.
Die Lösung besteht darin, die Gewichte der Neuronen von vornherein „geeignet“ zu initialisieren. Dabei bedeutet „geeignet“, dass die initialisierten Gewichte aus einer e-Umgebung um die realen Bilddaten stammen. Der Anwender kann e in Abhängigkeit vom Anwendungsbereich und der Datenlage spezifizieren. Der Wert für e kann von null bis unendlich reichen. Im Fall $e = 0$, werden die Gewichte mit zufälligen Werten der zu verarbeitenden Ausgangsdaten initialisiert. Im Fall $e = \infty$, werden die Gewichte rein zufällig initialisiert. Ist e gegeben, so muss gelten:

$$\text{Sei } e > 0, \text{ fest}: \forall\, i \in \{1,...,q\}\ \exists\, j \in \{1,...,r\} : \left\| W_i(0) - P_j \right\|_2 \leq e \ . \qquad (4.21)$$

Dabei gibt r die Anzahl unterschiedlicher Datenwerte und q die Anzahl gesuchter Klassen an. $W_i(0)$ steht für das Gewicht des i-ten Neurons im Initialisierungsschritt $s = 0$ und P_j gibt den j-ten Datenwert an. Der Radius der Umgebung, innerhalb der die Gewichte um die Daten streuen dürfen, wird mit e bezeichnet.

Mit Hilfe des so definierten Winner-takes-all-Netzes lassen sich Pixel-Clusterungen beliebiger Bilder erstellen. Die empirische Untersuchung in Kapitel 7 wird zeigen, dass die Konvergenzgeschwindigkeit dieses Netzes sehr groß ist. Dadurch werden nur wenige Lernzyklen benötigt. Es kann gegenüber den selbstorganisierenden Karten relativ schnell eine Clusterung erstellt werden.

4.3.4 Die Lernparameter der Standard-selbstorganisierenden-Karten

Bei den Standard-selbstorganisierenden-Karten (S-SOMs), wie sie Kohonen (1982) vorstellt, treten gegenüber dem vorgestellten Winner-takes-all-Netz einige Veränderungen auf. Diese Art von Netz lernt sehr langsam. Daher wird im Gegensatz zu der allgemeinen Vorstellung der Lernraten aus (4.9) eine andere Variante gewählt. Diese hängt nicht vom Lernschritt s, sondern dem aktuellen Lernzyklus zks und der Gesamtanzahl an Lernzyklen zk ab. Die Lernrate ändert sich damit nicht mehr mit jeder neuen Lerneingabe, sondern erst bei Wechsel von einem Lernzyklus zum nächsten. Bei der Lernrate aus (4.22) nach Kohonen (1982) geht noch ein weiterer Parameter ein. Dieser wird als Adaptionshöhe ah bezeichnet. Die Adaptionshöhe liegt im halboffenen Intervall (0,1] und wird vor Beginn der Lernphase festgesetzt. Die Adaptionshöhe ah legt fest, wie stark sich die Anpassung einer Lerneingabe auf das Netz auswirkt. Je größer die Adaptionshöhe, desto stärker ist die Anpassung. Kohonen (1982) definiert die Lernrate h für die S-SOMs wie folgt:

$$h(zk; ah; zks) = \frac{ah \cdot (zk - zks)}{zk} \quad . \qquad (4.22)$$

Die Anzahl der Lernzyklen zk beeinflusst proportional die Qualität der Netzanpassung. Die Praxis zeigt, dass erst nach 200 bis 500 Lernzyklen verwertbare Netze entstehen. Gut angepasste Netze werden ab 1000 Lernzyklen erzielt. Entsteht zunächst keine gute Clusterung, muss die Netzanpassung ggf. mehrfach mit geänderten Parametern (e für Initialisierung und Lernzyklen zks) wiederholt werden. In Abb. 4.17 ist die Lernrate aus (4.22) dargestellt. Dabei wurden die Parameter wie folgt gewählt: Adaptionshöhe $ah = 1$ und Anzahl der Lernzyklen $zk = 1000$.

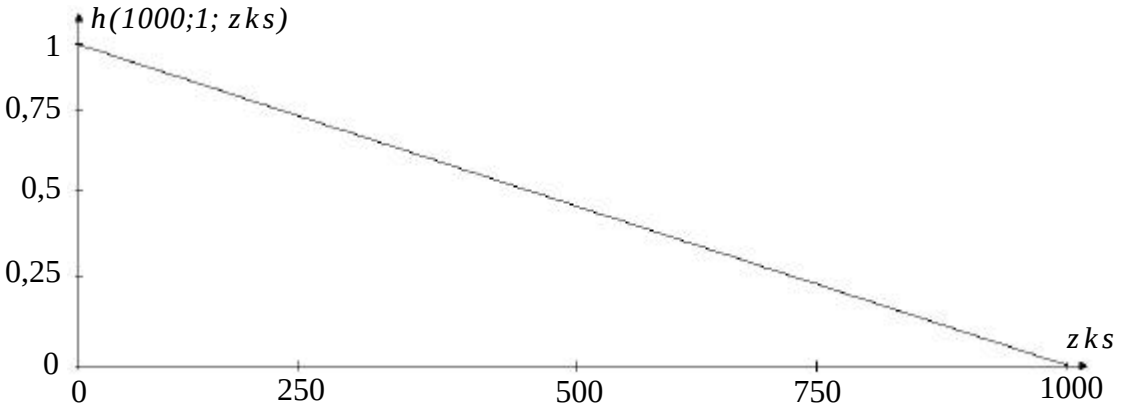


Abb. 4.17: Beispiel der Lernrate $h(1000;1; zks)$ gemäß Formel (4.22)

Auch die zweite Einflussgröße der Lernfunktion aus (4.9), die Nachbarschaftsfunktion $d_{ij\text{-}S\text{-}SOM}$, ist bei den Standard-selbstorganisierenden-Karten anders als beim

WTAN. Die Funktion basiert auf dem Abstand der Neuronen auf der Kohonenkarte und nicht mehr auf dem Abstand der Gewichte. Bei der Herleitung der Koordinaten zur Abstandsberechnung, wird nur auf zweidimensionale Strukturen eingegangen. Eigene empirische Versuche haben gezeigt, dass die Nachbarschaftsstruktur, die durch die Farbwerte in Aufnahmen aus erosionsgefährdeten Gebieten Nordafrikas induziert wird, ansatzweise linearen Charakter hat. Trotz dieser Tatsache soll eine zweidimensionale Struktur herausgearbeitet werden, da diese bessere Möglichkeiten für eine inhaltliche Interpretation bietet. Im Falle der quadratischen Kohonenkarte wird die Nachbarschaft auf Grund der Koordinaten der Neuronen nach (4.3) bestimmt. Bei hexagonaler Karte sind die Koordinaten der Neuronen nach (4.6) entscheidend. Die unterschiedliche Struktur der Kohonenkarte drückt sich nur durch die Abstandsbestimmung auf Grund der Neuronenkoordinaten aus. Die Nachbarschaftsfunktion selbst ist in beiden Fällen identisch. Außerdem hängt $d_{ij\text{-}S\text{-}SOM}$, wie auch die Lernrate der SOMs, nicht vom Lernschritt s ab, sondern vom aktuellen Lernzyklus zk und der Gesamtanzahl an Lernzyklen zks. Nach Kohonen (1982) gibt es noch einen weiteren Parameter, der als Adaptionsradius ar bezeichnet wird. Damit definiert sich die Nachbarschaftsfunktion für die regelmäßigen, zweidimensionalen SOMs gemäß:

$$d_{ij^*\text{-}S\text{-}SOM}(zk;ar;zks) = Exp\left\{ -\frac{\left(\left\| \begin{pmatrix} x_{Koo}(N_i) \\ y_{Koo}(N_i) \end{pmatrix} - \begin{pmatrix} x_{Koo}(N_{j^*}) \\ y_{Koo}(N_{j^*}) \end{pmatrix} \right\|_2 \right)^2}{\left(\frac{ar \cdot (zk - zks)}{zk} \right)^2} \right\} . \quad (4.23)$$

Wie (4.23) zeigt, basiert die grundsätzliche Struktur dieser Nachbarschaftsfunktion auf der Normalverteilung. Der mit ar bezeichnete Adaptionsradius (Lernradius) ist ein Parameter, der angibt, wie stark sich der Abschwächungseffekt des Lernens für weiter entfernt liegende Neuronen auswirkt. Je größer der Adaptionsradius, desto stärker werden auf der Kohonenkarte weiter entfernt liegende Neuronen mit angepasst. Des Weiteren weist die Funktion keine Abstoßungsbereiche wie bei der Nachbarschaftsfunktion der Winner-takes-all-Netze auf. Das liegt daran, dass das Übereinanderschieben der Gewichte der Neuronen beim SOM nicht problematisch ist. Die Koordinaten sind für alle Neuronen unterschiedlich und ändern sich selbst nie. Daher werden aufeinander geschobene Gewicht im nächsten Lernschritt wieder getrennt. Das Aufeinanderschieben von Gewichten tritt bei den SOMs also nur temporär auf. Zum Abschluss der Lernphase sind die Gewichte aller Neuronen unterschiedlich. Ansonsten ist die Nachbarschaftsfunktion unpassend definiert. Durch eine andere Wahl der Nachbarschaftsfunktion lassen sich diese Probleme umgehen. Für eine an-

schauliche Präsentation der Nachbarschaftsfunktion wird die Funktion (4.23) wie folgt verändert.

$$d_{ij^*-S-SOM}(zk;ar;zks) = Exp\left\{ -\frac{(dist)^2}{\left(\frac{ar\cdot(zk-zks)}{zk}\right)^2} \right\} \quad , \qquad (4.24)$$

mit

$$dist = sign\left(\begin{pmatrix} x_{Koo}(N_i) \\ y_{Koo}(N_i) \end{pmatrix} - \begin{pmatrix} x_{Koo}(N_{j^*}) \\ y_{Koo}(N_{j^*}) \end{pmatrix} \right) \cdot \left\| \begin{pmatrix} x_{Koo}(N_i) \\ y_{Koo}(N_i) \end{pmatrix} - \begin{pmatrix} x_{Koo}(N_{j^*}) \\ y_{Koo}(N_{j^*}) \end{pmatrix} \right\|_2 \quad .$$

Um die Abhängigkeit der Nachbarschaftsfunktion vom aktuellen Lernzyklus zk zu verdeutlichen, ist in Abb. 4.18 die Nachbarschaftsfunktion für drei unterschiedliche Lernzyklen grafisch dargestellt. Dabei wurde die Anzahl der Lernzyklen auf 1000 und die Adaptionshöhe auf 0,999 festgesetzt. Für den aktuellen Lernzyklus werden die Werte 700, 500 und 100 eingesetzt:

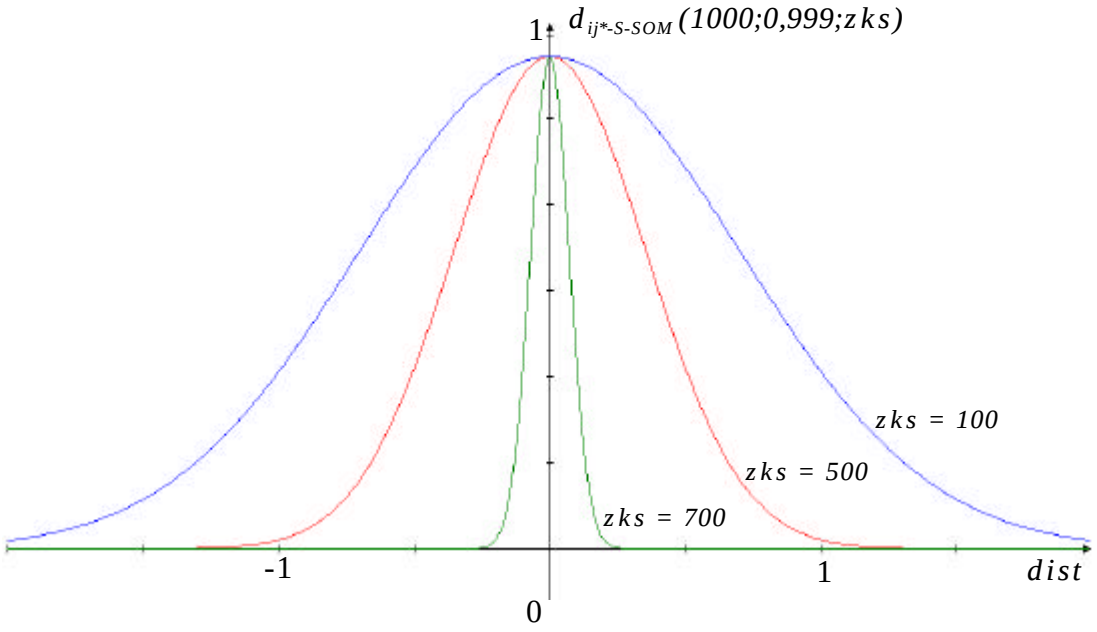


Abb. 4.18: Drei Beispiele für die Nachbarschaftsfunktion $d_{ii\text{-}S\text{-}SOM}$ mit den Parametern: Anzahl der Lernzyklen $zk = 1000$; Adaptionshöhe $ah = 0{,}999$ und drei Werte für den aktuellen Lernzyklus $zks = 100$; $zks = 500$; $zks = 700$.

Wie erwähnt, benötigen diese Netze eine Vielzahl von Lernzyklen, um gute Ergebnisse zu liefern. Daher ist es von großem Interesse, für diese Art der Netze eine

gleich gute, aber deutlich schnellere Variante zu finden: Dazu werden die schnellen selbstorganisierenden Karten betrachtet.

4.3.5 Die Lernparameter der schnellen selbstorganisierenden Karten

Um schnelles Lernen im Rahmen der selbstorganisierenden Karten zu erreichen, werden ähnliche Strategien wie bei den Winner-takes-all-Netzen verfolgt. Die Lernparameter werden analog definiert.

Die Lernrate wird identisch zu der des Winner-takes-all-Netzes gewählt. Diese Lernrate hat sich dort bereits bewährt, um in kurzer Zeit ein neuronales Netz ausreichend zu trainieren. Damit hängt sie vom Lernschritt s ab und nicht mehr vom Lernzyklus, wie bei den Standard-SOMs. Das ist dadurch bedingt, dass der Lernvorgang mit wenigen Lernzyklen auskommen soll. Bleibt die Lernrate im gesamten Lernzyklus gleich, kann die Adaption in jedem Lernschritt nur klein sein, weil sonst die Konvergenz des Netzes nicht gewährleistet ist. Das Netz würde sonst innerhalb eines Lernzyklus, je nach Lerneingabe, stark in dessen Richtung verschoben werden. Damit ein Netz schneller lernen kann, muss die Adaption in jedem einzelnen Schritt größer werden.

Die Standard-SOMs lernen langsamer und feiner, während die schnellen SOMs schneller, dafür aber gröber lernen. „Gröber lernen“ bedeutet in diesem Zusammenhang, dass die Anpassung an die Ausgangsdaten weniger gut ist und dadurch gewisse Feinheiten nicht gefunden werden, die aber im Zusammenhang der Erosionsüberwachung nicht so wichtig erscheinen.

Auch die Grundstruktur der Nachbarschaftsfunktion für schnelle selbstorganisierende Karten (Q-SOMs) wird von den WTANs übernommen. Diese Grundstruktur baut auf (4.16) auf, bei der der Abstand zwischen Gewinner und zu adaptierendem Neuron im Nenner steht. Der entscheidende Abstand ist bei den schnellen SOMs, wie auch bei den Standard-SOMs, der auf der Kohonenkarte und nicht der Abstand zwischen den Gewichten der Neuronen. Daher benötigt diese Funktion auch keinen Abstoßungsbereich, um das Aufeinanderschieben von Gewichten zu verhindern.

Die Definition der Nachbarschaftsfunktion für schnelle selbstorganisierende Karten $d_{ij\text{-}Q\text{-}SOM}$ lautet:

$$d_{ij^*-Q-SOM}(s) = 0{,}3 \cdot \left(s^9 \cdot \left\| \begin{pmatrix} x_{Koo}(N_i) \\ y_{Koo}(N_i) \end{pmatrix} - \begin{pmatrix} x_{Koo}(N_{j^*}) \\ y_{Koo}(N_{j^*}) \end{pmatrix} \right\|_2 + 1 \right)^{-1} . \qquad (4.25)$$

Es wird keine spezielle Abstimmung auf Grund der quadratischen oder hexagonalen Struktur der Kohonenkarte benötigt. Diese wird implizit durch die Verwendung der

entsprechenden Koordinaten der Neuronen berücksichtigt. Für eine anschauliche Darstellung, wird eine Transformation des Definitionsbereichs durchgeführt.

$$d_{ij^*-Q-SOM}(s) = 0{,}3 \cdot \left(s^9 \cdot \left| dist \right| + 1 \right)^{-1} \quad , \qquad (4.26)$$

mit

$$dist = sign\left(\begin{pmatrix} x_{Koo}(N_i) \\ y_{Koo}(N_i) \end{pmatrix} - \begin{pmatrix} x_{Koo}(N_{j^*}) \\ y_{Koo}(N_{j^*}) \end{pmatrix} \right) \cdot \left\| \begin{pmatrix} x_{Koo}(N_i) \\ y_{Koo}(N_i) \end{pmatrix} - \begin{pmatrix} x_{Koo}(N_{j^*}) \\ y_{Koo}(N_{j^*}) \end{pmatrix} \right\|_2 \quad .$$

Die grafische Darstellung von (4.26) ist Abb. 4.19 zu entnehmen.

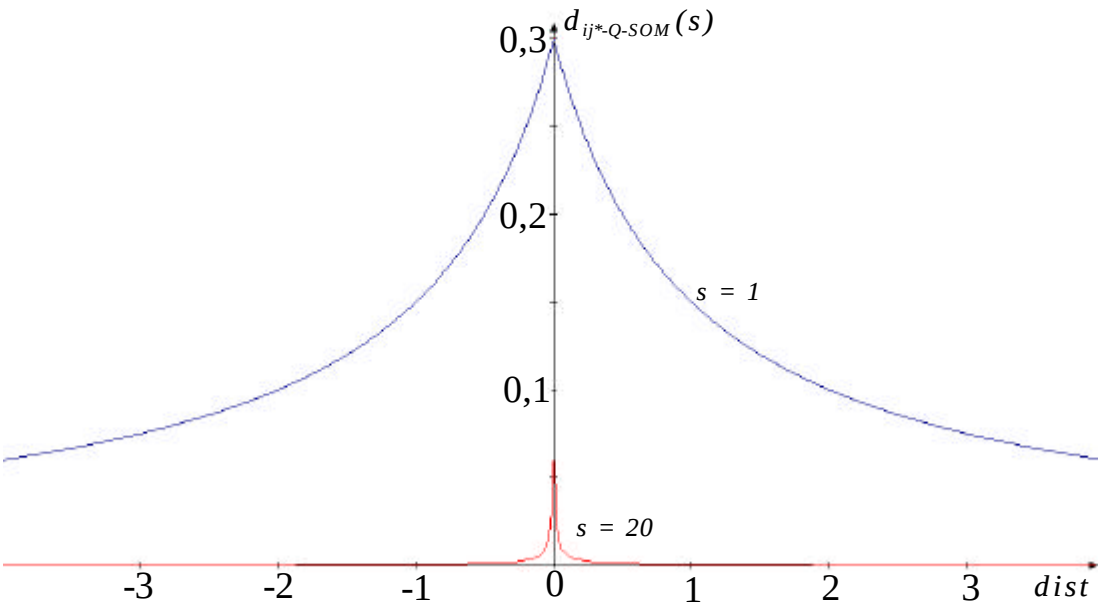


Abb. 4.19: Visualisierung der Nachbarschaftsfunktion für schnelle selbstorganisierende Karten mit den Parametern $s = 1$ und $s = 20$.

Wie sich anhand von Abb. 4.19 zeigt, weist $d_{ij\text{-}Q\text{-}SOM}$ sein Maximum auf, wenn der Abstand zwischen Neuron und Gewinner bei null liegt. Ferner fällt es in jeder Richtung vom Maximum schnell ab. Bei großen Abständen liefert $d_{ij\text{-}Q\text{-}SOM}$ entsprechend kleine Werte zurück. Mit fortschreitenden Lernschritten verstärkt sich dieser Effekt noch. Das zeigt der Vergleich der Kurven für $s = 1$ und $s = 20$. Im fortgeschrittenen Stadium des Lernprozesses wird nur noch der Gewinner angelernt. Die Adaption der

Nachbarn des Gewinners geht gegen null. Auf Grund folgender Effekte können die schnellen SOMs gegenüber den Standard-SOMs rascher trainieren:

1. Die Lernrate der schnellen SOMs ändert sich in jedem Lernschritt s, während sie sich bei den Standard-SOMs nur in jedem Lernzyklus z_k ändert. Daher brauchen die Standard-SOMs mehr Lernzyklen als die schnellen SOMs. Die Konvergenz des Netzes wird durch größere Adaptionsschritte schneller erreicht, auch wenn dadurch das Ergebnis weniger optimal im Sinne der Anpassung an die Ausgangsdaten ist.

2. Die Nachbarschaftsfunktionen von Standard- und schnellen SOMs unterscheiden sich deutlich, wie Abb. 4.18 und Abb. 4.19 zeigen. Bei den Standard-SOMs werden zu Anfang der Lernphase die Neuronen in einem großen Abstand um den Gewinner stark modifiziert. Bei den schnellen SOMs werden von Anfang an nur die Neuronen nahe am Gewinner adaptiert. Dadurch können die einzelnen Lernschritte mit größeren Auswirkungen durchgeführt werden. Weil Lern- und Adaptionsradius begrenzt sind, wird das Netz nicht komplett in eine Richtung verzerrt. Nur die relevanten Bereiche werden adaptiert.

5 Methoden zur Beurteilung von Clusterungen

In Kapitel 3 und 4 sind folgende Clusterungsmethoden vorgestellt worden:

- Maximum Linkage
- Winner-takes-all-Netz
- Standard-selbstorganisierende-Karten mit quadratischer oder hexagonaler Nachbarschaftsstruktur der Kohonenkarte
- Schnelle selbstorganisierende Karten mit quadratischer oder hexagonaler Nachbarschaftsstruktur der Kohonenkarte.

Alle Verfahren haben Vor- und Nachteile. Welche Methode anzuwenden ist, hängt von der Anwendung ab. Im Folgenden werden die Methoden anhand von drei Kriterien verglichen. Diese Kriterien dienen dazu, in der jeweiligen Anwendung die geeignete Clusterungsmethode zu wählen.

5.1 Der Zeitaspekt

Ein Kriterium zur Beurteilung von Clusterungen ist die Zeit, die zur Erstellung benötigt wird. Dabei hängt die Rechenzeit von der Komplexität der verwendeten Methode ab. Je komplexer die Methode ist, desto länger dauert die Berechnung. Dabei sind alle neuen Methoden schneller, als die auf dem k-Mittelwert-Algorithmus beruhenden klassischen Verfahren.
Auf Grund seiner einfachen Struktur, ist der Maximum-Linkage-Algorithmus das bei weitem schnellste Verfahren. Die Clusterung mit dem Winner-takes-all-Netz dauert auf Grund der Anzahl an Berechnungsschritten und der mehrfach zu durchlaufenden Lernzyklen deutlich länger. Bei den selbstorganisierenden Karten dauert die Berechnung erheblich länger, weil neben der eigentlichen Clusterung zusätzlich noch die Nachbarschaftsbeziehung auf der Kohonenkarte herausgearbeitet werden muss. Das ist i. Allg. nur durch eine deutlich höhere Anzahl an Lernzyklen zu erreichen.
Untersucht man die Berechnungsdauer der einzelnen Methoden genauer, so liegt der größte Unterschied in der Art der Bestimmung der Zentroide begründet. Operationen wie das Einlesen der Daten und die Erzeugung der Häufigkeitstabelle sind bei allen Methoden identisch. Auch die Datenvorverarbeitungen, wie Häufigkeitstransformation und Vorinitialisierung der Neuronen, haben vernachlässigbar kleine Auswirkungen auf die Berechnungsdauer. Das Verfahren zur Klassifikation der Werte nach der Bestimmung der Zentroide ist bei allen Methoden gleich.
Der Hauptunterschied zwischen den Methoden liegt in der Anzahl der Distanzberechnungen zur Bestimmung der Zentroide. Die Distanzberechnungen werden beim Maximum Linkage zur Bestimmung der Maximin-Bedingung eingesetzt. Bei den neuronalen Netzen werden sie zur Identifikation des Gewinners sowie zur Modifikation der Neuronen verwendet.

Im Folgenden wird die Anzahl der Rechenoperationen zur Bestimmung der Zentroide für die unterschiedlichen Methoden ermittelt. Als Operation bzw. Rechenoperation wird dabei z. B. das Bestimmen des Abstandes zwischen zwei Punkten gewertet, obwohl dazu mathematisch gesehen mehrere Rechenoperationen notwendig sind. Auch das Aufnehmen eines Wertes in die Zentroidmenge kostet eine Operation. Ebenso wird das Anpassen eines Zentroids, bzw. Gewichts beim Anlernen neuronaler Netze, als eine Operation gewertet.

Maximum-Linkage-Algorithmus

Sei n die Anzahl der Pixel des Ausgangsbildes, r die Anzahl unterschiedlicher Pixel im Bild, das in q Klassen aufgeteilt werden soll. Damit ergeben sich für die Iterationsschritte beim Maximum Linkage nachstehende Anzahlen an Berechnungsschritten:

1.	$\frac{(r-1)\cdot r}{2} + 2$	Operationen zur Bestimmung der ersten zwei Zentroide und Aufnahme der ausgewählten zwei Werte in die Menge der Zentroide,
2.	$2\,(r-2) + 1$	Operationen zur Bestimmung des dritten Zentroids und Übernahme des Wertes in die Menge der Zentroide,
3.	$3\,(r-3) + 1$	Operationen zur Bestimmung des vierten Zentroids und Eintragen des Wertes in die Menge der Zentroide,
..		
q-1.	$(q-1)\,(r-(q-1)) + 1$	Operationen zur Bestimmung des q-ten Zentroid und Aufnahme des Wertes in die Menge der Zentroide.

Insgesamt ergeben sich für den Maximum-Linkage-Algorithmus

$$\frac{(r-1)\cdot r}{2} + 2 + \sum_{i=2}^{q-1} \left(i\left(r-i\right)+1\right).$$

Rechenoperationen.

Winner-takes-all-Netz

Bei den Winner-takes-all-Netzen hängt die Anzahl der Rechenoperationen nur von den n Pixeln des Bildes und der vorgegebenen Klassenzahl q ab.

Es ergeben sich folgende Operationen:

1. Berechnung von q Distanzen zwischen den Gewichten der Neuronen und der Lerneingabe zur Bestimmung des Gewinner-Neurons.
2. Adaption der q Gewichte der Neuronen, dazu für jedes Neuron:
   2 Operationen zur Berechnung von d_{ij},
   1 Operation zur Berechnung von h,
   1 Operation zur Distanzbestimmung zwischen dem Gewicht des Neurons und dem Gewicht der Lerneingabe für die eigentliche Modifikation,
   1 Operation zur Modifikation der Gewichte.

Die oben aufgeführten Operationen müssen für jeden Wert P durchgeführt werden, der zum Training des Netzes verwendet wird. Bei der vorliegenden Bilduntersuchung werden dazu i. Allg. alle n Pixel des Bildes verwendet. Das einmalige Trainieren aller Pixel wird als Lernzyklus bezeichnet. Damit ergeben sich pro Lernzyklus

$$n \cdot (q + q \cdot (2 + 1 + 1 + 1)) = 6 \cdot n \cdot q \quad . \qquad (5.1)$$

an Rechenoperationen.

Selbstorganisierende Karten

Bei den selbstorganisierenden Karten ist zur Berechnung von d_{ij} eine zusätzliche Operation je Neuron und Lerneingabe notwendig, da neben dem Abstand zwischen den Gewichten der Neuronen auch noch der Abstand zwischen den Neuronen auf der Kohonenkarte bestimmt werden muss. Somit ergeben sich

$$6 \cdot n \cdot q + n \cdot q = 7 \cdot n \cdot q \qquad (5.2)$$

Rechenoperationen.

Globale Operationen bei neuronalen Netzen

Bei allen neuronalen Netzen werden die in (5.1) und (5.2) abgeschätzten Rechenoperationen in jedem Lernzyklus einmal ausgeführt. Die Anzahl der Lernzyklen schwankt zwischen Werten von eins und mehreren tausend. Die Abschätzung der Rechenoperationen ist damit nur unter Berücksichtigung dieses Parameters möglich. Daher sind (5.1) und (5.2) mit der Anzahl vorgegebener Lernzyklen z_k zu multiplizieren.

Tab. 5.1 und Tab. 5.2 beinhalten die Abschätzung der Anzahl an Rechenoperationen und stellen diese je Verfahren gegenüber. Zur Veranschaulichung wird dabei ein Beispiel angegeben.

Dazu wird von einem Bild ausgegangen, das aus $n = 768 \times 512 = 393.216$ Pixeln besteht. In diesem Bild gibt es $r = 8730$ unterschiedlichen Farbwerte. Gesucht sind $q = 25$ Klassen.

Methode	**Maximum Linkage**
Rechenoperationen	$\frac{(r-1)\cdot r}{2} + 2 + \sum_{i=2}^{q-1} (i(r-i)+1)$
Rechenoperationen im Beispiel	$4{,}0925105 \cdot 10^7$

Tab. 5.1: Notwendige Rechenoperationen des Maximum-Linkage-Algorithmus bei einem Beispiel für $n = 393.216$ Pixel, $r = 8730$ unterschiedliche Farbwerte und $q = 25$ Klassen

Methode	**Winner-takes-all-Netz**	**selbstorganisierende Karte**
Rechenoperationen	$6 \cdot n \cdot q \cdot zks$	$7 \cdot n \cdot q \cdot zks$
$zk = 1$	$5{,}89824 \cdot 10^7$	$1{,}032192 \cdot 10^{10}$
$zk = 5$	$2{,}94912 \cdot 10^8$	$2{,}58048 \cdot 10^{11}$
$zk = 100$	$5{,}89824 \cdot 10^9$	$1{,}032192 \cdot 10^{14}$
$zk = 1000$	$5{,}89824 \cdot 10^{10}$	$1{,}032192 \cdot 10^{16}$

Tab. 5.2: Notwendige Rechenoperationen des Winner-takes-all-Netzes und der selbstorganisierenden Karte bei einem Beispiel für $n = 393.216$ Pixel, $r = 8730$ unterschiedliche Farbwerte und $q = 25$ Klassen bei 1, 5, 100 und 1000 Lernzyklen.

Aus Tab. 5.2 geht hervor, wie aufwendig die Berechnung einer Clusterung mit Winner-takes-all-Netzen und selbstorganisierenden Karten ist. Im Vergleich dazu, ist das Maximum-Linkage-Verfahren in dem angegebenen Beispiel aus Tab. 5.1 immer schneller als die neuronalen Netze. Selbst bei nur einem einzigen Lernzyklus des Winner-takes-all-Netzes, das ohnehin schon den geringsten Aufwand hat, benötigt der Maximum-Linkage-Algorithmus weniger Rechenzeit, um eine Clusterung zu erstellen.

Unter dem Aspekt, dass neuronale Netze immer mehrere Lernzyklen brauchen und im Falle der Standard-selbstorganisierenden-Karten oft viele tausend Zyklen nötig sind (Kohonen, 1982), tritt der Laufzeitvorteil von Maximum Linkage deutlich hervor.
Ein weiterer interessanter Aspekt ist, dass die Anzahl der Rechenoperationen beim Maximum Linkage von der Anzahl unterschiedlicher Werte r und den Klassen q abhängt. Bei den neuronalen Netzen hängt die Anzahl der Rechenoperationen neben der Anzahl der Klassen q nur noch von der Gesamtanzahl an Bildpunkten n im Ausgangsbild ab. Je nach Größenordnung dieser Parameter kann sich das Verhältnis des Rechenaufwands zwischen den Methoden geringfügig verschieben.

Der Zeitaspekt ist für den Anwender zwar wichtig, aber definitiv entscheidend ist die Qualität einer Clusterung. Zur Beurteilung der Qualität sind Varianzbetrachtungen der Clusterung notwendig, wie sie im folgenden Abschnitt vorgestellt werden.

5.2 Das notwendige Kriterium

Zur Beurteilung einer Clusterung, werden in Analogie zu (3.12) zwei Parameter k_{SSW} für die Inter- und k_{SSB} für die Intra-Class-Varianz definiert (vgl. (2.2)). Damit stehen Größen für vergleichende Aussagen über die Qualität verschiedener Clusterungen zur Verfügung. Wie erwähnt, hängen diese Größen von der Anzahl n zu verarbeitender Daten und der Anzahl q zu bildender Klassen ab. Werden durch unterschiedliche Methoden erzeugte Clusterungen miteinander verglichen, müssen die Parameter n und q bei allen Methoden gleich sein.
Sei zusätzlich m die Dimension der zu verarbeitenden Daten P_j , $j = 1,...,n$ und n_i , $i = 1,...,q$ die Anzahl der Werte P_j der i-ten Klasse. Dann sei:

$$k_{SSB} = \sum_{i=1}^{q} n_i \sum_{k=1}^{m} \left[\left(\frac{1}{n_i} \sum_{j=1}^{n_i} P_{ijk} \right) - \left(\frac{1}{n} \sum_{i=1}^{n} P_{im} \right) \right]^2 . \qquad (5.3)$$

In Vektorschreibweise:

$$k_{SSB} = \sum_{i=1}^{q} n_i \left(\overline{P}_i - \overline{P} \right)' \left(\overline{P}_i - \overline{P} \right) .$$

Analog zu (5.3) ist k_{SSW} definiert durch:

$$k_{SSW} = \frac{1}{n} \sum_{i=1}^{q} \sum_{j=1}^{n_i} \sum_{k=1}^{m} \left(P_{ijk} - \left(\frac{1}{n_i} \sum_{j=1}^{n_i} P_{ijk} \right) \right)^2 . \qquad (5.4)$$

Die Gesamtvarianz k_{SST} ergibt sich als Summe aus k_{SSB} und k_{SSW}. Damit ergibt sich die Gesamtvarianz zu:

$$k_{SST} = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{m} \left(P_{ik} - \frac{1}{n} \sum_{j=1}^{m} P_{jk} \right)^2 . \qquad (5.5)$$

Bei der Rechnerumsetzung der Berechnung von k_{SST}, k_{SSB} und k_{SSW} wird eine andere Darstellung von (5.3) bis (5.5) verwendet. Diese ist notwendig, weil bei der Berechnung nicht alle n Datenwerte bekannt sind, sondern nur die r Datenwerte mit unterschiedlichen Ausprägungen für P_j, $j = 1,...,r$ und deren Häufigkeit h_i. Sei also r_i die Anzahl unterschiedlicher Datenwerte der i-ten Klasse und h_{ij} die Häufigkeit des j-ten Datenwertes der i-ten Klasse, dann ergibt sich für k_{SSB} aus (5.3):

$$k_{SSB} = \sum_{i=1}^{q} r_i \sum_{k=1}^{m} \left[\frac{\sum_{j=1}^{r_i} h_{ij} P_{ijk}}{\sum_{j=1}^{r_j} h_{ij}} - \frac{\sum_{i=1}^{q} \sum_{j=1}^{r_j} h_{ij} P_{im}}{\sum_{i=1}^{q} \sum_{j=1}^{r_j} h_{ij}} \right]^2 . \qquad (5.6)$$

Und analog dazu k_{SSW} aus (5.4):

$$\kappa_{SSW} = \frac{\sum_{i=1}^{q} \sum_{j=1}^{r_i} \sum_{k=1}^{m} \left(P_{ijk} - \frac{1}{\sum_{j=1}^{n_j} h_{ij}} \sum_{j=1}^{r_i} h_{ij} P_{ijk} \right)^2}{\sum_{i=1}^{q} \sum_{j=i}^{r_i} h_{ij}} . \qquad (5.7)$$

Es stellt sich an dieser Stelle die Frage, wie Clusterungen anhand dieser Kenngrößen beurteilt werden. Verschiedene Clusterungen, bzw. mit verschiedenen Verfahren erstellte Clusterungen, werden anhand von k_{SSB} verglichen. Die Clusterung mit dem kleinsten k_{SSB} ist als die beste einzustufen, weil sie dem SSW-Clusterungs-Kriterium (2.2) nach Bock (1974) am nächsten kommt. Im hier vorliegenden Fall wird die Entscheidung über die beste Clusterung einem anderen Kriterium überlassen. Anhand von k_{SSB} muss lediglich sichergestellt werden, dass die entstandenen Klassen ein gewisses Maß an Homogenität aufweisen. Daher wird folgende Bedingung eingeführt:

Sei $C = (c_1,...,c_q)$ die Clusterung der Werte $P_i, i = 1,\ldots,r$ mit $C_j = \{P_{j1},\ldots,P_{jr_i}\}$. Für eine Clusterung soll gelten:

$$k_{SSB}(C) \leq \tfrac{1}{3} \cdot k_{SST}(C) . \qquad (5.8)$$

Durch Bedingung (5.8) wird ein gewisses Maß an Homogenität in den Klassen garantiert. Dadurch werden „unsinnige“ Clusterungen, bei denen sehr unterschiedliche Werte in den Klassen zusammengefasst sind, ausgeschlossen.

Obwohl bei einer Clusterung die Bedingung an die Intra-Class-Varianz unbedingt notwendig ist, wird die Entscheidung für eine durch ein bestimmtes Verfahren erzeugte Clusterung von einem zusätzlichen Kriterium abhängig gemacht. Die Beurteilung durch das SSW-Clusterungs-Kriterium reicht nicht aus, weil es eine entscheidende Schwäche aufweist. Clusterungen, die gemäß dieses Kriteriums gebildet werden, neigen dazu, Klassen mit Werten geringer Häufigkeiten nicht zu identifizieren. Es werden für die Werte P_i, mit den größten Häufigkeiten h_i, eigene Klassen gebildet. Die Werte P_j, mit kleinerer Häufigkeit h_j, fallen in die nächstgelegene Klasse, die durch die Werte mit großen Häufigkeiten gebildet wurden. Auf Grund der geringen Häufigkeit, vergrößern diese Werte die Intra-Class-Varianz kaum, auch wenn sie vom Klassenmittelwert – und damit dem Klassenrepräsentanten – einen großen Abstand haben. Damit gehen seltene Elemente verloren, auch wenn sie weit von allen anderen Werten entfernt liegen und deshalb durch eine eigene Klassen repräsentiert werden müssten. Des Weiteren besteht das Problem, dass nahe beieinanderliegende Werte mit großen Häufigkeiten eigene Klassen bilden. Gemäß der inhaltlichen Interpretation gehören diese Werte auf Grund ihrer geringen Distanz in eine einzige Klasse. Andernfalls wird nur eine geringe Trennschärfe zwischen den Klassen erreicht. Das erschwert die Interpretation und kann zu Problemen bei der Weiterverarbeitung in der Erosionsbeobachtung führen. Eine Lösung zur Beurteilung der Trennschärfe bietet das folgende Kriterium:

5.3 Das entscheidende Kriterium

Das entscheidende Kriterium soll zwei Eigenschaften einer Clusterung gewährleisten. Zum einen soll dadurch eine klare Trennung zwischen den Klassen garantiert werden. Zum anderen muss vermieden werden, dass Werte mit geringer Häufigkeit von großen, inhaltlich unpassenden Klassen „verschluckt“ werden. Diese seltenen Elemente sind für die in dieser Arbeit betrachteten Anwendungen sehr wichtig.
Dieses zusätzliche Kriterium wird von der Maximum-Linkage-Bedingung (3.1) abgeleitet. Es beruht auf dem *mittleren minimalen Abstand zwischen den Zentroiden* der Klassen. Im ersten Schritt wird zunächst für jeden Zentroid sein nächster Nachbar bestimmt und die Distanz zwischen beiden ermittelt. Im zweiten Schritt wird über die Distanzen gemittelt. Je größer der so ermittelte Wert ist, desto höher ist die Trennschärfe zwischen den Klassen und desto größer ist auch die Chance, seltene Elemente eindeutig zu identifizieren. Bei Zerbst et al (2001) wird dieses Kriterium

mit $AvgMinDist$ abgekürzt. Das steht für Average Minimal Distance. Formal berechnet sich das Kriterium durch:

$$AvgMinDist = \frac{1}{q}\sum_{j=1}^{q} \min_{i \in \{1,\ldots,q\} \setminus i} \| z_i - z_j \|_2 . \qquad (5.9)$$

Während (3.1) die Maximierung des minimalen Abstands zwischen den Zentroiden vorschreibt, wird durch (5.9) ermittelt, wie groß der mittlere minimale Abstand bei der entsprechenden Clusterung tatsächlich ist.
Dieses Kriterium ist für alle Clusterungen berechenbar, egal mit welchem Verfahren sie erstellt wurden. Damit ist ein entscheidender Vergleich zwischen beliebigen Clusterungen möglich, egal ob diese durch neuronale Netze oder mit Maximum Linkage erzeugt wurden.
Die „beste“ Clusterung wird folgendermaßen bestimmt: Aus allen Clusterungen, die (5.8) genügen, wird die mit dem größten Wert für (5.9) ausgewählt. Dadurch wird eine Clusterung ausgewählt, die sowohl homogene Klassen besitzt, als auch eine hohe Trennschärfe zwischen den Klassen aufweist. Dieser Sachverhalt lässt sich durch die folgenden Skizzen verdeutlichen.

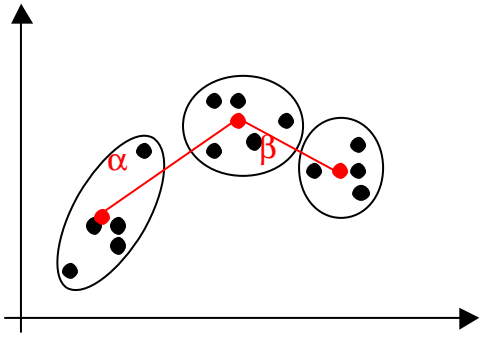


Abb. 5.1.a: Clusterung der schwarzen Punkte. Die Klassen werden durch Ellipsen dargestellt. Die farbigen Punkte stellen die Klassenrepräsentanten dar. Die Länge der farbigen Strecken sind die Ausgangswerte der AvgMinDist. Das Kriterium dieser Clusterung ist ein kleines k_{SSW}.

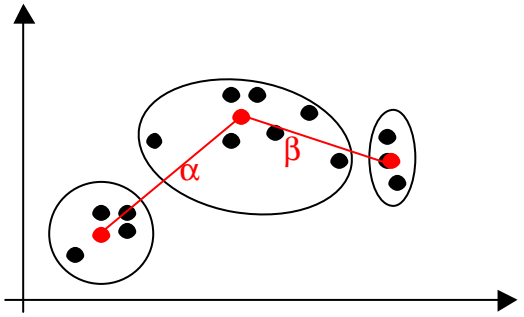


Abb. 5.1.b: Clusterung der schwarzen Punkte. Die Klassen werden durch Ellipsen dargestellt. Die farbigen Punkte stellen die Klassenrepräsentanten dar. Die Länge der farbigen Strecken sind die Ausgangswerte der AvgMinDist. Das Kriterium dieser Clusterung ist eine große AvgMinDist.

Abb. 5.1.a und Abb. 5.1.b skizzieren zwei mögliche Clusterungen der ansonsten identischen schwarzen Punktemenge. Die Ellipsen um die Punkte stellen die Klassengrenzen dar. Außerdem sollen die Größenverhältnisse der Ellipsen ein entsprechendes Verhältnis der k_{SSW} repräsentieren. Dass k_{SSW} auch von der empirischen Häufigkeit der unterschiedlichen Werte in den Klassen abhängt, wird hierbei vernachlässigt. Die farbigen Punkte skizzieren die Klassenrepräsentanten der jeweiligen

Klassen. Die Länge der Strecken α und β stellen die Ausgangswerte für die Berechnung des AvgMinDist dar.
Die Clusterung in Abb. 5.1.a besitzt ein kleineres k_{ssw} als die in Abb. 5.1.b, was durch die kleineren Ellipsoide skizziert wird. Auf Grund der optischen Beurteilung, wirken beide Clusterungen vernünftig. Der Vorteil der Clusterung in Abb. 5.1.b besteht in dem um 12 % größeren AvgMinDist. Dadurch wird eine bessere inhaltliche Trennung der Klassen erreicht, was die Interpretation i. Allg. erleichtert.

Es stellt sich die Frage, ob die Intra-Class-Varianz zur Beurteilung einer Clusterung überhaupt nötig ist und die AvgMinDist dazu nicht ausreicht. Die folgenden beiden Skizzen verdeutlichen, warum die alleinige Beurteilung einer Clusterung durch die AvgMinDist nicht ausreicht.

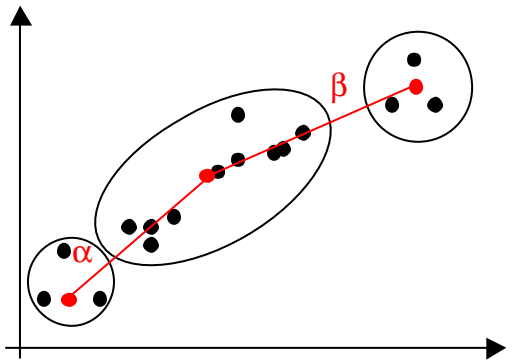


Abb. 5.2.a: Clusterung der schwarzen Punkte. Die Klassen werden durch Ellipsen dargestellt. Die farbigen Punkte stellen die Klassenrepräsentanten dar. Die Länge der farbigen Strecken sind die Ausgangswerte der AvgMinDist. Das Kriterium dieser Clusterung ist eine große AvgMinDist.

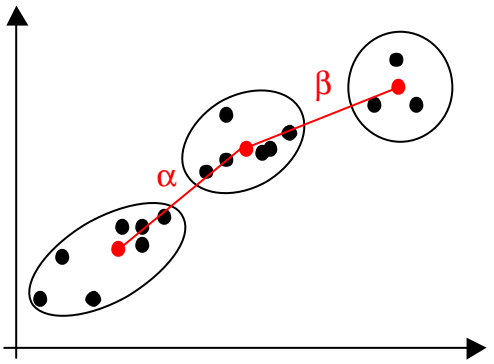


Abb. 5.2.b: Clusterung der schwarzen Punkte. Die Klassen werden durch Ellipsen dargestellt. Die farbigen Punkte stellen die Klassenrepräsentanten dar. Die Länge der farbigen Strecken sind die Ausgangswerte der AvgMinDist. Das Kriterium dieser Clusterung ist ein kleines k_{ssw}.

Die AvgMinDist der Clusterung in Abb. 5.2.a ist um 19,8 % größer als die der Clusterung in Abb. 5.2.b. Dennoch ist die Clusterung in Abb. 5.2.b besser, weil die Intra-Class-Varianz dort wesentlich kleiner sein dürfte als bei der Clusterung in Abb. 5.2.a. Die große Klasse aus Abb. 5.2.a ist gegenüber den anderen Klassen inhomogen, da in dieser Klasse sehr viele unterschiedliche Punkte zusammengefasst werden. Damit ist eine Interpretation des Klassenrepräsentanten nicht möglich. Hierbei wird deutlich, dass eine kleine Intra-Class-Varianz Grundvoraussetzung für eine brauchbare Clusterung ist und die alleinige Beurteilung nach der AvgMinDist nicht immer ausreichend ist.

Bei der Entscheidung zwischen zwei oder mehreren Clusterungen gilt somit:

1. Der zeitliche Aspekt bleibt weitgehend unberücksichtigt, aber es entstehen folgende Rangfolgen:
   - Maximum Linkage, ist am schnellsten,
   - die Winner-takes-all-Netze sind nur wenig langsamer,
   - die schnellen SOMs brauchen deutlich mehr Zeit zum Anlernen,
   - die Standard-SOMs brauchen sehr lange zur Clusterungserstellung.

2. Damit „unsinnige“ Clusterungen ausgeschlossen werden können, müssen sie Bedingung (5.8) erfüllen.

3. Die Entscheidung zwischen den Clusterungen, die (5.8) genügen, wird auf Grund von Kriterium (5.9) getroffen. Die Clusterung mit dem größten $AvgMinDist$ wird für die folgenden Untersuchungen als die beste festgelegt.

6 Methoden zur Ergebnisdarstellung in PicAna

Neben der eigentlichen Berechnung und Erstellung einer Clusterung, ist die Ergebnisdarstellung und -präsentation von Bedeutung. Nur bei einer optimalen Aufbereitung der Clusterungsergebnisse ist anschließend eine gute Interpretation der Ergebnisse möglich. Im Fall der pixelbasierten Bildclusterung, ergeben sich eine ganze Reihe von Darstellungsmöglichkeiten. Diese liefern einen umfangreichen Einblick in die Datenlage und Clusterung. Damit bietet sich eine gute Ausgangsbasis für eine umfangreiche Interpretation. Im Folgenden werden die hier verwendeten Darstellungsmöglichkeiten erläutert.

1. Die Tabellenform, mit
   a) Informationen über das Cluster und dessen Kenngrößen,
   b) zusätzlicher Auflistung aller dem Cluster zugeordnete Werte.

2. Das „geclusterte“ RGB-Bild.

3. 3-D-Darstellung des Merkmals- bzw. Parameterraumes, mit den Ausgangswerten, Zentroiden und Klassenrepräsentanten.

4. Die SOM-Darstellung der zweidimensionalen Kohonenkarte.

6.1 Ergebnispräsentation in Tabellenform

Grundlegend erfolgt die Darstellung der Ergebnisse in Tabellenform. Abb. 6.1 zeigt eine derartige Tabelle, wie sie von dem Programm PicAna erzeugt wird.

ClusterTable

Verfahren: Maximum Linkage, 20 Klassen
Table: E:\testbild_ml0_20_2.htm
Picture: E:\testbild_ml0_20_2.bmp
PicInfo: 768 x 512 = 393216
Colours: 8730
Clusters: 20
SST: 797.619
SSB: 663.765
SSW: 133.853
AvgMinDist: 631.142
Time: 27460 (27460) ms

Cluster-Nr.	Start-Colour	Avg-Colour	Min_Cl	Min_Di	Cluster-Size
1	(29, 23, 10)	(43, 29, 17)	14	1319.3134	16
2	(158,122, 98)	(158,122, 97)	19	906.3772	2272
3	(247,253,255)	(238,249,251)	20	553.2236	19
4	(221,188,169)	(214,190,173)	17	684.9349	4
5	(74, 88, 65)	(83, 81, 57)	18	267.7958	744
6	(180,234,251)	(192,234,247)	20	719.4640	5
7	(160,164,170)	(166,162,154)	13	1668.1227	12
8	(106,117,126)	(122,122,117)	13	882.7269	6
9	(85, 47, 26)	(74, 55, 41)	14	316.2768	208

Abb. 6.1: Ausschnitt aus der Ergebnistabelle der Clusterung und Klassifizierung, wie sie von der Programmumsetzung PicAna erzeugt wird. In diesem Ausschnitt fehlen die Detailinformationen der Cluster 10 bis 20.

Zu der in Abb. 6.1 dargestellten Clustertabelle gehören grundsätzliche Informationen über die Clusterung und die zugehörigen Kenngrößen:

- **Table** gibt Namen und Pfad der Datei an, die die Clustertabelle beinhaltet.
- **Picture** gibt Namen und Pfad der Bilddatei an, die das geclusterte Bild enthält.
- **PicInfo** gibt die Pixelanzahl in Breite x Höhe = Gesamtanzahl an Pixeln an.
- **Colours** gibt an, wie viele verschiedene Farben im Originalbild enthalten sind.
- **Clusters** gibt an, wie viele Klassen für die Clusterung vorgegeben worden sind.

- Die drei Einträge **SST, SSB** und **SSW** geben die zur Beurteilung der Clusterung wichtigen varianzbasierten Größen aus (5.3) bis (5.5) an.
- **AvgMinDist** enthält den Wert für das Clusterungskriterium (5.9).
- **Time** gibt die für die Berechnung benötigte Zeit in Millisekunden an. Diese Größe ist nur zwischen Clusterungen vergleichbar, die auf dem selben Rechner durchgeführt wurden, ohne dass der Rechner dabei gleichzeitig durch andere Anwendungen belastet wird.

Danach folgt die eigentliche Clustertabelle. Pro Cluster gibt es eine Zeile an Information. Die acht Spalten der Tabelle enthalten folgende Informationen:

1. Die **Cluster-Nr.** gibt die fortlaufende Nummer der Klasse.
   Beim Maximum-Linkage-Algorithmus ist dies die Nr. der Reihenfolge, in der die Zentroide gefunden werden. Zwischen den ersten beiden Zentroiden gibt es keine Reihenfolge.
   Bei den neuronalen Netzen ist dies die Nummer des Neurons, zu dem die Werte der Zeile gehören. Anhand dieser Nummer wird bei den SOMs die Nachbarschaftstruktur berechnet.
2. **Start-Colour** enthält den bei der Clusterung berechneten RGB-Wert des entsprechenden Zentroids. Dieser Zentroid wird zur Klassifikation der übrigen Farbwerte verwendet.
3. Das **Farbfeld** enthält die Farbe, die dem unter Start-Colour angegebenen Wert entspricht.
4. **Avg.-Colour** beinhaltet den RGB-Wert aus der Mittelung über alle Farbwerte der entsprechenden Klasse. Er dient als Klassenrepräsentant.
5. Das zweite **Farbfeld** enthält die Farbe, die dem unter Avg.-Colour angegebenen Wert entspricht.
6. **Min_Cl** steht für Minimum-Cluster. Der Wert repräsentiert die Nummer des Clusters, das den geringsten euklidischen Abstand zum aktuellen Cluster hat. Ausschlaggebend dafür sind die unter Avg.-Colour angegebenen Klassenrepräsentanten.
7. **Min_Di** enthält die Distanz zwischen dem Klassenrepräsentanten des aktuellen Clusters und dem nächstgelegenen Klassenrepräsentanten eines anderen Clusters.
8. Die **Cluster-Size** gibt die Anzahl unterschiedlicher Farbwerte an, die dieser Klasse zugeordnet sind. Auch wenn der Farbwert im Originalbild bei mehr als einem Pixel auftritt, wird er hier nur genau einmal gezählt.

Damit enthält diese Tabelle fast alle Ergebnisse, die durch die Clusterung und Klassifikation berechnet werden. Es fehlt nur die Information, welche RGB-Farbwerte in die entsprechende Klasse einsortiert sind. Daher wird die Tabelle aus Abb. 6.1 um die Spalten **Colours** und ein **Farbfeld** erweitert. In der Spalte Colours werden die

RGB-Werte eingetragen, die zu einer Klasse gehören. Das anschließende Farbfeld enthält die Farbe, die dem unter **Colours** angegebenen Wert entspricht.

Cluster-Nr.	Start-Colour	Avg-Colour	Min_Cl	Min_Di	Cluster-Size	Colours
16	(225,222,202)	(216,217,204)	12	439.0833	5	(225,222,202) / 1
						(208,211,192) / 1
						(214,223,228) / 1
						(218,213,199) / 1
						(219,217,200) / 1
17	(166,166,174)	(166,166,177)	4	604.9349	14	(166,166,174) / 1
						(170,182,182) / 1
						(179,179,159) / 1
						(180,173,158) / 1
						(181,186,184) / 1
						(185,174,156) / 1
						(188,191,192) / 1
						(189,191,176) / 1
						(189,195,190) / 1
						(194,180,166) / 1
						(195,188,185) / 1
						(196,197,169) / 1
						(202,202,186) / 1
						(202,206,187) / 1
18	(100, 70, 48)	(98, 75, 58)	5	267.7958	1424	(100, 70, 48) / 1
						(71, 64, 47) / 2
						(71, 68, 42) / 1

Abb. 6.2: Ausschnitt aus dem Tabellenteil der erweiterten Tabellen-Ergebnisdarstellung der Clusterung, wie sie von der Programmumsetzung PicAna erzeugt wird.

6.2 Ergebnisdarstellung in Bildform

Die zweite Ergebnisdarstellung ist die Bildform. Dabei wird ein Originalbild mit den durch die Clusterung berechneten Farben der Klassenrepräsentanten dargestellt. Abb. 6.3 zeigt eine Luftaufnahme aus Chikal, Niger. Die Luftaufnahme entstand während eines Forschungsprojekts über Landwirtschaft in Westafrika (Buekert, Mahler und Marschner, 1996) im Rahmen des SFB 308 „Adapted Farming in West Africa“. In dieser Studie wurden Techniken zur Erstellung bodennaher Luftaufnahmen entwickelt (Gérad, Buerkert et al (1997) sowie Gérad und Buerkert (1999)).

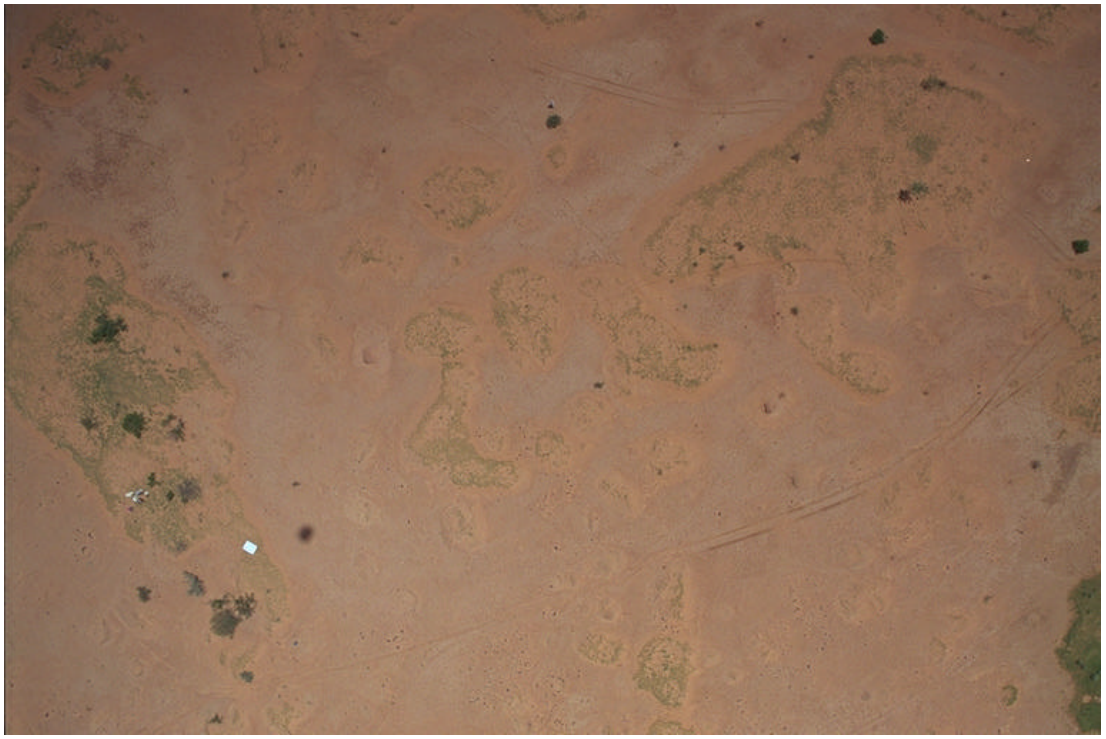


Abb. 6.3: Bodennahe Luftaufnahme aus Chikal, Niger. Erstellt im Rahmen des SFB 308 „Adapted Farming in West Africa“. Das Luftbild enthält 8730 verschiedene Farbwerte bei 393216 Pixeln.

Bei der Clusterung mit dem Maximum-Linkage-Algorithmus entsteht Abb. 6.4:

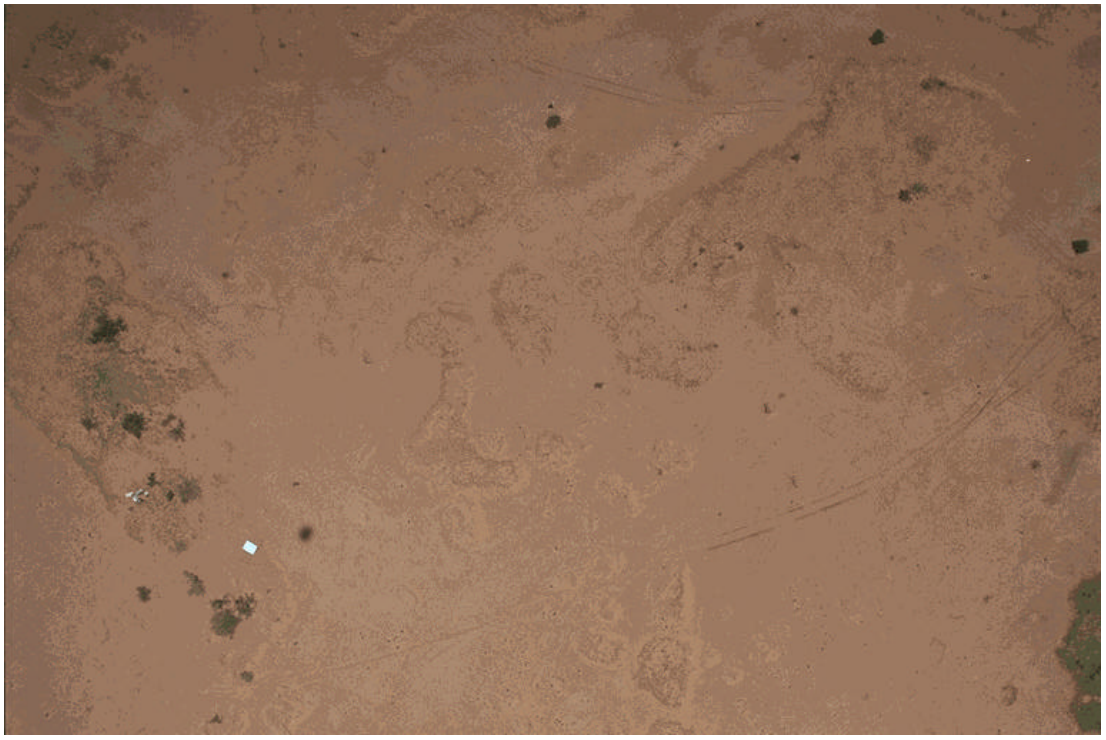


Abb. 6.4: Ergebnisses der Clusterung von Abb. 6.3 mit Maximum Linkage bei 25 Klassen (verschiedenen Farbwerten).

Durch die Clusterung wird das Originalbild (Abb. 6.3) von 8730 verschiedene Farben auf 25 reduziert. Das in Abb. 6.4 dargestellte Ergebnisbild der Clusterung zeigt, wie sich die Luftaufnahme durch die Clusterung verändert. Durch den Vergleich zwischen Originalaufnahme und geclustertem Bild, lassen sich Erkenntnisse über die Clusterung gewinnen. Zunächst erhält man einen genereller Eindruck darüber, was bei der Clusterung von Abb. 6.3 geschieht. Im Weiteren ist optisch zu überprüfen, ob wichtige Elemente auch korrekt identifiziert worden sind. Korrekt identifiziert bedeutet hier:

1. Offensichtlich unterschiedliche Elemente müssen auch in verschiedenen Klassen eingeteilt worden seien. Im vorgestellten Beispiel aus Abb. 6.4 wurden die grünen, weißen und braunen Bildelemente klar voneinander getrennt. Des Weiteren sind diese drei generellen Gruppen jeweils noch in eine Reihe weiterer Klassen eingeteilt worden.
2. Die seltenen Elemente sollen in eigenen, separaten Klassen identifiziert werden und sich gut von den anderen Klassen abgrenzen. Im Beispiel sind die seltenen weißen Objekte auch im geclusterten Bild gut erkennbar und sind nicht mit anderen Klassen oder Farbwerten vermengt worden.
3. Wichtige Strukturen des Ausgangsbildes müssen auch bei der Clusterung erkannt werden. Im Beispiel sind das z. B. die Fahrwege, die von der Mitte des oberen Bildrandes zur Mitte des rechten Bildrandes, bzw. von der Ecke links unten zur Mitte des rechten Bildrandes, führen. Der Vergleich von Abb. 6.3 und Abb. 6.4 zeigt, dass das bei der hier vorliegenden Clusterung gut gelungen ist.

Neben dem generellen Eindruck lassen sich durch diese Darstellung auch inhaltliche Erkenntnisse aus der Klassenlage gewinnen. Im hier vorliegenden Beispiel ist das besonders deutlich. Um diesen Sachverhalt zu demonstrieren, wird Abb. 6.4 in Abb. 6.5 wiederholt und mit Hilfslinien versehen, die das geclusterte Bild in Flächen einteilen. Die einzelnen Flächen werden mit Buchstaben von A bis F und X gekennzeichnet.

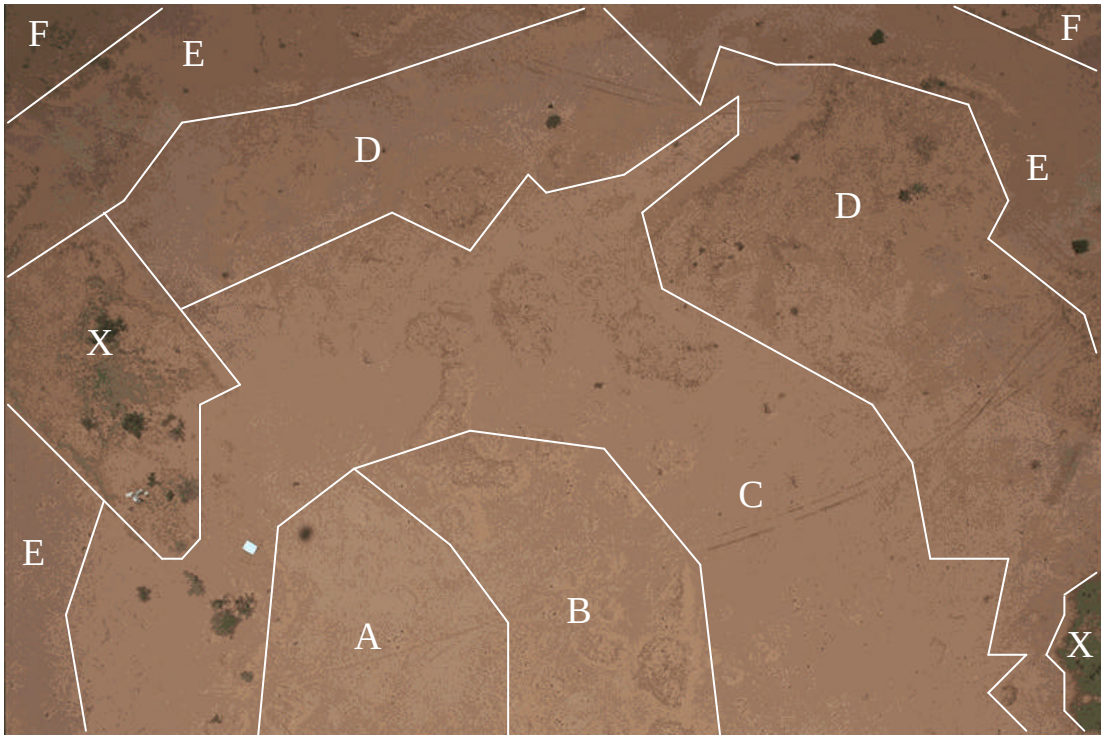


Abb. 6.5: Geclustertes Bild wie in Abb. 6.4, ergänzt um Hilfslinien, die das Bild in Flächen einteilen. Die Flächen sind mit den Buchstaben A – F und X gekennzeichnet.

Die Interpretation der Flächen aus Abb. 6.5 ergibt sich wie folgt: Es ist eine systematische Farbänderung der Flächen von A nach F erkennbar. Während A die hellste Einfärbung aufweist, liegt bei F die dunkelste vor. Bei der Interpretation dieser Flächeneinteilung werden die kleinen grünen Enklaven der Flächen A, C, D und E nicht berücksichtigt. Die mit X gekennzeichneten Flächen sind sehr stark bewachsen oder weisen eine sehr unterschiedliche Farbgebung auf. Damit passen sie nicht in die klar erkennbare Rangfolge der Flächen A bis F. Die genaue Interpretation dieses Schemas muss ein Fachwissenschaftler für Erosionsforschung vornehmen, aber es liegt nahe, dass eine Erosionszunahme von F nach A vorliegt.

6.3 Grafische Darstellung der Farbwerte und Zentroide

Weil Merkmals- und Parameterraum drei Dimensionen aufweisen, ist eine grafische 3-D-Darstellung möglich. Dadurch ergeben sich Einblicke in die Daten des Originalbildes sowie der Clusterung und Klassifizierung.

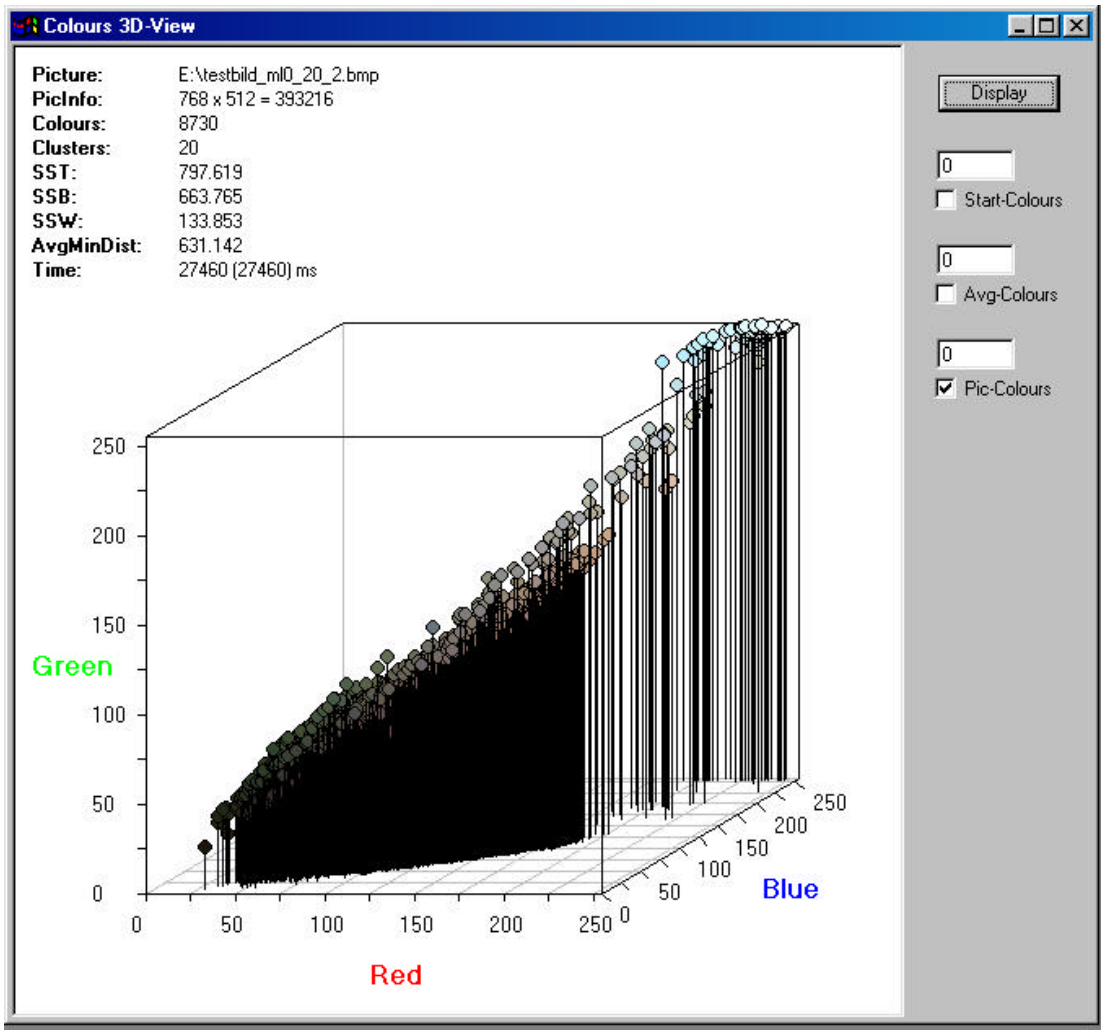


Abb. 6.6: Darstellung aller unterschiedlichen Farbwerte des Originalbildes aus Abb. 6.3

Abb. 6.6 zeigt die Lage der verschiedenen Farbwerte des Originalbildes (Abb. 6.3) im Merkmalsraum. Auf Grund der Beschränktheit des Merkmalsraumes in allen drei Dimension, ist eine 3-D-Darstellung des vollständigen Merkmalsraumes möglich. Dabei werden die RGB-Einträge in ein kartesisches Koordinatensystem übertragen. Der jeweilige Punkt wird entsprechend der Farbe des RGB-Wertes eingefärbt. In Abb. 6.6 liegen die einzelnen Werte sehr dicht gedrängt. Die dadurch auftretende Unübersichtlichkeit lässt sich im interaktiven Programmdialog von PicAna auflösen.

Dort ist es möglich, nur Werte ab einer gewissen Häufigkeit anzuzeigen. Dazu ist im Feld über dem Checkbutton „Pic-Colours“ ein Wert einzutragen. Durch einen Klick auf „Display“ werden anschließend nur Werte mit einer Häufigkeit größer/gleich dem angegebenen Wert angezeigt. Durch stufenweises Höhersetzen dieses Wertes, lässt sich ein Eindruck über die Datenlage gewinnen.
Diese Ergebnisdarstellung ist auch für die Clusterung, bzw. das berechnete Bild, möglich. Wird in PicAna der Checkbutton „Start-Colours“ statt „Pic-Colours“ gesetzt, wird die Lage der Zentroide im Parameterraum angezeigt.

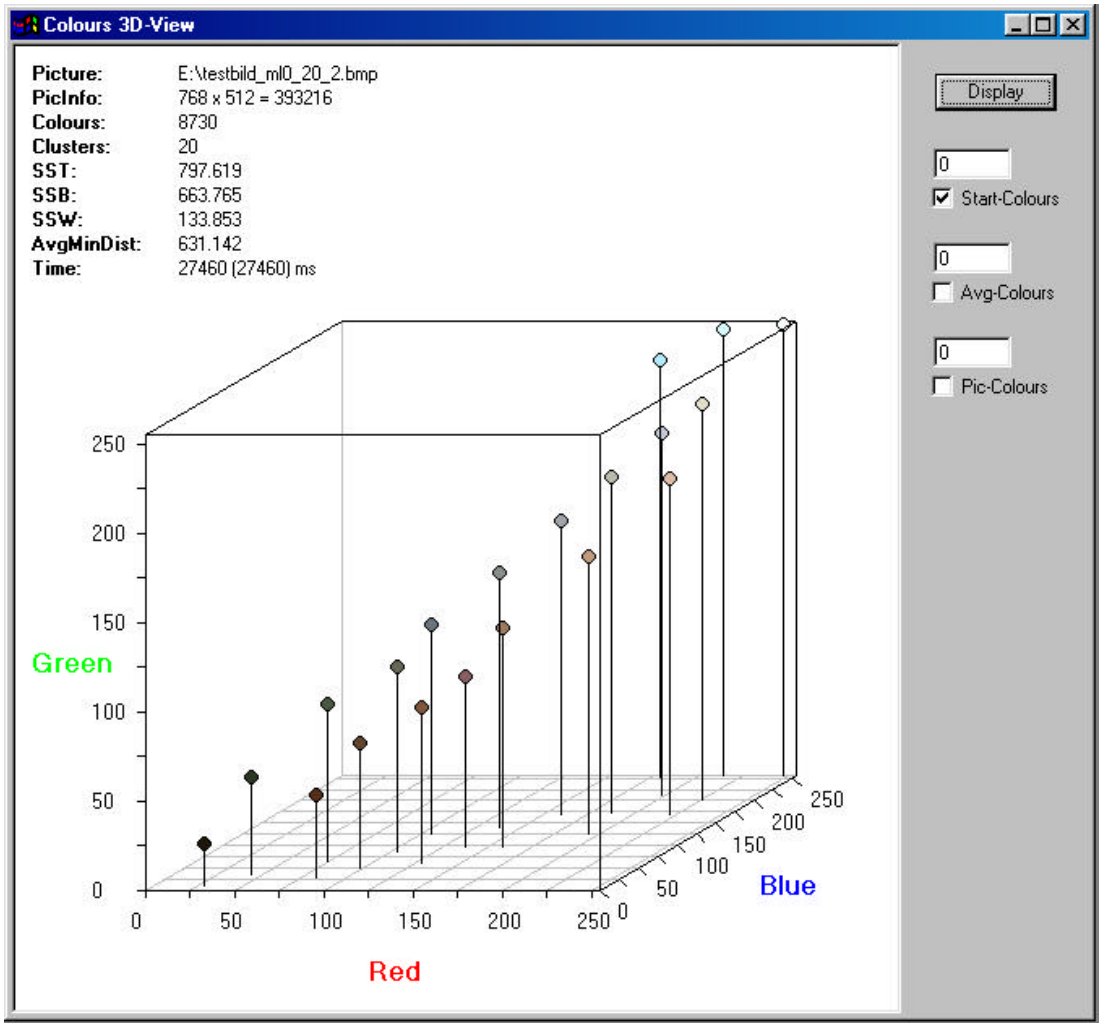


Abb. 6.7: Darstellung der Lage der Zentroide im Merkmalsraum, die auf Grund der Clusterung von Abb. 6.3 entstehen und als Ausgangsbasis für die Klassifizierung der Farbwerte des Originalbildes aus Abb. 6.3 dienen.

Auf Grund der reduzierten Anzahl an Klassen, ist diese Darstellung sehr übersichtlich. Analog zur Darstellung der Ausgangsfarbwerte des Bildes besteht weiterhin die Möglichkeit, nur Werte größer/gleich einer vorgegebenen Häufigkeit anzuzeigen. Dazu muss die gewünschte Häufigkeit im Dialogfeld über dem Checkbutton „Start-Colours“ angegeben werden.

Es ist ferner möglich, die Lage der Klassenrepräsentanten im Merkmals- bzw. Parameterraum anzuzeigen. Dazu muss der Checkbutton mit der Beschriftung „Avg-Colours“ gesetzt sein. Abb. 6.8 zeigt diese Darstellung. Auch hierzu existieren analoge Einstellungsmöglichkeiten, wie bei der Darstellungen der Ausgangswerte bzw. Zentroide.

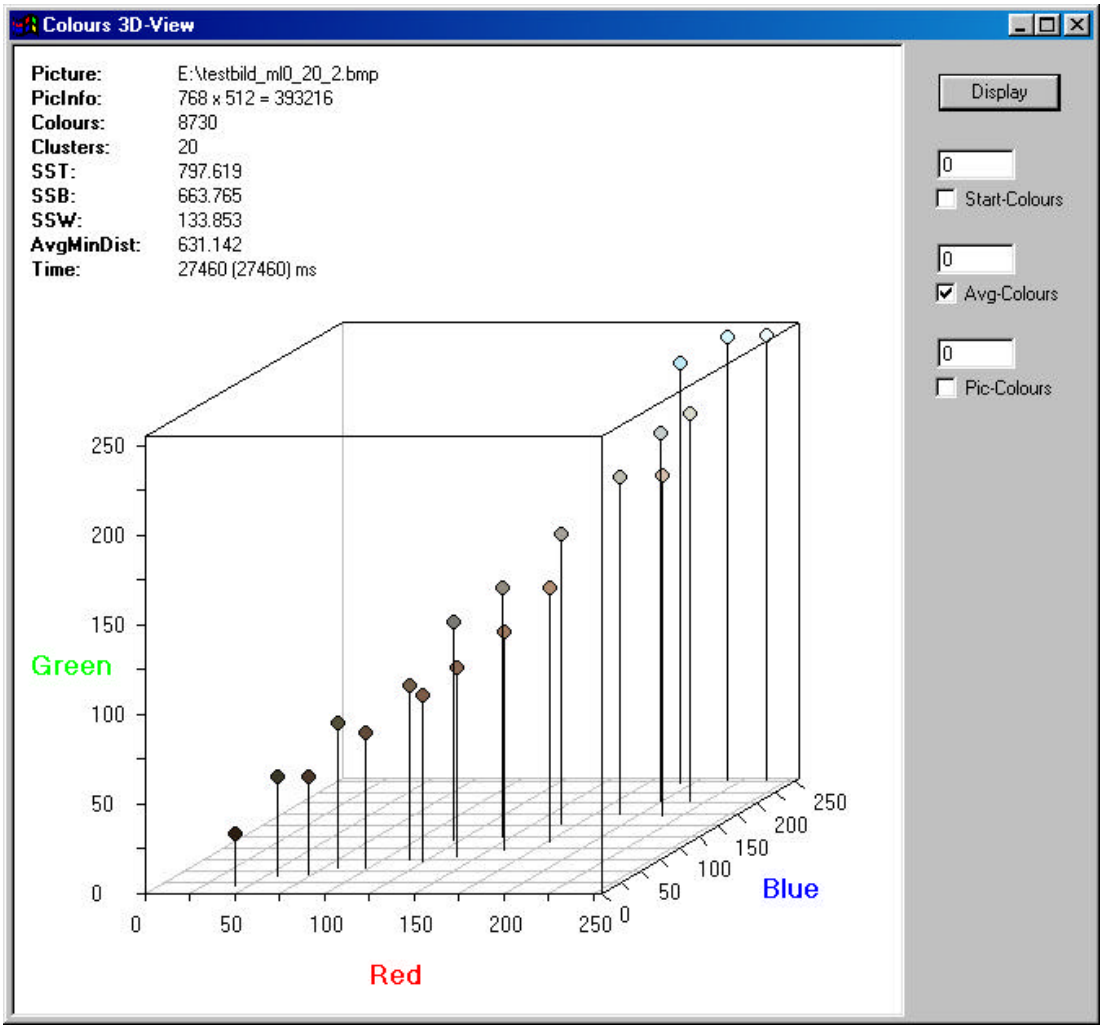


Abb. 6.8: Darstellung der Lage der Klassenrepräsentanten im Merkmalsraum, die bei der Mittelung der Werte in den Klassen nach der Klassifizierung von Abb. 6.3 auf Grund der mit dem Maximum-Linkage-Algorithmus ermittelten Ausgangswerte für die Zentroide entstehen.

Bei dieser Art der Ergebnispräsentation ist der Vergleich zwischen den Darstellungen der Zentroide (Abb. 6.7) und der der Klassenrepräsentanten (Abb. 6.8) interessant. Durch den Vergleich wird erkennbar, wie weit die bei der Clusterung bestimmten Zentroide von den Klassenrepräsentanten abweichen. Das Beispiel zeigt, dass bei großen Werten in allen Komponenten nur geringfügige Verschiebungen stattfinden. Die zu den Zentroiden klassifizierten Werte liegen also gleichmäßig um die in der Clusterung ermittelten Zentroide verteilt. Die Zentroide repräsentieren die zuge-

hörigen Klassen also schon bevor die eigentliche Klasse durch die Klassifizierung der übrigen Werte gebildet wird. Je geringer die Verschiebung zwischen Zentroid und Klassenrepräsentant ist, desto besser ist die Auswahl der Zentroide und die Repräsentationsfähigkeit des Zentroids vor der eigentlichen Klassenbildung.
Für kleine Werte in allen Komponenten finden dagegen deutlich sichtbare Verschiebungen statt. Dies ist ein Hinweis auf eine deutliche Veränderungen durch die Klassifizierung gegenüber der Zentroidauswahl bei der Clusterung. Die Auswahl der Zentroide ist also gegenüber dem Endergebnis nicht optimal gewesen. Je besser die Auswahl der Zentroide ist, desto weniger Verschiebungen ergeben sich zu den Klassenrepräsentanten nach der Klassifikation.
Abschließend werden die Kohonenkarten betrachtet, die nur im Fall der Clusterung mittels selbstorganisierender Karten zur Verfügung stehen.

6.4 Ergebnisdarstellung anhand der Kohonenkarte

Bei der Clusterung mittels selbstorganisierender Karten existiert eine Nachbarschaftsstruktur zwischen den Neuronen. Damit besteht diese Beziehung auch zwischen den ermittelten Klassen. Die Nachbarschaftsstruktur lässt sich grafisch darstellen. Dazu stehen in PicAna drei Modi zur Darstellung der Kohonenkarte zur Verfügung.

1. Beim **Farbmodus** sind die Felder der Kohonenkarte in der Farbe des RGB-Wertes des Gewichtsvektors eingefärbt.

2. Beim **Distanzmodus** ist der Abstand zwischen dem Gewicht des aktuellen Gewinner-Neurons zu den Gewichten der übrigen Neuronen veranschaulicht. Je dunkler der Grauwert ist, desto größer ist der Abstand zwischen den Gewichten.

3. Beim **Clustergrößen-Modus** gibt die Graustufe der Kohonenkarten-Felder an, wie stark die entsprechende Klasse besetzt ist. Je dunkler der Grauwert, desto weniger Werte enthält die Klasse.

6.4.1 Der Farbmodus

Der Farbmodus zeigt die durch die Gewichte der Neuronen repräsentierten RGB-Farbwerte an. Die Kohonenkarte wird durch nebeneinander liegende Flächen dargestellt, deren Zentrum das entsprechende Neuron darstellt. Für die folgenden grafischen Darstellungen werden die gesamten Flächen um das jeweilige Neuron eingefärbt. Die gestrichelten Linien in Abb. 6.9 skizzieren im Falle der quadratischen Nachbarschaftsstruktur diese Flächen um die Neuronen. Bei hexagonaler Struktur werden die in Abb. 4.7 dargestellten Hexagone um die Neuronen eingefärbt.

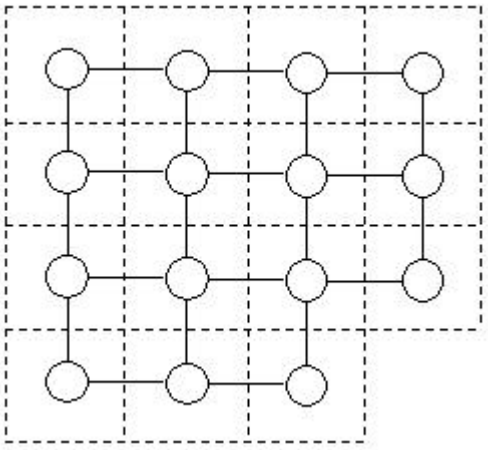


Abb. 6.9: Neuronengitter, bei dem die durch gestrichelte Linien angedeuteten Flächen mit dem RGB-Wert des Neuronengewichts eingefärbt werden. Das so entstehende Muster ergibt die Visualisierung der Kohonenkarte.

Beim Farbmodus werden die einzelnen Flächen der Kohonenkarte in dem durch den Gewichtsvektor repräsentierten RGB-Farbwert eingefärbt. Dabei handelt es sich um das „wirkliche“ Gewicht des Neurons, wie es zur Klassifizierung benutzt wird. Abb. 6.10 zeigt ein Beispiel für eine Karte, bei dem eine Clusterung des Originalbildes mit einem Standard-SOM mit quadratischer Nachbarschaftsstruktur bei 100 Klassen und 1000 Lernzyklen durchgeführt wurde.

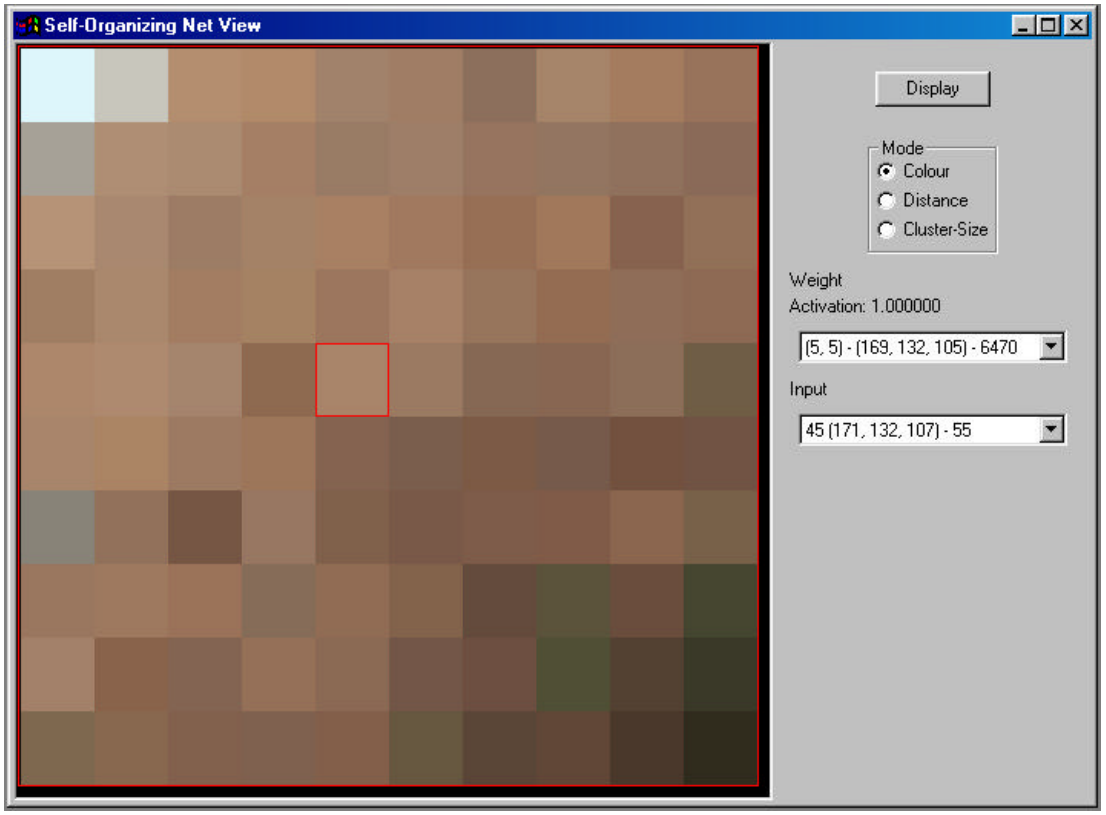


Abb. 6.10: Farbmodus der quadratischen Kohonenkarte bei einem Standard-SOM bei 100 Klassen und 1000 Lernzyklen.

In PicAna können im Anzeigefenster einzelne Neuronen aus der Combobox unter „Weight Activation“ ausgewählt werden. Durch Auswahl eines Neuron werden zusätzlich Informationen angezeigt und das Neuron wird in der Grafik umrandet dargestellt. Abb. 6.11 erläutert die Details zur oberen Combobox.

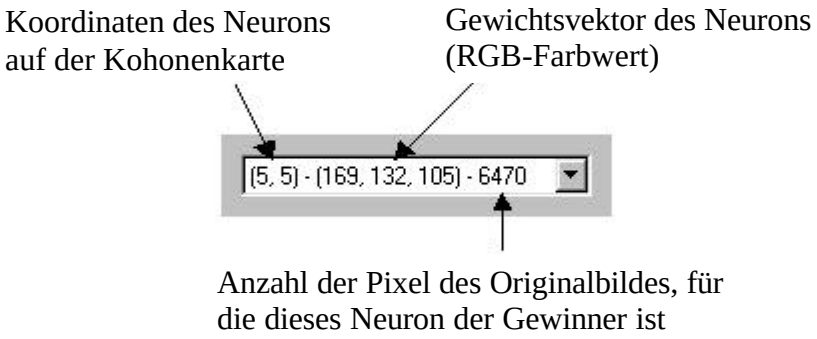


Abb. 6.11: Bedeutung der Einträge in der oberen Combobox von Abb. 6.10, mit deren Hilfe sich einzelne Neuronen direkt auswählen lassen.

Die zweite Einstellmöglichkeit erfolgt durch die Combobox „Input“. Darin lassen sich alle r verschiedenen Farbwerte des Originalbildes auswählen. Die Angaben in dieser Combobox lesen sich wie folgt:

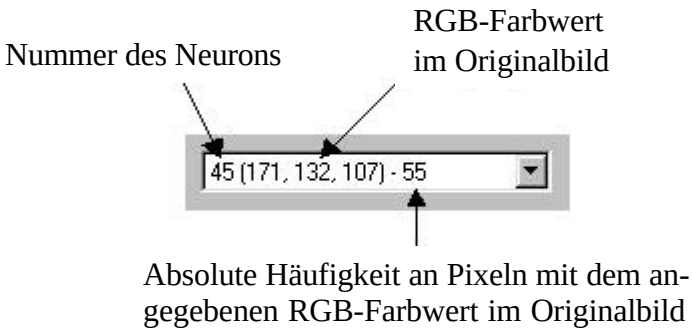


Abb. 6.12: Bedeutung der Einträge in der unteren Combobox von Abb. 6.10, mit deren Hilfe sich die verschiedenen RGB-Farbwerte des Originalbildes anwählen lassen.

Bei Auswahl eines Wertes werden automatisch in der entsprechenden Combobox die Informationen über das zugehörige Gewinner-Neuron aufgeführt. Das Neuron wird auf der Kohonenkarte umrandet dargestellt.

Der Eintrag „Weight Activation“ in Abb. 6.10 gibt die Aktivierung des Gewinner-Neurons in der Combobox gegenüber dem Neuron in der Combobox „Input“ an. Gemäß (4.21) ist diese für das Gewinner-Neuron eins und für alle anderen Neuronen null. Es lässt sich eine Mitaktivierung für die Nicht-Gewinner-Neuronen gemäß folgender Bedingung definieren:

$$f_{KOH}(N_i) = 1 - \frac{\left| P - W_i \right| - \min\limits_{j=1,\ldots,q} \left| P - W_j \right|}{\max\limits_{j=1,\ldots,q} \left| P - W_j \right| - \min\limits_{j=1,\ldots,q} \left| P - W_j \right|} \qquad (6.1)$$

Nach (6.1) gibt es für alle Neuronen N_i eine Aktivierung. Diese liegt zwischen null und eins. Die eins wird erreicht, wenn die Eingabe dem Gewinner zugeordnet wird. Gehört die Eingabe zu dem Neuron mit der größten Entfernung zum Gewinner, erhält es eine Aktivierung von null. Dieser Wert gibt an, wie nahe die jeweilige Eingabe bei dem jeweiligen Neuron N_i liegt. Je näher dieser Wert bei eins liegt, desto näher liegt der Eingabewert bei diesem Neuron. Bei der Interpretation dieses Wertes ist Vorsicht geboten. Der Wert hängt stark vom maximal auftretenden Abstand

$$\max_{j=1,\ldots,q} \left| P - W_j \right|$$

ab. Daher sind die Werte aus verschiedenen Durchläufen nicht miteinander vergleichbar.

Die Interpretation einer Kohonenkarte

Es ist Abb. 6.10 zu entnehmen, dass die „hellen“ Werte in der linken, oberen Ecke dargestellt sind. Die dunklen in der rechten, unteren Ecke. Die dazwischenliegenden Werte bilden einen Verlauf von hell nach dunkel im Bereich von links oben nach rechts unten. Ähnliche Farben liegen bis auf wenige Ausnahmen nahe beieinander. Das zeigt, dass eine gute Kohonenkarte entstanden ist.

6.4.2 Der Distanzmodus der Kohonenkarte

Der Distanzmodus veranschaulicht den Abstand zwischen dem Gewicht des aktuellen Gewinner-Neurons zu den Gewichten der übrigen Neuronen. Abb. 6.13 zeigt eine solche Karte. Jedes Neuron kann als Gewinner eingestellt werden. In Abhängigkeit dazu werden die Abstände zu den anderen Neuronen in Graustufen angezeigt.

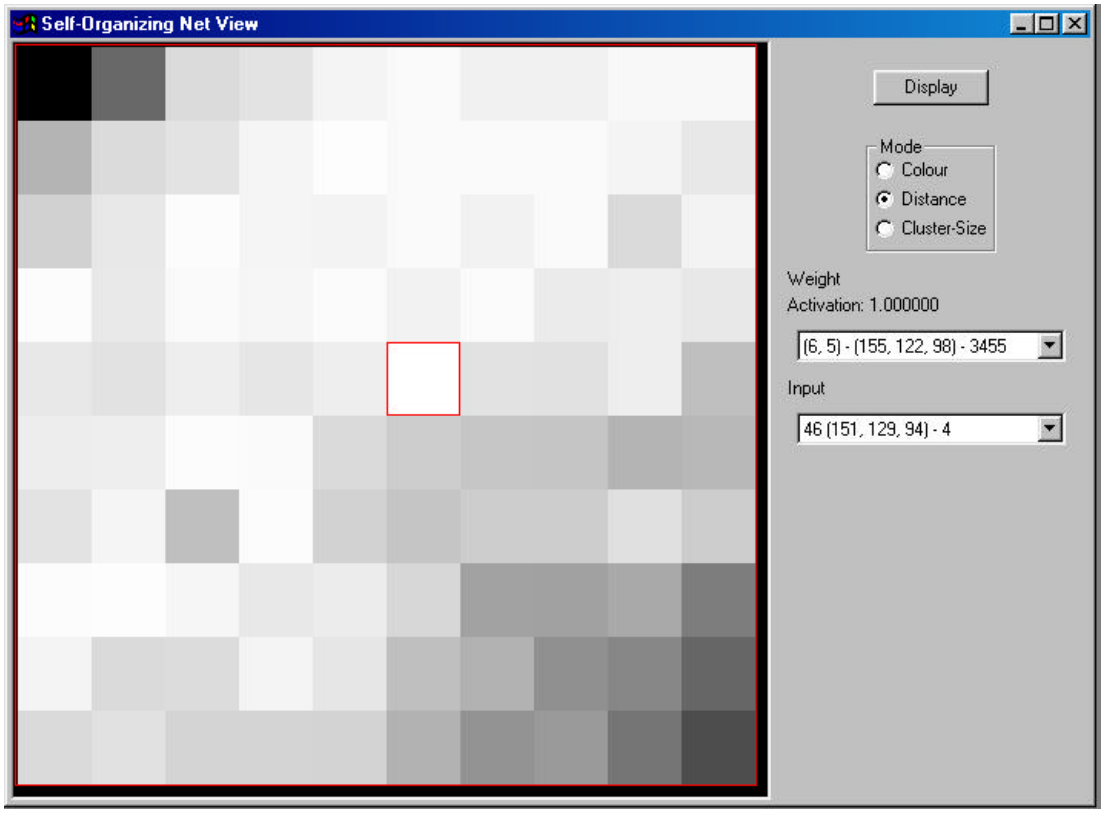


Abb. 6.13: Distanzmodus der quadratischen Kohonenkarte mit einem Standard-SOM bei 100 Klassen und 1000 Lernzyklen.

Interpretation der Kohonenkarte der Distanzen

Dem Eingabewert (Farbwert (151, 129, 94)) ist das Neuron Nr. 46 zugeordnet. Dieses ist auf der Kohonenkarte markiert. Die Graustufe der Flächen um die Neuronen gibt den Abstand zwischen dem Gewicht des Neurons und dem des Gewinners an. Je dunkler der Grauwert ist, desto größer ist der Abstand. Die vorliegende Beispielkarte zeigt, dass die Grauwerte auf der Gegendiagonalen der Karte in beide Richtungen vom Gewinner-Neuron aus schrittweise dunkler werden. Das bedeutet, dass sich die Gewichte der Neuronen mit zunehmender Entfernung deutlicher unterscheiden. Auf der Hauptdiagonalen ist dieser Effekt schwächer ausgeprägt. Das lässt darauf schließen, dass eine gut entwickelte Karte mit einer aussagekräftigen Nachbarschaftsbeziehung entstanden ist. Im Idealfall nimmt die Helligkeit der Grauwerte mit zunehmender Entfernung zum Gewinner in jede Richtung ab.

6.4.3 Der Clustergrößen-Modus der Kohonenkarte

Der Clustergrößen-Modus der Kohonenkarte liefert einen Anhaltspunkt dafür, wie stark die durch die Neuronen repräsentierten Klassen besetzt sind. Dazu wird eine Häufigkeitsverteilung der den Neuronen zugeordneten Pixel erstellt. Für jedes Neuron werden die zugeordneten Farben mit ihrer Häufigkeit gezählt. Die Anzahl der zugeordneten Werte lässt sich der oberen Combobox entnehmen. Im Beispiel aus Abb.

6.14 beträgt dieser Wert für das ausgewählte Neuron (Nr. 58) 159. Dieses Neuron repräsentiert 159 einzelne Pixel des Ausgangsbildes.

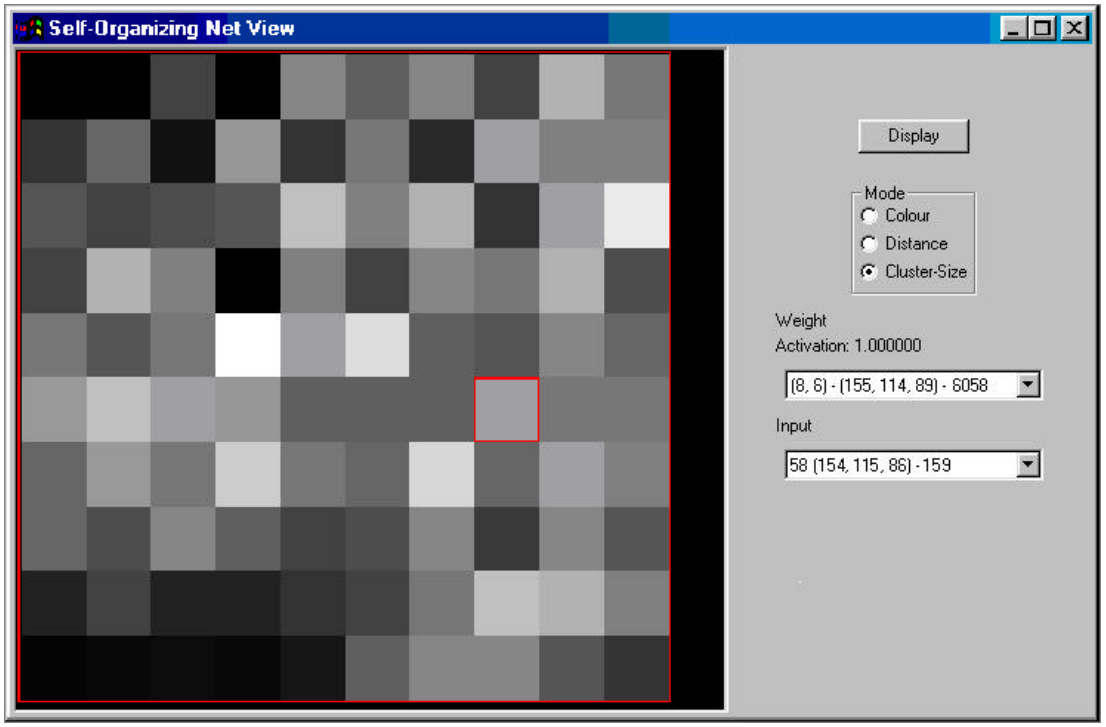


Abb. 6.14: Clustergrößen-Modus der quadratischen Kohonenkarte der Clusterung von Abb. 6.3 mit einem Standard-SOM bei 100 Klassen und 1000 Lernzyklen zum Anlernen des Netzes, wie sie von der Programmumsetzung PicAna erzeugt wird.

Der Clustergrößen-Modus ist dazu geeignet, Aussagen über die Qualität der Kohonenkarte zu treffen. Optimalerweise würde eine über die Karte gelegte Häufigkeitsverteilung einen Gipfel in der Mitte der Karte aufweisen. Das wird in dieser Form der Kartendarstellung dadurch deutlich, dass die Felder in der Mitte der Karte hell sind und zu allen Seiten hin dunkler werden. Obwohl dieser Effekt im Beispiel nicht sehr stark ausgeprägt ist, lässt sich diese Tatsache ansatzweise erkennen. Insbesondere die schwache Besetzung der Neuronen am Rand der Karte, wird sehr gut deutlich. Die wenigen hellen Felder sind in der Mitte der Karte angeordnet, wie es bei einer guten Karte der Fall ist. Der Übergang zwischen den dunklen und hellen Bereichen der Karten ist hier nicht optimal. Das muss aber nicht an einer schlecht entwickelten Karte liegen, sondern hängt vielfach mit der Struktur der Ausgangsdaten zusammen.

7 Empirischer Vergleich der Methoden

Zunächst werden alle sechs Clusterverfahren für zwei Beispielbilder exemplarisch durchgeführt. Die jeweiligen Ergebnisse werden zu Vergleichszwecken gegenübergestellt. Damit werden die Unterschiede zwischen den Verfahren aufgezeigt.
Abschließend werden die Auswirkungen der verschiedenen Parametereinstellungen anhand von empirischen Versuchsreihen dargestellt. Bei den Parametereinstellungen handelt es sich im Wesentlichen um die Vorgabe der Lernzyklenzahl bei neuronalen Netzen. Die notwendige Anzahl der Lernzyklen ist nicht von vornherein klar und nur empirisch bestimmbar.

7.1 Exemplarische Vorstellung aller Methoden

Die Clusterungsergebnisse in Bildform

Im Fall der pixelbasierten Bildclusterung, lässt sich das Ergebnis selbst wieder als Bild darstellen. Dies ist die anschaulichste Variante der Ergebnisdarstellung. Insbesondere der direkte Vergleich mit dem Ausgangsbild macht diese Form der Darstellung interessant. Dazu wird eine weitere im Rahmen des SFB 308 „Adapted Farming in West Africa“ (Buerkert, 1996) aufgenommene Luftaufnahme verwendet.

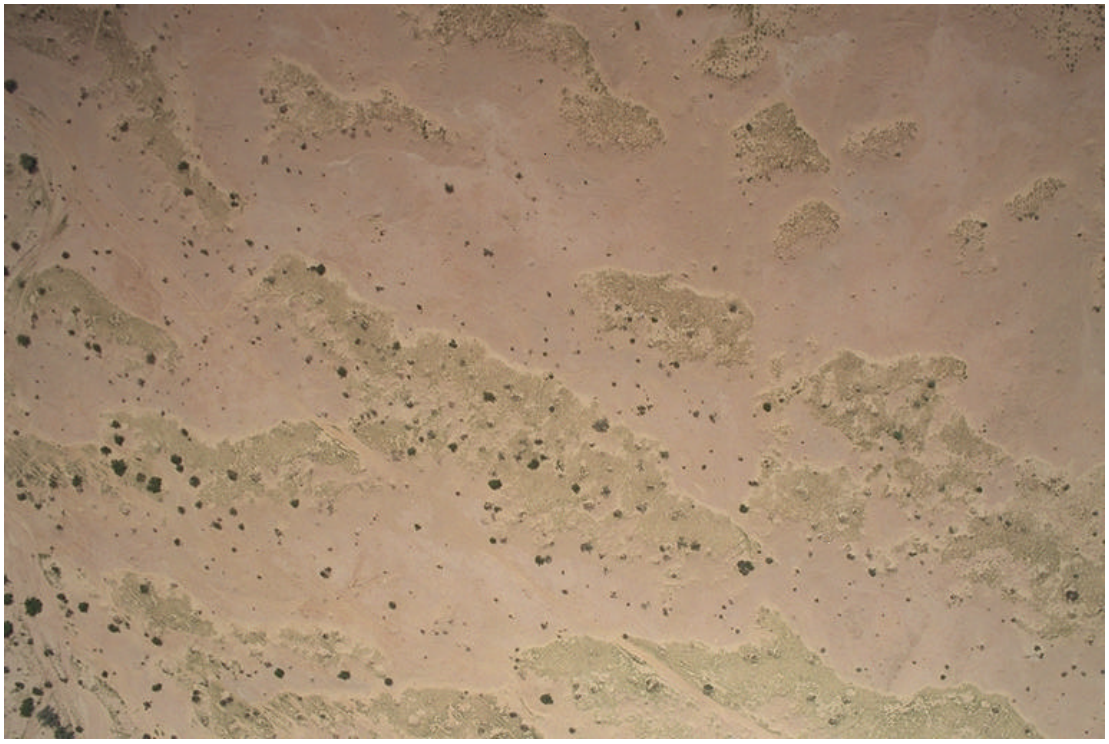


Abb. 7.1: Luftaufnahme mit 10573 verschiedenen Farbwerten bei 393216 Pixeln. Aufgenommen im Rahmen des SFB 308 „Adapted Farming in West Africa“ Region Chikal, Niger.

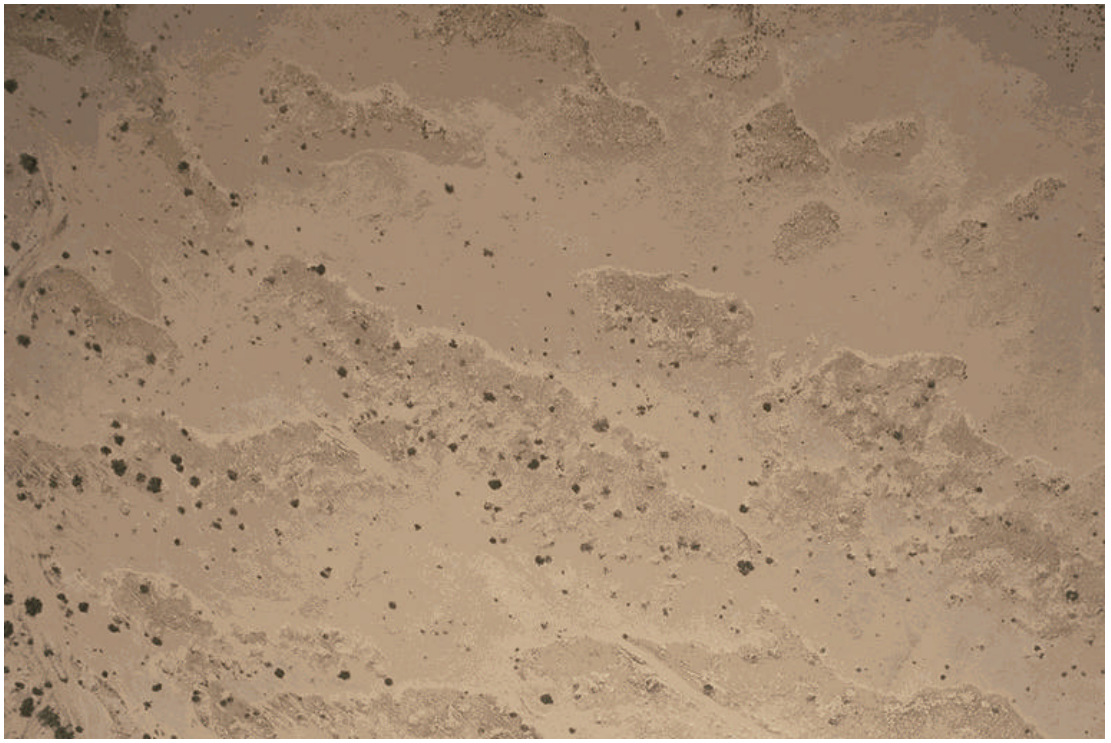


Abb. 7.2: Clusterung von Abb. 7.1 mit dem Maximum Linkage Verfahren bei 25 vorgegebenen Klassen.

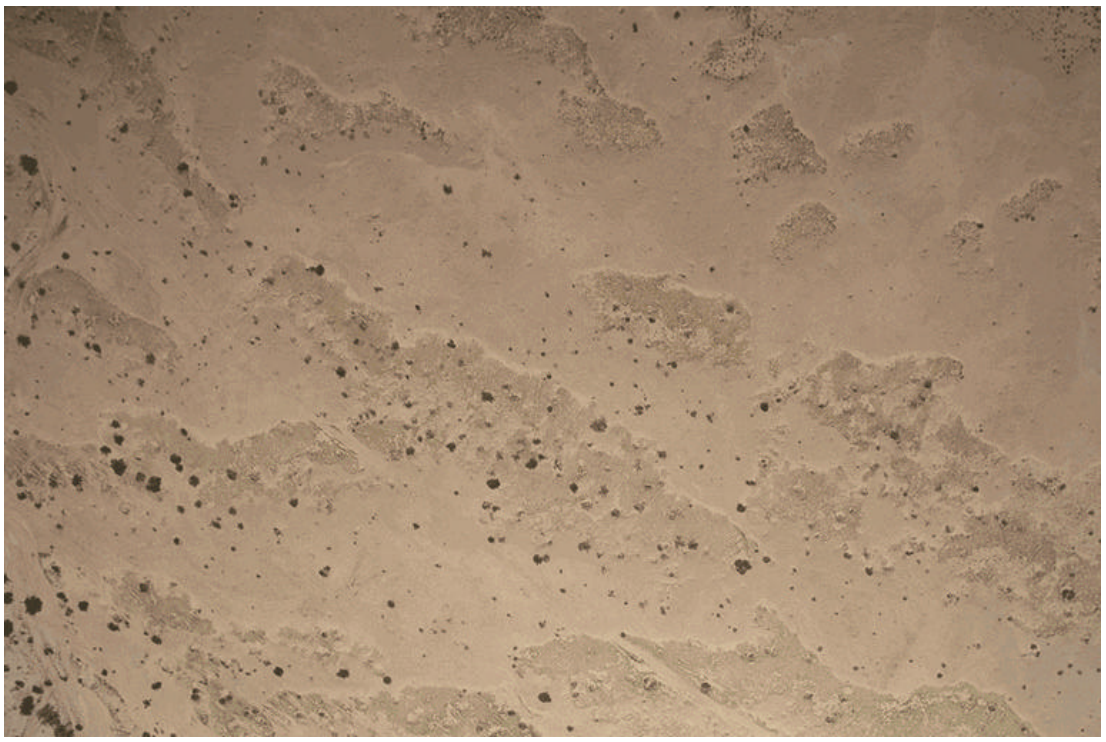


Abb. 7.3: Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 20 Lernzyklen zum Trainieren des Netzes.

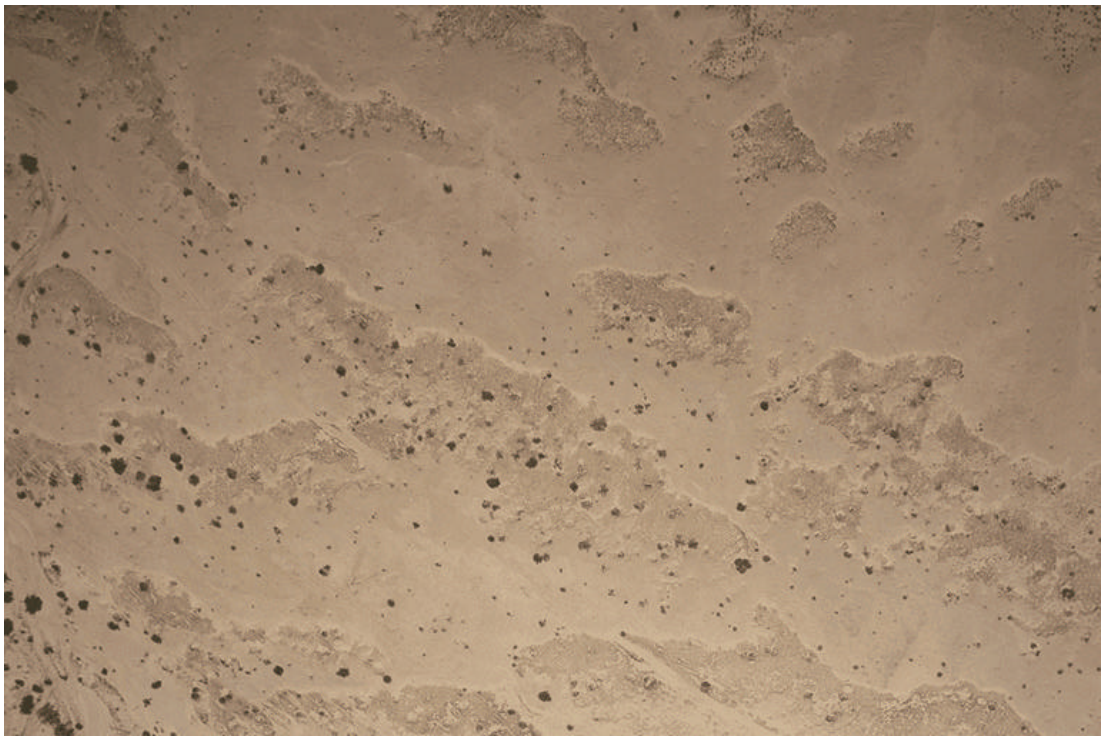


Abb. 7.4: Clusterung von Abb. 7.1 mit dem Standard-SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 2000 Lernzyklen zum Trainieren des Netzes.

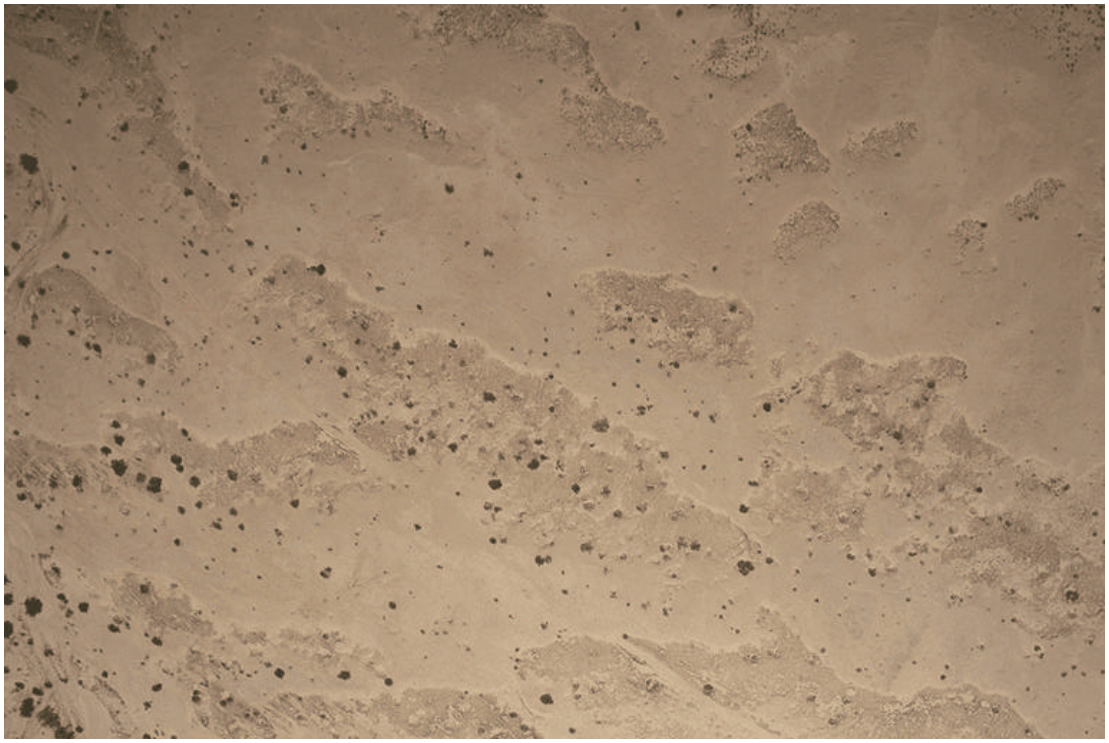


Abb. 7.5: Clusterung von Abb. 7.1 mit dem Standard-SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 4000 Lernzyklen zum Trainieren des Netzes.

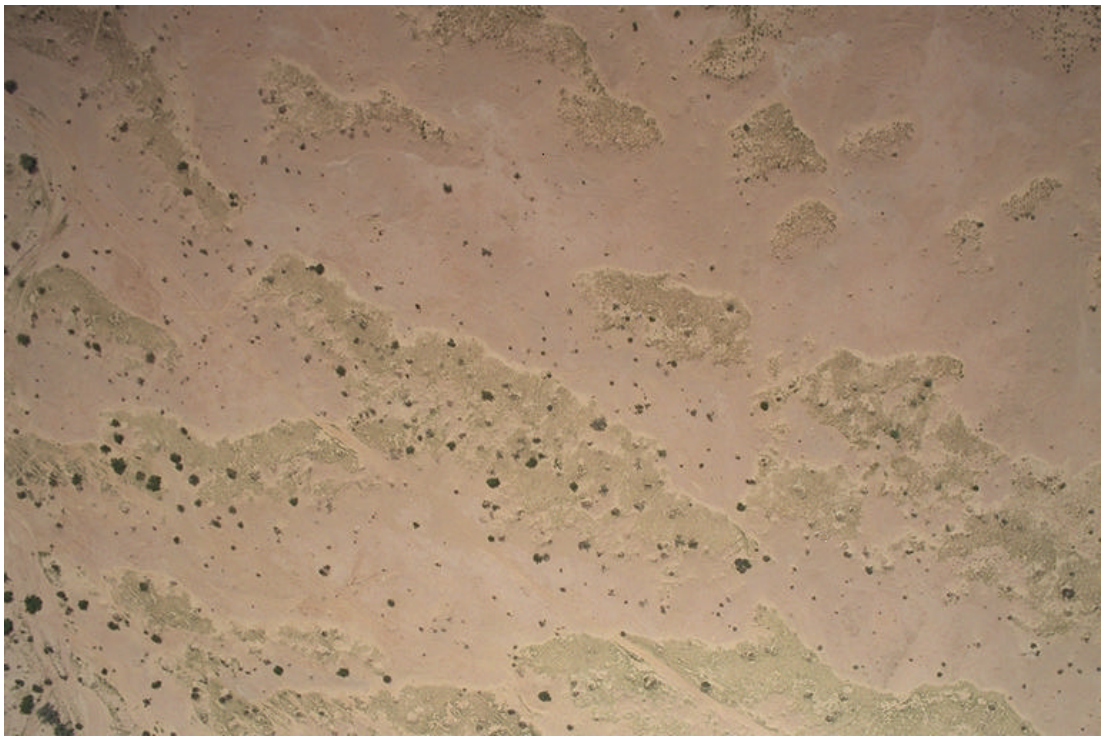


Abb. 7.6: Clusterung von Abb. 7.1 mit dem schnellen SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 20 Lernzyklen zum Trainieren des Netzes.

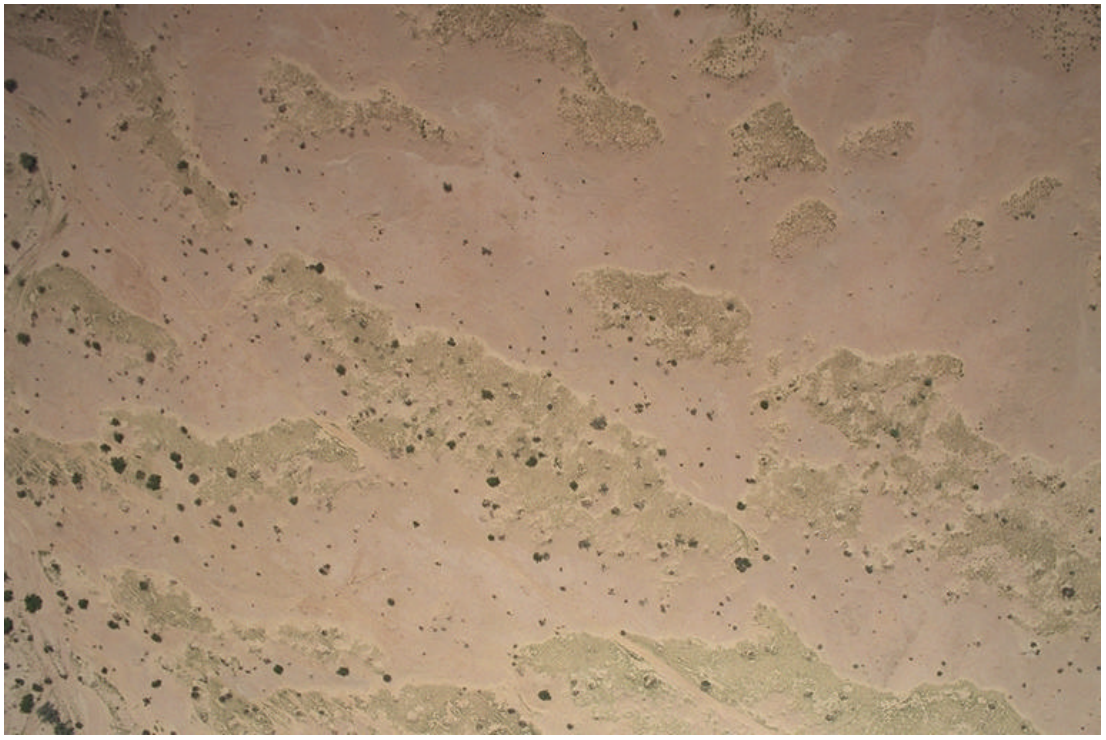


Abb. 7.7: Clusterung von Abb. 7.1 mit dem schnellen SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 40 Lernzyklen zum Trainieren des Netzes.

Die vorgestellten Clusterungsergebnisse aus Abb. 7.2 bis Abb. 7.7 zeigen alle eine große Ähnlichkeit zum Originalbild. Beim Maximum-Linkage-Algorithmus fällt auf, dass dieser zur „Flächenbildung“ neigt. Es werden zusammenhängende Flächen erzeugt, die sich gut gegeneinander abgrenzen. Diese besondere Eigenschaft war von vornherein für die Anwendung im Bereich der Erosionsüberwachung und -untersuchung gefordert. Bei den neuronalen Netzen verhält es sich ähnlich; allerdings vermitteln die so erzeugten Bilder einen optischen Eindruck, der dem des Originals noch näher kommt. Bei der Entscheidung für ein Clusterverfahren ist die spätere Anwendung entscheidend. Die Qualitätsunterschiede zwischen den mit neuronalen Netzen erzeugten Bilder sind optisch sehr gering. Während bei allen selbstorganisierenden Karten optisch kaum Unterschiede erkennbar sind, neigt die Clusterung mit dem Winner-takes-all-Netz ein klein wenig zur Flächenbildung wie beim Maximum Linkage. Die Laufzeitunterschiede zwischen den Methoden sind jedoch erheblich.

Clusterverfahren	**Lernzyklen**	**Laufzeit in ms**
Maximum Linkage	1	42.570
Winner-takes-all-Netz	20	1.462.560
schnelles SOM (quadratisch)	20	7.573.180
schnelles SOM (hexagonal)	40	11.958.050
Standard-SOM (quadratisch)	2000	101.194.970
Standard-SOM (hexagonal)	4000	205.146.326

Tab. 7.1: Laufzeit der Clusterungen in Millisekunden. Da die Laufzeit auch von der Anzahl der Lernzyklen abhängt, ist deren Anzahl hier mit aufgeführt.

Aus Tab. 7.1 geht hervor, dass die Methoden auf Grund der Berechnungsdauer in vier Gruppen einzuteilen sind: Erstens der sehr schnelle Maximum-Linkage-Algorithmus. Zweitens das Winner-takes-all-Netz. Drittens die schnellen SOMs, unabhängig von der Nachbarschaftsstruktur. Und viertens die sehr langsamen Standard-SOMs, unabhängig von der Nachbarschaftsstruktur.

Die Kenngrößen der Clusterungen

Wie in Kapitel 5 beschrieben, sind die entscheidenden Kenngrößen die Intra-Class-Varianz und der mittlere minimale Abstand zwischen den Klassenrepräsentanten. Abb. 7.2 zeigt die Kenngrößen für die oben vorgestellten Beispiele:

Methode	**Nachbar-schafts-struktur**	**Lern-zyklen**	**Intra-Class-Varianz**	**mittlerer minimaler Abstand**
Maximum Linkage	keine	-	71,5	330,9
Winner-takes-all-Netz	keine	20	34,7	173,3
Standard-SOM	quad.	2000	24,3	131,7
Standard-SOM	hex.	4000	23,8	135,8
schnelles SOM	quad.	20	28,2	137,5
schnelles SOM	hex.	40	29,3	138,9

Tab. 7.2: Kenngrößen der Clusterungen des Beispielbildes (Abb. 6.3) mit den verschiedenen Methoden bei einer vorgegeben Klassenanzahl von 25. Das Ausgangsbild hat eine Gesamtvarianz von 1287,5. Alle Zahlenwerte sind auf eine Dezimalstelle gerundet.

Betrachtung der Intra-Class-Varianz:

1. Die **Standard-SOMs** liefern die kleinste Intra-Class-Varianz und damit, im Sinne dieses Kriteriums, die beste Clusterung.
2. Die **schnellen SOMs** sind gegenüber den Standard-SOMs nur geringfügig schlechter.
3. Das **Winner-takes-all-Netz** weist eine größere Varianz auf, d. h. es ist im Sinne dieses Kriteriums weniger optimal.
4. Das **Maximum-Linkage-Verfahren**, weist eine deutlich größere Intra-Class-Varianz auf. Dies ist auf Grund der Art, wie die Klassen gebildet werden, so zu erwarten.

Dennoch ist festzuhalten, dass selbst der größte Wert für die Intra-Class-Varianz von 71,5 beim Maximum Linkage gegenüber der Gesamtvarianz des Ausgangsbildes von 1287,5 klein genug ist, um die Clusterungen als „gut“ einzustufen.

Vergleich des mittleren minimalen Abstands zwischen den Klassenrepräsentanten:

Nach den Ausführung von Abschnitt 5.3 ist die Clusterung am besten, die den größten mittleren minimalen Abstand aufweist. Bei genauer Betrachtung der experimentellen Ergebnisse aus Tab. 7.2, sind die Verfahren in vier Gruppen einzuteilen:

1. Der mit Abstand größte mittlere minimale Abstand wird bei der Clusterung mit dem **Maximum-Linkage-Algorithmus** erreicht. Daher ist dieses Verfahren das beste.
2. Der zweitgrößte Wert ergibt sich aus der Clusterung mit dem **Winner-takes-all-Netz**. Es bildet mit deutlichem Abstand die zweitbeste Clusterungsmethode.
3. Das drittbeste Verfahren wird von den **schnellen SOMs** gebildet. Der Abstand zu den mit den Standard-SOMs erzeugten Clusterungen ist minimal.
4. Das vierte und letzte Verfahren entsteht durch die Clusterung mit den **Standard-SOMs**.

Für das vorliegende Beispiel sind die in die neuen Verfahren gestellten Erwartungen erfüllt worden. Die neuen Verfahren sind alle deutlich schneller als die Standard-SOMs. Auch die durch $AvgMinDist$ repräsentierte Trennschärfe ist bei den neuen Verfahren deutlich höher als bei den Standard-SOMs. Insbesondere der Maximum-Linkage-Algorithmus weist hierfür hervorragende Werte auf. Auch das speziell angepasste WTAN zeigt gute Werte.

Darstellung der Clusterungsergebnisse im Merkmalsraum

Mit zu den anschaulichsten Ergebnisdarstellungen gehört die Visualisierung der Lage der Zentroide im Merkmals- bzw. Parameterraum. Sie gibt dem Anwender den größten Aufschluss über die Clusterung. Die folgende Bildserie zeigt die Lage der Klassenrepräsentanten nach der Clusterung und Klassifizierung.

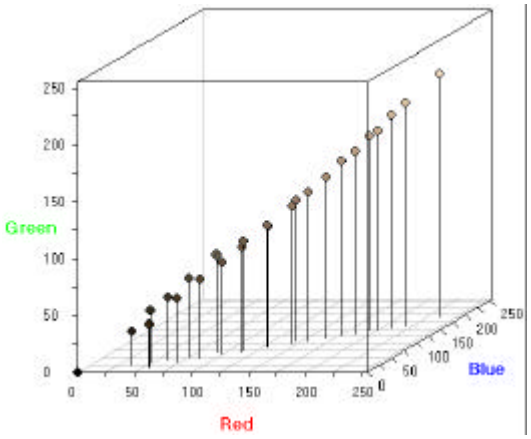


Abb. 7.8: Klassenrepräsentanten nach der Clusterung von Abb. 7.1 mit dem Maximum-Linkage-Verfahren bei 25 vorgegebenen Klassen

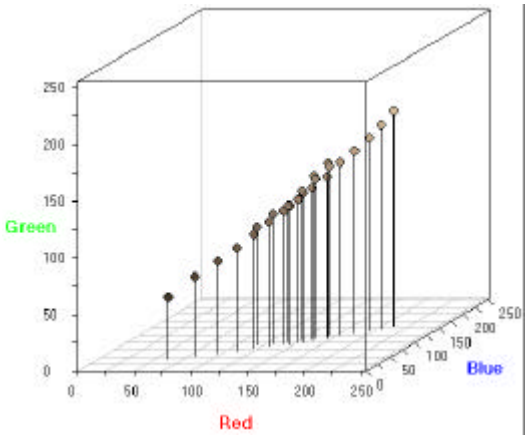


Abb. 7.9: Klassenrepräsentanten nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 Klassen und 20 Lernzyklen zum Trainieren des Netzes

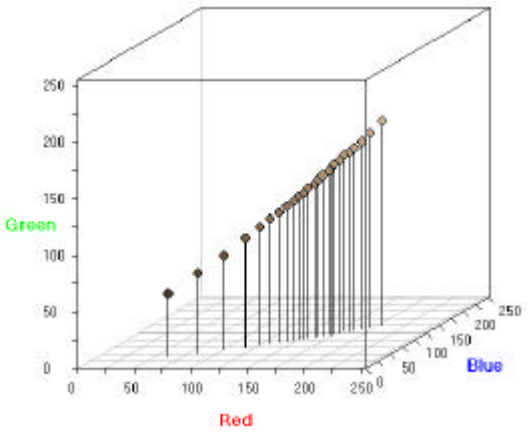


Abb. 7.10: Klassenrepräsentanten nach der Clusterung von Abb. 7.1 mit dem Standard-SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 2000 Lernzyklen zum Trainieren des Netzes

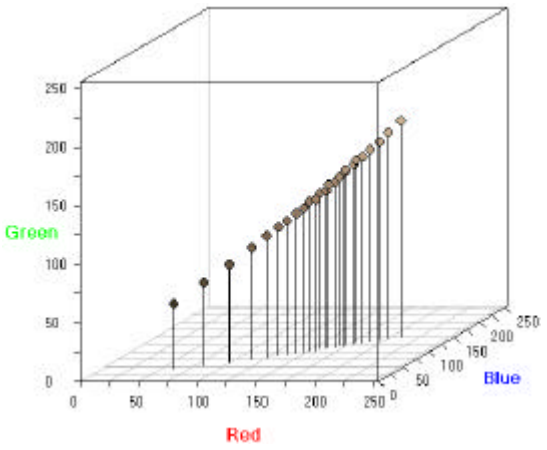


Abb. 7.11: Klassenrepräsentanten nach der Clusterung von Abb. 7.1 mit dem Standard-SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 2000 Lernzyklen zum Trainieren des Netzes

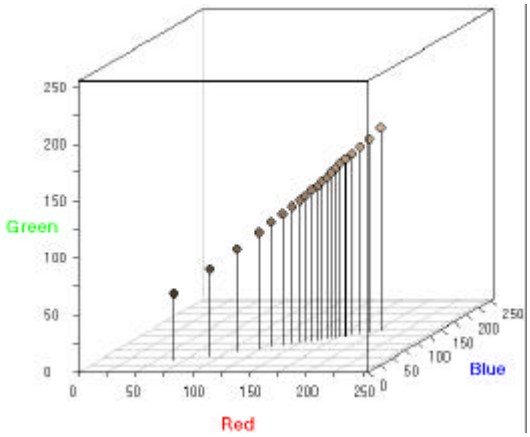


Abb. 7.12: Klassenrepräsentanten nach der Clusterung von Abb. 7.1 mit dem schnellen SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 20 Lernzyklen zum Trainieren des Netzes

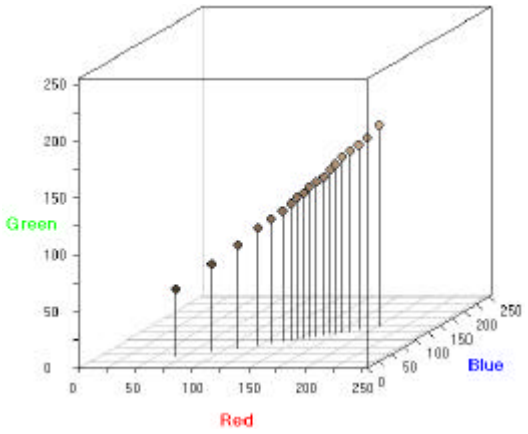


Abb. 7.13: Klassenrepräsentanten nach der Clusterung von Abb. 7.1 mit dem schnellen SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 40 Lernzyklen zum Trainieren des Netzes

Die sechs Abbildungen aus Abb. 7.8 bis Abb. 7.13 zeigen, dass die Klassenrepräsentanten fast auf einer Geraden zwischen (0, 0, 0) und (243, 219, 191) liegen. Die Struktur ist beinahe linear. Das ist aber nicht grundsätzlich der Fall. Zum Zweiten fällt auf, dass die Abb. 7.10 bis Abb. 7.13 eine große Ähnlichkeit zueinander haben, während die Klassenrepräsentanten in Abb. 7.9 über eine größere Spannweite verteilt sind. Abb. 7.8 unterscheidet sich deutlich von den übrigen. Sie zeigt, dass die Repräsentanten der Clusterung mit Maximum Linkage deutlich stärker im Parameterraum verteilt sind als bei allen neuronalen Netzen. Die dicht besetzten Bereiche zwischen

(64, 57, 41) bis (214, 191, 158) sind bei den neuronalen Netzen sehr gut herausgearbeitet worden. Die Bereiche abseits davon werden nur durch sehr wenige Werte repräsentiert. Das hat zur Folge, dass die mit SOMs geclusterten Bilder einen sehr hohen optischen Wiedererkennungswert haben. Die Abgrenzung zwischen den einzelnen Klassen ist aber nicht deutlich, weil sie dicht beieinander liegen. Erst mit dem Maximum-Linkage-Algorithmus werden überhaupt Werte im Bereich zwischen (0, 0, 0) und (64, 57, 41) gefunden. Dabei werden auch Werte oberhalb von (214, 191, 158) bis hin zu (243, 219, 191) gefunden. Diese Lagedarstellung der Klassenrepräsentanten im Merkmals- bzw. Parameterraum ist gut geeignet, um die Qualität einer Clusterung bzgl. des Auffindens seltener Elemente und der Trennschärfe zwischen den Klassen optisch zu beurteilen.

Die Kohonenkarten der Clusterungsergebnisse bei den SOMs

In diesem Abschnitt wird die Nachbarschaftsbeziehung der selbstorganisierenden Karten untersucht. Dazu werden die Gewichte der Neuronen gemäß der Beschreibung in Abschnitt 6.4.1 visualisiert. Diese Farbdarstellung der Kohonenkarten für die vier auf selbstorganisierenden Karten beruhenden, Verfahren ist in den folgenden Abbildungen dargestellt:

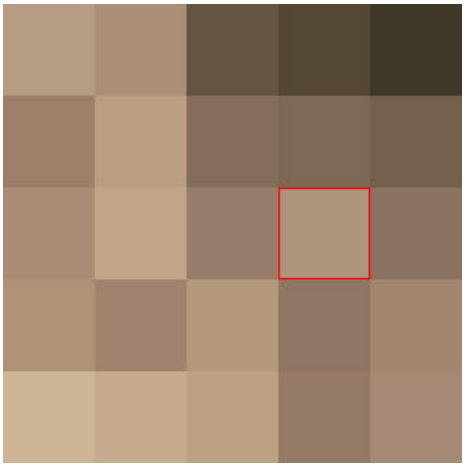


Abb. 7.14: Kohonenkarte im Farbmodus nach der Clusterung von Abb. 7.1 mit dem Standard-SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 2000 Lernzyklen zum Trainieren des Netzes

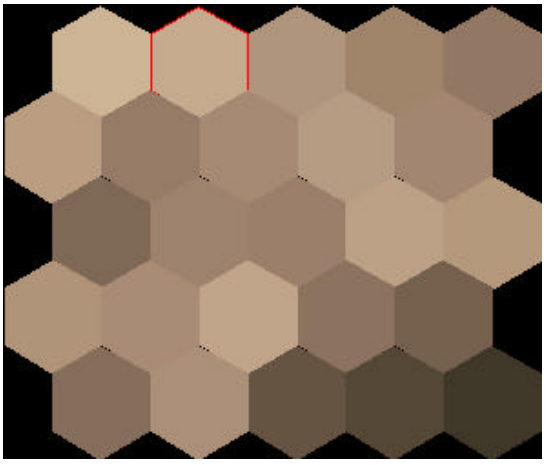


Abb. 7.15: Kohonenkarte im Farbmodus nach der Clusterung von Abb. 7.1 mit dem Standard-SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 4000 Lernzyklen zum Trainieren des Netzes

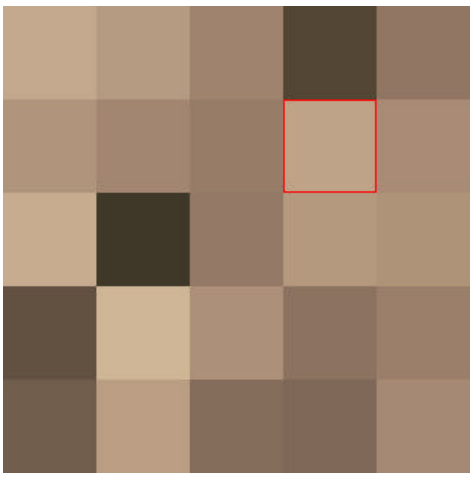


Abb. 7.16: Kohonenkarte im Farbmodus nach der Clusterung von Abb. 7.1 mit dem schnellen SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 20 Lernzyklen zum Trainieren des Netzes

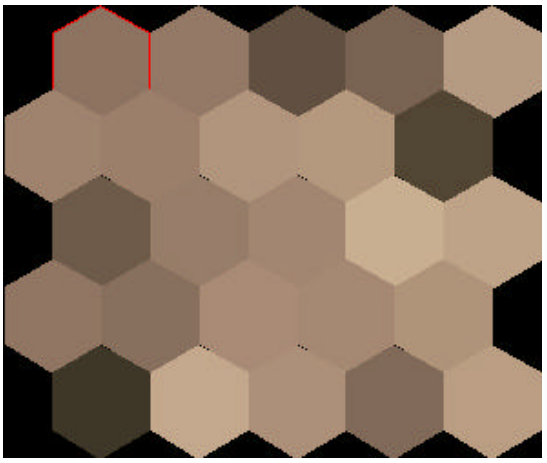


Abb. 7.17: Kohonenkarte im Farbmodus nach der Clusterung von Abb. 7.1 mit dem schnellen SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 40 Lernzyklen zum Trainieren des Netzes

Die Kohonenkarten aus Abb. 7.14 bis Abb. 7.17 sind nicht optimal entwickelt. Bei einer optimal entwickelten Karte liegen ähnliche Farbwerte nebeneinander und die Farbveränderung verläuft systematisch in jede Richtung der Karte. Abb. 6.10 aus Abschnitt 6.4.1 ist ein Beispiel für eine relativ gut entwickelte Kohonenkarte.
Die Karte aus Abb. 6.10 ist hingegen gut entwickelt. Kohonen (1995) stellt dar, dass sich große Karten mit vielen Neuronen besser entwickeln als kleinere. Das zeigt sich auch in dem oben dargestellten Fall. Auch empirische Versuche mit PicAna haben diese Tatsache bestätigt. Karten mit einer Neuronenanzahl, die einer echten Quadratzahl entsprechen, entwickeln sich leichter als Karten, bei denen die letzte Reihe des Neuronengitters nur teilweise besetzt ist. Nach Kohonen (1995) sind Lernzyklen von mehreren tausend oder zehntausend notwendig, damit sich eine Kohonenkarte gut entwickeln kann.
Die Probleme der oben abgebildeten Kohonenkarten gestalten sich wie folgt:
Die Abb. 7.14 der quadratischen Kohonenkarte des Standard-SOMs zeigt die beste Karte, auch wenn einige dunkle Felder in Bereichen liegen, in denen nur helle Felder liegen sollten. Die Karte in Abb. 7.15 ist geringfügig schlechter. Hier liegen noch mehr dunkle Felder in den Bereichen, die den hellen Feldern vorbehalten sein sollten. Die dunklen Felder liegen nicht so gut zusammen wie in Abb. 7.14. Der Grund dafür liegt u. a. in der komplexeren Nachbarschaftsstruktur, weil bei der hexagonalen Struktur jedes Neuron zwei Nachbarn mehr als bei der quadratischen Struktur hat. Da die Datenstruktur der Ausgangsdaten eher linearen Charakter hat, ist die hexagonale Nachbarschaftsstruktur weniger geeignet als die quadratische. Die Karten der schnellen SOMs (Abb. 7.16 und Abb. 7.17) sind schlecht entwickelt, weil ähnliche Farbflächen nicht nebeneinander platziert sind. Der Grund dafür liegt nach Kohonen

(1995) in der zu geringen Anzahl an Lernzyklen für diese Karten. Erstaunlich dabei ist, dass die Clusterung an sich „gut“ ist, also die Zentroide bzw. Klassenrepräsentanten gut gefunden wurden. Lediglich deren Anordnung auf der Kohonenkarte konnte sich nicht entwickeln.

Abschließend ist zu den Kohonenkarten Folgendes festzuhalten:

1. Karten mit vielen Neuronen entwickeln sich leichter als mit wenigen Neuronen.
2. Karten, deren Neuronenanzahl einer Quadratzahl entspricht, entwickeln sich leichter als andere.
3. Die quadratischen Karten entwickeln sich besser als die hexagonalen.
4. Die Anzahl der Lernzyklen ist entscheidend für die Qualität der Karte. Je mehr Lernzyklen durchgeführt werden, um so besser wird die Karte. Deshalb sind die Karten der Standard-SOMs besser als die der schnellen SOMs.
5. Im vorliegenden Fall gibt die 3-D-Darstellung der Klassenrepräsentanten im Merkmals- bzw. Parameterraum mehr und genauere Erkenntnisse als die Nachbarschaftsstruktur. Das liegt auch daran, dass diese 3-D-Darstellung direkt aus den Daten abgleitet ist, während die Nachbarschaftsstruktur auf der Kohonenkarte künstlich vorgegeben ist.

Insgesamt lassen sich im hier vorliegenden Fall auf Grund der geringen Anzahl an Lernzyklen keine umfangreichen Informationen aus der Interpretation der Kohonenkarte gewinnen.

7.2 Beispiel für die Identifizierung seltener Elemente

Im Folgenden wird ein weiteres Beispiel betrachtet. Anhand dieses Beispiels wird die unterschiedliche Wirkungsweise des Maximum-Linkage-Algorithmus gegenüber dem Winner-takes-all-Netz und den verschiedenen selbstorganisierenden Karten dargestellt. Dabei zeigt sich, dass alle Verfahren seltene Elemente finden. Außerdem zeigt dieses Beispiel, welche Probleme bei der Beurteilung einer Clusterung durch das Kriterium des mittleren minimalen Abstands auftreten können.

Die Clusterungsergebnisse in Bildform

Als Beispiel dient das bereits in Abb. 6.3 vorgestellte Bild. Die Ergebnisse der Clusterungen mit den unterschiedlichen Methoden werden in Bildform gegenübergestellt.

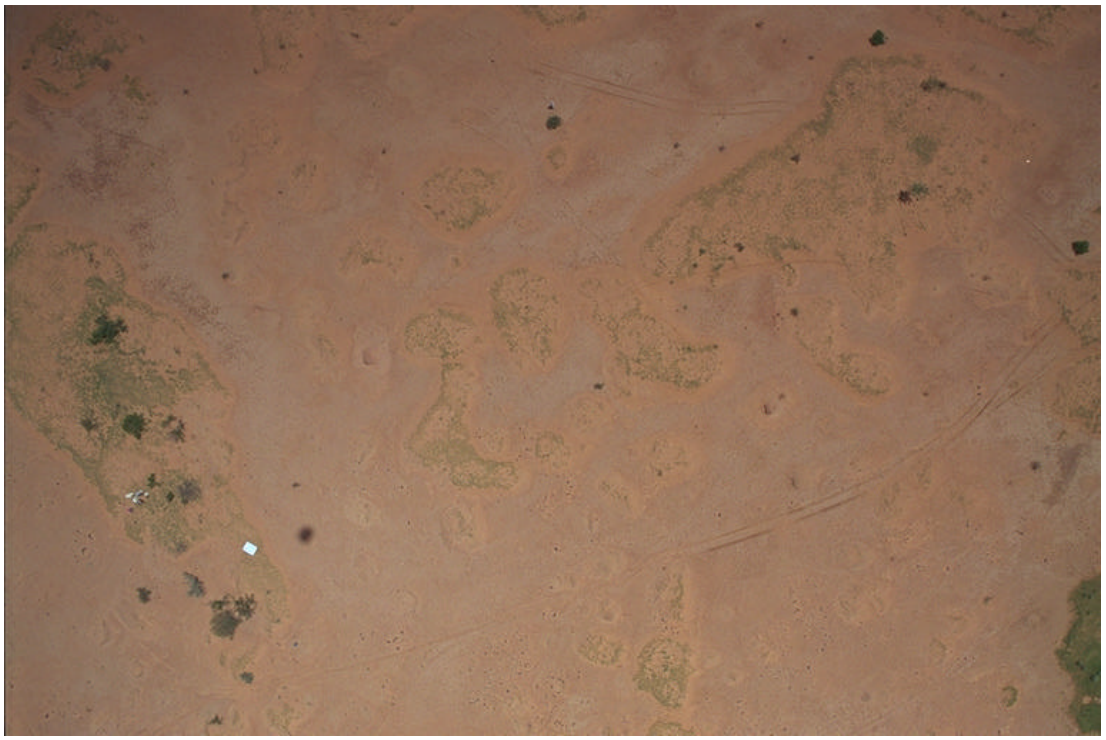


Abb. 7.18: Wiederholung des in Abb. 6.3 dargestellten Ausgangsbildes. Dies enthält 8730 Farben. Die Anzahl der Farben entspricht im Sinne einer Clusterung der Anzahl an Ausgangsklassen.

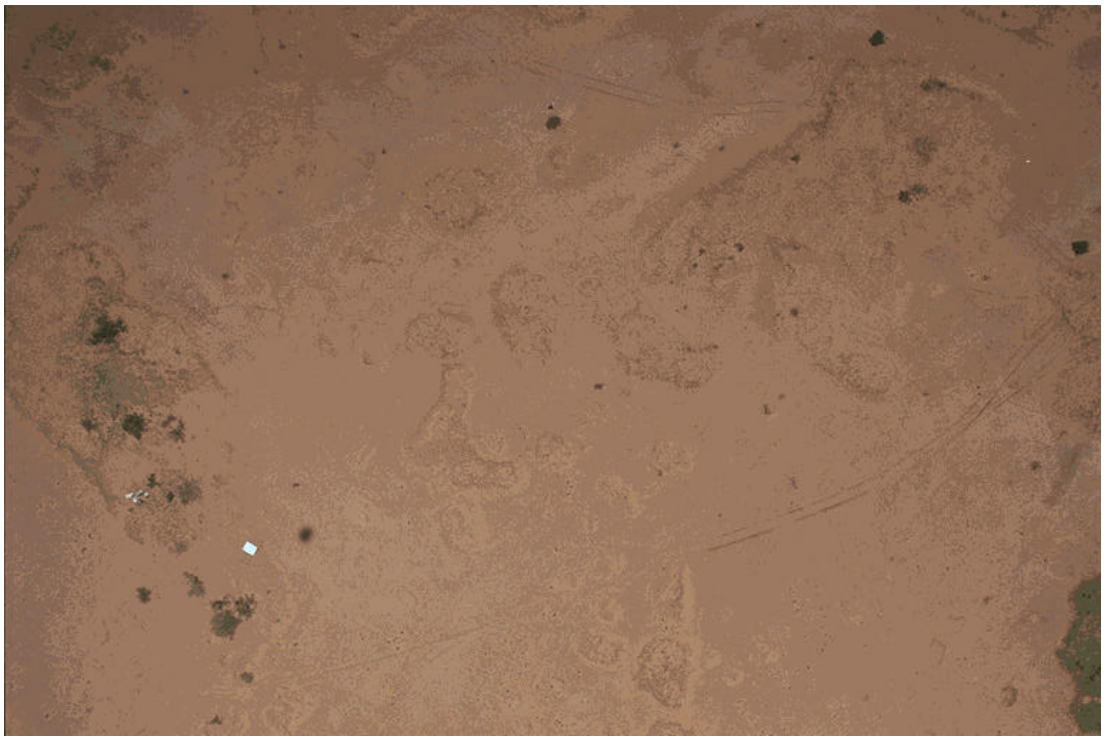


Abb. 7.19: Clusterung von Abb. 7.1 mit dem Maximum-Linkage-Verfahren bei 25 vorgegebenen Klassen.

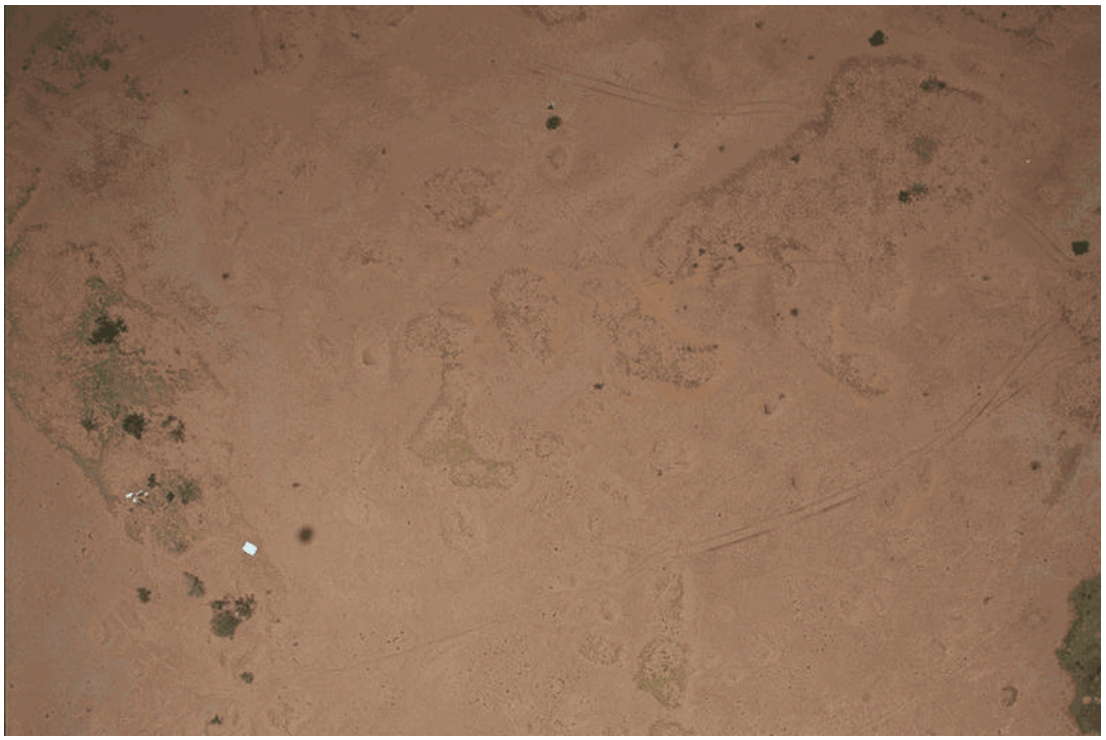


Abb. 7.20: Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 120 Lernzyklen zum Trainieren des Netzes.

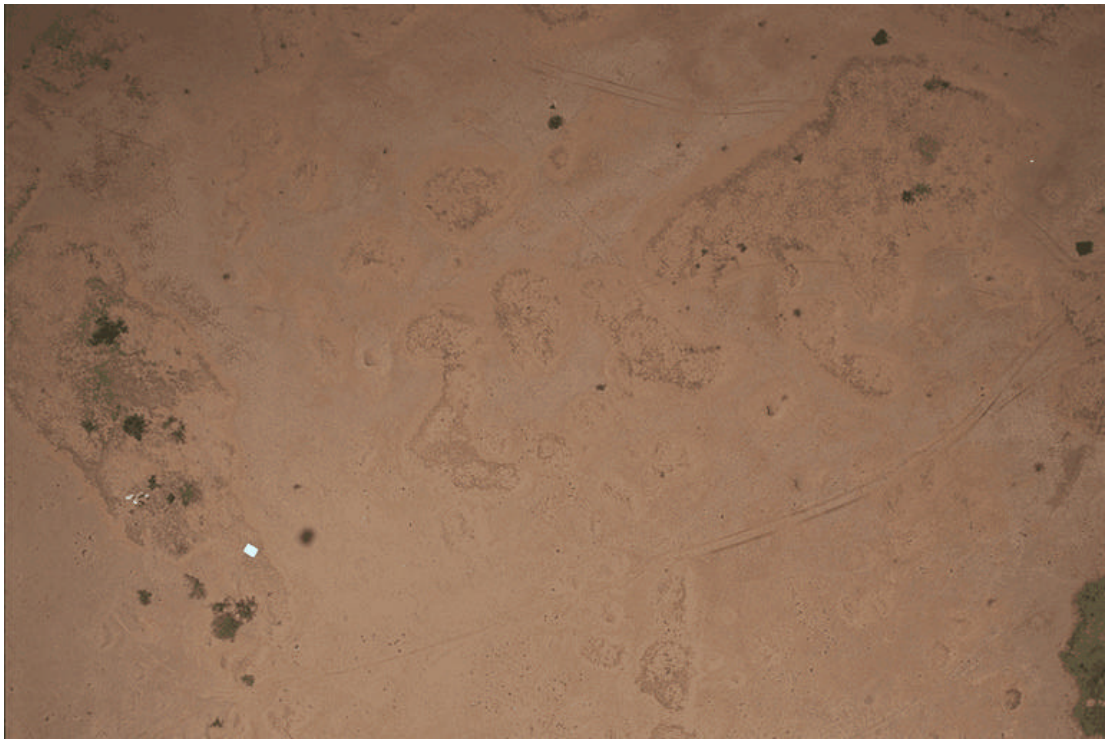


Abb. 7.21: Clusterung von Abb. 7.1 mit dem Standard-SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 2000 Lernzyklen zum Trainieren des Netzes.

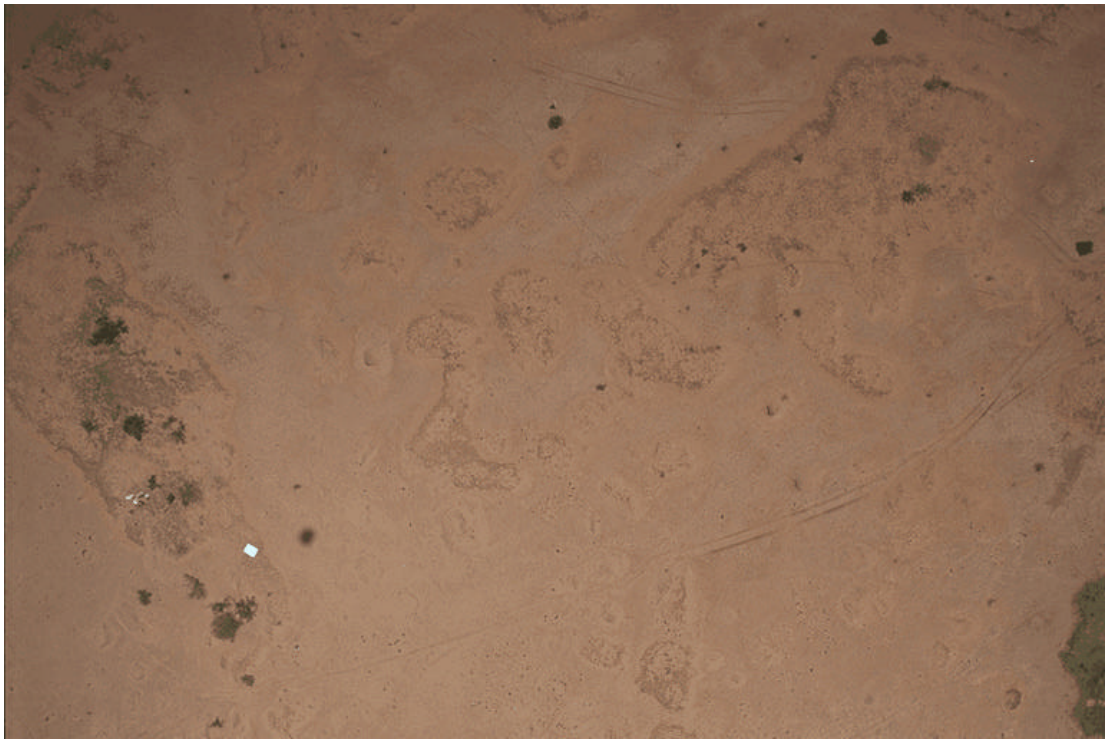


Abb. 7.22: Clusterung von Abb. 7.1 mit dem Standard-SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 4000 Lernzyklen zum Trainieren des Netzes.

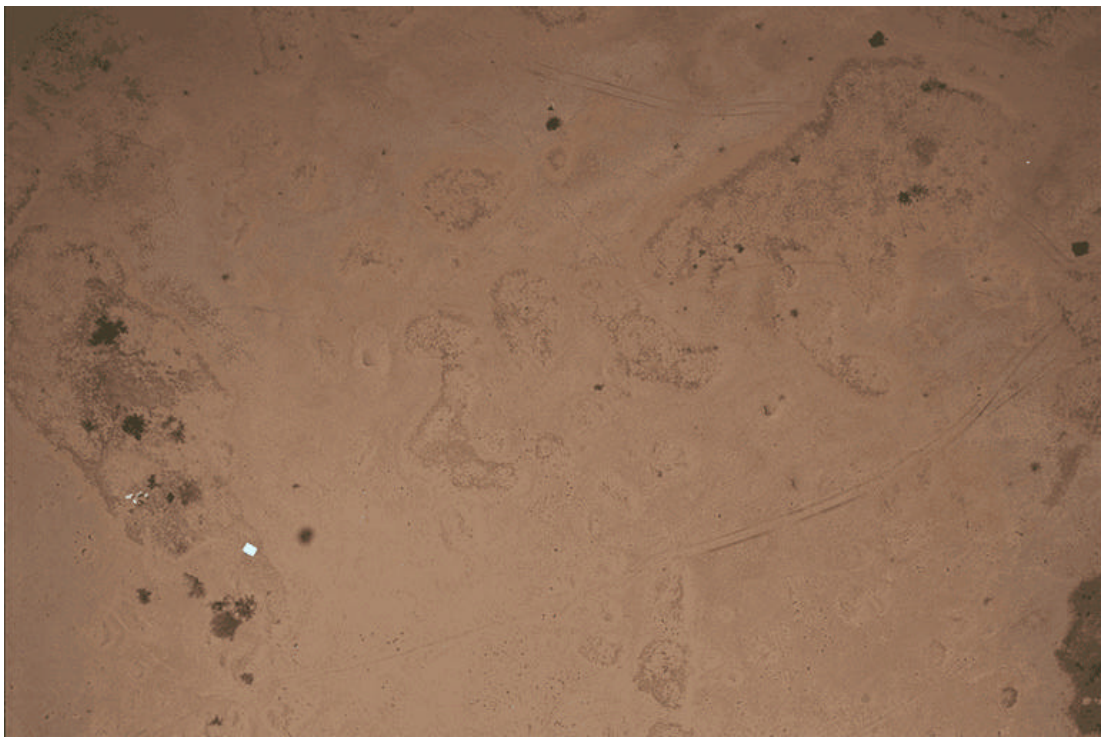


Abb. 7.23: Clusterung von Abb. 7.1 mit dem schnellen SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 20 Lernzyklen zum Trainieren des Netzes.

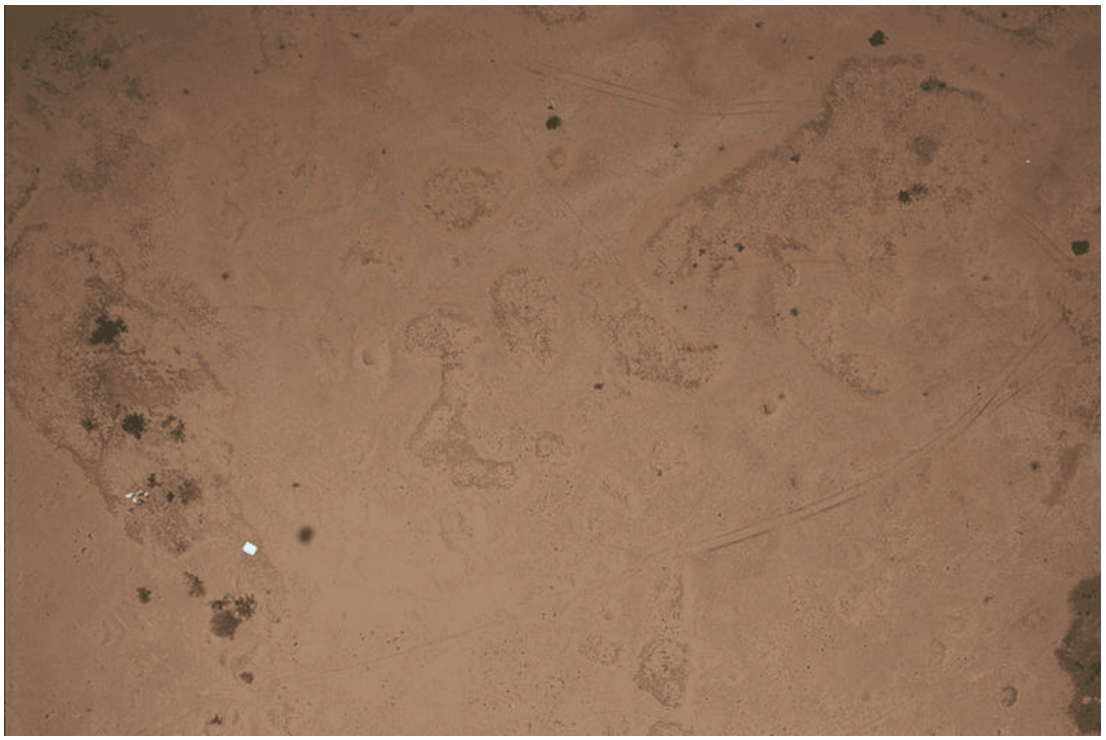


Abb. 7.24: Clusterung von Abb. 7.1 mit dem schnellen SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 40 Lernzyklen zum Trainieren des Netzes.

Die Abb. 7.2 bis Abb. 7.7 zeigen, dass die Clusterungsergebnisse, wie beim ersten Beispiel, alle eine große Ähnlichkeit zum Originalbild aufweisen. Die seltenen weißen Elemente im Bild links unten sind bei jeder Clusterung gut zu erkennen. Alle Methoden haben diese seltenen Elemente korrekt identifiziert. Auch die etwas weniger seltenen grünen Flächen werden bei allen Verfahren durch eigene Klassen gefunden.

Die Clusterung durch den Maximum-Linkage-Algorithmus neigt hier noch deutlicher zur „Flächenbildung“ als im ersten Fall. Die zusammenhängenden Flächen grenzen sich gut gegeneinander ab. Es wird eine gute Trennschärfe zwischen den Klassen erreicht. Die Flächenbildung und Trennschärfe sind in dieser Qualität nur bei Clusterungen mittels Maximum Linkage zu beobachten. Das ist eine wichtige Eigenschaft für die spätere Weiterverarbeitung der Clusterungsergebnisse. Die durch neuronale Netze erzeugten Bilder dagegen, sind dem Original sehr ähnlich. Alle mit den unterschiedlichen Netzen erzeugten Bilder sind sich optisch ähnlich.

Die folgende Tab. 7.1 zeigt, dass sich die Laufzeitunterschiede zwischen den Methoden, genau wie beim ersten Beispiel, ergeben. Die etwas längere Laufzeit des WTANs gegenüber den schnellen SOMs hat aber keine Auswirkung auf die generelle Aussage, dass das WTAN prinzipiell schneller als die schnellen SOMs ist. Diese ist im vorliegenden Fall auf die deutlich höhere Lernzyklenanzahl des WTANs gegenüber den schnellen SOMs zurückzuführen. Diese wurde hier so groß gewählt,

weil im Rahmen einer Testreihe dabei die besten Ergebnisse erzielt wurden, wie in Abschnitt 7.3 gezeigt wird.

Clusterverfahren	Lernzyklen	Laufzeit in ms
Maximum Linkage	1	29.050
Winner-takes-all-Netz	120	4.175.224
schnelles SOM (quadratisch)	20	1.378.460
schnelles SOM (hexagonal)	40	6.566.390
Standard-SOM (quadratisch)	2000	108.185.210
Standard-SOM (hexagonal)	4000	205.045.071

Tab. 7.3: Laufzeit der Clusterungen in Millisekunden. Diese hängt auch von der Anzahl der Lernzyklen ab. Daher ist die Lernzyklenanzahl mit angegeben.

Die Kenngrößen der Clusterungen

Tab. 7.4 zeigt die Kenngrößen für die Clusterungen dieses Beispiels. Die nachfolgende Interpretation des mittleren minimalen Abstands deckt eine Schwäche dieses Kriteriums auf.

Methode	Nachbarschaftsstruktur	Lernzyklen	Intra-Class-Varianz	mittlerer minimaler Abstand
Maximum Linkage	keine	-	86,3	444,3
Winner-takes-all-Netz	keine	120	35,5	826,9
Standard-SOM	quad.	2000	22,1	625,7
Standard-SOM	hex.	4000	20,9	774,6
schnelles SOM	quad.	20	28,7	505,1
schnelles SOM	hex.	40	28,3	475,5

Tab. 7.4: Kenngrößen der Clusterungen des Beispielbildes (Abb. 6.3) mit den verschiedenen Methoden bei einer vorgegebenen Klassenanzahl von 25. Das Ausgangsbild hat eine Gesamtvarianz von 797,6. Alle Zahlenwerte sind ggf. auf eine Dezimalstelle gerundet.

Was die Betrachtung der Intra-Class-Varianz betrifft, werden gegenüber dem ersten Beispiel keine neuen Erkenntnisse gewonnen. Im Vergleich zum ersten Beispiel wird die Einteilung der Methoden in Gruppen und deren Reihenfolge bestätigt. Auch hier ist die Intra-Class-Varianz bei jeder Methode gegenüber der Gesamtvarianz klein genug, um Bedingung (5.8) zu genügen.
Bei Betrachtung des mittleren minimalen Abstands zwischen den Klassenrepräsentanten bestätigt sich ein Großteil der Erkenntnisse aus Abschnitt 7.1. Die Einteilung der unterschiedlichen Methoden der neuronalen Netze in Gruppen und deren Reihenfolge ist identisch zum ersten Beispiel. Der mittlere minimale Abstand der Clusterung mit Maximum Linkage ist kleiner als alle mittleren minimalen Abstände bei den neuronalen Netzen. Dass sollte aber nach den bisherigen Erkenntnissen nicht der Fall sein. Da das Maximum-Linkage-Verfahren für die vorliegende Anwendung speziell angepasst ist, sollte es im Vergleich zu allen anderen Verfahren den größten mittleren minimalen Abstand aufweisen. Warum das bei diesem Beispiel nicht der Fall ist, wird sich bei Betrachtung der 3-D-Darstellung der Klassenrepräsentanten zeigen.

Darstellung der Clusterungsergebnisse im Merkmalsraum

Abb. 7.25 bis Abb. 7.30 zeigen die 3-D-Darstellung der Klassenrepräsentanten für die unterschiedlichen Methoden. In diesen Darstellungen sind auch die Klassen der seltenen Elemente zu erkennen. Diese liegen im oberen Drittel des Parameterraumes. Das obere Drittel des Parameterraumes wird durch Größen gekennzeichnet, die in jeder einzelnen Komponente Werte im oberen Drittel ihres Bereichs aufweisen.

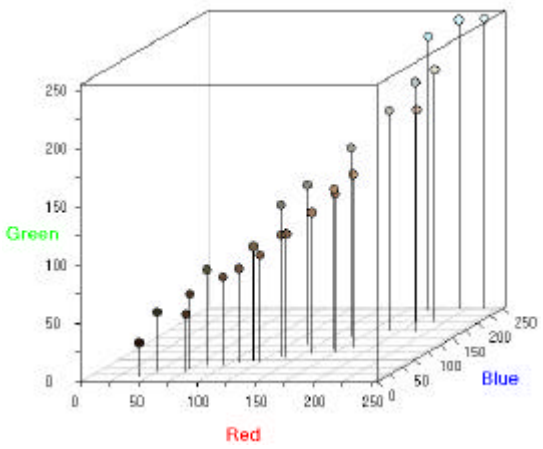


Abb. 7.25: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Maximum-Linkage-Algorithmus bei 25 vorgegebenen Klassen

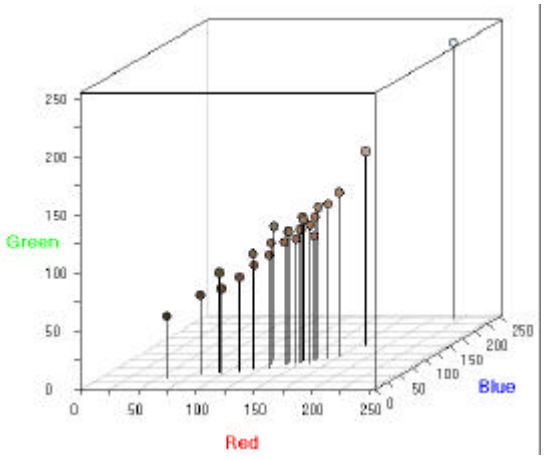


Abb. 7.26: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 120 Lernzyklen zum Trainieren des Netzes

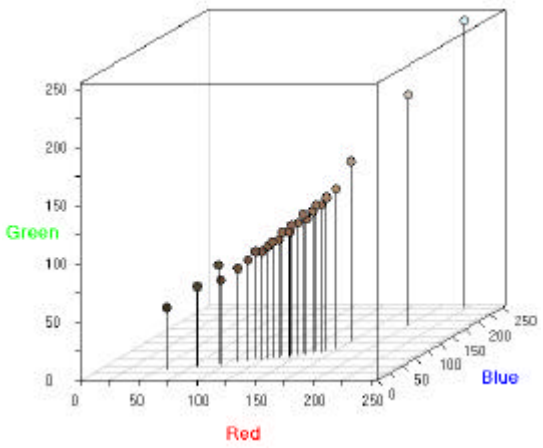


Abb. 7.27: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Standard-SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 2000 Lernzyklen zum Trainieren des Netzes

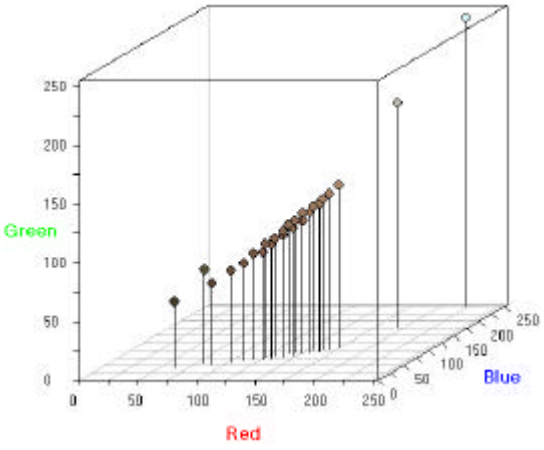


Abb. 7.28: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Standard-SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 4000 Lernzyklen zum Trainieren des Netzes

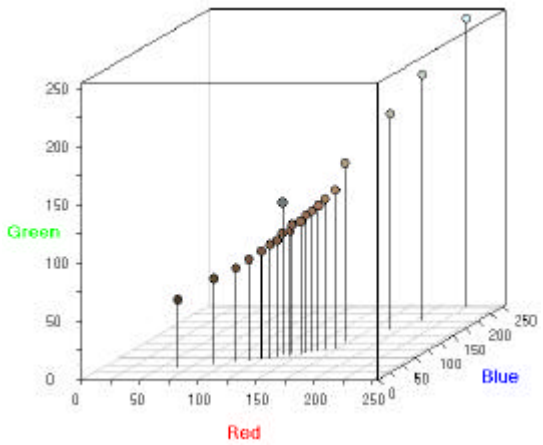


Abb. 7.29: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen SOM bei quadratischer Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 20 Lernzyklen zum Trainieren des Netzes

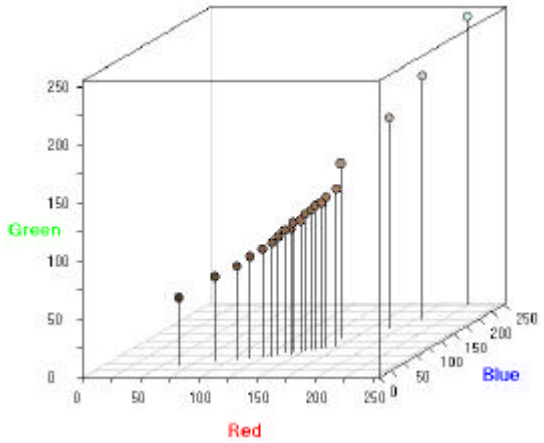


Abb. 7.30: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen SOM bei hexagonaler Nachbarschaftsstruktur, 25 vorgegebenen Klassen und 40 Lernzyklen zum Trainieren des Netzes

Die Betrachtung der sechs Abb. 7.25 bis Abb. 7.30 zeigt, dass die Hauptdiagonale durch den Parameterraum nur beim Maximum Linkage gleichmäßig besetzt ist. Bei allen neuronalen Netzen liegen im oberen Drittel des Parameterraumes maximal drei Werte. Konkret ist dieser Bereich bei den verschiedenen Methoden wie folgt besetzt:

- Bei den Standard-SOMs ist der Bereich nur mit drei Repräsentanten besetzt,
- bei den schnellen SOMs sind in diesem Bereich vier Repräsentanten vorhanden,
- beim Winner-takes-all-Netz sind es nur zwei und
- beim Maximum Linkage sind es neun Werte.

Diese Werte sind über das obere Drittel des Parameterraumes gleichmäßig verteilt. Das führt dazu, dass der Abstand zum nächsten Nachbarn für diese Werte, gegenüber den dicht besetzten Regionen, sehr groß ist. Nur beim Maximum Linkage sind diese Werte nicht groß, weil neun gleichmäßig verteilte Werte vorliegen. Dadurch entstehen bei den neuronalen Netzen bei Berechnung des mittleren minimalen Abstands gegenüber den übrigen Abständen drei bis vier extrem große Minimalabstände. Zur Berechnung des Kriteriums, wird über alle diese Werte gemittelt. Der Wert des Kriteriums wird dadurch von diesen wenigen, extrem großen Werten stark vergrößert. Das liegt daran, dass das arithmetische Mittel eines der am wenigsten robustesten Kenngrößen ist, denn sein Bruchpunkt liegt bei null (Hartung, 1989). Das bedeutet, dass schon ein einziger Extremwert, den so berechneten mittleren minimalen Abstand völlig verzerren kann.
Bei dem hier vorliegenden Beispiel hat das zur Folge, dass der Wert für das Kriterium bei den neuronalen Netzen durch die drei bis vier Extremwerte so vergrößert wird, dass sie oberhalb des Kriteriumwertes für Maximum Linkage liegen. Daher wird in diesem Fall durch das Kriterium nicht das beurteilt, was beabsichtigt ist. Das Kriterium soll dazu dienen, eine hohe Trennschärfe zwischen den Klassen zu gewährleisten. Wie das Beispiel zeigt, sollten dazu nicht alle minimalen Abstände berücksichtigt werden, sondern nur die „kleinen“ minimalen Abstände. Ein Lösungsansatz zur Verbesserung des Kriteriums könnte darin bestehen, statt des Durchschnitts robustere Berechnungsmethoden anzuwenden. Es lassen sich z. B. ein- oder zweiseitig, symmetrisch oder asymmetrisch getrimmte oder winsorisierte Mittel zur Berechnung des mittleren minimalen Abstands verwenden.
Auf die Betrachtung der Kohonenkarten wird bei diesem Beispiel verzichtet, da die Qualität dieser Karten noch schlechter ist als im ersten Beispiel. Damit bieten sich durch die Kohonenkarten keine Möglichkeiten für eine weitere Interpretation.

7.3 Auswirkungen der Clusterungsparameter auf die Ergebnisse

Dieser Abschnitt untersucht empirisch die Auswirkungen, die unterschiedliche Anzahlen von Lernzyklen beim Anlernen neuronaler Netze haben. Dazu werden Testreihen mit zwei Luftaufnahmen durchgeführt. Dabei wurden zum einen die verschiedenen Methoden durchgetestet, als auch innerhalb der einzelnen Methoden Tests mit unterschiedlicher Anzahl an Lernzyklen für die neuronalen Netze durchgeführt. Zur Analyse werden die Kenngrößen und 3-D-Darstellungen der Testreihen untersucht. Anhand dieser Untersuchungen lässt sich ein Eindruck vermitteln, wie sich die unterschiedliche Lernzyklenanzahl bei den verschiedenen Methoden auswirken.

Testreihe mit Luftaufnahme aus Abb. 7.1

Für die erste Testreihe wird die Aufnahme aus Abb. 7.1 verwendet. Tab. 7.5 enthält die Kenngrößen für die verschiedenen Methoden mit den unterschiedlichen Anzahlen an Lernzyklen. Zu jeder aufgeführten Clusterung gibt es auch die entsprechende 3-D-Darstellung der Lage der berechneten Zentroide im Merkmals- bzw. Parameterraum. Diese sind in Anhang A aufgeführt.

Methode	**Nachbar-schafts-struktur**	**Lern-zyklen**	**Intra-Class-Varianz**	**mittlerer minimaler Abstand**	**Berechnungs-zeit in ms**
Maximum Linkage	-	-	71,5	330,9	42.570
Winner-takes-all-Netz	-	5	38,3	159,8	145.330
	-	10	35,5	173,6	997.280
	-	20	34,7	173,3	1.462.560
	-	40	35,1	174,6	3.806.070
	-	80	35,7	175,6	5.403.130
	-	120	35,9	169,7	7.490.790
	-	140	34,5	170,3	19.449.340
	-	160	35,4	175,1	30.340.020
Standard-SOM	quad.	500	24,7	131,5	36.887.690
	"	1000	24,8	122,5	75.577.700
	"	2000	24,3	131,7	121.194.970
	"	4000	23,9	132,9	204.921.403
Standard-SOM	hex.	500	25,3	128,2	48.712.490
	"	1000	24,7	130,5	71.116.590
	"	2000	24,2	135,7	102.764.170
	"	4000	23,8	135,8	205.146.326
schnelles SOM	quad.	5	27,2	130,1	111.220
	"	10	31,9	183,9	1.477.330
	"	20	28,2	137,5	7.573.180
	"	40	27,1	130,5	11.795.390
	"	80	25,7	128,5	16.495.390
	"	160	26,1	137,5	7.949.820
schnelles SOM	hex.	5	29,9	233,0	892.210
	"	10	30,3	141,1	4.634.890
	"	20	28,0	129,2	7.658.320
	"	40	29,3	138,9	11.958.050
	"	80	28,7	136,6	15.586.300
	"	160	26,2	137,3	24.819.350

Tab. 7.5: Kenngrößen der Clusterungen des Beispielbildes (Abb. 7.1) mit verschiedenen Methoden bei einer vorgegeben Klassenanzahl von 25. Das Ausgangsbild hat bei einer Größe von 768 x 512 = 393.216 Pixeln und 10573 verschiedenen Farben, eine Gesamtvarianz von 1287,5. Alle Zahleneinträge sind auf eine Dezimalstelle gerundet.

Die Angaben aus Tab. 7.6 und die 3-D-Darstellungen der Clusterungen in Anhang A sollen dazu dienen, die „besten“ Clusterungen innerhalb einer Methode zu identifizieren. Das erfolgt in drei Schritten. Im ersten wird auf Grund der Kenngrößen beurteilt, bei welcher Anzahl an Lernzyklen die besten Ergebnisse entstehen. Im zweiten Schritt wird anhand der 3-D-Darstellungen festgestellt, bei wie vielen Lernzyklen die Methoden die besten Ergebnisse erzielen. Beim dritten werden die Erkenntnisse aus Schritt eins und zwei in Kombination betrachtet. Auf Grund dessen wird eine Empfehlung gegeben, welche Lernzyklenanzahl unter Berücksichtigung beider Gesichtspunkte am besten ist. Die Ergebnisse der ersten drei Schritte sind in Tab. 7.6 zusammengefasst.

Methode	**Anzahl Lernzyklen, bei denen die Methode die besten Kenngrößen liefert**	**Anzahl Lernzyklen, bei denen die Methode die beste Lagedarstellung der Zentroide liefert**	**Empfehlung auf Grund der Kombination der Erkenntnisse aus beiden vorhergehenden Spalten**
Maximum Linkage	-	-	-
Winner-takes-all-Netz	160	20	20
Standard-SOM mit quadratischer Nachbarschaftsstruktur	4000	2000	2000
Standard-SOM mit hexagonaler Nachbarschaftsstruktur	4000	4000	4000
schnelles SOM mit quadratischer Nachbarschaftsstruktur	10	20	20
schnelles SOM mit hexagonaler Nachbarschaftsstruktur	5	40	40

Tab. 7.6: Übersicht, wie viele Lernzyklen bei den einzelnen Methoden bei verschiedenen zu Grunde gelegten Kriterien optimale Clusterungsergebnisse bei einer empirischen Testreihe mit dem Standard-Testbild (Abb. 7.1) liefern.

Da beim Maximum-Linkage-Algorithmus keine Lernzyklen nötig sind, sondern die Zentroide in eindeutig deterministischer Weise ermittelt werden, sind keine weiteren Ausführungen zu diesem Punkt notwendig.

Beim Winner-takes-all-Netz ergibt sich auf Grund der Betrachtung der Kenngrößen eine Anzahl von 160 Lernzyklen als optimal. Die Kenngrößen der WTANs unterscheiden sich jedoch kaum. Nur bei 5 Lernzyklen gibt es einige Abweichungen, aber diese sind schlechter als bei anderen Lernzyklenanzahlen. Auf Grund der 3-D-Darstellung ist eindeutig die Clusterung mit 20 Lernzyklen zu bevorzugen. Daher geht die Empfehlung zu 20 Lernzyklen.
Beim Standard-SOM mit quadratischer Nachbarschaftsstruktur ergeben sich für die Kenngrößen die besten Ergebnisse bei 4000 Lernzyklen. Diese sind aber nur minimal besser als bei 2000 Lernzyklen. Die 3-D-Darstellung zeigt die besten Ergebnisse bei 2000 Lernzyklen. Somit geht die Empfehlung zu 2000 Zyklen.
Beim Standard-SOM mit hexagonaler Nachbarschaftsstruktur weisen sowohl die Kenngrößen als auch die 3-D-Darstellung bei 4000 Lernzyklen die besten Ergebnisse auf.
Bei den schnellen SOMs mit quadratischer Nachbarschaftsstruktur weisen die Kenngrößen bei 10 Lernzyklen die besten Ergebnisse auf. Die 3-D-Darstellung der Zentroide weist bei 20 Lernzyklen die besten Ergebnisse auf. Der Unterschied in der 3-D-Darstellung zwischen 20 und 160 Lernzyklen ist nur gering. Da die 3-D-Darstellung bei 10 Lernzyklen nicht optimal ist, geht die Empfehlung zu 20 Lernzyklen hin.
Bei den schnellen SOMs mit hexagonaler Nachbarschaftsstruktur, weisen die 3-D-Darstellungen für 20, 40, 80 und 160 Lernzyklen gute Eigenschaften auf. Die Unterschiede zwischen diesen vier sind nur gering. Die besten Eigenschafen sind bei 40 Lernzyklen zu finden. Die besten Kenngrößen ergeben sich bei 5, 10 und 40 Lernzyklen in genau dieser Reihenfolge. Da die 3-D-Darstellungen bei 5 und 10 Lernzyklen gegenüber den anderen deutlich schlechter sind, wird als Gesamtempfehlung eine Anzahl von 40 Lernzyklen angegeben.
Es ist festzuhalten, dass alle Methoden, die empfohlen worden sind, brauchbare Ergebnisse liefern. Das insgesamt beste Ergebnis wird mit dem Maximum-Linkage-Verfahren und Winner-takes-all-Netz erreicht. Das zeigt sich sowohl anhand der Kenngrößen als auch bei der 3-D-Darstellung der Klassenrepräsentanten.

Testreihe mit zweiter Luftaufnahme

Analog zur obigen Testreihe wird auch für die Luftaufnahme aus Abb. 6.3 eine Testreihe mit unterschiedlichen Anzahlen an Lernzyklen durchgeführt. Die Kenngrößen dieser Testreihe sind in Tab. 7.7 aufgeführt. Die zugehörigen 3-D-Darstellungen sind in Anhang B aufgeführt.

Methode	**Nachbar-schafts-struktur**	**Lern-zyklen**	**Intra-Class-Varianz**	**mittlerer minimaler Abstand**	**Berechnungs-zeit in ms**
Maximum Linkage	-	-	86,3	444,3	29.050
Winner-takes-all-Netz	-	1	60,0	96,0	42.130
	-	5	38,7	1053,1	141.380
	-	10	38,5	960,2	292.750
	-	20	36,8	962,4	762.150
	-	40	36,5	1138,4	2.464.830
	-	80	33,0	1161,2	3.106.920
	-	120	35,5	826,9	4.175.224
	-	140	35,4	813,2	4.742.890
	-	160	37,3	1000,4	5.415.898
Standard-SOM	quad.	500	23,0	771,1	77.494.000
	"	1000	23,2	813,5	92.821.390
	"	2000	22,1	625,7	108.185.210
		4000	21,6	624,1	204.455.537
Standard-SOM	hex.	500	23,5	661,4	77.682.010
	"	1000	22,4	769,2	66.359.220
	"	2000	22,4	617,4	47.305.290
		4000	20,9	774,6	205.045.071
schnelles SOM	quad.	5	31,7	402,7	304.890
	"	10	29,8	37,8	982.010
	"	20	28,7	505,1	1.378.460
	"	40	27,1	410,5	6.543.440
	"	80	25,5	445,0	8.619.290
	"	160	28,7	805,7	9.809.750
schnelles SOM	hex.	5	29,9	429,9	154.890
	"	10	29,9	424,3	930.930
	"	20	28,0	416,5	1.329.910
	"	40	28,3	475,5	6.566.390
	"	80	26,7	825,9	8.651.970
	"	160	27,8	450,1	4.563.370

Tab. 7.7: Kenngrößen der Clusterungen des Beispielbildes (Abb. 6.3) mit verschiedenen Methoden, bei einer vorgegebenen Klassenanzahl von 25. Das Ausgangsbild hat bei einer Größe von 768 x 512 = 393.216 Pixeln, 7830 verschiedene Farben und eine Gesamtvarianz von 797,6. Alle Tabelleneinträge sind ggf. auf eine Dezimalstelle gerundet.

Analog zu der ersten Testreihe werden innerhalb jeder Methode die „besten“ Clusterungen identifiziert. Dazu werden die Kenngrößen und die 3-D-Darstellungen wieder separat beurteilt. Außerdem werden diese Beurteilungen wieder in Kombination betrachtet und auf Grund dessen wird eine Empfehlung für eine Clusterung gegeben. Die Ergebnisse dieser Bemühungen sind in Tab. 7.8 dargestellt.

Methode	**Anzahl Lernzyklen, bei denen die Methode die besten Kenngrößen liefert**	**Anzahl Lernzyklen, bei denen die Methode die beste Lagedarstellung der Zentroide liefert**	**Empfehlung auf Grund der Kombination der Erkenntnisse aus beiden vorhergehenden Spalten**
Maximum Linkage	-	-	-
Winner-takes-all-Netz	80	120 oder 140	140
Standard-SOM mit quadratischer Nachbarschaftsstruktur	1000	2000	2000
Standard-SOM mit hexagonaler Nachbarschaftsstruktur	4000	4000	4000
schnelles SOM mit quadratischer Nachbarschaftsstruktur	160	40	40
schnelles SOM mit hexagonaler Nachbarschaftsstruktur	80	40	40

Tab. 7.8: Übersicht, wie viele Lernzyklen bei den einzelnen Methoden bei verschiedenen zu Grunde gelegten Kriterien optimale Clusterungsergebnisse bei einer empirischen Testreihe mit dem Standard-Testbild (Abb. 6.3) liefern.

Beim Winner-takes-all-Netz ergibt sich auf Grund der Betrachtung der Kenngrößen eine Anzahl von 80 Lernzyklen. Dieser Wert ist durch die Verzerrung des mittleren minimalen Abstands, durch wenige Werte im oberen Drittel des Parameterbereichs, bedingt. Die 3-D-Darstellung der Zentroide zeigt die besten Ergebnisse bei 120 oder

140 Lernzyklen. Insgesamt sind 140 Lernzyklen zu empfehlen, weil dabei die 3-D-Darstellung am besten den geforderten Eigenschaften entspricht.
Beim Standard-SOM mit quadratischer Nachbarschaftsstruktur, ergeben sich die besten Kenngrößen bei 2000 Lernzyklen. Diese Anzahl ergibt sich aus einem Beurteilungs-Kompromiss zwischen Intra-Class-Varianz und verzerrtem mittlerem minimalen Abstand. Die beiden Größen weisen hier in unterschiedliche Richtungen. Die beste 3D-Darstellung ergibt sich bei 2000 Lernzyklen.
Beim Standard-SOM mit hexagonaler Nachbarschaftsstruktur, ergeben sich die besten Kenngrößen bei 4000 Lernzyklen. Auch die 3-D-Darstellung zeigt bei 4000 Lernzyklen die beste Struktur. Daher ist die Clusterung mit 4000 Lernzyklen zu empfehlen.
Bei den schnellen SOMs mit quadratischer Nachbarschaftsstruktur weisen die Kenngrößen bei 160 Lernzyklen die besten Ergebnisse auf. Auch dieser Wert ist durch die schwache Besetzung des oberen Drittels des Parameterbereichs verzerrt. Die 3-D-Darstellung dagegen zeigt eindeutig bei 40 Lernzyklen die besten Ergebnisse.
Bei den schnellen SOMs mit hexagonaler Nachbarschaftsstruktur, verhält es sich genauso wie bei denen mit quadratischer Nachbarschaftsstruktur. Allerdings weisen die Kenngrößen auf 80 statt 160 Lernzyklen hin. Auch diese sind aus den selben Gründen wie bei den quadratischen SOMs verzerrt. Die Empfehlung geht daher zu 40 Lernzyklen.

Werden die besten Clusterungen aus allen Methoden gewählt, so führt das zu folgenden Ergebnissen: Das beste Ergebniss für die gewünschten Eigenschaften liefert das Maximum-Linkage-Verfahren. Ähnlich gute Eigenschaften weist das schnelle, hexagonale SOM bei 40 Lernzyklen auf. Die Entscheidung für das SOM wurde dabei auf Grund des Vergleichs der 3-D-Darstellung aller als empfehlenswert erachteten Clusterungen getroffen.

8 Ausblick

Der Ausblick gliedert sich in drei Teilbereiche. Im ersten Teilbereich geht es um die Weiterentwicklung des eingeführten Maximum-Linkage-Algorithmus. Der zweite Bereich beschäftigt sich mit der Weiterentwicklung von Kenngrößen zur Beurteilung der Clusterungsergebnisse. Abschließend wird vorgestellt, wie die Clusterungsergebnisse weiterverarbeitet werden sollen und welche weiteren Ziele verfolgt werden.

Die Weiterentwicklung des Maximum-Linkage-Algorithmus

Die Verbesserung des Maximum-Linkage-Algorithmus setzt bei der Einbeziehung zusätzlicher Informationen über die Verteilung der Daten (Farbwerte) im Ausgangsbild an. Dazu könnten im Vorfeld mehr Informationen über die empirische Verteilung der Ausgangsdaten ermittelt werden. Diese Informationen sind bei der Auswahl der Zentroide, in den Ansatz einzuarbeiten. Welche Informationen im Detail von Nutzen sind und wie diese in die Auswahl der Zentroide einzuarbeiten sind, muss noch erarbeitet werden. Ein erster Ansatz in diese Richtung wäre, die Auswahl der Zentroide zu ändern. Dabei ist nicht mehr ein Teil der Zentroide gewichtet und ein anderer ungewichtet zu bestimmen, sondern alle werden unter einer Gewichtung gewählt. Für diese Gewichtung wird nicht mehr die absolute Häufigkeit der Farbwerte im Originalbild verwendet. Statt dessen basiert die Gewichtung auf einer Transformation der absoluten Häufigkeiten. Die Transformation sollte auf den Eigenschaften der empirischen Verteilung der Ausgangsdaten beruhen.

Die Weiterentwicklung der Kenngrößen zur Beurteilung von Clusterungen

Eine der wesentlichen Kenngrößen, zur Beurteilung einer Clusterung im vorliegenden Fall, ist der in Abschnitt 5.3 eingeführte mittlere minimale Abstand. Wie sich in Abschnitt 7.2 gezeigt hat, weist dieses Kriterium in einigen Fällen Defizite auf. Im Falle der neuronalen Netze, kann es passieren, dass der Wert des Kriteriums deutlich größer wird, als er nach der optischen Beurteilung der 3-D-Darstellung der Zentroide sein dürfte. Das kann dazu führen, dass auf Grund dieses Kriteriums eine „falsche" Clusterung als die beste identifiziert wird. Wie in Abschnitt 7.2 erwähnt, liegt die Lösung in der Weiterentwicklung des Kriteriums. Wie diese Modifizierung im Detail auszusehen hat und ob sich damit die Beurteilung der Lage der Zentroide im Parameterraum durch den Analysten ersetzen lässt, ist noch offen.

Die Weiterverarbeitung der Clusterungsergebnisse

Es wurden unterschiedliche Verfahren zur pixelbasierten Bildclusterung erarbeitet. Dabei wurde die Weiterverwendung in der Erosionsanalyse in den Vordergrund gestellt. Abschließend soll auf die weiterführenden Untersuchungen eingegangen werden. Liegen Bodendaten (Bodenzusammensetzung und –beschaffenheit, Grad der Versandung usw.) und Bilddaten gemeinsam vor, müssen diese zur weiteren Analyse miteinander verknüpft werden. Das Schema in Abb. 8.1 skizziert das dazu notwendige Vorgehen:

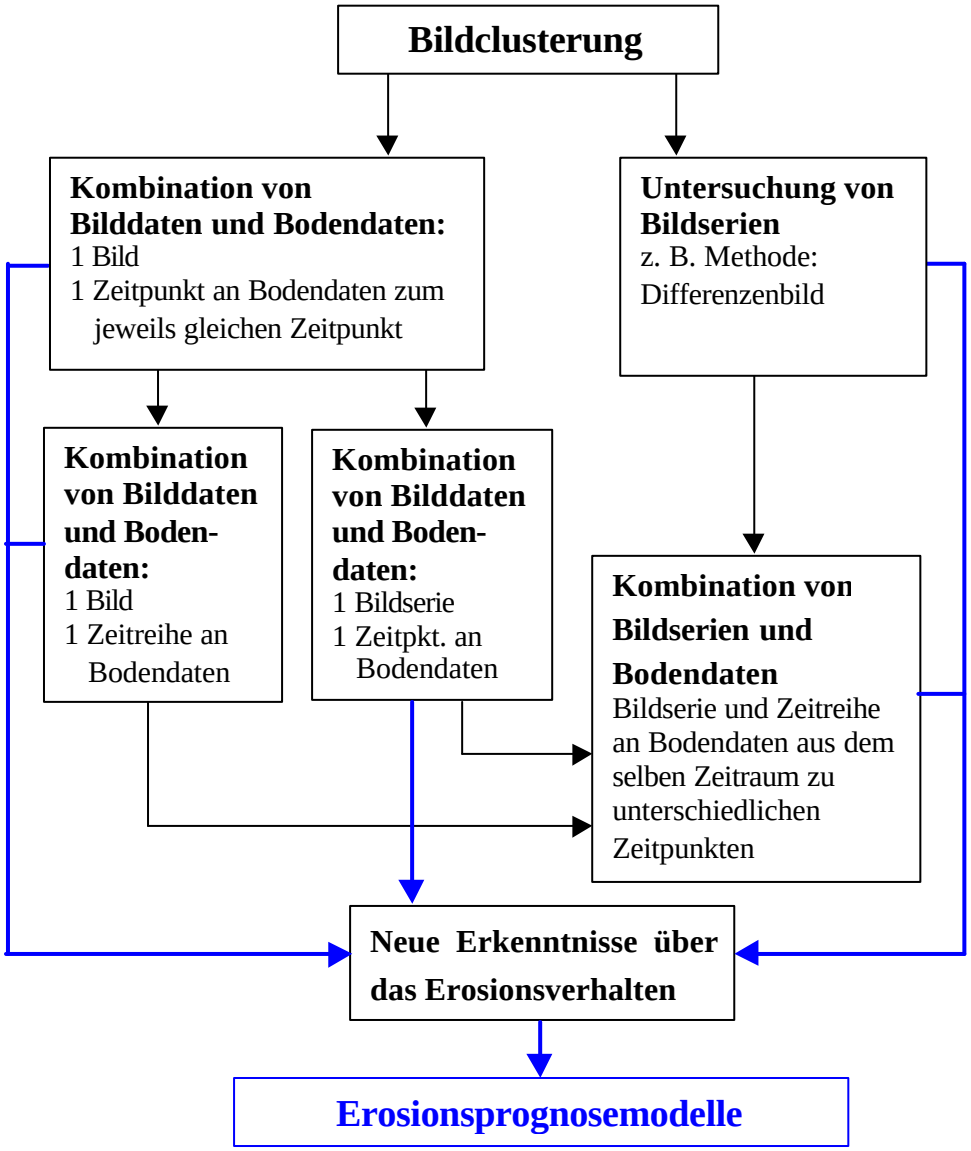


Abb. 8.1: Weitergehende Methodenentwicklung auf Grund der vorgestellten Bildclusterung

Ausgehend von der Bildclusterung, sollen nähere inhaltliche Erkenntnisse über Versandung und/oder Erosion von den abgebildeten Flächen gewonnen werden. Anhand der in Abb. 8.1 skizzierten Ideen, sollen diese Ziele erreicht werden.

Zum einen können zusätzliche Erkenntnisse allein auf Grund der geclusterten Bilder ohne zusätzliche Verwendung von Bodendaten gewonnen werden. Wenn Bildserien existieren, lassen sich über deren Vergleich Veränderungen im Untersuchungsgebiet feststellen. Dazu werden die einzelnen Bilder geclustert und miteinander verglichen. So ein Vergleich könnte zum Beispiel über ein „Differenzenbild“ erfolgen. Dabei werden die geclusterten Bilder deckungsgleich übereinander gelegt und die pixelweise Differenz gebildet. Ist die Differenz nahe null, hat sich an dieser Stelle fast nichts verändert. Je größer die Differenz ist, desto größer ist die Veränderung. Dabei müssen aber unbedingt „Vorzeicheneffekte“ berücksichtigt werden, um nicht aus einem antiproportionalen Verhalten der beiden „Bilder“ falsche Schlüsse zu ziehen. Sicherlich sind vor Anwendung solcher Methoden noch eine Reihe von Überlegungen und Untersuchungen nötig, z. B.: Sind alle Aufnahmen separat zu clustern oder sollen nur im ersten Bild Zentroide berechnet werden? Sollen alle Bilder anhand dieser Zentroide des ersten Bildes klassifiziert werden?
Zum anderen besteht ein wesentlicher Aspekt der Fortführung der Untersuchungen in der Verbindung der geclusterten Bilder mit Bodendaten. Die Art der Verbindung ist noch zu erarbeiten. Dabei sind verschiedene Vorgehensweisen denkbar:

- Der Vektor der Farbwerte für jeden Pixel wird vor der Clusterung erweitert und um den Vektor der Bodendaten ergänzt. Die Bodendaten werden in diesem Fall in die Clusterung mit einbezogen.
- Jeder Pixel des Bildes wird nach der Clusterung um einen Vektor von Bodendaten ergänzt.
- Jede in der Clusterung ermittelte Klasse wird um Bodendaten ergänzt.
- Die Aufnahme und die Bodendaten werden unabhängig voneinander geclustert und anschließend kombiniert. Diese Kombination kann sowohl pixelweise als auch klassenweise vorgenommen werden.
- Des Weiteren ergeben sich zusätzliche Probleme, wenn für den Zeitraum einer Bildserie nur ein Zeitpunkt an Bodendaten zur Verfügung steht. Analoges gilt für den Fall, dass zu einer Aufnahme eine Reihe von Bodendaten vorliegt.

Für alle diese Fälle müssen geeignete Modelle entwickelt werden, so dass die daraus resultierenden Ergebnisse weitere Erkenntnisse über Erosions- und Verwüstungsprobleme liefern.

Insbesondere in der Molekulargenetik spielen Clusterverfahren eine herausragende Rolle bei der Auswertung von Gen-Expressionsdaten. Solche Daten werden mit Hilfe sogenannter DNA-Chips erhoben und erlauben die gleichzeitige Untersuchung von vielen tausend Genen bezüglich ihres Expressionsverhaltens in Tumorgeweben. Sowohl die Clusterung von Expressionsvektoren der unterschiedlichen Gene als auch die Clusterung von Gewebe-Expressionsprofilen liefern wichtige Informationen zur Diagnose und Therapie von Tumorerkrankungen. Über die Perspektiven dieser wich-

tigen Anwendungsmöglichkeit von Clusterverfahren in der Krebsforschung berichtet Urfer (2001).

So bleibt zu hoffen, dass diese Arbeit dazu beiträgt, mit Clusterungsproblemen erfolgsorientierter umzugehen. Besonders bei der Anwendung in der Erosionsforschung wird ein Beitrag geliefert, diese Prozesse besser zu verstehen. Damit ist die Erwartung verbunden, den zerstörenden Folgen dieser Effekte gemäß UNCCD (1994) entgegenwirken zu können.

Anhang A: Testreihe zum ersten Beispiel

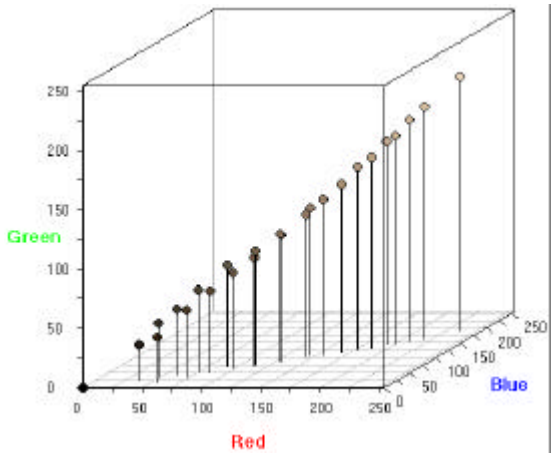


Abb. 8.2.: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Maximum-Linkage-Verfahren bei 25 vorgegebenen Klassen

Anhang A beinhaltet die Darstellung der Lage der Zentroide im Parameterraum für die Testserie, deren Kenngrößen in Tab. 7.5 aufgeführt sind. Da die Kenngrößen allein zur Beurteilung der Qualität einer Clusterung nicht ausreichen, erschließt sich die vollständige Beurteilung erst in Verbindung mit den hier aufgeführten Grafiken.

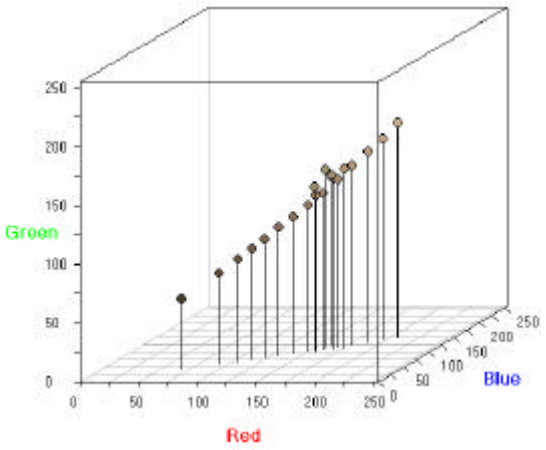


Abb. 8.3.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 5 Lernzyklen

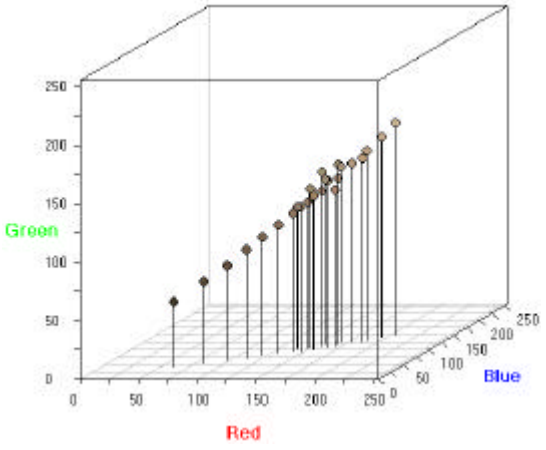


Abb. 8.3.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 10 Lernzyklen

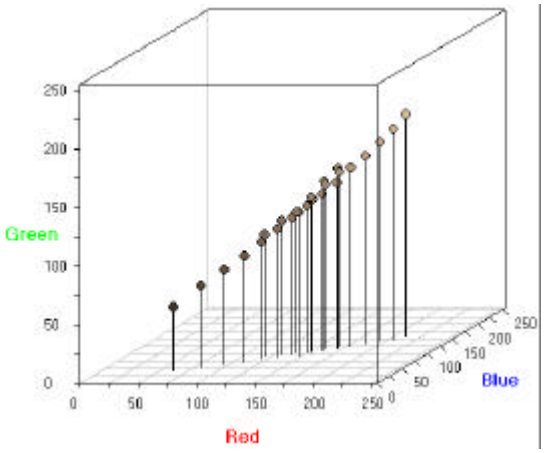


Abb. 8.3.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 20 Lernzyklen

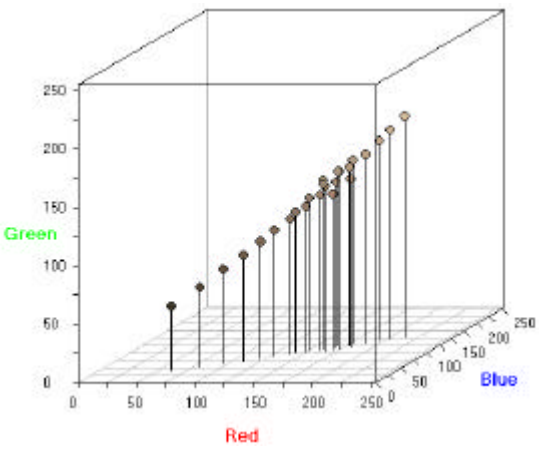


Abb. 8.3.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 40 Lernzyklen

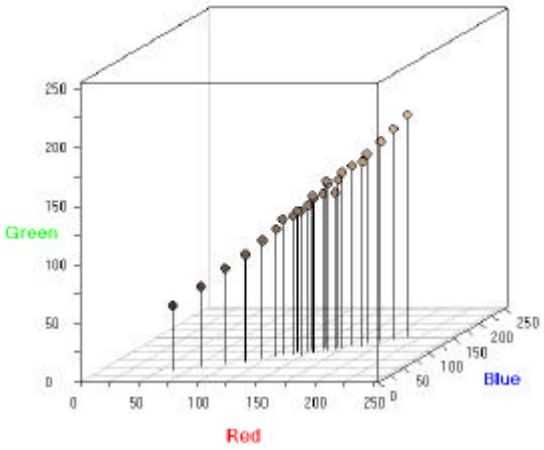


Abb. 8.3.e: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 80 Lernzyklen

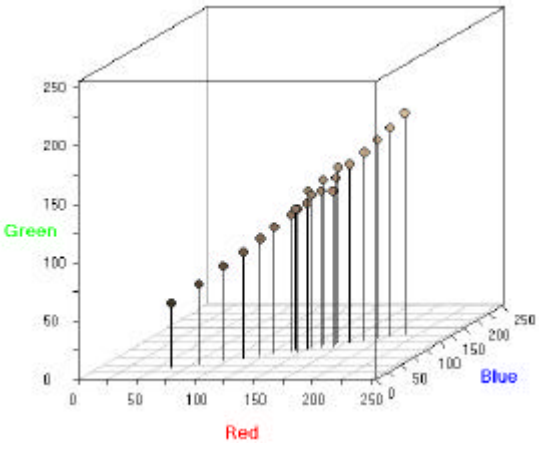


Abb. 8.3.f: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 120 Lernzyklen

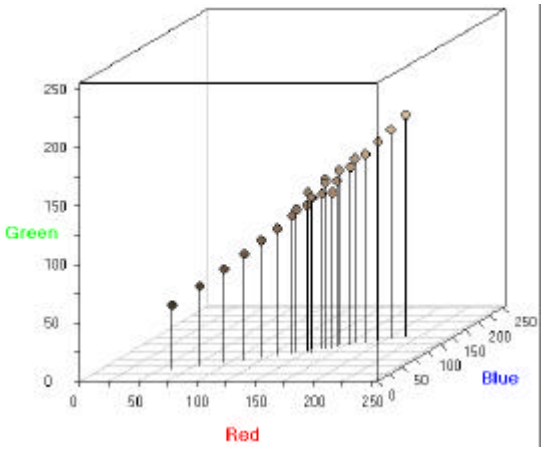


Abb. 8.3.g: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 140 Lernzyklen

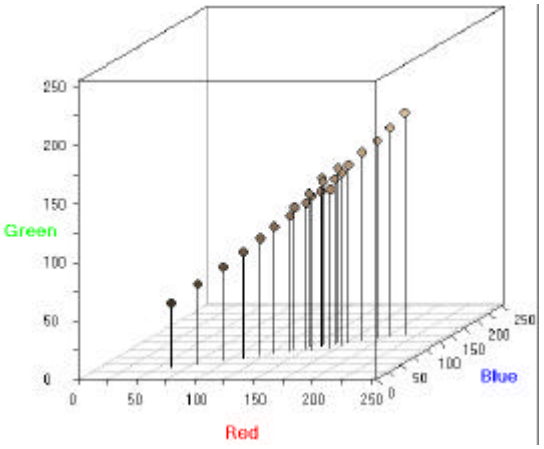


Abb. 8.3.h: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 160 Lernzyklen

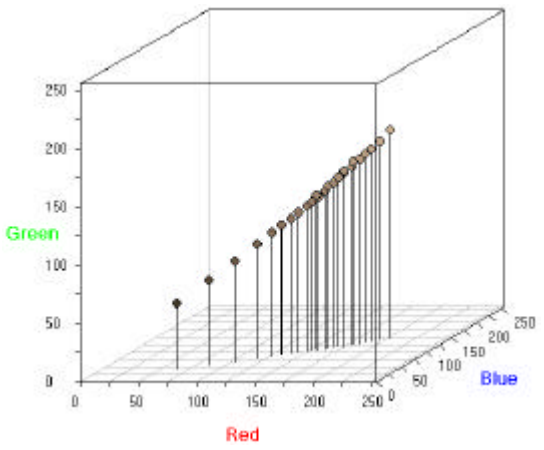


Abb. 8.4.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem quadratischen Standard-SOM bei 25 vorgegebenen Klassen und 500 Lernzyklen

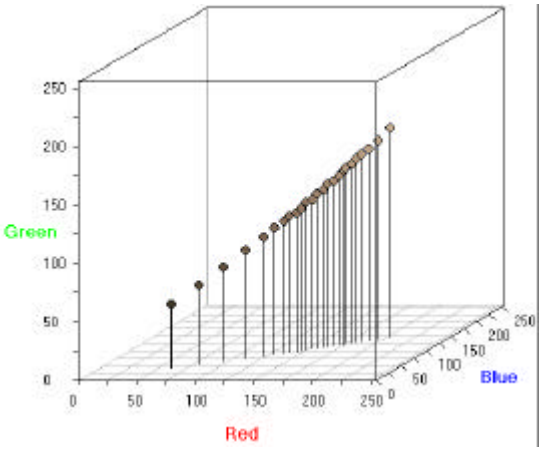


Abb. 8.4.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem quadratischen Standard-SOM bei 25 vorgegebenen Klassen und 1000 Lernzyklen

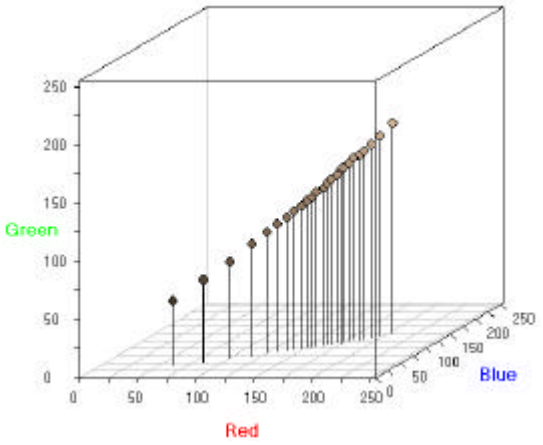


Abb. 8.4.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem quadratischen Standard-SOM bei 25 vorgegebenen Klassen und 2000 Lernzyklen

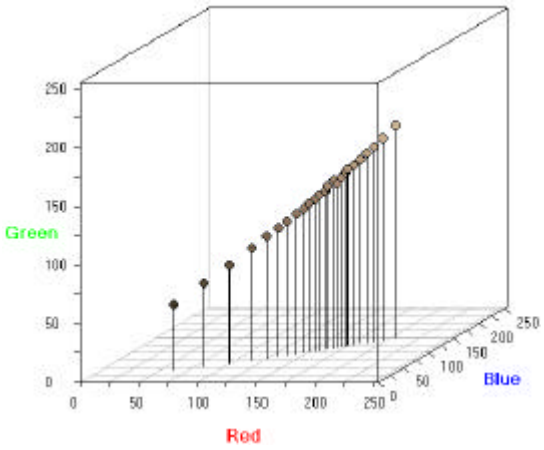


Abb. 8.4.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem quadratischen Standard-SOM bei 25 vorgegebenen Klassen und 4000 Lernzyklen

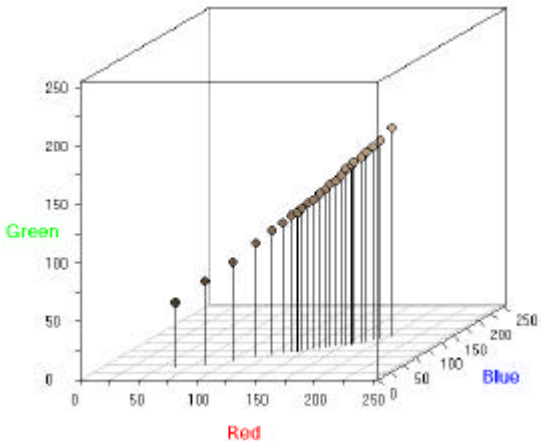


Abb. 8.5.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem hexagonalen Standard-SOM bei 25 vorgegebenen Klassen und 500 Lernzyklen

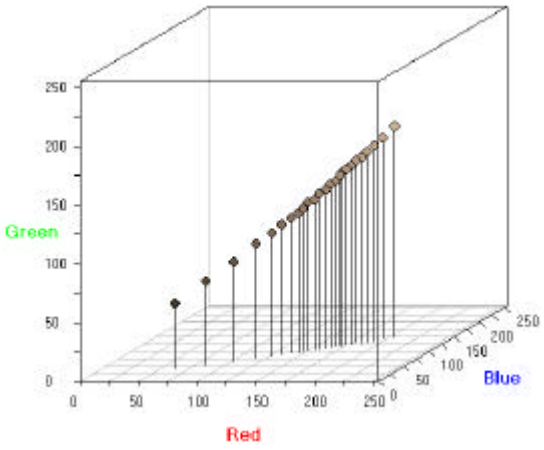


Abb. 8.5.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem hexagonalen Standard-SOM bei 25 vorgegebenen Klassen und 1000 Lernzyklen

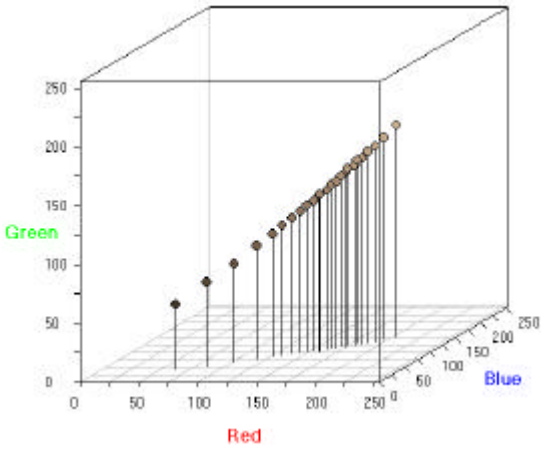


Abb. 8.5.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem hexagonalen Standard-SOM bei 25 vorgegebenen Klassen und 2000 Lernzyklen

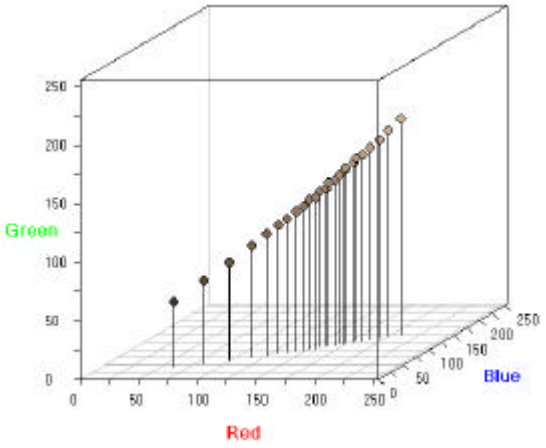


Abb. 8.5.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem hexagonalen Standard-SOM bei 25 vorgegebenen Klassen und 4000 Lernzyklen

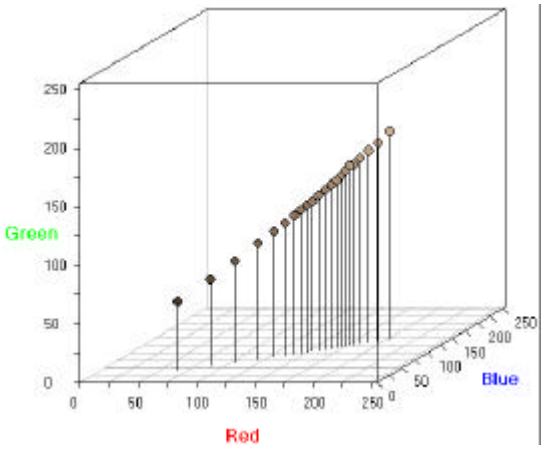


Abb. 8.6.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 5 Lernzyklen

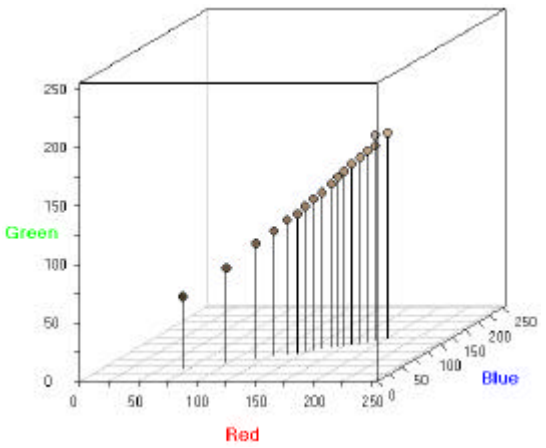


Abb. 8.6.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 10 Lernzyklen

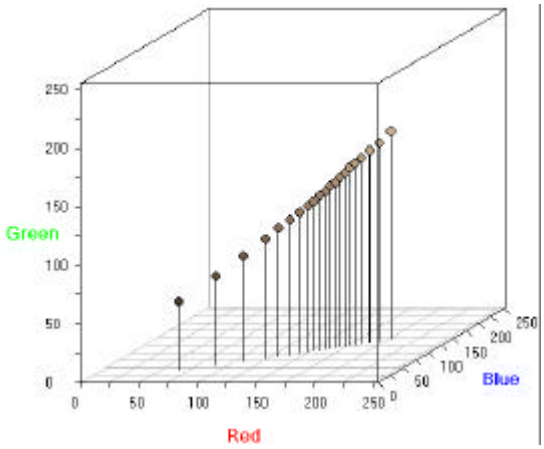


Abb. 8.6.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 20 Lernzyklen

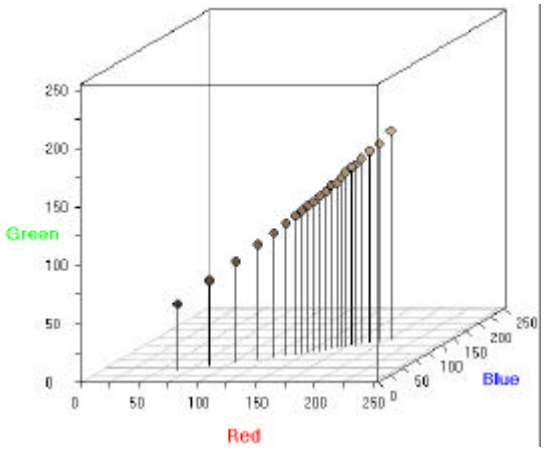


Abb. 8.6.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 40 Lernzyklen

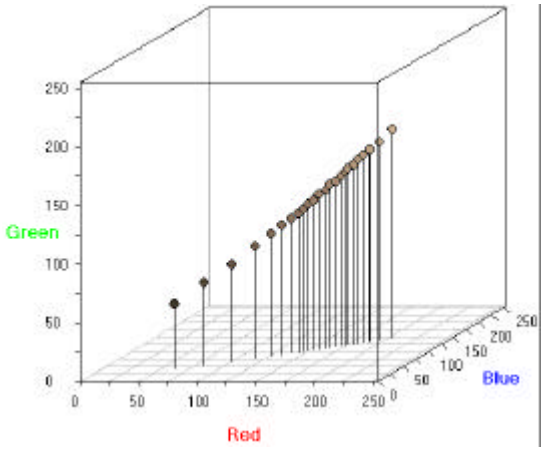


Abb. 8.6.e: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 80 Lernzyklen

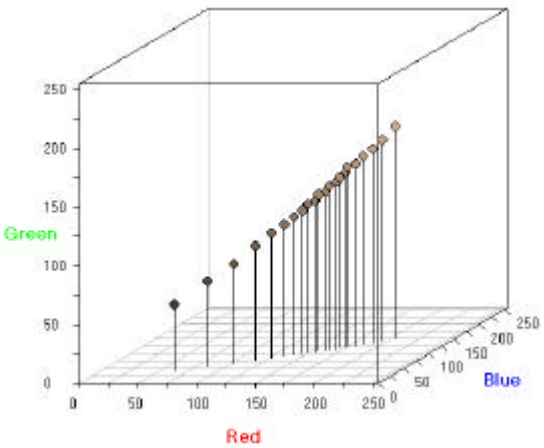


Abb. 8.6.f: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 160 Lernzyklen

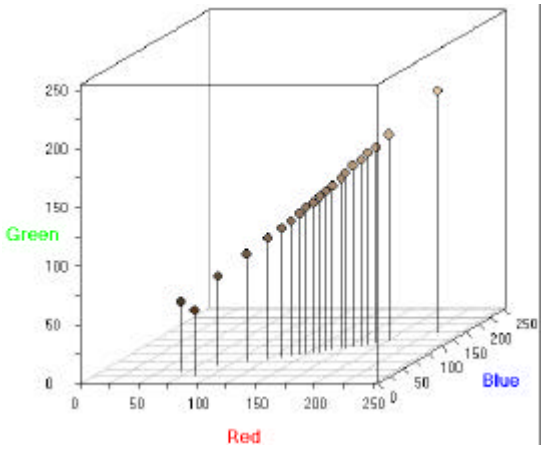


Abb. 8.7.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 5 Lernzyklen

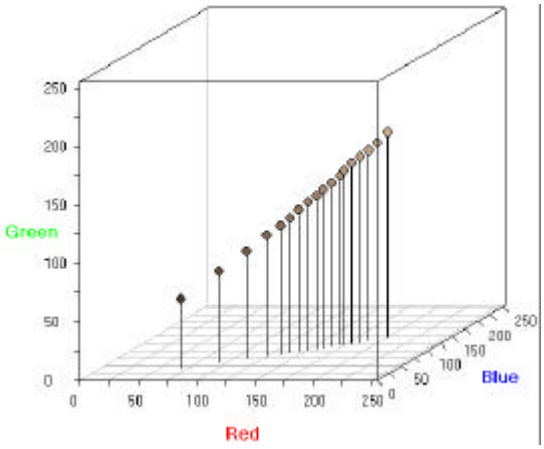


Abb. 8.7.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 10 Lernzyklen

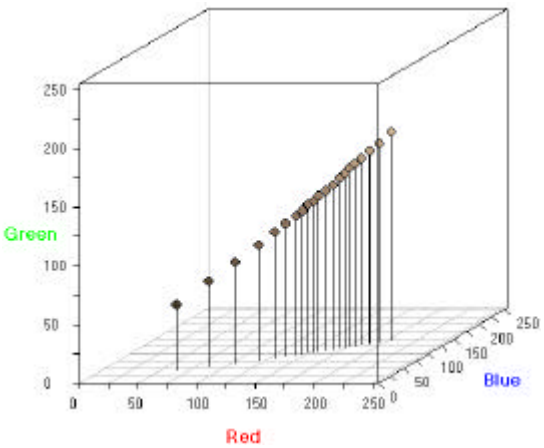


Abb. 8.7.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 20 Lernzyklen

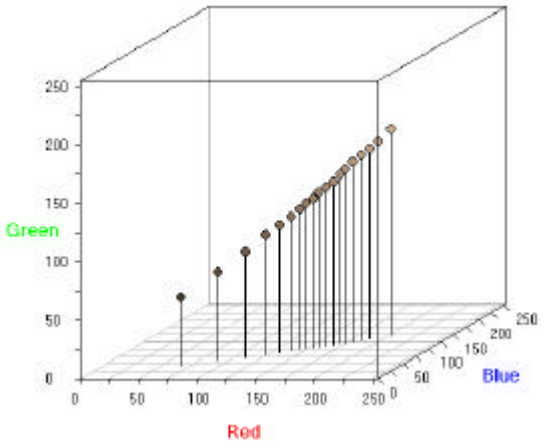


Abb. 8.7.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 40 Lernzyklen

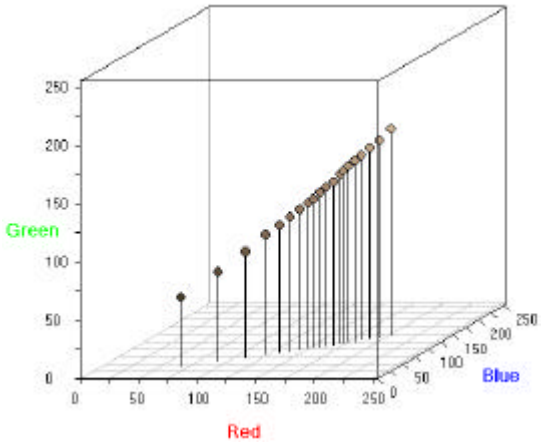


Abb. 8.7.e: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 80 Lernzyklen

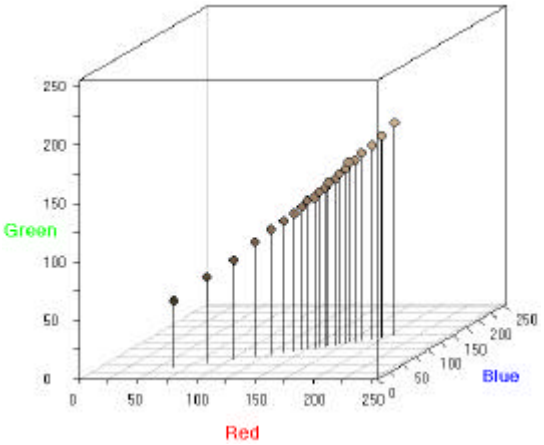


Abb. 8.7.f: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 7.1 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 160 Lernzyklen

Anhang B: Testreihe zum zweiten Beispiel

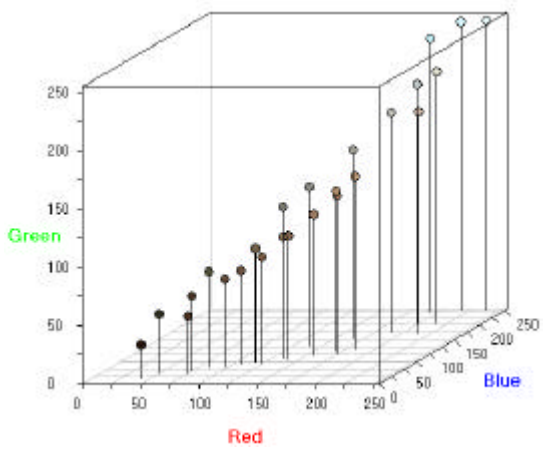


Abb. 8.8.: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Maximum-Linkage-Verfahren bei 25 vorgegebenen Klassen

Dieser Abschnitt zeigt die 3-D-Darstellung der Zentroide im Parameterraum für die Testserie mit Abb. 6.3. Die Kenngrößen dieser Testreihe sind in Tab. 7.7 aufgelistet. Erst die hier aufgeführten 3-D-Darstellungen ermöglichen die optimale Beurteilung der Qualität einer Clusterung. Die Kenngrößen allein reichen oft nicht aus oder weisen gar in eine falsche Richtung.

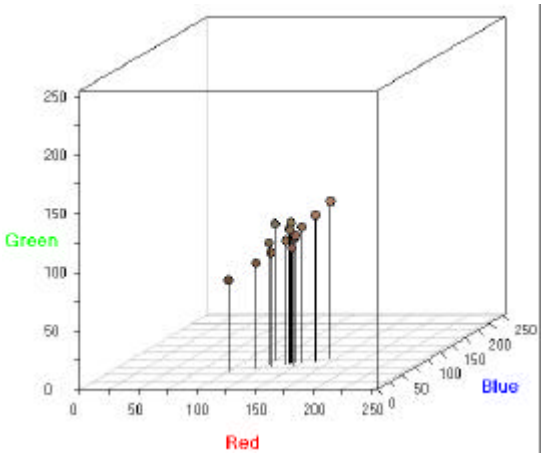


Abb. 8.9.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 1 Lernzyklus

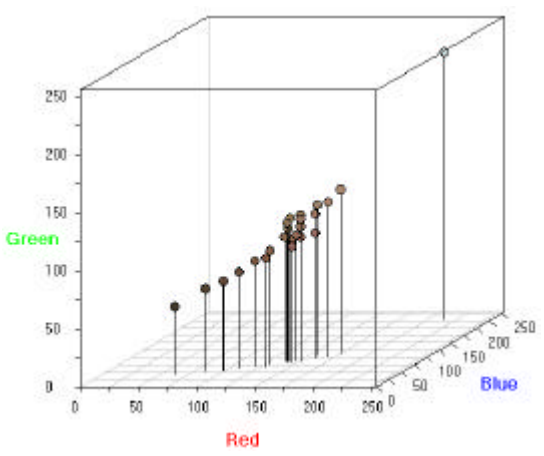


Abb. 8.9.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 5 Lernzyklen

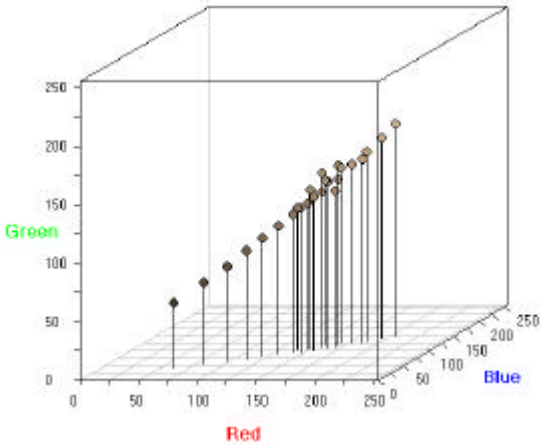


Abb. 8.9.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 10 Lernzyklen

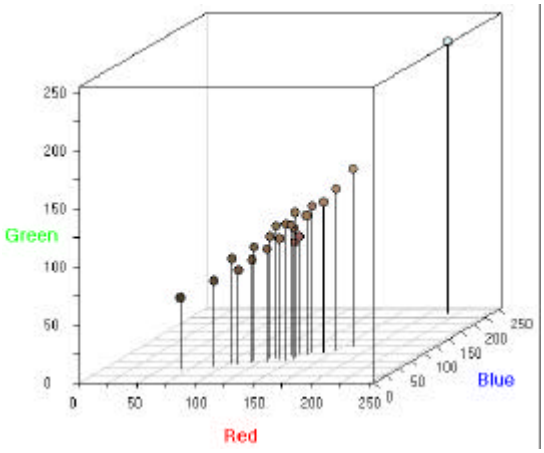


Abb. 8.9.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 20 Lernzyklen

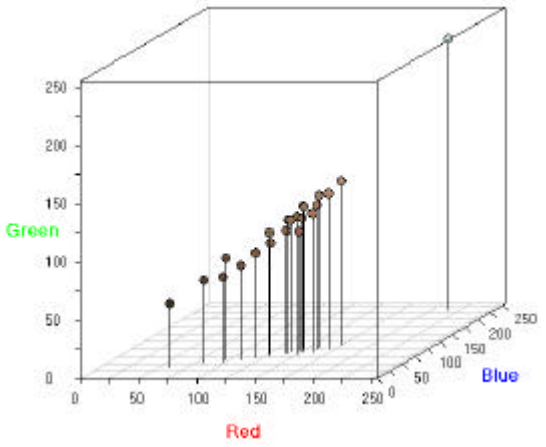


Abb. 8.9.e: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 40 Lernzyklen

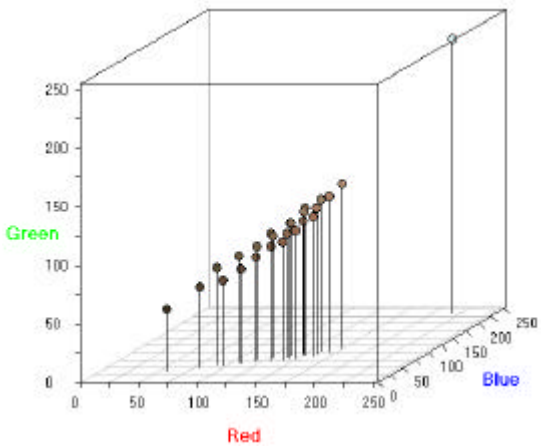


Abb. 8.9.f: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 80 Lernzyklen

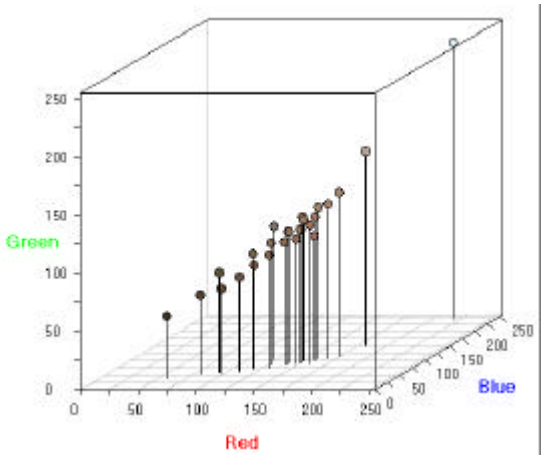


Abb. 8.9.g: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 120 Lernzyklen

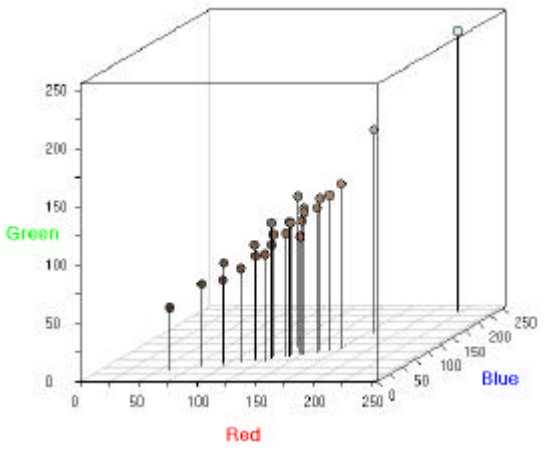


Abb. 8.9.h: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 140 Lernzyklen

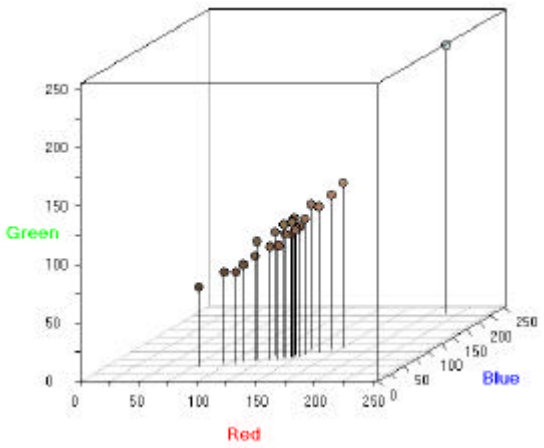


Abb. 8.9.i: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem Winner-takes-all-Netz bei 25 vorgegebenen Klassen und 160 Lernzyklen

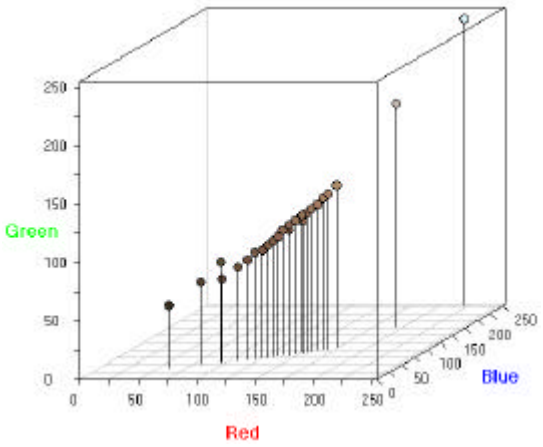


Abb. 8.10.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem quadratischen Standard-SOM bei 25 vorgegebenen Klassen und 500 Lernzyklen

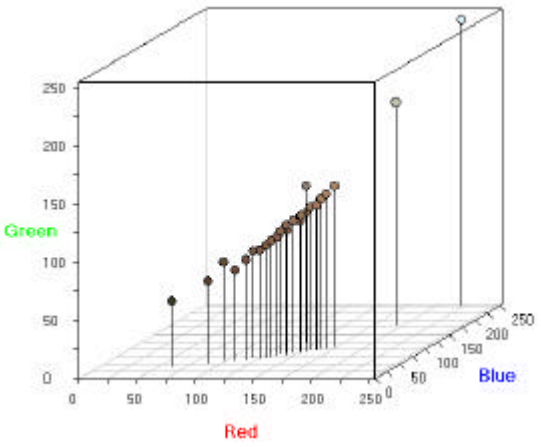


Abb. 8.10.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem quadratischen Standard-SOM bei 25 vorgegebenen Klassen und 1000 Lernzyklen

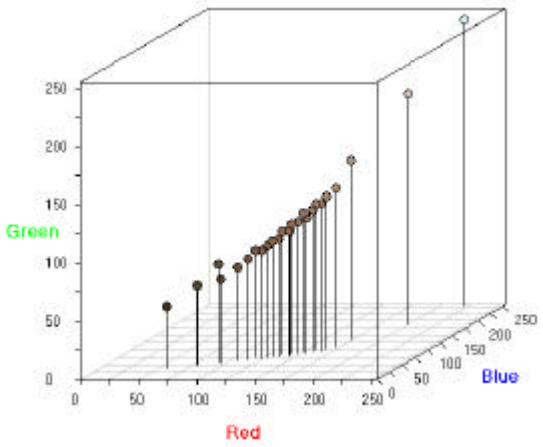


Abb. 8.10.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem quadratischen Standard-SOM bei 25 vorgegebenen Klassen und 2000 Lernzyklen

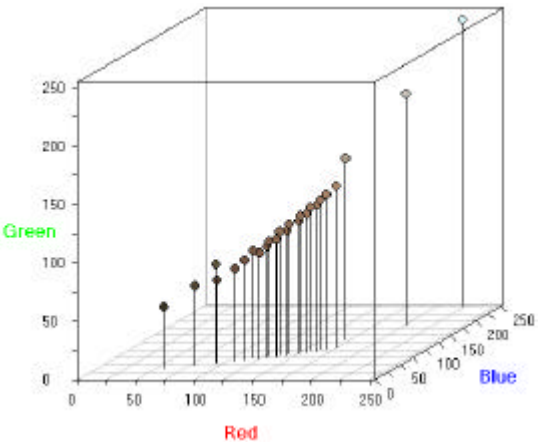


Abb. 8.10.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem quadratischen Standard-SOM bei 25 vorgegebenen Klassen und 4000 Lernzyklen

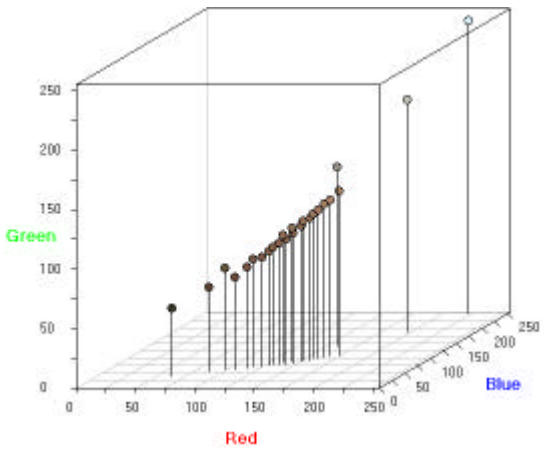


Abb. 8.11.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem hexagonalen Standard-SOM bei 25 vorgegebenen Klassen und 500 Lernzyklen

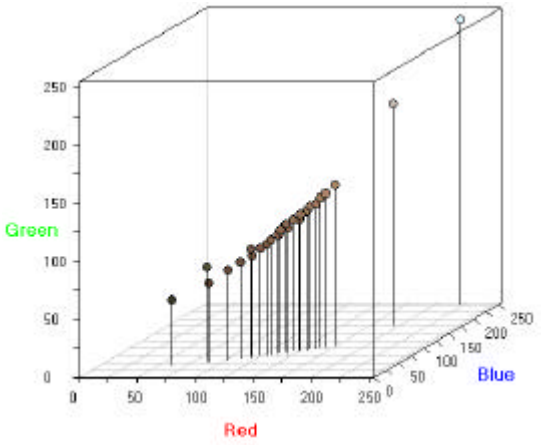


Abb. 8.11.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem hexagonalen Standard-SOM bei 25 vorgegebenen Klassen und 1000 Lernzyklen

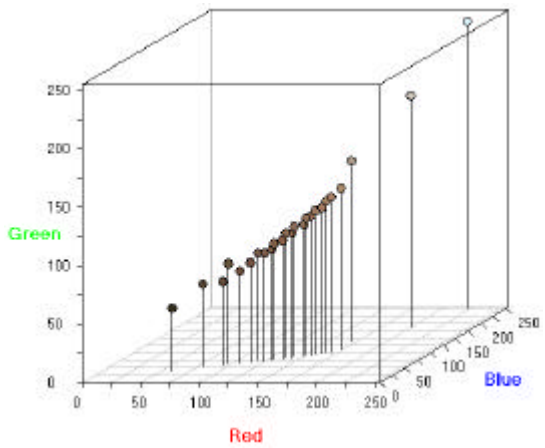


Abb. 8.11.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem hexagonalen Standard-SOM bei 25 vorgegebenen Klassen und 2000 Lernzyklen

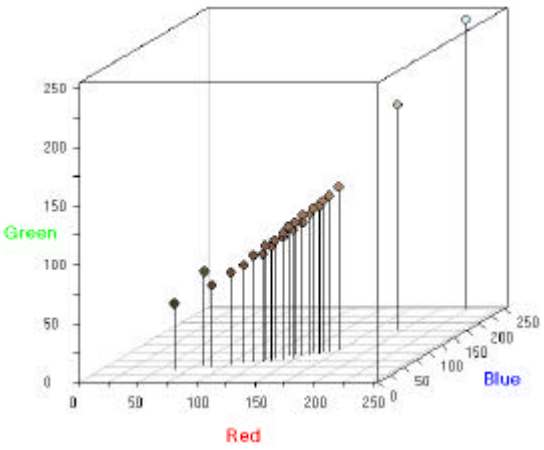


Abb. 8.11.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem hexagonalen Standard-SOM bei 25 vorgegebenen Klassen und 2000 Lernzyklen

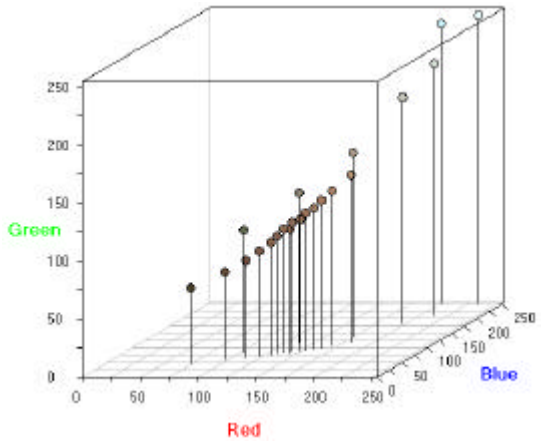


Abb. 8.12.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 5 Lernzyklen

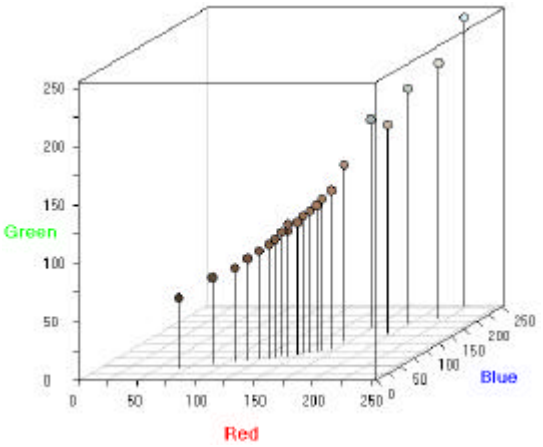


Abb. 8.12.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 10 Lernzyklen

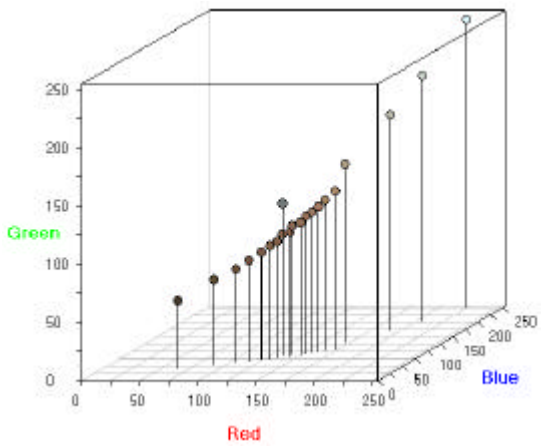


Abb. 8.12.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 20 Lernzyklen

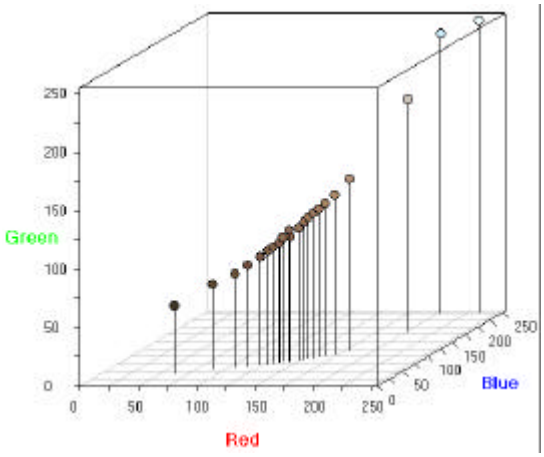


Abb. 8.12.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 40 Lernzyklen

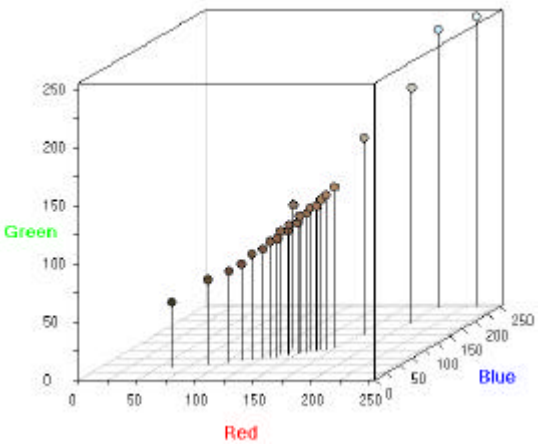


Abb. 8.12.e: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 80 Lernzyklen

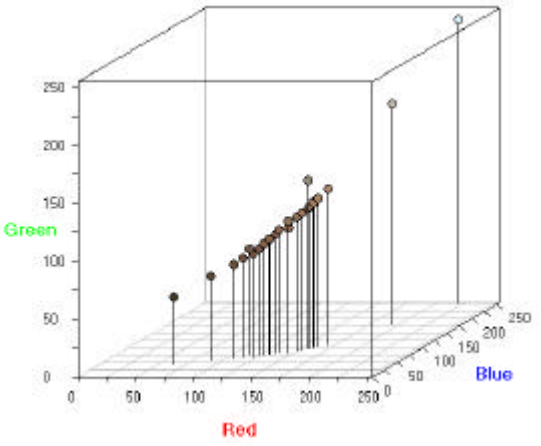


Abb. 8.12.f: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, quadratischen SOM bei 25 vorgegebenen Klassen und 160 Lernzyklen

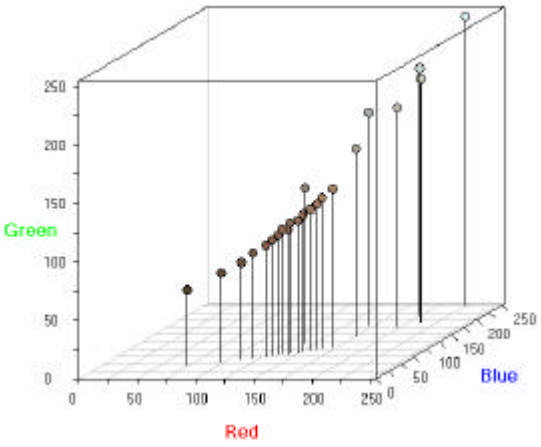


Abb. 8.13.a: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 5 Lernzyklen

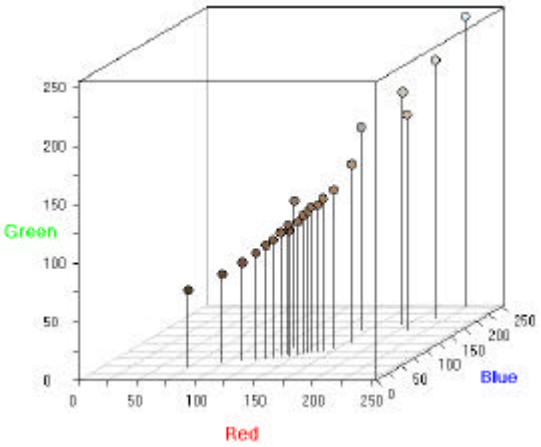


Abb. 8.13.b: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 10 Lernzyklen

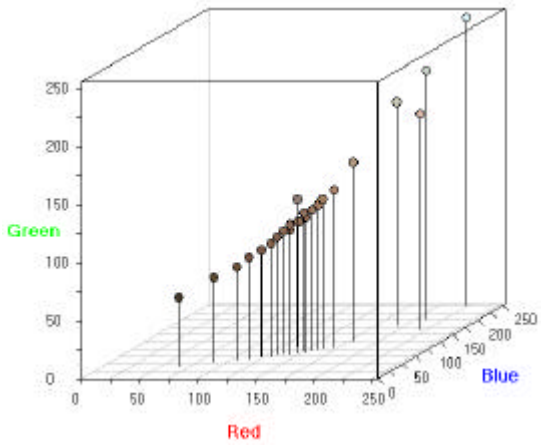


Abb. 8.13.c: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 20 Lernzyklen

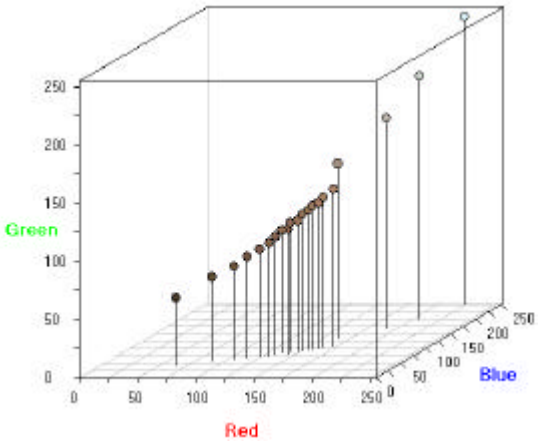


Abb. 8.13.d: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 40 Lernzyklen

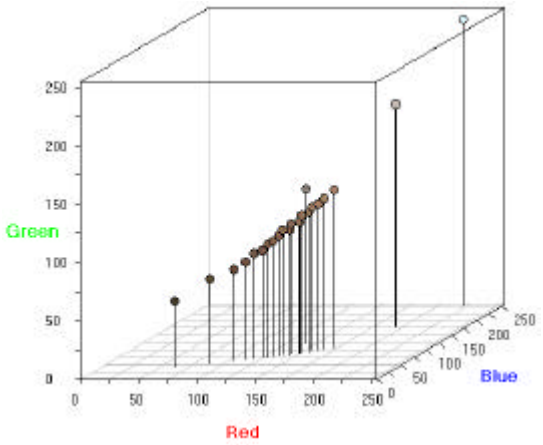


Abb. 8.13.e: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 80 Lernzyklen

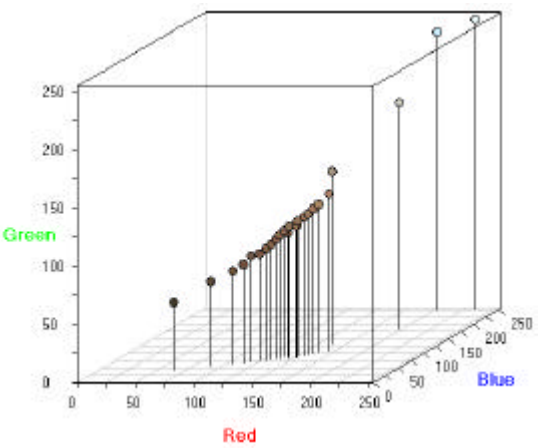


Abb. 8.13.f: Klassenrepräsentanten im Merkmalsraum nach der Clusterung von Abb. 6.3 mit dem schnellen, hexagonalen SOM bei 25 vorgegebenen Klassen und 160 Lernzyklen

Anhang C: PicAna-Dateilisting

Anhang C enthält die Auflistung aller Klassendateien des C++ Programms PicAna. Das Programm stellt die Praxisumsetzung der vorgestellten Methoden dar. Alle in dieser Arbeit vorgestellten Clusterungsergebnisse und die darauf basierenden Darstellungen wurden mit Hilfe dieser Programmumsetzung erzeugt. Das Programm ist in drei Hauptgruppen gegliedert: Algorithmen, Basis und grafische Benutzeroberfläche (GUI). Die Auflistung der Klassendateien erfolgt geordnet nach diesen drei Gruppen.

Algorithmen

- CLUSTERINFO.H
  CLUSTERINFO.CPP
- CLUSTERTABLE.H
  CLUSTERTABLE.CPP
- COLOUR.H
  COLOUR.CPP
- COLOUR_LESS.H
  COLOUR_LESS.CPP
- COLOURFREQPAIR.H
  COLOURFREQPAIR.CPP
- COLOURINFO.H
  COLOURINFO.CPP
- COLOURTABLE.H
  COLOURTABLE.CPP
- DOUBLE_COLOUR.H
  DOUBLE_COLOUR.CPP
- MAXIMUMLINKAGE.H
  MAXIMUMLINKAGE.CPP
- MAXLINKPROGRESSDATA.H
  MAXLINKPROGRESSDATA.CPP
- NEURONET.H
  NEURONET.CPP
- NEURONETPROGRESSDATA.H
  NEURONETPROGRESSDATA.CPP
- PICANALYSER.H
  PICANALYSER.CPP
- PICANAMETHOD.H
  PICANAMETHOD.CPP
- PICANAMETHODPROGRESSDATA.H
  PICANAMETHODPROGRESSDATA.CPP
- PICTURE.H
  PICTURE.CPP

- PROGRESSDATA.H
  PROGRESSDATA.CPP
- PROGRESSMESSAGE.H
  PROGRESSMESSAGE.CPP
- SELFORGNET.H
  SELFORGNET.CPP

Basis

- ANALYSIS.H
  ANALYSIS.CPP
- COLOUR_ZLESS.H
  COLOUR_ZLESS.CPP
- COLOURLIST.H
  COLOURLIST.CPP
- EVENTS.H
- MAXIMUMLINKAGETHREAD.H
  MAXIMUMLINKAGETHREAD.CPP
- MAXLINKTHRPROGRESSMSG.H
  MAXLINKTHRPROGRESSMSG.CPP
- METHOD.H
  METHOD.CPP
- NEURONETTHREAD.H
  NEURONETTHREAD.CPP
- NEURONETTHRPROGRESSMSG.H
  NEURONETTHRPROGRESSMSG.CPP
- PROJECT.H
  PROJECT.CPP
- SOURCEPIC.H
  SOURCEPIC.CPP
- THREAD.H

GUI

- COL3D_PANEL.H
  COL3D_PANEL.CPP
- COL3D_VIEW.H
  COL3D_VIEW.CPP
- COL3D_WIN.H
  COL3D_WIN.CPP
- FUNC.H
  FUNC.CPP

- MAIN_WIN.H
  MAIN_WIN.CPP
- MAXLINK_CTRL_DLG.H
  MAXLINK_CTRL_DLG.CPP
- MAXLINK_PGBAR_DLG.H
  MAXLINK_PGBAR_DLG.CPP
- NET_PANEL.H
  NET_PANEL.CPP
- NET_VIEW.H
  NET_VIEW.CPP
- NET_WIN.H
  NET_WIN.CPP
- NEURONET_CTRL_DLG.H
  NEURONET_CTRL_DLG.CPP
- NEURONET_PGBAR_DLG.HJ
  NEURONET_PGBAR_DLG.CPP
- PIC_PANEL.H
  PIC_PANEL.CPP
- PIC_WIN.H
  PIC_WIN.CPP
- PROGRESS_BAR_DLG.H
  PROGRESS_BAR_DLG.CPP
- PROJECT_INFO.H
  PROJECT_INFO.CPP
- PROJECT_LEFT.H
  PROJECT_LEFT.CPP
- PROJECT_LIST.H
  PROJECT_LIST.CPP
- PROJECT_RIGHT.H
  PROJECT_RIGHT.CPP
- PROJECT_WIN.H
  PROJECT_WIN.CPP
- TAB_WIN.H
  TAB_WIN.CPP

Literaturverzeichnis

Anderberg, M. R. (1973): *Cluster Analysis for Applications*. Academic Press, New York.

Anderson und Rosenfeld (1988): *Neurocomputing: Fundations of Research*, MIT Press, Cambridge, Massachusetts.

Banzhaf, J. (1989): *Auswirkungen von Windschutzstreifen aus Brachlandvegetation auf Wachstum und Ertragsbildung von Perlhirse (Pennisetum americanum (L.) Leeke) und Cowpea (Vigna unguiculata (L.) Walp.) im südlichen Sahel Westafrikas*, Dissertation, Universität Hohenheim.

Bock, H.-H. (1974): *Automatische Klassifikation.* Vandenhoeck & Ruprecht, Göttingen.

Bock, H.-H. (1998): *Clustering and neural networks.* In: Rizzi, A.; Vichi, M.; Bock, H.-H. (eds): *Advances in data science and classification*. Springer, Heidelberg, 265-278.

Braun, H. (1997): *Neuronale Netze, Optimierung durch Lernen und Evolution.* Springer-Verlag, Berlin.

Buerkert, A.; Mahler, F.; Marschner, H. (1996): *Soil productivity management and plant growth in the Sahel: Potential of an aerial monitoring technique.* Plant and Soil, 180, 29-38.

Cass, R. (1999): *The BitMap Format*, Multimedia Teaching Strategies. MTS Project Team, http://www.cs.mun.ca/k12media/resources.formats.graphics.bitmaps

Chou, Y.-H. (1997): *Exploring Spatial Analysis in Geographic Information Systems.* OnWord Press, Santa Fe, New Mexico.

Cressie, N.A. (1993): *Statistics for spatial data.* Revised Edition Wiley, New York, New York.

Everitt, B. (1980): *Cluster Analysis*. 2nd Edition, John Wiley & Sons, New York

Fahrmeir, L.; Brachinger H. W. (1996): *Multivariate statistische Verfahren***.** de Gruyter, Berlin.

Friedman, J.H. (1987): *Exploratory Projection Pursuit*. Journal of the American Statistical Association, 82, 249-266

Gérad, B.; Buerkert, A.; Hiernaux, P.; Marschner, H. (1997): *Non-destructive measurement of plant growth and nitrogen status of pearl millet with low-altitude aerial photgraphy.* Soil Science and Plant Nutrition 43, 993-998.

Gérad, B.; Buerkert, A. (1999): *Aerial photography to determine fertiliser effects on pearl millet and Guiera senegalensis growth.* Plant and Soil 210, 167-178.

Gibbons, J.D.; Chakraborti, S. (1992): *Nonparametric Statistical Inference.* 3rd Edition, revised and expanded, Maral Dekker, New York, New York.

Güting, R. H. (1992): *Datenstrukturen und Algorithmen*, Teubner Verlag, Stuttgart.

Guimarães, G.; Urfer, W.(2000): *Self-Organizing Maps and its applications in sleep apnea research and molecular genetics.* Technical Report 23/2000, SFB 475, Department of Statistics, University of Dortmund.

Hartigan, J.A. (1975): *Clustering algorithms.* Wiley, New York, New York.

Hartung, J.; Elpelt B.; Klösener, K.-H. (1989): *Statistik Lehr- und Handbuch der angewandten Statistik, 8. Auflage*. R.Oldenbourg, München.

Hebb, D. (1949): *Organisation of Behaviour.* Wiley, New York

Hopfield, J. (1982): *„Neural Networks and physical systems with emergent collective computational abilities“*, Proceedings of the National Academy of Science, 2554-2558

Jiang, X. (1997): *Effiziente Mustererkennung durch spezielle neuronale Netze.* Dissertation, FB Elektrotechnik, Universität der Bundeswehr, Hamburg.

Johnson, R.A.; Wichern, D.W. (1992): *Applied multivariate statistical analysis.* 3rd-Edition, Prentice Hall, Engelwood Cliffs, California.

Kaballo, W. (1997): *Einführung in die Analysis II.* Spektrum Akademischer Verlag, Heidelberg.

Kelly, P.; White, J. (1993): *Preprocessing remotely-sensed data for efficient analysis and classification. Application of artifical intelligence.* Proceedings, SPIE 1993, Knowledge Based Systems in Aerospace and Industry, 24-30.

Kötting, J.;Bonney, G.E.; Urfer, W. (1998): *Disposition models for the analysis of dynamic changes in forest-ecosystems.* Technical Report 25/1998, SFB 475, Department of Statistics, University of Dortmund.

Kohonen, T. (1982): *Self-organizing formation of topologically correct feature maps.* Biological Cybernetics, 43, 59-69.

Kohonen, T.; Kangas J.; Laaksonen J. (1992): *SOM-PAK. The Self-organizing Map Program Package.* Report A30, Helsinki University of Technology, Faculty of Information Technology, Laboratory of Computer and Information Science, Espoo.

Kohonen, T. (1995): *Self-organizing maps, 3rd Edition.* Springer, Heidelberg.

Kohonen, T.; Hynninen J.; Kangas J.; Laaksonen J.; Torkkola K. (1996): *LVQ-PAK. The Learning Vector Quantization Program Package.* Report A31, Helsinki University of Technology, Faculty of Information Technology, Laboratory of Computer and Information Science, Espoo.

Matecki, U. (1999): *Automatische Merkmalsauswahl für Neuronale Netze mit Anwendung in der pixelbasierten Klassifikation von Bildern*. Dissertation, FB Mathe / Informatik, Universität Osnabrück, Shaker Verlag, Aachen.

Minski, M. (1985): *The Society of Mind*, Simon and Schuster, New York.

Myers, W.; Patil, G.P.; Taillie, C. (1997a): *PHASE formulation of synoptic multivariate landscape data.* Technical report Number 97-1102. Center for statistical ecology and enviromental statistics, Department of Statistics, PennState University, Pennsylvania.

Myers, W.; Patil, G.P.; Taillie, C. (1997b): *Adaptimg quantitatiove multivariate geographic informaion system data for purposes of sample design: The PHASE approach.* Technical report Number 97-1201, Center for statistical ecology and enviromental statistics, Department of Statistics, PennState University, Pennsylvania.

NASA (1999): *Landsat 7 Science Data Users Handbook.*
url: http:// ltpwww.gsfc.nasa.gov / IAS / handbook / handbook_toc.html.

Niemann (1983): *Klassifikation von Mustern*. Springer-Verlag, Berlin.

Reade Ch. (1991): *Elements of functional programming*. Addison-Wesley, Wokingham

Ritter, H.; Martinez, Th.; Schulten, K. (1991): *Neuronale Netze, eine Einführung in die Neuroinformatik selbstorganisierender neuronaler Netze.* 2. Erweiterte Auflage, Addison-Wesley, Bonn.

Rojas, R. (1993): *Theorie der neuronalen Netze – Eine systematische Einführung.* Springer-Verlag, Berlin.

Rosenblatt, F. (1958): *The perceptron: a probabilistic model for information storgae and organization in the brain*, Psychological Review, Nachgedruckt in Anderson und Rosenfeld (1988**)**

Schmidt, J.W. (1983): *Relational database systems.* Springer-Verlag, Berlin.

Talbi, M. (1993): *Contribution á l'etude de la désertification par Télédétection dans la Jeffara: Sud est de la Tunisie.* These de Doctorat en Géographie. Université de Tunis-I.

Talbi, M. (1999): *Spectral signature for desertification pattern recognition in arid environment, using remote sensing.* Technical paper, Institut des Régions Arides. Tunisia.

Tou, J.T.; Gonzalez, R.C. (1974): *Pattern recognition principles.* Addison-Wesley, Reading, Mass.

Tschiersch, L. (1998): *Auswertung von zeitabhängigen Daten von Eroions-Dauerbeobachtungsflächen im Wassereinzugsgebiet von Rabat.* Fachbereich Statistik, Universität Dortmund, Dortmund.

UNCCD (1994): *United Nations Convention to Combat Desertification.* url: http://www.unccd.int/convention/menu.php

Ultsch, A. G. H.; Siemon, H. P. (1989): *Exploratory data analysis using Kohonen networks on transputers*, Technical Report 329, Fachbereich Informatik, Universität Dortmund, Dortmund.

Urfer, W.; Schwarzenbach, J.; Kötting, J.; Müller, P. (1994): *Multistate models for monitoring individual trees in permanent oberservation plots.* Environmental and Ecological Statistics 1, 171-199.

Urfer, W. (2001): *Statistical tools for extracting information from DNA sequence data.* In: Kunert, J.; Trenkler, G. (eds.): *Mathematical Statistics with Applications in Biometry: Festschrift in Honour of Siegfried Schach*. Eul Verlag, Lohmar, 103-112.

Wyszecki, G.; Stiles, W.S. (1982): *Color science, concepts and methods. Quantitative data and formula.* John Wiley, New York.

Zell, A. (2000): *Simulation neuronaler Netze.* 3. Unveränderter Nachdruck, R.Oldenburg, München.

Zerbst, M. (1999): *Untersuchung von Einflußgrößen der Erosion in Waldgebieten Marokkos.* Fachbereich Statistik, Universität Dortmund, Dortmund.

Zerbst, M.; Tschiersch, L.; Guimarães, G.; Talbi, M.; Urfer, U. (2001): On clustering of aerial photographs and high resolution satellite images. Eingereicht bei ENVIRONMETRICS, Canada.

Ehrenwörtliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Dortmund, den 05.11.2001 Matthias Zerbst