

# **Neuronale Netzwerke und Clusteringverfahren zur Analyse von Geodaten**

Lehrstuhl für Geoinformatik

Robin Bially

[HTTPS://GITHUB.COM/ROBINBIA/PROJEKTARBEIT-GEOINFORMATIK.GIT](https://github.com/ROBINBIA/PROJEKTARBEIT-GEOINFORMATIK.GIT)

Projektarbeit unter der Betreuung von PD Dr. Dr.-Ing. Wilfried Linder von 11.2017 - 11.2018  
als Vorbereitung der sich anschließenden Masterarbeit.

*Fertigstellung, November 2018*



# Inhaltsverzeichnis

<b>1</b>	<b>Motivation</b>	5
<b>2</b>	<b>Geodaten und Geoinformation</b>	7
2.1	<b>Definition und Gestalt von Geodaten</b>	7
2.2	<b>Geografische Koordinaten</b>	8
2.3	<b>Qualitätsmerkmale</b>	8
2.4	<b>Georeferenzierung</b>	10
2.5	<b>Geoinformationssysteme</b>	11
2.5.1	Geoobjekte	12
2.5.2	Rastermodell	12
2.5.3	Vektormodell	13
2.5.4	Beispiele von Raster und Vektordaten	15
2.6	<b>Algorithmen in der Geoinformatik</b>	16
2.7	<b>Verschiedene Arten und ihre Anwendungszwecke</b>	16
<b>3</b>	<b>Deep Learning</b>	17
3.1	<b>Was ist Machine Learning?</b>	17
3.2	<b>Motivation und Anwendungsgebiete</b>	17
3.2.1	Linear Classifier	20
3.2.2	Regularisierer	21
3.2.3	Gradientenabstieg	22
3.2.4	Multi-Layer Neural Network	22
3.2.5	Trainieren eines Netzwerks mittels Backpropagation	22

3.2.6	Parameterupdates . . . . .	24
3.2.7	Convolutional Neural Network . . . . .	25
3.2.8	Recurrent Neural Network . . . . .	27
<b>3.3</b>	<b>Machine Learning in Geowissenschaften</b>	<b>29</b>
3.3.1	Herausforderungen und Chancen von Machine Learning . . . . .	29
<b>4</b>	<b>Clusteringverfahren . . . . .</b>	<b>35</b>
<b>4.1</b>	<b>Definition und Anwendungszweck</b>	<b>35</b>
4.1.1	Die fünf Hauptschritte des Datenclustering . . . . .	37
<b>4.2</b>	<b>Crisp Clustering</b>	<b>37</b>
4.2.1	Partitionierende Verfahren und ihre Nachteile . . . . .	37
4.2.2	Silhouettenkoeffizient - Wahl der Clusteranzahl . . . . .	39
4.2.3	OPTICS - Dichtebasiertes Clustering . . . . .	40
4.2.4	Outlier Detection - Local Outlier Factor . . . . .	48
4.2.5	Lineare Separation - Support Vector Machine . . . . .	48
<b>4.3</b>	<b>Fuzzy-Clustering</b>	<b>52</b>
4.3.1	Fuzzy C-Means (FCM) . . . . .	52
4.3.2	Possibilistic C-Means (PCM) . . . . .	54
4.3.3	Possibilistic Fuzzy C-Means (PFCM) . . . . .	56
4.3.4	Scale Space Filter Clustering . . . . .	56
4.3.5	Graphbasiertes Clustering I (NSCABDT) . . . . .	58
4.3.6	Graphbasiertes Clustering II (NSFCDT) . . . . .	62
4.3.7	Kontextbasiertes Clustering . . . . .	66
<b>4.4</b>	<b>Finden der Clusteranzahl</b>	<b>66</b>
4.4.1	NPC . . . . .	66
4.4.2	FHV . . . . .	66
4.4.3	Otsu-Binarisierung . . . . .	66
4.4.4	VAT-Algorithmus . . . . .	66
<b>4.5</b>	<b>Clustering auf unvollständigen Daten</b>	<b>66</b>
<b>4.6</b>	<b>Assoziationsregeln</b>	<b>66</b>
<b>5</b>	<b>Implementierungen . . . . .</b>	<b>67</b>
<b>6</b>	<b>Fazit und Ausblick . . . . .</b>	<b>69</b>
<b>6.1</b>	<b>Ausblick - Mein Thema für die Masterarbeit</b>	<b>69</b>



## 1. Motivation

Die folgende Projektarbeit liefert einen groben Überblick über Methoden und Algorithmen im Bereich der geoinformationsverarbeitenden Systeme mit dem Schwerpunkt Machine Learning (ML). Darunter werden sämtliche Verfahren verstanden, die auf Basis von Datenbeständen Fähigkeiten erlernen, mithilfe derer abstraktere Fragenstellungen und Probleme automatisiert gelöst werden können. Das während des Lernprozesses angehäufte Wissen hilft dem Algorithmus dabei, bezüglich einer mathematischen Vorschrift immer bessere Lösungen zu finden. Die Projektarbeit beschränkt sich auf die ML-Teilbereiche Deep Learning und Data Mining, welche Hauptsächlich im Bereich der Wissensextraktion (Assoziationsmustererkennung), Klassifikation und Clustering eingesetzt werden. Die Auswertung geowissenschaftlicher Fachliteratur soll Erkenntnisse über die Einsatzbereiche von Machine Learning liefern. Außerdem soll recherchiert werden, worin die Herausforderungen bestehen und welche offenen Probleme es gibt.

Ziel der Arbeit ist die Eingrenzung der Themenbereiche auf Fragestellungen, die sich hinsichtlich einer wissenschaftlichen Ausarbeitung im Rahmen der sich anschließenden Masterarbeit eignen.





## 2. Geodaten und Geoinformation

Kapitel wahrscheinlich irrelevant und häufig nur eine Aneinanderreihung zusammenhangsloser Definitionen (Ausnahme 2.3). löschen?

### 2.1 Definition und Gestalt von Geodaten

Geodaten sind digitale Informationen, welche Sachdaten mit Geometriedaten<sup>1</sup> (und Chronometriedaten) vereinen , z.B. {Luftdruck 1 bar, Ort Düsseldorf, Datum 26.11.2017}. Die räumliche Information kann in unterschiedlichen Formen vorliegen, z.B. symbolisch als Ortsname oder Postleitzahl, aber auch als mathematisch atomare Referenz auf Positionen der Erde mittels Koordinaten. Diese können in unterschiedlichster Dimensionalität vorliegen<sup>2</sup>

- Ein Objekt ohne bestimmte Länge (0D)
- Ein Linienstück (1D)
- Gauß-Krüger oder geografische Koordinaten mit Bezug auf die Oberfläche der Erde ohne Berücksichtigung von Höhenunterschieden (2D)
- 2D-Koordinaten mit einer zusätzlichen Sachinformation für die Höhe über dem Geoiden (2.5D).
- Kugelkoordinaten mit Bezug auf jeden Punkt im Volumen der Erde als Geoid oder Rotationsellipsoid (3D)
- Zusätzlich zu den 3 Koordinaten im Raum wird eine vierte Information mitgeführt, die sich aus dem zeitlichen Ablauf ergibt (4D)

---

<sup>1</sup>[https://www.hdm-stuttgart.de/riekert/lehre\\_gis.pdf](https://www.hdm-stuttgart.de/riekert/lehre_gis.pdf)

<sup>2</sup><http://www.mathematik.uni-ulm.de/sai/ws04/biosem/GIS.pdf>

## 2.2 Geografische Koordinaten

Ein geeignetes und weit verbreitetes Koordinatensystem zur verzerrungsarmen Darstellung sind die Geografischen Koordinaten.

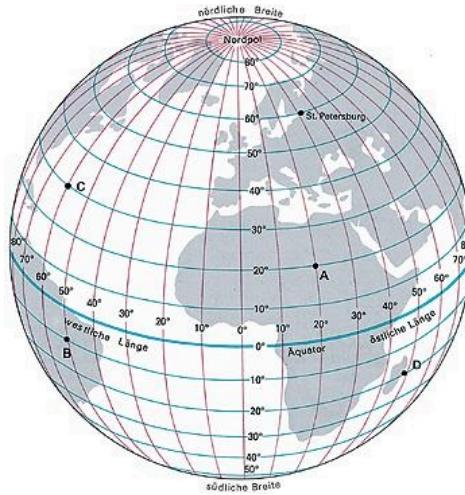


Abbildung 2.1: Das Gradnetz der Erde

Beschrieben wird ein Punkt auf der Erde durch gedachte Kreise um den Globus, welche senkrecht zueinander stehen. Insgesamt existieren 180 Breitenkreise (Richtung Ost-West) und 360 Längenkreise (Richtung Nord-Süd). Die Abweichung von den beiden Referenzkreisen Äquator und Nullmeridian wird in Grad östlicher/westlicher Länge und nördlicher/südlicher Breite angegeben. Als Äquator ( $0^\circ$  nördliche/südliche Breite) wird der Breitenkreis bezeichnet, auf welchem die Erdachse senkrecht steht. Der Nullmeridian ( $0^\circ$  westliche/östliche Länge) ist der Längenkreis, welcher durch die britische Stadt Greenwich verläuft.

Weitere wichtige Koordinatensysteme sind die Gauß-Krüger und UTM-Koordinaten. Die Vorteile dieser Systeme ist, dass sich eine geografische Position direkt ablesen lässt.

Geografische Koordinaten erschweren dies bedingt durch die sich verändernden Abstände zwischen den Längenkreisen in zunehmender Nord- oder Südrichtung.

## 2.3 Qualitätsmerkmale

Ein wichtiger Forschungszweig ist die automatische Beurteilung von Qualitätsmerkmalen von Geodaten hinsichtlich einer bestimmten Fragestellung. Ein geeignetes Maß ist die gewichtete Summe verschiedener Datenmerkmale, welche in der aktuellen ISO-Norm *ISO 19157:2013*<sup>3</sup> spezifiziert sind.

Die Beurteilung der Datenqualität ist ebenfalls in sämtlichen Verarbeitungsschritten des KDD-Prozessmodells [FPS96] erforderlich. Das KDD-(Knowledge Discovery in Databases-)Prozessmodell beschreibt, wie aus einer unstrukturierten Datensammlung neues, relevantes Wissen extrahiert werden kann. Es umfasst folgende Schritte:

<sup>3</sup><https://www.iso.org/standard/32575.html>

1. **Fokussieren/Selektieren** Die Daten werden zusammengetragen, in Form einer Datenbank strukturiert und irrelevante Einträge gelöscht. Relevante Daten haben das Potenzial hinsichtlich eines vorgegebenen Ziels vielversprechende Erkenntnisse zu liefern.
2. **Vorverarbeitung** Es wird überprüft, ob die Datenbank konsistent ist und ob es fehlende Einträge gibt. Die Konsistenz wird verletzt, wenn zum Beispiel einer Koordinate ein Land zugeordnet wird, welches den umgebenen Datenpunkten nicht zugeordnet wurde. Die Wahrscheinlichkeit einer Fehlzuordnung ist dann groß. Die Datenbank hat fehlende Einträge, wenn es Attribute mit nicht existierenden Werten gibt. Strategien zum Umgang mit fehlerhaften Datensätzen beim Clustering werden im Kapitel über Clusteringverfahren erläutert.
3. **Transformation** Wurde ein komplexes Objekt in die Datensammlung aufgenommen, so müssen seine Merkmale zuerst numerisch diskretisiert werden. Beispielsweise hat ein Reisfeld eine Länge und Breite, aber auch eine Vegetationsdichte und einen Erntestatus. All diese Merkmale müssen vermessen und in die Datenbank aufgenommen werden. Merkmale, welche nach der Transformation in mangelhafter Qualität vorliegen, können entfernt werden.
4. **Data Mining** Methoden zur Mustererkennung. Dazu gehören sowohl Neuronale Netze und Clusteringverfahren, als auch Assoziationsregeln. In folgenden Kapiteln werden diese erklärt.
5. **Evaluation** Ergebnisse der Mustererkennung werden statistisch überprüft und die Nützlichkeit vom Nutzer bewertet.

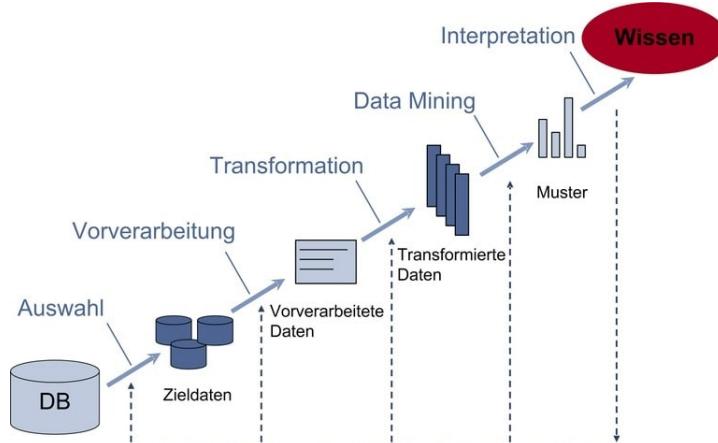


Abbildung 2.2: Das KDD-Prozessmodell <sup>4</sup>

Die folgende Auflistung ist eine informelle Beschreibung der oben genannten ISO-Norm *ISO 19157:2013* durch Fragestellungen und Beispiele:

- **Vollständigkeit**

<sup>4</sup><http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Business-Intelligence/Analytische-Informationssysteme--Methoden-der-/Data-Mining/index.html>

- **Datenüberschuss** - Enthält der Datensatz mehr Objekte und Beziehungen als angegeben?
- **Datenmangel** - Enthält der Datensatz weniger Objekte und Beziehungen als angegeben?

- **Logische Konsistenz**

- **Konzeptuelle Konsistenz** - Wurde die Gestalt des Datenmodells bei Aktualisierungen nicht verändert?
- **Wertekonsistenz** - Sind alle Werte sinnvoll?
- **Formatkonsistenz** Passen die Daten zu angegebenen physikalischen Einheiten?
- **Topologische Konsistenz** Bleiben topologische Beziehungen bei Änderungen des Datensatzes bestehen (Der botanische Garten befindet sich im Umkreis von 1km von der HHU)?
- **Geometrische Konsistenz** - Ist der digitalisierte Datensatz geometrisch sinnvoll und widerspruchsfrei?

- **Positionsgenauigkeit**

- **Außere Genauigkeit** - Wie gut stimmen die Koordinatenwerte des Datensatzes mit den wahren Koordinaten überein?
- **Innere Genauigkeit** - Wie gut stimmen die relativen Positionen von Objekten zueinander mit den wahren relativen Positionen überein?
- **Rasterdatengenauigkeit** - Wie gut stimmen die Rasterdatenpositionsvalenzen mit den wahren Werten überein?

- **Zeitliche Genauigkeit**

- **Genauigkeit von Zeitmessungen** - Wie genau ist die Zeitangabe (minutengenau, taggenau)?
- **Zeitliche Konsistenz** - Ist die Reihenfolge der Ereignisse korrekt?
- **Zeitliche Gültigkeit** - Ist der Datensatz in Bezug auf das geforderte Zeitformat korrekt?

- **Thematische Genauigkeit**

- **Richtigkeit der Klassifikation** - Stimmen Objekte, oder ihre Attribute mit den zugewiesenen Klassen überein, z. B. Zuordnung zu Fluss, statt zu Weg
- **Richtigkeit nichtquantitativer Attribute** - Beispiel: Ist das Grundstück wirklich eine Bananenplantage?
- **Genauigkeit quantitativer Attribute** - Beispiel: Ist die Fläche des Grundstücks korrekt?

Viele der oben genannten Punkte lassen einen subjektiven Spielraum für die Bewertung zu. Sowohl Skalierungen als auch Gewichtungen sind nicht eindeutig definiert, was einen Vergleich verschiedener Datensätze erschwert. Aus diesem Grund ist eine algorithmische Interpretation in Kombination mit verfahren der künstlichen Intelligenz hilfreich. So ließe sich aus der Norm ein universeller und allgemeingültiger Indikator zur Bewertung der Datenqualität ermitteln.

## 2.4 Georeferenzierung

### Definition

Unter dem Vorgang der Georeferenzierung versteht man die Zuweisung raumbezogener Informationen, auch Georeferenz genannt, zu einem Datensatz.

Es gibt folgende vier Arten der Georeferenzierung:

- Bei der **Adresskodierung** wird dem Datensatz eine Postanschrift zugewiesen und somit ein indirekter Raumbezug hergestellt. Mithilfe geokodierter Adressen lassen sich funktionale Zusammenhänge zwischen Daten, Postanschrift und Adresse herstellen und somit ressourcenschonende und schnelle Zugriffe ermöglichen.
- Als **Geotagging** bezeichnet man das Einfügen eines Attributes (Geotag) inkl. Realweltkoordinate in einen raumbezogenen Datensatz wie ein Bild oder eine Website. Dies ist bei der räumlichen Einordnung der Information hilfreich.
- Bei der **Kartenkalibrierung** wird ein räumlicher Datensatz ohne Realkoordinatenbezug mithilfe einer Transformationsvorschrift im Bezug auf die Realwelt so orientiert, dass sich die Koordinaten des Bildes in Realweltkoordinaten einfach umrechnen lassen.
- Bei der **Rektifizierung** werden geometrische Verzerrungen in räumlichen Daten entzerrt, indem jedem Datum eine Realweltkoordinate zugeordnet wird.

#### **Bestimmung einer Transformationsvorschrift**

Um eine Transformationsgleichung zu finden, werden in der Regel Passpunkte verwendet. Die Passpunkte müssen im Datensatz eindeutig zu erkennen sein. Die Koordinaten der Passpunkte im Realweltkoordinatensystem sind entweder bekannt oder werden einem Referenzdatensatz entnommen. Bei Vektordaten werden die Koordinaten abgegriffen oder interpoliert. Bei Bilddaten werden die Bildkoordinaten der Passpunkte gemessen. Die Transformation sollte unter Berücksichtigung der Abbildungsgeometrie bestimmt werden. Bei Fotos ist somit die Zentralprojektion zu berücksichtigen, bei Karten der entsprechende Kartennetzentwurf. Das automatische Finden von Gemeinsamkeiten in digitalen Bildern und die Bestimmung der Transformation wird in der Bildverarbeitung Bildregistrierung genannt. Die Registrierung von Laserscanning-Punktwolken kann mit dem ICP-Algorithmus erfolgen.<sup>5</sup>

## **2.5 Geoinformationssysteme**

Ein Geoinformationssystem ist eine Software, mit welcher Geodaten erfasst, verwaltet, analysiert und ausgegeben werden können.

Man unterscheidet bei der Abfrage von Daten unter folgenden verschiedenen Typen:

- Alphanumerische Daten (Attribute als Text oder Zahlen)
- Text-Dokumente
- Multimediale Informationen, wie Videos, Audiosequenzen, Animationen
- Fotos, Scans, Satellitenbilder

Der Unterschied zu einer Datenbank ist, dass jedes Sachdatum einen expliziten Raumbezug hat, über welchen die Selektion erfolgt. In einer Datenbank erfolgen Zugriffe stattdessen über Schlüsselattribute. Eine weitere Stärke eines GIS ist die grafische Aufbereitung der Daten zur anschaulich-interaktiven Analyse.

Beispiele für solche räumlichen Analysewerkzeuge sind Routenfindung, räumliche Suche. Ein implementiertes Kartografiesystem ermöglicht zudem das markieren von Punkten und Linien, färben von Flächen und die Anzeige und Überlagerung verschiedener Ebenen.

---

<sup>5</sup><https://de.wikipedia.org/wiki/Georeferenzierung>

### 2.5.1 Geoobjekte

Ein Geoobjekt ist ein tatsächlich auf der Erde vorhandenes Objekt, welches durch Geodaten eindeutig referenziert wurde. Man unterscheidet zwischen Gegenständen und Sachverhalten. Gegenstände sind konkrete, visuell wahrnehmbare Erscheinungen auf der Erdoberfläche. Sachverhalte dagegen sind nicht sofort visuell wahrnehmbar, sondern bezeichnen Beziehungen zwischen Gegenständen oder die Interaktion mit der Umwelt und Oberflächengestalt. Außerdem unterscheidet man zwischen verschiedenen Arten der Datenspeicherung:

- Flächenhafte Daten
- Linienhafte Daten
- Punkthafte Daten

Je nach Kartenmaßstab, Auflösungstyp und Speichertyp (digital/analog) werden Daten unterschiedlich repräsentiert. So wird beispielsweise ein flächenhafter quadratischer Gebäudekomplex (10\*10 Meter) auf einem Satellitenfoto mit dem Maßstab 1:10.000 nur noch als Punkt wahrgenommen. Linienhafte Daten bieten sich vor allem bei Flüssen, Straßen, Wasser-Land-Grenzen, starken Flankensteigungen usw. an.

#### Modellierung von Geoobjekten

Die vier informationstechnischen Dimensionen zur Modellierung von geografischen Informationssystemen sind:

- Geometrie (Ort des Objekts)
- Topologie (Lage der Objekte relativ zueinander)
- Semantik (Bedeutung des Objekts im fachspezifischen Kontext, z.B. gut-schlecht, viel-wenig, groß-klein)
- Dynamik (Änderung des Objekts im zeitlichen Verlauf)

Jedes unikate Objekt gehört zu einer Objektklasse, in welcher es nach den oben genannten vier Kriterien beschrieben und mit anderen Objekten der Klasse verglichen wird. Jedes der Objekte besitzt einen eindeutigen Schlüssel zur Identifikation. Möglichkeiten zur Klassifizierung und Clustering von Geoobjekten werden in den folgenden Kapiteln vorgestellt.

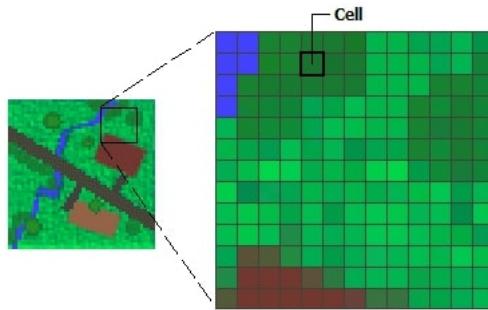
Der Ort eines Geoobjektes kann auf folgende zwei verschiedene Arten beschrieben werden.

### 2.5.2 Rastermodell

Eine analoge topografische Karte oder Zeichnung digitalisiert und in quadratische Gitterzellen aufgeteilt, welche alle über die gleiche Semantik verfügen. Diese Semantik wird stellvertretend durch eine Matrix beschrieben, welche für jede Gitterzelle eine numerische Pixelwertinformation enthält.<sup>6</sup>

---

<sup>6</sup><https://de.wikipedia.org/wiki/Geoobjekt>

Abbildung 2.3: Rastermodell-Zoom<sup>7</sup>

Diese Pixelinformationen repräsentieren Daten wie Temperatur, Höhe, Vegetationsdichte, Landnutzung, Bodenbeschaffenheit. Rasterdaten werden in der Regel als Bilddatei gespeichert (BMP, GIF, JPEG).

Die Rastergeometrie eignet sich gut zur Beschreibung flächiger, homogener Sachverhalte. Die einfache Struktur bietet viele Vorteile, aber auch Nachteile:

#### Vorteile

- Einfache Datenstruktur
- Geeignet für räumliche und statistische Analyse
- Alles ist einheitlich speicherbar (Punkte, Linien, Polygone)
- Überlagerung von Ebenen sehr schnell und einfach

#### Nachteile

- Genauigkeitsverlust beim Scannen und Neustrukturieren
- Endliche Auflösung => räumliche Ungenauigkeit
- Pixelwerte haben keine Beziehung zueinander
- Hoher Speicheraufwand bei hoher Auflösung, keine Kompression möglich.

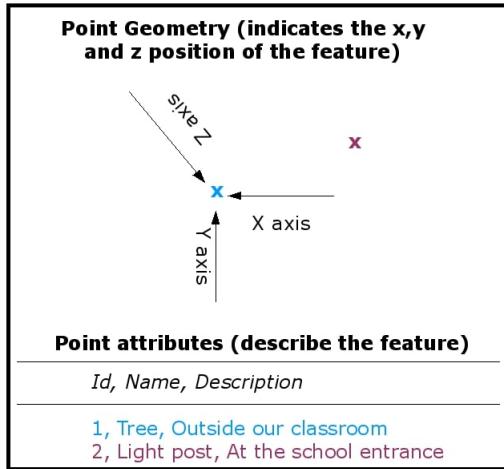
### 2.5.3 Vektormodell

Im Gegensatz zu Rasterdaten werden Vektordaten bei linien- und punkthaften Informationen eingesetzt, also Informationen, die sich nicht mit homogener Eigenschaft über die gesamte Karte verteilen. Man nennt solche Informationen auch Features. Beispiele hierfür sind Straßen, Staatsgrenzen, Gewässergrenzen, Höhenlinien, Flüsse, Bäume.

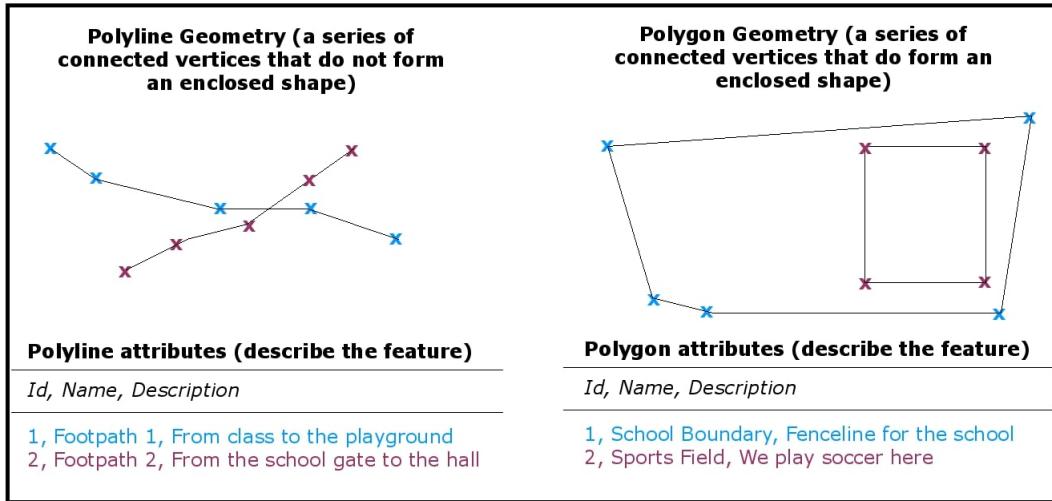
Eine punkthafte Vektorinformation wird auch als Vertex bezeichnet. Dieser beschreibt eine Raumlage durch Angabe einer (x,y,z)-Koordinate und ein dazugehöriges Attribut, welches die Art des Punktes beschreibt, z.B. Baum oder Laterne:

---

<sup>7</sup><http://desktop.arcgis.com/de/arcmap/10.3/manage-data/raster-and-images/what-is-raster-data.htm>

Abbildung 2.4: Punkt-Feature<sup>8</sup>

Punkteverläufe wie Straßen werden durch sogenannte Polylinien beschrieben. Diese bestehen aus mehreren miteinander verbundenen Vertices. Im Kreis laufende Polylinien bezeichnet man auch als Polygone:

Abbildung 2.5: Polylinien und Polygone<sup>9</sup>

Wie auch bei Rasterdaten gibt es bei Vektordaten nicht nur Vorteile, sondern auch Nachteile.<sup>10</sup>

#### Vorteile:

- Unendliche Linienauflösung und sehr hohe Genauigkeit
- Beschreibung von mehreren einzigartigen Features in nur einer Ebene möglich
- Geringer Speicherbedarf

<sup>8</sup>[https://docs.qgis.org/2.8/de/docs/gentle\\_gis\\_introduction/vector\\_data.html](https://docs.qgis.org/2.8/de/docs/gentle_gis_introduction/vector_data.html)

<sup>9</sup>[https://docs.qgis.org/2.8/de/docs/gentle\\_gis\\_introduction/vector\\_data.html](https://docs.qgis.org/2.8/de/docs/gentle_gis_introduction/vector_data.html)

<sup>10</sup><http://romanharcke.de/geoinformationssysteme-geodaten-kapitel-4/>

- Einfache Erzeugung von Topologie (Knoten, Kanten, Flächen)
- Gute Performance
- Ermöglicht Attributierung und Objektdefinitionen

#### Nachteile:

- Flächenhafte Informationen können nicht gespeichert werden
- Durch Scannen können diese Daten nicht erzeugt werden. Es bedarf hier einer Raster-Vektorwandlung (Hoher Erfassungsaufwand)
- Hoher Rechenaufwand bei Verschneidungen

#### 2.5.4 Beispiele von Raster und Vektordaten

Geodaten müssen heutzutage nicht mehr selbstständig erstellt werden. Es gibt eine Vielzahl an staatlichen und privaten Institutionen, welche Ihre Daten kostenlos bereitstellen. So lassen sich zahlreiche Inhalte im ESRI Shapefile Vektordateiformat finden, welches als Quasi-Standard für Desktop-GIS gilt.<sup>11</sup> Der Datensatz *Natural Earth*<sup>12</sup> ist eine Abbildung der Erde im Maßstab 1:10 Millionen. Er ist sowohl als SHP-Vektordatei als auch als Tiff-Rasterbild verfügbar.

Ein ESRI Shapefile besteht aus mindestens drei Dateien zur Speicherung der Geometriedaten, Sachdaten und der Geometrieindizierung zur Verknüpfung von Geometrie und Sachdaten (.shp, .dbf, .shx). Die Geometrie eines Shapefile definiert sich aus nur 4 verschiedenen Formdatenstrukturen: Punkte, Linien, Flächen (Polygone) und Multipunkte.<sup>13</sup>

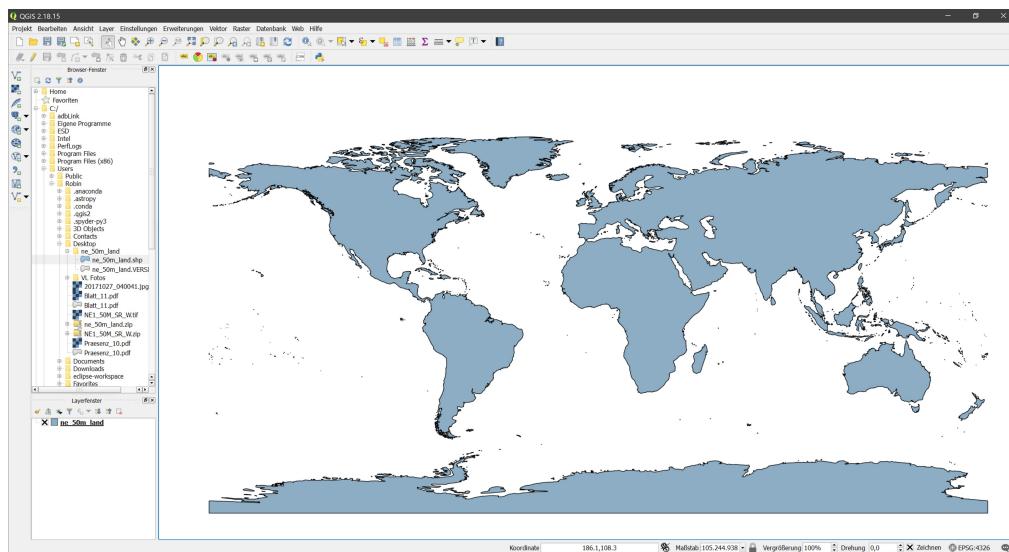


Abbildung 2.6: .shp-Geometriedatei in dem Geoinformationssystem QGIS dargestellt<sup>14</sup>

Leider eignen sich Vektordaten nicht zur Klassifikation von Features mit Hilfe von Deep

<sup>11</sup><http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Geoprocessing%20considerations%20for%20shapefile%20output>

<sup>12</sup><http://www.naturalearthdata.com/>

<sup>13</sup><http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>

<sup>14</sup><https://www.qgis.org/de/site/>

Learning wie z.B. Convolutional Neural Networks (CNN), sondern stellen viel mehr das Ergebnis einer Rasterbildanalyse dar. Aus diesem Grund beziehen sich folgende Kapitel im Kontext von Geodaten immer auf Rasterdaten und Bildausschnitte.

Das dem Datensatz zugehörige farbige Rasterbild inklusive Schummerung (räumliche Schattierung), Wasser und Flüssen sieht so aus:

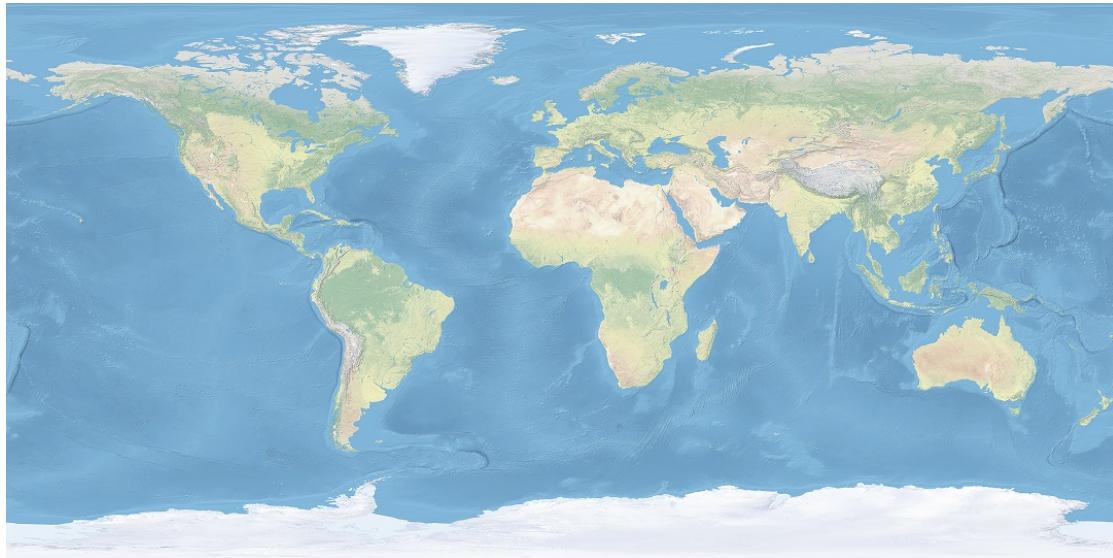


Abbildung 2.7: Rasterbild des Natural-Earth-Datensatzes<sup>15</sup>

## **2.6 Algorithmen in der Geoinformatik**

## **2.7 Verschiedene Arten und ihre Anwendungszwecke**

---

<sup>15</sup><http://www.naturalearthdata.com/downloads/10m-raster-data/10m-natural-earth-1/>



## 3. Deep Learning

### 3.1 Was ist Machine Learning?

#### Definition

Machine Learning ist eine Unterdisziplin der künstlichen Intelligenz und basiert auf der Idee, biologische Denkprozesse, wie sie in Gehirnen ablaufen, nachzuahmen [Lar+15].

*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E. [Mit97]*

Ein Computerprogramm lernt also genau dann dazu, wenn es sich hinsichtlich seiner Performance in bestimmten Aufgabengebieten mithilfe von Erfahrung selbstständig verbessert.

### 3.2 Motivation und Anwendungsgebiete

Ziel von Machine Learning in den Geowissenschaften ist es, Muster in Geodaten zu erkennen, Vorhersagen zu machen und Phänomene besser erkennen und verstehen zu können.

Eine Stadt ist ein Komplexes System, das aus vielen kleineren interagierenden Subsystemen besteht. Diese werden durch Faktoren wie Politik, Bevölkerungswachstum, Verkehrsinfrastruktur und den Arbeitsmarkt beeinflusst. Um zu verstehen, welche Kräfte strukturelle Änderungen von Städten vorantreiben, werden sowohl Satellitenbilder als auch nutzerbezogene Positionsdaten aus sozialen Netzwerken wie Facebook und Twitter und Attributierte Markierungen auf Geoinformationssystemen wie OpenStreetMap<sup>1</sup> verwendet, um Langzeitvorhersagen zur erstellen. Außerdem helfen diese Modelle und Simulationen dabei, die Mechanismen der urbanen Evolution zu erforschen und Städteplanung zu optimieren.

---

<sup>1</sup><https://www.openstreetmap.org>

Im Folgenden eine Auflistung verschiedener Probleme, für deren Lösung sich die Anwendung eines Neuronalen Netzwerks eignet:

- Klassifizierung - Was ist auf einem Bild zu sehen?
- Lokalisation - Wo ist das Objekt auf dem Bild?
- Segmentierung - Klassifizierung jedes Pixels
- Lineare Regression - Lässt sich ein funktionaler Zusammenhang zwischen den Daten des Datensatzes finden, welcher eine Vorhersage zum weiteren Verlauf der Daten ermöglicht?
- Clustering - Wie lassen sich Daten vergleichen und in Gruppierungen bei gewisser Ähnlichkeit Ihrer Attribute zusammenfassen?
- Image Captioning - Wie lassen sich die klassifizierten Objekte in Beziehung setzen?

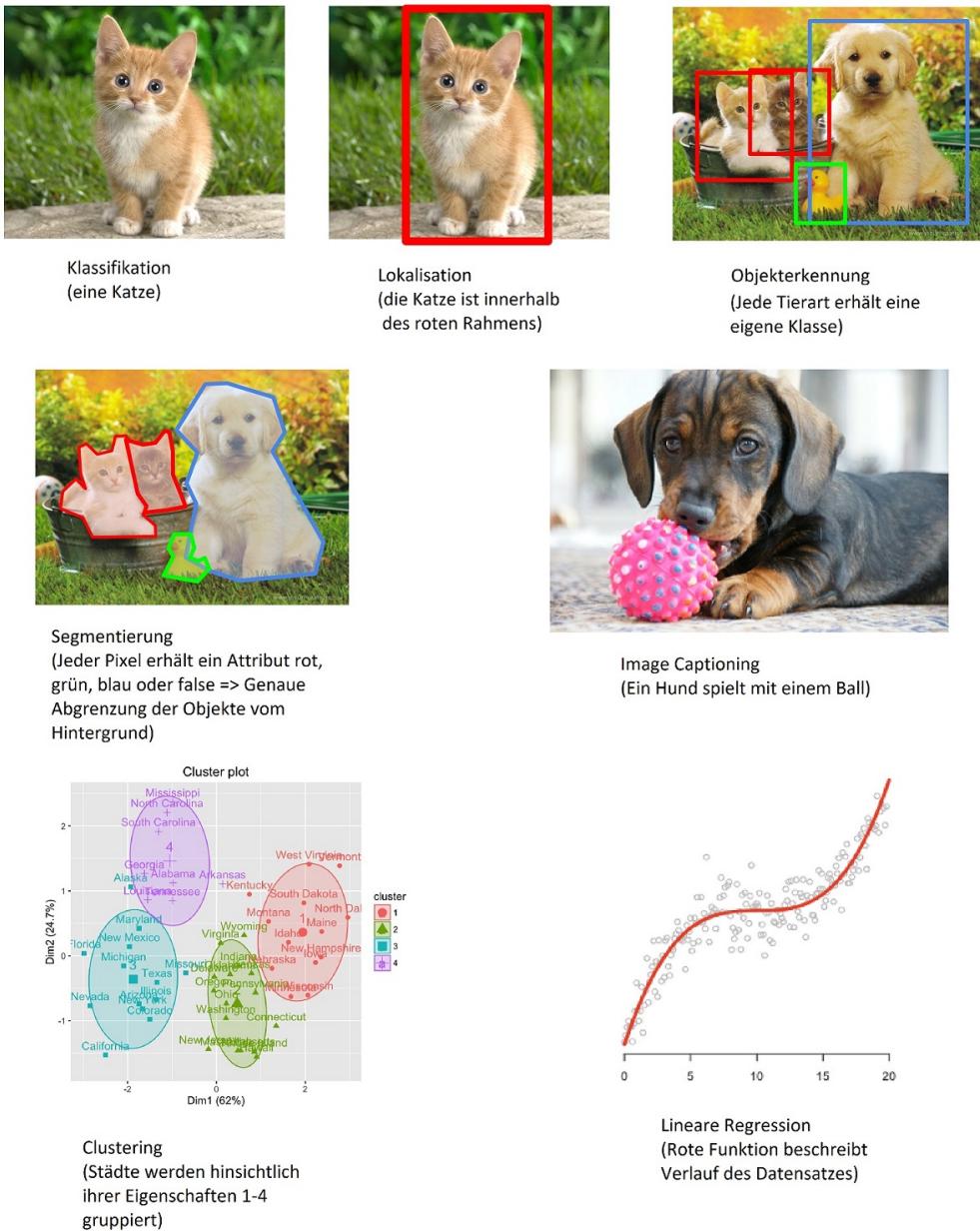


Abbildung 3.1: Vergleich verschiedener Anwendungszwecke von Deep Learning

### 3.2.1 Linear Classifier

Um bestimmen zu können, wie gut ein Bild  $x$  ( $x_i$  sind die einzelnen Pixelwerte) zu einer Klasse  $k_j$  passt, muss mithilfe einer Funktion  $f$  ein numerischer Vergleichswert bestimmt werden. Die Funktion  $f$  wird auch Score-Funktion genannt:

$$f(x_i, W, b) = W \cdot x_i + b$$

Der Parameter  $W$  ist die sogenannte Gewichtsmatrix. Sie besitzt die Dimensionalität  $i_{max} \times k_{max}$ . Der Parameter  $b$  heißt Bias und besitzt die Dimensionalität  $k_{max} \times 1$ . Er hat die gleiche Funktion wie die Gewichtsmatrix, ermöglicht aber eine zusätzliche additive Änderung beim Lernen. Im nächsten Unterkapitel wird die genauer erläutert, wie dies zu interpretieren ist.

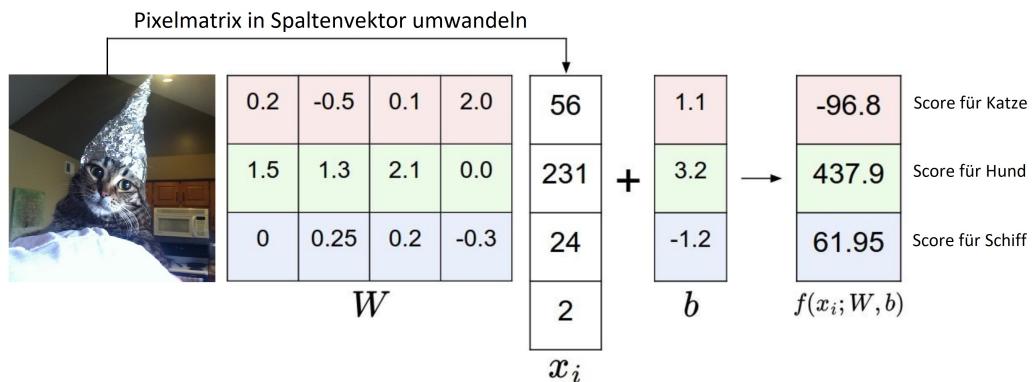


Abbildung 3.2: Interpretation der Score-Funktion anhand eines Beispiels

Wenn man ein Bild mit  $i_{max}$  Pixeln als  $i_{max}$ -dimensionalen Vektor auffasst, dann lässt sich ein Classifier als Hyperebenenseparator dieses Vektorraumes interpretieren. Je höher die Score Funktion für eine Klasse, desto geringer ist der Abstand zum Untervektorraum mit entsprechenden zur Klasse zugehörigen Mustern.

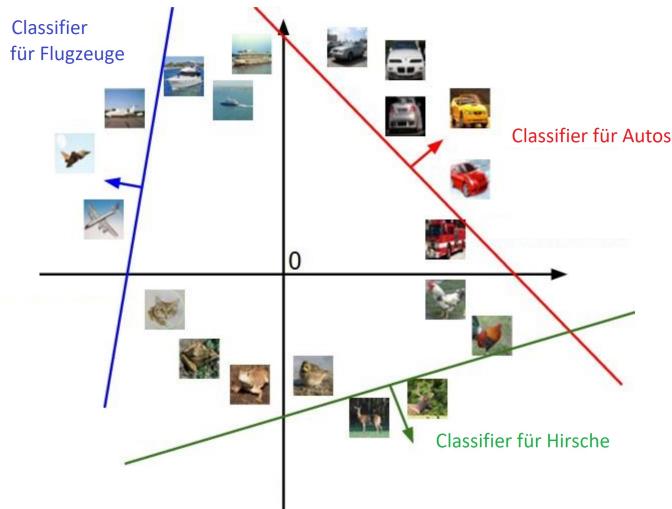


Abbildung 3.3: Hirsch-Auto-Flugzeug-Hyperraum mit Ebenenseparatoren

Äquivalent zur Score-Funktion definiert man auch eine sogenannte Loss-Funktion. Je besser ein Datum einer bestimmten Klasse zugeordnet werden kann, desto besser sind die Parameter der Gewichtsmatrix und des Bias und desto kleiner der Wert der Loss-Funktion. Im Rahmen eines Optimierungsprozesses soll die Loss-Funktion durch Änderung der Gewichte minimiert werden. Zwei häufig verwendete Loss-Funktionen für Daten sind:

- **Hinge-Loss**  $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta) \Rightarrow$  Richtige Klasse muss um mindestens Delta größer sein, als alle anderen Klassen
- **Softmax-Loss**  $L_i = -\log \left( \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right) \Rightarrow$  Minimieren der negativen logarithmischen Wahrscheinlichkeit für die korrekte Klasse

Der gesamte Loss eines Optimierungsproblems besteht jedoch nicht nur aus dem oben genannten Datenloss, sondern des Weiteren aus dem Regularisierungs-Loss.

### 3.2.2 Regularisierer

Ein Regularisierer soll verhindern, dass sich die Gewichte immer nur hinsichtlich einer bestimmten Klasse verändern. Zum Beispiel kann es sein, dass ein Netzwerk, das ohne Regularisierer trainiert wurde zwar gut Katzen erkennen kann, jedoch keine Hunde. Das Ziel ist hierbei also, ein Netzwerk zu erzeugen, dass die Generalisierung "Tier" versteht und darüber hinaus zwischen verschiedenen Tierarten unterscheiden kann. Eine Möglichkeit, dies zu realisieren, bieten Bestrafungsterme:

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

Dieser Bestrafungsterm geht ebenfalls in die Loss-Funktion mit ein. Die Quadrierung der Gewichte sorgt dafür, dass zuvor bereits deutlich größere Gewichte in der Loss-Funktion umso mehr berücksichtigt werden. Im Laufe des künftigen Trainingsprozesses können Ausreißer effektiv erkannt und vermieden werden. Die Loss-Funktion wird nur dann minimal sein, wenn sich die Gewichte nicht zu stark voneinander unterscheiden.

Die gesamte Loss-Funktion lautet also:

$$L_{ges} = \frac{1}{N} \sum_i L_i + \lambda R(W)$$

### 3.2.3 Gradientenabstieg

Um die Loss-Funktion zu minimieren, müssen die Gewichte  $W$  aktualisiert werden. Hierzu berechnet man den negativen Gradienten (Partielle Ableitungen der Loss-Funktion nach den Gewichten). Auf der Ebene, welche die Loss-Funktion aufspannt, zeigt der Gradient in Richtung des steilsten Abstiegs.

### 3.2.4 Multi-Layer Neural Network

Ein Multi-Layer Neuronales Netzwerk ist ein gerichteter azyklischer Graph, wobei die Knoten in Ebenen angeordnet werden. Die erste Knotenebene besteht aus den Eingabedaten. Im Bildbeispiel wäre jeder Knoten ein Pixelwert. Die Kanten des Graphen entsprechen den Werten der Gewichtsmatrix  $W$ . Die Knoten der Hidden-Layers unter der Output-Layer lassen sich als Neuronen interpretieren, welche durch Aktivierung oder Deaktivierung bestimmte Muster der Eingabedaten lernen. Ob ein Neuron aktiviert wird oder nicht, bestimmt die Aktivierungsfunktion  $g$ . Das Ergebnis wird an eine weitere Neuronenschicht weitergeleitet und die gelernten Muster somit weiter verfeinert.

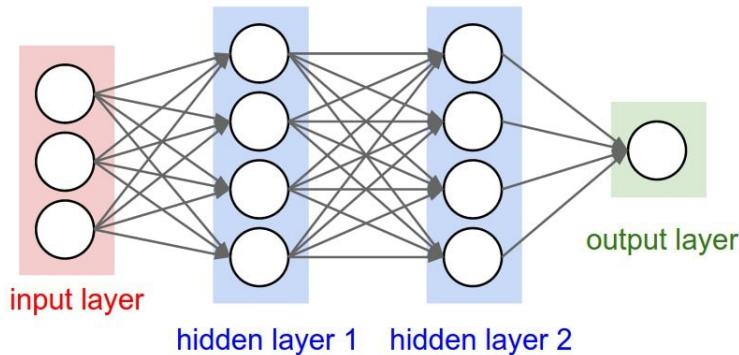


Abbildung 3.4: Multi-Layer NN. mit 2 Hidden-Layers, 32 Gewichten, 9 Biases und 9 Neuronen

Die Score-Funktion für dieses Netzwerk lautet:

$$y(x) = W_3^T \cdot g[W_2^T \cdot g[W_1^T \cdot x + b_1] + b_2] + b_3$$

### 3.2.5 Trainieren eines Netzwerks mittels Backpropagation

Sei  $f$  eine Score-Funktion für ein Single Layer Neural Network mit nur einer Aktivierung.

$$f(x, w) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

Diese Aktivierungsfunktion nennt sich auch Sigmoidfunktion. Sie transformiert reelle Zahlen auf ein Intervall  $[0, 1]$ .

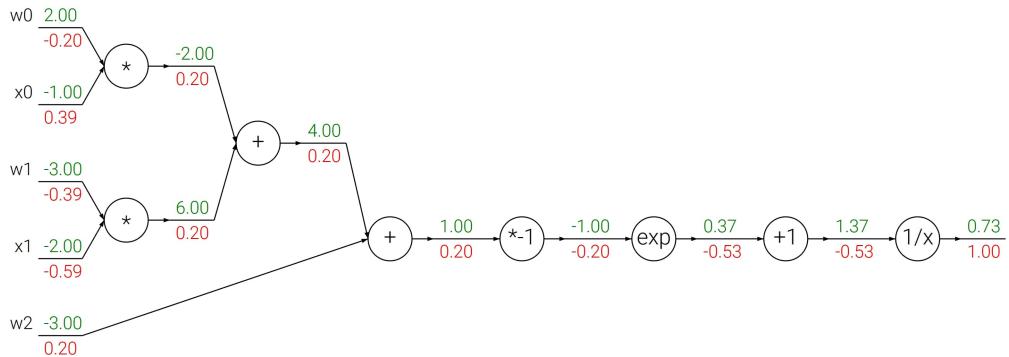


Abbildung 3.5: Beispiel für Backpropagation anhand der Sigmoid-Funktion

Das Ergebnis von  $f(x, w)$  mit  $w = (2, -1, -3)$  und  $x = (-1, -2)$  ist 0.73. Die Ableitung von  $\frac{1}{x}$  ist  $-\frac{1}{x^2}$ . Anschließend rechnet man  $-\frac{1}{1.37^2} \cdot 1 = -0.53$ . Die Ableitung von  $x + 1$  ist 1. Also rechnet man  $1 \cdot -0.53 = -0.53$ . Die Ableitung von  $e^x$  ist  $e^x$ . Man rechnet  $e^{-1} \cdot 0.53 = -0.2$ . Diese Vorgehensweise entspricht der Anwendung der Kettenregel. Fährt man fort, erhält man Gewichtsänderungen  $W_{exch} = (-0.2, -0.4, 0.2)$ . Diese Änderungen subtrahiert man von den alten Gewichten mit einem Parameter  $step$ :  $W := step \cdot W_{exch}$ . Dabei gibt Lernrate  $step$  an, wie stark die Änderung der Gewichte ausfallen soll. Eine zu große Änderung kann bedeuten, dass man über das Minimum der Loss-Funktion hinaus springt. Sind die Änderungen zu klein, so muss der Gradient oft berechnet werden, was sehr lange dauern kann. Das Ergebnis der Score-Funktion für  $W' = (1.8, -3.4, -2.8)$  ist 0.9. Das Netzwerk klassifiziert nun also besser.

Neben der Sigmoidfunktion gibt es noch viele andere Aktivierungsfunktionen. So hat die Maximumsfunktion (Rectifier Linear Unit - ReLU) gegenüber der Sigmoidfunktion den Vorteil, dass bei relativ hohen Werten nicht alle Neuronen aktiviert werden und es somit weiterhin zu einem Lerneffekt kommt. Ein weiterer Vorteil des Rectifiers ist die schnellere Konvergenz des Gradienten. So stellt sich bereits nach wenigen Trainings ein starker Lerneffekt ein. Sind die gewichtete jedoch zu Beginn sehr klein, so ist die Gefahr groß, dass bei einer weiteren Verkleinerung alle Neuronen deaktiviert werden.

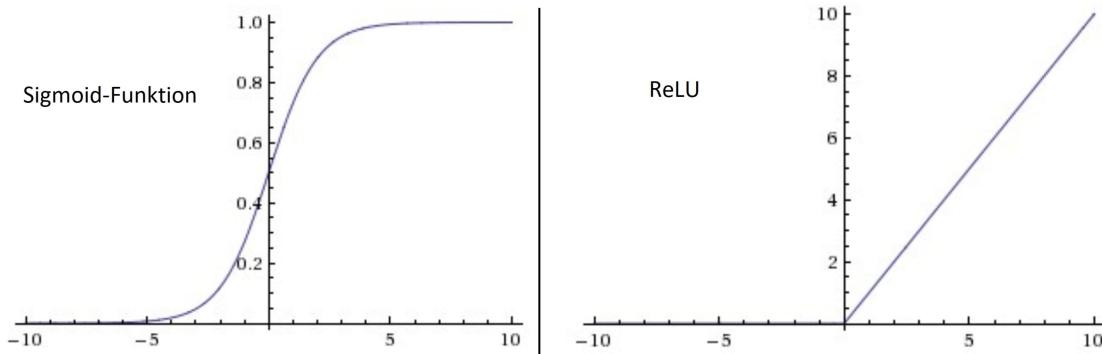


Abbildung 3.6: Vergleich zwischen Rectifier und Sigmoid

### 3.2.6 Parameterupdates

Die folgenden Verfahren sind die Grundlegendsten und am häufigsten eingesetzten Methoden zur Aktualisierung der Gewichte. Aus ihnen sind viele weitere Optimierungen hervorgegangen, die auch in der heutigen Forschung noch intensiv diskutiert werden [Rud].

**Batch Gradient Descent (BGD)** Normaler einmaliger Updateprozess wie oben beschrieben. Diese Variante ist sehr langsam, da für jedes Update der Gradient für den gesamten Datensatz berechnet werden muss.

$$W = W - \eta \nabla_W L(W)$$

**Stochastic Gradient Descent (SGD)** In diesem Fall werden die Gewichte nach jedem Trainingsbeispiel aktualisiert. Dies beschleunigt den Trainingsprozess und verhindert Redundanzen:

$$W = W - \eta \nabla_W L(W; x_i)$$

Zur Beschleunigung des Gradientenabstiegs wurden ebenfalls Verfahren entwickelt:

- **Momentum** Falls der Gradient zu groß ist, wird das Ziel beim aktualisieren der Gewichte übersprungen. Diese Methode beschleunigt den Gradientenabstieg, indem die Anzahl der übersprungenen Minima reduziert wird (quantitative Oszillationsreduktion). Hierzu wird der Update-Vektor des vorherigen Schrittes durch einen Parameter  $\gamma$  gewichtet und zum neuen Update-Vektor addiert:

$$v_t = \gamma v_{t-1} + \eta \nabla_W L(W)$$

$$W = W - v_t$$

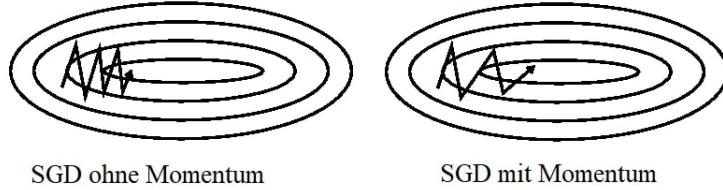


Abbildung 3.7: Momentum verringert die Oszillation der Loss-Funktion

- **Adagrad** Dieses Verfahren findet für jeden Parameter in jedem Updateschritt eine individuelle Lernrate. Dies bewirkt, dass das Minimum bei starker Steigung nicht übersprungen wird. Die Oszillation wird qualitativ reduziert.

$$W_{t+1,i} = W_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \nabla_W L(W_{t,i})$$

Wobei  $G_t$  eine Diagonalmatrix ist und das Element  $i, i$  ist die Summe der Quadrate der Gradienten bezüglich  $W_t$  bis Zeitschritt  $t$ . Das  $\epsilon$  verhindert eine Nulldivision.

### 3.2.7 Convolutional Neural Network

Ein Convnet ist ein Neuronales Netzwerk, das speziell für das Klassifizierungsproblem geeignet ist. Ein Standardnetzwerk besteht aus folgenden vier Layern:

- **1. Convolution Layer** Dies ist ein Filter  $W$ , der Dimension  $F \times F \times c$ , wobei das Bild  $P$  die Dimension  $N \times N \times c$  hat und  $N > F$ . Die neue Größe (Höhe/Breite  $N$ ) des Ausgabebildes nach der Convolution beträgt  $\frac{N-F}{stride+1}$ , wobei der Stride die Anzahl der Gitterzellen ist, die der Filter jedes mal weiter springt. Als Padding bezeichnet man die Anzahl der Gitterreihen, die um das Ursprungsbild herum virtuell angefügt werden, um zu verhindern, dass das Bild durch die Convolution kleiner wird. Zu Beginn wird der Filter an die linke obere Ecke angesetzt. Anschließend wird der erste Wert des neuen Bildes berechnet.

$$\sum_{i=1}^F \sum_{k=1}^F P_{i,k} \cdot w_{i,k}$$

Danach wird der Filter um *stride* Gitterzellen nach rechts verschoben und es wird erneut aufsummiert.

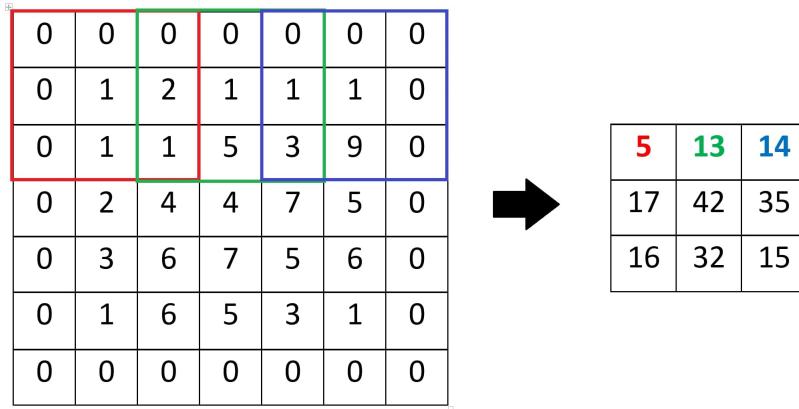


Abbildung 3.8: Convolution eines  $7 \times 7$  Bildes mittels  $3 \times 3$  Eins-Filter ( $w_{i,k} = 1$ ), Padding = 1 und Stride = 2

Jeder Filter erstreckt sich über die gesamte Tiefe des Bildes. Wird also ein  $B \times B \times 3$  Filter auf ein  $A \times A \times 3$  Bild angewandt, so hat das Ausgabebild die Dimension  $C \times C \times 1$  mit  $C = \frac{A-B}{\text{stride}+1}$ . Werden 10 Filter der Größe  $B \times B \times 3$  angewandt, so hat die Ausgabe die Dimension  $C \times C \times 10$ . Je mehr Filter angewandt werden, desto besser können Muster gelernt werden. Zu große und zu kleine Filter führen zu Overfitting und zu Underfitting.

- **2. ReLU Layer** Diese Layer übernimmt das Aktivieren der einzelnen Gewichte mit z.B.  $\max(0, x)$ .
- **3. Pooling Layer** Diese Layer reduziert die Ausgabe nach der Filterung und Aktivierung auf wesentliche Merkmale des Bildes und verkleinert die Anzahl der Pixel, die für die Speicherung dieser Merkmale notwendig sind. Zum Einsatz kommt meistens das Max-Pooling. Hierbei wird wie bei der Convolution eine Maske auf das Bild gesetzt. Das Maximum aller Bildwerte, die sich hinter der Maske befinden, ist der Wert des neuen Bildes nach dem Pooling.

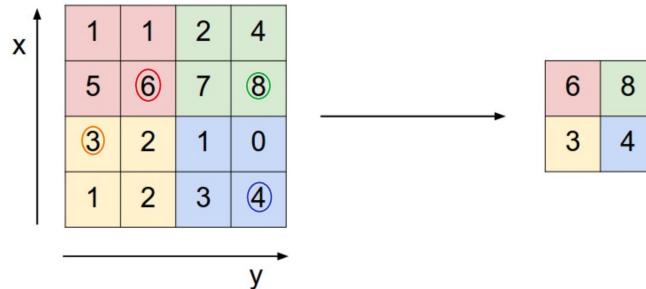


Abbildung 3.9: Pooling mit stride = 2

- **4. Fully Connected Layer** Diese Neuronen-Schicht soll die Ausgabe nach den oben genannten Schritten interpretierbar machen und auf Klassen-Scores abbilden. Jedes

Neuron ist mit jedem aktivierten Gewicht davor liegenden Schicht verbunden.

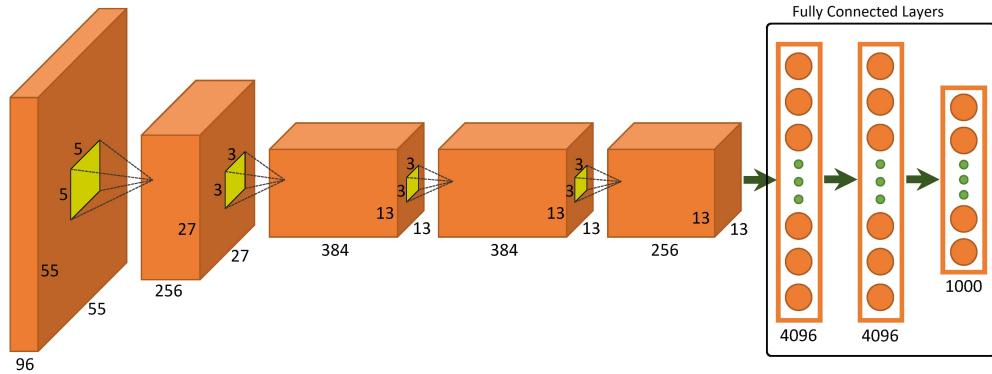


Abbildung 3.10: Gewöhnliches Convnet mit Fully-Connected-Layers zum Schluss

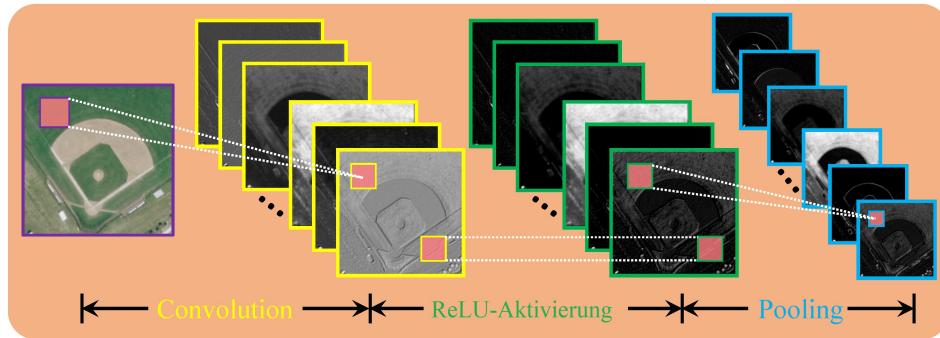


Abbildung 3.11: Verbildlichung der Schritte 1-3 durch Anwendung von Convolution auf ein Luftbild<sup>2</sup>

### 3.2.8 Recurrent Neural Network

Ein Recurrent Neural Network (RNN) ist ein zeitabhängiges neuronales Netz, mit welchem sich Zustände vorhersagen lassen. So lassen sich mit einem RNN z.B. folgende Probleme lösen:

- Image Captioning
- Sentimentanalyse (Stimmungserkennung im Text Mining)
- Sprachübersetzung, Textproduktion und Vorhersage von Wörtern
- Videoklassifikation
- Vorhersage von geologischen Aktivitäten

Ein Zustand  $h_t$  und eine Ausgabe  $y_t$  eines RNN zu einem bestimmten Zeitpunkt  $t$  berechnet sich wie folgt:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b1)$$

$$y_t = g(W_{hy}h_t + b2)$$

<sup>2</sup><http://www.mdpi.com/2072-4292/7/11/14680/htm>

Hierbei enthalten die Gewichtsmatrizen  $W_{xh}$ ,  $W_{hh}$ ,  $W_{hy}$  die zu lernenden Parameter zwischen Input Layer, Hidden Layer und Output Layer. Die Funktionen  $f$  und  $g$  sind Nichtlinearitäten.

### Beispiel:

Ein RNN wird mit einer gegebenen Sequenz ('h' 'e' 'l' 'l' 'o') trainiert, so dass es bei der Eingabesequenz  $x = ('h' 'e' 'l' 'l')$  die Sequenz ('e' 'l' 'l' 'o') und somit den Buchstaben 'o' korrekt vorhersagt.

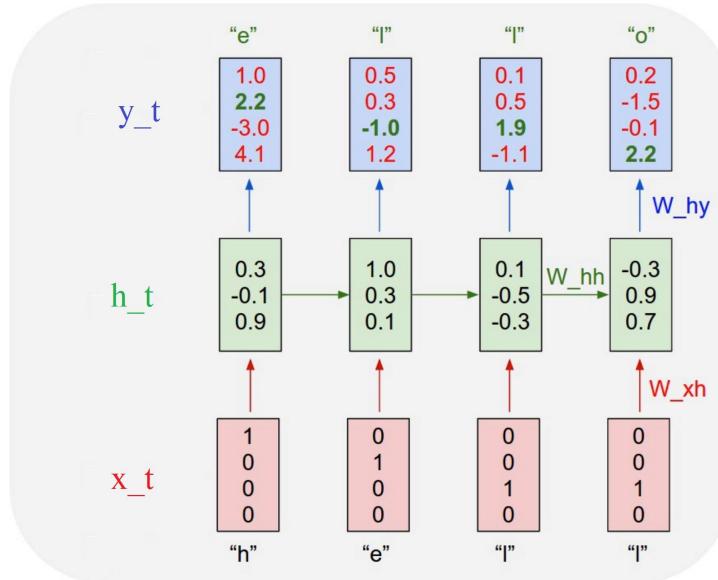


Abbildung 3.12: RNN vervollständigt das Wort *hello*

Die Gewichtsmatrix  $W_{xh}$  speichert, wie Buchstaben zu Hidden States umgewandelt und inkrementiert werden.  $W_{hy}$  lernt, welcher Ausgabebuchstabe zu gegebenem Hidden State passt und die Matrix  $W_{hh}$  akkumuliert und migriert alle bisherigen Eingaben im zeitlichen Verlauf zu einem gemeinsamen Hidden State.

### 3.3 Machine Learning in Geowissenschaften

#### 3.3.1 Herausforderungen und Chancen von Machine Learning

##### Einleitung

Die moderne Geowissenschaft steht vor vielen Herausforderungen, wie den Prognosen zu Klimawandel, Luftverschmutzung, Naturkatastrophen, Ressourcenverbrauch oder den Risiken für Erdbeben, Landrutsch und Vulkanausbrüchen. Die Forschung an solchen Problemen macht die interdisziplinäre Arbeit sämtlicher Wissenschaften unumgänglich. Geowissenschaften haben sich in den letzten Jahrzehnten zu einer Big-Data Disziplin entwickelt. Angestoßen wurde diese Entwicklung durch technologische Verbesserungen, wie dem Wachstum der Rechenleistung. Aufwändige Simulationen und die Demokratisierung von Datenbeständen, deren öffentliche Verbreitung im Internet und vielseitige Beteiligungsmöglichkeiten trugen ebenfalls dazu bei. Die wachsende Verbreitung großer geowissenschaftlicher Datenbestände ermöglicht ein enormes Potenzial für Maschinelles Lernen [Kar+17].

##### Sammeln von Geodaten

Die Erde ist ein komplexes dynamisches System, bestehend aus Litosphäre, Biosphäre, Atmosphäre, Hydrosphäre. Einzelne Bestandteile dieses Systems (z.B. Ozeanschichten oder die Bodenbedeckung) befinden sich in einem ständigen Wandel und interagieren miteinander. Um Daten solcher Phänomene zu erheben, gibt es vor allem zwei Möglichkeiten:

- Messen mithilfe von Sensoren in/auf Satelliten, Flugzeugen, Ballons, Drohnen, Wetterstationen, Schiffen, Bojen.
- Sensorbasierte geowissenschaftliche Beobachtungen sind nicht uniform gerastert und beziehen sich häufig auf irreguläre Zeitintervalle (zum Beispiel schwankt die Position einer Boje mit der Zeit). Sie eignen sich jedoch Hervorragend zum sammeln von Daten wie Oberflächentemperatur, Luftfeuchtigkeit, Reflektionintensität, Chemischer Zusammensetzung der Atmosphäre, Strömungen und Drücke, Emissionen, seismischer Aktivität und der Oberflächengestalt der Erde. Diese Vielfalt an geologischen Eigenschaften erfordert bezüglich eines gegebenen Problems eine individuelle Zusammenstellung von relevanten Datensätzen. Die Datentypen müssen gegebenenfalls konvertiert und die Datenbestände interpoliert werden, um sie besser interpretierbar zu machen.
- **Ableiten aus mathematischen Modellen und Simulationen** Geologische Prozesse, ihre Interaktionen und Änderungen sind auf physikalische Gesetze zurückzuführen. Zum Beispiel sind Bewegungen von Wasser in der Liosphäre auf die Strömungsdynamic zurückzuführen. Ein Nachteil physikalischer Berechnungen auf Grundlage von Modellen für komplexe Systeme ist leider deren Ungenauigkeit. Dennoch eignen sie sich zur näherungsweisen Darstellung des zeitlichen Verlaufes von geophysikalischen Phänomenen. Ein weiterer Vorteil ist, dass Simulationen besonders große Datensätze erzeugen können, wenn Daten für große Zeitintervalle von Interesse sind. Diese ermöglichen dann wiederum eine Datenbasierte Analyse mittels Machine Learning.

**Herausforderungen** Leider ist die Nützlichkeit von Machine Learning für Knowledge Discovery häufig begrenzt. Geophysikalische Objekte sind häufig nicht klar definiert (keine

klar definierten Grenzen) und ändern sich häufig. Daten zu solchen Objekten können ebenfalls unterschiedliche Auflösungen haben, rauschen, unvollständig oder ungenau sein. Auch kann die zeitliche Auflösung aus historischen Gründen stark variieren (z.B. ein Weltkrieg, in dem historische Datensätze zerstört wurden). Auch liegen häufig nicht ausreichend Geländedaten vor. Die Herausforderungen lassen sich in 3 Hauptkategorien unterteilen:

- **Eigenschaften Geologischer Prozesse**

- **Amorphe Grenzen (Wellen, Flüsse, Stürme)** Segmentierungs- und Clusteringverfahren sowie Maßnahmen für Feature-Charakterisierungen sind notwendig.
- **Raumzeitliche Struktur** Für viele Machine Learning Methoden nimmt man an, dass beobachtete geophysikalische Eigenschaften nicht korrelieren und die erhobenen Daten gleichverteilt sind. Die Realität sieht anders aus. Benachbarte Orte sind stark korreliert (Wahrscheinlichkeit für Grasland ist in der Nähe eines Waldes größer als in einer Wüste). Änderungen (Wald => Wüste) bewirken Zustände, die für eine unbestimmbare Zeit persistieren (Die Wüste wird nicht regelmäßig zum Wald und umgekehrt), was auf Klimaveränderungen zurückzuführen ist. Zwei weit entfernte Orte können ebenfalls stark korrelierte Eigenschaften besitzen (z.B. Temperatur, Druck). Man nennt diese meist meteorologischen Korrelationen Telekonnektionen .
- **Hochdimensionalität** Die Erde ist eines der komplexesten bekannten Systeme mit einer extrem großen Anzahl an Variablen, die alle miteinander in sowohl räumlich als auch zeitlich relativ zur Größe der Erde winzigen Skalen miteinander wechselwirken. Für eine Verarbeitung und Speicherung von Daten solcher Systeme ist die Rechenleistung heutiger Computer nicht ausreichend.
- **Raumzeitliche Variabilität** Geologische Prozesse können stark schwanken, sowohl in kurzen Zeitintervallen (Jahreszeiten, Tidenhub), in langen Zeitintervallen (Polsprung, Präzession, Klimawandel) als auch räumlich (Gebirgsformationen, Vegetationszonen, Klimazonen). Es ist sehr schwierig ein Modell zu trainieren, dass alle diese Prozesse vereint. Eine lokale und zeitliche Begrenzung der Datensätze ist zwingend erforderlich.
- Seltene Phänomene - Seltene Ereignisse wie z.B. Vulkanausbrüche, Tsunamis und Erdbeben verfälschen das Modell, da sie nur für ein Training auf viel größeren Zeitskalen geeignet sind. Aus diesem Grund müssen sie erkannt und aus dem Modell herausgerechnet werden. Dies ist nahezu unmöglich, da es hierzu keine ausreichenden Erfahrungswerte gibt.

- **Sammeln von Geodaten**

- Daten mit verschiedenen Auflösungen - Beispiel: Zur Beurteilung von Waldbränden müssen Bilder aus Luftaufnahmen und Satellitenbildern miteinander kombiniert werden. Die Flugzeugbilder haben eine höhere räumliche Auflösung, die Satellitenbilder wurden jedoch in regelmäßigen Zeitabständen aufgenommen. Aus diesem Grund müssen Interpolations- oder Upsamplingmethoden entworfen werden, um die beiden Arten von Bildern vergleichbar zu machen.
- Rauschen, Unvollständigkeit, Ungenauigkeit - Viele Geodatensätze sind unvollständig oder rauschen, weil z.B. Sensoren temporär ausgefallen sind oder unter verschiedenen Wetterbedingungen Messungen durchgeführt haben. Manche

Daten sind erst dann interpretierbar, wenn Sie mit einem mathematischen Modell kombiniert werden. Dieses kann die Interpretierbarkeit jedoch ebenfalls beeinflussen.

- **Mangelhafte Datensätze**

- **Kleine Sample-Größe** Viele Datenbestände sind niedrigfrequent und extrem ungenau, wenn sie aus einer Zeit stammen, in welcher es entweder keine oder nur wenige Messinstrumente gab. Auch gibt es Orte, an denen es nur schwer möglich ist (zeitlich hochfrequente) Messungen durchzuführen (Bohrkernanalyse in Antarktis o.ä., Bäume mit ausreichendem Alter für Jahresringe). Eisbohrkerne aus der Antarktis lassen zudem keine Rückschlüsse über die Klimatischen Verhältnisse in anderen Regionen zu.
- **Mangelhafte gelabelte Geländedaten** Oft liegen Geländedaten in mangelnder Qualität vor. Dies liegt daran, dass sie nur mit Zeit- und Kostenintensiven Maßnahmen zu beschaffen sind. Eine geringe Datenqualität führt zu einem langsamem Trainingsprozesses und zu Unter- oder Überanpassung des Modells, was die Aussagekraft stark einschränkt.

Aus beiden oben genannten Gründen müssen Trainingsmodelle entwickelt werden, die mit kleineren Datensätzen zurecht kommen.

Maschinelles Lernalgorithmen können dazu beitragen, Geowissenschaftliche Objekte und Ereignisse zu Charakterisieren und somit helfen, das Erdsystem besser zu verstehen. Während traditionelle Ansätze auf handgefertigten Algorithmen basieren, können Machine Learning Algorithmen mit ihrer automatischen Mustererkennung die Berechnungszeit deutlich verkürzen. Eine große Herausforderung stellt die Charakterisierung von physikalisch ungenau definierten Objekten dar. Unsupervised Learning kann dabei helfen, anomale Objekte aufzuspüren (z.B. Landminen).

Ein weiterer großer Vorteil von Machine Learning ist die Erzeugung von Geodaten aus nur schwer beobachtbaren Prozessen (z.B. Methanausstoß und Konzentration in der Atmosphäre). Supervised Learning kann verwendet werden, um Fernerkundungsdaten zu analysieren und daraus Aussagen über das Ökosystem abzuleiten (z.B. Gesundheit der Vegetation oder Wasserqualität). Eine besondere Herausforderung ist dabei die Heterogenität solcher Daten, wie bereits weiter oben beschrieben. Eine mögliche Lösung sind Multi-Task-Learning-Frameworks, welche Datensätze zuerst in homogene Partitionen unterteilen (hierarchisches Clustern), und dann auf jeder dieser Partitionen einzeln trainieren. Dieses Vorgehen ist eine Regularisierungstechnik, welche die Überanpassung des Modells verhindern soll. Folgende Abbildung zeigt die Verbesserung der Genauigkeit der geschätzten Waldbedeckung vier brasilianischer Staaten, wobei die rot markierten Bereiche die Residuen darstellen.

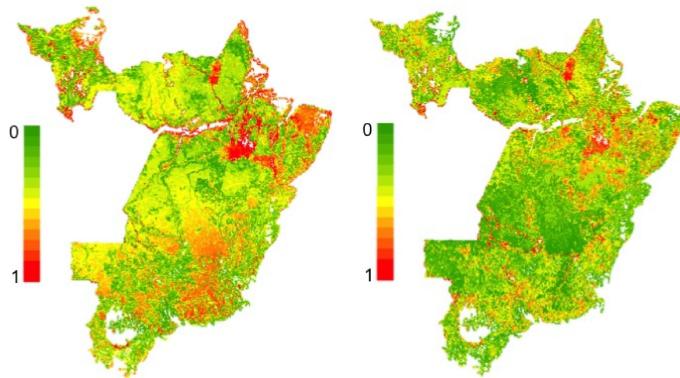


Abbildung 3.13: Schätzung der Waldbedeckung in Brasilien<sup>3</sup>

### Nichtstationarität, Heterogenität und Mangeldaten

Zum Umgang mit der Nichtstationarität (= Variable folgt keinem konstanten Wert) von Klimadaten wurden lernende Algorithmen entworfen, welche die Vorhersagen verschiedener Klimamodelle kombinieren. Statt eines Mittelwertes über die Klimamodelle berücksichtigt der selbstlernende Algorithmus zusätzlich die nichtstationären Dateneigenschaften und erzeugt wesentlich genauerer Vorhersagen.

Heterogene und minderqualitative Daten können mithilfe von *Adaptive Ensemble Learning* und *Label Refinement* besser analysiert werden.

Zur Näherung von Geophysikalischen Variablen mit Daten geringer zeitlicher Auflösung und fehlender Labels können folgende Techniken verwendet werden: - Regularisierer (für zeitl. Auflösung) - Semi-Supervised-Learning (fehlende Labels) - Active Learning (genauere Ergebnisse für Berechnungsprobleme) - Unsupervised Learning (bessere Näherung für geophysikalische Größen und Kartierung von Flächenänderungen durch z.B. Insektensterben, Waldabholzung und Ackerlandumwandlung)

### Techniken für Langzeitvorhersagen

Eine Möglichkeit für Langzeitvorhersagen war bisher die Verwendung von physikalischen Modellsimulationen. Diese können jedoch auch als Zeitreihen-Regressions-Problem aufgefasst werden. Mögliche Methoden zur Lösung solcher Probleme sind z.B.:

- Exponential Smoothing Techniques
- Autoregressive Integrated Moving Average (ARIMA) Modelle
- State-space Modelle
- Hidden Markov Modelle
- Kalman Filter

Transfer Learning - Ein zu trainierendes Modell für ein Problem mit wenig Daten soll mithilfe eines zuvor trainierten Modells mit vielen Daten das Problem besser lösen.

### Relationen und Kausalität

---

<sup>3</sup>A. Karpatne, Z. Jiang, R. R. Vatsavai, S. Shekhar, and V. Kumar, Monitoring Land-Cover Changes. IEEE Geoscience and Remote Sensing Magazine, 2016

Geophysikalische Zusammenhänge (z.B. Telekonnektionen und Dipole), sollen sich mithilfe datenbasierter Ansätze besser verstehen lassen. Man erhofft sich durch sie die Entdeckung neuer Korrelationsmuster. Darüber hinaus können graphenbasierte Repräsentationen von Klimadaten mithilfe von Clustering und Mustererkennung besser analysiert werden. Eine

besondere Herausforderung bei der Korellationserkennung ist der besonders große Suchraum mit all seinen raumzeitlichen Objekten und dynamischen, rauschenden und unvollständigen Geodaten. Es herrscht ein großer Bedarf an neuen Ansätzen, die gleichzeitig sowohl Zusammenhänge als auch die dazugehörigen interagierenden Objekte erkennen.

Ursache-Wirkungszusammenhänge zu entdecken ist eine weitere wichtige Aufgabe in den Geowissenschaften. Ein häufig eingesetztes Tool für die Analyse von solchen kausalen Zusammenhängen ist die multivariate Grangeranalyse mittels Vektorautoregression. Auf diese Weise können z.B. Sturmverläufe vorhergesagt werden. Weitere kaum erforschte Möglichkeiten sind das Reinforcement Learning und stoastische Anstze der dynamischen Programmierung zur Lösung von Entscheidungsproblemen.

### **Deep Learning**

Neuronale Netze haben die Eigenschaft, komplexe Features mithilfe der Verknüpfung weniger komplexer Features darzustellen. In Kombination mit dem Training großer Datensätze und der Fähigkeit, Fehler an den Nodes der Hidden-Layers zu minimieren, haben neuronale Netze weite Felder im Bereich Machine Learning revolutioniert. Darunter auch Supervised, Semi-Supervised, und Reinforcement Learning. Häufig werden neuronale Netze eingesetzt, wenn es schwierig ist, mittels handgeschriebener Algorithmen die wirklich relevanten Features aus einem komplexen Datensatz (wie es auch Geodatensätze sind) zu extrahieren. Geophysikalische Fragestellungen und Probleme haben viele Ähnlichkeiten zu den Themen, die im Bereich Computer Vision und Spracherkennung behandelt werden.

Während man ein Convolutional Neural Network einsetzt, um auf einem Bild eine Katze zu klassifizieren, kann man dieses auch verwenden, um Wetterphänomene wie Tornados auf Satellitenbildern zu erkennen.

Rekurrente Neuronale Netze mit Longshort-Term-Memory Zellen (LSTM) können z.B. genutzt werden, um zeitlich dynamische Plantagen mittels Fernerkundungsdaten kartografieren. RNNs besitzen die Eigenschaft besitzen, zeitlich vergangene Information zu speichern und in zukünftige Vorhersagen mit einzubeziehen. Aus diesem Grund eignen Sie sich zur Vorhersage von geologischen Ereignissen mit angemessener Vorlaufzeit.

Deep Learning kann bisher nur bei ausreichend gelabelten Daten zum Einsatz kommen. Aus diesem Grund herrscht hier ein Bedarf an neuen Verfahren, die mit nur wenigen Daten zureckkommen.

### **Fazit**

Die Forschung der letzten Jahre hat gezeigt, dass weder ein reiner Datenansatz noch ein reiner mathematischer Modellansatz ausreichend ist, um Knowledge Discovery effizient betreiben zu können. Aus diesem Grund sollten zukünftige physikalische Erkenntnisse möglichst früh und tief in Datenwissenschaftlichen Ansätze einbezogen werden. Auf diese Weise lässt sich auch die Wahrscheinlichkeit für Overfitting reduzieren, vor allem bei mangelnden Trainingsdaten.





## 4. Clusteringverfahren

### 4.1 Definition und Anwendungszweck

Ziel des Data Mining Prozesses ist es, Muster in Datenbanken zu erkennen und diese in Form von Wissen aufzubereiten. Im Falle des Clusterings werden Datenpunkte in Gruppen (Cluster) eingeteilt, so dass Daten mit ähnlichen Eigenschaften dem gleichen Cluster und Daten mit unterschiedlichen Eigenschaften verschiedenen Clustern zugeordnet werden. Im Bezug auf Geodaten wären zum Beispiel alle Punkte mit Farbwerten dunkelblau dem Cluster Wasser zugeordnet und alle Daten mit grüner Farbe dem Cluster Vegetation. Auf diese Weise lassen sich topologische Grenzen finden. Eine andere Möglichkeit bietet die gewichtete n-dimensionale Clusteranalyse. Sie ermöglicht die Zuordnung komplexer Daten zu komplexen Klassen. Wurde nach der numerischen Diskretisierung eines Objekts eine hohe Anzahl an Eigenschaften gefunden, so müssen Clusteringverfahren angewandt werden, welche unterschiedlich dichte Cluster unterschiedlichster Form erkennen können. Unter Umständen müssen die Daten mittels Hauptkomponentenanalyse auf Unterräume transformiert werden oder die Relevanz und Unabhängigkeit einzelner Attribute mittels Entropieanalyse bestimmt werden. Ein Beispiel hierfür ist die Analyse von Spektralklassen des von Geoobjekten reflektierten Lichts. Da das Reflektionsverhalten vieler Objekte für jeden Wellenlängenbereich charakteristisch ist, kann mittels 7D-Clustering (dem Clustering nach Einteilung der Wellenlängen in 7 Bereiche, auch Spektralkanäle genannt) bestimmt werden, um welches Objekt es sich handelt, oder wie eine Landfläche genutzt wird.

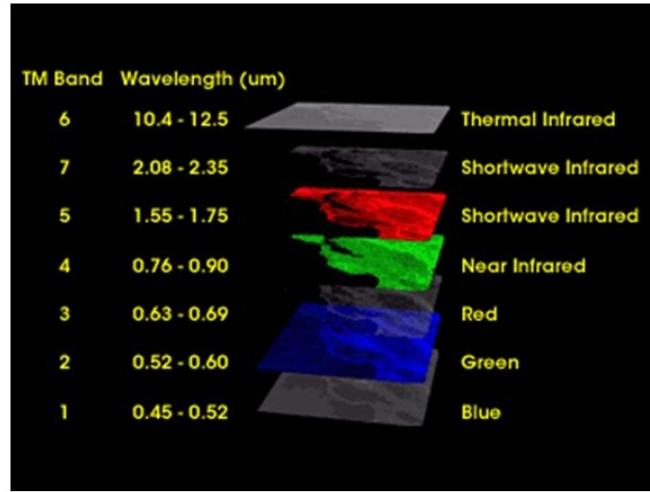


Abbildung 4.1: Landsat Spektralklassen <sup>1</sup>

Zu neuronalen Netzwerken, deren Funktionsweise nach dem Trainingsprozess nicht mehr mathematisch nachvollzogen werden kann, bieten Clusteringverfahren eine konkrete, statistische Alternative. Wird ein neuronales Netzwerk hinsichtlich einer Aufgabe angepasst, so ist eine Verbesserung nur durch Ausprobieren verschiedener Architekturen und Kernels verifizierbar. Optimierungen an Clusteringverfahren können jedoch meist als Anpassung einer Optimierungsfunktion formal mathematisch nachvollzogen werden.

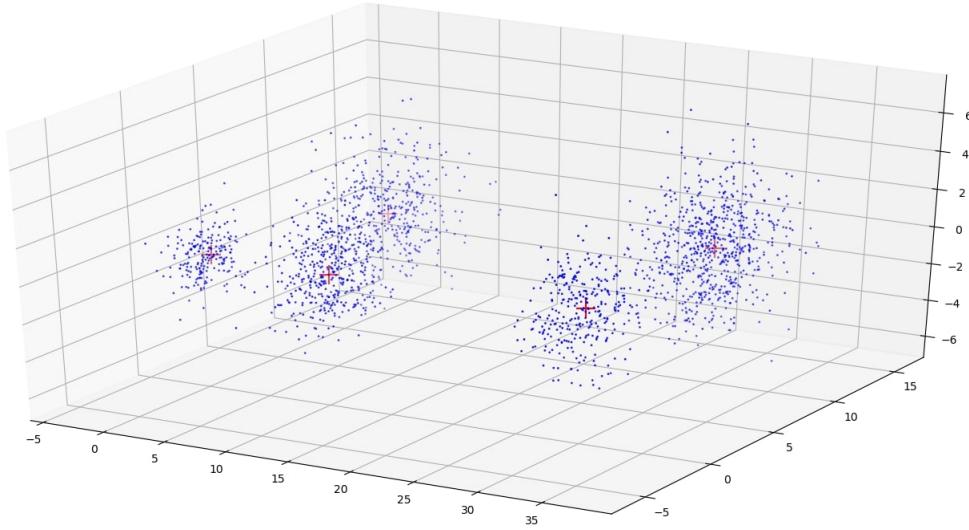


Abbildung 4.2: Geclusterter Datensatz: Jedes Datum (blaue Punkte) hat 3 Attribute. Jedes Kreuz ist das Zentrum eines gefundenen Clusters. Die optimale Clusteranzahl wurde mithilfe von Cluster Validity Indizes bestimmt.

<sup>1</sup><https://player.slideplayer.org/3/1324211>

Man unterscheidet zwischen harten (crisp) und weichen (fuzzy) Clusteringverfahren. Harte Methoden ordnen jedes Datum einer Datenbank einem Cluster eindeutig zu. Häufig kollidiert dieser Ansatz mit dem intuitiven Verständnis eines Menschen. Beispielsweise kann nicht gesagt werden, dass ein 49-jähriger Mensch jung ist und 50-Jähriger alt. Der Übergang von jung nach alt ist fließend. Aus diesem Grund gibt es weiche Methoden, die jedem Datum einen Grad der Zuordnung zu einem Cluster geben. Beispielsweise ist ein 70-Jähriger zu 80% alt und zu 20% jung. Der Grad der Zuordnung erhält jedes Datum also für jeden Cluster. Die Summe dieser Zuordnungsgrade beträgt meistens 100%.

#### 4.1.1 Die fünf Hauptschritte des Datenclustering

Der Prozess des Clusterings ist wie das Data Mining in Hauptschritte unterteilt[JMF99]. Sie lauten:

- **Feature Extraction** Seien  $n$  Objekte  $o_1, o_2, \dots, o_n$  gegeben. Im ersten Schritt werden die relevanten Features extrahiert, welche die Menge der Objekte am besten beschreiben. Formal lassen sich die Objekte durch die Menge folgender Feature-Vektoren beschreiben:

$$S = p_1, p_2, p_3, \dots, p_n \forall p_i \in R^m$$

- **Aufstellen der Ähnlichkeitsmatrix** Die Ähnlichkeitsmatrix  $P(S)$  der Dimension  $C \times n$  repräsentiert ein Clustering in  $C$ -viele Cluster, wobei jeder Eintrag  $u_{ij}, \forall i = 1, \dots, n$  und  $j = 1, \dots, C$  dem Grad der Zugehörigkeit des  $i$ -ten Objekts zum  $j$ -ten Cluster entspricht.
- **Berechnung der Gruppierungen** Anwenden eines harten oder weichen Clusteringalgorithmus
- **Datenabstraktion** Finden einer anschaulichen Repräsentation der Clusterergebnisse, meistens auf Basis der Clusterzentroide.
- **Evaluation der Ergebnisse** Wie gut ist das Clustering im Vergleich zu anderen Algorithmen? Hat der Algorithmus Cluster gefunden, wo gar keine sind? Wie ist die Clustertendenz zu bewerten, d.h. gibt es bevorzugte Cluster? Schneidet das Clustering für andere Werte  $C$  besser ab?

## 4.2 Crisp Clustering

### 4.2.1 Partitionierende Verfahren und ihre Nachteile

Einem partitionierenden Clusteringalgorithmus wird die feste Anzahl  $k$  an Clustern und die Kostenfunktion übergeben, welche für eine optimale Zuordnung einen kleinen Wert zurück gibt.

#### Clustering durch Varianzminimierung

Das einfachste Verfahren basiert auf der Idee, dass die Zuordnung genau dann optimal ist, wenn die Summe der Abstände zwischen Datenpunkten und ihren sogenannten Clusterrepräsentanten möglichst klein wird. Ein Clusterrepräsentant  $\mu_{C_i}$  eines Clusters  $C_i$  ist der Mittelwert aller Punkte  $p = (x_1, \dots, x_d)$  (Dimensionalität  $d$  ist die Anzahl der Attribute von  $p$ ), die  $C_i$  zugeordnet wurden. Zu Beginn des Algorithmus werden die Clusterprototypen (initiale Repräsentanten) entweder zufällig initialisiert, oder auf Basis einer Wahrscheinlichkeitsfunktion gewählt, so dass sie möglichst weit voneinander entfernt

sind (k-means++[AV06]). Dies reduziert die Anzahl der Iterationen bei Optimierung der Kostenfunktion. Anschließend wird jeder Punkt dem Cluster zugeordnet, dessen Repräsentant den kleinsten Abstand zu diesem Punkt hat (1). Auf Basis dieser Zuordnung werden die Repräsentanten erneut berechnet (2). Dabei wird die Kostenfunktion für das gesamte Clustering berücksichtigt:

$$\sum_{i=1}^k \sum_{p \in C_i} dist(p, \mu_{C_i})^2$$

Die Distanzfunktion hängt von der Dimensionalität des Datensatzes ab. Die Neuberechnung von Zuordnung (1) und Repräsentanten (2) wird solange wiederholt, bis keine ausreichende Änderung mehr auftritt.

### k-means

Eine beschleunigte Version der Varianzminimierung ist der **k-means-Algorithmus** [For65]. Wenn ein Datenpunkt  $p$  vom Cluster  $C_1$  seine Zugehörigkeit an den Cluster  $C_2$  verliert, dann wird nicht auf die restlichen Punkte bei der Neuzuordnung gewartet. Stattdessen werden die Clusterzentroide der betreffenden Cluster  $C_1$  und  $C_2$  wie folgt sofort aktualisiert:

$$\mu_j'^{C_1} = \frac{1}{|C_1| - 1} \cdot (|C_1| \cdot \mu_j^{C_1} - x_j^p)$$

und

$$\mu_j'^{C_2} = \frac{1}{|C_2| + 1} \cdot (|C_2| \cdot \mu_j^{C_2} + x_j^p)$$

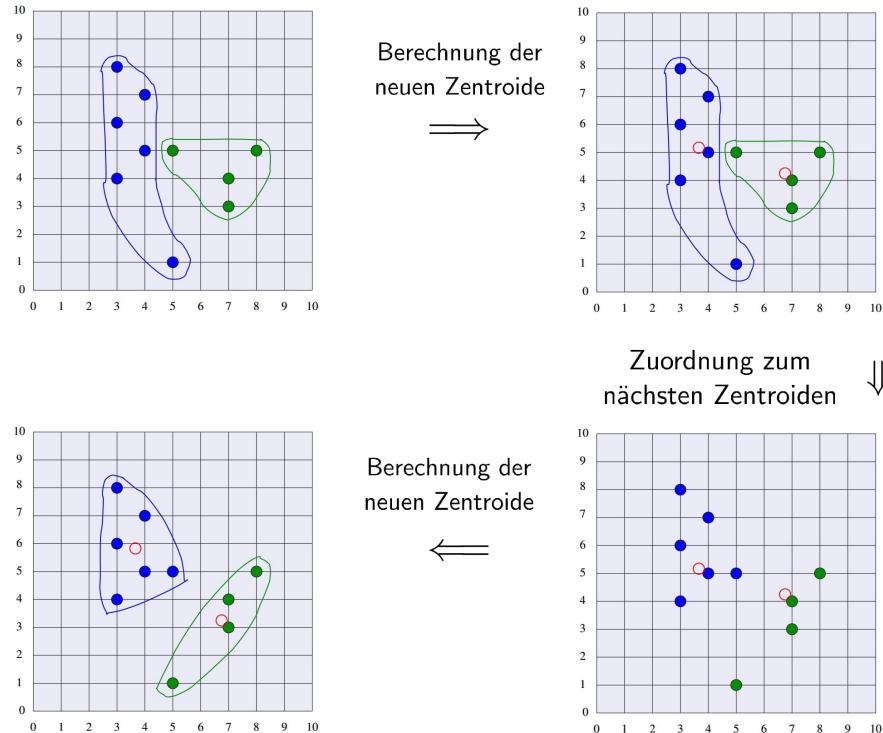


Abbildung 4.3: Visualisierung des Ablaufs der beiden alternierenden Schritte

Leider erkennt der k-means-Algorithmus weder Cluster unterschiedlicher Dichte, noch unterschiedlicher Form. Auch werden sich überlappende Cluster nicht erkannt. Zudem ist die Qualität des Endergebnisses stark von der initialen Wahl der Clusterprototypen abhängig.

### EM-Algorithmus

Ein anderer Ansatz der alternierenden Optimierung ist die Erwartungsmaximierung [DLR77]. Ihr Vorteil ist, dass auch unterschiedlich geformte und dichte Cluster berücksichtigt werden können. Dabei wird jeder Cluster durch eine Wahrscheinlichkeitsverteilung beschrieben, welche aus der Erzeugung aller Datenpunkte eines Clusters  $C$  mittels Kovarianzmatrix  $\sum_C$  der Dimension  $d \times d$  resultiert. Wie bei der Varianzminimierung bezeichnet  $\mu_C$  auch hier den Mittelpunkt aller Daten, die dem Cluster  $C$  zugeordnet wurden. Die Wahrscheinlichkeit, mit der bei einer einzigen Normalverteilung  $C$  das Datum  $x$  erzeugt wurde, beträgt:

$$P(x|C) = \frac{1}{\sqrt{(2 \cdot \pi)^d \cdot |\sum_C|}} \cdot e^{-\frac{1}{2}(x-\mu_C)^T \cdot (\sum_C)^{-1} \cdot (x-\mu_C)}$$

Da es mehrere Cluster gibt und alle den Punkt  $x$  mit einer unterschiedlichen Wahrscheinlichkeit erzeugen, muss die Notation der bedingten Wahrscheinlichkeit erfolgen. Die Wahrscheinlichkeitsdichte eines Clusters bezüglich  $x$  lautet  $P(x|C_i)$ . Sind alle Cluster gaußverteilt, so kann die Gesamtdichte für den Punkt  $x$  berechnet werden:

$$P(x) = \sum_{i=1}^k W_i \cdot P(x|C_i)$$

Wobei  $W_i$  der relative Anteil der Datenpunkte ist, die zu Cluster  $C_i$  gehören, was der Gesamtwahrscheinlichkeit des Clusters  $C_i$  entspricht. Der Satz von Bayes ( $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$ ) ermöglicht nun die Berechnung der Wahrscheinlichkeit, mit der  $x$  zum Cluster  $C_i$  gehört:

$$P(C_i|x) = W_i \cdot \frac{P(x|C_i)}{P(x)}$$

Durch abwechselnde Berechnung von Zugehörigkeiten  $P(C_i|x)$  und Schätzung neuer Verteilungsparameter  $\mu_C$ ,  $\sum_C$  und  $W_i$  kann das Gütemaß des Clusterings maximiert werden. Es lautet:

$$E(M) = \log \left( \prod_{x \in D} P(x) \right)$$

Wobei  $M$  die Menge aller Cluster ist:  $M = \{C_1, \dots, C_k\}$ . Die Optimierung wird abgebrochen, wenn  $E(M) - E(M') < \epsilon$ , das Gütemaß also nicht mehr in ausreichendem Maße kleiner wird.

#### 4.2.2 Silhouettenkoeffizient - Wahl der Clusteranzahl

Beide vorgestellten Varianten haben die Eigenschaft, das Gütemaß für größere Werte des Parameters  $k$  immer weiter zu verbessern. Das Optimum wird erreicht, wenn jedes Datenobjekt genau einen Cluster darstellt, dessen Repräsentant das Datum selbst ist. Um

dies zu vermeiden, braucht es ein Gütemaß, welches das Clustering unabhängig von der gewählten Clusteranzahl bewertet. Dieses Gütemaß ist der Silhouettenkoeffizient [ES00]. Sei  $dist(o, C_i)$  der Abstand eines Punktes  $o$  von Cluster  $C_i$ , also der gemittelte Abstand zwischen  $o$  und allen Punkten im Cluster  $C_i$ . Sei  $a(o)$  der Abstand von  $o$  zum Cluster, dem  $o$  zugeordnet ist und  $b(o)$  der Abstand zwischen  $o$  und dem nächstgelegenen Cluster (nicht der Cluster, der in  $a(o)$  betrachtet wird!). Dann ist die Silhouette eines Objekts  $o$ :

$$s(o) = \begin{cases} 0 & \text{wenn } a(o) = 0 \\ \frac{b(o) - a(o)}{\max\{a(o), b(o)\}} & \text{sonst} \end{cases} \quad (4.1)$$

Ist der Wert  $s(o) = 1$ , dann wurde  $o$  seinem Cluster korrekt zugeordnet, ist  $s(o) = -1$ , dann ist die Zuordnung sehr schlecht.

Der Silhouettenkoeffizient ist dann die durchschnittliche Silhouette aller Datenobjekte aller Cluster des Clusterings  $C_M = \{C_1, \dots, C_k\}$ :

$$s(C_M) = \frac{\sum_{C \in C_M} \sum_{o \in C} s(o)}{|O|}$$

Der Koeffizient nimmt Werte zwischen 0 und 1 an, wobei Werte über 0.75 darauf hindeuten, dass eine sehr gute Zuordnung gefunden wurde. Werte unter 0.25 deuten auf ein unbrauchbares Clustering hin.

Die optimale Clusteranzahl  $k$  kann jetzt bestimmt werden, indem der Silhouettenkoeffizient für verschiedene Clusterings  $k = 2, \dots, n - 1$  berechnet wird, und dann der Wert  $k$  ausgewählt wird, für den der Koeffizient den höchsten Wert hat.

#### 4.2.3 OPTICS - Dichtebasierter Clustering

Einer der bedeutendsten Clusteringalgorithmen ist das dichtebasierte DBSCAN-Verfahren. Es ist neben seiner Abwandlungen und Optimierungen (OPTICS und HDBSCAN) in vielen Geoinformationssystemen, wie z.B. arcGIS implementiert<sup>2</sup>.

DBSCAN nimmt an, dass Cluster Punktregionen mit höherer Dichte sind, die durch Regionen mit geringerer Dichte (Noise) voneinander getrennt werden. Es kann ausschließlich Cluster desselben Dichteniveaus erkennen. Seine Optimierung OPTICS, welches als hierarchisches dichtebasierter Clusteringverfahren zählt, kann dagegen auch unterschiedlich dichte Cluster und Subcluster erkennen.

---

<sup>2</sup><http://pro.arcgis.com/de/pro-app/tool-reference/spatial-statistics/densitybasedclustering.htm>

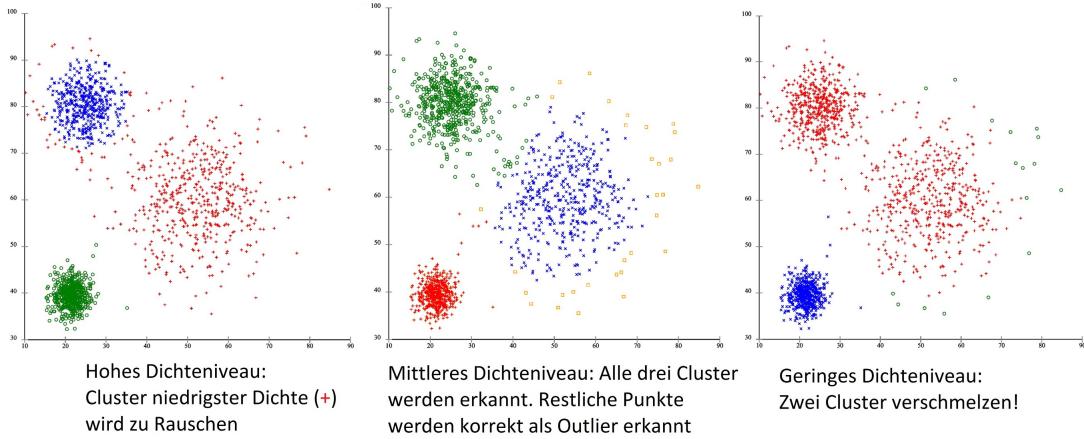


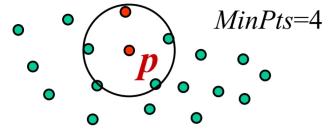
Abbildung 4.4: Probleme von DBSCAN bei Erkennung von Clustern mit unterschiedlicher Dichte

### Erreichbarkeitseigenschaften [Est+96]

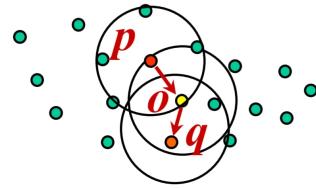
Die  $\epsilon$ -Nachbarschaft  $N_\epsilon(p)$  ist die Menge aller Punkte, welche maximal  $\epsilon$  Einheiten von  $p$  entfernt sind.

Ein Punkt heißt **Kernobjekt**, wenn die Anzahl der Objekte in der  $\epsilon$ -Nachbarschaft mindestens  $minPts$  beträgt, wobei  $\epsilon$  und  $minPts$  feste Werte sind, die dem Algorithmus initial übergeben werden. Sie spezifizieren zusammen die Grenzdichte eines Clusters.

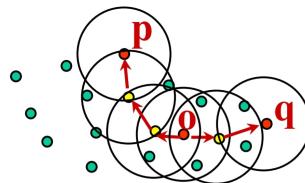
Ein Objekt  $o$  ist **direkt-dichte-erreichbar**, wenn es in der  $\epsilon$ -Nachbarschaft von  $p$  liegt und  $p$  ein Kernobjekt ist.



Ein Objekt  $q$  ist **dichte-erreichbar**, wenn es eine Kette von direkt-dichte-erreichbaren Objekten von  $p$  nach  $q$  gibt.



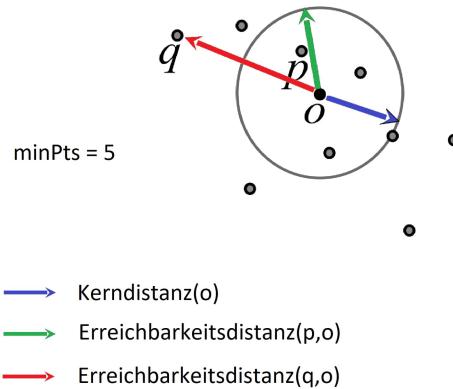
Zwei Objekte  $p$  und  $q$  sind **dichte-verbunden**, wenn es einen gemeinsamen Punkt  $o$  gibt, von dem aus sowohl  $p$  als auch  $q$  dichte-erreichbar ist.



Im Folgenden beschränke ich mich auf die hierarchische Variante des DBSCAN-Algorithmus, da Sie in der Praxis besonders auf Geodatensätzen bessere Ergebnisse erzielt und das Problem aus Abbildung 4.4 löst. Gegenüber der Grundvariante ist OPTICS allerdings rechenintensiver.

### Erweiterte Grundbegriffe

Die Kerndistanz  $kd_{\epsilon, \text{min}Pts}(o)$  ist genau dann definiert, wenn  $o$  ein Kernobjekt ist, also in der  $\epsilon$ -Nachbarschaft mindestens  $\text{min}Pts$  Punkte liegen. Sie entspricht dann dem kleinsten Radius um  $o$ , der erforderlich ist, um  $\text{min}Pts$  Punkte einzuschließen.



Die Erreichbarkeitsdistanz  $rd_{\epsilon, \text{min}Pts}(p, o)$  ist genau dann definiert, wenn  $o$  ein Kernobjekt ist. Sie entspricht dann der Kerndistanz von  $o$ , aber mindestens dem Abstand zwischen  $p$  und  $o$ .

Der OPTICS(Ordering Points To Identify the Clustering Structure)-Algorithmus [Ank+99] liefert bezüglich der Parameter  $\epsilon$  und  $\text{min}Pts$  eine Ordnung von Clustern in Form eines Erreichbarkeitsdiagramms. Dazu werden die Punkte so sortiert, dass ein nachfolgender Punkt im Diagramm zur Menge der bereits besuchten Objekte die kleinste Erreichbarkeitsdistanz besitzt.

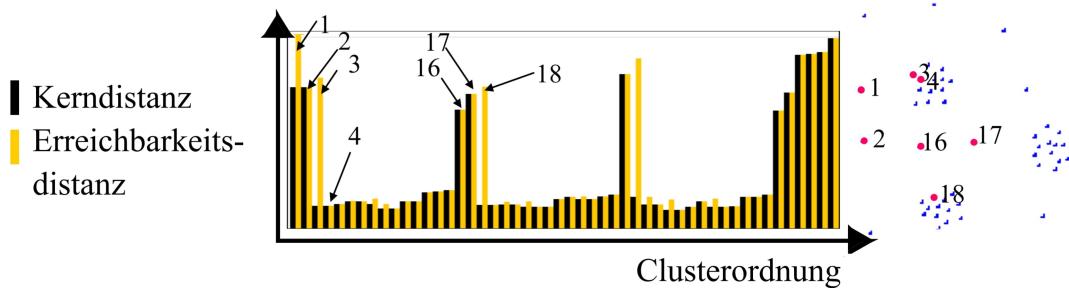


Abbildung 4.5: Erreichbarkeitsdiagramm (OrderedFile) nach Ausführung des OPTICS-Algorithmus<sup>3</sup>

### OPTICS-Algorithmus

Die äußere Schleife iteriert über jedes Objekt der Datenmenge  $D$ , welches noch nicht als Bearbeitet markiert wurde und ruft ExpandClusterOrder auf. Die Öffnung und Schließung des OrderedFiles dient dem Erstellen des Erreichbarkeitsdiagramms.

<sup>3</sup>[http://www-old.dbs\\_ifi.lmu.de/Lehre/KDD/WS0708/skript/kdd-5-clustering\\_02.pdf](http://www-old.dbs_ifi.lmu.de/Lehre/KDD/WS0708/skript/kdd-5-clustering_02.pdf)

```

1 OPTICS(Objektmenge D, Real  $\epsilon$ , Integer MinPts, OutputFile OrderedFile)
2   OrderedFile.open();
3   for i from 1 to D.size() do
4     Objekt := D.get(i);
5     if not Objekt.Bearbeitet then
6       ExpandiereClusterOrder(D, Objekt,  $\epsilon$ , MinPts, OrderedFile);
7   OrderedFile.close();

```

*ExpandClusterOrder* ist der eigentliche Algorithmus. Er bestimmt für den aktuell betrachteten Datenpunkt  $O$  die  $\epsilon$ -Nachbarschaft und Kerndistanz, markiert  $O$  als bearbeitet und fügt es dem Erreichbarkeitsdiagramm hinzu. Die zuvor definierte Undefiniertheit der Erreichbarkeit des ersten Punktes im *ExpandClusterOrder* wird als maximaler Wert (z.B.  $\infty$ )interpretiert. Wenn  $O$  ein Kernobjekt ist, wird *update(Nachbarn, O)* aufgerufen. Das bedeutet, dass auf der sogenannten Seeds-Liste, einer Pufferdatei, in welcher die Nachbarn des Zentrumsobjekt für Ihre Abarbeitung vorsortiert werden, operiert wird. Alle  $\epsilon$ -Nachbarn von  $O$  werden dieser Liste hinzugefügt, sofern sie noch nicht enthalten sind und aufsteigend nach ihrer Erreichbarkeitsdistanz zu  $O$  sortiert. Ist ein Nachbar schon enthalten, so wird sein Erreichbarkeitswert aktualisiert, wenn die neue Erreichbarkeitsdistanz kleiner ist als die Vorherige. Anschließend wird über eine innere Schleife iteriert. Hier werden alle Objekte der Pufferdatei ähnlich wie das Zentrumsobjekt abgearbeitet, bis kein Element mehr übrig ist. Wird *ExpandClusterOrder* verlassen, kann kein weiteres Objekt dem aktuellen Cluster zugeordnet werden und der Algorithmus wird mit einem zufälligen anderen Objekt fortgesetzt, das noch nicht bearbeitet wurde. Anschließend wird auch dessen Cluster expandiert.

```

1 ExpandClusterOrder(Objektmenge D, Objekt O,
2   Real  $\epsilon$ , Integer MinPts, OutputFile OrderedFile):
3   Nachbarn:= D.bestimmeNachbarschaft(O,  $\epsilon$ );
4   O.Erreichbarkeitsdistanz:= UNDEFINIERT;
5   O.setzeKerndistanz(Nachbarn,  $\epsilon$ , MinPts);
6   O.Bearbeitet:= TRUE;
7   OrderedFile.write(o);
8   if O.Kerndistanz  $\neq$  UNDEFINIERT then // Objekt O ist Kernobjekt
9     OrderSeeds.update(Nachbarn, O);
10    while OrderSeeds  $\neq$   $\emptyset$  do
11      aktObjekt := OrderSeeds.next();
12      Nachbarn := D.bestimmeNachbarschaft(aktObjekt, eps);
13      aktObjekt.setzeKerndistanz(Nachbarn, eps, MinPts);
14      aktObjekt.Bearbeitet := TRUE;
15      OrderedFile.write(aktObjekt);
16      if aktObjekt.Kerndistanz  $\neq$  UNDEFINIERT then
17        OrderSeeds.update(Nachbarn, aktObjekt)

```

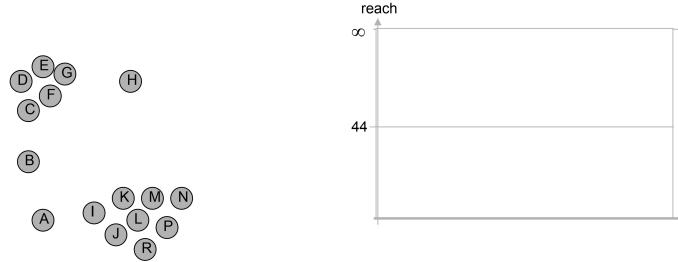
```

1 OrderSeeds :: update(Nachbarn, ZentrumsObjekt):
2   c_dist := ZentrumsObjekt.Kerndistanz;
3   for each Objekt from Nachbarn do
4     if not Objekt.Bearbeitet then
5       new r_dist := max(c_dist, ZentrumsObjekt.dist(Objekt));
6       if Objekt.Erreichbarkeitsdistanz == UNDEFINIERT then
7         Objekt.Erreichbarkeitsdistanz := new r_dist;
8         insert(Objekt, new r_dist);
9       else // Objekt ist schon in OrderSeeds enthalten
10      if new_r_dist < Objekt.Erreichbarkeitsdistanz then
11        Objekt.Erreichbarkeitsdistanz := new r_dist;
12        decrease(Objekt, new r_dist);

```

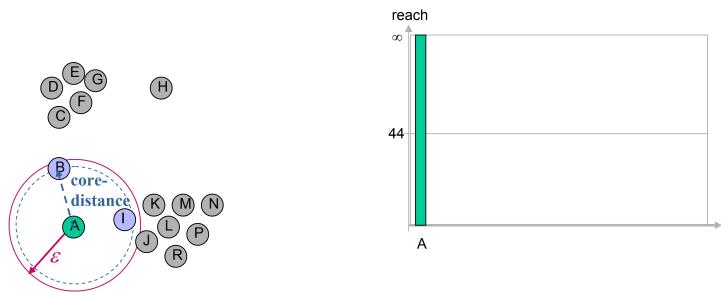
Es folgt eine Erklärung des Algorithmus in Form eines Beispiels:

Sei  $\epsilon = 44$  und  $minPts = 3$ . Die betrachtete Datenbank hat 16 zweidimensionale Punkte (von A - H). Zu Beginn sind alle Objekte unbearbeitet und das Erreichbarkeitsdiagramm ist leer.



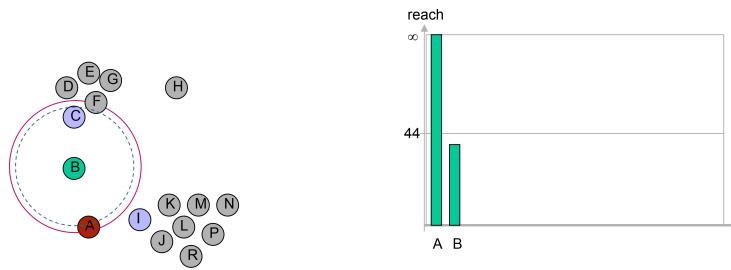
seed list:

Sei  $A$  der erste Punkt. Die Erreichbarkeitsdistanz ist somit undefiniert. Die Kerndistanz ist definiert, da innerhalb des Radius  $44$  die  $minPts = 3$  Punkte  $A, B$  und  $I$  liegen. Die Kerndistanz von  $A$  ist die Distanz zum am weitesten entfernten Punkt in der  $\epsilon$ -Nachbarschaft, also  $40$ . Jetzt wird der Punkt  $A$  dem Erreichbarkeitsdiagramm mit dem Wert  $\infty$  hinzugefügt und als bearbeitet markiert. Anschließend werden alle  $\epsilon$ -Nachbarn von  $A$  inklusive ihrer Erreichbarkeitsdistanzen der Seeds-Liste hinzugefügt (Zeile 9, *ExpandClusterOrder* erreicht).



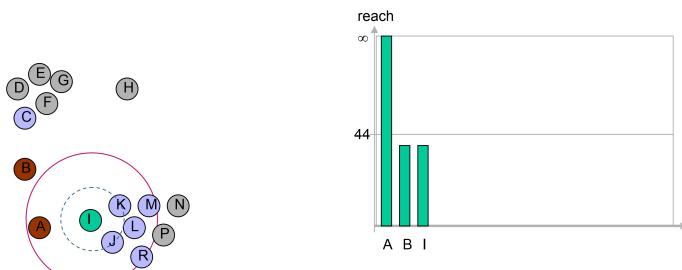
seed list: (B,40) (I, 40)

Im nächsten Schritt (Zeile 10) wird über alle Objekte der Seeds-Liste iteriert. Das vorderste Objekt ist der Punkt  $B$ , in dessen  $\epsilon$ -Nachbarschaft wieder drei Punkte liegen ( $A$ ,  $B$  und  $C$ ). Also ist auch  $B$  ein Kernobjekt und die Kerndistanz lautet 40. Nun wird Punkt  $B$  als bearbeitet markiert und seine Erreichbarkeit dem Diagramm hinzugefügt. Da das Objekt  $C$  noch nicht in der Seeds-Liste ist, wird es hinzugefügt.



seed list: (I, 40) (C, 40)

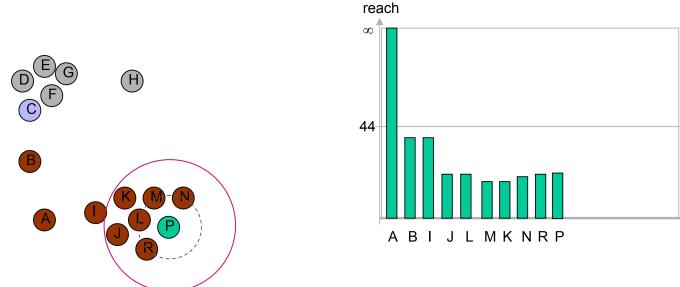
Das nächste Objekt der Seeds-Liste ist der Punkt  $I$ . Auch dieser ist wieder ein Kernobjekt und hat 7 Punkte in seiner  $\epsilon$ -Nachbarschaft. Davon werden  $K$ ,  $J$ ,  $M$ ,  $L$  und  $R$  der Seeds-Liste hinzugefügt.



seed list: (J, 20) (K, 20) (L, 31) (C, 40) (M, 40) (R, 43)

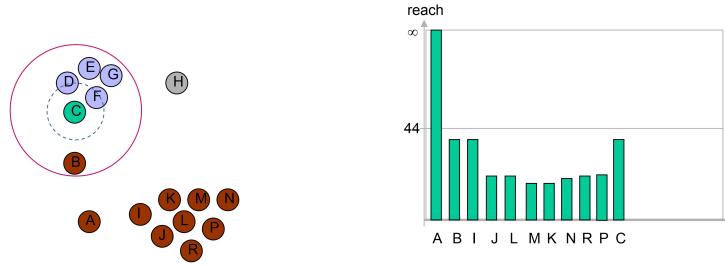
Der Algorithmus setzt sich fort, bis ein Objekt bearbeitet wurde, für welches keine weiteren

unbearbeiteten  $\epsilon$ -Nachbarn hinzugefügt werden konnte. In diesem Fall kommt es zu einem Peak im Erreichbarkeitsdiagramm, welcher den Beginn eines Subclusters kennzeichnet. Wird die Seeds-Liste leer und muss die äußere Schleife einen neuen zufälligen Startpunkt auswählen, beginnt ein neuer Cluster. In diesem Fall jedoch wird die innere Schleife fortgesetzt, da das Objekt  $C$  noch in der Seeds-Liste steht.



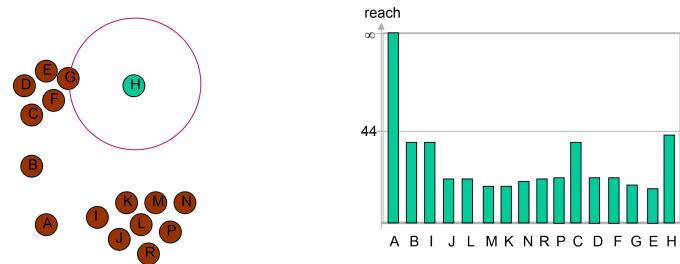
seed list: (C, 40)

Da  $C$  ein Kernpunkt ist, werden dessen  $\epsilon$ -Nachbarn  $D, F, E$  und  $G$  der Seeds-Liste hinzugefügt.



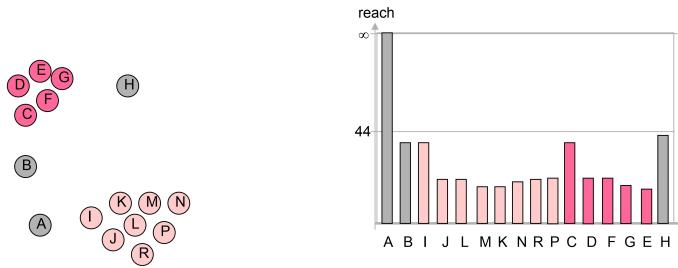
seed list: (D, 22) (F, 22) (E, 30) (G, 35)

Der Algorithmus setzt sich fort, bis es kein unbearbeitetes Objekt mehr gibt.



seed list: -

Im Diagramm können nun Cluster und Subcluster verschiedener Dichte, also verschiedener Erreichbarkeitsdistanzneaus abgelesen oder algorithmisch extrahiert werden. Dazu werden Teilsequenzen gesucht, welche in einem Gebiet steil abfallender Erreichbarkeitsdistanzen beginnen und in einem Gebiet steil ansteigender Erreichbarkeitsdistanzen enden, sodass der Endpunkt in etwa die gleiche Erreichbarkeitsdistanz besitzt, wie der Startpunkt. Enthält die Teilsequenz mindestens  $minPts$ -viele Punkte, so kann sie als Cluster interpretiert werden.



### Wahl der Parameter $\epsilon$ und $minPts$

Die Wahl der Parameter  $\epsilon$  und  $minPts$  sollte so erfolgen, dass die Cluster im Diagramm vertikal zentriert sind. Es dürfen also nur wenige Erreichbarkeitsdistanzen  $\infty$  sein. Außerdem darf die Varianz der Erreichbarkeitsdistanzen zwischen zwei benachbarten Punkten nicht zu groß werden. Um dies zu garantieren, wird  $\epsilon$  entweder so gewählt, dass die  $minPts$ -Eigenschaft bezüglich einer kleinen Punktestrichprobe immer erfüllt ist oder es wird die durchschnittliche  $minPts$ -Distanz berechnet. Es wird also diejenige  $\epsilon$ -Distanz gewählt, welche im Durchschnitt von allen Punkten aus  $minPts$ -viele Punkte einschließt. Der Wert für  $minPts$  muss in Abhängigkeit der Größe des Datensatzes gewählt werden. Je feiner die Auflösung der Datenbank, desto größer muss der Wert sein.

### Anwendungsgebiete

Die Werkzeugreferenz von **ArcGIS Pro**<sup>4</sup> gibt folgende Anwendungsszenarien für das Werkzeug *Dichte-basierte Cluster-Bildung* an:

- Clustering von Rohrbrüchen und Rissen städtischer Wasserversorgungssysteme kann auf bevorstehende Probleme hinweisen. Je nach Größe und Dicke eines Clusters können Gefahrenpotentiale abgeschätzt und vorbeugende Maßnahmen getroffen werden.
- Positionsdaten von erfolgreichen und nicht erfolgreichen Sportlern z.B. Torschützen beim Fußball clustern, um Muster zu erkennen und daraus neue Spielstrategien ableiten zu können.
- Angenommen es liegt ein Datensatz vor, der für ein begrenztes Untersuchungsgebiet (z.B. für eine Stadt) aufführt, ob ein Haushalt unter Schädlingsbefall leidet oder nicht. Mittels Clusterbildung können die größten und dichtesten Cluster betroffener Haushaltsgruppen identifiziert und behandelt werden. Dies steigert die Effizienz der Schädlingsbekämpfung.
- Das Verorten von Tweets infolge von Naturgefahren oder Terroranschlägen kann geclustert und notwendige Rettungsmaßnahmen und Evakuierungen können basierend

<sup>4</sup><https://pro.arcgis.com/de/pro-app/tool-reference/spatial-statistics/how-density-based-clustering-works.htm>

auf der Größe und Position des identifizierten Clusters vermittelt werden.

#### 4.2.4 Outlier Detection - Local Outlier Factor

Ein Problem der dichtebasierteren Clusteringverfahren ist, dass manche Punkte entweder als Outlier oder als Clusterpunkt markiert werden. Manchmal möchte man jedoch aus analytischen Gründen selbstständig beurteilen und steuern können, was als Outlier markiert werden soll. Aus diesem Grund wurde der Local Outlier Factor (LOF) erfunden, welcher für Punkt den Grad seiner Zuordenbarkeit berechnet [Bre+00].

##### Grundbegriffe

Die  $k$ -Distanz eines Punktes  $p$  ist der Radius  $kd(p)$ , für den mindestens  $k$  Punkte innerhalb oder auf ihm liegen müssen, aber maximal  $k - 1$  Punkte innerhalb liegen dürfen.

Die  $k$ -Distanz-Nachbarschaft  $N_{kd(p)}(p)$  ist die Menge aller Punkte, welche maximal den Abstand  $kd(p)$  von  $p$  haben.

Die Erreichbarkeitsdistanz  $rd(o, p)$  ist die  $k$ -Distanz von  $p$ , aber mindestens der Abstand  $dist(p, o)$ .

##### Berechnung von LOF

Zuerst wird die lokale Erreichbarkeitsdichte eines Punktes  $p$  berechnet. Diese entspricht dem Kehrwert der durchschnittlichen Erreichbarkeitsdistanzen von den Nachbarn aus:

$$Ird_{minPts}(p) = \left( \frac{\sum_{o \in N_{minPts}(p)} rd_{minPts}(p, o)}{|N_{minPts}(p)|} \right)^{-1}$$

Je höher der Wert, desto besser ist  $p$  von seinen Nachbarn aus erreichbar. Die Wahrscheinlichkeit, dass  $p$  ein Outlier ist, sinkt.

Auch die lokalen Erreichbarkeitsdichten der Nachbarn von  $p$  müssen mit einfließen, um beurteilen zu können, wie gut der Erreichbarkeit von  $p$  im Verhältnis zu seiner Nachbarschaft ist. Sind die Nachbarn viel besser zu erreichen als  $p$ , dann ist  $p$  ein starker Outlier:

$$LOF_{minPts}(p) = \frac{\sum_{o \in N_{minPts}(p)} \frac{Ird_{minPts}(o)}{Ird_{minPts}(p)}}{|N_{minPts}(p)|}$$

Ist der Wert von  $LOF$  in etwa 1, handelt es sich bei dem betrachteten Objekt um einen durchschnittlich ähnlich gut erreichbaren Punkt (im Vergleich zu seinen Nachbarn) und somit um einen Clusterpunkt. Für Werte, die größer als 1 sind, ist der Punkt deutlich schlechter erreichbar als seine Nachbarn und somit ein Outlier.

#### 4.2.5 Lineare Separation - Support Vector Machine

Die Support Vector Machine ist ein Klassifikator für linear und nichtlinear separierbare Daten. Er kommt in Geoinformationssystemen häufig zum Einsatz, da er besonders robust gegenüber Rauschen und Variabilität raumbezogener Daten ist [LS].

Die Grundidee der SVM ist es, ähnlich der Klassifikatoren neuronaler Netze eine optimal separierende Hyperebene zu finden. Ist der Datensatz nicht linear separierbar, so kann er

mittels Kernelfunktion in einen Raum transformiert werden, für den eine linear separierende Hyperebene gefunden werden kann [Ber].

Im Folgenden erkläre ich anhand eines Beispiels den zweidimensionalen Fall zur Berechnung der optimal separierenden Gerade.

Seien die roten und grünen Punkte in der Abbildung rechts die Objekte, welche den Klassen rot und grün zugehörig sind. Die Herausforderung ist jetzt, die Parameter der Gerade  $H_0$  ( $y$ -Achsenabschnitt und Steigung) so zu bestimmen, dass der Abstand zu den Randpunkten der beiden Klassen maximal wird. Zur Vereinfachung werden zwei weitere Geraden durch die Randpunkte beider Klassen gelegt, welche parallel zueinander stehen ( $H_1$  und  $H_2$ ). Die optimal separierende Gerade ist dann der Median beider Randgeraden.

Die Klassenlabels der Punkte oberhalb der Geraden  $H_1$  sind  $y_+ = +1$  bzw. unterhalb der Geraden  $H_2$   $y_- = -1$ . Sie sind also ein Teil ihrer Koordinate. Dies funktioniert, da sich die  $y$ -Werte für das Einstzen der Punkte  $x_+$  und  $x_-$  in die Geradengleichung von  $H_0$  mithilfe der Signumfunktion transformieren lassen. Gilt für einen Punkt  $x_i$ , dass  $\omega \cdot x_i + b \geq 0$ , dann gehört dieser zur Klasse rot. Es gilt gleichzeitig für alle Werte  $H_0(x_i) \geq 0$ , dass  $\text{sign}(H_0(x_i)) = +1 = y_i$ .

Setzt man die Labels in die Geradengleichungen der Geraden  $H_1$  und  $H_2$  ein, so erhält man die Ungleichungen

$$\omega \cdot x_i + b \geq +1 \quad (4.2)$$

und

$$\omega \cdot x_i + b \leq -1 \quad (4.3)$$

Für Punkte auf Gerade  $H_0$  hingegen gilt

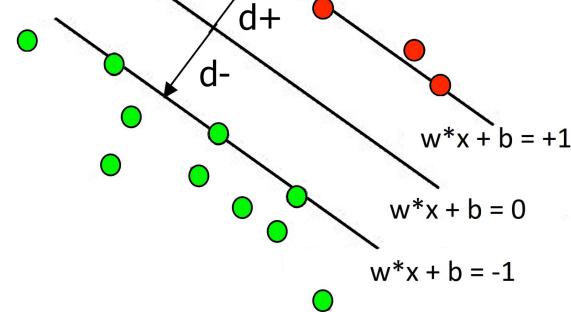
$$\omega \cdot x_i + b = 0$$

Der Abstand zwischen Randpunkten  $x_i$  und der optimal separierenden Gerade kann wie folgt berechnet werden: Der Abstand eines Punktes  $(x_0, y_0)$  von einer Geraden  $A \cdot x + B \cdot y + C = 0$  beträgt

$$\frac{A \cdot x_0 + B \cdot y_0 + C}{\sqrt{A^2 + B^2}}$$

Setzt man für den Punkt einen Randpunkt und für die Gerade die Gleichung von  $H_0$ , so ergibt sich

$$\frac{\omega \cdot x_+ + b}{\|\omega\|}$$



Da  $x_+$  auf  $H_1$  liegt, gilt  $\omega \cdot x_+ + b = 1$  und somit beträgt der Abstand zwischen  $x_+$  und  $H_0$

$$d_+ = \frac{1}{\|\omega\|}$$

Wenn  $\frac{1}{\|\omega\|}$  maximiert wird, dann muss  $\|\omega\|$  minimiert werden. Zur Vereinfachung der Ableitung kann man auch  $\frac{1}{2} \cdot \|\omega\|^2$  schreiben. Anschließend wird die Lagrange-Funktion aufgestellt. Unter der Voraussetzung, dass zwischen  $H_1$  und  $H_2$  keine Punkte liegen, kann man die Gleichungen (4.2) und (4.3) zusammenfassen:

$$y_i \cdot (\omega \cdot x_i + b) \geq 1 \quad (4.4)$$

Unter dieser Nebenbedingung lautet die Lagrange-Funktion:

$$\mathcal{L}(\vec{\omega}, b, \vec{\alpha}) = \frac{1}{2} \|\vec{\omega}\|^2 - \sum_{i=1}^n \alpha_i \cdot [y_i \cdot (\vec{\omega} \cdot \vec{x}_i + b) - 1] \quad (4.5)$$

Partielle Ableitungen gleich null setzen:

$$\frac{\partial \mathcal{L}}{\partial \vec{\omega}} = 0 \Leftrightarrow \vec{\omega} = \sum_{i=1}^n \alpha_i \cdot y_i \cdot \vec{x}_i \quad (4.6)$$

und

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Leftrightarrow \sum_{i=1}^n \alpha_i \cdot y_i = 0 \quad (4.7)$$

Nach Einsetzen von (4.6) in (4.5):

$$\mathcal{L}(\vec{\omega}, b, \vec{\alpha}) = \frac{1}{2} \left( \sum_{i=1}^n \alpha_i \cdot y_i \cdot \vec{x}_i \right) \cdot \left( \sum_{j=1}^n \alpha_j \cdot y_j \vec{x}_j \right) - \left( \sum_{i=1}^n \alpha_i \cdot y_i \cdot \vec{x}_i \right) \cdot \left( \sum_{j=1}^n \alpha_j \cdot y_j \cdot \vec{x}_j \right) - \sum_{i=1}^n \alpha_i \cdot y_i \cdot b + \sum_{i=1}^n \alpha_i \quad (4.8)$$

Nach Einsetzen von (4.7) in (4.8) ergibt sich folgende Gleichung:

$$\mathcal{L}(\vec{\omega}, b, \vec{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \frac{\vec{x}_i \cdot \vec{x}_j}{\langle \vec{x}_i, \vec{x}_j \rangle} \quad (4.9)$$

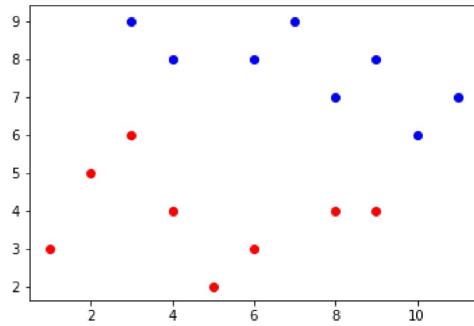
Diese lässt sich unter Nebenbedingung (4.7) und  $\alpha_i \geq 0$  maximieren.

Im Folgenden habe ich einen Optimierer (`scipy.optimize.minimize`) zur Maximierung von (4.9) für ein selbstgewähltes Beispiel konkret in Python implementiert. Die Maximierung von  $\mathcal{L}$  entspricht der Minimierung von  $-\mathcal{L}$ :

```
In [1]: import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import minimize

In [2]: x1 = np.asarray([1,2,3,4,5,6,8,9])
y1 = np.asarray([3,5,6,4,2,3,4,4])
x2 = np.asarray([3,4,6,7,8,9,10,11])
y2 = np.asarray([9,8,8,9,7,8,6,7])

In [3]: fig, ax = plt.subplots()
ax.scatter(x1,y1,color='red')
ax.scatter(x2,y2,color='blue')
```



```
In [4]: x = np.asarray([[3,6],[9,4],[4,8],[10,6]],dtype=np.float64)
y = np.asarray([-1,-1,1,1],dtype=np.float64)

In [5]: #Definition der zu maximierenden Lagrangefunktion
def dualOpt(a):
    sumsum = 0
    for i in range(0,len(x)):
        for j in range(0,len(x)):
            sumsum += a[i]*a[j]*y[i]*y[j]*np.dot(x[i],x[j])
    L = np.sum(a) - (1/2)*sumsum
    return -L

In [6]: #Definition der Nebenbedingung sum(alpha_i*y_i) = 0
a = [0,0,0,0]
def constraint1(a):
    return a[0]*y[0]+a[1]*y[1]+a[2]*y[2]+a[3]*y[3]
con1 = {'type': 'eq', 'fun': constraint1}
res = minimize(dualOpt, a, constraints=[con1],options={'disp': True})
print(res.x) #alpha_i

Optimization terminated successfully.      (Exit mode 0)
      Current function value: -0.40816326530605496
      Iterations: 3
      Function evaluations: 21
      Gradient evaluations: 3
[0.19387752  0.21428558  0.21428567  0.19387743]
```

```
In [7]: #Gewichtsvektor w berechnen
w = 0
for i in range(0,len(x)):
    w += (res.x[i]*x[i]*y[i])
print(w)

[0.28571416  0.85714251]
```

```
In [8]: d = 1/np.linalg.norm(w)
d #Dies ist der Abstand zwischen H0 und H1, bzw H0 und H2
#Nun lässt sich daraus ganz einfach
#(z.B. mithilfe der Hesseschen Normalform von H1 oder H2)
#ein Punkt auf H0 z.B. p0 = (1 / 7.87)
#und somit auch der y-Achsenabschnitt (Parameter b = 8.17) berechnen.

Out[8]: 1.1067976307258798

In [9]: #Länge von Normalenvektoren für Hessesche Normalform
#1/3 und 1 sind die Faktoren vor x1 und x2 der Hesseschen Normalform von H0.
#Diese wurden von H1 und H2 übernommen
np.linalg.norm([1/3,1])

Out[9]: 1.0540925533894598

In [10]: #Für H0 ergibt sich mit
#(-(1/3)*x1-x2+8.17)/1.054 = 0
#die optimal separierende Gerade
```

## 4.3 Fuzzy-Clustering

### 4.3.1 Fuzzy C-Means (FCM)

Neuere Methoden der letzten Jahre verwenden häufig modifizierte Versionen des FCM(Fuzzy C-Means)-Algorithmus[Roy13]. Dieser berechnet für jeden Punkt den Grad seiner Zuordnung zu allen Clustern. Er gehört zu den probabilistischen partitionierenden Clusteringverfahren. Die zu minimierende Kostenfunktion lautet:

$$J_m(X, U, V) = \sum_{i=1}^c \sum_{k=1}^n u_{ik}^m \cdot d^2(v_i, x_k) \quad \forall i, k \in [0, 1]$$

Wobei  $u_{ik}$  die Eingangs beschriebene Zugehörigkeitsmatrix ist und  $d^2$  ein Abstandsmaß zur Gewichtung der Zugehörigkeit in Abhängigkeit des Abstandes zwischen Punkt  $x_k$  und Clusterzentroid  $v_i$ . Parameter  $m > 1$  ist die Vagheit der Zerlegung und gibt an, wie stark die Zugehörigkeitsgrade abfallen. In der Regel wird hier der Wert  $m = 2$  verwendet. Dabei sind folgende Nebenbedingungen Einzuhalten:

$$\forall k \in 1, \dots, n : \sum_{i=1}^c u_{ik} = 1$$

Dies sorgt dafür, dass jedes Objekt das Gesamtgewicht 1 erhält und somit kein Punkt bevorzugt wird.

$$\forall i \in \{1, \dots, c\} : \sum_{i=1}^n u_{ik} > 0$$

Dies garantiert, dass es keinen Cluster gibt, dem nichts zugeordnet wurde. Es muss also mindestens ein Objekt geben, das eine positive Zugehörigkeit zu Cluster  $i$  hat.

Zur Optimierung der Zielfunktion, bildet man die partiellen Ableitungen der Lagrangefunktion

$$\mathcal{L}(X, U, V, \Lambda) = \sum_{i=1}^c \sum_{k=1}^n u_{ik}^m \cdot d^2(v_i, x_k) + \sum_{k=1}^n \lambda_k \left( \sum_{i=1}^c u_{ik} - 1 \right)$$

Man erhält die zu aktualisierenden Werte von Zugehörigkeitsmatrix und Clusterzentroiden:

$$u_{ik} = \frac{1}{\sum_{j=1}^c \left( \frac{d^2(v_i, x_k)}{d^2(v_j, x_k)^{\frac{1}{m-1}}} \right)} \quad (4.10)$$

und

$$v_i = \frac{\sum_{k=1}^n u_{ik}^m \cdot x_k}{\sum_{k=1}^n u_{ik}^m} \quad \forall i \in \{1, \dots, c\} \quad (4.11)$$

Der Algorithmus zum Finden der besten Clusterzentroiden ist wie beim  $k$ -means Verfahren ein alternierendes Programm.

---

#### Algorithm 1 $FCM(X, c, m, \epsilon)$

---

**Require:**  $X$  ist ein  $d$ -dimensionaler Datensatz mit  $n$  Datenpunkten,  $2 \leq c \leq n$  ist die Clusteranzahl,  $m > 1$  ist die Vagheit und  $\epsilon > 0$  ist die Terminierungsgenauigkeit

- 1: Initialisiere die Clusterprototypen  $v' = \{v'_1, \dots, v'_c\}$
  - 2:  $v = \emptyset$
  - 3: **repeat**
  - 4:      $v = v'$
  - 5:     Berechne die Zugehörigkeitsgrade  $u_{ik}$  von jedem Datenpunkt  $x_k$  zu jedem Cluster  $C_i$  nach Formel (4.10) // Schritt 1
  - 6:     Berechne die neuen Clusterprototypen  $v' = \{v'_1, \dots, v'_c\}$  nach Formel (4.11) // Schritt 2
  - 7: **until**  $\|v - v'\| < \epsilon$
  - 8: **return**  $v'$
- 

Leider hat das ursprüngliche FCM-Verfahren viele Nachteile. Man weiß leider nie, ob ein lokales oder ein globales Minimum für die Kostenfunktion gefunden wurde. Je nach Initialisierungsmethode muss FCM unter Umständen mehrmals mit unterschiedlichen Parametern ausgeführt werden, um das beste Ergebnis zu finden. Wird für das Distanzmaß die euklidische Distanzfunktion verwendet, können nur gleich große, sphärische Cluster gefunden werden. Aus diesem Grund wurden viele weitere Fuzzy-Clusteringalgorithmen entwickelt, die auf dem gleichen Grundprinzip basieren, jedoch andere Kostenfunktionen minimieren.

Da die Zugehörigkeiten der Datenobjekte relativ sind, können Outlier fälschlicherweise überberücksichtigt werden:

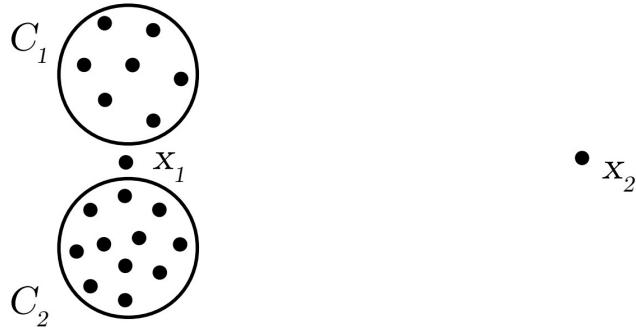


Abbildung 4.6: Punkt  $X_1$  hat den gleichen Zugehörigkeitsgrad wie der Outlier  $X_2$

### 4.3.2 Possibilistic C-Means (PCM)

Eine Lösung bieten possibilistische Clusteringverfahren. Werden die relativen Zugehörigkeiten  $u_{ik}$  durch absolute Ähnlichkeitswerte  $t_{ik}$  (typicality values) ersetzt, so fällt die Nebenbedingung  $\sum_{i=1}^c u_{ik} = 1$  weg. Die Gesamtzugehörigkeit jedes Datums zu allen Clustern wird also nicht mehr auf alle Cluster verteilt. Stattdessen soll ein Wert zwischen 0 und  $c$  gewählt werden:

$$0 < \sum_{i=1}^c t_{ik} \leq c$$

Eine mögliche Lösung für die Kostenfunktion aus FCM mit Ähnlichkeitswerten wäre trivial. Sei  $\epsilon > 0$ :

$$\forall k \exists i : t_{1k} = 0, \dots, t_{i-1k} = 0, t_{ik} = \epsilon, t_{i+1k} = 0, \dots, t_{ck} = 0$$

Es muss also nur ein einziger Ähnlichkeitswert größer als 0 sein. Um diese triviale Lösung zu verhindern, führt man einen Bestrafungsterm ein:

$$\sum_{i=1}^c \gamma_i \sum_{k=1}^n (1 - t_{ik})^m, \quad \gamma_i > 0$$

Werden die typicality Values zu klein, so wird der Bestrafungsterm immer größer. Der Parameter  $\gamma_i$  regelt dabei, wie stark der Einfluss der Bestrafung für den jeweiligen Cluster  $C_i$  ausfallen darf. Außerdem beeinflusst er die Wahl von  $t_{ik}$  indirekt. So konvergiert  $t_{ik}$  für sehr große Werte von  $\gamma_i$  gegen den Wert 1. Entspricht  $\gamma_i$  dem Wert der Distanzfunktion  $d^2(v_i, x_k)$ , so hat  $t_{ik}$  den Wert  $\frac{1}{2}$ .

Die Kostenfunktion für PCM lautet insgesamt:

$$J_m(T, V, X, \gamma) = \sum_{i=1}^c \sum_{k=1}^n t_{ik}^m \cdot d^2(v_i, x_k) + \sum_{i=1}^c \gamma_i \sum_{k=1}^n (1 - t_{ik})^m, \quad \gamma_i > 0 \forall i$$

Leider führt auch diese Funktion je nach Initialisierung der Clusterprototypen zu einer trivialen Lösung, bei der nur noch ein einziger Cluster erkannt wird. Mögliche Lösungen sind die Initialisierung mit der Ausgabe eines zuvor ausgelösten FCM-Durchlaufs oder das Hinzufügen eines weiteren Bestrafungsterms, der Clusterabstoßung. Je näher sich die

Clusterzentroiden im Laufe der Optimierung kommen, desto größer soll die Kostenfunktion werden. Gleichzeitig sollen die Punkte nicht nur einem Cluster zugeordnet und alle anderen Cluster möglichst weit weg sein. Zwei Abstoßungsfunktionen [Tim+04], die solche Kriterien erfüllen, sind:  $f(x) = \frac{1}{x^2}$  und  $g(x) = \frac{1}{e^{x^2}}$ .

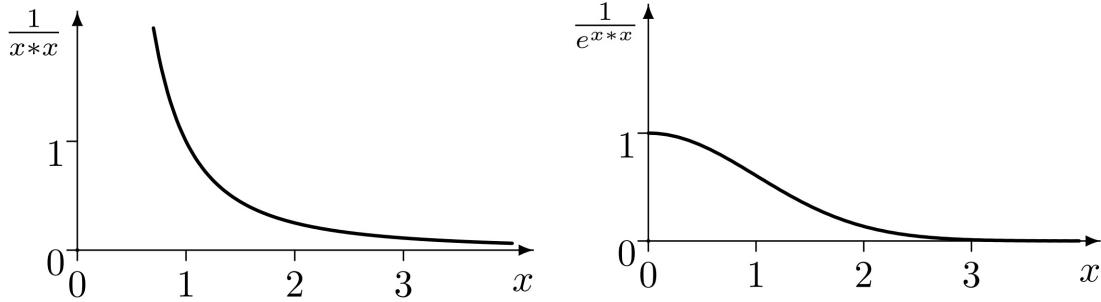


Abbildung 4.7: Geeignete Abstoßungsfunktionen: Je größer der Clusterabstand, desto kleiner die Belohnung in der Kostenfunktion

$f(x)$  führt zu Kostenfunktion

$$\begin{aligned} J_m(T, V, X, \gamma, \eta, \zeta) &= \sum_{i=1}^c \sum_{k=1}^n t_{ik}^m \cdot d^2(v_i, x_k) + \sum_{i=1}^c \gamma_i \sum_{k=1}^n (1 - t_{ik})^m + \\ &\quad \sum_{i=1}^c \eta_i \sum_{j=1, j \neq i}^c \frac{1}{\zeta \cdot d^2(v_i, v_j)}, \quad \gamma_i > 0 \forall i, \zeta \in \mathbb{R}_+ \end{aligned} \quad (4.12)$$

und  $g(x)$  führt zu

$$\begin{aligned} J_m(T, V, X, \gamma, \eta, \zeta) &= \sum_{i=1}^c \sum_{k=1}^n t_{ik}^m \cdot d^2(v_i, x_k) + \sum_{i=1}^c \gamma_i \sum_{k=1}^n (1 - t_{ik})^m + \\ &\quad \sum_{i=1}^c \eta_i \sum_{j=1, j \neq i}^c e^{-\zeta \cdot d^2(v_i, v_j)}, \quad \gamma_i > 0 \forall i, \zeta \in \mathbb{R}_+ \end{aligned} \quad (4.13)$$

Dabei normiert  $\zeta$  den Abstand zwischen je zwei Clusterprototypen und  $\eta_i$  setzt den Abstand eines Clusters zu den benachbarten Clustern in Relation zu den Abständen zu Datenpunkten innerhalb des Clusters. Ein clusterspezifisches Abstoßungsmaß ist erforderlich, damit große Cluster nicht stärker berücksichtigt werden, als kleine Cluster. Nach Minimierung der Kostenfunktion ergeben sich für die Aktualisierungen der Ähnlichkeitswerte  $t_{ik}$  und Clusterzentroide  $v_i$  folgende Formeln:

$$t_{ik} = \frac{1}{1 + \left( \frac{d^2(v_i, x_k)}{\gamma_i} \right)^{\frac{1}{m-1}}}, \quad 1 \leq i \leq c, 1 \leq k \leq n$$

und

$$v_i = \frac{\sum_{k=1}^n t_{ik}^m x_k - \eta_i \zeta \sum_{j=1, j \neq i}^c v_j e^{-\zeta d^2(v_j, v_i)}}{\sum_{k=1}^n t_{ik}^m x_k - \eta_i \zeta \sum_{j=1, j \neq i}^c e^{-\zeta d^2(v_j, v_i)}}, \quad 1 \leq i \leq c$$

und  $\gamma_i$  wird auf die intra-cluster Distanz gesetzt:

$$\gamma_i = K \cdot \frac{\sum_{k=1}^n u_{ik}^m d^2(x_k, v_i)}{\sum_{k=1}^n u_{ik}^m}, \quad K > 0$$

#### 4.3.3 Possibilistic Fuzzy C-Means (PFCM)

Eine erweiterte Kombination aus den FCM und PCM Algorithmen kommt ganz ohne Clusterabstoßung aus und verhindert, dass die absoluten Zugehörigkeitsgrade bei großen Datensätzen zu klein werden und ihre Aussagekraft verlieren. Die Zielfunktion von PFCM[Pal+05] lautet:

$$J_{m,\eta}(U, T, V, X, \gamma) = \sum_{i=1}^c \sum_{k=1}^n (a \cdot u_{ik}^m + b \cdot t_{ik}^\eta) \cdot d^2(v_i, x_k) + \sum_{i=1}^c \gamma_i \sum_{k=1}^n (1 - t_{ik})^\eta, \quad (4.14)$$

$$m > 1, n > 1, a > 0, b > 0, 0 \leq u_{ik}, t_{ik} \leq 1$$

Dabei steuern  $a$  und  $b$  den relativen Einfluss der verschiedenen Arten von Zugehörigkeitsgraden. Je ungeeigneter die typicality Values ( $t_{ik}$ ) in Anbetracht der Größe des Datensatzes sind, desto stärker wird die Gewichtung bei den Zugehörigkeitsgraden ( $u_{ik}$ ). Mit  $a = 0$  wird PFCM zu PCM und mit  $b = 0$  zu FCM.

#### 4.3.4 Scale Space Filter Clustering

Eine weitere Variante des FCM-Algorithmus, der Selective Scale Space based FCM wurde entworfen, um den Clusteringprozess raumbezogener Daten zu verbessern.

Gegeben einer Menge von Objekten kann es vorkommen, dass sich bestimmte Objektfeatures nicht clustern lassen. Dies ist der Fall, wenn sich die Werte eines Featurevektors zu ähnlich sind und keine deutlichen Cluster bilden. Man sagt, dass die Cluster nicht linear separabel sind. Der Scale Space Filter löst dieses Problem, indem er mithilfe einer Transformationsvorschrift nah beieinander liegende Punkte weiter zusammen schiebt und weiter entfernte Punkte voneinander abstößt. Wird der Filter jedoch auf bereits für das Clustering geeignete Parameter angewendet, kann der Informationsgehalt verschlechtert werden. Aus diesem Grund müssen zuerst Features selektiert werden, welche für eine Vorverarbeitung geeignet sind und somit das Potential haben, nach ihrer Transformation die Clusteringgenauigkeit zu verbessern. Ein weiterer Vorteil des Selektionsprozesses ist die Optimierung der Laufzeiteffizienz. Ein Parameter ist genau dann geeignet, wenn seine Standardabweichung einen bestimmten Schwellwert unterschreitet. Das Maß für die Separation und Ähnlichkeit bietet der Xie Beni Cluster Validity Index. Er setzt die Distanzen zwischen Datenpunkten und Clusterzentroiden ins Verhältnis zu den Distanzen zwischen den Clustern:

$$V_{XB}(U, X, V) = \frac{\sum_{i=1}^c \sum_{k=1}^n u_{ik}^2 \cdot \|x_k - v_i\|^2}{n \cdot \min_{1 \leq i, j \leq c, i \neq j} \|v_i - v_j\|^2}$$

Es fließt also ein, wie kompakt die einzelnen Cluster sind und wie stark die topologische Unähnlichkeit bzw. Separation zwischen verschiedenen Cluster ist.

Sei  $X$  eine Datenmatrix der Dimensionalität  $m \times n$ . Es gibt also  $n$  Objekte mit je  $m$  Parametern.

Ein Parameter  $j$  ist genau dann krank, wenn für den Feature-Vektor  $x_{ij} \forall i \in \{1, \dots, n\}$  gilt:

$$\theta - \epsilon \leq x_{ij} \leq \theta + \epsilon$$

für ein  $\theta$  und einen sehr kleinen Wert  $\epsilon > 0$ . In diesem Fall wird der Parameter mit einer Transformationsfunktion modifiziert. Möglich sind hier eine Vielzahl von Funktionen. In der Publikation beschränken sich die Autoren jedoch auf die Gaußsche Transformationsformel:

$$f(x, \sigma) = \frac{1}{(\sigma \cdot \sqrt{2\pi})^2} e^{-\frac{\|x - x_j\|^2}{2\sigma^2}}$$

Diese wird auf jeden Parameter einzeln angewandt. Anschließend werden die neuen Werte normalisiert:

$$x[j] = \frac{x_{ij} - \min(X[j])}{\max(x[j]) - \min(x[j])}, \quad \forall x_{ij} \in X[j]$$

Hierbei ist  $x[j]$  der  $j$ -te Featurevektor,  $x_{ij}$  der Wert von Parameter  $j$  von Datum  $i$ .

Der Pseudocode für den gesamten Algorithmus sieht dann folgendermaßen aus:

---

**Algorithm 2** S.D. based Selective Scale Spaced FCM

---

**Input:**  $X = \{x_1, \dots, x_n\}$ , standardAbweichungGrenzwert  
**Output:** Clustering  $C$

- 1: Deklariere Zugehörigkeitsmatrix  $u_{ik}$  mit  $i \in \{1, \dots, c\}$  und  $k \in \{1, \dots, n\}$ .
- 2: Deklariere Clustervaliditätsindex (Hier  $V_{XB}$ )
- 3: **for**  $j$  in  $X_i$  **do**
- 4:      $sd \leftarrow \text{berechneStandardabweichung}(j)$
- 5:     **if**  $sd < \text{standardAbweichungGrenzwert}$  **then**
- 6:          $j \leftarrow \text{gaussianScaleSpace}(j)$
- 7:     **end if**
- 8: **end for**
- 9:  $V \leftarrow \text{initialisiereClusterprototypen}(V)$
- 10: **while**  $V_{XB} \leq \text{Validitätsgrenze}$  **do**
- 11:      $U \leftarrow \text{berechneZugehörigkeitsmatrix}(X, U, V)$
- 12:      $V \leftarrow \text{aktualisiereClusterzentroide}(U, V)$
- 13:      $V_{XB} \leftarrow \text{aktualisiereXieBeniIndex}(U, V)$
- 14: **end while**

---

### 4.3.5 Graphbasiertes Clustering I (NSCABDT)

Eine weitere FCM-Variante, die speziell zum Clustering von geographischen Strukturen entwickelt wurde, basiert auf der Delaunay Triangulation, also der planaren Dekomposition einer Punktemenge [YC10]. Dies war nötig, da bisherige Entwicklungen es nicht geschafft hatten, räumliche Abhängigkeiten und Heterogenitäten zu erkennen. Beide gelten als starke Indikatoren für geographische Prozesse. Dabei bezeichnet die räumliche Abhängigkeit die Tendenz zweier Punkte, sich mit zunehmender räumlicher Nähe gegenseitig zu beeinflussen oder ähnliche Attributwerte anzunehmen [Goo92]. Räumliche Heterogenität bezeichnet die Ungleichverteilung von Populationen auf einem bestimmten Gebiet. Ein weiterer Vorteil der Methode ist, dass nun auch Cluster erkannt werden können, die durch eine schmale Brücke verbunden sind. Da eine Delaunay-Triangulation viele Ähnlichkeitsinformationen implizit speichert, eignet sie sich für eine robuste Clusteranalyse. Außerdem werden Delaunay-Triangulationen bei der Analyse von Topologieproblemen, Autokonturierung, der Visualisierung von 2.5D Geodaten, Oberflächencharakterisierung und in vielen weiteren raumbezogenen Analysewerkzeugen eingesetzt.

Eine Delaunay-Triangulation  $D(X)$  ist eine Vernetzung aller Punkte aus  $X$  zu Dreiecken, so dass kein Punkt  $p \in X$  innerhalb des Umkreises eines dieser Dreiecke liegt. Dabei wird der kleinste Innenwinkel über alle Dreiecke maximiert.

Ein einfacher Algorithmus zur Erzeugung der Delaunay-Triangulation ist die inkrementelle Konstruktion: Zu Beginn muss ein Dreieck gefunden werden, welches die Delaunay-Bedingung (DB) erfüllt. Anschließend werden alle anderen Punkte inkrementell hinzugefügt. Der nächste Punkt aus  $X$  wird  $D(X)$  hinzugefügt und mit allen Punkten des initialen Dreiecks verbunden. Es entstehen drei weitere Dreiecke, die die DB eventuell verletzen. Jedes der Dreiecke wird auf die Umkreisbedingung getestet. Schließt ein Umkreis einen Punkt eines angrenzenden Dreiecks ein, so wird die gemeinsame Kante gelöscht und eine Kante zwischen den beiden Punkten erzeugt, die vorher nicht verbunden waren:

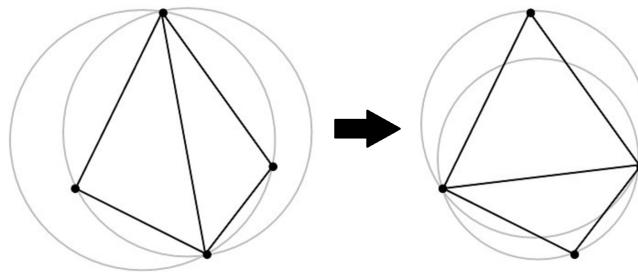


Abbildung 4.8: Flip einer gemeinsamen Kante zur Erfüllung der Delaunay-Bedingung

Anschließend existieren weitere Dreiecke, welche die Bedingung nicht erfüllen könnten. Es werden nacheinander alle Dreiecke überprüft und gegebenenfalls Flips durchgeführt, bis die Bedingung global erfüllt ist.

Während der Triangulierung werden Informationen zu Knoten, Kanten und Oberflächen in einer Datenbank für das spätere Clustering gespeichert:

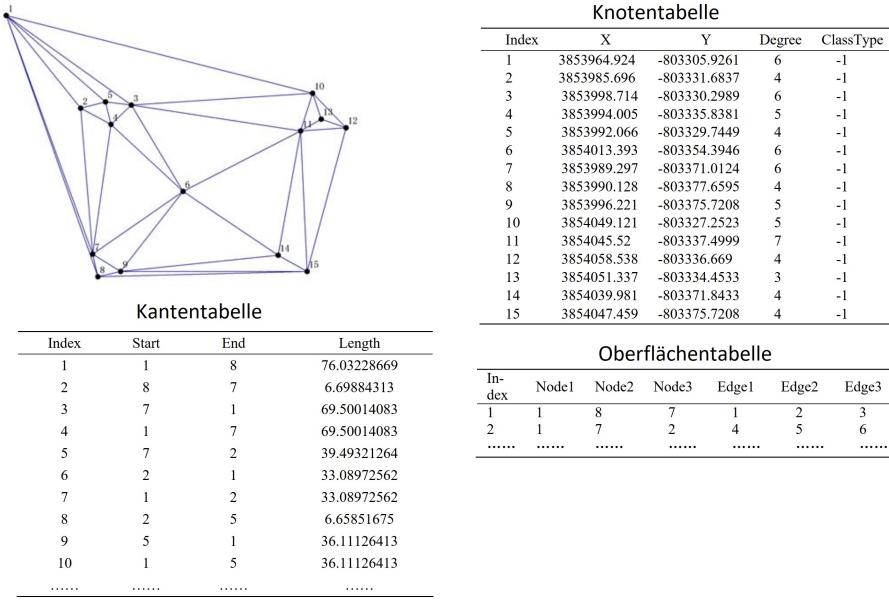


Abbildung 4.9: Beispiel einer DT von 15 Datenpunkten mit Tabellen für sämtliche Informationen

Die Knotentabelle enthält zu jedem der 15 Punkte die Koordinaten, den Knotengrad und die Kategorie bzw. die Clusternummer ( $-1 = \text{Outlier}$ ).

Die Kantentabelle enthält Infos zur Richtung der Kanten, wobei  $\text{Start} = 1$  und  $\text{End} = 8$  bedeutet, dass von Knoten 1 eine gerichtete Kante Richtung Knoten 8 zeigt. Außerdem besitzt sie die Länge 76.

Die Oberflächentabelle enthält alle Knoten und Kanten, welche an den jeweiligen Ebenen (Dreiecken) beteiligt sind.

Im Gegensatz zur gewöhnlichen Delaunay Triangulation sollen in diesem Verfahren nur solche Knoten verbunden werden, die näher beieinander liegen, als ein vorgegebenes Ähnlichkeitsmaß.

### Bezeichnungen

Im Folgenden werden Kanten  $e_k$  geschrieben als  $(p_i, p_j)$ , wobei  $p_i$  der Startpunkt ist und  $p_j$  der Endpunkt.  $N(p_i)$  bezeichnet die Menge aller eingehenden und Ausgehenden Kanten von Knoten  $p_i$ . Die lokale Durchschnittslänge  $local\_mean(p_i)$  ist die durchschnittliche Länge aller Kanten in  $N(p_i)$ :

$$local\_mean(p_i) = \frac{\sum_{j=1}^{deg(p_i)} len(e_j)}{deg(p_i)}$$

Die globale Durchschnittslänge ist definiert als Durchschnittslänge aller Kanten in  $E$ .

$$global\_mean(X) = \frac{\sum_{j=1}^{|E|} len(e_j)}{|E|}$$

Die globale Standardabweichung ist die durchschnittliche Abweichung des *global\_mean* von allen Kantenlängen:

$$\text{global\_sta\_dev}(X) = \frac{\sqrt{\sum_{i=0}^n (\text{global\_mean}(X) - \text{len}(e_i))^2}}{n}$$

Der relative Durchschnitt ist das Verhältnis von lokalem zu globalem Durchschnitt:

$$\text{relative\_mean}(p_i) = \frac{\text{local\_mean}(p_i)}{\text{global\_mean}(X)}$$

Eine Kante wird als positive Kante bezeichnet, wenn sie kleiner ist als das Kriterium  $F(p_i)$ . Positive Kanten und dazu inzidente Punkte formen einen neuen Wahrscheinlichkeitsgraphen, der gleichzeitig Subgraph der Delaunay-Triangulation ist. Das Kriterium berechnet sich wie folgt:

$$F(p_i) = \text{global\_mean}(X) + \text{global\_sta\_dev}(X) \times \frac{\text{global\_mean}(X)}{\text{local\_mean}(p_i)}$$

Ein positiver Pfad ist ein Pfad von positiven Kanten und alle verbundenen Punkte gehören zu einem Cluster.

Die effektive Region des Punktes  $p$  mit Radius  $\delta$  ist wie folgt definiert:

$$p_\delta = \{x \mid |x - p| < \delta, p \in X, \forall x \in \mathbb{R}^2\}$$

Die effektive Region einer Punktmenge ist dann die Vereinigung der effektiven Regionen aller Punkte dieser Menge:

$$\overline{X} = \bigcup_{p \in X} p_\delta$$

Die  $\gamma$ -Grenze ist die Menge aller Punkte im Kreis mit Radius  $\delta$

$$V_\gamma = \overline{V_\gamma} \cap V, \quad \overline{V_\gamma} = \frac{\overline{V}}{\{\overline{V} \Theta \gamma Dm(\delta)\}}, \quad \gamma > 1$$

mit

$$Dm = \{x \mid |x| \leq \delta\}$$

Die  $\gamma$ -Kurve ist die Mannigfaltigkeit des Punktes in der  $\gamma$ -Grenze einer Punktemenge.

Der Algorithmus für NSCABDT kann wie folgt beschrieben werden:

Zu Beginn gehört kein Punkt einer gegebenen Punktemenge  $S$  zu einem Cluster. Außerdem ist die Menge der Punkte des ersten Clusters  $C$  leer:  $C = \emptyset$

1. Delaunay-Triangulation berechnen und dabei die Knoten, Kanten und Flächentabellen ausfüllen
2. Für jeden Punkt  $p_i \in S$  alle inzidenten Kanten  $N(p_i)$  finden,  $\text{local\_mean}(p_i)$  und  $F(p_i)$  berechnen.
3. Für jede Kante  $e \in N(p_i)$ : Wenn  $\text{len}(e) \geq F(p_i)$ , dann Kante aus DT entfernen
4. Wenn  $\text{deg}(p_i) = 0$ , Knoten  $p_i$  löschen, ansonsten zu  $C$  hinzufügen.

5. Iterativ für alle Knoten durchführen, mit denen  $p_i$  verbunden ist.
6. Grenze von Cluster C extrahieren und alle Brücken eliminieren. Dies erfolgt per Hauptkurvenanalyse, welche eine Generalisierung der Hauptachsenanalyse ist. Wenn für einen Cluster zwei verschiedene Grenzen gefunden werden kann, ist dies ein Indiz dafür, dass es zwei kleinere Cluster gibt, die über eine Brücke verbunden sind. Der Algorithmus zur Eliminierung von Brücken lautet wie folgt:
  - (a) Setze  $\delta$  auf die Medianlänge aller Kanten  $E$
  - (b) Berechne die effektive Region einer Punktemenge  $V$
  - (c) Berechne die  $\gamma$ -Grenze von  $V$
  - (d) Berechne die  $\gamma$ -Kurve von  $V$
7. Gibt es noch unbearbeitete Knoten, obwohl im Iterationsprozess alle Knoten positiver Pfade betrachtet wurden, dann konnten nicht alle Knoten erreicht werden und es mussten zu lange Kanten gelöscht werden (negative Pfade). Es wird ein neuer Cluster  $C'$  erstellt und die Iteration mit einem beliebigen unbearbeiteten Punkt forgesetzt.

### Clusteringergebnisse

In der Originalpublikation wird untersucht, wie sich der NSCABDT-Algorithmus beim Clustern von drei verschieden geformten Polygonen verhält. Diese sollen aus mehreren Hundert Punkten bestehen und von Outliern umgeben sind. Außerdem sind zwei Polygone über mehrere Brücken verbunden.

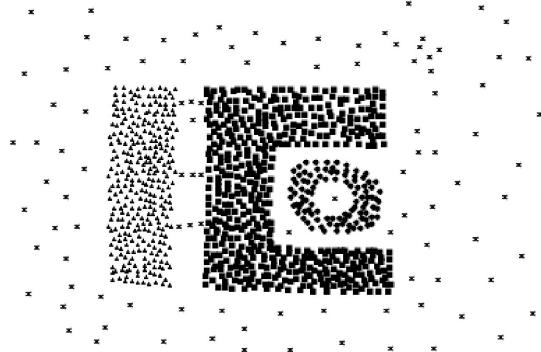


Abbildung 4.10: Beispieldatensatz mit drei Clustern

Im direkten Vergleich mit DBSCAN ist der NSCABDT-Algorithmus zwar viel langsamer, findet jedoch in jedem Untersuchten Fall die korrekten Cluster mit etwas höherer Genauigkeit.

Number of Points	5000		6000		7000		8000		9000		10000	
	Correct rate	Run time										
DBSCAN	97.8%	36.7	97.2%	52.8	97.4%	72.6	97.9%	93.7	97.5%	121.5	97.8%	154.6
NSCABDT	99.7%	48.2	99.8%	71.4	99.8%	93.8	99.7%	117.9	99.7%	151.3	99.7%	189.4

Abbildung 4.11: NSCABDT schlägt DBSCAN immer, braucht dafür aber länger

#### 4.3.6 Graphbasiertes Clustering II (NSFCDT)

Eine weitere Optimierung, die zwei Jahre später vorgestellt wurde, basiert ebenfalls auf der Delaunay-Triangulation mit anschließendem Fuzzy Refinement. Ein Hauptvorteil gegenüber NSCABDT ist eine deutlich verbesserte Laufzeiteffizienz bei gleichwertigen Ergebnissen [RM12]. Die Methode wurde speziell für sehr große Datensätze entworfen, wie es bei Geodaten gewöhnlich ist.

Das Verfahren besteht aus zwei Schritten. Im ersten Schritt wird die Delaunay Triangulation berechnet und ein ähnlichkeitsbasiertes Clustering gebildet. Im zweiten Schritt wird eine Fuzzy Logik zur Verfeinerung der Ergebnisse angewandt.

Der folgende Algorithmus beschreibt den ersten Schritt des Verfahrens. Er einen Datensatz  $X$  mit  $n$  Punkten entgegen und clustert ihn auf Basis der Delaunay-Triangulation.

In der Originalpublikation wird für die Erzeugung der  $DT$  der Bowyer-Watson Algorithmus empfohlen, der nur  $\mathcal{O}(n \log n)$  Operationen benötigt, um  $n$  Punkte zu triangulieren und somit besonders effizient ist. Anschließend startet in Zeile 5 die harte Partitionierung. Dabei wird für ein Initialdreieck aus  $DT$  geprüft, ob es größer ist, als ein bestimmter Schwellwert und ob es noch nicht besucht wurde. Wurde ein Initialdreieck gefunden, das klein genug ist, werden von ihm aus alle Nachbarn betrachtet. Für jedes Nachbardreieck wird geprüft, ob es klein genug ist. Ist dies der Fall, wird es auf einem Stack gespeichert. Alle zu großen Nachbardreiecke werden hingegen ignoriert. Der Algorithmus wird solange fortgesetzt, bis alle Elemente auf dem Stack abgearbeitet wurden und kein gültiger Nachbar eines Stackelements mehr gefunden werden konnte. Die Menge der gültigen Nachbardreiecke inklusive Initialdreieck bilden dann einen Cluster. Der Algorithmus wird danach für alle unbesuchten Initialdreiecke aus  $DT$  fortgesetzt. Zum Schluss enthält die Menge  $ClusterSet$  Mengen von Dreiecken, die die Cluster des Datensatzes bilden.

**Algorithm 3** Schritt 1: Delaunay Klassifikation

---

```

Input:  $X = \{x_1, \dots, x_n\}$ 
1:  $DT \leftarrow \text{DELAUNAYTRIANGLES}(X)$ 
2:  $ClusterSet \leftarrow \emptyset$ 
3:  $NewSet \leftarrow \emptyset$ 
4:  $TmpSet \leftarrow \emptyset$ 
5: for all  $dt \in DT$  do
6:   if  $\text{VISITED}(dt) = False$  then
7:      $\text{INITIALIZE}(NewSet)$ 
8:      $\text{PUSH}(dt, Stack)$ 
9:     while  $\text{EMPTY}(Stack) = False$  do
10:       $Triangle T \leftarrow \text{POP}(Stack)$ 
11:      if  $\text{SATISFIABLE}(T) \& \text{NOTVISITED}(T)$  then
12:         $T.Visited \leftarrow True$ 
13:         $NewSet \leftarrow NewSet \cup \{T\}$ 
14:         $TmpSet \leftarrow \text{NEIGHBOURS}(T)$ 
15:        for all  $Element \in TmpSet$  do
16:          if  $\text{NOTVISITED}(Element)$  then
17:             $\text{PUSH}(Element, Stack)$ 
18:          end if
19:        end for
20:      end if
21:    end while
22:     $ClusterSet \leftarrow ClusterSet \cup newSet$ 
23:  end if
24: end for

```

---

Im zweiten Schritt wird eine Fuzzy-Funktion angewandt, um das Clustering zu verfeinern. Dies ist notwendig, um Dichteabfälle an den Randbereichen großer Cluster zu erkennen. Beispielsweise hat ein Wald keine harte Grenze, an der das Baumwachstum plötzlich aufhört. Auch die Bäume in den weniger dichten äußeren Regionen sollten dem Waldcluster zugeordnet werden. Gleichzeitig soll die Größe der Randregionen bei zunehmender Größe des Kernwaldes ebenfalls zunehmen. Die hierzu erforderliche Fuzzy-Zugehörigkeitsfunktion lautet:

$$\mu(x : \alpha, \beta) = \begin{cases} 0 & x < \alpha \\ 2 \cdot \left(\frac{x-\alpha}{\beta-\alpha}\right)^2 & \alpha \leq x < \frac{\alpha+\beta}{2} \\ 1 - 2 \cdot \left(\frac{x-\beta}{\beta-\alpha}\right)^2 & \frac{\alpha+\beta}{2} \leq x < \beta \\ 1 & x \geq \beta \end{cases}$$

Es handelt sich hierbei um eine Sigmoidfunktion. Der Parameter  $\alpha$  gibt den Wendepunkt der Kurve an, der Parameter  $\beta$  bestimmt die Neigung der Kurve, welche mit zunehmendem  $\beta$  immer flacher wird. Je nach Klasseneigenschaft können die Fuzzy Parameter variiert werden. Beispielsweise könnte die Baumart Buche, die einem Nadelwald steht, eine steilere Zugehörigkeitskurve besitzen, als die Baumart Tanne. Das bedeutet, dass eine Buche gegenüber einer Tanne viel näher an einem Nadelkernwald stehen muss, um mit einer hohen Zugehörigkeit klassifiziert zu werden.

Der Funktionswert  $\mu$  wird als Stärke der Inklusion bezeichnet. Je größer dieser Wert wird, desto wird der Randbereich, in dem weitere Punkte berücksichtigt werden.

Der Gesamtalgorithmus inklusive Triangulation und Fuzzy Refinement sieht dann folgendermaßen aus:

---

**Algorithm 4** Fuzzy Basiertes NSFCDT

---

```

Input:  $X = \{x_1, \dots, x_n\}$ 
1: BUILD(ClusterSet)
2: for all  $C_i \in ClusterSet$  do
3:    $Area \leftarrow GETAREA(C_i)$ 
4:    $Degree \leftarrow GETFUZZYDEGREE(Area)$ 
5:   SETSATISFIABILITYCONDITION( $Degree$ )
6:   for all  $Triangle \in Cluster$  do
7:      $N \leftarrow Neighbour(Triangle)$ 
8:     for all  $E \in N$  do
9:       if FUZZYSATISFIABLE( $E$ ) then
10:        if  $E \in C_j, \forall j = 1 \dots m \& j \neq i$  then
11:           $C_i \leftarrow C_i \cup C_j$ 
12:        else
13:           $C_i \leftarrow C_i \cup \{E\}$ 
14:        end if
15:      end if
16:    end for
17:  end for
18: end for

```

---

### Ergebnisse

Die Untersuchung hat gezeigt, dass NSFCDT gegenüber NSCABDT bei größeren Datensätzen eine bessere Laufzeit besitzt, vor allem, wenn der Refinement-Schritt weggelassen wird. Bei kleineren Datensätzen ist das Fuzzy-NSFCDT etwas langsamer, erzielt jedoch bessere Ergebnisse, da manche Cluster sinnvollerweise verschmolzen werden können.

No of Points	Run Time in Seconds		
	NSCABDT	NSFCDT	
		Crisp Clustering	Fuzzy Clustering
	Run Time	Run Time	Run Time
5000	48.2	40.04	55.18
6000	71.4	54.72	81.99
7000	93.8	63.19	87.45
8000	117.9	97.37	135.82
9000	151.3	129.48	183.88
10000	189.4	167.73	228.31

Abbildung 4.12: Laufzeiten von NSFCDT und NSCABDT bei verschiedenen großen Datensätzen

So wurde zum Beispiel ein Graustufenbild einer indischen Stadt mit NSFCDT sowohl mit als auch ohne Refinement geclustert. Die Fuzzy-Methode fand dabei vier Gebiete mit Baumbestand. Die harte Methode jedoch unterteilte zwei der Gebiete in zwei weitere Cluster, obwohl dies zu sinnlosen Ergebnissen führte. Die Untersuchung zeigt, welch hohes Potential das Fuzzy basierte NSFCDT beim Clustern von großen Datensätzen mit vielseitigen Eigenschaften besitzt. So ist es von zentraler Bedeutung, dass je nach Geländeart eine individuelle Fuzzy-Kurve gefunden wird. So führt die Unterscheidung zwischen Gebäuden, Flüssen und Wäldern bei der Berechnung des Inklusionsfaktors dazu, dass für jede Klasse sehr präzise Clusteringergebnisse gefunden werden kann, welche die Präzision des NSCABDT-Algorithmus deutlich übertrifft.

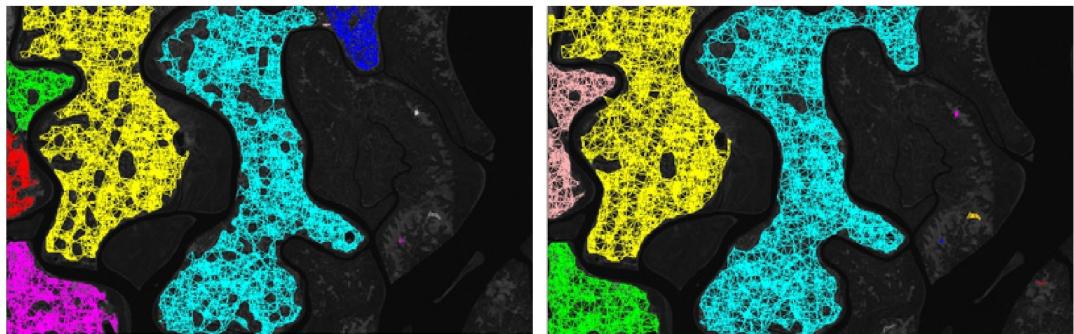


Abbildung 4.13: Clusteringergebnis nach hartem NSFCDT (links) und nach dem Fuzzy Refinement (rechts)

Als Vorteil sehen die Autoren, dass die Gesamlaufzeit des Algorithmus mit  $\mathcal{O}n\sqrt{n}$  sehr klein ist. Trotzdem fügen Sie hinzu, dass es noch viel Spielraum für Verbesserungen gibt. So können zwei benachbarte Dreiecke nicht berücksichtigt werden, wenn sie anstelle einer gemeinsamen Kante einen gemeinsamen Knoten haben. Dieses Problem kann nur durch einen weiteren Teilalgorithmus gelöst werden. Außerdem könnten noch effizientere Datenstrukturen oder statistische Modelle zu einer weiteren Verbesserung der Laufzeit führen.

**4.3.7 Kontextbasiertes Clustering****4.4 Finden der Clusteranzahl****4.4.1 NPC****4.4.2 FHV****4.4.3 Otsu-Binarisierung****4.4.4 VAT-Algorithmus****4.5 Clustering auf unvollständigen Daten****4.6 Assoziationsregeln**

In Kapitel 10 Geoinf. kommt der naive Bayes-Klassifikator vor! Paper:

[https://www.mii.lt/na/issues/NA\\_0602/NA06203.pdf](https://www.mii.lt/na/issues/NA_0602/NA06203.pdf) Entscheidungsbäume,  
Splitstrategien, Anwendung



## 5. Implementierungen





## 6. Fazit und Ausblick

### 6.1 Ausblick - Mein Thema für die Masterarbeit





## Literaturverzeichnis

- [Ank+99] Mihael Ankerst u. a. *OPTICS: Ordering Points To Identify the Clustering Structure*. Institute for Computer Science, University of Munich, 1999 (siehe Seite 42).
- [AV06] David Arthur und Sergei Vassilvitskii. *The Advantages of Careful Seeding*. Indiana University Bloomington, 2006 (siehe Seite 38).
- [Ber] R. Berwick. *An Idiot's guide to Support vector machines (SVMs)*. URL: <http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf> (besucht am 19.09.2018) (siehe Seite 49).
- [Bre+00] Markus M. Breunig u. a. *LOF: Identifying Density-Based Local Outliers*. Institute for Computer Science, University of Munich und Department of Computer Science, University of British Columbia, 2000 (siehe Seite 48).
- [DLR77] A. P. Dempster, N. M. Laird und D. B. Rubin. “Maximum Likelihood from Incomplete Data via the EM Algorithm”. In: *Journal of the Royal Statistical Society* 39 (1977), Seiten 1–38 (siehe Seite 39).
- [ES00] Martin Ester und Jörg Sander. *Knowledge Discovery in Databases: Techniken und Anwendungen*. Springer, 2000, Seite 66 (siehe Seite 40).
- [Est+96] Martin Ester u. a. *A Density-Based Algorithm for Discovering Clusters*. Institute for Computer Science, University of Munich, 1996 (siehe Seite 41).
- [FPS96] Usama Fayyad, Gregory Piatetsky-Shapiro und Padhraic Smyth. *Knowledge Discovery and Data Mining: Towards a Unifying Framework*. 1996. URL: <https://www.aaai.org/Papers/KDD/1996/KDD96-014.pdf> (siehe Seite 8).
- [For65] E.W. Forgy. “Cluster analysis of multivariate data: efficiency versus interpretability of classifications”. In: *Biometrics* 21 (1965), Seiten 768–769 (siehe Seite 38).

- [Goo92] Michael F. Goodchild. *Geographical Information Systems*. University of California, 1992, Seite 33 (siehe Seite 58).
- [JMF99] A.K. JAIN, M.N. MURTY und P.J. FLYNN. *Data Clustering: A Review*. Michigan State University, Indian Institute of Science und The Ohio State University, 1999 (siehe Seite 37).
- [Kar+17] Anuj Karpatne u. a. *Machine Learning for the Geosciences: Challenges and Opportunities*. Version 1. 13. Nov. 2017. arXiv: 1711.04708 (siehe Seite 29).
- [Lar+15] David J. Lary u. a. “Machine learning in geosciences and remote sensing”. In: *Geoscience Frontiers* (2015), Seiten 3–10 (siehe Seite 17).
- [LS] Alina Lazar und Bradley A. Shellito. *Classification in GIS Using Support Vector Machines*. URL: <http://www.irma-international.org/viewtitle/20393/> (besucht am 19.09.2018) (siehe Seite 48).
- [Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Education, 1997 (siehe Seite 17).
- [Pal+05] N.R. Pal u. a. “A possibilistic fuzzy c-means clustering algorithm”. In: *IEEE Transactions on Fuzzy Systems* 13 (2005), Seiten 517–530 (siehe Seite 56).
- [Roy13] Parthajit Roy. *Clustering and its application to GIS*. The University of Burdwan, 2013 (siehe Seite 52).
- [RM12] Parthajit Roy und J.K. Mandal. “A Novel Spatial Fuzzy Clustering using Delaunay Triangulation for Large Scale GIS Data (NSFCDT)”. In: *Procedia Technology* 6 (2012), Seiten 452–459 (siehe Seite 62).
- [Rud] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. URL: <http://ruder.io/optimizing-gradient-descent/> (besucht am 21.09.2018) (siehe Seite 24).
- [Tim+04] H. Timm u. a. “An extension to possibilistic fuzzy cluster analysis”. In: *Fuzzy Sets and Systems* 147 (2004), Seiten 3–16 (siehe Seite 55).
- [YC10] Xiankun Yang1 und Weihong Cui. *A Novel Spatial Clustering Algorithm Based on Delaunay Triangulation*. Institute of Remote Sensing Application, Chinese Academy of Sciences, 2010 (siehe Seite 58).