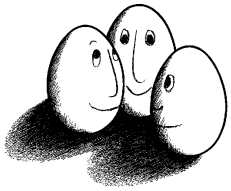
Diplomarbeit

Visualisieren und Verstehen hochdimensionaler Entscheidungsmodelle mit Self-Organizing Maps

Sebastian Land



Diplomarbeit
an der Fakultät für Informatik
der TU Dortmund

Dortmund, 23. April 2009

Betreuer:

Prof. Dr. Katharina Morik
Dipl.-Inf. Christian Bockermann

für Michael

Danksagung

Ich danke meinen Betreuern Prof. Dr. Katharina Morik und Dipl.-Inf. Christian Bockermann dafür, dieses Machwerk gelesen und mir mit ihrer konstruktiven Kritik dabei geholfen zu haben, den Wald trotz der Bäume zu sehen. Weiterhin danke ich allen Mitarbeitern und Ehemaligen des Lehrstuhl 8, die über die Jahre immer ein offenes Ohr für meine zahlreichen Fragen hatten und von denen ich während meines Studiums viel lernen durfte. Mein ganz besonderer Dank aber gilt Katharina Borg für ihren unermüdlichen Einsatz als Lektorin und Reviewerin, ohne deren Tipps und Aufmunterungen die Arbeit wohl nicht in dieser Form vorliegen würde.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Verfahren des maschinellen Lernens finden zur Zeit immer breitere Anwendung. Steigende Rechenleistung und stark fallende Speicherpreise ermöglichen die Anwendung in zahlreichen neuen Bereichen. Aus der Menge dieser Verfahren tun sich vor allem Entscheidungsmodelle hervor, da sie in einem weiten Bereich anwendbar sind. Sie lassen sich auf einfache Weise in alle Prozesse integrieren, in denen Entscheidungen bisher von Menschen getroffen wurden.

Mit den sich verbreiternden Anwendungsmöglichkeiten geht eine Zunahme des praktischen Einsatz in Industrie, Wirtschaft und Medizin einher. Die Anforderungen bei einem praktischen Einsatz unterscheiden sich jedoch grundlegend von denen im wissenschaftlichen Betrieb:

Zum einen werden die Verfahren in der Praxis häufig von Fachkräften aus der Anwendungsdomäne benutzt, die zwar umfangreiches Fachwissen aus dieser Domäne mitbringen, jedoch mit den komplexen Algorithmen nicht oder nur oberflächlich vertraut sind. Zum anderen erfordert der praxisnahe Einsatz die Möglichkeit, Ergebnisse verständlich zu präsentieren, da die Entscheidungsträger nicht mit den Verfahren vertraut sind und gleichzeitig die Anwendung der Ergebnisse häufig mit einem Riskio verbunden ist. Je nach Anwendungsbereich könnten fehlerhafte Güter produziert, Geld verloren oder in der Medizin dem Patienten durch eine falsche Diagnose geschadet werden.

Ein mögliches Szenario wäre eine Produktionslinie, die mit zahlreichen Sensoren die Fertigung protokolliert. Hier könnte der menschliche Operator, der den Ablauf überwacht, durch ein gelerntes Modell unterstützt oder sogar ganz ersetzt werden. Wird bei der Überwachung etwas übersehen oder eine falsche Entscheidung getroffen, kann eine ganze Charge unbrauchbar oder schlimmstenfalls die Produktionslinie zerstört werden, was enorme Kosten verursachen würde.

Obwohl natürlich auch Menschen Fehler machen können, muss eine neue Technik nicht nur ihre Leistungsfähigkeit beweisen, sondern auch immer erst Berührungsängste überwinden. Um diese ab- und stattdessen Vertrauen aufzubauen, ist aber Verständnis für die Funktionsweise unumgänglich. Dies gilt für die Verfahren des maschinellen Lernens heute genau so, wie es für die Eisenbahn am Anfang des 19. Jahrhunderts galt. Um dieses Verständnis zu vermitteln wird in der Praxis gerne auf statistische Modelle zurückgegriffen, die von sich aus eine gewisse Verständ-

lichkeit mitbringen und deren Entscheidungen daher einfach nachvollziehbar sind. Beispiele für Verfahren die solche Modelle erzeugen sind:

- Thresholdlerner
- Lineare Regression
- Naive Bayes
- Baumlerner
- Regellerner

Allerdings gehen alle diese Verfahren Einschränkungen ein, um die Verständlichkeit zu erreichen. Beispielsweise verwenden Entscheidungsbäume und Regellerner nur Entscheidungsgrenzen, die orthogonal zu einer Achse verlaufen und Naive-Bayes nimmt für alle gemessenen Werte eine Normalverteilung unter vollständiger Unabhängigkeit an. Zwar beseitigen diese Verfahren damit häufig die Berührungsängste, jedoch kommen sie dann aufgrund ihrer eingeschränkten Anwendbarkeit nur bei wenigen Problemen zum Einsatz.

Gerade moderne und leistungsfähige Lernverfahren wie SVMs oder neuronale Netze entziehen sich durch ein komplexes Wirkungsgeflecht zwischen den einzelnen Werten und Messungen dem menschlichen Verständnis. Dabei bekommen die Anwender die Probleme zu spüren, die bei zahlreichen, gleichzeitig gemessenen Werten immer auftreten: Das menschliche Sinnessystem entdeckt zwar sehr schnell Regelmäßigkeiten in drei oder vier Dimensionen, scheitert aber sobald die Anzahl der Dimensionen weiter zunimmt. Genau hier liegt aber die Stärke der Lernverfahren, so dass sich ein grundsätzliches Vermittlungsproblem ergibt.

Bisher konnte für dieses Vermittlungsproblem noch keine Lösung gefunden werden, die mit allgemeinen Entscheidungsmodellen arbeitet. Existierende Ansätze beschränkten sich immer auf die Darstellung einer beschränkten Klasse von Modellen, so dass Vergleiche zwischen Modellen verschiedener Verfahren unmöglich waren.

Daher soll in dieser Arbeit eine Technik entwickelt werden, die die Komplexität der Daten und Modelle einerseits für den Benutzer herunterbricht und verständlich wiedergibt, andererseits jedoch ermöglicht, weiterhin beliebige Lernverfahren auf den hochdimensionalen Daten einzusetzen. Gleichzeitig muss dem Benutzer die Möglichkeit gegeben werden die Zusammenhänge in den Daten zu erkennen und die Interaktion zwischen Daten und Entscheidungsmodellen zu beobachten. Dabei sollten seine Fähigkeit, Strukturen in niedrigen Dimensionen zu schnell zu entdecken, sinnvoll ausgenutzt werden.

Die hier vorgestellte Technik basiert auf der Kombination von geeigneten Dimensionsreduktions- und Interpolationsverfahren. Bei den Verfahren zur Dimensionsreduktion wird besonderes Augenmerk auf Self-Organizing Maps gelegt, deren

ursprüngliche Version sich stark an neurobiologische Vorgänge bei der Sinnesverarbeitung anlehnte. Im Rahmen dieser Arbeit werden verschiedene Weiterentwicklungen vorgestellt und darauf aufbauend ein eigenes Verfahren entwickelt. Alle Varianten werden später mit ausgewählten anderen Verfahren verglichen um ihren Einsatz bei der Modellvisualisierung zu motivieren.

Die Arbeit beginnt in Kapitel 2 mit einer kurzen Einführung in die Grundbegriffe des maschinellen Lernens und stellt beispielhaft drei verschiedene Lernverfahren vor. Anhand dieser Lernverfahren wird in Kapitel 3 die Problematik der Modellvisualisierung erläutert und motiviert. Anschließend wird in diesem Kapitel skizziert, wie sich das Problem mit einem zweischrittigen Verfahren über Dimensionsreduktion und Interpolation lösen lässt. Danach werden die verwendeten Verfahren für die beiden Schritte kurz vorgestellt und ihre Auswahl über eine Klassifizierung motiviert. Die folgenden drei Kapitel behandeln die ausgewählten Verfahren dann in der Tiefe: In Kapitel 4 und 5 werden die Varianten der Self-Organizing Map beziehungsweise ihre Alternativen detailiert beschrieben, während Kapitel 6 die Interpolationsverfahren vorstellt. Nachdem mit einer Kombination der Verfahren aus den letzten drei Kapiteln eine Abbildung berechnet wurde, muss diese für den Benutzer geeignet graphisch aufbereitet werden. Daher erläutert Kapitel 7 die visuelle Umsetzung der von Dimensionsreduktion und Interpolation gelieferten Ergebnisse. Im letzten Kapitel 8 wird abschließend die Leistungsfähigkeit der vorgestellten Verfahren überprüft und ein kurzer Ausblick gegeben.

2. Grundlagen

Wie bereits in der Einleitung angedeutet, beschäftigt sich diese Arbeit allgemein mit der Visualisierung von statistischen Entscheidungsmodellen und speziell mit den Problemen hoher Dimensionen. Dazu ist zunächst ein grundlegendes Verständnis über Entscheidungsmodelle und deren konkrete Eigenschaften notwendig, das dieses Kapitel vermitteln soll. Den Prozess der Modellbildung, also die als Lernverfahren bezeichneten Algorithmen zum Bestimmen der Modellparameter, detailliert zu beschreiben, würde den Rahmen der Arbeit sprengen und wird daher im Folgenden ausgeklammert. Auch die Verwendung einer Vielzahl verschiedener Modelle würde den Leser eher verwirren als ihm nutzen, so dass sich die Arbeit auf die Vorstellung jeweils eines Modells aus drei verschiedenen Klassen von Modellen beschränkt. Dabei soll die Einteilung der Klassen hier nicht auf den statistischen Eigenschaften beruhen, sondern auf der Verständlichkeit für den Anwender. Ausgewählt wurden Entscheidungsbäume als Vertreter der Klasse der anschaulichen Modelle und die Modelle der Linearen Diskriminanzanalyse und der Support Vector Machine als Vertreter der Klasse der interpretierbaren respektive der komplexen Modelle, die im Folgenden zunächst kurz beleuchtet werden sollen. Eine Übersicht über weitere Lernverfahren und ihre Modelle findet sich in (Hastie et al. 2001) und (Mitchell 1997).

Entscheidungsbäume

Die Art und Weise in der eine Entscheidung getroffen wird, lässt sich an einem Entscheidungsbaum direkt ablesen, ähnlich den weit verbreiteten Entscheidungsdiagrammen. So sind diese auf (Breiman et al. 1984) und (Quinlan 1993) zurückgehenden Modelle auch für Fachfremde sofort verständlich. Sie gehören entsprechend in die Klasse der verständlichen Modelle.

Lineare Diskriminanzanalyse

Obwohl die Lineare Diskriminanzanalyse nach dem einfachen Prinzip einer Grenze, die das eine vom anderen trennt, arbeitet, ist die Grenze durch die mathemati-

sche Repräsentation als lineare Hyperebene bereits schwieriger zu verstehen und in höheren Dimensionen nicht mehr vorstellbar. Durch die Beschränkung auf Linearität ist sie jedoch für Fachleute noch interpretierbar, da sie Rückschlüsse auf die Zusammenhänge in den Daten ziehen können. Die Modelle gehören also zur Klasse der interpretierbaren Modelle. Eine Beschreibung dieses weit verbreiteten Verfahrens findet sich im Abschnitt 2.3. Weitergehende Informationen finden sich zum Beispiel in (Hastie et al. 2001).

Support Vector Machines

Aufgrund ihrer mathematischen Funktionsweise können Support Vector Machines Modelle verschiedenster Komplexität liefern, obwohl sie wie die der linearen Diskriminanzanalyse immer auf dem Prinzip der Abgrenzung beruhen. Während die Modelle im einfachsten Fall als lineare Trenngrenze ähnlich der linearen Diskriminanzanalyse ausgedrückt werden können, verlieren sie sämtliche Interpretierbarkeit, sobald komplexere Funktionen für die Grenzziehung eingesetzt werden und die Anzahl der Dimensionen steigt. Sie gehören entsprechend in die Klasse der komplexen Modelle, aus denen keine Rückschlüsse über die Daten gezogen werden können. Eine kurze Einführung findet sich im Abschnitt 2.4. Ausführlicher wird zum Beispiel (Schölkopf und Smola 2001).

Zunächst wird nun eine Einführung in die Begriffe des Data Mining gegeben, bevor anschließend in den folgenden Abschnitten genauer auf die drei Modelle eingegangen wird.

2.1. Notation und Begriffe

Um das Lesen und Verstehen der in dieser Diplomarbeit vorgestellten Algorithmen und Formeln zu erleichtern, wird hier kurz die Systematik der Notation vorgestellt. Daten werden in einfache und komplexe Datentypen unterteilt, wobei einfache Datentypen durch die Verwendung von Kleinbuchstaben gekennzeichnet sind, komplexe Typen durch Großbuchstaben. Bei einfachen Datentypen wird zwischen Skalaren und Mengenelementen a und andererseits Vektoren $\mathbf{x}$ unterschieden. Komplexe Daten fallen in die drei Kategorien Mengen $\mathcal{N}$, Tupel N und Matrizen $\mathbf{V}$.
Funktionen werden gemäß ihres Rückgabetyps gekennzeichnet, eine vektorwertige Funktion also zum Beispiel mit $\mathbf{f}$(x), eine skalare mit f(x).
Eine Übersicht gibt Tabelle 2.1.

Einfache Datentypen		**Komplexe Datentypen**	
Skalare und Mengenelemente	a	Mengen	$\mathcal{N}$
Vektoren	$\mathbf{x}$	Tupel	N
		Matrix	$\mathbf{V}$

Tabelle 2.1.: verwendete Notation

Der folgende Abschnitt erklärt einige elementare Begriffe aus dem Bereich des Data Mining um ein Grundverständnis für die ablaufenden Prozesse zu vermitteln. Dieses Verständnis wird in der weiteren Arbeit vorausgesetzt, da die Ergebnisse dieser Prozesse weiter verarbeitet werden sollen.

Definition 2.1.1 (Entscheidungsmodell)
Ein Entscheidungsmodell ist eine Funktion f*, die jedem Punkt* $\mathbf{x}$ *eines durch eine Domäne* $\mathbb{D}$ *bestimmten Beispielraums* $\mathbb{D}^m$ *einer Klasse* c *aus einer Menge von möglichen Klassen* $\mathcal{C}$ *zuweist:*

$$\mathrm{f} : \mathbb{D}^m \rightarrow \mathcal{C} \tag{2.1}$$

Die konkrete Funktion und deren Parameterwerte werden dabei als Modell bezeichnet, da sich aus ihnen die Zuordnungsregeln ableiten.

Da ein Entscheidungsmodell offensichtlich über einem Raum arbeitet, der durch eine Domäne festgelegt wird, soll zunächst der Begriff der Domäne definiert werden:

Definition 2.1.2 (Domäne)
Die Domäne oder auch Problemdomäne $\mathbb{D}$ *bezeichnet das Umfeld, in dem ein Entscheidungsmodell arbeiten soll, ist also die Abstraktion der relevanten physikalischen Umwelt. Sie gibt nicht nur vor, welche Entscheidung getroffen werden muss und damit welche Klassen* c *in der Menge der möglichen Klassen* $\mathcal{C}$ *enthalten sind, sondern auch wie der Beispielraum beschaffen ist. Jede Domäne besitzt genau ihren Beispielraum* $\mathbb{D}^{\mathrm{m}}$ *mit bestimmter Dimensionalität* m *und bestimmten, aber meist unbekannten Eigenschaften der Wahrscheinlichkeitsverteilung* F*. Jede Dimension kann dabei von unterschiedlichem Datentyp sein, also zum Beispiel reele oder ganze Zahlen beinhalten, aber auch nominale Werte.*

Beispiel 2.1.1 (Domäne)
Als Problemdomäne betrachten wir hier beispielsweise einen Prozess zum Brennen von Dachziegeln. Jeder Dachziegel durchläuft dabei verschiedene Stationen wie Formen, Brennkammern, Lackierstraßen etc. Jede dieser Stationen besitzt Sensoren, die den aktuellen Zustand messen. Sind m *Sensoren vorhanden, besitzt die*

Domäne die Dimensionalität m *und jede Dimension repräsentiert den Wert des entsprechenden Sensors.*
Ein Dachziegel, der durch die verschiedenen Stationen läuft, wird nun durch die jeweils zu diesem Zeitpunkt aktuellen Werte beschrieben, so dass er am Ende durch einen Punkt im Beispielraum $\mathbb{D}^m$ beschrieben wird. Eine mögliche Klasseneinteilung kann sich durch verschiedene Qualitätsstufen 1A, 1B, etc. ergeben. Zu welcher Qualität ein Dachziegel gehören wird, entscheidet sich über die unbekannte, der Domäne inhärenten Wahrscheinlichkeitsverteilung F*. Diese bildet den Zusammenhang zwischen dem Zustand der Produktionsstufen und dem Endergebnis ab und bestimmt daher, mit welchen Wahrscheinlichkeiten die Dachziegel den Qualitätsstufen entsprechen werden, gegeben die Sensorwerte.*

Um nun ein Entscheidungsmodell für eine Domäne zu erzeugen, können Verfahren des maschinellen Lernens angewendet werden. Jedes Lernverfahren benötigt aber etwas, aus dem es lernen kann. Diese Arbeit beschränkt sich auf die drei oben genannten Verfahren, die alle aus der Klasse der überwachten Lernverfahren stammen. Verfahren dieser Klasse benötigen kein Hintergrundwissen über die Domäne, stattdessen verarbeiten sie eine Menge von einzelnen Beispielen, mit deren Hilfe die Parameter eines Entscheidungsmodells angepasst werden sollen.

Definition 2.1.3 (Beispiel)
Ein Beispiel ist eine Beschreibung eines einzelnen Objektes oder Ereignisses in der Problemdomäne. Diese Beschreibung erfolgt in der durch die Domäne festgelegten Weise durch einen m*-dimensionalen Vektor* $\mathbf{x}$ *aus dem Beispielraum* $\mathbb{D}^m$*. Soll ein Beispiel zur Modellbildung genutzt werden, muss es mit der gewünschten Klasse* $y \in \mathcal{C}$ *markiert werden, man sagt auch "es wird gelabelt".*

Beispiel 2.1.2 (Beispiel)
In der in Beispiel 2.1.1 beschriebenen Domäne einer Fabrik ist ein Dachziegel ein solches Objekt. Er wird durch die Sensorwerte zum Zeitpunkt des Passierens der entsprechenden Maschine beschrieben und je nach Qualität durch einen Kontrolleur gelabelt.

Man spricht also von überwachtem Lernen, weil vor dem Lernen bereits für eine gewisse Menge von Beispielen von einem meist menschlichen Überwacher entschieden wurde, zu welcher Klasse sie gehören. Auf diesen Daten werden dann die Modelle gebildet.

Definition 2.1.4 (Training)
Bei überwachten Lernverfahren werden aus den gegebenen, gelabelten Beispielen, den Trainingsbeispielen, die Parameter der Entscheidungsmodelle berechnet um die

Modelle möglichst gut an die Strukturen in den Daten anzupassen. Diesen Prozess nennt man Training.

Jedes Lernverfahren trifft beim Training seine eigenen heuristischen Annahmen über die Strukturen der Wahrscheinlichkeitsverteilung F, über die es dann ein bestimmtes Kriterium optimiert. Dieses Kriterium soll bei zutreffenden Annahmen zu einem möglichst geringen Vorhersagefehler auf unbekannten Daten aus der Problemdomäne führen.

2.2. Entscheidungsbäume

Im Folgenden werden als erstes der hier betrachteten Modelle Entscheidungsbäume besprochen, wie sie in (Quinlan 1993) und auch in (Breiman et al. 1984) beschrieben werden.
Wie der Name schon andeutet, bestehen Entscheidungsbäume aus baumartig angeordneten Entscheidungen. Diese Entscheidungen setzen sich aus Bedingungen über den Beispielraum zusammen, die für ein Beispiel entweder zutreffen oder nicht. Wie jeder Baum beginnen auch Entscheidungsbäume mit einer Wurzel, man spricht hier von dem Wurzelknoten. Knoten in Entscheidungsbäumen entsprechen Verzweigungsstellen in echten Bäumen, von jedem der Knoten gehen also Äste ab, die zu anderen Knoten führen. Jedem Ast ist dabei eine Bedingung zugeordnet. Formal ergibt sich so ein Entscheidungsknoten zu:

Definition 2.2.1 (Entscheidungsknoten)
Ein Entscheidungsknoten $\mathcal{K}_\mathrm{i}$ *besteht aus einer Menge von Tupeln der Form* $(\mathcal{K}_\mathrm{j}, \mathrm{C}_\mathrm{j})$. *Dabei ist* $\mathcal{K}_\mathrm{j}$ *der Nachfolgeknoten, der über eine Bedingung* C_j *erreicht werden kann. Für jedes mögliche Beispiel* $\mathbf{x} \in \mathbb{D}^\mathrm{m}$ *muss genau eine Bedingung erfüllt sein, das heißt die Zuordnung zum Nachfolger ist in jedem Knoten eindeutig und definiert.*

Da Entscheidungsbäume nicht unendlich groß sein können, muss es Blattknoten geben, über die keine weiteren Nachfolgeknoten erreichbar sind, diese müssen das Beispiel dann einer Klasse der Problemdomäne zuweisen.

Definition 2.2.2 (Blattknoten)
Ein Blattknoten $\mathcal{K}_\mathrm{l}$ *ist ein Entscheidungsknoten, dessen Menge von Nachfolgetupeln leer ist und der stattdessen mit einer Klasse* c *aus der Menge der möglichen Klassen* $\mathcal{C}$ *markiert ist.*

Nun kann für jedes Beispiel an dem Wurzelknoten begonnen werden, einen Pfad

zu beschreiten, der durch die jeweils erfüllte Bedingung definiert ist. Dieser führt anschaulich gesprochen von der Wurzel den Baum hinauf, von Entscheidungsknoten zu Entscheidungsknoten, bis ein Blattknoten erreicht wird. Dem Beispiel wird dann diejenige Klasse zugeordnet, mit der dieser Knoten markiert ist. Siehe hierzu Abbildung 2.1, in der ein angenommenes Beispiel am Wurzelknoten die Bedingung c_2 erfüllt und über den Nachfolgeknoten und dessen Bedingung c_3 den Blattknoten erreicht, der mit der Klasse Blau markiert wurde. Anders als bei echten Bäumen wird die Wurzel bei Entscheidungsbäumen oben dargestellt, so dass der Entscheidungsfluss mit der gewohnten Leserichtung übereinstimmt.

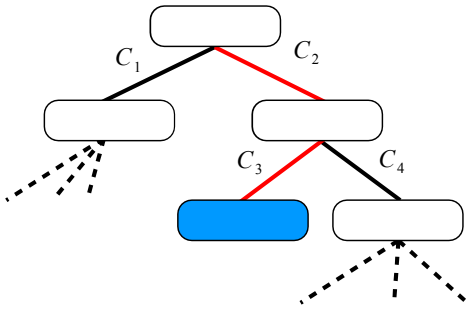


Abbildung 2.1.: Pfand in einem Entscheidungsbaum

Dabei bleibt die Menge aller möglichen Bedingungen $\{c_i | i \in \mathbb{N}\}$, die in Quinlans C4.5 Algorithmus und seinen Verwandten Anwendung findet, stark beschränkt. Sie beinhaltet nur einfache Bedingungen, von denen jede nur jeweils eine Dimension prüft. Sie teilen bei ordinalen Daten in größer und kleiner in Bezug zu einem Referenzwert und unterteilen bei nominalen Daten mit der Gleichheitsrelation. Interaktionen zwischen den Dimensionen, wie zum Beispiel $\mathbf{x}_1 + \mathbf{x}_4 < 5$, können so von den Bedingungen nicht erfasst werden.
Aus der Anforderung, dass immer genau eine Bedingung erfüllt sein muss, folgt für alle Bedingungen eines Knotens notwendigerweise, dass sie sich auf dieselbe Dimension beziehen. Andernfalls ist nicht sicherzustellen, dass unter jeder möglichen Wertekombination eine gültige Bedingung gefunden werden kann.
Entsprechend unterteilt das Passieren eines Entscheidungsknotens den Datenraum mit einem zur Koordinatenachse der geprüften Dimension orthogonalen Schnitt. Wie man sich dieses vorzustellen hat, zeigt Abbildung 2.2. Die ersten beiden Knoten unterteilen den Datenraum an den mit durchgezogenen Linien markierten Stellen. Der Blattknoten markiert den dadurch entstehenden, nach unten und rechts

offenen Bereich als zur Klasse "Blau" zugehörig. Ausgeblendete, im Baum nur durch die gestrichelten Linien angedeutete Knoten sorgen für die weitere Unterteilung in die durch gestrichelte Linien getrennten Bereiche. Blattknoten weisen ihnen dann die durch die Schraffur angedeutete Farbe zu, so dass jedes Element des gesamten Datenraums einer der Klassen zugewiesen wird.
In Abbildung 2.3 erkennt man, dass Interaktionen zwischen den Dimensionen sehr wohl vom Modell abgebildet werden, obwohl eine einzelne Bedingung sie nicht zu fassen vermag. Dabei bleibt diese Abbildung beschränkt auf eine orthogonale Annäherung an die tatsächliche, hier gestrichelt dargestellte optimale Bayes Klassifikationsgrenze. Eine kurze Erklärung zur Bayes Klassifikationsgrenze findet sich im nächsten Abschnitt.

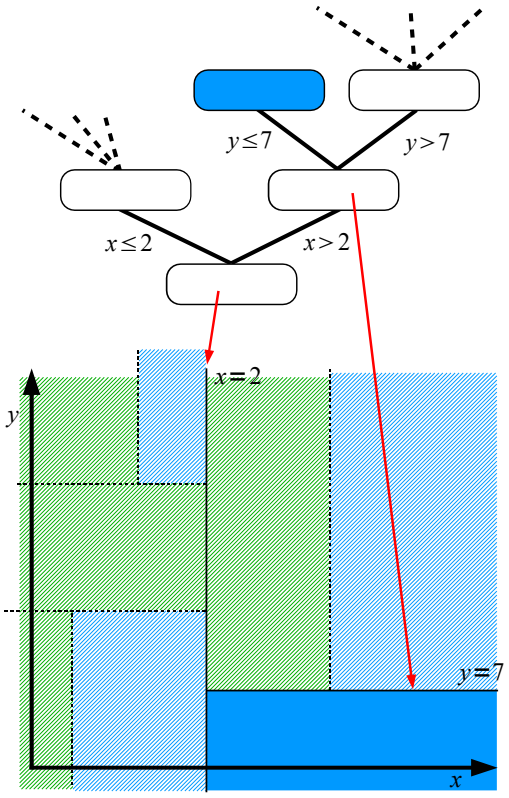


Abbildung 2.2.: Unterteilung des Datenraumes durch trennende Bedingungen

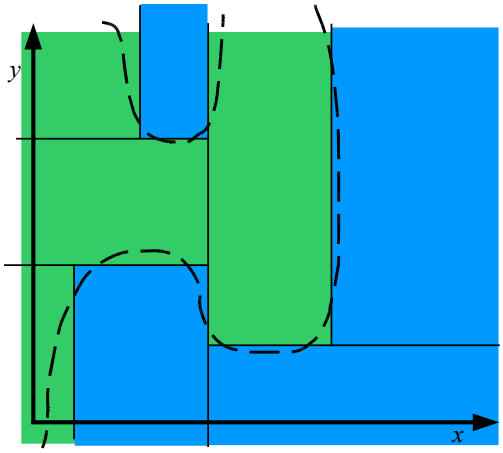


Abbildung 2.3.: Modellrestriktionen eines Entscheidungsbaums

2.3. Lineare Diskriminanzanalyse

Nach dem Entscheidungsbaumlerner wird nun die lineare Diskriminanzanalyse, kurz LDA, als Vertreter der interpretierbaren Verfahren vorgestellt. Sie wurde bereits vor über 70 Jahren in (Fisher 1936) vorgestellt. Inzwischen ist sie ein weit verbreitetes Verfahren zur Klassifikation von Daten und hat einen festen Platz im Werkzeugkasten vieler Dataminer. Eine detailierte Beschreibung mit Bezugnahme zur statistischen Entscheidungstheorie findet sich in (Hastie et al. 2001).

Jede Domäne besitzt eine Wahrscheinlichkeitsverteilung F, wie in ihrer Definition (siehe 2.1.2) beschrieben. Die Idee der linearen Diskriminanzanalyse ist, diese unbekannte Verteilung zu bestimmen um mit ihrer Hilfe trennende Grenzen zu berechnen. Diese Grenzen trennen die Bereiche des Beispielraums, in denen eine Klasse wahrscheinlicher ist als alle anderen. Dazu ist die Kenntnis einiger statistischer Begriffe nötig, die zunächst definiert werden.

Definition 2.3.1 (Wahrscheinlichkeitsverteilung)
Die Wahrscheinlichkeitsverteilung F *gibt an, wie sich die Wahrscheinlichkeiten auf mögliche Zufallsereignisse verteilen. Häufigkeitsverteilungen bilden das empirische Gegenstück zu einer theoretischen Wahrscheinlichkeitsverteilung.*

Speziell im Bereich des Datamining werden diese Zufallsereignisse auf die Rea-

lisation von Beispielen bezogen. Die Wahrscheinlichkeitsverteilung bestimmt also wo und mit welcher Klasse im Datenraum Beispiele mit welcher Wahrscheinlichkeit auftauchen.

Definition 2.3.2 (Wahrscheinlichkeitsdichte)
Die Wahrscheinlichkeitsdichte d*, oft auch nur Dichte, ist definiert durch eine Wahrscheinlichkeitsverteilung* F*. Ist diese Verteilung stetig differenzierbar, ergibt sich die Dichte zu:*

$$\mathrm{d(x)} = \frac{d\mathrm{F(x)}}{d\mathrm{x}} \tag{2.2}$$

Andererseits gilt auch:

$$\mathrm{F} = \int_{-\infty}^{\mathrm{x}} \mathrm{d(t)}dt \tag{2.3}$$

Die Dichte stellt damit anschaulich ein Hilfsmittel dar um die Wahrscheinlichkeit zu berechnen, mit der x in ein bestimmtes Intervall fällt.
Bei mehrdimensionalen Wahrscheinlichkeitsverteilungen bestimmt sich die Dichte durch mehrfaches partielles Differenzieren.

Definition 2.3.3 (Bedingte Wahrscheinlichkeit)
Seien $\mathcal{A}$ und $\mathcal{B}$ Ereignisse. $\mathrm{p}(\mathcal{B}|\mathcal{A})$ *heißt bedingte Wahrscheinlichkeit und gibt die Wahrscheinlichkeit an, mit der das Ereignis $\mathcal{B}$ eintritt, wenn $\mathcal{A}$ schon eingetreten ist.*

Definition 2.3.4 (Apriori Klassenwahrscheinlichkeit)
Die apriori Klassenwahrscheinlichkeit gibt an, mit welcher Wahrscheinlichkeit ein neues, noch ungesehenes Beispiel einer Klasse angehört. Die noch unbekannten Werte des Beispiels spielen dabei keine Rolle.

Die apriori Klassenwahrscheinlichkeit ist also das theoretische Gegenstück zur empirischen, relativen Klassenhäufigkeit einer Beispielmenge.

Für eine weiterführende Einführung in die statistische Entscheidungstheorie siehe (Hastie et al. 2001).

Wurde die Verteilung erst einmal bestimmt, lässt sich aus ihr die unbedingte Dichte $\mathrm{d}(\mathbf{x})$ einfach ableiten. Allerdings sind weitere Informationen wie die klassenbedingten Dichten $\mathrm{d}(\mathbf{x}|\mathrm{c})$ und die apriori Wahrscheinlichkeiten der Klassen notwendig um die Grenzen zu bestimmen. Denn mit diesen Informationen lässt sich dann der Satz von Bayes nutzen:

Satz 2.3.1 (Satz von Bayes)
Seien $\mathcal{A}$ und $\mathcal{B}$ Ereignisse. Dann sind $\mathrm{p}(\mathcal{A})$ *und* $\mathrm{p}(\mathcal{B})$ *jeweils die apriori Wahr-*

scheinlichkeiten der Ereignisse $\mathcal{A}$ *und* $\mathcal{B}$ *und* $\mathrm{p}(\mathcal{B}|\mathcal{A})$ *die bedingte Wahrscheinlichkeit.*
Der Satz von Bayes erlaubt uns nun $\mathrm{p}(\mathcal{A}|\mathcal{B})$ *zu berechnen:*

$$\mathrm{p}(\mathcal{A}|\mathcal{B}) = \frac{\mathrm{p}(\mathcal{B}|\mathcal{A}) \cdot \mathrm{p}(\mathcal{A})}{\mathrm{p}(\mathcal{B})} \tag{2.4}$$

Mit Hilfe des Satzes von Bayes lässt sich also die Wahrscheinlichkeit einer Klasse gegeben ein Beispiel $\mathbf{x}$ berechnen:

$$\mathrm{p}(\mathrm{c}|\mathbf{x}) = \frac{\mathrm{d}(\mathbf{x}|\mathrm{c}) \cdot \mathrm{p}(\mathrm{c})}{\mathrm{d}(\mathbf{x})} \tag{2.5}$$

Sind nur zwei Klassen gegeben, dann ergibt sich die Grenze zwischen den Klassen durch die folgende Gleichung:

$$\mathrm{p}(\mathrm{c}_1|\mathbf{x}) = \mathrm{p}(\mathrm{c}_2|\mathbf{x}) \tag{2.6}$$

Man klassifiziert also Beispiele in diejenige Klasse, die wahrscheinlicher ist und ist genau auf der trennenden Grenze indifferent, da dort beide Klassen gleich wahrscheinlich sind.
Ein Klassifizierer, der nach diesem Prinzip arbeitet, macht die geringste theoretisch mögliche Fehlerrate, wenn er die Wahrscheinlichkeitsverteilung kennt. Diesen Klassifizierer nennt man Bayes-Klassifizierer und seine Fehlerrate entsprechend Bayes-Fehlerrate. In der Praxis kennt man jedoch die Verteilung nicht, und so liegt die Herausforderung in einer möglichst guten Approximation der wahren Verteilung.

Für die lineare Diskriminanzanalyse trifft man dazu die Annahme, dass die wahren, klassenbedingten Verteilungen multivariate Normalverteilungen mit identischen Kovarianzmatrizen sind. Die klassenbedingten Dichten ergeben sich damit zu:

$$\mathrm{d}(\mathbf{x}|\mathrm{c}) = \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_\mathrm{c})^T \Sigma^{-1}(\mathbf{x}-\mu_\mathrm{c})} \tag{2.7}$$

Es müssen also nur die Klassenmittelwertvektoren μ_c, die Kovarianzmatrix Σ und die apriori Klassenwahrscheinlichkeiten $\mathrm{p}(\mathrm{c})$ aus den Beispieldaten bestimmt werden.
Die resultierende Grenze kann mit Hilfe der Gleichung 2.6 explizit bestimmt und in eine einfache Hyperebenenform gebracht werden, so dass sie verständlicher wird:

Definition 2.3.5 (Hyperebene)
Jeder Untervektorraum ist dann eine Hyperebene, wenn er genau eine Dimension weniger hat als der Vektorraum.
Jede Hyperebene lässt sich darstellen durch eine Gleichung der Form

$$x_1\beta_1 + x_2\beta_2 + \ldots + x_n\beta_n + \beta_0 = 0 \tag{2.8}$$

Oder kürzer in Vektorschreibweise mit je n-dimensionalem Vektor $\mathbf{x}$ *und Parametervektor* β*:*

$$\mathbf{x}^T\beta = \beta_0 \tag{2.9}$$

Damit sind Hyperebenen die Verallgemeinerung einer normalen Ebene in der dritten Dimension auf beliebige endliche Vektorräume.
Durch eine Analyse der Elemente des Parametervektors kann man herausfinden, welche Dimensionen des Beispielraums wie auf die Zuordnung zu einem Label Einfluss nehmen. Werte nahe 0 sprechen für einen verschwindend geringen Einfluss, große positive oder negative Werte haben dagegen größeren Einfluss. Da hierbei immer die Skalierung der einzelnen Dimensionen des Beispielraums eine Rolle spielt, benötigt man Kenntnisse über die Domäne um eine zuverlässige Interpretation liefern zu können.
Zunächst muss dazu allerdings die Gleichung 2.6 in die eben beschriebene Form 2.9 gebracht werden:

$$\begin{aligned}
(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1) &= (\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2) && (2.10)\\
\mathbf{x}^T\Sigma^{-1}\mathbf{x} - \mathbf{x}^T\Sigma^{-1}\mu_1 - \mu_1^T\Sigma^{-1}\mathbf{x} + \mu_1^T\Sigma^{-1}\mu_1 &= \mathbf{x}^T\Sigma^{-1}\mathbf{x} - \mathbf{x}^T\Sigma^{-1}\mu_2 && (2.11)\\
&\quad -\mu_2^T\Sigma^{-1}\mathbf{x} + \mu_2^T\Sigma^{-1}\mu_2 &&\\
\mathbf{x}^T\Sigma^{-1}(\mu_2-\mu_1) + (\mu_2-\mu_1)^T\Sigma^{-1}\mathbf{x} &= \mu_2^T\Sigma^{-1}\mu_2 - \mu_1^T\Sigma^{-1}\mu_1 && (2.12)\\
\mathbf{x}^T\Sigma^{-1}(\mu_2-\mu_1) + (\Sigma^{-1}\mathbf{x})^T(\mu_2-\mu_1) &= \mu_2^T\Sigma^{-1}\mu_2 - \mu_1^T\Sigma^{-1}\mu_1 && (2.13)\\
\mathbf{x}^T\Sigma^{-1}(\mu_2-\mu_1) + \mathbf{x}^T(\Sigma^{-1})^T(\mu_2-\mu_1) &= \mu_2^T\Sigma^{-1}\mu_2 - \mu_1^T\Sigma^{-1}\mu_1 && (2.14)\\
\mathbf{x}^T\Sigma^{-1}(\mu_2-\mu_1) + \mathbf{x}^T(\Sigma^{-1})^T(\mu_2-\mu_1) &= \mu_2^T\Sigma^{-1}\mu_2 - \mu_1^T\Sigma^{-1}\mu_1 && (2.15)\\
\mathbf{x}^T\cdot\underbrace{(\Sigma^{-1}+(\Sigma^{-1})^T)(\mu_2-\mu_1)}_{\beta} &= \underbrace{\mu_2^T\Sigma^{-1}\mu_2 - \mu_1^T\Sigma^{-1}\mu_1}_{\beta_0} && (2.16)
\end{aligned}$$

Dazu wird wiederholt die Regel $\mathbf{xM} = (\mathbf{M}^T\mathbf{x}^T)^T$ angewendet. In Schritt 2.13 kann die äußere Transponierung entfallen, da es sich bei dem gesamten Summanden $(\Sigma^{-1}\mathbf{x})^T(\mu_2-\mu_1)$ um ein Skalar handelt.
Mit Hilfe des Resultates 2.16 lassen sich die Parameter β und β_0 leicht bestimmen.

Wie ein mögliches Resultat einer LDA aussehen kann, zeigt Abbildung 2.4. Die Dichten der Normalverteilungen sind durch die Höhenlinien im Konturplot angedeutet, während die Hintergrundfarbe die Klassenzuordnung wiedergibt, die für unbekannte Beispiele vorgenommen würde. Die optimale Trenngrenze des Bayes-Klassifizierers wird durch die gestrichelte Linie gezeigt, also nur approximiert.

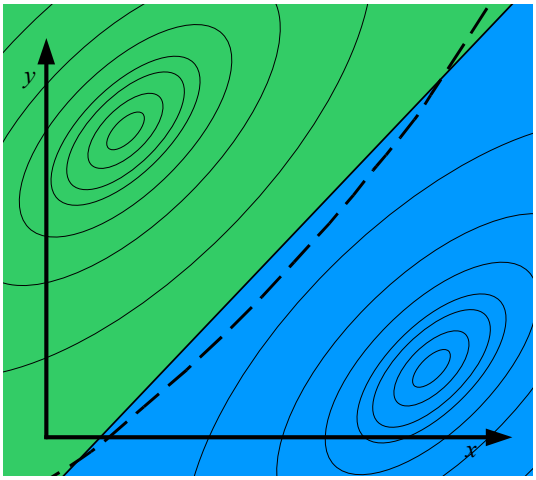


Abbildung 2.4.: Modellrestriktionen einer LDA

2.4. Support Vector Machine

Als drittes Verfahren wird im Folgenden die Support Vector Machine, häufig auch mit SVM abgekürzt, vorgestellt. Sie liefert im einfachsten Fall eine trennende Hyperebene wie die LDA. Die Berechnung der Hyperebene folgt dabei allerdings einer anderen Motivation, wodurch sie sich auf nicht lineare Trenngrenzen verallgemeinern lässt, aber auf Zwei-Klassen-Probleme beschränkt bleibt.

Definition 2.4.1 (Zwei-Klassen-Problem)
Man bezeichnet eine Domäne als Zwei-Klassen-Problem, wenn gilt:

$$|\mathcal{C}| = 2 \tag{2.17}$$

Die folgenden Abschnitte führen die SVM Schritt für Schritt ein, um die mögliche Komplexität des Modells zu vermitteln. Da die Abschnitte sich aber auch auf diese Vermittlung beschränken sollen, lassen sie notwendigerweise Details zur

Berechnung einer SVM, sowie zur Herleitung und Praxis weitestgehend aus. Eine tiefergehende Einführung in die Theorie vermittelt (Burges 1998).
Der Ansatz der LDA führte, wie im letzten Kapitel gesehen, zu einer Hyperebene der Form $\mathbf{x}^T\beta = \beta_0$, die sich aus den geschätzten Normalverteilungen ergab. Die Schätzung der Normalverteilungen war also notwendig um die Parameter der Gleichung zu bestimmen. Falls die Beispiele so im Beispielraum angeordnet sind, dass auf den beiden Seiten einer passend gewählten, linearen Hyperebene nur Beispiele einer Klasse liegen, spricht man von linear separierbaren Klassen. In diesem einfachsten Fall lässt sich die trennende Hyperebene einer SVM ebenfalls auf eine solche Form bringen, wird aber auf eine direktere, intuitivere Art bestimmt.
Dazu werden die Parameter der Hyperebene so bestimmt, dass die Hyperebene die Punktewolken zweier Klassen voneinander trennt und dabei den größten möglichen Abstand zu den nächsten Beispielen der Klassen einhält. Abbildung 2.5 illustriert dies. Die jeweils nächsten Punkte zur Hyperebene werden hier mit einem Kreis markiert. Sie definieren auch die parallelen Hyperebenen H_1 und H_2. Die durch den durchgezogenen Strich markierte Entscheidungsgrenze liegt genau in der Mitte zwischen diesen beiden Ebenen H_1 und H_2. Der Abstand zwischen H_1 und H_2 wird als Margin c bezeichnet. In der intuitiven Annahme, dass ein größerer Abstand zwischen den Beispielen der Klassen zu einer besseren Trennung und damit zu geringeren Fehlklassifikationswahrscheinlichkeit auf neuen Beispielen führt, versucht man ihn zu maximieren.

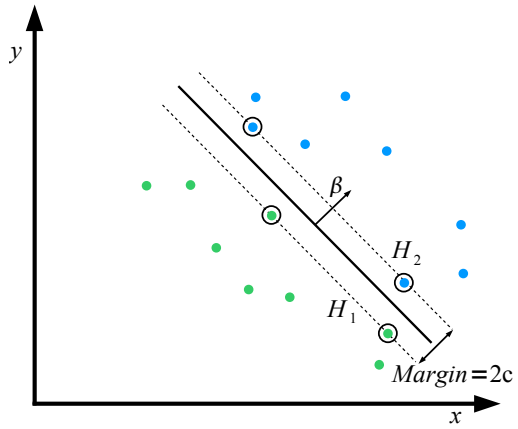


Abbildung 2.5.: Lineare SVM im separierbaren Fall

Um nun den Parametervektor β der Hyperebene zu bestimmen, müssen zunächst einige Vorbereitungen getroffen werden. Die zwei Klassen müssen durch eine 1 und -1 Kodierung des Labels repräsentiert werden. Dadurch lässt sich die abschließende Klassifikationsregel für die lineare SVM leicht definieren:

Definition 2.4.2 (Klassifikationsregel der linearen SVM)
Wurden die Parameter β und β_0 bestimmt, lässt sich für ein beliebiges $\mathbf{x}$ berechnen auf welcher Seite der Hyperebene es liegt, indem man es in folgende Gleichung einsetzt:

$$\mathrm{f}(\mathbf{x}) = \mathbf{x}^T\beta - \beta_0 \tag{2.18}$$

Mit Hilfe der Vorzeichenfunktion lässt sich dann die Klassifikationsregel formulieren:

$$\mathrm{c}(\mathbf{x}) = sign(\mathbf{x}^T\beta - \beta_0) \tag{2.19}$$

Diese kompakte Formel ergibt sich durch die 1 und -1 Kodierung der Label und das Verhalten der Funktion $f(\mathbf{x})$, die je nachdem auf welcher Seite der Hyperebene $\mathbf{x}$ liegt, positive oder negative Werte zurückgibt.

Da der Betrag, also anschaulich die Länge, des Vektors β für die Ausrichtung der Hyperebene keine Rolle spielt, kann er benutzt werden um die Größe des Margins C zu bestimmen:

$$\mathrm{C} = \frac{1}{\|\beta\|} \tag{2.20}$$

So ist sichergestellt, dass allen Punkten außerhalb des Margins betragsmäßig Werte größer als 1 zugewiesen werden. Gleichzeitig lässt sich der Margin jetzt auch maximieren, indem $\|\beta\|$ minimiert wird. Statt $\|\beta\|$ zu minimieren kann aber äquivalent auch $\frac{1}{2}\|\beta\|^2$ optimiert werden. Als Optimierungsschritt ergibt sich dadurch:

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 \tag{2.21}$$

unter Nebenbedingungen, die die korrekte Klassenzuordnung der Trainingsbeispiele sicherstellen:

$$\forall i : \mathrm{y}_i(\mathbf{x}_i^T\beta + \beta_0) \geq 1 \tag{2.22}$$

Dieses Optimierungsproblem liefert die Parameter β und β_0 der Hyperebene, die die Entscheidungsgrenze markiert, und somit das Entscheidungsmodell der SVM.

Linear nicht separierbarer Fall

Da Datensätze in der Praxis häufig linear nicht fehlerfrei zu separieren sind, muss vorgesehen werden, dass Beispiele innerhalb des Margins oder sogar auf der

falschen Seite der Hyperebene liegen können. Dazu führt man sogenannte Slackvariablen ξ_i ein, die die Nebenbedingungen aus 2.22 aufweichen, dafür aber den zu minimierenden Term erweitern. Mittels eines Tuningparameters γ kann festgelegt werden, wie stark die Fehlplatzierungen von Beispielen bestraft werden. Man erhält jetzt:

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i \tag{2.23}$$

unter den Nebenbedingungen

$$\forall i : \mathrm{y}_i(\mathbf{x}_i^T\beta + \beta_0) \geq 1 - \xi_i \tag{2.24}$$

$$\forall i : \xi_i \geq 0 \tag{2.25}$$

Lösung des Optimierungsproblems

Das entstehende Minimierungsproblem 2.23 unter den Nebenbedingungen 2.24 und 2.25 lässt sich mit Hilfe des Lagrange-Ansatzes lösen. Dazu werden sogenannte Lagrange-Multiplikatoren eingeführt, mit denen die Nebenbedingungen direkt in die zu minimierende Funktion eingebunden werden. Dies führt dann zu folgender Funktion, die man als primales Problem bezeichnet:

$$L_P = \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i(\mathrm{y}_i(\mathbf{x}_i^T\beta + \beta_0) - 1 - \xi_i) - \sum_{i=1}^{N} \mu_i\xi_i \tag{2.26}$$

Über die partiellen Ableitungen nach β, β_0 und ξ_i lässt sich das sogenannte duale Problem herleiten. Dabei heben sich alle eingeführten Lagrange Multiplikatoren bis auf die α_i wieder auf und es ergibt sich folgende Formel:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \tag{2.27}$$

Maximiert man diesen Term über die α_i, ergeben sich die optimalen α_i, mit deren Hilfe man β bestimmen kann:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{2.28}$$

Da die Zusammenhänge in realen Datensätzen häufig nicht-linear sind, erlaubt das Zulassen von falsch klassifizierten Beispielen zwar, überhaupt eine Hyperebene zu finden, die Klassifikationsleistung kann aber nicht gut werden, solange das

Modell die komplexeren Zusammenhänge nicht ausdrücken kann.
Bei anderen Verfahren bedient man sich oft der Basisexpansion, das heißt man berechnet aus den vorhandenen Dimensionen neue, zusätzliche. Legt man eine lineare Trenngrenze durch diesen aufgeblähten Raum, kann diese eine nicht-lineare Trenngrenze in den Ursprungsraum projizieren. Da aber jede Expansion der Basis die benötigte Rechenzeit verlängert, bedient man sich bei der SVM eines Tricks um die explizite Berechnung der Transformation zu sparen und gleichzeitig doch die expandierte Basis zu benutzen:
In Gleichung 2.27 sieht man, dass von den Beispielen während der Optimierung nur das Skalarprodukt gebildet wird. Würde man zunächst eine Basisexpansion $\Phi(\mathbf{x})$ durchführen, ließe sich das wie folgt schreiben:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \tag{2.29}$$

Es existiert nun aber eine Klasse von Funktionen, den Kernfunktionen, die $\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ berechnen können, ohne die Basisexpansion erst explizit durchzuführen.

Definition 2.4.3 (Kernfunktion)
Eine Funktion $K : X \times X \to \mathbb{R}$ heißt Kernfunktion, oder kurz Kernel, falls gilt:
Es existiert ein Vektorraum F *mit Skalarprodukt und eine Funktion $\Phi : X \to \mathrm{F}$, so dass gilt:*

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}') \quad \forall \mathbf{x}, \mathbf{x}' \in X \tag{2.30}$$

Beispiel 2.4.1 (Einige Kernfunktionen)
Die folgende Auswahl von Kernfunktionen ist weit verbreitet und wird in der Praxis häufig angewandt:
Linearer Kern:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}' \tag{2.31}$$

Polynomieller Kern n-ten Grades:

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^n \tag{2.32}$$

Radialer Basis Kern:

$$K(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{c}} \tag{2.33}$$

Man sieht, dass Gleichung 2.29 bereits so formuliert ist, dass sich eine Kernfunktion durch Einsetzen einfach anwenden lässt. Es ergibt sich dann als neues

Optimierungsproblem:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, \mathbf{x}_j) \tag{2.34}$$

Entsprechend ändert sich aber auch die Klassifikationsregel aus Gleichung 2.19. Wenn man die Lösung für β einsetzt, sieht man, dass auch hier das innere Produkt berechnet wird:

$$\begin{aligned} \mathrm{c}(\mathbf{x}) &= sign(\mathbf{x}^T \beta - \beta_0) && (2.35) \\ \Leftrightarrow \ \mathrm{c}(\mathbf{x}) &= sign(\mathbf{x}^T \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i - \beta_0) && (2.36) \\ \Leftrightarrow \ \mathrm{c}(\mathbf{x}) &= sign(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}^T \mathbf{x}_i - \beta_0) && (2.37) \end{aligned}$$

Daher muss an dieser Stelle auch die Kernfunktion benutzt werden:

$$\mathrm{c}(\mathbf{x}) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) - \beta_0 \right) \tag{2.38}$$

Dies sorgt aber dafür, dass sich β nicht mehr berechnen lässt, sondern tatsächlich für jeden zu klassifizierenden Vektor die komplette Summe der Kernfunktionen über alle Trainingsbeispiele mit dem Vektor selbst gebildet werden muss. Daraus folgt, dass die Struktur im Datensatz durch die Gewichte der Beispielvektoren abgebildet wird. Selbst bei kleinen Datensätzen ist so schnell der Punkt erreicht, bei dem Menschen die Berechnung nicht mehr nachvollziehen können und daher auch Domänenspezialisten nur in sehr seltenen Sonderfällen in der Lage sind, aus dieser Funktion Wissen über die Domäne zu ziehen. Die SVM erlaubt somit zwar eine hohe Klassifikationsgüte, bietet aber kaum Einsicht in die Zusammenhänge selbst.

3. Problemstellung und Ansätze

Dieses Kapitel wird zunächst eine Einführung in die zugrundeliegende Problematik geben und daraus erste Ideen ableiten. Diese Lösungsansätze werden dann in den anschließenden Abschnitten genauer beleuchtet und die Auswahl der in dieser Arbeit verwendeten Verfahren motiviert.

3.1. Problemstellung

Ziel dieser Arbeit ist es, eine Methode vorzustellen, mit der hochdimensionale Entscheidungsmodelle visualisiert werden können, um Zusammenhänge in der Domäne zu entdecken und zu verstehen. Im letzten Kapitel wurden drei verschiedene Entscheidungsmodelle vorgestellt, an denen nun die Problematik der Visualisierung erläutert werden soll. Wie gezeigt, bieten die Modelle der drei Verfahren von sich aus gänzlich unterschiedlich umfangreiche und unterschiedlich verständliche Möglichkeiten zur Visualisierung. Diese beruhen aber jeweils auf den speziellen formalen Eigenschaften des Modells. Während sich die achsen-orthogonalen Schnitte des Baumlerners anschaulich in einer Baumstruktur darstellen lassen, bleibt bei der LDA nur das Betrachten der Komponenten eines hochdimensionalen Koeffizientenvektors. Im Modell der SVM werden die Zusammenhänge durch ein Gewicht pro Trainingsbeispiel und die Bildung einer Kernfunktion abgebildet und es entzieht sich aufgrund der Anzahl und komplexen Wirkweise dieser Faktoren der Anschauung.

Durch diese speziellen Eigenschaften ist es aber unmöglich die Modelle untereinander, über Verfahrensgrenzen hinweg, zu vergleichen, wenn man auch ihre Bedeutung, also die gefundenen Zusammenhänge selbst, vergleichen möchte. Schließlich entbehrt der Vergleich von einem Koeffizientenvektor mit einem Baum jeglicher Grundlage. Bisherige Ansätze, wie (Johnston 2002), begnügen sich daher mit dem Vergleichen von Kennzahlen wie der Klassifikationsleistung auf einem Testdatensatz, oder beschränken sich wie in (Thearling et al. 2002) auf Vergleiche innerhalb von Modellen eines Verfahrens.

Diese Arbeit stellt einen neuen Ansatz vor, um beliebige Entscheidungsmodelle unabhängig von ihrer formalen Definition zu visualisieren. Um diese Unabhängig-

keit zu erreichen muss zunächst der kleinste gemeinsame Nenner aller möglichen Entscheidungsmodelle gesucht werden, um auf ihm aufbauend eine Möglichkeit zur Visualisierung allgemeiner Entscheidungsmodelle zu finden. Dieser Nenner findet sich naheliegenderweise in der Definition für Entscheidungsmodelle 2.1.1. Da in ihr nichts über die statistischen Eigenschaften oder formalen Definitionen der Modelle ausgesagt wird, können sie auch nicht zu Visualisierung herangezogen werden. Gemein ist allen Entscheidungsmodellen nur, dass sie einer Funktion $f : \mathbb{D}^m \rightarrow \mathcal{C}$ entsprechen. Damit lässt sich das Problem darauf zurückführen, eine Funktion zu visualisieren, die Elemente aus einem hochdimensionalen, kontinuierlichen Raum auf Elemente einer endlichen und meist kleinen Menge abbildet.

Da der menschliche Wahrnehmungsapparat auf die Verarbeitung alltäglicher Dinge optimiert ist, die in der Regel dreidimensional und farbig wahrgenommen werden, muss die Visualisierung ebenfalls in einer niedrigen Dimension erfolgen. Andernfalls würde das Ziel, Verständnis zu vermitteln, verfehlt werden. Die Anzeige auf gewöhnlichen Monitoren bedeutet eine weitere Einschränkung der Visualisierung auf zwei Dimensionen und Farbe. Da in zwei Dimensionen nicht so viele Informationen dargestellt werden können wie im hochdimensionalen Ursprungsraum, muss während der Dimensionsreduktion eine geeignete Auswahl der Informationen getroffen werden.

Dabei handelt es sich bei dem hochdimensionalen Ursprungsraum um den Beispielraum unserer Problemdomäne. Wie in 2.1.2 festgestellt, sind dessen Eigenschaften unbekannt und sollen erst durch die Modellvisualisierung verstanden werden. Da beim Training des Modells Beispieldaten benötigt werden, stehen diese auch zur Visualisierung zur Verfügung. Diese sollen im Folgenden genutzt werden um einen Daten-basierten Ansatz zur Dimensionsreduktion zu verfolgen:

Die Idee der Arbeit ist es deshalb, aus diesen Daten eine Abbildung des Datenraumes zu berechnen, die die Eigenschaften der Daten möglichst gut wiedergibt und anhand derer das Verhalten der Entscheidungsmodelle untersucht werden kann. Um in einem dem Benutzer vertrauten Rahmen zu bleiben, soll die Analogie einer Landkarte genutzt werden. Der Datenraum wird also auf eine zweidimensionale Fläche projiziert, auf dem Farbe weitere Informationen vermitteln kann um zum Beispiel das Klassifikationsergebnis der Entscheidungsmodelle anzuzeigen.

Um den hochdimensionalen Datenraum auf eine zweidimensionale Fläche zu projizieren, benötigen wir eine Methode zur Dimensionsreduktion:

Definition 3.1.1 (Dimensionsreduktion)
Eine Funktion zur Dimensionsreduktion $\mathbf{r} : \mathbb{R}^m \rightarrow \mathbb{R}^p$ *bildet Elemente eines m-dimensionalen Definitionsraumes auf den p dimensionalen Bildraum ab. Dabei gilt:*

$$m > p \tag{3.1}$$

Verfahren zur Bestimmung solcher Abbildungen werden im Abschnitt 3.2 beschrieben.

Durch die Anwendung eines Verfahrens zur Dimensionsreduktion wurden alle Beispiele auf eine zweidimensionale Fläche projiziert. Will man das Verhalten eines Entscheidungsmodells auf dieser Fläche visualisieren, steht man allerdings vor dem Problem, dass die Entscheidungsmodelle nur über dem Beispielraum definiert sind, nicht über der Fläche des Kartenraums. Da eine dimensionsreduzierende Abbildung nicht injektiv sein kann, kann keine Umkehrabbildung bestimmt werden. Damit ist der Rückweg von der Fläche in den Beispielraum zunächst verbaut, so dass das Verhalten des Modells nur an wenigen Punkten der Fläche geprüft werden kann. Nämlich genau an den Positionen, die von den abgebildeten Beispielen eingenommen werden.
Eine Lösung für dieses Problem wäre, für jeden Punkt aus dem Beispielraum das Modell anzuwenden und im Kartenraum eine Art Voting durchzuführen, um herauszufinden, mit welcher Klasse die Urbilder am häufigsten klassifiziert wurden. Da aber dieser Mechanismus nicht nur verwirrend für den Benutzer ist, sondern der Beispielraum auch unendlich groß ist und selbst die Klassifikation einer relativ kleinen, endlichen Teilmenge nicht in angemessener Rechenzeit durchgeführt werden kann, verbietet sich dieser Ansatz.
Eine intuitivere Alternative stellt die Verwendung von Interpolationsverfahren dar. Um sie anwenden zu können, bestimmt man die Abbildungen der Beispiele. Dadurch ergeben sich Tupel, die sowohl eine Position im Kartenraum einnehmen, nämlich die Abbildung des Beispiels, als auch im Beispielraum, nämlich das Beispiel selbst. Aus dieser Menge von bekannten Tupeln können Werte für alle unbekannten Kartenraumpositionen berechnet werden. Diese Beispielraumwerte können anschließend durch das Entscheidungsmodell klassifiziert werden.
Entsprechend ist eine geeignete Interpolationsfunktion nötig, eine Auswahl wird im Abschnitt 3.3 gegeben.

Um sich in den zahlreichen Abbildungen einzelner Dimensionen zu orientieren, benötigt der Anwender geeignete Darstellungsformen. Neben der offensichtlichen Anforderung, die Werte in den einzelnen Dimensionen anzeigen zu können und sich die Klassifikationsergebnisse als transparente Schicht darüber zu legen, sind weitere Hilfsmittel möglich. Diese sollen Informationen aus den Daten berechnen und sie als weitere Schicht, hier Layer genannt, anzeigen. Alle angezeigten Layer werden überlagert und verfügen dazu über einen Transparentwert. Aus der Überlagerung ergibt sich das Bild, das der Anwender auswerten kann. Die Beschreibung der verschiedenen Layer erfolgt in Kapitel 7.

3.2. Verfahren zur Dimensionsreduktion

Da die Verfahren zur Dimensionsreduktion im Rahmen dieser Arbeit nur angewendet werden sollen, gibt dieser Abschnitt lediglich eine kurze Einführung. Diese wird genutzt um die Auswahl der später verwendeten Verfahren zu motivieren, denn in der Vergangenheit wurden zahlreiche Verfahren zur Dimensionsreduktion vorgestellt, die sich inzwischen zu Standardtechniken entwickelt haben. Eine allgemeine Übersicht über solche Verfahren gibt (Fodor 2002) und eine detailliertere Einführung in eine Auswahl (Carreira-Perpinán 1997). Verfahren zur Dimensionsreduktion lassen sich allgemein in zwei Klassen einteilen: lineare und nicht-lineare Verfahren.

3.2.1. Lineare Verfahren zur Dimensionsreduktion

Alle linearen Verfahren lassen sich durch eine Transformationsmatrix $\mathbf{T}$ ausdrücken, so dass sich die Matrix $\mathbf{E}'$ der transformierten Beispiele aus der nicht transformierten Beispielmatrix $\mathbf{E}$ ergibt durch:

$$\mathbf{E}' = \mathbf{E}\mathbf{T}^T \tag{3.2}$$

Dabei werden die einzelnen Beispiele als Zeilen in der Beispielmatrix kodiert. Für einzelne Beispiele ergibt sich dadurch die folgende Transformationsvorschrift:

Definition 3.2.1 (Lineare Transformation)
Sei ein Beispielvektor $\mathbf{x}$ *und eine Transformationsmatrix* $\mathbf{T}$ *gegeben. Dann ergibt sich das transformierte Beispiel* $\mathbf{x}'$ *durch:*

$$\mathbf{x}' = \mathbf{T}\mathbf{x} \tag{3.3}$$

Damit lässt sich jede Komponente x'_i von $\mathbf{x}'$ darstellen als:

$$\mathrm{x}'_i = T_{i1}x_1 + T_{i2}x_2 \ldots T_{im}x_m \tag{3.4}$$

$$= \sum_{j=1}^{m} T_{ij}x_j \tag{3.5}$$

Man spricht von einer linearen Transformation, da die einzelnen Vektorkomponenten mit einem Skalar multipliziert werden.

Solange $\mathbf{T}$ eine quadratische Matrix ist, findet keine Dimensionsreduktion statt und $\mathbf{x}'$ ist von derselben Dimension wie $\mathbf{x}$. Tatsächlich lässt sich mit der Inversen von $\mathbf{T}$ einfach eine Umkehrabbildung finden, falls $\mathbf{T}$ vollen Zeilenrang hat. Eine Reduktion findet statt, sobald $\mathbf{T}$ weniger Zeilen als Spalten besitzt, so dass $\mathbf{x}'$ weniger Komponenten besitzt als das ursprüngliche Beispiel. Dann lässt sich keine Umkehrabbildung mehr finden, es werden Informationen verworfen.

Bekannte lineare Verfahren sind die Hauptkomponentenanalyse, kurz PCA, Faktorenanalyse (Mardia et al. 1979) und sowie die Independent Component Analysis (Comon 1994). Da diese Arbeit auch die Visualisierung von Modellen nichtlinearer Lernverfahren behandelt und eine Anwendung dieser Modelle nur auf Daten mit nicht-linearem Zusammenhang lohnenswert ist, wird nur ein lineares Verfahren implementiert. Damit soll hauptsächlich die Hypothese überprüft werden, dass lineare Dimensionsreduktionen nicht mächtig genug zur verständlichen Visualisierung nicht-linearer Zusammenhänge sind. Als Vertreter für lineare Verfahren wird dabei die PCA gewählt, da es sich um ein weit verbreitetes Verfahren handelt. Ihre Funktionsweise wird in Kapitel 5 erklärt.

Da sich lineare Verfahren auf eine einzelne Matrixmultiplikation zurückführen lassen, können sie auch nachträglich auf Beispiele angewandt werden, die zum Transformationszeitpunkt noch unbekannt waren. Alle anderen Methoden zur Dimensionsreduktion fallen in die Klasse der nicht-linearen Verfahren, bei denen dies nicht möglich ist.

3.2.2. Nicht-lineare Verfahren zur Dimensionsreduktion

Wie der Name schon andeutet, gehört dieser Klasse genau die Komplementmenge zu den linearen Verfahren an. Dazu gehören auch die Verfahren, die auf Self-Organizing Maps beruhen. Wie bereits dem Titel dieser Arbeit zu entnehmen ist, bilden diese den Hauptteil der Arbeit. Alle hier vorgestellten Varianten werden daher im Kapitel 4 detailiert beschrieben.

Zu Vergleichszwecken wird ein weiteres nicht-lineares Verfahren vorgestellt: Locally Linear Embedding wie in (Roweis und Saul 2000) beschrieben. Sowohl selbstorganisierende Karten als auch Locally Linear Embedding sind in der Lage Mannigfaltigkeiten in den Daten zu entdecken und korrekt wiederzugeben. Der Vergleich dient also in erster Linie dazu herauszufinden, welches Verfahren verständlichere Ergebnisse liefert. Locally Linear Embedding wird zusammen mit der PCA in Kapitel 5 beschrieben.

3.3. Interpolationsverfahren

Nachdem die Dimensionsreduktion durchgeführt und die Beispielvektoren auf eine Position im Kartenraum abgebildet wurden, müssen Werte an den nicht besetzten Positionen berechnet werden. Erst dadurch kann dem Nutzer eine vollständige und an allen Stellen definierte Karte präsentiert werden. Dazu benötigen wir Approximations- oder Interpolationsverfahren, deren Auswahl in diesem Abschnitt begründet wird.

Definition 3.3.1 (Approximationsverfahren)
Ein Approximationsverfahren ist ein Verfahren, das eine Funktion $\hat{f} : \mathbb{R}^m \to \mathbb{R}$ *zur Schätzung eines Funktionsverlaufes einer tatsächlichen Funktion* $f : \mathbb{R}^m \to \mathbb{R}$ *anhand bekannter Wertepaare* $(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, \ldots, n$ *mit* $\mathbf{x}_i \in \mathbb{R}^m$ *bestimmt. Dabei muss für die Wertepaare die Eindeutigkeitsbedingung* $\mathbf{x}_i \neq \mathbf{x}_j \quad \forall i, j = 1, \ldots, n, i \neq j$ *gelten.*

Definition 3.3.2 (Interpolationsverfahren)
Ein Interpolationsverfahren ist ein Approximationsverfahren, das nur Funktionen bestimmt, die zusätzlich die Interpolationseigenschaft erfüllen.

Definition 3.3.3 (Interpolationseigenschaft)
Sei $\hat{f} : \mathbb{R}^m \to \mathbb{R}$ *eine Funktion zur Schätzung des tatsächlichen Verlaufs einer Funktion* $f(\mathbf{x})$ *und seien* $(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, \ldots, n$ *mit* $\mathbf{x}_i \in \mathbb{R}^m$ *bekannte Wertepaare der tatsächlichen Funktion. Dann hat* $\hat{f}(\mathbf{x})$ *die Interpolationseigenschaft, falls gilt:*

$$\hat{f}(\mathbf{x}_i) = f(\mathbf{x}_i) \qquad \forall i = 1, \ldots, n$$

Eine Funktion hat also die Interpolationseigenschaft, wenn sie an den bekannten Stellen genau die vorgegebenen Werte zurückgibt.

Nachdem die Dimensionsreduktion die Beispiele im Kartenraum positioniert hat,

können die Beispiele mit den Positionen als bekannte Wertepaare verwendet werden um eine Approximation oder Interpolation zu bestimmen. Die Verwendung von Approximationsverfahren hat den Nachteil, dass nicht sichergestellt ist, dass an der Position eines Beispiels die geschätzten Werte den Werten des Beispiels entsprechen. Dadurch kann der Benutzer verwirrt werden, vor allem wenn das Beispiel vom Entscheidungsmodell zu einer Klasse, der Wert der Karte an seiner Position aber zu einer anderen Klasse zugeordnet wird. Diese Arbeit wird sich also auf die Beschreibung von Interpolationsverfahren konzentrieren, jedoch zu Vergleichszwecken ein Approximationsverfahren vorstellen, um abwägen zu können, ob Vor- oder Nachteile überwiegen.

Eine wichtige Eigenschaft aller Verfahren ist dabei die Stetigkeit, die visuell die Glätte des Funktionsverlaufs bestimmt:

Definition 3.3.4 (Stetigkeit von Interpolationsverfahren)
Man bezeichnet ein Interpolationsverfahren als C_0-stetig, falls dessen Interpolationsfunktion $\hat{f}$ selbst stetig ist. Man bezeichnet es als C_i-stetig, falls die i-te Ableitung seiner Interpolationsfunktion stetig ist.

Menschen akzeptieren approximierte Funktionsverläufe eher als natürlich, je höher die Stetigkeit ausfällt. Wird zum Beispiel linear interpoliert, also im eindimensionalen die einzelnen Punkte durch Geraden verbunden, dann wird diese Approximation als weniger übereinstimmend mit der realen Funktion empfunden, als eine stetige, glatte Funktion.

3.3.1. Klassifizierung von Interpolationsverfahren

Interpolationsverfahren lassen sich über zwei Ansätze in unterschiedliche Klassen gliedern. Zum einen kann eine strukturelle Unterscheidung dabei über die Anforderungen an die Menge der bekannten Werte, also datenbasiert, getroffen werden, denn die meisten Interpolationsverfahren benötigen eine komplette orthogonale Gitterstruktur von bekannten Punkten.

Zum anderen lassen sich nach (Hoschek und Lasser 1992) die Verfahren weiterhin über ihre Funktionsweise in vier verschiedene Gruppen trennen: gewichtungsbasierte Verfahren, radiale Basisfunktionsmethoden sowie Verfahren, die auf einer Finite Elemente oder Spline-basierten Methoden beruhen.

Datenbasierte Klassifizerung

Definition 3.3.5 (Gitterinterpolator)
Ein Interpolationsverfahren gehört zur Klasse der Gitterinterpolatoren, falls die Menge der Wertepaare $(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, \ldots, n$ *mit* $\mathbf{x}_i \in \mathbb{R}^m$ *aus Paaren besteht, die an den Kreuzungspunkten eines äquidistanten und orthogonalen Gitters gemessen wurden. Jeder Kreuzungspunkt muss dabei in der Menge enthalten sein.*

Alle Verfahren die keine Anforderungen an die Position der bekannten Werte stellen, heißen Scatterinterpolator:

Definition 3.3.6 (Scatterinterpolator)
Ein Interpolationsverfahren gehört zur Klasse der Scatterinterpolatoren, falls es außer der Eindeutigkeitsbedingung keine Anforderungen an die Menge der Wertepaare stellt.

Damit ist jeder Scatterinterpolator automatisch ein Gitterinterpolator, die Menge der Scatterinterpolatoren stellt also eine Obermenge der Gitterinterpolatoren dar.

Funktionsbasierte Klassifizierung

Im Folgenden werden verschiedene mathematische Herangehensweisen vorgestellt, aus denen sich die einzelnen Verfahren ableiten. Diese Herangehensweisen können genutzt werden um die Verfahren in Klassen einzuteilen.

Gewichtungsbasierte Verfahren beruhen darauf, für jeden Vektor $\mathbf{x}$ Gewichte für alle bekannten Wertepaare zu bestimmen und das gewichtete Mittel über deren Funktionswerte zu berechnen. Mit geeigneten Gewichtungsfunktionen lässt sich die Interpolationseigenschaft herstellen. Da für jeden neuen Eingabevektor neue Gewichte bestimmt werden müssen, sind diese Verfahren in der Anwendung rechenaufwendig. Diese Arbeit wird aus diesen Verfahren eine Variante des Shepard Ansatzes vorstellen, die erstmals in (Shepard 1968) beschrieben wurde. Eine genaue Beschreibung erfolgt in Abschnitt 6.1.

Radiale Basisfunktionsmethoden basieren auf der Bestimmung korrekter Parameter für eine Funktion, so dass diese die Interpolationsbedingung erfüllt und zum Interpolanten wird. Die Besonderheit ist, dass diese Parameter nicht nur aus Skalaren bestehen, sondern auch Polynome k-ten Grades umfassen können.

Definition 3.3.7 (Interpolant der radialen Basisfunktionsmethoden)
Der Interpolant $\hat{f}_r : \mathbb{R}^m \to \mathbb{R}$ der radialen Basisfunktionsmethoden entstammt aus der Funktionsfamilie:

$$\hat{f}_r(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i R(d(\mathbf{x}, \mathbf{x}_i)) + p_m(\mathbf{x}) \tag{3.6}$$

$$mit \quad p_m(\mathbf{x}) = \sum_{j=1}^{m} \beta_j p_j(\mathbf{x}) \tag{3.7}$$

Dabei ist R eine radial positive Funktion abhängig vom Abstand $d(\mathbf{x}, \mathbf{x}_i)$ zwischen dem Eingabewert $\mathbf{x}$ und der Position des bekannten Wertes bei $\mathbf{x}_i$. Weiterhin sind $p_j \in \mathbb{P}^m$, Polynome aus dem Raum der Polynome von höchstens Grad m, und entsprechend auch deren Summe p_m.

Die Parameter dieser Funktion lassen sich bestimmen, indem man ein Gleichungssystem löst. Dieses ergibt sich aus den n Gleichungen, die die Interpolationseigenschaft 3.3.3 sicherstellen und weiterhin aus folgenden m Gleichungen:

$$\sum_{i=1}^{N} \alpha_i p_j(\mathbf{x}) = 0, \quad j = 1, \ldots, m \tag{3.8}$$

Ein generelles Problem bei diesen Verfahren ist das Lösen des Gleichungssystems. Neben der benötigten hohen Rechenzeit stößt man hier häufig an die endliche Genauigkeit der Computer. Durch Rundungsfehler können numerische Instabilitäten entstehen, die die Lösung unmöglich machen oder verfälschen. Hierauf wird in Abschnitt 6.2 eingegangen, in dem Hardy's Multiquadric (Hardy 1971) vorgestellt wird.

Interpolationsverfahren, die auf der Finite Elemente Methode, kurz FEM, basieren, stellen zunächst eine Triangulierung der bekannten Punkte her. Nachdem die gesamte Fläche durch Dreiecke abgedeckt wurde, liegt jeder Punkt innerhalb eines Dreiecks und es wird nur noch das Dreieck benötigt um den Wert des Punktes zu interpolieren. Dadurch entsteht eine polygonale Fläche, die an den Seiten der Dreiecke Kanten bildet, sie ist nur C_0 stetig. Will man einen stetigeren Verlauf, muss man zusätzlich zur Triangulierung noch Ableitungsdaten bestimmen. Diese Arbeit wird jedoch nur den einfachen Fall in Abschnitt 6.3 betrachten.

Spline-basierte Methoden sind als Gitterinterpolatoren weit verbreitet. Sie beruhen darauf, jede Gitterfläche getrennt zu betrachten. So lässt sich für jede Fläche

eine einfache Funktion finden, die alle vier Eckpunkte trifft und mit bereits interpolierte Seiten zu Nachbarflächen übereinstimmt. Durch die Beschränkung auf einfache Funktionen wird ein Überschwingen verhindert, das bei hochgradigen Polynomen häufig auftritt. Dafür muss aber weiterer Aufwand getrieben werden, um eine mehr als C_0-stetige Interpolationsfunktion zu erzeugen, denn wenn die einzige Bedingung das Übereinstimmen der Werte an den Eckpunkten und an den schon bestimmten Seiten ist, dann kann die Ableitung beim Übergehen auf die benachbarte Fläche sprunghaft wechseln. Um also mehr als eine C_0-Stetigkeit zu erreichen, müssen in die Berechnung der Funktionen Informationen über die Ableitungen der Nachbarflächen eingehen.
Splineverfahren erfüllen nur in Gitterstrukturen die Interpolationseigenschaft. Der Ansatz aus (Lee et al. 1997) zeigt aber Möglichkeiten auf, Splineverfahren für scattered Data als Approximationsverfahren zu benutzen. Die Besonderheit dieses Ansatzes ist, dass er sich einer Interpolation unter steigendem Rechenaufwand beliebig nahe annähern kann. Hiermit soll überprüft werden, ob eine nicht exakte Approximation die Zusammenhänge in den Daten eventuell akkurater darstellen kann, als eine Interpolation, die die Beispieldaten exakt abbildet und damit für Überanpassung anfällig sein kann.

4. Self-Organizing Maps

Zu Beginn des Kapitel 3 wurde das Problemfeld beschrieben, in dem Self-Organizing Maps, auch kurz SOMs, eingesetzen werden sollen: Ihre Aufgabe ist es, einen hochdimensionalen Datenraum in eine verständliche niedrige Dimension abzubilden und dabei Informationen über die innere Struktur des hochdimensionalen Raumes zu erhalten, um das Verhalten von Entscheidungsmodellen zu visualisieren. Der darauf folgende Abschnitt 3.2 hat beschrieben, in welche Klassen sich Verfahren zur Dimensionsreduktion unterteilen und Self-Organizing Maps in die Klasse der nicht-linearen Verfahren eingeordnet. Daran anschließend behandelt nun dieses Kapitel die Self-Organizing Maps selbst und wird dazu zunächst ihre allgemeinen Grundlagen beschreiben, die allen Varianten gemein sind. Dadurch wird auch deutlich werden, warum sie zur Klasse der nicht-linearen Verfahren gehören. Auf den Grundlagen aufbauend, wird dann detailiert auf die einzelnen Varianten und ihre Vor- und Nachteile eingegangen werden.

4.1. Grundlagen

Wie oben beschrieben, gehören Self-Organizing Maps zu den Dimensionsreduktionsverfahren. Entsprechend arbeiten sie auf zwei Räumen: Zum einen dem hochdimensionalen Beispielraum $\mathbb{D}^m$, wie durch die Domäne in Definition 2.1.2 bestimmt, und zum anderen auf einem niedriger dimensionalen Zielraum, in den sie die Elemente des Beispielraums abbilden. Dieser Zielraum soll im weiteren Kartenraum genannt werden:

Definition 4.1.1 (Kartenraum)
Der Kartenraum entspricht dem Zielraum einer Dimensionsreduktion durch eine Self-Organizing Map. Er hat eine Dimensionalität von p und wird mit $\mathbb{K}^p$ bezeichnet.

Der Kartenraum entspricht in den meisten Fällen dem $\mathbb{R}^p$, kann aber auch andere Geometrien annehmen, so dass er zum Beispiel einem Toroiden entspricht. Im Weiteren wird nur der Fall $p = 2$ betrachtet, da bei einer Abbildung in einen Kartenraum von höherer Dimension die Visualisierung an Verständlichkeit verliert.

Diese Verständlichkeit zu erlangen ist aber genau der Sinn der Dimensionsreduktion. Dennoch lassen sich alle später beschriebenen Verfahren auf den Fall eines höher dimensionalen Ausgaberaums verallgemeinern.
Eine vom euklidischen Koordinatensystem abweichende geometrische Form des Ausgaberaums nimmt Einfluss auf die Distanzen im Kartenraum. Die Abbildung der Karte auf einen Toroid sorgt zum Beispiel dafür, dass die Punkte auf den Rändern der Karte mit dem Punkt auf dem gegenüberliegenden Rand benachbart sind.

Die Beziehung zwischen hochdimensionalem Beispielraum und niedriger dimensionalem Kartenraum wird in Abbildung 4.1 illustriert. Das Koordinatensystem auf der linken Seite zeigt einen hochdimensionalen Beispielraum, der hier der Anschaulichkeit wegen auf nur drei Dimensionen beschränkt bleiben muss, während das Rechte den zweidimensionalen Kartenraum zeigt. Ziel der Dimensionsreduktion durch die SOM muss es also sein, diese Elemente von links aus dem Beispielraum nach rechts in den Kartenraum abzubilden.

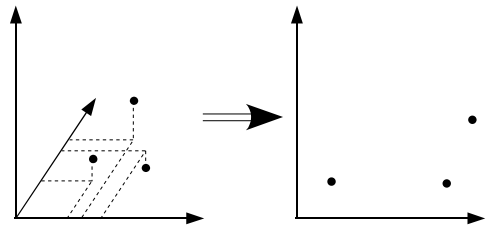


Abbildung 4.1.: Beispiel- und Kartenraum

Alle Varianten der Self-Organinzig Maps verwenden zur Realisierung dieser Abbildung Knoten, im Folgenden Nodes genannt. Diese Nodes bestehen aus einem Paar von Koordinaten, eine aus dem Beispielraum und die andere aus dem Kartenraum. Über diese Koordinaten sind die Nodes in beiden Räumen verankert und bilden so, anschaulich gesprochen, einen Tunnel zwischen den Räumen.

Definition 4.1.2 (Node)
Eine Node N_i *einer Self-Organizing Map besteht aus einem Tupel* $(\mathbf{w}_i, \mathbf{c}_i)$ *mit* $\mathbf{w}_i \in \mathbb{D}^m$ *und* $\mathbf{c}_i \in \mathbb{K}^p$*. Entsprechend ist* $\mathbf{w}_i$ *ein Vektor aus dem Beispielraum und* $\mathbf{c}_i$ *einer aus dem Kartenraum.*

Über die Tupel definiert sich dann die Abbildungsfunktion einer Self-Organizing Map:

Definition 4.1.3 (Abbildungsfunktion der SOM)
Eine Self-Organizing Map realisiert eine Abbildungsfunktion $\mathbf{som} : \mathbb{D}^m \rightarrow \mathbb{K}^p$. *Dazu greift sie auf die Menge* $\mathcal{N} = \{\mathrm{N}_1, \ldots, \mathrm{N}_k\}$ *aller Nodes zurück und benötigt ein beliebiges Distanzmaß d:*

$$\mathbf{som}(\mathbf{x}) = \mathbf{c}_{\arg\min_i \quad d(\mathbf{x},\mathbf{w}_i)} \tag{4.1}$$

Die Elemente des Beispielraums werden also auf die Kartenraumkoordinate derjenigen Node abgebildet, die im Beispielraum am nächsten ist. Wegen dieser nächste-Nachbarn-Relation kann es sich bei einer Self-Organizing Map nicht um eine lineare Dimensionsreduktion handeln, denn sie sorgt für Sprünge in der Abbildungsfunktion, die bei linearen Verfahren nicht vorkommen können.

4.2. Verfahren

Nachdem der obige Abschnitt festgestellt hat, auf welcher gemeinsamen Grundlage alle Self-Organizing Maps arbeiten, bleibt die Frage offen, auf welche Weise eine Menge $\mathcal{N}$ von Nodes bestimmt werden kann, so dass die Abbildung die Forderung erfüllt, die innere Struktur zu erhalten. Dazu gibt es eine breite Variation von Verfahren, von denen im weiteren Verlauf des Kapitels fünf vorgestellt werden sollen: Da ist zum einen die Kohonen Map, der Urvater aller späteren Erweiterungen, wie der Emergent SOM und der Growing Grid SOM. Noch weiter von der Kohonen-Map entfernt sich die Growing Cell SOM, die bei der Positionierung der Nodes als erste auf eine feste Gitterstruktur verzichtet. Die darauf folgende, letzte Variante benutzt dann ein gänzlich anderes Verfahren um die Nodes zu bestimmen. Als einziges Verfahren passt es die Koordinaten der Nodes im Kartenraum und nicht die im Beispielraum an, weshalb man es als Inverted SOM bezeichnen könnte. Da ein gleichnamiger Algorithmus zum Graphlayout aber bereits existiert, von dem sich dieses Verfahren auch ableitet, wird es als Distance SOM bezeichnet. Die Namensgebung ist dadurch motiviert, dass zur Anpassung der Kartenraumpositionen die Distanzen zwischen den Nodes im Beispielraum verwendet werden. Diese Arbeit wird dann zeigen, dass sich mit diesem Ansatz Kernfunktionen verwenden lassen, um die SOM über einem transformierten Beispielraum zu bilden ohne diese Transformation explizit durchführen zu müssen. Diese Erweiterung wird in einem eigenen Abschnitt als Kernel Distance SOM beschrieben.

4.2.1. Kohonen Map

Dieser Abschnitt stellt als Einführung die Kohonen Map vor, die Kohonen aufbauend auf (Kohonen 1981) in (Kohonen 1982) das erste mal beschrieb. Damit veröffentlichte er ein Verfahren, das in den folgenden Jahre zahlreich variiert wurde und deren Varianten allgemein unter dem Begriff der Self-Organizing Maps zusammengefasst werden. Kohonens Ziel war es, eine Abbildung zu finden, die hochdimensionale Eingabesignale in einen niedrig dimensionalen Kartenraum überträgt und dabei eine topographische Sortierung erhält. Das Verfahren ordnet dazu eine initiale Menge von Processing Units im Kartenraum an, die den Nodes aus Definition 4.1.2 entsprechen. Diese Anordnung wird genauer als Initialisierungsphase beschrieben werden. Die Beispielraumkoordinaten $\mathbf{w}$ der initialen Nodes werden dann in einem iterativen Verfahren angepasst, das im Folgenden als Adaptionsphase beschrieben werden soll.

Initialisierungsphase

In der Initialisierungsphase wird ein $k \times l$ großes Array angelegt, dessen Zellen aus Nodes bestehen. Das Array überdeckt dabei den gesamten Kartenraum, so dass alle Zellen äquidistant über den Kartenraum verteilt und in einer rechtwinkeligen Gitterstruktur angeordnet sind. Über die Positionen im Gitter werden die Kartenraumkoordinaten $\mathbf{c}_i$ der Node-Tupel bestimmt. Diese Koordinaten bleiben konstant und werden in der folgenden Adaptionsphase nicht verändert. Die Beispielraumkoordinaten $\mathbf{w}_i$ werden zufällig mit einem Vektor der Länge 1 initialisiert, was anschaulich dafür sorgt, dass jeder Vektor vom Ursprung aus auf einen Punkt des Einheitskreises zeigt. Sie unterscheiden sich also nur im Winkel.

Adaptionsphase

Nachdem in der Initialisierungsphase ein Gitter von Nodes erzeugt wurde, werden in dieser Phase die Kartenraumkoordinaten der Nodes angepasst. Dazu wird wiederholt eines der bekannten Beispiele zufällig mit Zurücklegen ausgewählt und die Nodes an dieses angepasst. Die Anpassung erfolgt dabei mehrstufig:
Zunächst wird diejenige Node $\mathrm{N_a}$ gesucht, für die eine Distanzfunktion zwischen Beispiel $\mathbf{x}$ und Beispielraumkoordinate $\mathbf{w}_a$ minimal wird. Dafür wird die folgende Funktion verwendet:

Definition 4.2.1 (Kohonen Distanzfunktion)
Die Kohonen Distanzfunktion $d_k : \mathbb{D}^m \times \mathbb{D}^m \to \mathbb{R}$ *gibt die Distanz zwischen zwei Elementen des Beispielraums an. Sie ist definiert durch* [1]*:*

$$d_k(\mathbf{x}, \mathbf{w}) = \frac{1}{\mathbf{w}^T\mathbf{x} + 1} \tag{4.2}$$

Die Node $\mathrm{N_a}$ bestimmt sich also durch folgende Gleichung:

$$\mathrm{N_a} = \mathrm{N}_{\arg\min_i \; d_k(\mathbf{x},\mathbf{w}_i)} \tag{4.3}$$

Anschaulich wird so diejenige Node bestimmt, deren Beispielraumvektor den kleinsten Winkel mit dem Beispielvektor einschließt. Man sagt, das Beispiel wurde dann dieser Node zugewiesen. Zeichnung 4.2 illustriert diesen Ablauf. Der linke Teil der Abbildung zeigt ein Gitter von Nodes. Der schwarze Pfeil im Kreis respräsentiert den Beispielraumvektor der Node, der hier notwendigerweise ebenfalls auf zwei Dimensionen begrenzt bleiben muss. Der Vektor des aktuellen Beispiels wird hier durch den roten Pfeil dargestellt. Man sieht, dass es der ähnlichsten Node zugewiesen wird. Das rechte Bild zeigt nun, wie die Node und ihre Nachbarschaft an das Beispiel angepasst werden, indem sich ihre Beispielraumvektoren in Richtung des Beispiels drehen. Die resultierenden Vektoren sind in grau dargestellt.

Formal wird zur Anpassung folgende Adaptionsregel verwendet:

Definition 4.2.2 (Kohonen Adaptionsregel)
Die Kohonen Adaptionsregel weist den anzupassenden Nodes neue Gewichte zu:

$$\mathbf{w}' = \frac{\mathbf{w} + \alpha\mathbf{x}}{\|\mathbf{w} + \alpha\mathbf{x}\|} \tag{4.4}$$

α *bestimmt dabei die Stärke des Einflusses eines einzelnen Beispiels auf die Nodes.*

Bildlich gesprochen, werden also die Beispielraumvektoren der angepassten Nodes ein Stück in Richtung des Beispielvektors gedreht. Dies wird durch die gewichtete Summe gewährleistet. Die Division durch den Betrag des Resultats sorgt dafür, dass die Vektoren wieder normiert werden.

[1] Diese Funktion entspricht nicht exakt der von Kohonen in (Kohonen 1982) vorgeschlagenen, denn sein Algorithmus arbeitete mit einer Ähnlichkeitsfunktion. Durch die Inversenbildung wird sie hier in eine Distanzfunktion umgeformt und durch die Addition von 1 wird der Wertebereich zwischen 0 und 1 eingeschränkt. Damit bleibt die Funktion unter einer Minimierung statt Maximierung analog zum Original und die Beschreibung des Algorithmus konsistent im Zusammenhang.

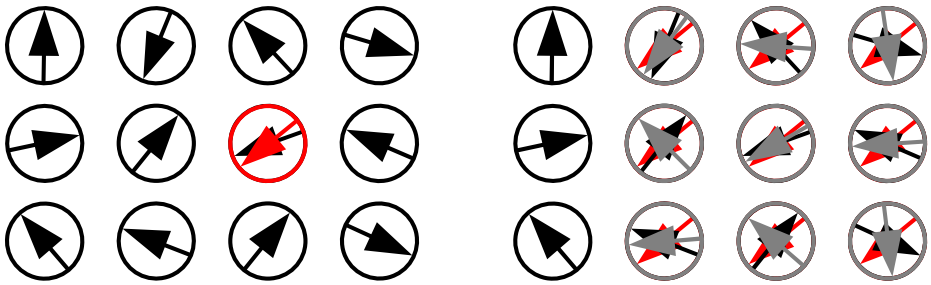


Abbildung 4.2.: Bestimmung der ähnlichsten Node und anschließende Adaption

Erweiterungen

Die Nachbarschaft bezieht sich zunächst auf die im Gitter direkt benachbarten Nodes. Kohonen beschreibt aber auch die Möglichkeit, diesen Begriff auszuweiten. Dazu nimmt er zusätzlich die Nachbarn der direkt benachbarten Nodes hinzu und viertelt für diese den α-Wert. Er stellt dabei fest, dass die Konvergenz in weniger Iterationen erreicht ist, jedoch das Ergebnis nicht ganz so genau ist, da durch die gröbere Nachbarschaft lokale Eigenschaften verloren gehen können. Einer weiteren Vergrößerung der Nachbarschaft bescheinigt er dabei schlechtere Ergebnisse.

4.2.2. Emergent Self-Organizing Map

Dieser Abschnitt beschreibt nicht so sehr eine Erweiterung der Kohonen Map, als vielmehr eine geänderte Anwendung. Ultsch argumentiert in (Ultsch 1999), dass die Anwendung einer Kohonen Map mit nur einigen wenigen Nodes zu einem K-Means ähnlichen Clustering führen würde und damit Potential verschenkt. Sein Vorschlag ist, die Anzahl der Nodes drastisch zu erhöhen, auch über die Anzahl der bekannten Beispiele hinaus. Seine Begründung dafür ist, dass eine genügend große Anzahl von einfachen, zusammengeschalteten Einheiten ein emergentes Verhalten zeigen würden, also ein bestimmtes Verhalten auf einem höheren Niveau. Als Beispiel dienen ihm die Atome einer Laserdiode, deren Verhalten für sich genommen nichts besonderes ist, aber die gemeinsam durch das gleichzeitige Emitieren von Licht derselben Wellenlänge den Laserstrahl erzeugen. Wie auch immer man über die Trefflichkeit dieser Analogie denken mag, unstrittig ist, dass die Verwendung nur einiger weniger Nodes zu einer Art Clustering führt, da Beispiele der ähnlichs-

ten Node zugeordnet werden und diese Node dann an das Beispiel angepasst wird. Daher konvergiert die Node ungefähr im Mittelwert aller ihr zugeordneten Beispiele, was mit dem Cluster-Zentroiden eines Verfahrens wie zum Beispiel K-Means (Hastie et al. 2001) identisch wäre. Da sich die Art der Clusterbildung jedoch unterscheidet, werden nicht dieselben Cluster gefunden, wie bei den klassischen Verfahren, siehe dazu (Ultsch 1995).
Wenn man das ursprüngliche Ziel der Dimensionsreduktion betrachtet, wird jedoch schnell klar, dass mit wenigen Nodes die Abbildungsqualität stark leiden muss. Um eine möglicherweise komplexe Struktur im Beispielraum auf den Kartenraum zu übertragen, benötigt man eine größere Auflösung als ein Gitter mit beispielsweise 25 Nodes bietet. Ultsch schlägt die Verwendung von 1000 bis 10000 Nodes vor, um den Emergenzeffekt zu erzielen. Dies entspräche einer Gitterauflösung von 100×100 Nodes und macht eine Erweiterung der Nachbarschaftsrelation unumgänglich. Denn wie bereits oben beschrieben, gehen lokale Informationen mit steigender Größe der Nachbarschaft verloren, verfolgt man Kohonens Ansatz. Um beides zu ermöglichen, sowohl globale Ordnung, als auch lokale Besonderheiten, führt die Literatur, zum Beispiel (Kohonen 1997), eine weichere Form der Nachbarschaftsbestimmung ein und passt die Adaptionsfunktion an den Abstand und die Anzahl der bereits durchgeführten Trainingsphasen an. Desweiteren wird eine andere Distanzfunktion verwendet, die auch die Länge des Vektors berücksichtigt. Im Ablauf des Algorithmus ändert sich ansonsten nichts, so dass hier nur verkürzt die Änderungen in den einzelnen Phasen vorgestellt werden:

Initialisierungsphase

In der Initialisierungsphase wird lediglich auf eine Normalisierung aller zufälligen Vektoren verzichtet, so dass sie jetzt irgendwo im Einheitshyperkubus liegen können, also jede Komponente Werte zwischen 0 und 1 annehmen kann. Gleichzeitig werden die Dimensionen der Beispiele auf denselben Bereich normalisiert.

Adaptionsphase

Die Adaptionsphase wird nun in einzelne Runden aufgeteilt, in denen jeweils N der Beispiele ohne Zurücklegen gezogen werden. Dabei sei t die Nummer der aktuellen Runde. Desweiteren gibt es eine maximale Anzahl von Runden, die hier t_{max} genannt wird.
Dann erfolgt die Anpassung weiterhin ausgehend von einer Node N_a, für die gilt:

$$a = \arg\min_i \; d_e(\mathbf{x}, \mathbf{w}_i) \tag{4.5}$$

Also für die der Abstand zwischen Beispielraumvektor der Node und Beispiel minimal wird. Dazu wird aber hier der euklidische Abstand benutzt, da für die Entdeckung von Strukturen im Beispielraum der Betrag eines Vektors entscheidend sein kann:

Definition 4.2.3 (ESOM Distanzfunktion)
Die ESOM Distanzfunktion $d_e : \mathbb{D}^m \times \mathbb{D}^m \rightarrow \mathbb{R}$ *gibt die Distanz zwischen zwei Elementen des Beispielraums an. Sie ist definiert durch:*

$$d_e(\mathbf{x}, \mathbf{w}) = \sqrt{\mathbf{w}^T \mathbf{x}} \tag{4.6}$$

Wurde die Node N_a gefunden, werden von ihr ausgehend die neuen Gewichte der benachbarten Nodes angepasst. Dabei wird die Stärke der Anpassung mit der Zeit und zunehmendem Kartenraumabstand zu Node N_a geringer:

Definition 4.2.4 (ESOM Adaptionsregel)
Die ESOM Adaptionsregel weist einer anzupassenden Node $\mathrm{N} = (\mathbf{w}, \mathbf{c})$ *ein neues Gewicht* $\mathbf{w}'$ *zu:*

$$\mathbf{w}' = \mathbf{w} + \alpha(t)(\mathbf{x} - \mathbf{w}) \tag{4.7}$$

mit

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_{t_{max}}}{\alpha_0} \right)^{\frac{t}{t_{max}}} e^{-\frac{\mathbf{c}_a^T \mathbf{c}}{2\sigma(t)}} \tag{4.8}$$

und

$$\sigma t = \sigma_0 \left(\frac{\sigma_{t_{max}}}{\sigma_0} \right)^{\frac{t}{t_{max}}} \tag{4.9}$$

α_0 *ist dabei die Stärke in der ersten Runde,* $\alpha_{t_{max}}$ *entsprechend die Stärke in der letzten Runde. Gleiches gilt für die* σ*.*

Einen Eindruck wie die Anpassungsstärken verlaufen gibt die Abbildung 4.3. In der Grafik wurden die Stärken in jeder Runde t abhängig vom Abstand d mit folgenden Werte berechnet: $\alpha_0 = 0.4$, $\alpha_{t_{max}} = 0.05$, $t_{max} = 10$, $\sigma_0 = 5$ und $\sigma_{t_{max}} = 0.5$. Die Adaptionsstärke wurde auf der y-Achse abgetragen.

Mit einer so angepassten Adaptionsregel lassen sich auch größere Gitter von Nodes anpassen, ohne lokale Informationen zu verlieren. Denn während die starke Adaptionstärke am Anfang die globale Ordnung herstellt, ändert sich mit steigender Rundenzahl nur noch eine immer lokalere Nachbarschaft, so dass diese nur noch an die zugewiesenen Beispiele angepasst wird.

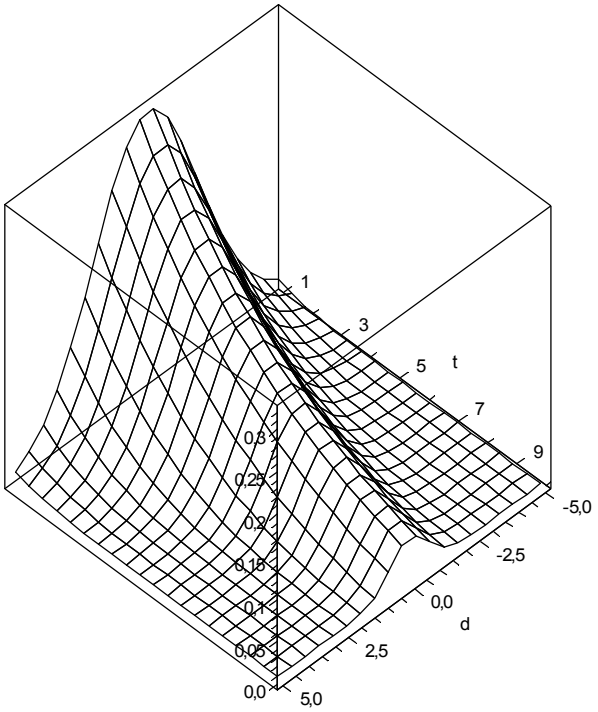


Abbildung 4.3.: Adaptionsstärken in verschiedenen Runden und Distanzen

Erweiterungen

Dieses Verfahren hat allerdings den Nachteil, dass die Größe des Gitters und die Anzahl von Trainingsrunden, sowie die Radien und Anpassungsstärken vom Nutzer passend gewählt werden müssen. Dies erfordert einige Erfahrung und häufiges Herumprobieren. Das nächste Verfahren adressiert diesen Schwachpunkt und versucht die Wahl der richtigen Gittergröße und der Trainingsdauer in das Training einzubauen.

4.2.3. Growing Grid Self-Organizing Map

Die Growing Grid SOM wurde von Fritzke in (Fritzke 1995) mit dem Ziel vorgestellt, die manuelle Wahl der Gittergröße unnötig zu machen. Dazu soll das Git-

ter an Stellen, die eine Gegend mit besonders hoher Wahrscheinlichkeitsdichte im Beispielraum repräsentieren, verfeinert werden. Durch ein eingebautes Abbruchkriterium wird ebenfalls die Wahl der nötigen Trainingsrunden automatisiert. Um beides zu erreichen, benötigt er eine zusätzliche Datenstruktur, in der die Anzahl a_i der einer Node zugewiesenen Beispiele gespeichert wird. Für jede Node N_i gibt a_i also an, wie häufig diese Node die nächste zum jeweiligen Beispielvektor gewesen ist. Der Algorithmus baut auf dem der ESOM auf und bleibt nahezu unverändert, es werden in den Phasen nur folgende Änderungen vorgenommen:

Initialisierungsphase

In der Initialisierungsphase wird wie bisher ein Gitter von Nodes erzeugt, dessen Größe aber fest auf 2×2 beschränkt ist. Gleichzeitig wird die entsprechende Datenstruktur für die a_i angelegt, was mit Hilfe eines gleichgroßen Integerarrays einfach gelöst werden kann.

Adaptionsphase

Wie der Algorithmus der ESOM wird auch hier die Trainingsphase in Runden aufgeteilt. Während jeder Runde wird bei der Zuweisung eines Beispiels zu Node N_i auch deren Zähler a_i um eins erhöht.

Nach einer Runde wird geprüft ob ein Abbruchkriterium erfüllt ist, das sich unter anderem auf die Anzahl der zugewiesenen Beispiele zu einer Node beziehen kann. Ist keines erfüllt, wird die nächste Runde vorbereitet.

Dazu wird diejenige Node N_i gesucht, für die gilt, dass a_i maximal ist. Danach wird diejenige Node N_k bestimmt, die aus der direkten Gitternachbarschaft $\mathcal{S}(\mathrm{N}_i)$ von N_i stammt und für die gilt:

$$d_e(\mathbf{w}_k, \mathbf{w}_i) \geq d_e(\mathbf{w}_{k'}, \mathbf{w}_i) \quad \forall \mathrm{N}_{k'} \in \mathcal{S}(\mathrm{N}_i) \tag{4.10}$$

Es wird also die Nachbarnode gesucht, die im Beispielraum am weitesten entfernt ist. Zwischen beiden Nodes wird dann eine neue Zeile oder Spalte von Nodes, je nach relativer Lage von N_k zu N_i, in das Gitter eingefügt. Die Beispielraumkoordinaten der neuen Nodes ergeben sich als Mittelwert der Beispielraumkoordinaten der beiden gegenüberliegenden Nachbarnodes. Wurde eine Zeile eingefügt, werden also die im Gitter darüber und darunter liegenden Nodes, bei einer Spalte die rechts und links liegenden Nodes zur Berechnung herangezogen. Danach müssen sämtliche Positionen im Kartenraum neu kalkuliert werden, da das Gitter dichter geworden ist und sich entsprechend alle Koordinaten geändert haben. Bevor die neue Runde beginnt, muss noch die Zählerstruktur für die a_i angepasst und alle

Zähler zurückgesetzt werden.

Obwohl man intuitiv annimmt, dass eine Änderung der Anpassungsregel notwendig ist, da das Gitter während der ersten Runden sehr klein ist und im Gegensatz dazu später sehr groß werden kann, ist dies nicht der Fall. Bei näherer Betrachtung der Adaptionsregel aus 4.2.4 stellt man nämlich fest, dass die Adaptionsregel sich nur auf die Kartenraumkoordinaten bezieht und keineswegs auf die Position im Gitter. Dadurch ist sichergestellt, dass die Adaptionsstärke konsistent über das gesamte Training wirkt, sowohl bei sehr gering aufgelösten Karten, als später auch bei großer Gitterdichte.

Erweiterungen

Dieser Algorithmus hat den Vorteil, dass die Gittergröße sich adaptiv an die Daten anpasst und dabei nicht zu klein gewählt werden kann. Da jedoch neben den interessanten Punkten immer eine komplette Zeile oder Spalte eingefügt wird, kann es sein, das zahlreiche eingefügte Nodes garnicht mehr in einem Bereich mit hoher Dichte liegen und damit die Auflösung in Bereichen erhöhen, die nicht interessant sind. Um diese mögliche Verschwendung von Auflösung abzustellen, wäre jedoch eine Trennung von der Gitterstruktur notwendig. Da eine Gitterstruktur aber zahlreiche Vorteile bei der Berechnung bietet, ist fraglich ob die zu erwartenden Gewinne eine Auflösung rechtfertigen. Dieser Aspekt macht den Vergleich mit dem nächsten Verfahren besonders interessant.

4.2.4. Growing Cell Self-Organizing Map

Eine Variante der Growing Grid SOM, die sich von der Gitterstruktur der Nodes löst, hat Fritzke bereits in (Fritzke 1991) und (Fritzke 1992) vorgestellt. Das Verfahren versucht das oben genannte Problem der Auflösungssteigerung in uninteressanten Bereichen zu adressieren. Das Ziel ist also, die Nodes im Beispielraum so entsprechend der dortigen Dichteverteilung zu positionieren, dass alle Nodes dieselbe Wahrscheinlichkeit haben, einen aus der Verteilung gezogenen Vektor zugewiesen zu bekommen. Würde man also aus der Verteilung eine hohe Anzahl neuer Vektoren ziehen, und sie der nächsten Node zuweisen, hätten danach alle Nodes ungefähr gleich viele Vektoren zugewiesen bekommen. Formal gesprochen sollen die Nodes also gleich viel Wahrscheinlichkeitsmasse in ihren Voronoi Regionen auf sich vereinen.

Definition 4.2.5 (Voronoi Region)
Die Voronoi Region $\mathcal{V}$ einer Node N_i bestimmt sich über die Distanzen im Beispielraum und stellt den Bereich des Beispielraums dar, aus dem Vektoren dieser Node zugewiesen werden.

$$\mathcal{V}(\mathrm{N}_i) = \{\mathbf{x} | d_e(\mathbf{x}, \mathbf{w}_i) \leq d_e(\mathbf{x}, \mathbf{w}_k) \; \forall \; \mathrm{N}_k \in \mathcal{N}\} \tag{4.11}$$

Damit ergibt sich mit der Dichtefunktion $p(\mathbf{x})$ die Bedingung für die Gleichverteilung der Wahrscheinlichkeitsmasse über den Nodes zu:

$$\int_{\mathbf{x} \in \mathcal{V}(\mathrm{N}_i)} p(\mathbf{x}) d\mathbf{x} = \frac{1}{|\mathcal{N}|} \quad \forall \; \mathrm{N}_i \in \mathcal{N} \tag{4.12}$$

Die Erfüllung dieser Bedinung soll wieder durch ein iteratives Verfahren approximiert werden. Geht man dabei vom Algorithmus der Growing Grid SOM aus, ergeben sich nur wenige Änderungen, die wieder in der üblichen Einteilung vorgestellt werden:

Initialisierungsphase

Die Gittergröße wird nicht mehr starr initialisiert, sondern kann tatsächlich wieder durch den Benutzer eingestellt werden, um eine Grundauflösung sicherzustellen. Es werden auch wieder Zähler a_i initialisiert, die jedoch nun nicht mehr durch eine simple Matrix dargestellt werden können, da die feste Gitterstruktur später aufgelöst wird.

Adaptionsphase

Wie bei der Growing Grid SOM, wird nach Rundenende geprüft, ob ein Abbruchkriterium erfüllt ist. Falls nicht, wird auch hier nach der Node mit höchstem a_i-Wert gesucht, statt einer neuen Reihe wird hier aber neben ihr nur eine neue Node eingefügt. Die Position sowohl im Beispielraum, als auch im Kartenraum bestimmt sich dabei wiederum analog zur Growing Grid SOM über den entferntesten Nachbarn von N_i, indem die Mittelwerte der Beispielraum- und Kartenraumkoordinaten gebildet werden.
Anschließend werden alle Zähler zurückgesetzt. [2]

[2] Dies weicht vom Originalpapier ab, in dem vorgeschlagen wird, den Wert der Zählvariablen basierend auf der Größe der Voronoi Regionen der Nodes im Beispielraum zwischen den alten und der neuen Node zu verteilen, statt alle Zähler zurückzusetzen. Diese Arbeit verfolgt stattdessen den Ansatz der Growing Grid SOM weiter, nach der Runde die Zähler komplett

Abschlussphase

Falls das Abbruchkriterium erfüllt ist, kann eine Nachbearbeitung durchgeführt werden, in der überflüssige Nodes wieder entfernt werden. Als überflüssig können Nodes bezeichnet werden, deren empirische Dichte unter einem vom Benutzer vorgegebenen Schwellwert liegt.

Erweiterungen

Obwohl sich dieses Verfahren von einer festen Gitterstruktur trennt, bleiben auch bei allen später eingefügten Nodes die einmal gewählten Positionen im Kartenraum konstant. Da sich mit den veränderten Koordinaten im Beispielraum aber auch die Dichte verändern kann, wird ein Nachbearbeitungsschritt nötig, der an ihrer Position im Kartenraum überflüssig gewordene Nodes wieder entfernt. Das im nächsten Abschnitt vorgestellte Verfahren wird solch eine Nachbearbeitung unnötig machen.

4.2.5. Distance Self-Organizing Map

In (Meyer 1998b) und (Meyer 1998a) stellt Meyer die Inverted Self-Organizing Map, kurz ISOM, als ein neuen Ansatz zum Graphlayout vor. Der folgende Absatz beschreibt ihn kurz, da die in dieser Arbeit vorgeschlagene Methode der Distance Self-Organizing Map, oder DSOM, auf ihm aufbaut.

Die ISOM

Wie der Name schon sagt, ist Graphlayout das Problem, bei dem ein Graph $G = (\mathcal{V}, \mathcal{E})$ mit der Menge der Knoten $\mathcal{V}$ und Kanten $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ zwischen diesen Knoten auf einer zweidimensionalen Fläche angeordnet werden soll, so dass die euklidischen Abstände in dieser Fläche den graphentheoretischen Distanzen entsprechen. Dazu schlägt Meyer folgende Herangehensweise vor:

zurückzusetzen und in der nächsten Runde die empirische Dichte pro Node neu aus den Beispieldaten zu bestimmen. Dadurch werden komplexe Rechnungen, oder ungenaue Approximationen wie in (Fritzke 1992) als Abhilfe beschrieben, vermieden. Weiterhin verändern die Nodes im Laufe der Adaption ihre Position im Beispielraum und damit die auf sich vereinte Dichtemasse, was eine Zurücksetzung der Counter ebenfalls sinnvoll erscheinen lässt. Andernfalls könnten sie aufgrund ihrer initialen Position dauerhaft bei der Bestimmung der Node mit dem maximalen a_i Wert bevorzugt oder benachteiligt werden.

Es wird für jeden Knoten im Graphen eine Node einer SOM erzeugt. Dabei kann deren Beispielraumkoordinate verworfen werden, da es bei diesem Problem kein Beispielraum gibt. Die Kartenraumkoordinate $\mathbf{c}_i$ wird zufällig initialisiert, wobei die einzelnen Koordinaten zwischen 0 und 1 liegen. Es ergibt sich also die Menge der Nodes

$$\mathcal{N} = \{\mathrm{N}_i | \mathrm{N}_i = (null, \mathbf{c}_i), i = 1, \ldots, |\mathcal{V}|\} \tag{4.13}$$

Nun sollen die Kartenraumkoordinaten so angepasst werden, dass die euklidischen Distanzen den Graphen widerspiegeln. Dies kann erstaunlich einfach dadurch erfolgen, dass aus einer Gleichverteilung über dem Kartenraum Vektoren $\mathbf{c}$ gezogen und die Kartenraumkoordinaten der Nodes an sie angepasst werden. Dazu wird diejenige Node bestimmt, die im Kartenraum am nächsten am gezogenen $\mathbf{c}$ liegt und diese und ihre im Kartenraum nicht zuweit entfernt liegenden, graphentheoretischen Nachbarn werden anschließend ein Stück in die Richtung des $\mathbf{c}$ geschoben. Die Stärke der Anpassung hängt dabei von der Entfernung im Kartenraum zwischen Node und $\mathbf{c}$ ab.
Dieses Verfahren liefert auch die Begründung für die Namenswahl, da im Unterschied zu anderen selbstorganisierenden Karten die Kartenraum- und nicht die Beispielraumkoordinaten der Nodes angepasst werden.

Von der ISOM zur DSOM

Der folgende Abschnitt stellt nun die Modifikationen des oben kurz beschriebenen Algorithmus vor, aus denen sich dann die DSOM ergibt. Die folgenden Abschnitte liefern dann eine detailierte Beschreibung, die auf der üblichen Phaseneinteilung beruht.

Offensichtlich eignet sich der ISOM Algorithmus nicht zur Dimensionsreduktion, da keine Beispielraumkoordinaten verfügbar sind. Dem kann abgeholfen werden, indem man die Menge aller Beispiele $\mathcal{E} = \{\mathbf{x}_1, \ldots \mathbf{x}_N\}$ als Knoten des Graphens betrachtet. Dann lässt sich für jedes Beispiel eine Node erzeugen, die das Beispiel selbst als Beispielraumkoordinate besitzt. Es ergibt sich also als Menge von Nodes:

$$\mathcal{N} = \{\mathrm{N}_i | \mathrm{N}_i = (\mathbf{x}_i, \mathbf{c}_i), i = 1, \ldots, N\} \tag{4.14}$$

Dabei wird die Kartenraumkoordinate wie oben zufällig gewählt.
Will man aber nun die Kartenraumposition auch noch günstig für den Betrachter anpassen, steht man vor dem Problem, dass der oben beschriebene Algorithmus zur Anpassung zwingend einen Graphen benötigt, da nur Nachbarn im Graphen

mit angepasst werden. Da bei einer Menge von Beispielen keine bekannte Struktur vorliegt, die man dazu ausnutzen könnte, man aber einen Graphen benötigt, liegt es nahe, den vollständigen Graphen zu verwenden: Dann liegen Nodes vor, die mit allen anderen Nodes verbunden sind und über Beispielraumkoordinaten verfügen. Damit ist man aber schon mit dem nächsten Problem konfrontiert: Im ursprünglichen Algorithmus wurde die Selektion der anzupassenden Nodes über die Nachbarschaft im Graphen durchgeführt. Jetzt sind aber alle Knoten miteinander verbunden, so dass der ursprüngliche Mechanismus nicht mehr funktioniert. Als Lösung wird in dieser Arbeit ein Ansatz vorgeschlagen, der die Distanzen im Ursprungsraum beachtet und im Ursprungsraum nahe Beispiele zum Anpassen auswählt. Da dann aber die Graphstruktur nicht mehr benötigt wird, kann sie komplett wegfallen. Stattdessen wird eine Distanzmatrix zwischen allen Beispielen benötigt, woher sich auch der Name Distance Self-Organizing Map ableitet.

Eigenschaften der DSOM

Dieses neue Verfahren bietet damit einige Vorteile. Während bei der Growing Cell Self-Organizing Map explizit und aufwendig versucht wurde, die Nodes so anzupassen, zu erzeugen und zu löschen, dass alle gleich viel Wahrscheinlichkeitsmasse auf sich vereinen, bekommt man diese Eigenschaft nun durch die Konstruktionsvorschrift für die Menge der Nodes geschenkt:

Bei der Realisierung von Beispielen richten sich die Werte und damit die Position im Beispielraum nach der Wahrscheinlichkeitsverteilung. In Bereichen hoher Dichte realisieren sich mehr Beispiele als in Bereichen niedriger. Bei genügend vielen Beispielen enthalten die Voronoiregionen der Beispiele daher ungefähr gleichviel Wahrscheinlichkeitsmasse. Da die Nodes durch die Konstruktion exakt den Beispielen entsprechen, gilt dies auch für sie. Ohne weitere Annahmen treffen zu wollen, ist die Gleichverteilung der Wahrscheinlichkeitsmasse über die Voronoiregionen auch exakt die beste Annäherung an die tatsächliche Verteilung.

Im Folgenden wird nun der komplette Algorithmus detailiert in der üblichen Einteilung beschrieben.

Initialisierungsphase

Während der Initialisierungsphase wird zunächst die Menge der Nodes $\mathcal{N} = \{\mathrm{N}_i | \mathrm{N}_i = (\mathbf{x}_i, \mathbf{c}_i), i = 1, \ldots, N\}$ erzeugt. Die Kartenraumkoordinaten $\mathbf{c}_i$ werden dabei zufällig zwischen 0 und 1 erzeugt. Gleichzeitig wird eine $N \times N$ Distanzmatrix $\mathbf{D}$

erzeugt:

$$\mathbf{D}_{i,j} = d(\mathbf{x}_i, \mathbf{x}_j) \tag{4.15}$$

Dabei kann d eine beliebige Distanzfunktion sein.

Adaptionsphase

Anders als bei bisherigen Verfahren muss die Adaptionsphase nicht in Runden eingeteilt werden. Stattdessen werden nacheinander Koordinaten $\mathbf{c}$ aus einer Gleichverteilung über dem gesamten Kartenraum gezogen, bis eine bestimmte Anzahl gezogen wurde oder ein beliebiges Abbruchkriterium erfüllt ist. Dann wird diejenige Node N_a bestimmt, die im Kartenraum am nächsten an $\mathbf{c}$ liegt:

$$\mathrm{N}_a = \mathrm{N}_{\arg\min_i \; d_d(\mathbf{c},\mathbf{c}_i)} \tag{4.16}$$

Dabei handelt es sich bei der DSOM Distanzfunktion ebenfalls um den euklidischen Abstand, diesmal jedoch im Kartenraum:

Definition 4.2.6 (DSOM Distanzfunktion)
Die DSOM Distanzfunktion $d_d : \mathbb{K}^2 \times \mathbb{K}^2 \to \mathbb{R}$ *gibt die Distanz zwischen zwei Elementen des Kartenraums an. Sie ist definiert durch:*

$$d_d(\mathbf{c}_i, \mathbf{c}_j) = \sqrt{\mathbf{c}_i^T \mathbf{c}_j} \tag{4.17}$$

Anschließend werden alle Nodes im Kartenraum in Richtung der Koordinate $\mathbf{c}$ geschoben, indem sie mit folgender Formel abhängig von der Beispielraumentfernung zu N_a an diesen Impuls angepasst werden:

Definition 4.2.7 (DSOM Adaptionsregel)
Die DSOM Adaptionsregel weist einer anzupassenden Node $\mathrm{N}_i = (\mathbf{w}_i, \mathbf{c}_i)$ *eine neue Kartenraumkoordinate* $\mathbf{c}_i'$ *zu:*

$$\mathbf{c}_i' = \mathbf{c}_i + \alpha(t)(\mathbf{c} - \mathbf{c}_i) \tag{4.18}$$

mit

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_{t_{max}}}{\alpha_0} \right)^{\frac{t}{t_{max}}} e^{-\frac{\mathbf{D}_{a,i}}{2\sigma(t)}} \tag{4.19}$$

und

$$\sigma t = \sigma_0 \left(\frac{\sigma_{t_{max}}}{\sigma_0} \right)^{\frac{t}{t_{max}}} \tag{4.20}$$

α_0 *ist dabei die Stärke in der ersten Runde,* $\alpha_{t_{max}}$ *entsprechend die Stärke in der letzten Runde. Gleiches gilt für die* σ*.*

Im Unterschied zur ansonsten ähnlichen ESOM Adaptionregel aus 4.2.4 bestimmt sich hier die Stärke der Adaption nicht aus der Enfernung im Kartenraum, sondern aus einem beliebigen Distanzmaß d und dessen Matrix $\mathbf{D}$ zwischen den Beispielraumvektoren der Node N_a und der anzupassenden Node N_i . Dadurch werden ähnliche Beispiele stärker in die Richtung des Impulses gezogen als unähnliche, wodurch sich eine Ordnung ergibt.

Ein kleines Beispiel eines solchen Anpassungsprozesses zeigt Abbildung 4.4. Es werden drei Schritte gezeigt, der erste links, der letzte rechts. Die jeweils zufällig gezogene Koordinate $\mathbf{c}$ wird durch das rote Kreuz markiert. Die drei Kreise stellen Nodes dar. Obwohl keine Graphstruktur verwendet wird, ist die Distanz zwischen den Nodes im Beispielraum hier als vollständiger Graph mit Kantengewichten visualisiert um die Übersichtlichkeit zu erhöhen. Die Kantengewichte repräsentieren die Entfernung zwischen den Nodes im Beispielraum. Die rot dargestellte Node ist jeweils die nächste Node N_a, die schwarzen die anderen. Die grauen Nodes zeigen jeweils die Situation, nachdem an das $\mathbf{c}$ angepasst wurde. Entsprechend nehmen die rot/schwarzen im nächsten Bild dann die gleichen Positionen ein, da es den nächsten Anpassungsschritt zeigt.
Man sieht bereits in diesen nur drei Schritten wie sich die Nodes anhand ihrer Distanz anordnen. Während im ersten Bild noch die 7er Kante die kürzeste ist, beginnt sich dieses schon im zweiten Bild zu verändern und ist im dritten Bild bereits die Längste. Dass im ersten Bild keine sinnvolle Adaption stattfindet, hängt mit der Position des Impulses zusammen, da diejenige Node ausgewählt wurde, die zu beiden anderen Nodes Kanten gleichen Gewichts besitzt. Dadurch verschieben sich beide Nodes relativ zu ihrer Entfernung gleich stark, was die Topologie aber nicht beeinträchtigt. Die Hoffnung ist daher, dass dieses Verfahren auch bei mehr Beispielen in einem Gleichgewicht konvergiert, das möglichst häufig die Entfernungsrelation des Beispielraums auch im Kartenraum einhält.

Abschlussphase

Empirische Untersuchungen der DSOM zeigen, dass nach der Positionierung der Nodes immer am Rand der Karte ein freier Bereich bleibt. Daher kann es sich lohnen, die Positionen so zu skalieren, dass der komplette Kartenraum genutzt wird. Dieses Verhalten kann beispielhaft in Abbildung 4.4 betrachtet werden. Nach drei Schritten konzentrieren sich die Nodes bereits stärker im Zentrum. Dies hängt damit zusammen, dass jeder Impuls, egal an welcher Position er liegt, für die Mehrzahl der Nodes in Richtung Zentrum wirkt. Selbst wenn der Impuls am äußersten Rand der Karte liegen würde, würde er auf alle Nodes, die von ihm aus jenseits der

Mitte liegen, in Richtung des Zentrums wirken. Da aber dabei auch einige nach außen gezogen werden, konvergiert das Verfahren in einem Gleichgewicht, bei dem ein äußerer Rand frei bleibt.

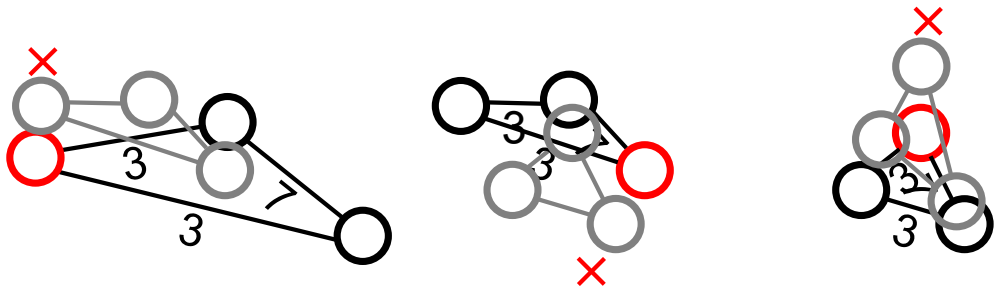


Abbildung 4.4.: Adaption einer DSOM

4.2.6. Kernel Distance Self-Organizing Map

Im letzten Abschnitt wurde eine SOM entwickelt, die auf einem Abstandsmaß im Beispielraum basiert. Dieser Abschnitt soll zeigen, dass sich eine so formulierte SOM mit Hilfe einer Kernfunktion ausdrücken lässt, wenn das euklidische Abstandsmaß gewählt wird. Dadurch wird die Benutzung von Kernfunktionen möglich, um den Abstand in einem transformierten Beispielraum $\Phi(\mathbb{D}^n)$ zu berechnen. Dazu wird nach (Schölkopf 2001) zunächst der euklidische Abstand passend umgeformt:

$$
\begin{aligned}
d(\mathbf{x},\mathbf{y}) &= \sqrt{\sum_i (x_i - y_i)^2} && (4.21)\\
&= \sqrt{\sum_i x_i^2 - 2x_i y_i + y_i^2} && (4.22)\\
&= \sqrt{\sum_i x_i^2 - 2\sum_i x_i y_i + \sum_i y_i^2} && (4.23)\\
&= \sqrt{\mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{y} + \mathbf{y}^T\mathbf{y}} && (4.24)
\end{aligned}
$$

Sei Φ eine beliebige Transformation des Beispielraumes. Dann würde sich der euklidische Abstand in diesem transformierten Raum ergeben durch folgenden Aus-

druck:

$$d(\mathbf{x},\mathbf{y}) \quad = \quad \sqrt{\Phi(\mathbf{x})^T\Phi(\mathbf{x}) - 2\Phi(\mathbf{x})^T\Phi(\mathbf{y}) + \Phi(\mathbf{y})^T\Phi(\mathbf{y})} \tag{4.25}$$

Zum einfachen Verständnis sei hier nochmal die Definition einer Kernfunktion aus 2.4.3 wiedergegeben:

Definition 4.2.8 (Kernfunktion)
Eine Funktion $K : X \times X \to \mathbb{R}$ heißt Kernfunktion, oder kurz Kernel, falls gilt: Es existiert ein Vektorraum F *mit Skalarprodukt und eine Funktion $\Phi : X \to \mathrm{F}$ so dass gilt:*

$$K(\mathbf{x},\mathbf{x}') = \Phi(\mathbf{x})^T\Phi(\mathbf{x}') \quad \forall \mathbf{x},\mathbf{x}' \in X \tag{4.26}$$

Man sieht leicht, dass sich der Ausdruck 4.25 für die Distanz im transformierten Raum durch folgende Formulierung mit Kernfunktionen ersetzen lässt:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{K(\mathbf{x},\mathbf{x}) - 2K(\mathbf{x},\mathbf{y}) + K(\mathbf{y},\mathbf{y})} \tag{4.27}$$

Damit kann die Distance Self-Organizing Map über die Entfernungen im transformierten Beispielraum angepasst werden. Die Intention ist, Einblicke in die Arbeitsweise von Entscheidungsmodellen wie der SVM zu liefern, die ebenfalls in diesen implizit transformierten Beispielräumen arbeiten. Zusätzlich ermöglicht die Formulierung über die Distanz den Einsatz einer größeren Klasse von Kernfunktionen, wie in (Schölkopf 2001) gezeigt wird. Diese Arbeit beschränkt sich aber auf die Verwendung der klassischen Kernfunktionen, wie sie auch in den Entscheidungsmodellen benutzt werden, da auf ihrer Visualisierung der Hauptaugenmerk dieser Arbeit liegen soll.

Algorithmisch unterscheidet sich die Kernel Distance Self-Organizing Map nicht von der ursprünglichen Variante. Da deren Formulierung komplett auf den Distanzen basiert, lässt sich die KDSOM durch die Verwendung einer geeigneten Distanzfunktion, die die Gleichung 4.27 umsetzt, ohne Änderung am Algorithmus realisieren.

5. Alternative Verfahren zur Dimensionsreduktion

Wie in Abschnitt 3.2 angekündigt, widmet sich dieses Kapitel der Vorstellung zweier Alternativen zu Self-Organizing Maps. Dies ist zum einen die Hauptkomponentenanalyse, meist mit der englischen Abkürzung PCA bezeichnet, als Vertreter der linearen Dimensionsreduktionen. Ihre Funktionsweise soll hier kurz wiedergegeben werden, um später die Ergebnisse besser mit nicht linearen Verfahren vergleichen zu können.
Dies gilt auch für die sich anschließende Beschreibung des nicht-linearen Dimensionsreduktionsverfahrens Locally Linear Embedding. Dabei handelt es sich um ein deterministisches Verfahren, was den Vergleich mit den nicht-deterministisch arbeitenden Self-Organizing Maps lohnenswert macht.

5.1. Hauptkomponentenanalyse

Bei der Hauptkomponentenanalyse handelt es sich um eine lineare Transformation. Entsprechend lässt sie sich durch Matrixmultiplikation mit der Transformationsmatrix $\mathbf{T}$ durchführen, siehe dazu Defintion 3.2.1. Anschaulich betrachtet, stellen die Spaltenvektoren der Transformationsmatrix die Basisvektoren eines neuen Koordinatensystems dar, in das die Daten überführt werden. Diese Spaltenvektoren bezeichnet man als Hauptkomponenten. Die Besonderheit der Hauptkomponentenanalyse ist, dass die Hauptkomponenten zwei Bedingungen erfüllen:
Zum einen stehen sie senkrecht aufeinander, zum anderen sind sie immer in Richtung der größten Varianz in den Daten ausgerichtet. Die erste Hauptkomponente beschreibt also die Richtung der größten Varianz in den Daten, während die nächste die dazu orthogonale Richtung mit der größten Varianz angibt. Dies wird in Abbildung 5.1 gezeigt. Im linken Bild sind die ursprünglichen Koordinatenachsen schwarz eingezeichnet, während in rot die Hauptkomponenten und damit die neuen Koordinatenachsen darüber gelegt wurden. Im rechten Bild sieht man das Ergebnis dieser Transformation, bei dem die Punkte in das neue Koordinatensystem übertragen wurden.

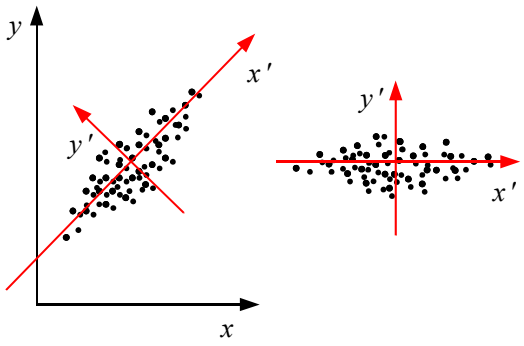


Abbildung 5.1.: Transformation des Koordinatensystems

Da alle Vektoren orthogonal zueinander sind, verlieren die einzelnen Dimensionen der transformierten Beispiele jegliche Korrelation und für die Daten ergibt sich eine Kovarianzmatrix, die einer Diagonalmatrix entspricht.

Berechnung der Hauptkomponenten

Um nun die Transformationsmatrix $\mathbf{T}$ zu bestimmen, müssen die Beispieldaten zunächst mittelwertbereinigt werden, das heißt die Mittelwerte jeder Dimension sind 0. Danach lässt sich die Kovarianzmatrix $\mathbf{C}$ bestimmen:

$$\mathbf{C} = \frac{1}{N-1}(\mathbf{x}_1, \ldots, \mathbf{x}_n)(\mathbf{x}_1, \ldots, \mathbf{x}_n)^T \tag{5.1}$$

Anschließend bestimmt man die Hauptkomponenten, indem man die Eigenvektoren ϕ_i der Kovarianzmatrix über die Lösung des folgenden Eigenwertproblems berechnet:

$$\mathbf{C}\phi_i = \lambda_i\phi_i \quad mit \quad \lambda_0 \geq \lambda_1 \ldots \geq \lambda_m \tag{5.2}$$

λ_i ist dabei der i-te Eigenwert. Dies lässt sich in einer einzigen Matrixmultiplikation zusammenfassen:

$$\mathbf{C}\Phi = \Lambda\Phi \tag{5.3}$$

Die Matrizen Φ und Λ sind definiert durch $(\phi_1, \ldots \phi_m)$ respektive eine Diagonalmatrix, deren diagonale Einträge den λ_i entsprechen. Die Transformationsmatrix

T ergibt sich dann durch die Kombination der p Eigenvektoren mit den höchsten Eigenwerten:

$$\mathbf{T} = \begin{pmatrix} \phi_1^T \\ \vdots \\ \phi_p^T \end{pmatrix} \tag{5.4}$$

Durch die Matrixmultiplikation erfolgt eine Dimensionsreduktion, wenn $p < m$ gewählt wird, da der resultierende Vektor vom Zeilenrang der Matrix ist. Dabei kann eine Fehlerabschätzung über die Eigenwerte erfolgen, da diese angeben, wieviel Varianz verloren geht, wenn eine Dimension nicht übernommen wird:

$$\varepsilon = \sum_{i=p+1}^{m} \lambda_i \tag{5.5}$$

Da die Eigenwerte der Größe nach absteigend sortiert sind, erklärt diese Fehlerabschätzung, warum die ersten p Eigenvektoren übernommen werden, denn dies minimiert genau diesen Fehler ε bei festem p.

5.2. Locally Linear Embedding

Beim Locally Linear Embedding Algorithmus, oder kurz LLE, handelt es sich wie bei den Self-Organizing Maps um eine nicht lineare Dimensionsreduktion. Das Verfahren wurde in (Roweis und Saul 2000) veröffentlicht. Eine genauere Beschreibung der folgenden Berechnungen findet sich in (Saul und Roweis 2000). LLE ist wie Self-Organizing Maps in der Lage Mannigfaltigkeiten in den Daten zu entdecken, die von niedrigerer Dimension sind, als der sie enthaltende Beispielraum. Dabei verwendet LLE aber einen Verfahren, durch das das Ergebnis geschlossen berechnet werden kann und sich entsprechend deterministisch verhält. Dazu nutzt es die lokal annähernd linearen Beziehungen zwischen den Daten aus, indem jeder Punkt durch seine Nachbarn dargestellt wird. Daher muss zunächst die Annahme getroffen werden, dass ein Datenpunkt $\mathbf{x}$ zusammen mit seinen k nächsten Nachbarn $\mathbf{n}(\mathbf{x})_i$ mit $i = 1, \ldots k$ auf einem linearen Teilstück der Mannigfaltigkeit liegt. Dann lässt sich der Datenpunkt durch eine Linearkombination seiner Nachbarn darstellen:

$$\mathbf{x} = \sum_{i=1}^{k} w_i \mathbf{n}_i(\mathbf{x}) \tag{5.6}$$

Da die Annahme fast nie exakt zutreffen wird, ergibt sich in der Praxis ein Fehler $\varepsilon(\mathbf{W})$, der von den gewählten Gewichten der Linearkombination abhängt:

$$\varepsilon(\mathbf{W}) = \sum_{i}^{N} \left| \mathbf{x}_i - \sum_{j=1}^{k} W_{i,j} \mathbf{n}_j(\mathbf{x}) \right|^2 \tag{5.7}$$

Dieser Fehler ergibt sich entsprechend aus der Summe der quadrierten Beträge der Differenzvektoren zwischen tatsächlichem Vektor und dem rekonstruierten Vektor. Die Gewichtsmatrix $\mathbf{W}$ gibt dabei den Einfluss des j-nächsten Nachbarn auf die Rekonstruktion des i-ten Vektor an, fasst also alle Gewichte in einer Matrix zusammen.
Man sieht, dass die Berechnung der Gewichte durch Minimierung des Fehlers $\varepsilon(\mathbf{W})$ aus 5.7 dafür sorgt, dass die Gewichte invariant gegenüber Rotation und Skalierung sind. Denn der verwendete euklidische Abstand als Fehlermaß ist invariant unter Rotation und eine gemeinsame Skalierung kann vor die Summe gezogen werden und entfällt so bei der Minimierung. Erzwingt man jetzt zusätzlich die Bedingung $\sum_{j=1}^{k} W_{i,j} = 1$ für alle i, dann sind die Gewichte auch gegenüber Translation invariant, da sie nun geometrische Eigenschaften der Nachbarschaft abbilden.
Dies kann man sich zunutze machen, da sich jedes lineare Teilstück einer Mannigfaltigkeit durch eine geeignete Kombination von Rotations-, Skalierungs- oder Translationsoperationen auf einen niedriger dimensionalen Raum abbilden lässt. Da sich die Gewichte genau unter diesen Operationen als invariant herausgestellt haben, lassen sie sich benutzen um diese Abbildung vorzunehmen. Anschaulich ergeben sich die Punkte in dem niedriger dimensionalen Raum wieder aus der Linearkombination ihrer nächsten Nachbarn. Gesucht ist also die Abbildung $\mathbf{y}(\mathbf{x})$, bei der für alle $\mathbf{x}$ gelten soll:

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^{k} w_i \mathbf{y}(\mathbf{n}_i(\mathbf{x})) \tag{5.8}$$

Da diese Bedingungen durch die Projektion in die niedrigere Dimension nicht einhaltbar sind, muss auch hier wieder ein Fehler zugelassen werden:

$$\Delta(\mathbf{y}) = \sum_{i}^{N} \left| \mathbf{y}(\mathbf{x}_i) - \sum_{j=1}^{k} W_{i,j} \mathbf{y}(\mathbf{n}_j(\mathbf{x})) \right|^2 \tag{5.9}$$

Die Minimierung dieses Problems ergibt die global optimale Lösung für die Abbildung in Bezug auf die Erhaltung der lokalen Nachbarschaftsbeziehungen, wie sie in den Gewichten Ausdruck finden. Der nächste Abschnitt beschreibt die notwendigen Optimierungsschritte.

Berechnung der Gewichte und Positionen

Um die optimalen Positionen bestimmen zu können, müssen zunächst die Gewichte bestimmt werden. Diese ergeben sich aus der Minimierung der Formel 5.7, die geschlossen erfolgen kann. Da es sich bei der Formel 5.7 um eine Summe handelt, kann jeder Summand einzeln minimiert werden, weshalb sich für ein $\mathbf{x}$ mit entsprechendem Gewichtsvektor $\mathbf{w}$ folgender Fehler ergibt:

$$\begin{aligned} \varepsilon &= \left| \mathbf{x} - \sum_{j=1}^{k} w_j \mathbf{n}_j(\mathbf{x}) \right|^2 && (5.10) \\ &= \left| \sum_{j=1}^{k} w_j (\mathbf{x} - \mathbf{n}_j(\mathbf{x})) \right|^2 && (5.11) \\ &= \sum_{j=1}^{k} \sum_{i=1}^{k} w_j w_i \mathbf{C}_{j,i} && (5.12) \end{aligned}$$

Wobei $\mathbf{C}_{j,i}$ die lokale Kovarianzmatrix bezeichnet, die definiert ist durch:

$$\mathbf{C}_{j,i} = (\mathbf{x} - \mathbf{n}_j(\mathbf{x}))(\mathbf{x} - \mathbf{n}_i(\mathbf{x}))^T \tag{5.13}$$

Mittels eines Lagrange Multiplikators für die Nebenbedingung, die die Gewichtssumme auf 1 bringt, ergibt sich dann das optimale Gewicht für den j-nächsten Nachbarn zu:

$$w_j = \frac{\sum_{i=1}^{k} \mathbf{C}_{j,i}^{-1}}{\sum_{i=1}^{k} \sum_{o=1}^{k} \mathbf{C}_{i,o}^{-1}} \tag{5.14}$$

Nachdem die Gewichte bestimmt wurden, lassen sich über sie die Positionen bestimmen. Dazu minimiert man den Fehler 5.9. Dieser lässt sich durch Ausmultiplizieren des Quadrates in folgende Form überführen:

$$\Delta(\mathbf{y}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \delta_{i,j} - W_{i,j} - W_{j,i} + \sum_{o=1}^{k} W_{o,i} W_{o,j} \mathbf{y}(\mathbf{x}_i)^T \mathbf{y}(\mathbf{x}_j) \tag{5.15}$$

Mit $\delta_{i,j}$ als Einträge einer $N \times N$ Identitätsmatrix lässt sich das zusammenfassen mit Hilfe einer Matrix $\mathbf{M}$:

$$\Delta(\mathbf{y}) = \sum_{i=1}^{N} \sum_{j=1}^{N} M_{i,j} \mathbf{y}(\mathbf{x}_i)^T \mathbf{y}(\mathbf{x}_j) \tag{5.16}$$

Dabei definiert sich die Matrix $\mathbf{M}$ über ihre Komponenten:

$$M_{i,j} = \delta_{i,j} - W_{i,j} - W_{j,i} + \sum_{o=1}^{k} W_{o,i} W_{o,j} \tag{5.17}$$

Da die Abbildung $\mathbf{y}$ wie gezeigt invariant unter Translation und Skalierung ist, müssen diese beiden Fälle ausgeschlossen werden um degenerierte Lösungen auszuschließen. Daher setzt man den Mittelwert aller Positionen auf den Nullvektor und ihre Kovarianz auf eine Einheitsmatrix. Die Lösung ergibt sich dann aus den $p+1$ Eigenvektoren mit den kleinsten Eigenwerten der Matrix $\mathbf{M}$. Der letzte Eigenvektor wird dabei verworfen, da er nur eine Translation beinhaltet, die restlichen p Eigenvektoren formen die neuen Positionen der abgebildeten Daten.

6. Interpolationsverfahren

Dieses Kapitel beschreibt detailiert die verwendeten Interpolationsverfahren, die die fehlenden Werte im Kartenraum berechnen sollen, nachdem mit einer Dimensionsreduktion die bekannten Wertepaare gebildet wurden. Die Beschreibung folgt der Reihenfolge aus Abschnitt 3.3, der bereits eine grobe Übersicht gegeben hat und die Verfahren in den Zusammenhang einordnete. Wo während der Implementierung Besonderheiten auftauchen, wird darauf hingewiesen.

6.1. Shepardinterpolation

Die Shepardinterpolation (Shepard 1968) gehört zur Gruppe der gewichtsbasierten Interpolationsverfahren. Entsprechend berechnet sich der Wert an einer Position $\mathbf{x}$ durch ein gewichtetes Mittel aller bekannten Wertepaare $(\mathbf{x}_i, y_i = f(\mathbf{x}_i)), i = 1, \ldots, n$ mit $\mathbf{x}_i \in \mathbb{R}^m$:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} w_i(\mathbf{x}) y_i \tag{6.1}$$

Als Gewichtsfunktion $w_i : \mathbb{R}^m \to \mathbb{R}$ verwendet sie dabei eine Funktion auf Grundlage des inversen euklidischen Abstandes:

$$w_i(\mathbf{x}) = \frac{[(\mathbf{x} - \mathbf{x}_i)^T (\mathbf{x} - \mathbf{x}_i)]^{-\mu}}{\sum_{j=1}^{N} [(\mathbf{x} - \mathbf{x}_j)^T (\mathbf{x} - \mathbf{x}_j)]^{-\mu}} \tag{6.2}$$

Bekannte Werte $\mathbf{x}_i$, die weiter vom Probenpunkt entfernt liegen, gehen dadurch also weniger stark in das Resultat ein, als näher gelegene. Dabei bleibt die relative Position der bekannten Werte aber unberücksichtigt, so dass Punkte, auf deren Verbindungsgeraden zu $\mathbf{x}$ sich andere Punkte befinden, genauso stark eingehen wie Punkte, die gleichweit entfernt sind, aber nicht verdeckt werden.
Man sieht, dass Gleichung 6.1 die Interpolationseigenschaft erfüllt, wenn man den Probenpunkt $\mathbf{x}$ gegen einen bekannten Datenpunkt $\mathbf{x}_i$ gehen lässt, denn dann konvergiert der Nenner von w_i und der entsprechende Summand im Zähler aller Gewichte gegen unendlich, so dass sich w_i im Grenzwert zu 1 ergibt und alle w_j

mit $j \neq i$ zu 0. Entsprechend ergibt $\hat{f}(\mathbf{x}_i)$ sich zu:

$$\hat{f}(\mathbf{x}_i) = 1 \cdot y_i + \sum_{j \neq i} 0 \cdot y_j = y_i \tag{6.3}$$

Da die beschränkte Genauigkeit des Computers die Grenzwertbildung unmöglich macht, muss dieser Fall in der Praxis speziell abgefangen werden.
Je nach Wahl von μ ergibt sich eine andere Form der interpolierten Oberfläche:

$0 \leq \mu \leq 0,5$**:** An den bekannten Stellen bilden sich Spitzen

$0,5 \leq \mu \leq 1$**:** An den bekannten Stellen bilden sich Ecken

$1 \leq \mu$**:** An den bekannten Stellen bildet sich eine horizontale Tangentialebene, die mit steigendem μ größer wird. Bei engen Abständen zwischen zwei Punkten kommt es so zu sehr hohen Gradienten, da nur noch wenig Platz für die Überwindung von Höhenunterschieden bleibt.

Gradientenberechnung

Ein Vorteil der Shepardmethode ist, dass sich die Gradientenvektoren an den interpolierten Stellen geschlossen berechnen lassen. Dies wird später zur Berechnung des Distanzlayers aus Abschnitt 7.4 notwendig sein. Die partielle Ableitung nach der ersten Komponente ergibt:

$$\begin{aligned}\frac{\partial f(\mathbf{x})}{\partial x_1} = & \frac{2\mu}{(\sum_{i=1}^{N}(x_1 - x_{i1})^2 + (x_2 - x_{i2})^2)^{-\mu})^2} \cdot \\ & \sum_{i=1}^{N}\sum_{j=1}^{N}((x_1 - x_{i1})^2 + (x_2 - x_{i2})^2)^{-\mu} \cdot \\ & ((x_1 - x_{j1})^2 + (x_2 - x_{j2})^2)^{-\mu-1}(x_1 - x_{j1})(y_j - y_i)\end{aligned}$$

Die Ableitung nach der zweiten Komponente erfolgt analog.

Implementierung

Wie in Gleichung 6.1 erkennbar, kann die Shepardinterpolation leicht auf Vektoren verallgemeinert werden. Dadurch entfällt die ständige Wiederholung des aufwendigsten Schrittes, der Berechnung der Gewichte, für jede Dimension. Weiterhin muss die Summe im Nenner von 6.2 nicht gesondert berechnet werden, sondern ergibt sich aus der sowieso notwendigen Berechnung aller Zähler für jedes Gewicht w_i.

6.2. Hardy's Multiquadric Methode

Wie bereits in Abschnitt 3.3 beschrieben, gehört Hardy's Multiquadric Methode (Hardy 1971) zu den auf radialen Basisfunktionen basierenden Verfahren. Entsprechend passt diese Methode die in Definition 3.3.7 beschriebene Interpolationsfunktion an. Dabei gilt $m = 0$, so dass das Verfahren keinen Gebrauch von polynomiellen Termen macht. Dennoch erhält es sich eine C_∞-Stetigkeit. Es folgt die vereinfachte Form des Interpolaten ohne die polynomiellen Terme:

$$\hat{f}_r(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i R(d(\mathbf{x}, \mathbf{x}_i)) \tag{6.4}$$

Damit bleibt eine gewichtete Summe einer radialen Basisfunktion erhalten. Als Basisfunktion R wählt Hardy:

$$R(r) = (r^2 + c_i^2)^{\mu_i} \quad mit \quad \mu_i \neq 0 \tag{6.5}$$

Durch die zusätzliche Festlegung auf ein einheitliches c und μ statt bestimmbarer Parameter c_i und μ_i pro bekanntem Wertepaar, vereinfacht sich die Funktion weiter. Es ergibt sich insgesamt:

$$\hat{f}_r(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i (d(\mathbf{x}, \mathbf{x}_i)^2 + c^2)^{\mu} \tag{6.6}$$

Es sind also einzig die N freien Variablen α_i zu bestimmen. Dazu lassen sich die N Gleichungen heranziehen, die benötigt werden, um die Interpolationseigenschaft sicherzustellen, so dass sich als Gleichungen ergeben:

$$\sum_{i=1}^{N} \alpha_i \underbrace{(d(\mathbf{x}_j, \mathbf{x}_i)^2 + c^2)^{\mu}}_{r_{ij}} = y_j \tag{6.7}$$

Diese lassen sich in Matrixschreibweise bringen, indem man die Berechnung der radialen Basisfunktion für jede Kombination von bekannten Werten durchführt und diese in eine symmetrische Matrix überführt. Dadurch ergibt sich dann als Gleichung:

$$\begin{pmatrix} r_{11} & \dots & r_{1N} \\ \vdots & \ddots & \vdots \\ r_{N1} & \dots & r_{NN} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \tag{6.8}$$

Diese Gleichung lässt sich mittels Matrixinvertierung einfach lösen, da sie unter der Voraussetzung von eindeutigen Wertepaaren vollen Zeilenrang besitzt und damit regulär ist. Dabei kann aber das resultierende Gleichungssystem zu schlecht

konditioniert sein, um es numerisch lösen zu können. In Literatur werden zahlreiche Verfahren vorgestellt, mit denen man die schlechte Konditionierung erkennen und umgehen kann, siehe dazu zum Beispiel (Hoschek und Lasser 1992). Dieses Problemfeld ist jedoch groß genug um eigene Arbeiten zu füllen und soll hier nicht weiter behandelt werden.

Gradientenberechnung

Wie schon bei der Shepardinterpolation lässt sich auch für diese Methode eine geschlossene Form für den Gradientenvektor finden. Sofern man als Distanzfunktion $d(x, x_i)$ den euklidischen Abstand verwendet, ergeben sich die partiellen Ableitungen nach den Komponenten von $\mathbf{x}$ zu:

$$\frac{\partial f(\mathbf{x})}{\partial x_1} = \mu \sum_{i=1}^{N} \alpha_i (x_2 - x_{i,2}) \cdot ((x_1 - x_{i,1})^2 (x_2 - x_{i,2})^2 + R^2)^{\frac{1}{2}\mu - 1} \tag{6.9}$$

$$\frac{\partial f(\mathbf{x})}{\partial x_2} = \mu \sum_{i=1}^{N} \alpha_i (x_1 - x_{i,1}) \cdot ((x_1 - x_{i,1})^2 (x_2 - x_{i,2})^2 + R^2)^{\frac{1}{2}\mu - 1} \tag{6.10}$$

Implementierung

Hardy's Multiquadric Methode ist von sich aus zunächst auf skalare Interpolation beschränkt. Man kann diese Beschränkung allerdings aufweichen. Es ergibt sich dann folgender Interpolator:

$$\hat{\mathbf{f}}_r(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i (d(\mathbf{x}, \mathbf{x}_i)^2 + R^2)^{\mu} \tag{6.11}$$

Wie man sieht, müssen aus den vormals skalaren Parametern α_i Vektoren werden, damit die Funktion überhaupt vektorwertig werden kann, denn eine Distanzfunktion kann nur skalar sein. Das Gleichungssystem zur Bestimmung der Parameter sieht also wie folgt aus:

$$\begin{pmatrix} r_{11} & \dots & r_{1N} \\ \vdots & \ddots & \vdots \\ r_{N1} & \dots & r_{NN} \end{pmatrix} \begin{pmatrix} \alpha_{11} & \dots & \alpha_{1m} \\ \vdots & \ddots & \vdots \\ \alpha_{N1} & \dots & \alpha_{Nm} \end{pmatrix} = \begin{pmatrix} y_{11} & \dots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{N1} & \dots & y_{Nm} \end{pmatrix} \tag{6.12}$$

Mit dieser Gleichung können die Parameter aller Dimensionen in einer Berechnung bestimmt werden, wodurch die erneute Berechnung und Invertierung der Distanzmatrix vermieden werden kann. Da es sich bei der Distanzmatrix um eine

symmetrische Matrix handelt, muss nur eines der Dreiecke und die Diagonale berechnet werden. Die andere Hälfte kann durch Spiegelung bestimmt werden, was den Rechenaufwand auf annähernd die Hälfte reduziert.

Um eine zu schlechte Konditionierung des Gleichungssystems zu verhindern, wird auf alle Diagonaleinträge eine sehr kleine Zehnerpotenz addiert. So lassen sich auch die Matrizen größerer Datensätze noch invertieren. Dabei führt eine solche Herangehensweise zur Aufweichung der Interpolationseigenschaft, da die Invertierbarkeit durch leicht von der eigentlichen Lösung abweichende Ergebnisse erkauft wird.

6.3. Lineare Interpolation über Finite Elemente Methode

Zur linearen Interpolation mittels finiten Elementen zerlegt man die Punktwolke der Wertepaare $(\mathbf{x}_i, y_i = f(\mathbf{x}_i)), i = 1, \ldots, n$ anhand ihrer $\mathbf{x}_i$ in Dreiecke. Diesen Vorgang nennt man Triangulation. Nach der Triangulation muss die gesamte, durch die Punkte aufgespannte, konvexe Hülle durch sich nicht überlappende Dreiecke abgedeckt sein. Dann gilt für jede Position $\mathbf{x}$ innerhalb der konvexen Hülle, dass sie innerhalb oder auf dem Rand eines Dreiecks $\Delta = ABC$ liegt. Dabei geben $A = (\mathbf{x}_A, y_A)$, $B(\mathbf{x}_B, y_B)$ und $C(\mathbf{x}_C, y_C)$ die Eckpunkte des Dreiecks an. Da es sich bei diesen Eckpunkten um bekannte Wertepaare handelt, liegt neben der Position auch jeweils ein Funktionswert vor. Jeder Punkt $\mathbf{x}$ innerhalb oder auf dem Rand des Dreiecks lässt sich dann als Linearkombination der Verbindungsvektoren von A nach B und von A nach C darstellen:

$$\mathbf{x} = s \cdot (\mathbf{x}_B - \mathbf{x}_A) + t \cdot (\mathbf{x}_C - \mathbf{x}_A) \tag{6.13}$$

Mit den beiden skalaren Faktoren s, t kann anschließend der Funktionswert interpoliert werden:

$$\hat{f}(\mathbf{x}) = y_A + s \cdot (y_B - y_A) + t \cdot (y_C - y_A) \tag{6.14}$$

Dieses Verfahren ist in Abbildung 6.1 illustriert. Auf der linken Seite wird gezeigt, wie sich die skalaren Faktoren s und t bestimmen. Diese werden dann auf der rechten Seite benutzt um die durch die Schraffur angedeutete Position auf der z-Achse zu bestimmen.

Neben der hier vorgeschlagenen Methode der linearen Interpolation, die nur eine C_0-Stetigkeit erreicht, gibt es noch weitere Verfahren, die eine Triangulierung ausnutzen. So stellt (Boissonnat und Cazals 2000) eine Methode vor, die auf der

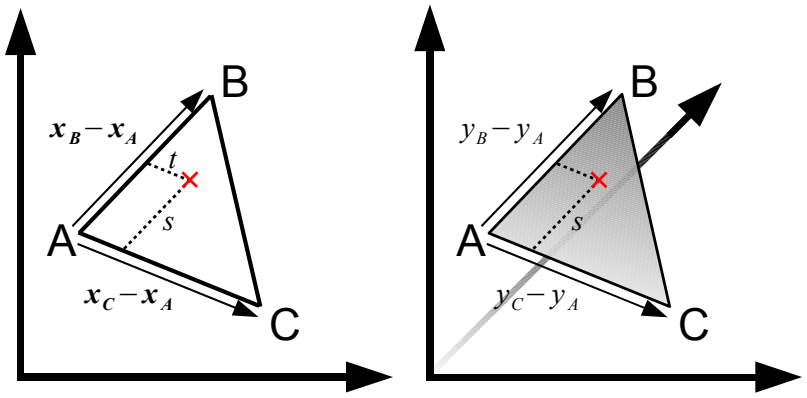


Abbildung 6.1.: Lineare Interpolation über Dreiecke

Größe der Voronoi Regionen basiert, jedoch einen vielfach gesteigerten Rechenaufwand zur Bestimmung eben dieser Voronoi Regionen benötigt. In dieser Arbeit wird trotz der zu erwartenden höheren Güte der alternativen Verfahren, lediglich die lineare Interpolation umgesetzt, um einen Kontrast zu den anderen Verfahren zu bieten, die bereits alle nicht linear arbeiten.

Daher liegt die wirkliche Herausforderung darin, ein brauchbares Dreiecksnetz zu bestimmen. In der Literatur finden sich verschiedene Verfahren zur Triangulation, die speziell auf die Interpolation von Oberflächen optimiert wurden, siehe dazu zum Beispiel (Dyn et al. 1990). Diese Arbeit wird sich allerdings auf die Vorstellung der weit verbreiteten Standardtriangulation beschränken: der Delaunay Triangulation. Die Delaunay Triangulation ist im Bereich der Computergrafik sehr verbreitet, was sich in der Entwicklung zahlreicher unterschiedlicher Verfahren zu ihrer Berechnung widerspiegelt. Mit der inkrementellen Konstruktion wird nun im Folgenden eines der Verfahren beschrieben.

Inkrementelle Triangulation

Dieser Abschnitt stellt einen einfachen Algorithmus vor, mit dem eine Delaunay Triangulation hergestellt werden kann. Nähere Informationen dazu findet man in (Fortune 1992). Zunächst wird die Delaunay Triangulation definiert:

Definition 6.3.1 (Delaunay Triangulation)
Eine Delaunay Triangulation ist eine Triangulation, bei der alle Dreiecke die Umkreisbedingung erfüllen. Die Umkreisbedingung ist erfüllt, wenn im Umkreis des Dreiecks keine anderen Punkte liegen.

Um eine solche Triangulation zu erstellen, muss zunächst eine beliebige Delaunay Triangulation bestimmt werden, die die konvexe Hülle aller Punkte $\mathbf{x}_i$ der bekannten Wertepaare umschließt. Die einfachste Variante besteht in der Erstellung eines zusätzlichen Dreiecks, das so konstruiert wird, dass es alle Punkte überdeckt. Nun können nacheinander die einzelnen Wertepaare eingefügt werden. Für jedes Wertepaar werden diejenigen Dreiecke gesucht, in deren Umkreis $\mathbf{x}_i$ liegt. Die Vereinigung aller dieser Dreiecke ergibt ein Polygon. Anschließend werden alle Ecken des Polygons mit dem neuen Punkt verbunden, so dass sich neue Dreiecke ergeben. Diese sind alle in Delaunay Triangulation und das nächste Wertepaar kann eingefügt werden.

Gradientenberechnung

Die Gradientenberechnung ist bei linearer Interpolation auf Grundlage von Dreiecken denkbar einfach, da sie über die gesamte Dreiecksfläche gleich ist. Entsprechend kann diese Berechnung pro Dreieck einmal erfolgen und muss anschließend nicht mehr durchgeführt werden. Zur Berechnung kann einfach die Stärke der Veränderung, also die Differenz an den beiden in jeder Richtung maximal entfernten Punkten, berechnet und durch die Distanz zwischen den beiden maximal entfernten Punkten geteilt werden.

6.4. Multilevel BSpline Approximation

Im Gegensatz zu allen vorher genannten Methoden, handelt es sich hier, wie der Name schon sagt, um eine Approximationsmethode. In (Lee et al. 1997) wird aber gezeigt, dass diese Approximation durch eine genügend starke Verfeinerung zu einer Interpolation wird. Dazu wird zunächst beschrieben, wie man das Problem transformieren muss, damit sich die auf Gitterstrukturen beschränkten BSplines überhaupt auf Scattered Data anwenden lassen. Zuletzt wird dann der Algorithmus vorgestellt, der die Verfeinerungen durchführen kann.

Bei der Approximation von Scattered Data liegen Wertepaare $(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, \ldots, n$ mit $\mathbf{x}_i \in \mathbb{R}^m$ vor. Dieses Verfahren ist auf $m = 2$ beschränkt, kann also nur auf zwei dimensionalen Daten angewandt werden. Daher wird im Folgenden

auch auf die Vektorschreibweise verzichtet und stattdessen beide Koordinaten als Skalare notiert, so dass gilt:

$$f(\mathbf{x}) = f(x, y) \tag{6.15}$$

Weiterhin wird vorausgesetzt, dass sich ein Rechteck finden lässt, in dem alle Datenpunkte liegen. Mit ausreichend großem Rechteck gilt dies aber für jede endliche Menge von Datenpunkten. Ohne Beschränkung der Allgemeinheit wird angenommen, dass die Daten im Rechteck zwischen 0 und a, beziehungsweise zwischen 0 und b liegen. Der durch diese Intervalle definierte Raum wird mit Ω bezeichnet:

$$\Omega = \{(x, y) | 0 \leq x \leq a, 0 \leq y \leq b\} \tag{6.16}$$

Um eine Approximationsfunktion zu beschreiben, wird über diesen Raum ein Kontrollgitter $\mathbf{L}$ der Größe $(a+3) \times (b+3)$ gelegt, das auf jeder Seite einen Index über den Raum Ω herausragt. Dadurch kommen die Elemente $L_{0,0}$, $L_{0,b}$, $L_{a,0}$, $L_{a,b}$ genau auf den Ecken von Ω zu liegen, während die Elemente $L_{-1,-1}$, $L_{-1,b+1}$, $L_{a+1,-1}$ und , $L_{a+1,b+1}$ die Ecken des Kontrollgitters bilden. Abbildung 6.2 illustriert, wie das Kontrollgitter als gestrichelte Linien über den grau gezeichneten Datenraum Ω gelegt wird.

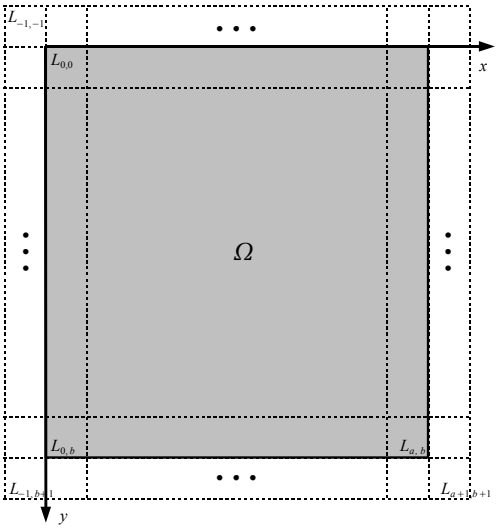


Abbildung 6.2.: Kontrollgitter

Wie ergibt sich nun aus dem Kontrollgitter die approximierte Funktion? Es wird benutzt um vier verschiedene kubische Spline Funktionen b_i zu gewichten, deren

gewichtete Summe die approximierte Funktion ergibt:

$$\hat{f}(x,y) = \sum_{k=0}^{3}\sum_{l=0}^{3} b_k(x - \lfloor x \rfloor) b_l(y - \lfloor y \rfloor) L_{\lfloor x \rfloor - 1 + k, \lfloor y \rfloor - 1 + l} \tag{6.17}$$

Um die Übersichtlichkeit zu steigern, wird im Folgenden die Funktion $p(x) = \lfloor x \rfloor - 1$, sowie die Funktion $r(x) = x - \lfloor x \rfloor$ benutzt, die den nicht ganzzahligen Teil einer Zahl zurückgibt, so dass sich die kürzere Form ergibt:

$$\hat{f}(x,y) = \sum_{k=0}^{3}\sum_{l=0}^{3} b_k(r(x)) b_l r(y)) L_{p(x)+k, p(y)+l} \tag{6.18}$$

mit b_k, b_l aus den folgenden Spline Funktionen:

$$b_0(t) = \frac{1}{6}(1-t)^3 \tag{6.19}$$

$$b_1(t) = \frac{1}{6}(3t^3 - 6t^2 + 4) \tag{6.20}$$

$$b_2(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1) \tag{6.21}$$

$$b_3(t) = \frac{1}{6}t^3 \tag{6.22}$$

Zur Gewichtung werden von der Koordinate ausgehend die vier umliegenden Punkte des Kontrollgitters und deren direkte Nachbarn genutzt. Dazu werden die Summen von 0 bis 3 gebildet, so dass sich ein Gewicht aus jedem betrachteten Kontrollpunkt ergibt. Die gewichteten Werte bestehen aus den Splinefunktionen, abhängig vom Abstand der Koordinate (x, y) zum nächsten oberen, linken Kontrollgitterpunkt. Abbildung 6.3 zeigt anschaulich, wie aus der Koordinate (x, y) die Faktoren des Interpolanten aus der Formel 6.18 bestimmt werden. Sie stellt dazu einen Ausschnitt aus dem Kontrollgitter dar, dessen Gitterpunkte durch die schwarzen Kreise markiert werden. Das schwarze Kreuz markiert die Koordinate (x, y).

Um die Werte des Kontrollgitters zu bestimmen, sei zunächst der Fall betrachtet, bei dem nur ein einziges Wertepaar $((x_c, y_c), f(x_c, y_c))$ bekannt ist. Dann lassen sich die Werte des Kontrollgitters so bestimmen, dass die Interpolationsbedingung erfüllt ist:

$$f(x_c, y_c) = \sum_{k=0}^{3}\sum_{l=0}^{3} b_k(r(x_c)) b_l(r(y_c)) L_{p(x_c)+k, p(y_c)+l} \tag{6.23}$$

Da dieses Problem mehrdeutig lösbar ist, wird in (Lee et al. 1997) als Lösung eine die Quadratsumme der Kontrollgitterwerte minimierende Formel vorgeschlagen:

$$L_{i+k,j+l} = \frac{b_k(r(x)) b_l(r(y)) f((x_c, y_c))}{\sum_{a=0}^{3}\sum_{b=0}^{3} b_a(r(x)) b_b(r(y))} \tag{6.24}$$

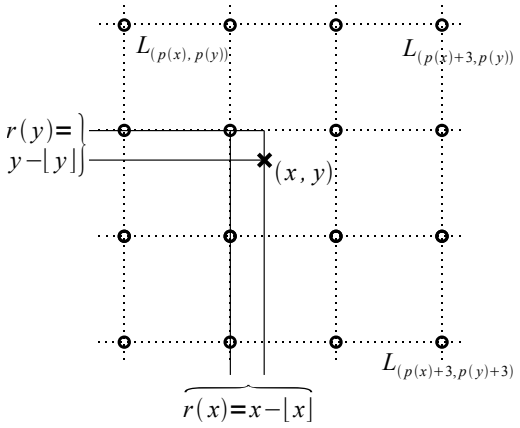


Abbildung 6.3.: Koordinatenauswertung

Dabei ist die Interpolationseigenschaft in dem Sinne lokal, dass nur die 16 benachbarten Punkte beeinflusst werden. Wenn also keine zwei Datenpunkte so nahe aneinander liegen, dass sich ihre Nachbarschaften überlappen, entspricht die Funktion einer Interpolation. Dies gibt einen Hinweis darauf, wie später durch Verfeinerung des Gitters die Interpolation angenähert werden kann.
Im Fall von sich überschneidende Nachbarschaften kann die Interpolationseigenschaft nicht erhalten bleiben. Für jeden Punkt $L_{i,j}$ im Kontrollgitter $\mathbf{L}$ gibt es dann eine Menge $\mathcal{P}_{i,j}$ von Wertepaaren, die diesen Punkt in der Nachbarschaft haben:

$$\mathcal{P}_{i,j} = \{(x,y) | i-2 \leq x < i+2, j-2 \leq y < j+2\} \tag{6.25}$$

Dann wird jeder Gitterpunkt so bestimmt, dass der Fehler auf allen bekannten Wertepaaren aus $\mathcal{P}_{i,j}$ minimiert wird. Bei der Lösung dieser Minimierungsaufgabe wird allerdings vorausgesetzt, dass alle anderen Gitterpunkte für alle Punkte die optimale Lösung aus 6.24 erfüllen. Da sich für zwei Wertepaare mehrere Gitterpunkte überlappen können, ist diese Voraussetzung aber häufig unzutreffend. Die unter dieser Einschränkung optimale Lösung ergibt sich zu:

$$L_{i,j} = \frac{\sum_{(x,y)\in\mathcal{P}_{i,j}} (b_{i+1-\lfloor x \rfloor}(r(x)) b_{j+1-\lfloor y \rfloor}(r(y)))^2 \phi(x,y)}{\sum_{(x,y)\in\mathcal{P}_{i,j}} (b_{i+1-\lfloor x \rfloor}(r(x)) b_{j+1-\lfloor y \rfloor}(r(y)))^2} \tag{6.26}$$

Dabei ist $\phi(x,y)$ der beste Wert für $L_{i,j}$, falls (x,y) der einzige Punkt in $\mathcal{P}_{i,j}$ wäre. Entsprechend bestimmt sich $\phi(x,y)$ aus Gleichung 6.24.

Multilevel Verfeinerung

Wie im vorigen Abschnitt gezeigt wurde, lässt sich über eine genügend starke Verfeinerung des Gitters die Interpolationseigenschaft sicherstellen. Da Kontrollgitterpunkte aber mit null initialisiert werden, und es bei einer so starken Verfeinerung zahlreiche Punkte geben wird, die in keiner Nachbarschaft liegen und daher weiterhin null bleiben, wird die Interpolationsfunktion dann in weiten Bereichen null. Da die Nachbarschaft auf 16 Knotenpunkte beschränkt bleibt, werden die von null verschiedenen Bereiche zu kleinen, steilen Peaks. Um nun also den weichen Verlauf einer Abschätzung mit einem groben Kontrollgitter mit der Interpolationseigenschaft zu kombinieren, kann ein Multilevel Ansatz gewählt werden. Dazu bildet man ein grobes, initiales Kontrollgitter. Berechnet dann für die bekannten Werte die Fehler und benutzt diese als bekannte Werte um ein doppelt so hoch aufgelöstes Kontrollgitter auf der nächsten Stufe zu berechnen. Dieses Verfahren kann iteriert werden um den Approximationsfehler zu verringern. Da bei der anschließenden Berechnung die Summe über zahlreiche Ebenen gebildet werden muss, steigt auch der Rechenbedarf bei der Anwendung. Um dies zu verhinden, kann die B-Spline Refinement Technik angewandt werden um alle Ebenen in einem Kontrollgitter der höchsten Auflösung darzustellen.
Die B-Spline Refinement Technik lässt sich einsetzen um ein Kontrollgitter $\mathbf{L}^t$ der t-ten Ebene mit einer Auflösung von $n+3 \times m+3$ in ein Kontrollgitter $\mathbf{L}'$ mit einer Auflösung von $2n+3 \times 2m+3$ umzurechnen. Dabei muss gelten, dass die Approximationsfunktion aus Gleichung 6.18 unter beiden Kontrollgittern identische Ausgaben liefert:

$$\hat{f}_{\mathbf{L}^t}(x,y) = \hat{f}_{\mathbf{L}'}(x,y) \tag{6.27}$$

Dazu werden in (Lee et al. 1997) folgende Formeln benutzt, mit denen sich aus jedem Punkt $L^t_{i,j}$ vier Punkte des Kontrollgitters $\mathbf{L}'$ berechnen lassen:

$$\begin{aligned}
L'_{2i,2j} &= \frac{1}{64}(L^t_{i-1,j-1} + L^t_{i-1,j+1} + L^t_{i+1,j-1}L^t_{i+1,j+1} + \\
&\quad 6(L^t_{i-1,j} + L^t_{i,j-1} + L^t_{i,j+1} + L^t_{i+1,j}) + 36L^t_{i,j}) \\
L'_{2i,2j+1} &= \frac{1}{16}(L^t_{i-1,j} + L^t_{i-1,j+1} + L^t_{i+1,j}L^t_{i+1,j+1} + 6(L^t_{i,j} + L^t_{i,j+1})) \\
L'_{2i+1,2j} &= \frac{1}{16}(L^t_{i,j-1} + L^t_{i,j+1} + L^t_{i+1,j-1}L^t_{i+1,j+1} + 6(L^t_{i,j} + L^t_{i+1,j})) \\
L'_{2i+1,2j+1} &= \frac{1}{4}(L^t_{i,j} + L^t_{i,j+1} + L^t_{i+1,j}L^t_{i+1,j+1})
\end{aligned}$$

Wurde auf diese Weise ein verfeinertes Gitter $\mathbf{L}'$ berechnet, kann auf den Approximationsfehlern der t-ten Ebene das Kontrollgitter $\mathbf{E}^{t+1}$ berechnet werden. Es

besitzt dann ebenfalls die Auflösung $2n+3 \times 2m+3$, dadurch lässt sich $\mathbf{L}^{t+1}$ durch einfaches Summieren von $\mathbf{L}'$ und $\mathbf{E}^{t+1}$ bestimmen. Wiederholt man dieses Verfahren, kann man eine beliebige Auflösungstiefe erreichen, ohne jemals mehr als zwei Gitter der höchsten Auflösung gleichzeitig im Speicher halten zu müssen.

Gradientenbildung

Auch bei diesem Verfahren kann der Gradient geschlossen berechnet werden, hier erscheint jedoch die Verwendung einer Näherungslösung brauchbarer, da sich die partiellen Ableitungen als sehr sperrig erweisen. Zur Demonstration gibt die nächste Gleichung die partielle Ableitung nach x an:

$$\begin{aligned}
&\frac{1}{12}(r(y)-1)^3(r(x)-1)^2 L_{p(x),p(y)} + \frac{1}{4}(r(x)-1)^2(r(y)^3 - \frac{7}{3} - r(y)^2)L_{p(x),\lfloor y\rfloor+1} + \\
&(\frac{1}{4}r(x) - 1/6)r(x)(r(y)-1)^3 L_{\lfloor x\rfloor+1,p(y)} + (\frac{3}{4}r(x) - \frac{1}{2})r(x)(r(y)^3 - \\
&\frac{7}{3} - r(y)^2)L_{\lfloor x\rfloor+1,\lfloor y\rfloor+1} - \frac{9}{4}(r(x)-1)^2 L_{p(x),\lfloor y\rfloor+2} + (-\frac{27}{4}r(x)^2 + \\
&\frac{9}{2}r(x))L_{\lfloor x\rfloor+1,\lfloor y\rfloor+2} - \frac{1}{4}(r(x)-1)^2 \frac{4}{3+r(y}^3 - 2r(y)^2)L_{p(x),\lfloor y\rfloor} + \\
&(-(\frac{1}{4}r(x) - 1/3)(r(y)-1)^3 L_{\lfloor x\rfloor,p(y)} - \\
&(\frac{3}{4}r(x) - \frac{1}{2})\frac{4}{3+r(y}^3 - 2r(y)^2)L_{\lfloor x\rfloor+1,\lfloor y\rfloor} + \\
&(\frac{3}{4}r(x) - 1)((r(y)^2 + \frac{7}{3} - r(y)^3)L_{\lfloor x\rfloor,\lfloor y\rfloor+1} + \\
&9L_{\lfloor x\rfloor,\lfloor y\rfloor+2} + \frac{4}{3+r(y}^3 - 2r(y)^2)L_{\lfloor x\rfloor,\lfloor y\rfloor}))r(x)
\end{aligned}$$

Als Näherungslösung wird stattdessen die Höhendifferenz zu vier umliegenden Punkten berechnet und diese gemittelt.

7. Visualisierungslayer

In den letzten Kapitel wurden Verfahren zur Dimensionsreduktion eingeführt. Um deren Ergebnisse von den endlich vielen definierten Punkten auf den gesamten Kartenraum auszudehnen, wurden Interpolationsverfahren vorgestellt. Diese sorgen dafür, dass jede Position auf der Karte einem Vektor aus dem Beispielraum zugewiesen ist. Da der Beispielraum als hochdimensional angenommen werden muss, benötigt man passende Visualisierungsverfahren, um aussagekräftige Informationen zu extrahieren.
Dieses Kapitel stellt nun einige sogenannte Layer vor, die jeweils einen Aspekt aus den zur Verfügung stehenden Informationen darstellen. Die Layer können dabei übereinander gelegt und transparent geschaltet werden um dem Benutzer verschiedene Aspekte gleichzeitig präsentieren zu können. Dies können zum Beispiel die Werte im Beispielraum von zwei verschiedenen Dimensionen sein, die gleichzeitig mit dem Ergebnis des Entscheidungsmodells verglichen werden sollen.

7.1. Punktlayer

Das Punktlayer ordnet die Beispiele im Kartenraum an und gibt ihre Positionen über kleine Kreise wieder. Die Kreise können nach einer der Beispielraumdimensionen eingefärbt werden. Dabei stellt sich die Frage nach der günstigsten Position im Kartenraum. Diese sollte den Abstand zwischen Beispielvektor und Beispielraumvektor der Karte minimieren, liegt aber nicht notwendigerweise direkt auf einer der Nodes, da die interpolierten Zwischenräume näher an dem Beispielvektor liegen können. Diese Arbeit stellt zur Positionierung vier verschiedene Strategien zur Verfügung. Dies ist zum einen die nächste Node zu wählen, was bei Verwendung der DSOM und eines Interpolationsverfahrens dafür sorgt, dass für bekannte Beispiele der Beispielvektor und der Beispielraumvektor der Node exakt gleich sind. Die Alternativen bestehen aus einem Gittersuchverfahren, einer Partikelschwarm oder evolutionären Optimierungsstrategie, die jeweils den Abstand minimieren und auch mit unbekannten Daten arbeiten können. Die Abbildung 7.1 zeigt die Punkteanordnung der Trainingsdaten des bekannten Iris-Datensatzes bei Verwendung einer DSOM.

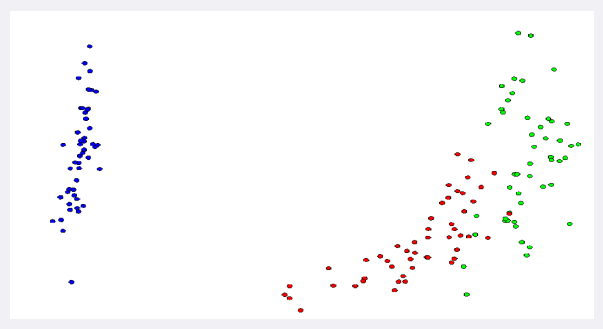


Abbildung 7.1.: Darstellung eines Punktlayers über einer DSOM

7.2. Modelllayer

Das Modelllayer bekommt als Eingabe ein Entscheidungsmodell $f : \mathbb{D}^m \rightarrow \mathcal{C}$ wie in Definition 2.1.1 vorgestellt. Da durch die Interpolationsverfahren an jeder Position $\mathbf{c}$ des Kartenraums $\mathbb{K}^2$ ein Vektor $\mathbf{x}$ aus dem Beispielraum $\mathbb{D}^m$ bekannt ist, kann dieser mit dem Entscheidungsmodell einer Klasse zugewiesen werden. Da außerdem die Menge $\mathcal{C}$ der Klassen endlich ist, kann jede Klasse über eine eigene Farbe dargestellt werden, so dass sich eine Art Flickenteppich ergibt. Ein Beispiel für ein Modelllayer gibt Abbildung 7.2.

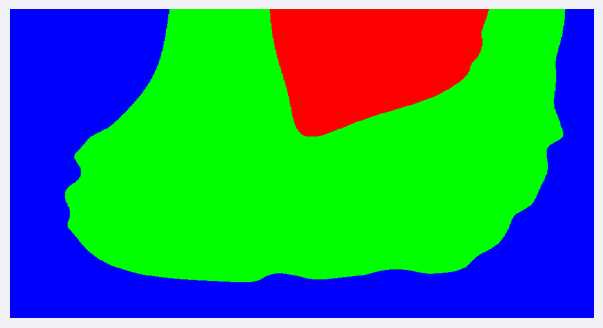


Abbildung 7.2.: Darstellung eines Modelllayers

7.3. Absolutlayer

Das Absolutlayer gibt, wie der Name schon andeutet, den absoluten Wert einer der Dimensionen im Ursprungsraum an. Dabei ist absolut hier nicht als Betrag zu verstehen, da die Daten eh zwischen 0 und 1 normalisiert sind, sondern vielmehr als "nicht-relativ". Im Unterschied zum Distanzlayer 7.4 wird einfach der Wert an dieser Position unabhängig von den umgebenden Werte wiedergegeben. Der Benutzer muss dabei die Dimension auswählen, die er betrachten will. Deren Werte werden dann intern zwischen 0 und 1 normalisiert, so dass sich eine Landschaft berechnen lässt, in der Meeresbecken Werte nahe 0 darstellen und schneebedeckte Gipfel Werte bei 1. Es ergibt sich also tatsächlich eine Landkarte, wie der Benutzer sie aus dem Alltag gewohnt ist. Dass ein großes Höhenniveau auch hohe Werte widerspiegelt, ist dabei sofort eingängig und verständlich. Ein Beispiel für ein solches Layer gibt Abbildung 7.3.

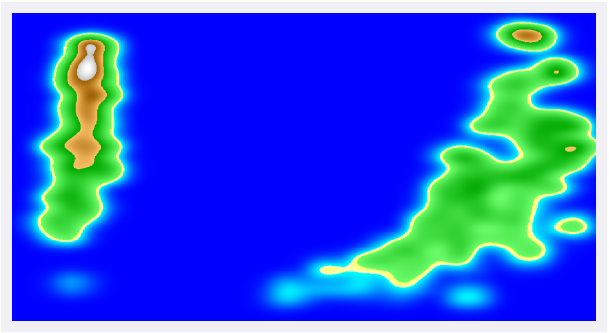


Abbildung 7.3.: Darstellung eines Absolutlayers

7.4. Distanzlayer

Das Distanzlayer deutet an, wie groß die Entfernung zwischen zwei benachbarten Punkten im Ursprungsraum ist. Dieses Layer wertet dazu die von den Interpolationsverfahren gelieferten Gradientenvektoren aus. Je länger diese sind, desto weiter sind die benachbarten Punkte voneinander entfernt. Diese Idee geht auf die Veröffentlichungen von Ultsch aus (Ultsch 2003a) und (Ultsch 2003b) zurück. Da dort aber lediglich auf ESOMs gearbeitet wurde, konnte Ultsch vereinfachend nur den Abstand zwischen den einzelnen Nodes der Gitterstruktur berechnen und dann deren Werte interpolieren. Da aber nicht bei allen hier vorgestellten Verfahren eine

Gitterstruktur vorliegt, wurde dieser Ansatz in dieser Arbeit verallgemeinert auf die Verwendung von Gradientenvektoren an den interpolierten Ebenen. Ein Beispiel für ein solches Distanzlayer liefert die Abbildung 7.4, über die ein Punktlayer gelegt wurde.

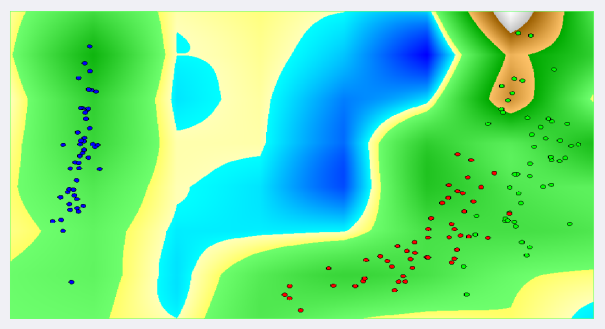


Abbildung 7.4.: Darstellung eines Distanzlayers mit angeordneten Punkten

8. Experimentelle Ergebnisse

In diesem Kapitel sollen die in den vorhergehenden Kapiteln vorgestellten Verfahren getestet und ihre Leistungen verglichen werden. Da das Verhalten aller Verfahren zur Dimensionsreduktion, mit Ausnahme der DSOM, bereits in den jeweiligen Papern untersucht wurde, wird an dieser Stelle auf eine genaue Beschreibung ihrer Verhalten verzichtet. Dafür wird die DSOM im Folgenden anhand verschiedener Datensätze ausführlich untersucht. Dabei wird auch der Einfluss der Verwendung von Kernfunktionen zur Distanzberechnung untersucht.
Anschließend sollen alle Verfahren auf Tauglichkeit für das gesetzte Ziel der für den Nutzer verständlichen Modellvisualisierung geprüft werden. Probleme bereitet dabei zum einen die Bestimmung von aussagekräftigen, quantifizierbaren Gütemaßen, die nicht nur den Vergleich der einzelnen Verfahren ermöglichen, sondern tatsächlich auch zur Optimierung ihrer Parameter dienen können. Denn ohne eine Optimierung der Parameter bleiben die Gütewerte bedeutungslos, da vor allem bei den Interpolationsverfahren ein falsch gewählter Parameter schnell zu völlig irreführenden Ergebnissen führen kann. Zum anderen ist der Raum der möglichen Parametereinstellungen durch die zahlreichen, verschieden zu kombinierenden Verfahren zur Interpolation und Positionierung sehr groß. Aber genau diese Kombination der Bestandteile bestimmt maßgeblich die Leistung des Gesamtsystems.
Daher soll die Untersuchung der einzelnen Bestandteile vor der Bestimmung der Gesamtleistung erfolgen. Da die Bestandteile alleine nicht mit dem gewünschten Ziel getestet werden können, werden hier jeweils alle anderen Teile konstant gehalten, so dass die Leistung einzelner Verfahren geprüft wird. So werden zunächst in Abschnitt 8.3 die Interpolationsverfahren untersucht, um das beste zu bestimmen. Da zur Validierung und einzig zur Validierung der Interpolationsverfahren Positionierverfahren notwendig sind, wird in diesem Abschnitt zunächst der beste Positionierer bestimmt, bevor die Interpolationsverfahren optimiert werden.
Nach der Untersuchung der Interpolatoren können auf deren Basis die Visualisierungsleistungen der SOMs verglichen werden, was im Abschnitt 8.4 erfolgt. Da eigentlich der Benutzer im Zentrum der Bemühungen steht, soll neben objektiven Performanzkriterien auch untersucht werden, ob die entwickelten Maße überhaupt dem Eindruck des Benutzers entsprechen. Ob also eine entsprechend hohe Performanz auch in einsichtigen Grafiken mündet. Dazu werden die Ergebnisse einer

Auswahl der besten Durchläufe abgebildet und besprochen.

8.1. Analyse der Distance Self-Organizing Map

Die Distance Self-Organizing Map versucht, wie alle anderen SOMs auch, eine in den Daten enthaltene Struktur möglichst gut auf zwei Dimensionen abzubilden. Da sie dabei grundlegend anders vorgeht als die anderen Verfahren, muss untersucht werden, ob das Verfahren überhaupt die erwarteten Ergebnisse liefert. Dazu werden hier drei verschiedene, künstlich generierte Datensätze mit bekannten Charakteristiken vorgestellt und die Ergebnisse der SOM gezeigt.

3 Ringe

Zunächst soll geprüft werden, ob der Algorithmus überhaupt in der Lage ist, die Struktur in den Daten nur über die Entfernung zwischen den Datenpunkten zu reproduzieren. Daher wird hier ein zweidimensionaler Datensatz gewählt, so dass Beispielraum und Kartenraum von derselben Dimension sind. Es können also keine Strukturen enthalten sein, die nicht im Kartenraum abgebildet werden können. Als Beispiel dient hier der von RapidMiner generierte Datensatz Three Ring Clusters mit 500 Beispielen, wie in Abbildung 8.1 zu sehen. Die darauf folgende Abbildung 8.2 zeigt die DSOM, wie sie sich der vorgegebenen Struktur anpasst. Im ersten Bild sieht man, wie zu Beginn die Verteilung der Nodes absolut zufällig erfolgt. Bereits nach nur 16 Schritten, die hier das Anpassen an jeweils nur einen einzigen Impuls bedeuten, sieht man, wie die Nodes der inneren Ringe zur Mitte streben, und die anderen nach außen. Diese gleichzeitige Implosion und Explosion hat nach nur 100 Schritten die grobe Struktur bereits wieder hergestellt. Im letzten Bild ist die Struktur bis auf eine Drehung komplett rekonstruiert.

Dabei ist es keine Überraschung, dass die Struktur gedreht vorliegt. Implizit stellt die gesamte DSOM einen vollständigen Graphen mit Kantengewichten dar. Anschaulich beschrieben handelt es sich um zahlreiche mit Federn verbundenen Kugeln, wobei die Federspannung der Entfernung entspricht. Eine solche Struktur besitzt ein Gleichgewicht, in dem sich alle Federspannungen ausgleichen. Nachdem ein solches erreicht wurde, kann sie beliebig gedreht und gespiegelt werden, ohne den Gleichgewichtszustand zu zerstören.

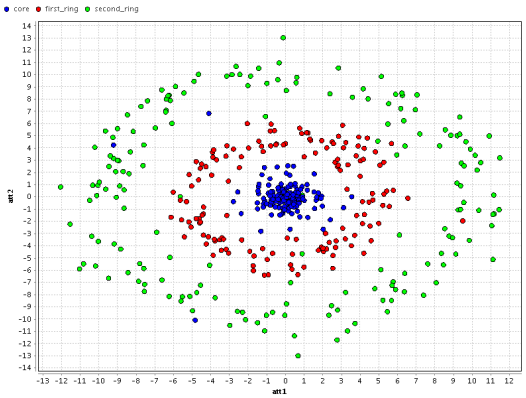


Abbildung 8.1.: Darstellung des zweidimensionalen Three Ring Datensatzes

S-förmige Mannigfaltigkeit

Der Datensatz wurde generiert um die Fähigkeiten zur Dimensionsreduktion zu testen. Dabei beschränkt er sich auf den einfachen Fall, bei dem dreidimensionale Daten mit einer enthaltenen zweidimensionalen Mannigfaltigkeit auf zwei Dimensionen abgebildet werden sollen. Entsprechend könnte man die Mannigfaltigkeit einfach "plattklopfen" um das optimale Resultat zu erhalten.

Die hier verwendete Mannigfaltigkeit ist S-förmig und wurde nach folgender Vorschrift generiert: x wird zufällig aus dem Intervall $[-3; 3]$ gezogen, $y = x^3 - 5x$ und z wird ebenfalls zufällig aus $[0; 1]$ gezogen. Es ergibt sich also ein S, das sich durch die ganze Tiefe des Raums zieht. Abbildung 8.3 zeigt den Datensatz. Die Farbe der Punkte illustriert dabei zusätzlich die z-Koordinate um die Struktur deutlich zu machen. Die folgende Abbildung 8.4 zeigt dann das Konvergenzverhalten der DSOM. Das Endresultat aus Abbildung 8.5 visualisiert die z-Koordinate als Punktfarbe und zusätzlich im Hintergrund die x-Koordinate als Landschaft.

Man sieht also, dass die DSOM sowohl die x, als auch die z Koordinate verwendet um die Punkte anzuordnen, während sie die y Koordinate vernachlässigt. Dies entspricht genau dem "Plattklopfen" der Mannigfaltigkeit.

Hyperkugel

In diesem Abschnitt soll getestet werden, wie sich die DSOM verhält, wenn die Daten eine Struktur enthalten, die nicht ohne häufige Verletzung der Entfernungsrelation auf zwei Dimensionen abbildbar ist. Will man eine normale, dreidimensionale Kugel auf zwei Dimensionen abbilden, bleiben nur zwei Möglichkeiten. Man

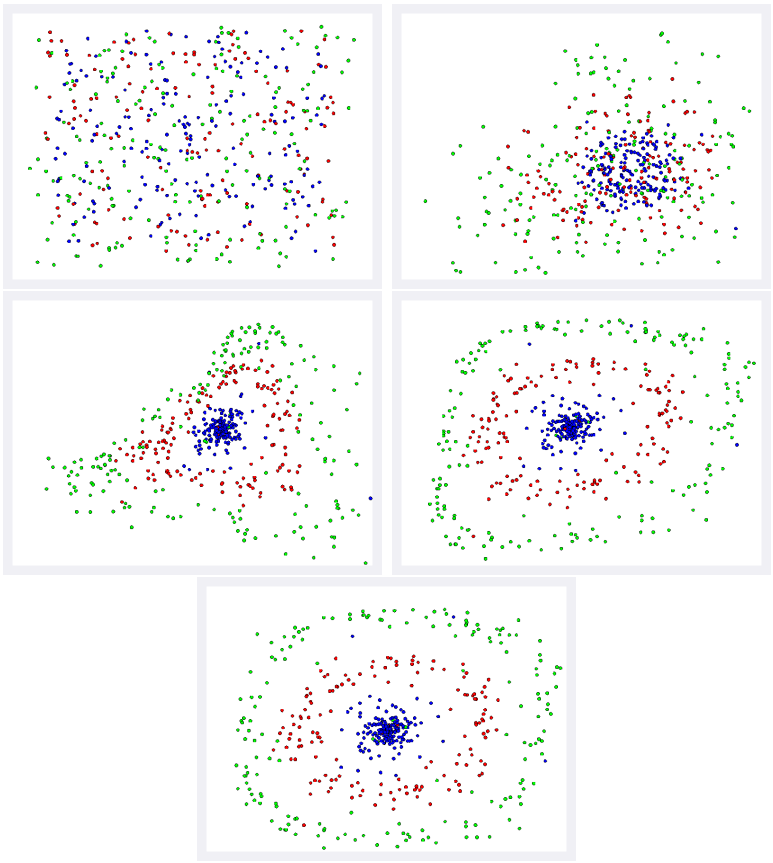


Abbildung 8.2.: Adaption an den Three Ring Datensatz nach den Schritten 1, 16, 40, 100 und 10.000

schneidet die Kugel entweder auf und breitet ihre Oberfläche im Kartenraum aus, oder man staucht sie an einer Achse durch den Mittelpunkt bis sie eine Ebene bildet. In beiden Fällen ist die Entfernungsrelation für bestimmte Punkte zerstört. Im ersten Fall gilt dies für die Punkte nahe der Schnittgrenze, deren ehemalige Nachbarn jenseits der Schnittgrenze nun die entferntesten Punkte darstellen. Im zweiten Fall gilt das für die Punkte nahe der Stauchungsachse, bei denen die ehemals entferntesten, genau auf der anderen Kugelseite liegenden Punkte nun die nächsten sind.
Da die DSOM mit der hier vorgestellten Adaptionsfunktion darauf ausgerichtet ist, ähnliche Beispiele nahe beieinander anzuordnen, kann man erwarten, dass die Kugel gestaucht und nicht aufgeschnitten wird. Denn dann sind zwar einige im

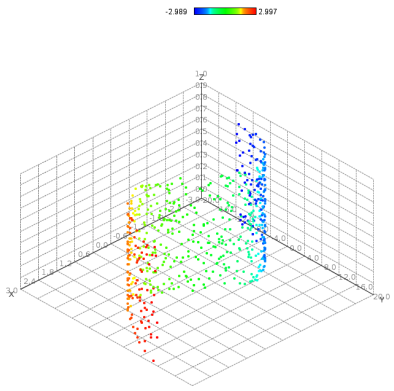


Abbildung 8.3.: Darstellung der dreidimensionalen Mannigfaltigkeit

Beispielraum entfernte Punkte zu nahe aneinander, dies wird durch die Adaptionsfunktion jedoch nicht bestraft.

Da die dritte Dimension anschaulich bleibt, wird zunächst das Verhalten in nur drei Dimensionen betrachtet. Dazu wurde ein Datensatz aus 500 Punkten erzeugt, die auf der Oberfläche einer Kugel im $\mathbb{R}^3$ liegen. Die Abbildung 8.6 zeigt die Ergebnisse einer DSOM Dimensionsreduktion. Die Farben der Punkte geben den Wert in der jeweiligen Dimension wieder, wobei jede Dimension durch ein Bild dargestellt wird. Wie vermutet, wurde die gesamte Kugel gestaucht, was man gut daran erkennen kann, dass die Farbverläufe in der x und y Dimension orthogonal verlaufen. Das letzte Bild zeigt eine Kugel, wie man sie auch von oben gesehen wahrnehmen würde. Zum Vergleich zeigt Abbildung 8.7 die Ursprungsdaten ebenfalls von oben und aus einem Winkel, der der Stauchungsachse ähnlich gewesen sein könnte.

Um zu überprüfen, ob sich dieses Verhalten auch bei höherer Dimension zeigt, wurde ein Datensatz von 1500 Punkten erzeugt, der eine Einheitshyperkugel aus der 5. Dimension beschreibt. Die höhere Anzahl an Punkten wird hier verwendet, da sich die Veränderung der Werte anhand der Farben besser erkennen lassen, wenn die Punkte dichter beeinander liegen. Weiterhin liegen alle Punkte auf der Oberfläche der Einheitshyperkugel. Die Ergebnisse aus Abbildung 8.8 zeigen zwei Auffälligkeiten.

Zum einen sind die Punkte nicht mehr kreisförmig angeordnet, sondern nähern sich einem Viereck mit abgerundeten Ecken an. Zum anderen fällt auf, dass sich nun mehr Punkte in der Mitte der Abbildung befinden als auf dem Rand. In

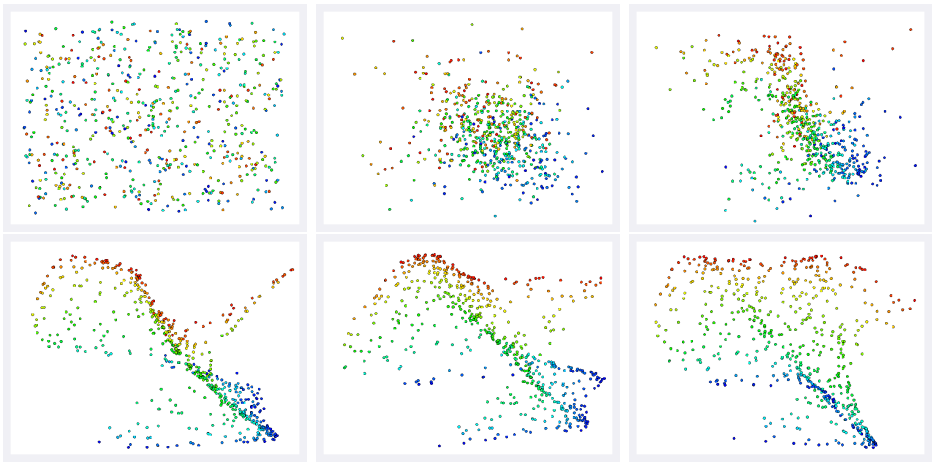


Abbildung 8.4.: Adaption an die S-Mannigfaltigkeit nach den Schritten 1, 27, 61, 139, 316 und 720

Verbindung legen die beiden Beobachtungen nahe, dass die Hyperkugel nicht mehr nur an einer Achse gestaucht wurde. Denn ein solches Bild entstünde, wenn man zwei gegenüberliegende Punkte einer Kugel eindrückt, so dass sie sich in der Mitte berühren. Dann erscheint die Kugel in der Seitenansicht als Viereck und in der Mitte befänden sich mehr Punkte als Außen.

Die Arbeitsweise bleibt jedoch im Grund erhalten: Die DSOM staucht die Hyperkugel auf zwei Dimensionen. Dies lässt sich besonders gut an den beiden Abbildungen für die Dimensionen a und b erkennen, deren Verläufe weiterhin orthogonal zueinander stehen. Die Winkel zwischen den Abbildungen der ursprünglichen Koordinatensystemachsen werden aber naturgemäß mit steigender Dimensionalität immer geringer, da sich weiterhin nur 360 Grad auf mehr Achsen verteilen müssen. Daher ist verständlich, warum man auf den ersten beiden Bildern nur erahnen kann, wie die Achsen verlaufen.

Vergleich zum Locally Linear Embedding

Bei Locally Linear Embedding handelt es sich um das einzige andere, hier vorgestellte, nicht-lineare Verfahren zur Dimensionsreduktion, das die Positionen der Daten im Kartenraum direkt bestimmt. Daher wurde es hier als Kandidat für einen Vergleich herangezogen, für den die Datensätze der S-förmige Mannigfaltigkeit und der Hyperkugel im $\mathbb{R}^5$ verwendet werden. Die Ergebnisse zeigt Abbildung 8.9. Man erkennt zwar auch hier eine Struktur in den Daten, jedoch ist diese nicht anschau-

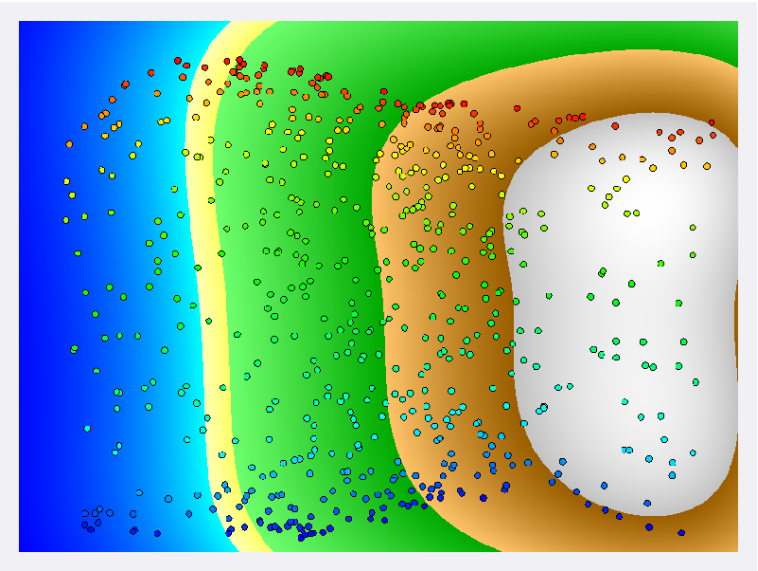


Abbildung 8.5.: Resultat der Adaption an die S-Mannigfaltigkeit nach 3728 Schritten

lich herleitbar aus den tatsächlichen Mannigfaltigkeiten. Desweiteren musste für das Verfahren eine Datenauswahl getroffen werden, da das Gleichungssystem auch nach über 20 Stunden Rechenzeit noch nicht gelöst war. Wurden einige Punkte durch ein zufälliges Sampling entfernt, lief die Berechnung in Sekunden ab. Dieses instabile Verhalten, das sich vermutlich auf numerische Ungenauigkeiten zurückführen lässt, ist in der Praxis kaum hinnehmbar und ein großer Nachteil des Verfahrens.

8.2. Analyse der Kernel Distance Self-Organizing Map

Wie beschrieben lässt sich die DSOM sehr einfach auf Entfernungen im Kernraum erweitern. Die Analysen im letzten Abschnitt haben gezeigt, dass die DSOM in der Lage ist, die ursprüngliche Anordnung der Punkte weitestgehend wiederherzustellen, wenn die Ursprungsdimension gleich der Kartenraumdimension ist. Dabei verwendete sie lediglich die Informationen über die Distanzen zwischen den Punkten. Die Überlegung ist nun, ob sie dann über die Distanz im Kernraum auch die Punkte gemäß ihrer Anordnung im Kernraum abbilden kann. Dies würde dem Be-

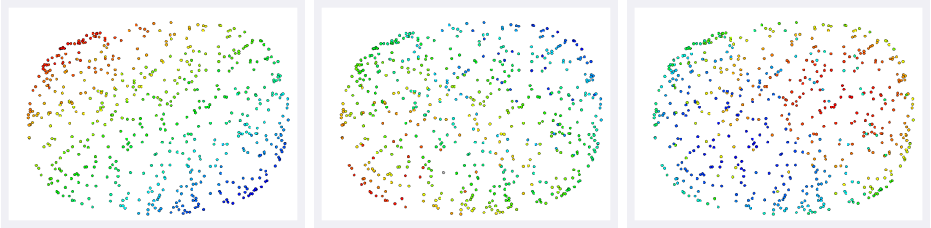


Abbildung 8.6.: Die Koordinaten x, y und z einer durch die DSOM auf zwei Dimensionen projiezierten Kugel

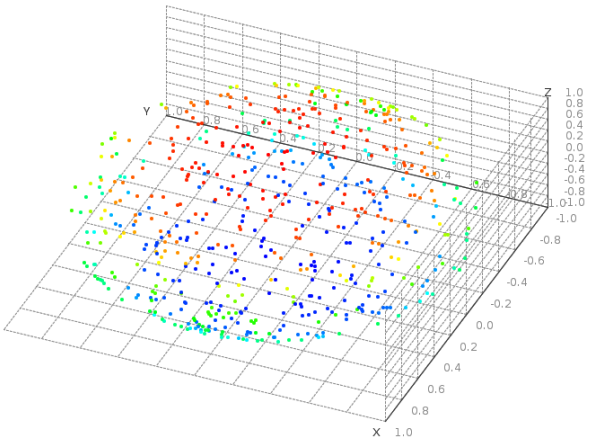


Abbildung 8.7.: 3D-Plot der Daten vor der Dimensionsreduktion

nutzer einen Einblick in das komplexe Verhalten von SVMs bieten, die die Daten im Kernraum mit einer linearen Hyperebene trennen. Da aber niemals der Kernraum sichtbar wird, kann keinerlei Verständnis für die Abhängigkeiten entwickelt werden. An dieser Stelle soll mit Hilfe einfacher Datensätze geprüft werden, ob die KDSOM dies leistet.

Quadratischer Zusammenhang

Dazu wurde ein Datensatz erzeugt, dessen einzige Dimension x gleichverteilt zwischen -1 und 1 belegt wurde. Die versteckte Variable ist das Quadrat dieser Dimension und das Label wird erzeugt nach der Vorschrift: "true" falls $x^2 - 0.1x > 0.8$, sonst "false". Mit der versteckten Variable ergibt sich das Bild, wie in Abbildung

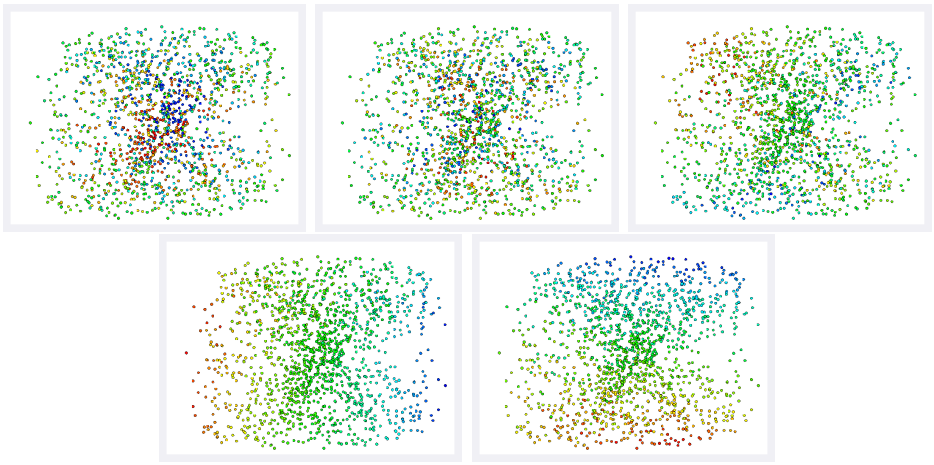


Abbildung 8.8.: Die Koordinaten x, y, z, a und b einer durch die DSOM auf zwei Dimensionen projiezierten Hyperkugel der 5. Dimension

8.10 gezeigt. Man sieht deutlich, wie die SVM hier die Klassen durch eine lineare Hyperebene separieren kann, während das nur mit den Informationen aus einer Dimension nicht funktionieren kann. Für den Test wurde die explizite quadratische Ergänzung aus dem Datensatz entfernt, so dass nur noch die Information über die Entfernung im Kernraum verfügbar ist. Abbildung 8.11 zeigt die Instabilität des Ergebnisses. Selbst kleine Änderungen an den Parametern können aus der gewünschten Darstellung eine Perlenschnur machen. Dabei spielt die Form der Darstellung eine besondere Rolle, denn die Perlenschnur ist besonders anfällig gegenüber den zufällig verteilten Impulsen in den weit entfernten Ecken: Die nächste Node ist eventuell eine ganze Seitenlänge entfernt und wird entsprechend stark in diese Richtung gezogen. Gleichzeitig zieht sie ihre Nachbarn sehr stark mit in diese Richtung, so dass sich die Nodes hin- und herwinden und jederzeit stark dem Zufall unterworfen sind. Diese Zufälligkeiten überlagern den Effekt der quadratischen Kernfunktion. Dass dieser vorhanden ist, zeigt die linke Abbildung, da dort nach einem Impuls beide Enden der Kette stark in eine Richtung gezogen wurden.

In zukünftigen Arbeiten könnte die Adaptionsfunktion so angepasst werden, dass sie stärker auf geringe Distanzunterschiede reagiert und dafür invarianter gegenüber den zufällig gezogenen Impulsen ist. Dadurch könnte sich der Effekt stärker herauskristallisieren.

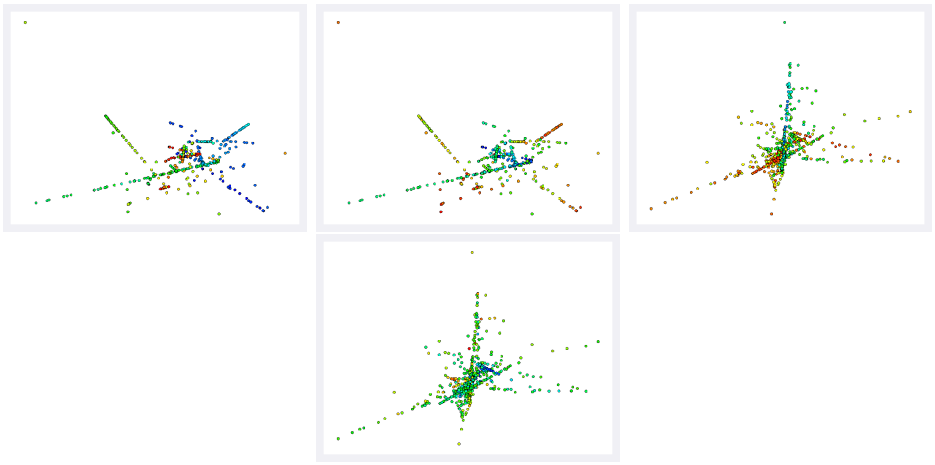


Abbildung 8.9.: Die Koordinaten x, y,z, a und b einer durch die DSOM auf zwei Dimensionen projiezierten Hyperkugel der 5. Dimension

8.3. Vergleich der Interpolatoren

Den Interpolatoren kommt im Gesamtkonzept eine entscheidende Rolle zu: Sie müssen dafür sorgen, dass sich von einigen wenigen Punkten aus eine Struktur ausbreitet, die möglichst nahe an derjenigen liegt, die tatsächlich in den Daten steckt. Da diese Struktur bei echten Datensätzen unbekannt bleibt, benötigt man einen anderen Ansatz. Zum Testen soll hier ein Verfahren analog zur Kreuzvalidierung verwendet werden. Der Gedanke dahinter ist, dass zunächst im Trainingsschritt auf einem Teildatensatz ein Dimensionsreduktionsverfahren gelernt wird. Auf dessen Ergebnissen arbeiten dann die Interpolatoren und bilden den ganzen Kartenraum in den Beispielraum ab. Diese Abbildung wird dann in einem Validierungsschritt genutzt um den Abbildungsfehler der Beispiele aus dem Testdatensatz zu berechnen. Das Verfahren spiegelt unter der Annahme, dass die unbekannten Beispiele ebenfalls der Struktur in den Daten entsprechen, wider, wie gut die Interpolation dem tatsächlichen Verlauf entspricht.

Definition 8.3.1 (Abbildungsfehler)
Der Abbildungsfehler wird bestimmt durch den euklidischen Abstand zwischen dem auf die Koordinate $\mathbf{c}$ *im Kartenraum abgebildeten Beispiel* $\mathbf{x}$ *und dem interpolierten Wert* $\hat{\mathbf{f}}(\mathbf{c})$ *an seiner Koordinate.*

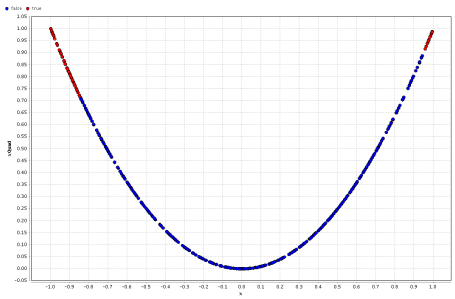


Abbildung 8.10.: Daten mit quadratischer Basisexpansion

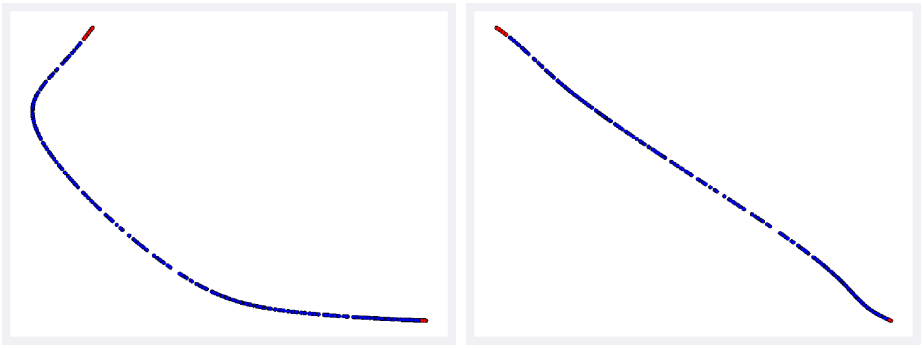


Abbildung 8.11.: Daten über Distanz im Kernraum angepasst.

Bei der Bestimmung des Abbildungsfehlers spielt natürlich die Positionierung im Kartenraum eine entscheidende Rolle. Da die üblichen Verfahren immer nur bekannte Beispiele abbilden, beschränken sie sich darauf ein Beispiel der nächsten Node zuzuweisen. Dabei interpolieren Verfahren wie die GrowingGrid SOM ja bereits implizit beim Trainieren der SOM, erreichen aber immer nur eine begrenzte Auflösung. Diese schränkt den Suchraum enorm ein und würde die Interpolatoren nicht testen. Stattdessen würde eher getestet, ob die verwendeten Beispiele, bzw die aus ihnen gebildeten Nodes, repräsentativ sind, was hier aber bereits eine Grundannahme ist.
Um die Interpolatoren selbst zu testen, kommen daher drei weitere Positioniermethoden zum Einsatz, nämlich eine Brute-Force Methode, die einfach eine hohe Anzahl an Positionen in äquidistantem Abstand testet und die mit dem geringsten Abstand auswählt, sowie zwei gerichtete Optimierungsverfahren auf Basis eines Partikelschwarms und eines evolutionären Ansatzes.

Da die Kombinationen von Interpolatoren und Positionierern den Parameterraum explodieren lässt, wird ein zweistufiges Verfahren verwendet um die beste Kombination zu bestimmen:

- Zunächst wird anhand der Datensätze Iris und BreastCancer aus dem UCI Repository untersucht, welcher Positionierer mit welchen Parametereinstellungen die beste Leistung für jeden Interpolator bietet. Dabei verwenden die Interpolatoren die Standardparameter.
- In einem zweiten Schritt werden dann mit dem jeweils besten Positionierer die besten Parameter für die Interpolatoren gesucht, indem der Abbildungsfehler minimiert wird.

8.3.1. Erster Schritt: Positionierverfahren

Verfahren	BSpline	Hardy MQ	M-BSpline	Shepard	Triangular
Evolutionär	0,113	0,090	0,111	0,197	0,081
Gittersuche	**0,095**	0,073	**0,093**	0,154	**0,075**
NächsteNode	0,215	0,138	0,499	**0,106**	0,106
Partikelschwarm	0,112	**0,065**	0,106	0,117	0,073

Tabelle 8.1.: Ergebnisse der Positionierverfahren für den Iris Datensatz

Verfahren	BSpline	Hardy MQ	M-BSpline	Shepard	Triangular
Evolutionär	0,719	0,563	0,535	0,675	0,506
Gittersuche	**0,502**	0,542	**0,501**	0,635	0,480
NächsteNode	0,911	0,734	0,999	**0,519**	0,519
Partikelschwarm	0,715	**0,538**	0,521	0,580	**0,468**

Tabelle 8.2.: Ergebnisse der Positionierverfahren für den BreastCancer Datensatz

Die Tabellen 8.1 und 8.2 zeigen die Güten der Positionierverfahren bei Verwendung der Standardparameter der Interpolatoren. Fettgedruckt wird jeweils der Wert des besten Positionierverfahrens.

Die Auswertung der Ergebnisse zeigt, dass die allgemeinen Optimierungsverfahren wie Partikelschwarmoptimierung und evolutionäre Optimierung einem einfachen Verfahren wie der Gittersuche nicht überlegen sind. Die evolutionäre Strategie ist trotz großzügig gewählter Parameter mit 1000 Individuen und einer Generationszahl von 500 der Gittersuche in keinem Fall überlegen, eine Verringerung

der Individuen oder Generationenzahl lieferte schlechtere Ergebnisse. Die Partikelschwarmoptimierung ist in drei Fällen besser, jedoch jeweils nur knapp. Da die Rechenzeit aber ungemein größer ist und die Gittersuche mit einem relativ grobmaschigen 100×100 Gitter durchgeführt wurde, erscheint es sinnvoller, diese Parameter zu erhöhen und für alle Verfahren nur die Gittersuche zu verwenden. Bei einer Erhöhung der Gitterauflösung ergeben sich die in Tabelle 8.3 gezeigten Ergebnisse. Fett gedruckte Werte kennzeichnen Kombinationen in denen das Gittersuchverfahren den geringsten Fehler verursacht.

Verfahren	100×100	200×200	400×400
BSpline	**0,095**	**0,078**	**0,072**
Hardy MQ	0,073	0,071	0,066
M-BSpline	**0,093**	**0,080**	**0,071**
Shepard	0,078	0,078	0,077
Triangular	**0,075**	**0,074**	**0,073**

Tabelle 8.3.: Ergebnisse des Gittersuchverfahrens für verschiedene Gitterauflösungen auf dem Iris Datensatz

Abbildung 8.12 gibt einen Eindruck der auftretenden Geschwindigkeitsunterschiede, die in diesem Ausmaß die effiziente Bestimmung der Parameter stark beeinträchtigen.
Für den späteren Anwendungsfall sind diese Unterschiede nicht mehr so gravierend, da dort bereits die Berechnung des Entscheidungsmodell einige Zeit in Anspruch nehmen wird. Außerdem wird auch auf eine Testmenge verzichtet, so dass im Prinzip keine Positionierer mehr gebraucht werden, da die nächste Node hinreichend gute Ergebnisse liefern sollte.

Sonderfall Shepardinterpolation

Ein Sonderfall ergibt sich durch die Shepardinterpolation, in der die nächste Node die beste Annäherung darstellt. Da es sich bei der Shepardinterpolation um eine Interpolationsfunktion handelt, die auch unter numerischer Begrenzung an den Nodes exakt den Beispielvektor zurückliefert, läuft dieses Verhalten darauf hinaus, dass einfach zum nächsten bekannten Beispiel zugeordnet wird. Dieses Verhalten ist natürlich nicht gewünscht, da es den ganzen Zweck der Interpolation hintertreibt, denn der war ja eben, die Strukturen über den gesamten Kartenraum abzubilden. Offensichtlich war die Wahl des Standardwertes für den einzigen Pa-

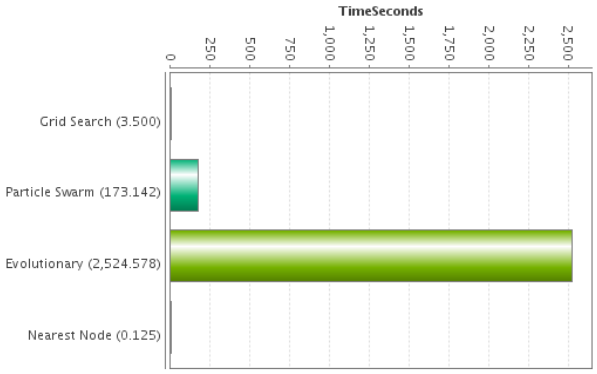


Abbildung 8.12.: Laufzeiten der gesamten Transformation auf dem Iris Datensatz mit BSpline Approximation in Sekunden

rameter unglücklich. Betrachtet man die visuellen Ergebnisse der Interpolation in Abbildung 8.13, werden die Probleme deutlich.

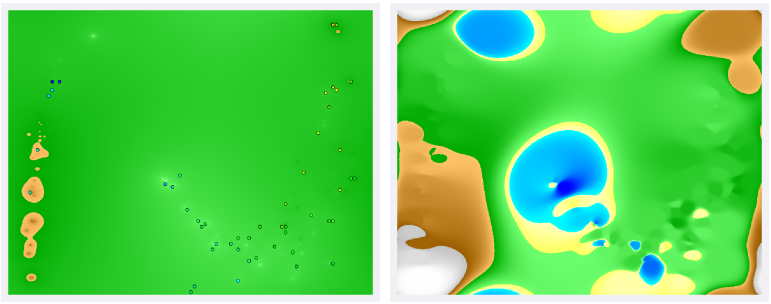


Abbildung 8.13.: Shepardinterpolation derselben SOM mit links $\mu = 1$ und rechts $\mu = 6$

In Tabelle 8.4 werden die Ergebnisse für die Shepardinterpolation auf Grundlage des visuell besseren Parameterwerts $\mu = 6$ angegeben. Diese sind nun für die optimierenden Verfahren besser als für die Zuordnung zur nächsten Node. Gleichzeitig zeigt sich hier auch wieder, dass die Partikelschwarmoptimierung zwar besser ist als die Gittersuche, der geringe Unterschied die enorm gestiegene Rechenzeit jedoch nicht ausgleicht. Daher wird im Folgenden nur noch auf die Gittersuche zurückgegriffen, die Auflösung des Gitters aber auf 200×200 gesteigert.

Datensatz	Iris	BreastCancer
Evolutionär	0,087	0,511
Gittersuche	0,078	0,483
NächsteNode	0,106	0,519
Partikelschwarm	**0,075**	**0,474**

Tabelle 8.4.: Ergebnisse für Shepardinterpolation mit $\mu = 6$

Interpolationseigenschaft

Anhand der Ergebnistabelle 8.1 für die Interpolationsverfahren lässt sich weiterhin prüfen, ob die Verfahren unter den numerischen Begrenzungen die Interpolationseigenschaft erfüllen. Dabei kann man den Fehler des nächsten-Node Positionierers beim Triangulationsverfahren als Vergleichswert benutzt. Bei diesem Verfahren ist sichergestellt, dass es die Interpolationseigenschaften erfüllt. Dadurch ist der Fehler des deterministisch arbeitenden nächsten-Node Positionierers bei allen anderen Verfahren genauso groß wie bei der Triangulation, wenn sie die Interpolationseigenschaft erfüllen.

Daraus kann man schlussfolgern, dass Hardy's Multiquadric die Interpolationseigenschaft nicht erfüllt, was vermutlich an den numerischen Instabilitäten und dem Implementierungstrick zur Invertierung der Matrix liegt. Die Approximationsverfahren verletzen erwartungsgemäß die Interpolationsbedingung noch stärker, was an dem im Vergleich zur Triangulation noch höheren Fehler erkennbar ist. Einzig das Shepardverfahren erfüllt zusätzlich die Interpolationsbedingung.

8.3.2. Zweiter Schritt: Interpolationsverfahren

Im ersten Schritt wurden die besten Parameter für die Positionierverfahren bestimmt unter der Annahme, dass dies unabhängig von den Interpolatoren erfolgen kann. Um nun die Interpolationsverfahren zu optimieren, werden ihre Parameter im Rahmen einer Gittersuche variiert und über eine wie oben beschriebene Kreuzvalidierung ausgewertet.

Leider sind die Ergebnisse stark verrauscht, es treten große Schwankungen der Performanz selbst bei kleinsten Parameteränderungen auf. Als Beispiel soll der anschauliche, weil eindimensionale Fall der BSpline Approximation dienen, auch wenn sich dieser Effekt bei allen Verfahren zeigt. In diesem Versuch wurden die horizontale und die vertikale Auflösung aneinander gekoppelt, so dass es nur noch einen Parameter gibt. Es ergibt sich Abbildung 8.14 für die kleinste Entfernung zur interpolierte Oberfläche abhängig von der Auflösung auf dem Iris Datensatz.

Die Fläche um die rote Linie gibt die Standardabweichung an, die während der 10-fachen Kreuzvalidierung aufgetreten ist.

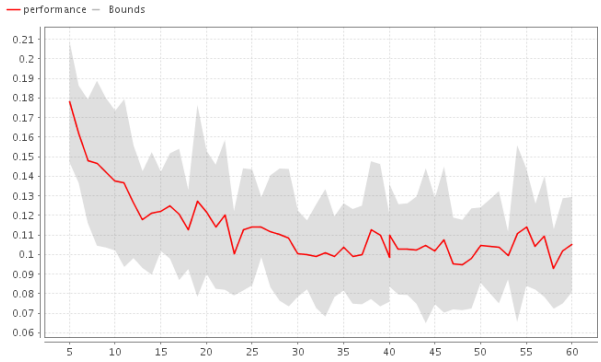


Abbildung 8.14.: Performanz mit Standardabweichung abhängig von der Auflösung der BSplines auf Iris

Man sieht zwar, dass bis zu einem gewissen Grad die Leistung besser wird, aber die Abweichungen sind zu groß um einen genauen Punkt festmachen zu können. Um diese Ergebnisse beurteilen zu können, stellt die Abbildung 8.15 die Interpolation einer Dimension bei verschiedenen Auflösungen vor. Wenn man dort das Verhalten der Punkte betrachtet, wird auch klar, warum die Leistung konstant bleibt: Während die Interpolation immer mehr den vorliegenden Datensatz auswendig lernt und überanpasst, rückt der Interpolator die Punkte immer näher zusammen. Das zeigt aber, dass der Sinn, die Struktur in den Daten über den Kartenraum auszubreiten, genau verfehlt wird. Entsprechend sollte eine relativ niedrige Auflösung gewählt werden, bei dem der Abstieg des Fehler sich gerade abschwächt.

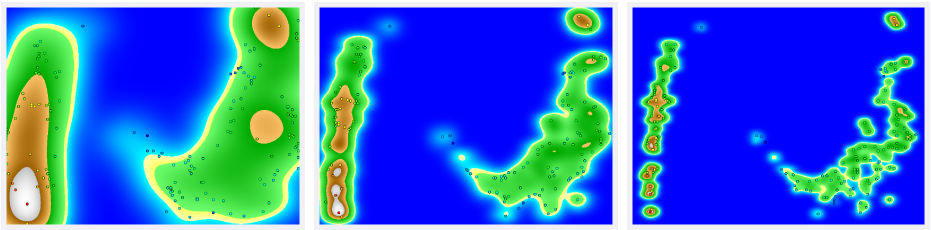


Abbildung 8.15.: Interpolation einer Dimension des Iris Datensatzes bei einer Auflösung von 10, 25 und 50

Zu überprüfen wäre jetzt, ob die besten Interpolationsparameter vom verwendeten Datensatz abhängen. Dazu wird derselbe Prozess auf dem Datensatz BreastCancer ausgeführt. Es ergibt sich ein ganz ähnlicher Verlauf für die Güte, wenn auch aufgrund des höherdimensionalen Datensatzes in anderer Größenordnung. Abbildung 8.16 zeigt den Verlauf.

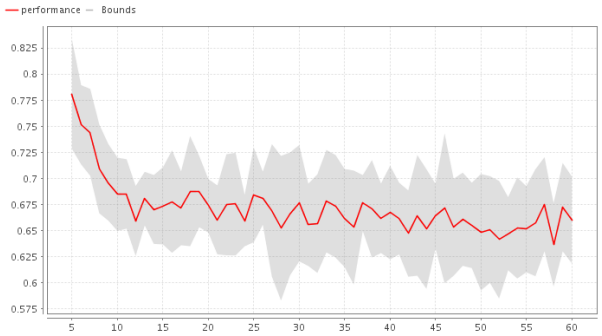


Abbildung 8.16.: Performanz mit Standardabweichung abhängig von der Auflösung der BSplines auf dem BreastCancer Datensatz

Zusammenfassung

Die folgende Tabelle fasst die jeweils beste Leistung aller Interpolationsverfahren zusammen. Bei der Interpretation der Tabelle bitte die Ausnahmen [1] und [2] beach-

Datensatz	Iris	BreastCancer	Laufzeit Iris	Laufzeit BreastCancer
Bspline	**0,056**	**0,397**	7,2	192
Hardy MQ	0,058	—[1]	1722	—[1]
M-BSpline	0,064	0,433	7,2	187
Shepard	0,059	0,418	650	4328[2]
Triangular	0,088	0,429	116	1965

Tabelle 8.5.: Ergebnisse der Interpolatoren

ten. An den Ergebnissen sieht man, dass die Approximationsverfahren nicht nur die geringsten Fehler machen, sondern auch mit Abstand am schnellsten arbeiten.

[1] Wert für diesen Datensatz unbestimmbar, da die Invertierung der entsprechenden $N \times N$ großen Matrix durch Instabilitäten scheitert

[2] Durch Rechnerwechsel Wert nicht vergleichbar

Sie empfehlen sich gleich doppelt für die weitere Verwendung. Dabei bietet die Multilevelerweiterung der BSplines zwar die etwas schlechtere Leistung, aber den subjektiv besseren Eindruck, weshalb sie im nächsten Abschnitt verwendet wird.

8.4. Vergleich der Visualisierungsleistung

Nachdem im letzten Abschnitt untersucht wurde, wie gut die Interpolatoren in der Lage sind die tatsächliche Struktur in den Daten wiederzugeben, beginnt dieser Abschnitt mit dem Vergleich der verschiedenen, hier vorgestellten Dimensionsreduktionsverfahren in Bezug auf die Problemstellung der verständlichen Datenvisualisierung. Dabei werden für die Multilevel BSpline Approximation die oben gefundenen, optimalen Parameter verwendet.

Da aber Verständlichkeit in erster Linie vom Benutzer abhängt, stellt sich hier die Schwierigkeit objektive, quantitative und vor allem messbare Größen zu finden, die die Verständlichkeit ausdrücken. Der oben beobachtete Effekt, dass der Fehler gleich bleibt, obwohl bereits eine Überanpassung stattfindet, muss dabei vermieden werden und zeigt die Schwierigkeiten ein gutes Kriterium zu finden.

Da die Dimensionsreduktion auf den Kartenraum dazu dient, das Verhalten der Modelle abzubilden, ist es wünschenswert, dass diese Reduktion dafür sorgt, dass Klassengebiete nicht willkürlich zerrissen werden. Dieser Effekt kann auftreten, wenn die Interpolationsfunktion überschwingt, oder durch Projektionsfehler der Dimensionsreduktion hervorgerufen werden. Da in dieser Testphase jedoch auf den sehr stabilen, Spline-basierten Approximationsansatz zurückgegriffen wird, ist die erste Fehlerquelle auszuschließen. Die Fehler ergeben sich also zum größten Teil durch Projektionsfehler während der Dimensionsreduktion. Man unterscheidet dabei zwei Fehlerarten (Ultsch und Herrmann 2005):

Definition 8.4.1 (Forward Projection Error)
Also Forward Projection Error, kurz FPE, bezeichnet man die Abbildung von zwei ähnlichen Beispielraumvektoren auf weit entfernte Punkte im Kartenraum.

Definition 8.4.2 (Backward Projection Error)
Als Backward Projection Error, kurz BPE, bezeichnet man die Abbildung von zwei unähnlichen Beispielraumvektoren auf benachbarte Punkte im Kartenraum.

Man sieht dabei, dass beide Fehlerarten in gleicher Weise für ein Zerreißen des Klassengebietes sorgen können, auch wenn eine solche Trennung der Klassengebiete in den Daten nicht vorhanden ist. Um diese Fehler zu messen, werden hier folgende Kriterien eingeführt:

Definition 8.4.3 (Prediction Area Count)
Der Prediction Area Count, kurz PAC, zählt die Anzahl der unabhängigen, das heißt nicht miteinander verbundenen, Flächen, in denen das Modell dieselbe Vorhersage trifft.

Dieses Kriterium zählt anschaulich die Farbflecken im Modellayer. In Abbildung 7.2 ergäbe sich also ein Wert von 3. Damit ist dieses Kriterium natürlich stark abhängig vom verwendeten Entscheidungsmodell. Da dieses jedoch konstant bleibt über der Verwendung verschiedener Dimensionsreduktions- und Interpolationsverfahren, gleicht sich das aus.

Ein weiterer unerwünschter Effekt sind stark verschlungene Entscheidungsgrenzen, also zum Beispiel Strukturen, die kammartig ineinander greifen. Da hierdurch die Grenze deutlich länger wird, muss der Benutzer mehr Informationen aufnehmen um herauszufinden, warum das Entscheidungsmodell dort eine Grenze zieht. Wünschenswert ist also eine möglichst kurze Grenze. Wieder können die Kammstrukturen natürlich im Zusammenhang in den Daten begründet sein, dann sollten sie aber für alle Methoden gleichermaßen auftreten und sich so wieder ausgleichen. Ein Kriterium, das die Länge der Grenze misst, ist gegeben durch:

Definition 8.4.4 (Border Point Count)
Gegeben sei ein Gitter über dem Kartenraum, an dessen Knotenpunkten jeweils geprüft wird, zu welcher Klasse das Entscheidungsmodell den Punkt zuordnet. Der Border Point Count, kurz BPC, zählt dann die Anzahl der Punkte des Gitters, die einen oder mehrere Punkte einer anderen Klasse in der Nachbarschaft haben.

Mittels dieser beiden Kriterien werden nun die Leistungen der verschiedenen Verfahren zur Dimensionsreduktion verglichen. Dazu wurde für jedes Verfahren eine umfangreiche Parameteroptimierung durchgeführt, deren Ergebnisse zunächst in den folgenden Tabellen 8.6 und 8.7 zusammengefasst werden. Anschließend wird eine Interpretation der Ergebnisse gegeben.

Beziehung zwischen PAC und BPC

In den Tabellen 8.6 und 8.7 wurden die besten Parametereinstellungen nicht durch eine Gewichtung der Gütemaße gewählt, sondern durch die Priorisierung des PAC vor dem relativen Border Point Count. Obwohl dies eine Verfälschung der Ergebnisse verursachen kann, ist dieses Vorgehen durch zwei Dinge gerechtfertigt. Zum einen sind zusammenhängende Klassengebiete dem Beutzer einsichtiger als Insellösungen und zum anderen scheinen PAC und BPC stark miteinander korreliert zu

Verfahren	PAC	BPC relativ	Parametereinstellungen
ESOM	3	0,050	rounds = 10, adaption = 0,8, gridSize = 132 × 132
GGSOM	2	0,012	rounds = 40, adaption =0,512, gridSize = 18 × 18
GCSOM	3	0,036	rounds =50, adaption = 0,34, gridSize = 119 × 119
DSOM	4	0,040	adaptions = 2500, startAdaptionStrength = 0,016, endAdaptionStrength = 0,016
LLE	2	0,004	k = 9
PCA	8	0,067	

Tabelle 8.6.: Ergebnisse der Dimensionsreduktionsverfahren auf dem Iris Datensatz

sein. Abbildung 8.17 illustriert die Verhältnisse zwischen PAC und BPC anhand der Ergebnisse für DSOM links und ESOM rechts. Das gleiche Verhalten zeigt sich auch auf dem BreastCancer Datensatz. Ebenfalls zeigen diese Abbildungen, dass auch hier wieder ein relativ starkes Rauschen über den Ergebnissen liegt und zahlreiche Parameterwerte zu ähnlicher Performanz führen.

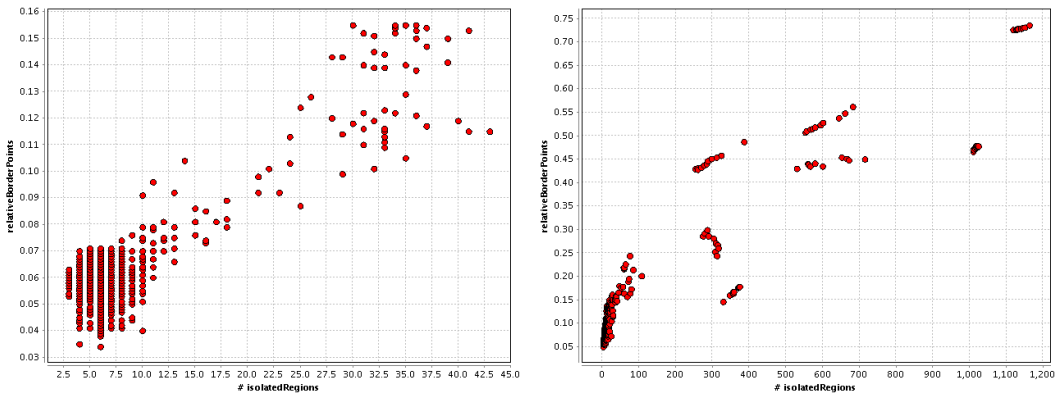


Abbildung 8.17.: Korrelation zwischen PAC und BPC auf Iris

Verfahren	PAC	BPC relativ	Parametereinstellungen
ESOM	3	0,032	rounds = 10, adaption = 0,289, gridSize = 78 × 78
GGSOM	2	0,027	rounds = 40, adaption =0,8, gridSize = 18 × 18
GCSOM	2	0,023	rounds =50, adaption = 0,26, gridSize = 71 × 71
DSOM	9	0,060	rounds = 4500, startAdaptionStrength = 0,361, endAdaptionStrength = 0,081
LLE	2	0,008	k = 21
PCA	14	0,050	

Tabelle 8.7.: Ergebnisse der Dimensionsreduktionsverfahren auf dem BreastCancer Datensatz

Sonderfälle LLE und GGSOM

Die auffälligsten Ergebnisse liefern LLE und die GGSOM. Beide entdeckten nur zwei Klassengebiete des drei-klassigen Iris Datensatzes, was Grund zur Vermutung gibt, dass es sich um degenerierte Lösungen handelt. Das gleiche Verhalten zeigt LLE auch auf dem BreastCancer Datensatz. Eine visuelle Überprüfung in Abbildung 8.18 bestätigt dies. Dieses Ergebnis zeigt aber nicht die generelle Untauglichkeit der Verfahren zur Modellvisualisierung, legt jedoch die Instabilität gegenüber den definierten Kriterien offen. Die Neigung sich den Kriterien überanzupassen macht es unmöglich, über sie zu optimieren und zu bewerten, weshalb sie in diesem Kontext nicht ohne weiteres verwendet werden können.

Subjektive Beurteilung der Ergebnisse

Um die Repräsentativität der verwendeten Performanzmaße zu prüfen, werden in dem folgenden Abschnitt die Grafiken der besten Lösungen präsentiert und bewertet. Da sich subjektive Eindrücke schlecht quantifizieren lassen, soll lediglich geprüft werden, ob die Relationen über den Performanzmaßen mit denen der subjektiven Eindrücke übereinstimmen. Da zur subjektiven Kontrolle ein Datensatz verwendet werden muss, der dem Betrachter bekannt ist, wird hier der Iris Datensatz benutzt.

Die Ergebnisse der besten Parameterwerte sind in Abbildung 8.19 dargestellt. Dabei repräsentiert der Hintergrund über die interpolierte Landkartendarstellung

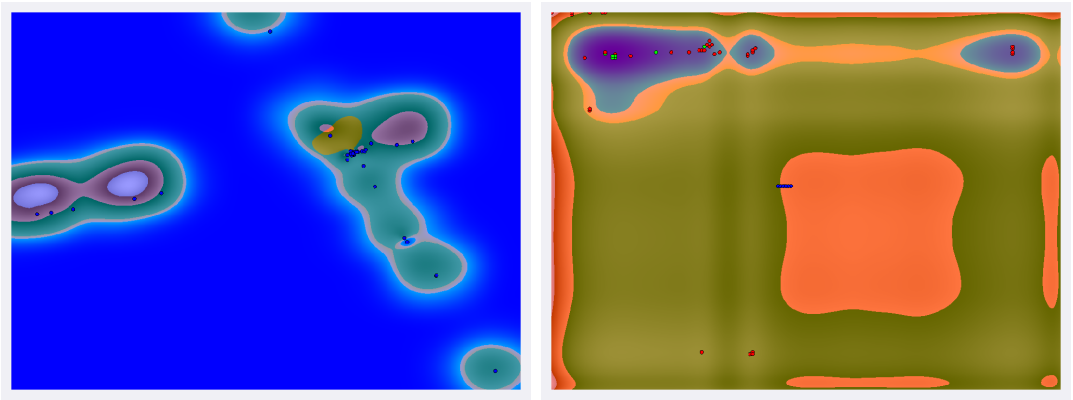


Abbildung 8.18.: Degenerierte Lösungen von Locally Linear Embedding und GrowingGrid SOM

das Attribut SepalWidth, während die farbigen Flächen die Entscheidung des Modells visualisieren. Die Punktefarbe gibt das wirkliche Label des jeweiligen Beispiels an.

Man sieht, dass die DSOM und die PCA beide ungefähr die Entfernungsrelationen erhalten, die im Datensatz auftreten. Die DSOM modeliert dieses explizit, während die PCA durch ihre lineare Natur die Entfernungsrelationen erhält. Im Gegensatz dazu sind die Daten bei ESOM und GCSOM annähernd gleichverteilt über den Kartenraum, was den visuellen Platz besser ausnutzt. Weiterhin besitzen beide Verfahren, ganz besonders aber die GCSOM, die schöneren Klassengrenzen und man erkennt sehr gut die Variabilität des im Hintergrund angezeigten Attributs.

Vor allem bei der DSOM, wie bei der PCA läßt der Verlauf der blauen Grenze darauf schließen, dass das im Hintergrund als Landkarte visualisierte Attribut wichtig für die Klassenzuordnung ist. Tatsächlich ist das nicht ganz korrekt, da der Entscheidungsbaum an einem anderen Attribut nach Blau aufsplittet, aber diese Attribute sind hochgradig korreliert. Sowohl bei der DSOM, als auch bei der PCA kommen die problematischen Regionen des Beispielraums besser und konzentrierter als bei den anderen Verfahren zur Geltung, was die Bestimmung der Schwierigkeiten bei der Klassifikation erleichtert.

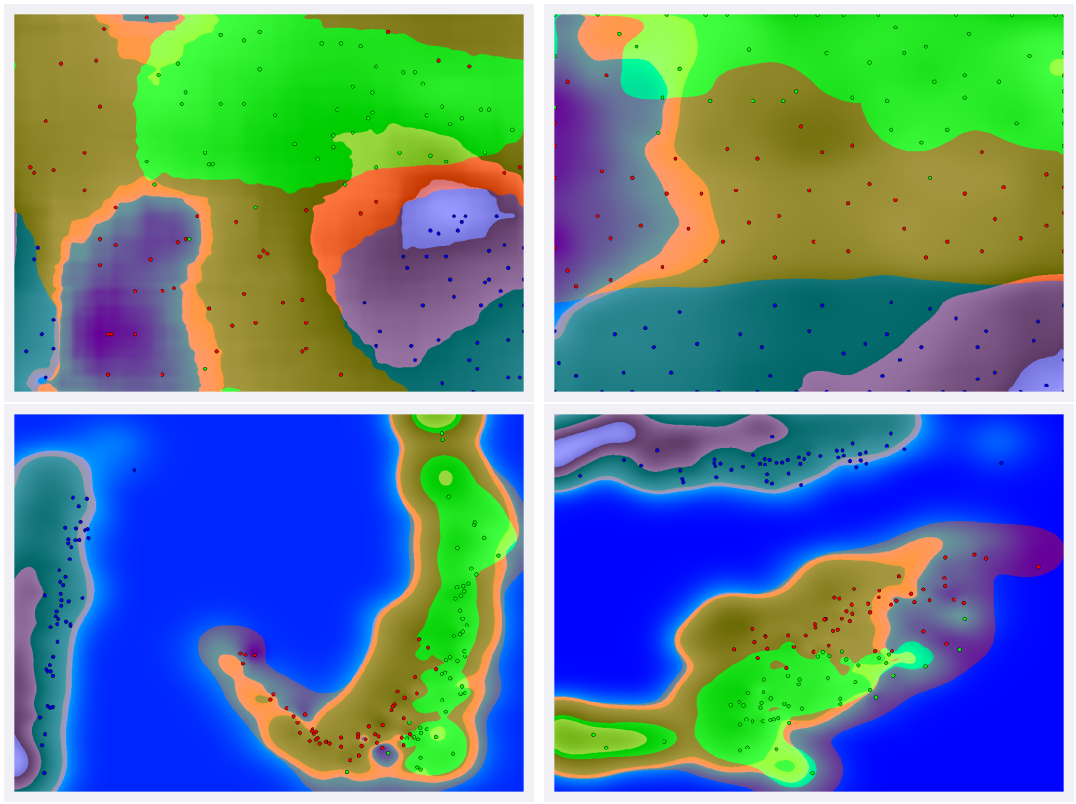


Abbildung 8.19.: Ergebnisse der Modelvisualisierung von ESOM, GCSOM, DSOM und PCA

Vergleichende Modellvisualisierung

Um nun zu überprüfen ob das Ziel, verschiedene Modelle miteinander vergleichen zu können, erreicht wird, werden die eingangs in Kapitel 2 beschriebenen Entscheidungsmodelle der linearen Diskriminanzanalyse, des Entscheidungsbaums und der SVM auf dem Iris Datensatz gelernt und mit einer DSOM visualisiert. Abbildung 8.20 zeigt links oben das visualisierte Modell eines Entscheidungsbaums. Rechts daneben wird das Modell der linearen Diskriminanzanalyse dargestellt. Die beiden Bilder darunter heben durch grafische Bearbeitung die Bereiche hervor, in denen die Ergebnisse der Verfahren unterschiedlich sind. Das Bild links zeigt einen Vergleich des Entscheidungsbaums mit der LDA, das Bild rechts daneben den Vergleich mit der SVM.

Man sieht zunächst, dass sich der Entscheidungsbaum im Gegensatz zur LDA

und SVM die blaue Klasse generalisiert, während LDA und SVM die rote Klasse bevorzugen. Daneben lassen sich schnell die Grenzen erkennen, bei denen sich die Verfahren weitestgehend einig sind. Auch der Bereich in dem die Verfahren zu unterschiedlichen Ergebnissen kommen, lässt sich so schnell identifizieren. Dies ist in diesem Beispiel genau der überlappende Bereich der beiden grün und rot markierten Klassen Iris-versicolor und Iris-virginica. Mit Hilfe der hier ausgeblendeten Hintergrundinformation kann versucht werden die Probleme zu identifizieren um Lösungsansätze zu entwickeln.

Auf dieselbe Weise lassen sich auch Modelle des gleichen Lernverfahrens vergleichen, die sich nur in der Parameterwahl unterscheiden. Dadurch kann, wie Abbildung 8.21 zeigt, dargestellt werden, wie sich eine Überanpassung auf die Modellgrenzen auswirkt.

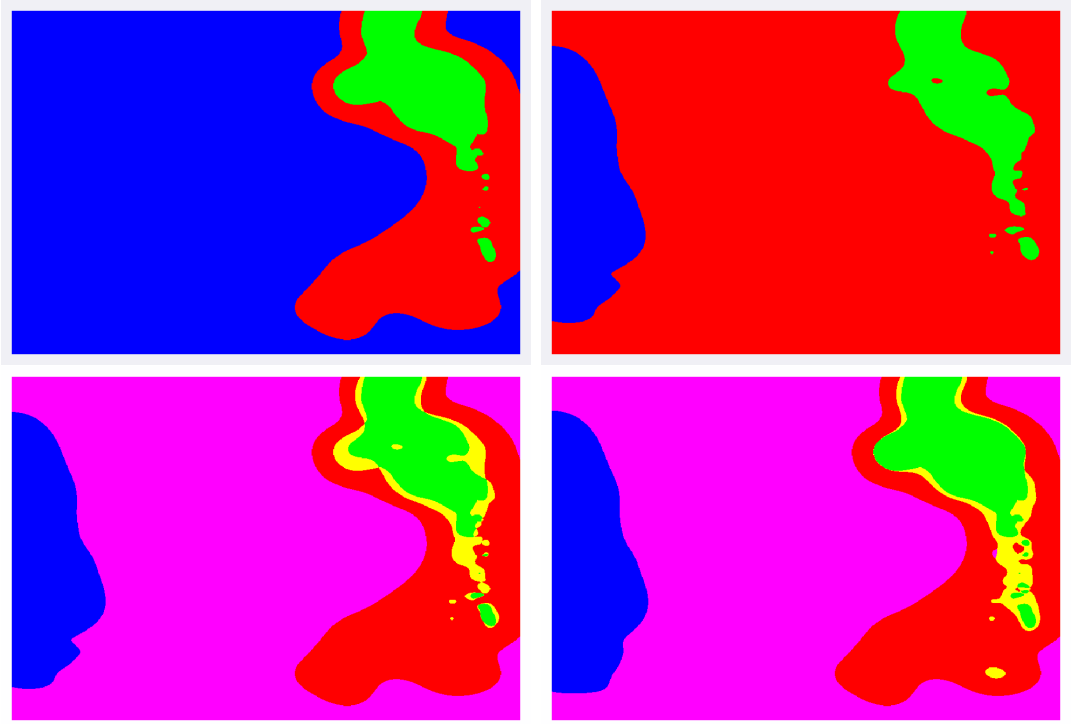


Abbildung 8.20.: Vergleich der visualisierten Modelle

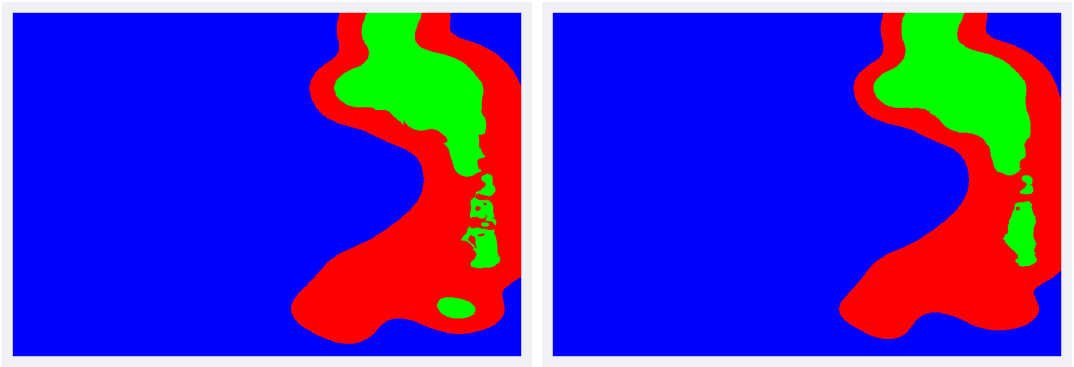


Abbildung 8.21.: Vergleich zwischen Modellen mit unterschiedlichen Parametern, links überangepasst, rechts normal

8.5. Fazit und Ausblick

Ziel dieser Arbeit war es, eine Methode zur Visualisierung von beliebigen Entscheidungsmodellen zu entwickeln. Der hier vorgestellte Ansatz über das Zusammenspiel von Dimensionsreduktion und Interpolation ist nach Wissen des Autors die erste allgemeine Methode überhaupt. In dem Sinn liefert die Arbeit eine solide theoretische und praktische Grundlage für die Integration von Modellvisualisierung in Datamining Tools, bleibt aber selbst Grundlagenforschung. Denn zum einen bieten die in dieser Arbeit abgedruckten Ergebnisse trotz ihres notwendigen Mangels an Interaktivität erste Erkenntnisse über die Entscheidungsmodelle, zum anderen fehlt aber zum Beispiel noch eine gute Heuristik für die besten Parameterkombinationen auf unbekannten Datensätzen. Denn erst eine solche Heuristik macht die Methode in der Praxis für Anwender einfach handhabbar.

Mit dieser Heuristik ließe sich die Methode jedoch auch anwenden, um während einer Präsentation interaktiv Fragen visuell zu beantworten und den Entscheidungsträgern anhand einer Mischung aus bekannten Beispielen, Hintergrundinformationen und bunten Bildern das nötige Vertrauen in die neuen Techniken zu vermitteln.

Ein weiteres Ergebnis dieser Arbeit ist die DSOM, die, auch wenn sie bei der Modellvisualisierung nicht die besten Ergebnisse erzielt, aufgrund ihrer theoretischen Eigenschaften weitere Aufmerksamkeit verdient. Da sie Daten nur über die Distanz anordnet, ist sie in der Lage numerische Koordinaten für ursprünglich nicht numerische Daten zu berechnen. Dadurch erschließt sie für diese Daten ein viel größeres Feld von Lernverfahren, die vielfach nur mit numerischen Daten umgehen können.

Aber auch hier bieten sich weitere Möglichkeiten der Optimierung, um etwa Extremsituationen zu vermeiden, wie sie in diesem Kapitel beschrieben wurden. Auch wird für die Verwendung als Vorverarbeitungsschritt eine Möglichkeit gebraucht, um zum Transformationszeitpunkt noch unbekannte Daten nachträglich passend anzuordnen.

Literaturverzeichnis

[Boissonnat und Cazals 2000] Boissonnat, Jean-Daniel und F. Cazals (2000). *Smooth Surface Reconstruction via Natural Neighbour Interpolation of Distance Functions*. In: *Proceedings of the sixteenth annual symposium on Computational geometry*, S. 223–232.

[Breiman et al. 1984] Breiman, Leo, J. H. Friedman, R. Olshen und C. J. Stone (1984). *Classification and regression trees*. Wadsworth International Group, Belmont, CA.

[Burges 1998] Burges, Christopher J. C. (1998). *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Min. Knowl. Discov., 2(2):121–167.

[Carreira-Perpinán 1997] Carreira-Perpinán, Miguel Á. (1997). *A Review of Dimension Reduction Techniques*. Technischer Bericht, University of Sheffield, Dept. of Computer Science.

[Comon 1994] Comon, Pierre (1994). *Independent component analysis, a new concept?*. Signal Process., 36(3):287–314.

[Dyn et al. 1990] Dyn, Nira, D. Levin und S. Rippa (1990). *Data Dependent Triangulations for Piecewise Linear Interpolation*. IMA J Numer Anal, 10(1):137–154.

[Fisher 1936] Fisher, R. A. (1936). *The Use of Multiple Measurements in Taxonomic Problems*. Annual Eugenics, 7:179–188.

[Fodor 2002] Fodor, Imola K. (2002). *A Survey of Dimension Reduction Techniques*. Technischer Bericht, Lawrence Livermore National Laboratory.

[Fortune 1992] Fortune, Steven (1992). *Voronoi diagrams and Delaunay triangulations*. In: *Computing in Euclidean Geometry*, S. 193–233.

[Fritzke 1991] Fritzke, Bernd (1991). *Let it grow - self-organizing feature maps with problem dependent cell structure*. In: *Artificial Neural Networks*, S. 403–408. North-Holland.

[Fritzke 1992] Fritzke, Bernd (1992). *Growing Cell Structures - a Self-Organizing Network in k Dimensions.* In: *Proceedings of International Conference on Artificial Neural Network (ICANN'1992).*

[Fritzke 1995] Fritzke, Bernd (1995). *Growing Grid - a self-organizing network with constant neighborhood range and adaption strength.* Neural Processing Letters, 2(5):9–13.

[Hardy 1971] Hardy, R.L. (1971). *Multiquadric equations of topography and other irregular surfaces.* Journal Geophys. Research, 76:1905–1915.

[Hastie et al. 2001] Hastie, T., R. Tibshirani und J. Friedman (2001). *The Elements of Statistical Learning.* Springer Verlag, Basel.

[Hoschek und Lasser 1992] Hoschek, Josef und D. Lasser (1992). *Grundlagen der geometrischen Datenverarbeitung.* B.G. Teubner Stuttgart, 2. Auflage Aufl.

[Johnston 2002] Johnston, Wesley (2002). *Information Visualization in Data Mining and Knowledge Discovery*, Kap. Model visualization, S. 223–227. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Kohonen 1981] Kohonen, Teuvo (1981). *Automatic Formation of topological maps of patterns in a self-organizing system.* In: Oja, E. und S. O., Hrsg.: *Proceedings of the 2nd scandinavian Conference on Image Analysis*, S. 214–220.

[Kohonen 1982] Kohonen, Teuvo (1982). *Self-Organizing Formation of Topologically correct Feature Maps.* Biological Cybernetics, 43:59–69.

[Kohonen 1997] Kohonen, Teuvo (1997). *Self-organizing maps.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Lee et al. 1997] Lee, S., G. Wolberg und S. Shin (1997). *Scattered data interpolation with multilevel B-splines.* Visualization and Computer Graphics, IEEE Transactions on, 3(3):228–244.

[Mardia et al. 1979] Mardia, K. V., J. T. Kent und J. M. Bibby (1979). *Multivariate analysis.* Probability and Mathematical Statistics, London: Academic Press, 1979.

[Meyer 1998a] Meyer, Bernd (1998a). *Competitive Learning of Network Diagram Layout.* In: *Proc.VL'98-1998 IEEE Symposium on Visual Languages*, S. 56–63. IEEE CS Press.

[Meyer 1998b] Meyer, Bernd (1998b). *Graph Drawing*, Bd. Volume 1547/1998 d. Reihe *Lecture Notes in Computer Science*, Kap. Self-Organizing Graphs - A Neural Network Perspective of Graph Layout, S. 246–262. Springer Berlin / Heidelberg.

[Mitchell 1997] Mitchell, Thomas M. (1997). *Machine Learning*. McGraw-Hill Higher Education.

[Quinlan 1993] Quinlan, J. Ross (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Roweis und Saul 2000] Roweis, Sam T. und L. K. Saul (2000). *Nonlinear Dimensionality Reduction by Locally Linear Embedding*. Science, 290:2323–2326.

[Saul und Roweis 2000] Saul, Lawrence K. und S. T. Roweis (2000). *An Introduction to Locally Linear Embedding*. Technischer Bericht, University College London and AT&T Labs.

[Schölkopf 2001] Schölkopf, Bernhard (2001). *Advances in Neural Information Processing Systems 13*, Kap. The Kernel Trick for Distances, S. 301–307. MIT Press.

[Schölkopf und Smola 2001] Schölkopf, Bernhard und A. J. Smola (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

[Shepard 1968] Shepard, Donald (1968). *A two-dimensional interpolation function for irregularly-spaced data*. In: *Proceedings of the 1968 23rd ACM national conference*, S. 517 – 524, New York, NY, USA. ACM.

[Thearling et al. 2002] Thearling, Kurt, B. Becker, D. DeCoste, W. D. Mawby, M. Pilote und D. Sommerfield (2002). *Information Visualization in Data Mining and Knowledge Discovery*, Kap. Visualizing data mining models, S. 205–222. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Ultsch 1995] Ultsch, Alfred (1995). *Self Organizing Neural Networks perform different from statistical k-means clustering*. Technischer Bericht, Universität Marburg.

[Ultsch 1999] Ultsch, Alfred (1999). *Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series*. In: *in Kohonen Maps*, S. 33–46. Elsevier.

[Ultsch 2003a] Ultsch, Alfred (2003a). *Maps for the Visualization of high-dimensional Data Spaces.* In: *Proceedings of the workshop on self-organizing maps*, S. 225–230.

[Ultsch 2003b] Ultsch, Alfred (2003b). *U*-Matrix: a Tool to visualize Clusters in high dimensional Data.* Technischer Bericht, Dept. of Mathematics and Computer Science, University of Marburg.

[Ultsch und Herrmann 2005] Ultsch, Alfred und L. Herrmann (2005). *The Architecture of Emergent Self-Organizing Maps to Reduce Projection Errors.* In: *ESANN'2005 proceedings - European Symposium on Artificial Neural Networks Bruges.*