
GMD Research Series

GMD –
Forschungszentrum
Informationstechnik
GmbH

Robin Höns

# Erkennung von Verkehrsschwerpunkten in Mobilfunknetzen mit Hilfe des GTM-Algorithmus

N° 6/2001

GMD – Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin
Germany
Telefon +49 -2241 -14 -0
Telefax +49 -2241 -14 -2618
http://www.gmd.de



**Die vorliegende Veröffentlichung entstand im/**
**The present publication was prepared within:**
Institut für Autonome intelligente Systeme (AiS)
Institute for Autonomous intelligent Systems
http://ais.gmd.de/

**Anschrift des Verfassers/Address of the author:**
Robin Höns
Königswinterer Straße 299
D-53227 Bonn
E-mail: robin.hoens@gmx.de

**Die Deutsche Bibliothek - CIP-Einheitsaufnahme:**

**Höns, Robin:**
Erkennung von Verkehrsschwerpunkten in Mobilfunknetzen mit Hilfe des GTM-Algorithmus / Robin Höns. GMD — Forschungszentrum Informationstechnik GmbH. - Sankt Augustin :
GMD — Forschungszentrum Informationstechnik, 2001
(GMD Research Series ; 2001, No. 6)
Zugl.: Bonn, Univ., Diplomarbeit, 2000
ISBN 3-88457-389-6

**ISSN 1435-2699**
**ISBN 3-88457-389-6**

# Kurzfassung

Während eines Telefonats in einem GSM-Mobilfunknetz werden die Empfangspegel der umliegenden Stationen gemessen und an den Server gemeldet. Der GSM-Standard ermöglicht es, diese Feldstärkedaten für alle Gespräche in einer Zelle an der $A_{bis}$-Schnittstelle gesammelt aufzuzeichnen. Dies induziert eine topologieerhaltende Abbildung der Erdoberfläche in den hochdimensionalen Raum der Empfangspegel. Das Bild dieser Abbildung ist eine zweidimensionale Mannigfaltigkeit, die an den Verkehrsschwerpunkten konzentriert ist.

Die vorliegende Arbeit hat zum Ziel, diese Mannigfaltigkeit vermittels neuronaler Karten nachzubilden und so die Verkehrsschwerpunkte zu erkennen. Als Verfahren wurde der wahrscheinlichkeitstheoretisch fundierte GTM-Algorithmus gewählt.

Es wird die Implementierung des GTM-Modells in C++ beschrieben, und Methoden zum Bayesschen Lernen, zum Schätzen der Dimension der Daten, zur Initialisierung, zur Behandlung von Overfitting und zur Visualisierung der Karte werden beleuchtet.

Des weiteren werden problemspezifische Anpassungen entwickelt, um Dämpfungseffekte zu modellieren und um unvollständige Datensätze nach verschiedenen Modellannahmen zu behandeln. Schließlich werden diverse Gütemaße untersucht, um den Lernerfolg in bezug auf Topologieerhaltung und Clustertrennung zu quantifizieren.

Die Ergebnisse des Verfahrens für echte $A_{bis}$-Meßdaten werden dargestellt und diskutiert.

**Schlagwörter:** Mobilfunk, Verkehrsdichte, neuronale Karten, GTM, fehlende Daten, effektive Dimension, Topologieerhaltung, Hauptkomponentenanalyse

Die vorliegende Arbeit entstand im Rahmen meiner Diplomarbeit am Institut für Autonome Intelligente System der GMD sowie am Institut für Informatik der Rheinischen Friedrich-Wilhelms-Universität Bonn. Sie entstammt einer Kooperation mit T-Mobil in Zusammenhang mit dem Forschungsprojekt „Validierung der Ortszuordnung von $A_{bis}$-Daten“.

# Abstract

During a call in a GSM mobile phone network the reception levels of the surrounding stations are measured and reported to the server. The GSM standard provides the $A_{bis}$ interface on which this reception level data can be recorded for all the calls within a cell. This induces a topology-preserving mapping of the earth's surface into the high-dimensional space of reception levels. The image of this mapping is a two-dimensional manifold which is concentrated at the traffic hot spots.

This report aims at reproducing this manifold by means of neural maps, thus recognizing the traffic hot spots. The method chosen is the GTM algorithm which has a basis in probability theory.

The GTM model has been implemented in C++ and trained with $A_{bis}$ data. Methods for bayesian learning, for estimating the dimension of the data, for initialization, for addressing overfitting and for visualization of the map are regarded.

Furthermore, problem specific adaptations are developed. They model reception level decay and handle incomplete data according to various model assumptions. Finally, diverse quality measures are surveyed for quantifying the learning success with regard to topology preservation and cluster separation.

The results of the method for real world data are described and discussed.

**Keywords:** mobile phone, traffic density, neural maps, GTM, missing data, effective dimension, topology preservation, principal component analysis

# Inhaltsverzeichnis

# 1 Einleitung

Aufgrund der seit Jahren ansteigenden Nachfrage müssen die Mobilfunknetze kontinuierlich ausgebaut werden, um eine flächendeckende und zuverlässige Versorgung zu gewährleisten.

Dabei muß die Netzstruktur genau geplant werden, da die Ressourcen – etwa die Standorte für die Antennen und die Frequenzen im Funkband – begrenzt sind. Neue Sender müssen also sehr sorgfältig plaziert werden.

Aus diesem Grunde ist es von enormem Wert, die Verkehrsdichte eines Mobilfunknetzes abschätzen zu können. Wenn beispielsweise ein Sender sehr stark belastet ist, ist es wichtig, die Verkehrsschwerpunkte (Hotspots) der Zelle zu kennen, um sie etwa durch einen neuen Sender abzudecken und dadurch die Zelle zu entlasten. Aber auch bei der Planung eines neuen Netzes, etwa nach dem neuen *Universal Mobile Telecommunications Standard (UMTS)*, ist es sehr nützlich, die Verkehrsschwerpunkte zu kennen.

Leider ist es nicht ohne weiteres möglich, den Standort eines Mobiltelefons mit ausreichender Genauigkeit zu bestimmen. Der Mobilfunkstandard GSM (Global Standard for Mobile Communications) sieht jedoch vor, daß jedes Handy in einer Zelle dem Sender die Feldstärkewerte der umliegenden Antennen meldet. Dort können die gesammelten Werte der ganzen Zelle aufgezeichnet werden. Die Universität Bonn und T-Mobil betreiben gemeinsam ein Projekt, welches zum Ziel hat, anhand dieser Daten die Verkehrsdichteverteilung in einer Zelle zu bestimmen [Anl98].

Im Rahmen dieses Projektes wurde 1997 der Ansatz begonnen, die Verkehrsschwerpunkte einer Zelle mit Hilfe von selbstorganisierenden Karten zu finden [Qua97]. Dem liegt das folgende Modell zugrunde: Die Feldstärkedaten der umliegenden Sender, die bei jedem Gespräch übermittelt werden, spannen einen hochdimensionalen Vektorraum (den *Pegelraum*) auf. In diesem Raum liegen die Daten auf einer zweidimensionalen Mannigfaltigkeit, da die gemessenen Werte im wesentlichen von den zwei Ortskoordinaten auf der Erdoberfläche abhängen. Diese zweidimensionale Mannigfaltigkeit soll auf eine selbstorganisierende Karte abgebildet werden, auf der sich dann die Häufungspunkte im Pegelraum abzeichnen.

Die resultierende Karte ist ein gemäß der Verkehrsdichte verzerrtes, jedoch topologieerhaltendes Abbild der Zelle. Dieses Verfahren hat den Vorteil, nicht von den sehr ungenauen Feldstärkekarten abhängig zu sein. Dafür enthält die Karte allerdings auch keine Ortsinformationen, d. h. die Position eines gefundenen Verkehrsschwerpunkts läßt sich nicht ohne weiteres ablesen. Da jedoch die Feldstärkewerte bekannt sind, ist eine anschließende Ortszuordnung mit dem Tool *LocAbis* [Hei00] möglich.

Zur Erzeugung topologieerhaltender Karten wurden in der Arbeit [Qua97] die Algorithmen SOM (Self-Organizing Feature Map; [Koh82]) und GTM (Generative Topographic Mapping; [BSW96]) angewandt. Die Ergebnisse waren vielversprechend, jedoch noch nicht optimal (von fünf Hotspots in einer Zelle wurden drei erkannt).

Inzwischen hat sich insbesondere der GTM-Algorithmus – der damals ganz neu war – weiterentwickelt. In [BSW98] und [Sve98] werden verschiedene Varianten entwickelt, etwa zur Behandlung unvollständiger Daten (siehe Abschnitt 5.1) oder ein semilineares Modell (siehe Abschnitt 5.4).

Die vorliegende Arbeit hat zum Ziel, diese neuen Erkenntnisse zu verwerten und das GTM-Modell in Hinblick auf das Hotspot-Problem zu erweitern. Außerdem soll der GTM-Algorithmus, der bislang nur als Matlab-Skript vorlag, in C++ implementiert werden.

Da GTM ein unüberwachtes, selbstorganisierendes Lernverfahren ist, ist es schwierig, den Lernerfolg zu beurteilen. Also sollen schließlich auch einige Gütemaße ausprobiert werden, um die Korrektheit einer Karte zu quantifizieren.

Die Arbeit ist wie folgt aufgebaut: Zunächst wird in Kapitel 2 die Problemstellung und das Ziel der Arbeit genauer umrissen. In Kapitel 3 werden die technischen Grundlagen des Mobilfunkstandards GSM und die gegebenen Daten beschrieben. Es folgt in Kapitel 4 die detaillierte Darstellung des GTM-Algorithmus. Kapitel 5 enthält die problemspezifischen Erweiterungen, die untersuchten Gütemaße und die Vorverarbeitung der Daten. Anschließend wird in Kapitel 6 auf die Implementierung des GTM-Verfahrens eingegangen. Die gegebenen Testdaten und die Ergebnisse finden sich schließlich in Kapitel 7, dem eine kurze Zusammenfassung folgt.

Die Anhänge enthalten die Klassenstruktur und Dateiformate der GTM-Software (Anhang A) und die detaillierten Daten und Parameter der im Ergebniskapitel dargestellten Karten (Anhang B).

# 2 Problemstellung

## 2.1 Verkehrsschwerpunkte

Um ein Mobilfunknetz optimal planen zu können, sollte der Betreiber möglichst genaue Informationen über die Verkehrsdichte des Gebietes sammeln. Wenn beispielsweise eine Zelle überlastet ist und geteilt werden soll, sollte man die neue Antenne so plazieren und ausrichten, daß sich die Verkehrsdichte gleichmäßig auf die beiden Sender verteilt – und nicht so, daß alle Verkehrsschwerpunkte in derselben Zelle bleiben, so daß diese überlastet bleibt.

Bislang war es nicht möglich, die Verkehrsdichteverteilung innerhalb einer Zelle zu bestimmen. Der GSM-Mobilfunkstandard erlaubt es jedoch, von allen Gesprächen, die innerhalb einer Zelle stattfinden, die Pegelwerte der umliegenden Stationen gesammelt aufzuzeichnen.

Bildet man diese gesammelten Daten von der Zelle in den hochdimensionalen Vektorraum der Pegelwerte ab, so wird sich dort unter Vernachlässigung des Rauschens die Zelle als zweidimensionale Mannigfaltigkeit wiederfinden.

Diesem Modell liegt die Grundannahme der *Topologieerhaltung* zugrunde. Sie besagt, daß im Ortsraum nahe beieinander stattfindende Gespräche sehr ähnliche Pegeldaten aufweisen. Da die Pegel durch Dämpfung, Beugung und Fading-Effekte verrauscht sind, ist diese Annahme eine Idealisierung. So kann es etwa beim Betreten eines Hauses zu starken Sprüngen in der Feldstärke kommen.

In der Arbeit [Qua97] wurde der Versuch begonnen, die Verkehrsschwerpunkte auf der Mannigfaltigkeit im Pegelraum zu suchen. Der alternative Ansatz besteht darin, zuerst jedem Gespräch einen Ort in der Zelle zuzuordnen und dann dort Verkehrsdichtekarten zu erzeugen. Die Ortszuordnung ist jedoch aufwendig und erfordert genaue, möglichst flächendeckende Feldstärkekarten. Die Suche nach Schwerpunkten im Pegelraum ist nicht von solchen Karten abhängig. Die Ortszuordnung gefundener Hotspots kann anschließend stattfinden und ist dann wesentlich effizienter als eine Lokalisierung jedes einzelnen Gespräches.

## 2.2 Selbstorganisierende neuronale Karten

Zur Ermittlung der Verkehrsdichte im Pegelraum wurden in [Qua97] *topologieerhaltende, selbstorganisierende neuronale Karten* gewählt. Diese Verfahren bilden die zweidimensio-

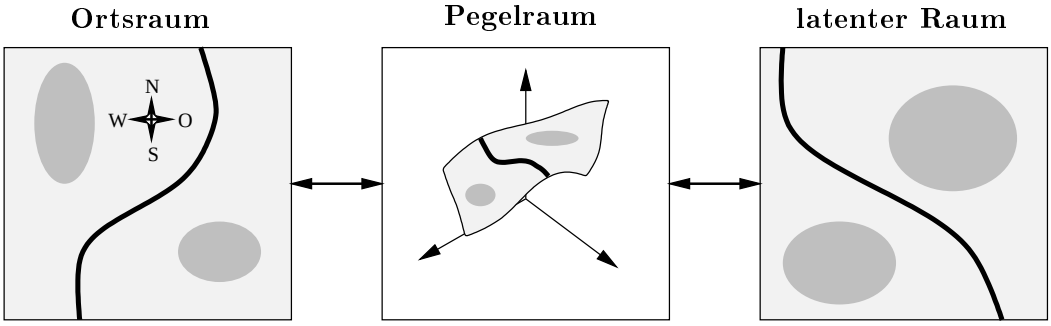



Abbildung 2.1: Zusammenhang zwischen Ortsraum (Zelle), hochdimensionalem Pegelraum und latentem Raum (zweidimensionaler Karte). Die Karte ist verzerrt gemäß der Verkehrsdichte; Hotspots werden vergrößert.

nale Mannigfaltigkeit im hochdimensionalen Pegelraum auf eine zweidimensionale Karte (den *latenten Raum*) ab. Dieser latente Raum ist nicht zu verwechseln mit dem Ortsraum (der Zelle). Er hat zwar dieselbe Dimension, ist jedoch verzerrt gemäß der Dichte der Daten, so daß Hotspots vergrößert werden (siehe Abb. 2.1). Zudem kann er auch gespiegelt oder gekippt sein. Die Topologie der Zelle, so etwa die Lage der Verkehrsschwerpunkte zueinander, bleibt jedoch erhalten.

Benutzt wurden die Algorithmen SOM (Self-Organizing Feature Map) [Koh82] – der Klassiker – und GTM (Generative Topographic Mapping) [BSW96] – ein damals ganz neuer, wahrscheinlichkeitstheoretisch fundierter Abkömmling.

Der SOM-Algorithmus bildet ein regelmäßiges Gitter von Neuronen in einen hochdimensionalen Raum ab. Die gegebenen Daten werden trainiert, indem für einen Datenpunkt das am nächsten liegende Neuron der Karte in seine Richtung gezogen wird. Seine Nachbarn auf der Karte werden ebenfalls in Richtung des Datenvektors gezogen, jedoch in abgeschwächtem Maße. Die Karte wird unüberwacht trainiert, das heißt, es wird keine Lösung vorgegeben, die reproduziert werden muß, sondern die Karte paßt sich den Daten selbständig an. Deswegen sagt man, die Karte *organisiert sich selbst*.

Das GTM-Verfahren baut auf SOM auf, enthält jedoch zusätzlich ein Wahrscheinlichkeitsdichtemodell im Datenraum. Das Rauschen der Daten wird mitmodelliert. Dies hat zur Folge, daß für jede Datenmenge eine Likelihood angegeben werden kann. Diese wird durch den Lernalgorithmus maximiert und zeigt so den Lernfortschitt an. Die SOM enthält keine solche Zielfunktion; somit ist eine Überprüfung des Lernerfolgs (etwa als Abbruchbedingung) nicht möglich. Darüber hinaus hat GTM den Vorteil, daß die Konvergenz des Lernverfahrens bewiesen ist und daß durch die theoretische Fundierung die Wahl der Parameter einfacher vonstatten geht, wohingegen die Parameter einer SOM willkürlich gewählt werden müssen.

Schließlich ist bei einem SOM-Modell die Abbildung des latenten Raumes in den Datenraum nur an den Positionen der Neuronen wohldefiniert. Zwischen den Neuronen muß

interpoliert werden. Beim GTM-Verfahren wird der latente Raum kontinuierlich in den Datenraum abgebildet, so daß die gesamte Mannigfaltigkeit wohldefiniert ist.

## 2.3 Ziele der Arbeit

Diese Diplomarbeit soll die Arbeit, die in [Qua97] geleistet wurde, wieder aufgreifen und versuchen, das Ergebnis von damals zu verbessern. Aufgrund der obengenannten Vorteile wurde der Schwerpunkt dabei auf das GTM-Verfahren gelegt.

Die Untersuchung hat die folgenden Hauptziele:

- Im Gegensatz zum SOM-Algorithmus, der in C implementiert frei verfügbar ist, steht GTM nur als Matlab-Skript zur Verfügung. Dies ist leider sehr speicheraufwendig und langsam. Ein Ziel dieser Arbeit ist, den GTM-Algorithmus in C++ zu programmieren, um ihn auf unsere großen Datenmengen anwenden zu können.
- Das GTM-Verfahren wurde inzwischen weiter untersucht und ausgebaut [BSW98, Sve98]; Varianten, die dabei entwickelt wurden, sind für unser Problem sehr interessant. Insbesondere die Behandlung fehlender Komponenten und die Modellierung von Dämpfungseffekten läßt sich so in das Verfahren aufnehmen.
- Der Vergleich trainierter Karten mit verschiedenen Parametersätzen und Modellvarianten ist nicht trivial. Für selbstorganisierende Karten existieren jedoch Gütemaße, die den Lernerfolg quantifizieren. Es soll untersucht werden, wie ein solches Maß auf das Problem angepaßt werden kann, und ob daraus ein geeignetes Kriterium entsteht.

Zur Evaluierung des Verfahrens stehen einerseits die Meßdaten von Bergheim 1997 zur Verfügung, die auch in [Qua97] untersucht wurden. Dabei wurden fünf Hotspots in einer Zelle erzeugt, die zu trennen und zu erkennen Aufgabe der Arbeit war. Damals wurden von diesen fünf Verkehrsschwerpunkten drei aufgefunden.

Zum anderen gibt es die Meßdaten von Euskirchen 2000, bei denen in zwei Zellen an je einem Dutzend Punkten telefoniert wurde. Diese bilden zwar keine Hotspots – die durch diese Gespräche erzeugten Daten machen nur ca. 2 % des gesamten Verkehrs aus –, aber an der Lage der Punkte auf der Karte kann man doch die Korrektheit der Topologie überprüfen. Also gilt es, anhand dieser Daten die Topologieerhaltung zu testen und auch zu untersuchen, ob es Maße gibt, die dies quantifizieren können, um den Lernerfolg verschiedener Karten zu vergleichen.

Die übrigen Meßaktionen (Bergheim/Bedburg 1998 und Darmstadt 1999) sind aufgrund mangelnder Hotspots und einer zu geringen Anzahl selbsterzeugter Gespräche für diese Arbeit nicht brauchbar.

# 3 Die gegebenen Daten

## 3.1 Der GSM-Standard

GSM [MP92] ist der europäische Mobilfunk-Standard. GSM stand zunächst für *Groupe Spécial Mobile* (die Arbeitsgruppe bei CEPT), dann für den von ihr entwickelten Standard, den *Global Standard for Mobile Communications*.

Im folgenden beschreibe ich die Teile des GSM-Standards, die für diese Arbeit von Bedeutung sind; Details finden sich in [MP92].

### 3.1.1 Antennen und Zellen

GSM ist ein zelluläres Netz. Jede Zelle des Netzes wird von einem Server (BTS; Base Transceiver Station) bedient, der über einen oder mehrere Kanäle mit den Mobiltelefonen (MS; Mobilstationen) kommunizieren. Die „Luftschnittstelle“ zwischen BTS und MS heißt $U_m$-Schnittstelle.

Die BTS ist hauptsächlich für den Funkverkehr mit den Mobilstationen zuständig. Mehrere BTS (bis zu einigen Dutzend) werden von einem BSC (Base Station Controller) gesteuert. Der BSC ist mit seinen BTS über die $A_{bis}$-Schnittstelle verbunden.

### 3.1.2 Frequenzen, Kanäle und Links

Das Funkband besteht aus 124 Frequenzen, auf denen gesendet werden kann. Im einfachen Fall ist jedem Kanal eine genaue Frequenz zugeordnet; neuerdings wird jedoch zunehmend „Frequency Hopping“ verwendet, d. h. die Kanäle verteilen sich über die verschiedenen Frequenzen, die der BTS zugeordnet sind. Dadurch sollen die Störungen auf einer Frequenz möglichst gleichmäßig auf die verschiedenen Gespräche verteilt werden, um die maximale Bitfehlerrate zu vermindern. Diese Technik ist jedoch für die vorliegende Arbeit nicht relevant.

An einem Sendemast befinden sich üblicherweise entweder eine omnidirektionale Antenne (Rundstrahler) oder bis zu drei direktionale (gerichtete) Antennen, die in verschiedene Richtungen weisen. Durch die Verwendung gerichteter Antennen kann die Gesprächskapazität des Netzes stark erweitert werden.

Jedem Sendemast sind mehrere *Links* zugeordnet. Diese sind durchnumeriert und entsprechen den verschiedenen Kanälen und damit auch, sofern kein Frequency Hopping

verwendet wird, den Frequenzen. Jeder Link ist einer festen Antenne zugeordnet, aber die verschiedenen Links eines Sendemastes können auch zu verschiedenen Antennen gehören, die sich an diesem Mast befinden.

Als Beispiel kann eine von uns betrachtete BTS die Frequenzen 46, 88 und 17 belegen, die den Links 5, 6 und 7 der zugehörigen Sendeanlage entsprechen. Einer der Links ist der BCCH (Broadcast Control CHannel), die anderen sind die TCHs (Traffic CHannels). Der BCCH ist der Hauptkanal der Antenne. Über ihn wird regelmäßig der Identifikationscode (BSIC; Base Station ID Code) der Station gesendet. Auch wenn nicht telefoniert wird, steht eine Mobilstation ständig mit ihm in Kontakt. Deshalb gelten für den BCCH einige Besonderheiten. So ist der BCCH beispielsweise vom Frequency Hopping ausgenommen ([MP92], S. 224).

Die BSIC ist keineswegs eindeutig einer Station zugeordnet. In ganz Deutschland werden nur acht verschiedene Werte verwendet (30 bis 37) – zusammen mit der Frequenznummer des BCCH ergibt sich jedoch eine Identifizierung der BTS, die in einer Region eindeutig sein sollte (es aber nicht immer ist).

### 3.1.3 Time Slots

Um die Kapazität des Netzes weiter zu erhöhen, trennt der Standard die Frequenzkanäle in acht Time Slots (Zeitschlitze) auf. Dies geschieht über ein Zeitmultiplexverfahren. Jeder Time Slot dauert $^{15}/_{26}$ ms. Somit dauert eine Periode von acht Time Slots etwa 4,5 ms. Auf den TCHs stehen diese acht Zeitschlitze für Gespräche zur Verfügung, auf dem BCCH ist einer für Broadcasts (z. B. Informationen zur Synchronisation oder die BSIC der Station) reserviert ([MP92], S. 209).

Damit ist jedes Gespräch eindeutig einer Kombination von Link und Time Slot zugeordnet.

### 3.1.4 Timing Advance

Da das Gespräch den ihm zugewiesenen Time Slot genau treffen muß und nicht die benachbarten stören darf, ist die zeitliche Abstimmung von entscheidender Bedeutung. Das bedeutet auch, daß die Signallaufzeiten zwischen Mobilstation und Server betrachtet werden müssen.

Eine Mobilstation, die weit vom Server entfernt ist, muß ihre Signale früher senden, um ihren Zeitschlitz zu treffen, als eine nahe. Deshalb mißt die BTS die Signallaufzeit (den *Timing Advance*) zur MS. Die Genauigkeit der Messung ist 1,85 $\mu$s, was bei Lichtgeschwindigkeit einem Abstand von 554 m entspricht.

Somit kann mit Hilfe des Timing Advance der Gesprächsort auf eine Kreisscheibe um den Server beschränkt werden (siehe Abbildung 3.1). Die Breite einer Scheibe, die einem Timing Advance-Wert entspricht, beträgt 554 Meter. Allerdings wird der Timing Advance nur mit einer Genauigkeit von $\pm 1$ ermittelt.

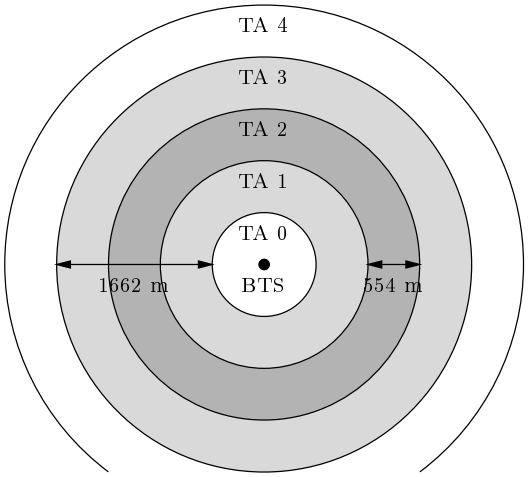



Abbildung 3.1: Mit Hilfe des Timing Advance kann man den Abstand eines Handys von der BTS auf 1662 Meter genau bestimmen (hier für einen gemeldeten Timing Advance von 2).

### 3.1.5 Power Control

Um die Akkus der Mobilstationen zu schonen, vor allem aber, um das Frequenzspektrum effizienter zu nutzen ([MP92], S. 342), gibt es sowohl für die BTS als auch für die Mobilstation die Möglichkeit, die Sendeleistung zu drosseln.

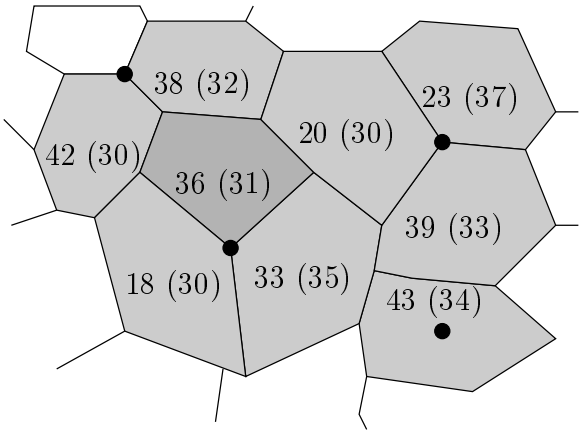
Wenn die empfangene Feldstärke einen bestimmten Wert übersteigt, nimmt der Sender seine Sendeleistung langsam zurück. Zur Berechnung der optimalen Sendeleistung wird auch die Qualität des empfangenen Signals berücksichtigt.

Genaueres kann in [MP92] nachgelesen werden.

### 3.1.6 Nachbarn und Handover

Zu Beginn eines Gespräches sendet die BTS der Mobilstation eine Liste von Frequenzen. Diese Frequenzen sind den BCCHs von Nachbarsendern zugeordnet. Die Liste heißt deswegen die „Nachbarschaftsliste“; sie ist nach Frequenznummern aufsteigend sortiert. Üblich sind zwischen sechs und sechzehn Nachbarn. Die Nachbarn sind in der Liste von Null beginnend durchnumeriert; die Nummer eines Nachbarn in der Nachbarschaftsliste heißt BAI (BCCH Allocation Index).

Das Mobiltelefon muß während des Gesprächs die Frequenzen der Nachbarn überwachen. Dank des Zeitmultiplex-Verfahrens hat es dazu in der Zeit Gelegenheit, in der es nicht mit seinem Server kommuniziert. Es versucht, sich mit den Nachbarn zu synchronisieren, zumal es jederzeit zu einem schnellen Handover bereit sein muß ([MP92], S. 333). Dabei mißt das Handy die Feldstärken der Nachbarn und versucht, deren BSICs, die ja regelmäßig gesendet werden, zu dekodieren. Wegen dieser Feldstärkemessung muß der BCCH




Nachbarschaftsliste:

| BAI | BCCH | BSIC |
|---|---|---|
| 00 | 18 | 30 |
| 01 | 20 | 30 |
| 02 | 23 | 37 |
| 03 | 33 | 35 |
| 04 | 38 | 32 |
| 05 | 39 | 33 |
| 06 | 42 | 30 |
| 07 | 43 | 34 |

Abbildung 3.2: Zellstruktur im GSM-Netz. Die schwarzen Kreise stellen die Sendemasten dar. Die hellgrauen Zellen bilden die Nachbarschaft der dunkelgrauen Zelle. Jede hat ihren BCCH auf dem angegebenen Kanal; die Zahlen in Klammern sind die BSICs. Die Nachbarschaftsliste enthält die Frequenzen, mit denen die Mobilstation sich zu synchronisieren versuchen wird.

des Nachbarn kontinuierlich ein gleichbleibend starkes Signal liefern ([MP92], S. 333). Alle 480 ms meldet die Mobilstation die gemessenen Pegelwerte und BSICs von bis zu sechs Nachbarn an seine BTS.

Zu beachten ist, daß es für das Mobiltelefon unerheblich ist, welche BSIC es dekodiert. Es kann vorkommen, daß auf der überwachten Frequenz eine andere BTS empfangen wird als die, die in der Nachbarschaftsliste steht. Das Handy meldet jedoch trotzdem die ermittelte BSIC hoch.

Der gemessene Pegelwert (RxLev) wird logarithmisch auf einer Skala zwischen 0 und 63 angegeben und kann nach dem Schlüssel in Tabelle 3.1 in dBm (Dezibel-Milliwatt; [Whi91]) umgerechnet werden.

Wenn festgestellt wird, daß ein Nachbar eine hohe Feldstärke aufweist, dann kann ein Handover in Betracht gezogen werden. Dies bedeutet ein Herüberreichen des Gesprächs

| RxLev | Empfangspegel (dBm) |
|---|---|
| 0 | weniger als -110 |
| 1 | -110 bis -109 |
| 2 | -109 bis -108 |
| ⋮ | ⋮ |
| 62 | -49 bis -48 |
| 63 | größer als -48 |

Tabelle 3.1: Wertebereich der Empfangspegel [Qua97]

```
7 6  8:24:54:979: 29,33,00,01,07,00,10,36,05,19,35,02,29,36,04,19,32,07,15,99,99,99,99,99,99
5 2  8:24:55:213: 34,48,00,00,06,02,18,33,03,30,31,04,26,35,02,36,32,07,34,36,05,38,31,01,31
5 5  8:24:55:394: 24,35,00,00,08,00,10,33,11,28,33,03,19,32,07,27,36,05,21,31,01,23,30,10,13
6 5  8:24:55:395: 25,30,00,00,07,00,10,30,08,30,30,10,28,36,04,21,33,03,22,99,99,99,99,99,99
7 6  8:24:55:459: 27,32,00,00,07,00,10,36,05,21,35,02,32,33,03,23,36,04,18,32,07,16,99,99,99
5 2  8:24:55:695: 24,40,00,00,06,02,18,33,03,25,31,04,28,35,02,29,32,07,39,36,05,43,37,09,30
6 5  8:24:55:872: 27,30,00,00,07,00,10,30,08,28,30,10,23,36,04,23,33,03,18,99,99,99,99,99,99
5 5  8:24:55:871: 26,36,00,00,08,00,10,33,11,27,33,03,17,32,07,26,36,05,20,31,01,24,35,02,20
7 6  8:24:55:936: 26,30,00,02,07,00,10,36,05,24,35,02,30,33,03,24,36,04,17,32,07,14,99,99,99
```

Abbildung 3.3: Beispiel für $A_{bis}$-Daten

zur Nachbarantenne, über die es dann weitergeführt wird. Die Entscheidung über ein Handover wird von dem BSC getroffen ([MP92], S. 397). Sie ist abhängig von den Feldstärken der Nachbarsender, von den herrschenden Interferenzbedingungen und von der Belastung der Zellen ([MP92], S. 328).

Damit der BSC diese Entscheidung treffen kann, werden ihm von allen BTS die Feldstärkedaten aller Gespräche auf der $A_{bis}$-Schnittstelle übermittelt. Aus diesem Grund sammeln sich an dieser Schnittstelle die Daten, die in der vorliegenden Arbeit zur Verkehrsdichtemessung benutzt werden sollen.

### 3.1.7 Die Abis-Schnittstelle

Abbildung 3.3 zeigt ein Beispiel für die an der $A_{bis}$-Schnittstelle gesammelten Daten. Jede Zeile ist ein Datensatz, der von einer Mobilstation stammt. Ein Datensatz enthält die folgenden Werte:

- Der Link und Time Slot des Gespräches. Die hier betrachtete BTS besitzt die Links 5, 6 und 7. Im Moment laufen über die Zelle vier Gespräche über die verschiedenen Frequenzen.
- Die Uhrzeit mit Stunden, Minuten, Sekunden und Millisekunden.
- Der Feldstärkepegel des Handys, gemessen von der BTS (Uplink-Pegel)
- Der Feldstärkepegel der BTS, gemessen vom Handy (Downlink-Pegel)
- Die Qualität der Übertragung (Bitfehlerwahrscheinlichkeit, Downlink und Uplink)
- Der Timing Advance
- Die Power Control-Drosselung des Servers (BSRed)
- Die Power Control-Drosselung des Handys (MSRed)
- Sechs Plätze für BSIC, Nummer in der Nachbarschaftsliste (BAI) und Feldstärke von Nachbarsendern. Wenn weniger als sechs Zahlentripel gemeldet werden, werden die restlichen Plätze mit „99“ aufgefüllt.

Zu beachten ist dabei die Drosselung der Sender. Um die korrekte Feldstärke des Servers zu erhalten, muß man die Reduktion seiner Sendeleistung (BSRed) auf den Downlink-Pegel addieren. So ergibt sich der ungedrosselte Wert des benutzten TCHs, der gleich der Feldstärke des BCCHs angenommen wird.

In dem obigen Beispiel entspricht Link 6 dem BCCH, der nie gedrosselt wird.

### 3.1.8 Prädiktionskarten und Blackbox-Karten

An jedem Punkt kann von jeder Station die Feldstärke anhand eines physikalischen Wellenausbreitungsmodelles berechnet werden. Dies leistet das T-Mobil-Tool *Pegasos*. Die daraus entstehenden Karten für die Pegel einer Station nennt man *Prädiktionskarten*.

Allerdings sind die so berechneten Prädiktionskarten sehr ungenau [Hei00]. Aus diesem Grunde wurde dazu übergegangen, sie durch sogenannte *Blackbox-Karten* zu ersetzen.

Blackboxen sind Meßgeräte, die in Taxis eingebaut werden. Sie sind bestückt mit gewöhnlichen Handys. Die Handys führen über eine Dachantenne permanent Gespräche und messen so die Pegel der umliegenden Stationen. Die gewonnenen Daten werden zusammen mit den dazugehörigen GPS-Koordinaten aufgezeichnet. So gewinnt man in relativ kurzer Zeit viel genauere Karten, die allerdings auf das Straßennetz beschränkt sind.

Eine weitere Einschränkung der Blackbox-Karten ist, daß – wie es bei normalen Gesprächen nun mal der Fall ist – höchstens die Pegel der sechs stärksten Nachbarn gemeldet werden. Um dies zu beheben, gibt es eine Variante im sogenannten *Scanmodus*. Dabei wird auf allen 124 GSM-Kanälen einmal pro Sekunde ein Pegel gemessen – insbesondere auch auf den BCCHs aller Nachbarn. So ist garantiert, daß vollständige Vektoren vorliegen. Anhand dieser Scandaten ist die Lokalisierung eines Gespräches sehr viel genauer möglich als mit Prädiktionskarten [Sch00].

## 3.2 Modellannahmen

Die Pegeldaten, die von den Mobilstationen gemessen werden, sind sehr unsauber und verrauscht. Um sie überhaupt auswerten zu können, müssen wir verschiedene Annahmen machen.

### 3.2.1 Stabilität der Daten

Das Modell benötigt zunächst die Voraussetzung, daß es an jedem Punkt von jeder BTS einen „wahren“ Empfangspegel gibt. Dieser ist in den Prädiktions- oder Blackbox-Karten eingetragen. Natürlich wird dieser wahre Pegel in den seltensten Fällen wirklich gemessen, da die Daten durch Rauschen, Fading-Effekte und dergleichen starken Schwankungen unterliegen. Dieses Rauschen wird entweder mit modelliert oder ignoriert.

Ignoriert wurde zum Beispiel der Einfluß des Wetters auf die Meßdaten. Wir gehen davon aus, daß bei verschiedenen Gesprächen an einem Ort stets die gleichen Bedingungen herrschen und Meßdaten gemäß der gleichen Verteilung zu erwarten sind. Da unsere Meßaktionen entweder an einem Tag oder an aufeinanderfolgenden Tagen mit ähnlichem Wetter stattfanden, ist dies vertretbar.

### 3.2.2 Die Dämpfung $\lambda$

Ein Telefonat kann auf der Straße, in einem Auto, in einem Haus oder sogar im Keller stattfinden. Des weiteren können Mobiltelefone verschiedener Qualität verwendet werden. Dies wirkt sich auf die empfangenen Pegelwerte aus. Das Datenmodell sieht vor, daß dieser Effekt alle Stationen in gleichem Maße betrifft.

Das bedeutet, daß alle Signale um den gleichen Faktor geschwächt werden. Da die Pegel logarithmisch gemessen werden, wird also von jedem Pegel entsprechend der Gesprächssituation ein bestimmter Betrag abgezogen, im Keller mehr als im Auto oder auf der Straße. Dieser Wert heißt die *Dämpfung* $\lambda$. [Anl98]

Allerdings ist diese Modellannahme nur näherungsweise erfüllt. Befinden wir uns etwa in einem Haus, so werden die Stationen, die zur Fensterseite hin liegen, schwächer gedämpft werden als jene auf der anderen Seite. Obendrein ist bei jedem Gespräch der Kopf des Telefonierers im Weg und schattet die Hälfte der Sender zusätzlich ab. Da es aber schlichtweg unmöglich ist, solche Effekte ebenfalls zu modellieren, ohne das Modell vollends zu verwässern, muß darauf verzichtet werden.

Bei den diversen Meßaktionen im Rahmen des Projektes hat sich die Annahme einer gleichmäßigen Dämpfung bewährt, natürlich im Rahmen der Genauigkeit, die man bei solchen Meßdaten erwarten kann. So wurden bei Gesprächen, die im Meßfahrzeug geführt wurden, stets schwächere Signale empfangen als außerhalb.

# 4 Algorithmische Grundlagen

In Abschnitt 4.1 wird das grundlegende GTM-Modell und der Lernalgorithmus dargestellt. Abschnitt 4.2 enthält ein alternatives, Bayessches Lernverfahren; dem folgt Abschnitt 4.3 über die intrinsische Dimension der Daten. Anschließend wird die Initialisierung der Karte behandelt, desweiteren Gegenmittel gegen das Overfitting sowie schließlich in Abschnitt 4.6 Methoden zur Visualisierung einer GTM-Karte.

## 4.1 Der GTM-Algorithmus

Der GTM-Algorithmus (Generative Topographic Mapping) wurde 1996 von Bishop vorgestellt [BSW96]. Diese Darstellung des GTM-Algorithmus folgt weitgehend [Sve98].

### 4.1.1 Das Modell

Gegeben sind Trainingsdaten $t_n \in \mathbb{R}^D$, $n = 1, \ldots, N$. Die Dimension $D$ des Datenraumes kann zwar sehr groß sein, doch wir nehmen an, daß die Daten in Wirklichkeit niedrigdimensional sind. Sei also $L$ die intrinsische Dimension unserer Daten. (Da eine GSM-Zelle – wie die Erdoberfläche – zweidimensional ist, nehmen wir an, $L$ ist 2. Näheres zur Dimension der Daten findet sich in Abschnitt 4.3.) Das bedeutet, die Daten liegen (vom Rauschen abgesehen) auf einer $L$-dimensionalen Mannigfaltigkeit im $\mathbb{R}^D$. Diese Mannigfaltigkeit ist das Bild einer Funktion $y : \mathbb{R}^L \to \mathbb{R}^D$, die die $L$ Variablen $x^1, \ldots, x^L$, von denen die Daten abhängen, auf den zugehörigen Datenvektor $y(x, W) \in \mathbb{R}^D$ abbildet. Dabei ist $x \in \mathbb{R}^L$ (wobei als Urbildraum der $L$-dimensionale Einheitswürfel $[-1; 1]^L$ benutzt wird) und $W$ eine Gewichtsmatrix, die die Abbildung parametrisiert.

Eine solche Abbildung wurde bereits realisiert in Kohonens SOM [Koh82], einem $L$-dimensionalen Neuronengitter, das im $\mathbb{R}^D$ liegt. GTM greift diese Idee auf, modelliert aber zusätzlich noch das Rauschen der Testdaten abseits der Mannigfaltigkeit. Dies ergibt das folgende Wahrscheinlichkeitsmodell:

$$\begin{aligned} p(t|x, W, \beta) &= N(y(x, W), \beta) \\ &= \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left\{-\frac{\beta}{2}(t - y(x, W))^2\right\} \end{aligned} \tag{4.1}$$

Dabei ist $\beta^{-1}$ die Varianz des Gaußschen Rauschens. Die Mannigfaltigkeit wird so in den umliegenden Raum „hineingeschmiert“ (s. Abb. 4.1).

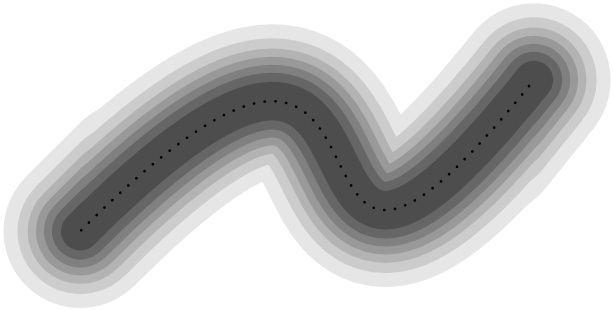


Abbildung 4.1: Die Wahrscheinlichkeitsdichte folgt einer Gaußverteilung um die Mannigfaltigkeit (gepunktet). Nimmt man das Idealbild einer Gleichverteilung der Wahrscheinlichkeitsdichte $p(x)$ auf der Mannigfaltigkeit an, so ergibt sich eine solche Wolke.

Durch Herausintegrieren der latenten Variablen $x$ ergibt sich eine Wahrscheinlichkeitsdichte im $D$-dimensionalen Raum der Testdaten:

$$p(t|W,\beta) = \int p(t|x,W,\beta)p(x)\,dx \tag{4.2}$$

Dieses Integral läßt sich im allgemeinen nicht analytisch lösen. Wir können jedoch $p(x)$, die Wahrscheinlichkeitsdichte im latenten Raum, so wählen, wie wir es von SOM her kennen: Als ein Gitter von $K$ diskreten Neuronen. Dann wird $p(x)$ zu einer Summe von Dirac-Deltas:

$$p(x) = \frac{1}{K}\sum_{k=1}^{K}\delta(x - x_k) \tag{4.3}$$

Dabei ist $x_k$ die Position des $k$-ten Neurons im latenten Raum; sein Codebook-Vektor (wie man es bei SOM nennt) ist dann $y(x_k, W)$. Sinnvollerweise ordnet man die $x_k$ auf einem regulären Gitter im $L$-dimensionalen Einheitswürfel an (s. Abb. 4.2).

Durch diese Wahl von $p(x)$ verwandelt sich das Integral aus Gleichung 4.2 in eine Summe:

$$p(t|W,\beta) = \frac{1}{K}\sum_{k=1}^{K}p(t|x_k,W,\beta) \tag{4.4}$$

Somit erhalten wir ein Gitter von Gaußblobs, die nichtlinear auf einer $L$-dimensionalen Mannigfaltigkeit im $\mathbb{R}^D$ liegen. All diese Gaußblobs haben die gleiche Varianz $\beta^{-1}$.

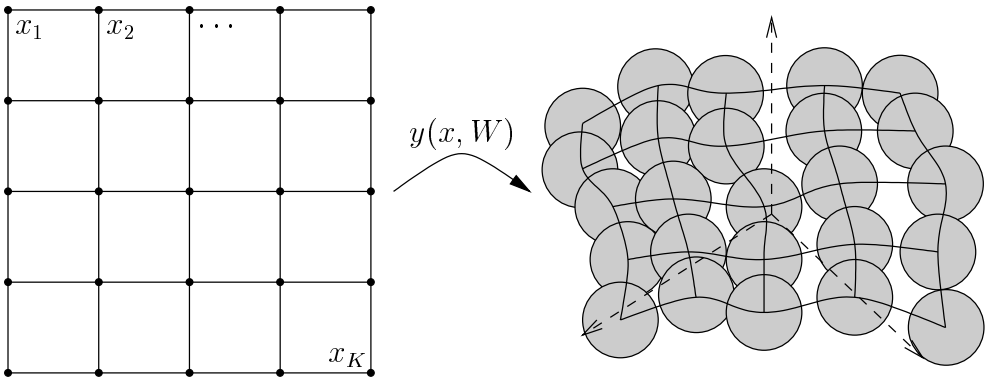



Abbildung 4.2: Das äquidistante Gitter im latenten Raum (links) wird in den hochdimensionalen Datenraum (rechts) abgebildet. Um jeden Gitterpunkt gibt es im Datenraum eine Sphäre Gaußscher Wahrscheinlichkeitsdichte (einen „Gaußblob“).

Ein solches Wahrscheinlichkeitsmodell, wie es der SOM-Algorithmus nicht besitzt, hat einen enormen Wert. Wir können z. B. für eine gegebene Testdatenmenge $\{t_1, \ldots, t_N\}$ die Log-Likelihood angeben, deren Maximierung wir dann anstreben wollen:

$$\begin{aligned} \ell &= \log \prod_{n=1}^{N} \left[ \frac{1}{K} \sum_{k=1}^{K} p(t_n | x_k, W, \beta) \right] \\ &= \sum_{n=1}^{N} \log \left( \frac{1}{K} \sum_{k=1}^{K} p(t_n | x_k, W, \beta) \right) \end{aligned} \tag{4.5}$$

### 4.1.2 Wahl der Mannigfaltigkeit

Es ist üblich, die Mannigfaltigkeit als Linearkombination von Basisfunktionen zu wählen.

$$y_d(x, W) = \sum_{m=1}^{M} \phi_m(x) w_{md} \tag{4.6}$$

In [Sve98] wird folgende Mischung empfohlen:

- $M_{NL}$ gaußförmige Basisfunktionen
- $L$ lineare Basisfunktionen
- eine konstante Basisfunktion als „bias“.

In Formelschreibweise:

$$\phi_m(x) = \begin{cases} \exp\left(-\frac{\|x-\mu_m\|^2}{2\sigma^2}\right) & \Longleftrightarrow m \leq M_{NL} \\ x^l & \Longleftrightarrow m = M_{NL} + l,\ l = 1, \ldots, L \\ 1 & \Longleftrightarrow m = M_{NL} + L + 1 = M \end{cases} \tag{4.7}$$

Dabei ist $\mu_m$ das Zentrum der $m$-ten Gaußschen Basisfunktion im latenten Raum (also auf dem Gitter im $L$-dimensionalen Einheitswürfel), $\sigma$ die gemeinsame Standardabweichung der Gaußschen Basisfunktionen (die die Flexibilität der Karte beeinflußt) und $x^l$ die $l$-te Komponente des Vektors $x \in \mathbb{R}^L$.

Es wird also allgemein angenommen, daß die Daten ein affinlineares Wesen haben, das durch die letzten $L+1$ Basisfunktionen modelliert wird; durch die $M_{NL}$ nichtlinearen, Gaußschen Basisfunktionen sind jedoch erhebliche Abweichungen von der affinlinearen Grundstruktur möglich. Man beachte weiterhin, daß diese Wahl der Basisfunktionen ebenfalls auf das SOM-Modell verweist; auch dort wird auf dem Neuronengitter gerne die Gaußfunktion als Nachbarschaftsfunktion verwendet.

Zu beachten ist der Unterschied zwischen den Gaußschen Basisfunktionen und den Gaußblobs im Datenraum. Die Gaußschen Basisfunktionen bilden die Mannigfaltigkeit. Ihre Standardabweichung $\sigma$ bestimmt die Flexibilität der Karte, wie stark sie sich im Datenraum krümmen, falten und verbiegen wird. Üblicherweise bleibt $\sigma$ während des Laufes konstant.

Die Gaußblobs im Datenraum um jeden Codebookvektor dienen hingegen dazu, das Rauschen der Daten zu modellieren. Ihre Standardabweichung $\beta^{-1/2}$ gibt an, wie stark das Rauschen, die Abweichung der Daten von unserer Mannigfaltigkeit, momentan ist. Diese Standardabweichung nimmt üblicherweise mit jedem Lernschritt ab, bis ein Maximum der Likelihood unserer Trainingsdaten erreicht ist.

Dies kann man so auffassen, daß der Anteil der Varianz der Daten, der durch das Rauschen erklärt wird, während des Lernens abnimmt; es wird immer mehr durch systematische Artefakte der Daten erklärt. Dies führt aber auch zum Phänomen des „Overfitting“: Zufällige Artefakte der Trainingsdaten, die eigentlich durch Rauschen entstanden sind, werden in das Modell der Mannigfaltigkeit eingebaut, so daß die Verallgemeinerungsfähigkeit leidet; auf einer unabhängig neu gewonnenen Datenmenge, die diese zufälligen Artefakte nicht enthält, wird die Likelihood schlechter sein. Mehr zu diesem Problem und zu Lösungsmöglichkeiten findet sich in Kapitel 4.5.

### 4.1.3 Der EM-Algorithmus

Der EM-Algorithmus („Expectation Maximization“) wurde in [DLR77] eingeführt. Eine schöne Darstellung findet sich in [Bis95], [Bil98] oder [NH98].

Der Rahmen des EM-Algorithmus ist der folgende: Gegeben sei eine Menge von Daten $T = \{t_1, \ldots, t_N\}$, die aus derselben Verteilung $p(t|\Theta)$ erzeugt worden sind. Dabei ist $\Theta$ eine Menge von Parametern der Verteilung; dies können z. B. Mittelwerte und Varianzen

einer Mischung von Gaußfunktionen sein; in unserem Falle sind es die Gewichtsmatrix $W$ sowie $\beta$.

Die Wahrscheinlichkeitsverteilung der Daten $T$ ist dann

$$p(T|\Theta) = \prod_{n=1}^{N} p(t_n|\Theta) =: \mathcal{L}(\Theta|T)$$

Wenn eine Datenmenge $T$ gegeben ist, interessiert uns diese Likelihood $\mathcal{L}$, die als Funktion der Parameter aufgefaßt wird; wir suchen dann die Parametermenge $\Theta^*$, die die Likelihood maximiert.

Manchmal kann man sie analytisch bestimmen. Der EM-Algorithmus hingegen ist eine brauchbare Methode vor allem in zwei Fällen:

- wenn die Daten unvollständig sind oder Werte fehlen,
- wenn man annehmen kann, daß die Daten von versteckten (latenten) Variablen abhängen, die die Optimierung vereinfachen.

Seien nun also $X$ die fehlenden oder versteckten Variablen und $Z$ die Kombination der gegebenen und fehlenden Daten, so daß $z_n = (t_n, x_n)$ ist. Die Wahrscheinlichkeitsdichte ist dann:

$$p(z|\Theta) = p(t, x|\Theta) = p(t|x, \Theta)p(x|\Theta)$$

Daraus ergibt sich die Likelihood anhand der vollständigen Daten:

$$\begin{aligned} \mathcal{L}(\Theta|Z) &= \prod_n p(z_n|\Theta) \\ &= \prod_n p(t_n|x_n, \Theta)p(x_n|\Theta) \end{aligned}$$

Diese Likelihood ist auch eine Wahrscheinlichkeitsfunktion, da die versteckten Variablen $X$ Zufallsvariablen sind.

Meistens wird anstelle der Likelihood ihr Logarithmus (die Log-Likelihood) benutzt, weil $\log \mathcal{L}$ analytisch einfacher zu behandeln ist. Da der Logarithmus auf $\mathbb{R}^+$ eine monotone Funktion ist, ändert sich bei der Maximierung nichts.

Der EM-Algorithmus zerfällt in zwei Schritte: den E-Schritt (Expectation) und den M-Schritt (Maximization). Ausgehend von initialen Parametern $\Theta^{(0)}$ wird wie folgt iteriert:

- Im *E-Schritt* wird eine Verteilung für die fehlenden Daten berechnet:

  $$\tilde{P}^{(i)}(x) = p(x|t, \Theta^{(i-1)})$$

  Oft hilft dabei die Bayessche Regel:

  $$p(x|t, \Theta) = \frac{p(t|x, \Theta)p(x)}{\sum_{x'} p(t|x', \Theta)p(x')} \tag{4.8}$$

- Im *M-Schritt* wird unter Annahme dieser Verteilung derjenige Parametersatz $\Theta^{(i)}$ bestimmt, der den Erwartungswert der Log-Likelihood maximiert:

$$\Theta^{(i)} = \underset{\Theta}{\operatorname{argmax}}\, E_{\tilde{P}^{(i)}}(\log \mathcal{L}(\Theta))$$

Man kann beweisen [DLR77, Bis95], daß dieses Verfahren gegen ein lokales Maximum der Likelihood konvergiert.

Wenn man im M-Schritt nicht diejenigen Parameter $\Theta^{(i)}$ berechnet, die die Likelihood maximieren, sondern solche, für die die Likelihood wächst (natürlich nur, solange noch kein Maximum erreicht ist), spricht man gemäß [DLR77] vom GEM-Algorithmus (generalisierter EM-Algorithmus); auch dieser konvergiert gegen ein lokales Maximum der Likelihood.

### 4.1.4 Ein EM-Algorithmus für das GTM-Verfahren

Der Rahmen des EM-Algorithmus paßt sehr gut zu unserem Problem, eine GTM-Karte zu trainieren. Auch wir haben versteckte (latente) Variablen $x$, deren Werte wir nicht kennen. Wäre hingegen für jeden Trainingsdatenvektor $t_n$ der Gaußblob $x_k$ bekannt, der ihn erzeugt hat, so könnte man diese Mischung von Gauß-Komponenten leicht optimieren.

Der EM-Algorithmus wird also wie folgt auf GTM angewandt:

#### Der E-Schritt

Im E-Schritt muß eine Wahrscheinlichkeitsverteilung

$$p(x|t, W, \beta)$$

berechnet werden. Nach Anwendung der Bayesschen Regel (Gleichung 4.8) ergibt sich für alle $k$ und $n$:

$$r_{kn} := p(x_k|t_n, W, \beta) = \frac{p(t_n|x_k, W, \beta)p(x_k)}{\sum_{k'} p(t_n|x_{k'}, W, \beta)p(x_{k'})} \tag{4.9}$$

Diese Größe nennen wir die *Responsibility* des $k$-ten Gaußblobs für den $n$-ten Testdatenpunkt. Sie gibt die Wahrscheinlichkeit an, daß der Datenpunkt $t_n$ von der Gaußschen Wahrscheinlichkeitsverteilung um $y_k = y(x_k, W)$ erzeugt worden ist.

Da wir $p(x_k)$, die Gewichte der einzelnen Gaußkomponenten, unabhängig von $k$ einfach als gleichverteilt annehmen,

$$p(x_k) = \frac{1}{K}$$

kürzt sich dies aus dem Bruch heraus, und wir erhalten

$$r_{kn} = \frac{p(t_n, x_k, W, \beta)}{\sum_{k'} p(t_n|x_{k'}, W, \beta)} \tag{4.10}$$

Die Berechnung dieser Responsibilities ist alles, was wir für den E-Schritt tun müssen.

### Der M-Schritt

Im M-Schritt müssen wir nun die Gewichte $W$ anpassen, um die Likelihood zu maximieren. Dazu wird die Log-Likelihood nach den Gewichten abgeleitet und gleich Null gesetzt, um das Maximum zu finden.

Für die Ableitung der Log-Likelihood (Formel 4.5)

$$\ell = \sum_{n=1}^{N} \log \left( \frac{1}{K} \sum_{k=1}^{K} p(t_n | x_k, W, \beta) \right) \tag{4.5}$$

ergibt sich für alle $m$ und $d$ (siehe auch [Sve98], S. 103):

$$\frac{\partial \ell}{\partial w_{md}} = - \sum_{n,k} r_{kn} \beta \left( \sum_{m'} \phi_{m'k} w_{m'd} - t_{nd} \right) \phi_{mk} \tag{4.11}$$

Setzt man dies gleich Null, so ergibt sich in Matrixschreibweise:

$$\Phi^T G \Phi W = \Phi^T R T$$

Dabei ist $\Phi \in \mathbb{R}^{K \times M}$ die Matrix der Aktivierungen der Basisfunktionen $\phi_m(x_k)$ (siehe Gleichung 4.7), $G \in \mathbb{R}^{K \times K}$ eine Diagonalmatrix mit $g_{kk} = \sum_{n=1}^{N} r_{kn}$, $R \in \mathbb{R}^{K \times N}$ die Matrix der Responsibilities und $T \in \mathbb{R}^{N \times D}$ die Matrix der Trainingsdaten.

Die Updateformel für die Gewichte ist dann

$$W = (\Phi^T G \Phi)^{-1} \Phi^T R T \tag{4.12}$$

Für die inverse Varianz der Gaußblobs $\beta$ ergibt sich das Optimum durch Ableitung der Log-Likelihood nach $\beta$:

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta} &= \sum_{n=1}^{N} \frac{\frac{1}{K} \sum_{k=1}^{K} \left( \frac{D \beta^{D/2-1}}{2(2\pi)^{D/2}} \exp(-\frac{\beta}{2}(t_n - y_k)^2) - \left( \frac{\beta}{2\pi} \right)^{D/2} \exp(-\frac{\beta}{2}(t_n - y_k)^2) \frac{(t_n - y_k)^2}{2} \right)}{\frac{1}{K} \sum_{k'=1}^{K} p(t_n | x_{k'}, W, \beta)} \\
&= \sum_{n,k} r_{kn} \left( \frac{D}{2\beta} - \frac{(t_n - y_k)^2}{2} \right) \\
&= \frac{D}{2\beta} \sum_{n,k} r_{kn} - \sum_{n,k} r_{kn} \frac{(t_n - y_k)^2}{2} \\
&= \frac{DN}{2\beta} - \sum_{n,k} r_{kn} \frac{(t_n - y_k)^2}{2}
\end{aligned} \tag{4.13}$$

Wird dies gleich Null gesetzt und nach $\beta^{-1}$ aufgelöst, so ergibt sich die Updateformel für die Varianz der Gaußblobs:

$$\beta^{-1} = \frac{1}{ND} \sum_{n,k} r_{kn} (t_n - y_k)^2 \tag{4.14}$$

### 4.1.5 Gewichtsregulierung

Wenn man die GTM-Karte trainiert, wie es oben beschrieben ist, stellt man oft fest, daß die Gewichte sehr groß werden, was die Karte sehr verbiegt und verzerrt. Eine Möglichkeit, dies zu lindern, ist natürlich die Wahl eines größeren $\sigma$ (die Standardabweichung der Gaußschen Basisfunktionen; siehe Gleichung 4.7) – oft jedoch wird die dadurch erreichte Glättung dann durch um so größere Gewichte wieder wettgemacht.

Es ist also ratsam, allzu große Gewichte zu bestrafen. Eine einfache Möglichkeit dafür ist, auf die Log-Likelihood noch einen Term zu addieren:

$$\ell_{neu} = \sum_{n=1}^{N} \log \left( \frac{1}{K} \sum_{k=1}^{K} p(t_n | x_k, W, \beta) \right) - \frac{\alpha}{2} \sum_{m,d} w_{md}^2$$

Wenn man dann die Log-Likelihood nach $w_{md}$ ableitet, wird Gleichung 4.11 zu

$$\frac{\partial \ell_{neu}}{\partial w_{md}} = - \sum_{n,k} r_{kn} \beta \left( \sum_{m'} \phi_{m'k} w_{m'd} - t_{nd} \right) \phi_{mk} - \alpha w_{md}$$

Wenn man dies gleich Null setzt, dann ergibt sich in Matrixschreibweise:

$$(\Phi^T G \Phi + \lambda I) W = \Phi^T R T \tag{4.15}$$

Dabei ist $I$ die $M$-dimensionale Einheitsmatrix und $\lambda = \alpha / \beta$ der Regulierungsparameter.

Diese einfache Methode wird Gewichtsregulierung („weight decay“) genannt (siehe [Bis95], S. 338).

Zu beachten ist, daß diese Manipulation das Maximum der Log-Likelihood verschiebt. Es ist nun nicht mehr die Gewichtsmatrix, die $\ell$ maximiert, sondern stellt einen Kompromiß dar zwischen Anpassung an die Daten und Minimierung der Gewichte. Deswegen ist es wichtig, $\alpha$ klug zu wählen. Wenn $\alpha$ zu groß ist, werden die Gewichte zu stark zur Null gezogen, und damit die Karte zum Nullpunkt des Datenraums.

## 4.2 Bayessches Lernen

Das Bayessche Lernen ist eine Alternative zum Maximum Likelihood-Ansatz, der oben vorgestellt wurde. Es liefert nicht nur ein Modell für die Gewichte $W$, sondern auch für die Parameter der Karte. Der Hauptvorteil gegenüber dem Maximum Likelihood-Ansatz ist, daß die Gewichtsregulierungskonstante $\alpha = \lambda\beta$ (siehe Abschnitt 4.1.5) nicht mehr gewählt werden muß, sondern auch optimiert wird.

Der Bayes-Ansatz ist ebenfalls [Sve98] entnommen. In diesem Abschnitt sind $W$ die *Gewichte* der Karte, $\alpha$, $\beta$ und $\sigma$ die *Parameter*. Zunächst wird auf die Parameter nicht eingegangen, sondern nur die Optimierung der Gewichte betrachtet.

Der Bayessche Ansatz arbeitet nicht mehr mit *einer* wahrscheinlichsten Gewichtsmatrix, sondern mit einer *Verteilung* im Raum der Gewichte. Nachdem man dem Algorithmus die Trainingsdaten präsentiert hat, kann dieser die a posteriori-Wahrscheinlichkeitsverteilung der Gewichte bestimmen, und zwar mit der Bayesschen Formel:

$$p(W|T) = \frac{p(T|W)p(W)}{p(T)} \tag{4.16}$$

Dabei ist $p(T|W)$ die Likelihood der Trainingsdaten, gegeben die aktuellen Gewichte; $p(W)$ ist die a priori-Wahrscheinlichkeitsdichte der Gewichte, ohne irgendwelche Daten gesehen zu haben; im Nenner schließlich steht die Normierungskonstante

$$p(T) = \int p(T|W)p(W)\,dW.$$

Wenden wir uns nun den einzelnen Komponenten von Gleichung 4.16 zu.

### 4.2.1 Die Likelihood der Trainingsdaten

Die Likelihood der Trainingsdaten ist gegeben durch Gleichung 4.5:

$$p(T|W) = \frac{1}{Z_T} \prod_{n=1}^{N} \sum_{k=1}^{K} \exp\left\{-\frac{\beta}{2}\|t_n - y_k\|^2\right\} \tag{4.17}$$

$$= \frac{1}{Z_T} \exp\{-S_T(W,\beta)\} \tag{4.18}$$

mit

$$S_T(W,\beta) = -\sum_{n=1}^{N} \ln \sum_{k=1}^{K} \exp\left\{-\frac{\beta}{2}\|t_n - y_k\|^2\right\} \tag{4.19}$$

$$Z_T = K^N \left(\frac{2\pi}{\beta}\right)^{ND/2} \tag{4.20}$$

(Die ausführliche Herleitung von Formel 4.20 für die Normierungskonstante $Z_T$ findet sich in [Sve98], S. 72.)

### 4.2.2 Die a priori-Verteilung der Gewichte

Üblicherweise hat man wenig Vorwissen über die Verteilung der Gewichte. Fest steht nur, daß wir uns möglichst kleine Gewichte wünschen (siehe Abschnitt 4.1.5). In [Sve98] wird daher für die a priori-Verteilung der Gewichte eine nullzentrierte Gaußverteilung eingeführt, die gerade der Gewichtsregulierung entspricht:

$$p(W) = \frac{1}{Z_W} \exp\{-S_W(W,\alpha)\} \tag{4.21}$$

mit

$$S_W(W,\alpha) = \frac{\alpha}{2}\sum_{m,d} w_{md}^2 \tag{4.22}$$

$$Z_W = \int p(W)\,dW = \left(\frac{2\pi}{\alpha}\right)^{MD/2} \tag{4.23}$$

### 4.2.3 Zusammenfassung und Approximation

Insgesamt ist

$$p(T|W)p(W) = \frac{1}{Z_T Z_W}\exp\{-(S_T(W,\beta) + S_W(W,\alpha))\}$$

Wir müssen also die Summe

$$S(W,\beta,\alpha) = S_T(W,\beta) + S_W(W,\alpha) \tag{4.24}$$

minimieren. $S_W$ ist quadratisch in den Gewichten. $S_T$ wäre dann quadratisch in den Gewichten, wenn für jeden Datenpunkt $t_n$ genau eine Komponente $y_k$ die ganze Responsibility übernähme. Da dies bei trainierten Karten häufig näherungsweise der Fall ist, können wir diese Approximation akzeptieren und von $S(W,\beta,\alpha)$ nur die Taylorentwicklung bis zur zweiten Ordnung betrachten:

$$S(W,\beta,\alpha) \approx S(W_{MP},\beta,\alpha) + \frac{1}{2}\Delta\vec{w}^T H \Delta\vec{w}$$

Dabei ist $W_{MP}$ das Minimum, um das wir entwickeln, $\Delta\vec{w}$ ist $W_{MP} - W$, als Vektor geschrieben, und $H \in \mathbb{R}^{(MD)\times(MD)}$ ist die Hessematrix, die Matrix der zweiten Ableitungen, ausgewertet am Minimum:

$$H_{ij} = \left.\frac{\partial^2 S(W,\beta,\alpha)}{\partial w_i \partial w_j}\right|_{W_{MP}}$$

Der lineare Term verschwindet, da um das Minimum entwickelt wird, wo der Gradient Null ist.

Diese Taylorentwicklung entspricht einer Approximation von $p(W|T)$ durch eine Gaußsche Wahrscheinlichkeitsverteilung, da es dann nur noch quadratisch von den Gewichten abhängt.

Die Hessematrix zerfällt in zwei Teile: $H = H_W + H_T$. $H_W$ entsteht aus den zweiten Ableitungen von $S_W$. Aus 4.22 folgt leicht

$$H_W = \alpha I$$

(Dabei ist $I$ die Einheitsmatrix.)

Die Berechnung von $H_T$, der zweiten Ableitung von $S_T$, ist da länglicher. Sie findet sich in [Sve98], S. 103-105. Neben der exakten Lösung, die jedoch für ein Online-Verfahren zu aufwendig zu berechnen ist, wird eine Approximation angegeben, die wiederum aus der Annahme entsteht, daß für jeden Datenpunkt $t_n$ eine Gaußkomponente $y_k$ existiert, die nahezu die gesamte Responsibility für diesen Datenpunkt übernimmt. Diese approximative Hessematrix ist eine $(MD) \times (MD)$-Blockdiagonalmatrix aus $D$ identischen $M \times M$-Blöcken:

$$\beta \Phi^T G \Phi \tag{4.25}$$

Diese Matrix kennen wir schon aus Gleichung 4.12.

Zusammen mit $H_W$ ergibt sich also für $H$ eine Blockdiagonalmatrix mit den Blöcken

$$\beta \Phi^T G \Phi + \alpha I,$$

wie wir sie aus Gleichung 4.15 kennen.

Insgesamt ist dann die a posteriori-Verteilung der Gewichte

$$p(W|T) = \frac{1}{Z} \exp\left\{-S(W_{MP}, \beta, \alpha) - \frac{1}{2}\Delta\vec{w}^T H \Delta\vec{w}\right\} \tag{4.26}$$

mit der Normierungskostante (siehe [Bis95], Anhang B)

$$Z = \exp\left\{-S(W_{MP}, \beta, \alpha)\right\} (2\pi)^{MD/2} (\det H)^{-1/2} \tag{4.27}$$

### 4.2.4 Einbeziehung der Parameter

Wenn man nun die Parameter $\alpha$, $\beta$ und $\sigma$ in die Formeln einbezieht, so wird Formel 4.16 zu

$$p(W|T, \alpha, \beta, \sigma) = \frac{p(T|W, \beta, \sigma) p(W|\alpha)}{p(T|\alpha, \beta, \sigma)} \tag{4.28}$$

Um $p(W|T)$ zu erhalten, muß man die Parameter herausintegrieren:

$$p(W|T) = \iiint p(W|T, \alpha, \beta, \sigma) p(\alpha, \beta, \sigma|T)\, d\alpha\, d\beta\, d\sigma$$

Dabei ist nach Bayes' Theorem

$$p(\alpha, \beta, \sigma|T) = \frac{p(T|\alpha, \beta, \sigma) p(\alpha, \beta, \sigma)}{p(T)}$$

In [Sve98] (S. 74) wird weiterhin hergeleitet:

$$\begin{aligned} p(T|\alpha, \beta, \sigma) &= \int p(T|W, \beta, \sigma) p(W|\alpha)\, dW \\ &= \frac{Z}{Z_T Z_W} \end{aligned} \tag{4.29}$$

Dies sind gerade die Normalisierungskonstanten aus den Formeln 4.20, 4.23 und 4.27.
Der Logarithmus von (4.29) ist

$$\ln p(T|\alpha,\beta,\sigma) = -S_T(W_{MP},\beta,\sigma) - S_W(W_{MP},\alpha) - \frac{1}{2}\ln(\det H) + \frac{MD}{2}\ln(\alpha) + \frac{ND}{2}\ln(\beta) - \frac{ND}{2}\ln(2\pi) - N\ln(K). \quad (4.30)$$

Die Determinante der Hessematrix läßt sich (mit der Approximation von $H_T$) schreiben als

$$\ln(\det H) = \ln \prod_{m=1}^{M} (\lambda_m + \alpha)^D = D \sum_{m=1}^{M} \ln(\lambda_m + \alpha), \quad (4.31)$$

wobei die $\lambda_m$ die Eigenwerte eines Blocks von $H_T$ (4.25) sind.

### 4.2.5 Online-Schätzung der Parameter

Durch Ableitung von (4.30) nach $\alpha$, $\beta$ und $\sigma$ ergeben sich jeweils die optimalen Parameterwerte.

Wir beginnen mit dem Parameter $\alpha$. Leiten wir Gleichung 4.30 nach $\alpha$ ab, so ergibt sich

$$\begin{aligned} \frac{d\ln p(T|\alpha,\beta,\sigma)}{d\alpha} &= -\frac{1}{2}\sum_{m,d} w_{md}^2 + \frac{MD}{2\alpha} - \frac{D}{2}\sum_m \frac{1}{\lambda_m + \alpha} \quad (4.32) \\ &= -\frac{1}{2}\sum_{m,d} w_{md}^2 + \frac{D}{2\alpha}\sum_m \frac{\lambda_m}{\lambda_m + \alpha} \end{aligned}$$

Dies können wir gleich Null setzen und bekommen

$$\alpha = \frac{\gamma}{\sum_{m,d} w_{md}^2} \quad (4.33)$$

mit

$$\gamma = D \sum_{m=1}^{M} \frac{\lambda_m}{\lambda_m + \alpha} \quad (4.34)$$

Dieses $\gamma$ läßt sich auffassen als ein Maß für die Anzahl der „gut bestimmten Parameter“ (siehe [Sve98], S. 75; [Bis95], S. 410).

Für $\beta$ ergibt sich zunächst – da die Blöcke von $H_T$ (4.25) linear von $\beta$ abhängen –

$$\frac{d\lambda_m}{d\beta} = \frac{\lambda_m}{\beta}$$

und somit als Ableitung von 4.31

$$\frac{d\ln(\det H)}{d\beta} = D\sum_m \frac{1}{\lambda_m + \alpha}\frac{\lambda_m}{\beta} = \frac{\gamma}{\beta}$$

Leiten wir nun Gleichung 4.30 nach $\beta$ ab und setzen dies gleich Null, so ergibt sich mit Gleichung 4.13:

$$\beta = \frac{ND - \gamma}{\sum_{n,k} r_{kn}(t_n - y_k)^2} \tag{4.35}$$

Diese Gleichung ist bis auf das $\gamma$ identisch mit Gleichung 4.14.

Für $\sigma$ schließlich läßt sich leider keine elegante Formel ableiten. Deshalb gibt es keine andere Möglichkeit, als $\sigma$ zu Beginn zu wählen und während des Lernens konstant zu halten. (Dies hat andererseits den Vorteil, daß auch die Aktivierungsmatrix $\Phi$ während des Lernens konstant bleibt.)

### 4.2.6 Der Lernalgorithmus

Insgesamt ergibt sich der folgende Lernalgorithmus (siehe [Sve98], S. 76):

**für** ein gegebenes $\sigma$
    Initialisiere die Karte mit $\sigma$
    **wiederhole**
        Optimiere $W$ mit EM-Algorithmus mit (4.10) und (4.15) ($\alpha$ und $\beta$ konstant)
        **wenn** Abbruchbedingung für $W$ erfüllt ist
            Update $\alpha$ und $\beta$ mit (4.33) und (4.35)
    **bis** Abbruchbedingung für $\alpha$ und $\beta$ erfüllt ist
    Speichere Wahrscheinlichkeit für $\sigma$ ab (Formel 4.30)

Die Abbruchkriterien ergeben sich aus einer Schranke für das Wachstum der Log-Likelihood oder einer Maximalzahl von Schritten. Für diese Arbeit wurde beispielsweise als Bedingung für das Update von $\alpha$ und $\beta$ gewählt: $\Delta\ell < 0{,}1$. Wenn die Änderung der Log-Likelihood einmal so klein ist, ist die Karte so weit trainiert, daß die Approximation greift, daß für jeden Datenpunkt eine Gaußkomponente den Großteil der Responsibility übernimmt (siehe Abschnitt 4.2.3). Dies ist die Voraussetzung für die Updateformeln von $\alpha$ und $\beta$.

Als äußere Abbruchbedingung kann man entweder eine ähnliche Bedingung aufstellen oder als Kriterium das „Early Stopping“ verwenden (siehe Abschnitt 4.5.3).

## 4.3 Die Dimension der Daten

### 4.3.1 Die effektive Dimension

Mit der Dimension einer Datenmenge ist hier die Dimension der Mannigfaltigkeit gemeint, auf der die Daten liegen. Meschkowski [Mes66] definiert: „Ein topologischer Raum heißt eine Mannigfaltigkeit von der Dimension $n$, wenn jeder seiner Punkte eine Umgebung besitzt, die homöomorph [d. h. umkehrbar stetig abbildbar] zum Innern der

$n$-dimensionalen Einheitskugel ist.“ Man stellt sich also in unserem hochdimensionalen Raum eine $n$-dimensionale Hyperebene vor, die beliebig gekrümmt und verbogen hineingeschrieben ist.

Dabei gibt es jedoch zwei Probleme: Erstens sind unsere Daten verrauscht, müssen also gar nicht exakt auf der Mannigfaltigkeit liegen. Zweitens ließe sich jede Datenmenge durch eine eindimensionale Mannigfaltigkeit beschreiben, eine einfache Zickzack-Linie, die alle Datenpunkte miteinander verbindet.

In der mathematischen Definition wird die Existenz einer Umgebung gefordert, aber sie gibt keine Aussage darüber, welche Größenordnung diese haben sollte. Da sie jedoch vernünftigerweise von der Größenordnung der Abstände unserer Datenpunkte sein sollte, stellt sich der Gedanke von der Zickzack-Linie, die wie eine Peanokurve alle Punkte verbindet, als unbrauchbar heraus. Zu oft käme es vor, daß Punkte, die im Datenraum benachbart sind, auf der Kurve sehr weit entfernt lägen, so daß bei einer solchen Abbildung viel Struktur der Daten verloren ginge.

Auf einer sehr kleinen Skala könnte man also die Daten durchaus als eindimensional gelten lassen. Auf einer etwas größeren Skala dann dürften die Daten $D$-dimensional wirken (wobei $D$ die Dimension des Datenraumes ist), da das Rauschen sie in alle Richtungen verschmiert. Die uns interessierende effektive Dimension müßte sich auf einer noch größeren Skala manifestieren – aber nicht auf einer so großen, daß alle Punkte ununterscheidbar sind und die Dimension Null ist. Gibt es eine Möglichkeit, die Dimension in Abhängigkeit von der Größenskala zu berechnen?

Die gibt es [Qua97, BS95], und sie folgt direkt aus der Definition: Man lege eine $D$-dimensionale Einheitssphäre des Radius $r$ um jeden Datenpunkt $t$ und zähle die Anzahl $N_t(r)$ der Datenpunkte, die darinnen liegen. Sei

$$\bar{N}(r) = \frac{1}{|\{t\}|} \sum_t N_t(r)$$

der Durchschnitt über alle Punkte. Nehmen wir nun an, wir verdoppelten den Radius der Sphäre. Wenn die Daten die Dimension $L$ haben (für „latente Dimension“), dann gilt

$$\frac{\bar{N}(2r)}{\bar{N}(r)} \approx 2^L$$

Bei dreidimensionalen Daten zum Beispiel verachtfacht sich die Anzahl der Datenpunkte, wenn man den Radius der Sphäre verdoppelt (vgl. Abb. 4.3).

Natürlich gilt dieselbe Formel auch für beliebiges $\alpha$:

$$\frac{\bar{N}(\alpha r)}{\bar{N}(r)} \approx \alpha^L$$

Nach $L$ aufgelöst:

$$L \approx \frac{\log \frac{\bar{N}(\alpha r)}{\bar{N}(r)}}{\log \alpha}$$

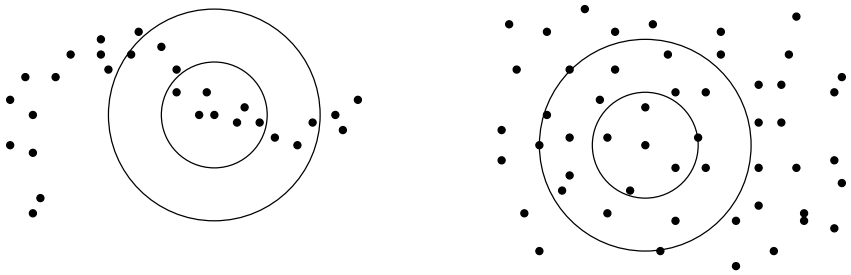


Abbildung 4.3: Die intrinsische Dimension einer Datenmenge: Die linke Menge ist eindimensional. Wenn der Radius der Sphäre verdoppelt wird, verdoppelt sich auch die Anzahl der darin liegenden Datenpunkte. Bei der rechten, zweidimensionalen Datenmenge vervierfacht sich die Anzahl.

Nun können wir $\log \alpha$ gegen Null gehen lassen und definieren so die Dimension der Daten abhängig von der Größenskala $r$:

$$L(r) := \lim_{\alpha \to 1} \frac{\log \frac{\bar{N}(\alpha r)}{\bar{N}(r)}}{\log \alpha}$$

Es gilt aber auch:

$$\begin{aligned} L(r) &= \lim_{\alpha \to 1} \frac{\log \bar{N}(\alpha r) - \log \bar{N}(r)}{\log \alpha r - \log r} \\ &= \frac{d}{d \log r} \log \bar{N}(r) \end{aligned} \tag{4.36}$$

Man trägt also $\bar{N}(r)$ doppeltlogarithmisch gegen $r$ auf und mißt dann die Steigung dieser Kurve. Ist diese Steigung über einen großen Bereich konstant, so kann man eine Gerade daranpassen und als deren Steigung die latente Dimension $L$ der Daten ablesen (siehe z. B. Abbildung 7.2).

$\bar{N}(r)$ bestimmt man am effizientesten, indem man für alle Paarungen von Datensätzen den Abstand berechnet und ein Histogramm anfertigt. Die Anzahl der Abstände, die kleiner als $r$ sind, ist dann proportional zu $\bar{N}(r)$, was nach dem Logarithmieren zu äquivalentem Vorgehen berechtigt.

Übrigens reicht es für die Dimensionsanalyse auch aus, nicht sämtliche $O(N^2)$ Paarungen von Datenpunkten zu betrachten, sondern eine genügend große Stichprobe.

### 4.3.2 Hauptkomponentenanalyse

Die Hauptkomponentenanalyse (Principal Component Analysis; PCA) ist eine sehr gängige Methode, Daten zu analysieren [Jol86]. Es wird eine orthonormale Basis $U = \{u_d | d =$

$1, \ldots, D\}$ des $\mathbb{R}^D$ berechnet. Dabei zeigt $u_1$ in die Richtung, in der die Daten die größte Varianz haben. $u_2$, orthogonal dazu, zeigt in die Richtung, die die nächstgrößte Varianz enthält (nachdem der von $u_1$ erzeugte Unterraum herausprojiziert wurde), und so weiter. Die Varianz der Daten entlang eines Basisvektors $u_d$ sei $\lambda_d$.

Man berechnet die Basis $U$, indem man zuerst die Kovarianzmatrix der Daten $\{t_n\}$ erzeugt:

$$S = \frac{1}{N-1} \sum_{n=1}^{N} (t_n - \bar{t})(t_n - \bar{t})^T$$

Dabei ist $\bar{t} = \frac{1}{N} \sum_n t_n$ der Mittelwert der Daten.

Dann ist die gesuchte Basis $U$ die Matrix der Eigenvektoren von $S$, $\lambda_d$ ist der zu $u_d$ gehörende Eigenwert [Sve98].

Die Hauptkomponentenanalyse kann auch eine Auskunft über die Dimension der Daten geben. Wenn es im sortierten Vektor der $\lambda_d$ ein $\tilde{d}$ gibt, hinter dem die Eigenwerte sehr klein werden, dann folgt daraus, daß die Varianz der Daten im wesentlichen in dem von $\{u_1, \ldots, u_{\tilde{d}}\}$ aufgespannten Unterraum liegt und der Rest als Rauschen vernachlässigt werden kann. In diesem Fall ist $\tilde{d}$ die effektive Dimension der Daten.

Die Hauptkomponentenanalyse kann allerdings nur lineare Strukturen in den Daten erkennen. Für Daten, die nichtlineare Strukturen enthalten, wird das Ergebnis suboptimal sein. Zur Initialisierung der GTM-Karte kann sie jedoch sehr gut dienen.

## 4.4 Initialisierung

### 4.4.1 Zufällige Werte

Eine Möglichkeit der Initialisierung der Gewichtsmatrix $W$ und der Varianz des Rauschens $\beta^{-1}$ ist die, $W$ mit Zufallszahlen zu initialisieren und $\beta^{-1}$ auf den Erwartungswert des quadratischen Abstandes jedes Datenpunkts von seinem nächsten Gaußblob zu setzen.

Es hat sich jedoch schon bei den frühesten Versuchen mit sehr einfachen Spielmodellen gezeigt, daß dies zu einer sehr verbogenen und verdrehten Karte führt. Dieses Problem (das SOM in diesem Maße nicht hat) wird bestätigt in [KO98]: „[...] it seems that the GTM requires a careful initialization to self-organize[...] it had difficulties to unfold after a random initialization[...]“

Solche Verdrehungen später aus der Karte zu entfernen, ist praktisch unmöglich. Aus diesem Grunde ist eine aufwendigere Initialisierungsmethode geboten.

### 4.4.2 Initialisierung mit Hauptkomponentenanalyse

Wie bereits in Abschnitt 4.3.2 ausgeführt, ermittelt die Hauptkomponentenanalyse den linearen Unterraum des Datenraumes, in dem die meiste Varianz der Daten wohnt.

Daraus ergibt sich eine Initialisierungsmethode, die die Gewichte so wählt, daß die Mannigfaltigkeit sich zu Beginn genau auf diesen Unterraum legt.

- Die Gewichte der konstanten (Bias-)Basisfunktion werden auf den Mittelwert der Daten gesetzt.
$$\vec{w}_M = \bar{t}$$
- Die Gewichte der linearen Basisfunktionen werden auf die ersten $L$ Eigenvektoren $u_l$ (diejenigen mit den größten Eigenwerten $\lambda_l$) gesetzt, jeweils gewichtet mit $2\sqrt{\lambda_l}$.
$$\vec{w}_{M_{NL}+l} = 2\sqrt{\lambda_l}u_l \qquad l = 1, \ldots, L$$
Da $\lambda_l$ die Varianz der Daten entlang des Eigenvektors $u_l$ ist, kann man, indem man die Mannigfaltigkeit mit dem Faktor $2\sqrt{\lambda_l}$ dehnt, etwa 95 % der Daten abdecken ([Ehr86], S. 60), denn – angenommen eine Normalverteilung der Daten – gilt
$$\int_{-2}^{2} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}\, dt \approx 0,9544.$$
- Die Gewichte der Gaußschen Basisfunktionen werden auf Null gesetzt.
$$\vec{w}_m = 0 \qquad m = 1, \ldots, M_{NL}$$

Die Varianz des Gaußschen Rauschens $\beta^{-1}$ wird auf den nächstgrößten Eigenwert $\lambda_{L+1}$ gesetzt, um die Varianz zu modellieren, die nicht durch die lineare Struktur eingefangen wird.

### 4.4.3 Initialisierung mit Prädiktionskarten

Die GTM-Karte kann auch mit Hilfe der Prädiktionskarten (s. Abschnitt 3.1.8) initialisiert werden, so daß sich die initiale Mannigfaltigkeit genau den prädizierten Werten anpaßt.

Dazu wird zunächst auf der Prädiktionskarte die Zelle gesucht, aus der die Trainingsdaten stammen. Der Einfachheit halber wird die Bounding Box der Zelle benutzt (d. h. das kleinste achsenparallele Rechteck auf der Karte, das die Zelle enthält).

Diese Bounding Box wird in $K$ gleichförmige Unterrechtecke zerteilt, von denen jedes einen Blob erhalten soll. Für jeden dieser Blobs wird ein $D$-dimensionaler Feldstärkevektor $y_k^{pred}$ berechnet, einfach als Durchschnitt der Pegelwerte in allen Pixeln des Unterrechtecks.

Dies ergibt die Matrix der $Y^{pred}$, die die gewünschten Positionen der Blobs enthält. Daraus kann man die gesuchte Gewichtsmatrix $W$ anhand der Gleichung $Y = \Phi W$ (siehe Gleichung 4.6) erzeugen – leider aber ist $\Phi \in \mathbb{R}^{K \times M}$ nicht notwendigerweise quadratisch. Die Lösung läßt sich jedoch mit der Pseudoinversen von $\Phi$ finden [Tom99]:
$$W = (\Phi^T \Phi)^{-1} \Phi^T Y^{pred} \tag{4.37}$$

Schließlich wird $\beta^{-1}$ als der Durchschnitt des quadratischen Abstands zweier benachbarter Blobs im Datenraum initialisiert.

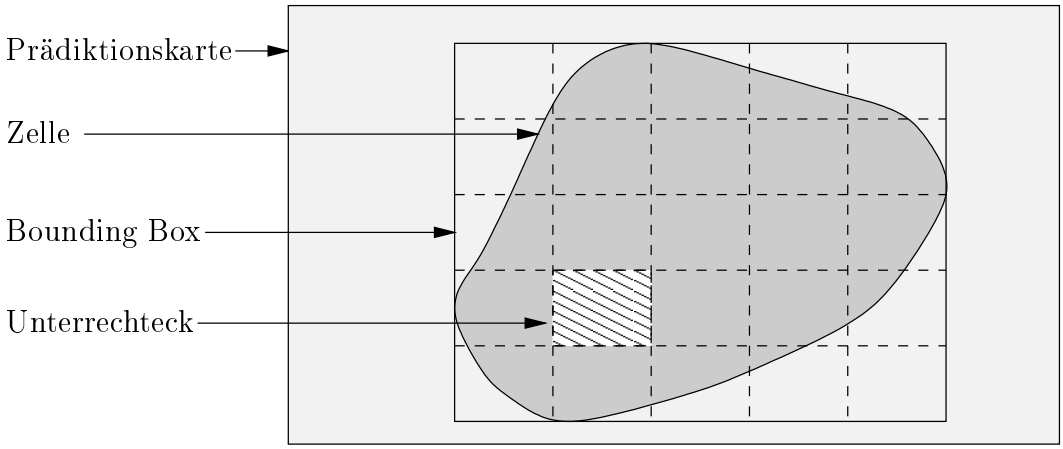



Abbildung 4.4: Initialisierung mit Prädiktionskarten. Zunächst wird die Bounding Box der Zelle ermittelt, dann wird sie in $K$ (hier 25) Unterrechtecke aufgeteilt, schließlich für jedes Unterrechteck der Mittelwert der Prädiktionswerte gebildet und damit die Position des zugehörigen Gaußblobs initialisiert.

#### Blackbox-Daten

Da die Blackbox- und Scankarten viel genauer sind als die Prädiktionskarten [Sch00], wäre es natürlich angebracht, diese statt der Prädiktionskarten zu verwenden. Diese aber haben den Nachteil, daß sie nicht flächendeckend, sondern auf das Straßennetz beschränkt sind. Man kann also nicht alle Werte aus einem Unterreckteck aufsummieren, da man damit rechnen muß, daß die allermeisten Pixel keinen Wert enthalten, weil das Meßfahrzeug dort nicht gewesen ist. Es kommt sogar vor, daß ein Unterrechteck überhaupt keinen Meßwert enthält.

Da die Initialisierung nicht besonders genau sein muß (der Versuch, genaue flächendeckende Prädiktionskarten anhand der lückenhaften Blackbox-Daten zu entwerfen, wird in [Ler01] unternommen), reicht eine unkomplizierte Handhabung dieses Problems:

Um einen Meßwert in einem Unterrechteck zu bekommen, wird der Durchschnitt aller darin gemessenen Werte gebildet, gleich wie viele Werte es sind. Wenn ein Unterrechteck überhaupt keinen Meßwert enthält, wird der Durchschnitt der bis zu acht umliegenden Unterrechtecke, die einen Datenwert enthalten, verwendet.

## 4.5 Overfitting

Nehmen wir an, die Trainingsdaten entstammen einer $L$-dimensionalen Mannigfaltigkeit $h(x)$ im $\mathbb{R}^D$, sind jedoch mit einem zufälligen Rauschen $\epsilon$ versehen. Wenn wir eine GTM-Karte trainieren, ist es unser Ziel, die Mannigfaltigkeit $y(x, W)$ möglichst gut an $h(x)$ anzunähern.

Das Hauptziel ist es nicht, die Trainingsdatenmenge optimal abzubilden, sondern die zugrundeliegende Struktur der Daten – eben $h(x)$ – zu erkennen.

Zu Beginn des Trainingsprozesses wird sich die Mannigfaltigkeit der Karte an $h(x)$ anpassen. Je länger jedoch trainiert wird, desto mehr wird man feststellen, daß das spezielle Rauschen der Trainingsdaten $\epsilon$ mitgelernt wird. Darunter leidet dann die Verallgemeinerungsfähigkeit der Karte. Diesen Effekt bezeichnet man als *Overfitting*.

### 4.5.1 Theorie des Overfitting – Bias und Varianz

In [Bis95] (Kapitel 9) wird auf dieses Problem ausführlich eingegangen.[1] Es rührt daher, daß nur eine endliche Datenmenge zur Verfügung steht. Haben wir etwa eine Datenmenge $T$ gegeben, die gemäß $h(x) + \epsilon$ verteilt ist, so kann man an einer Stelle $x$ den Fehler des Mappings $y$ angeben als

$$(y(x) - \langle t|x\rangle)^2 \tag{4.38}$$

Dabei ist $\langle t|x\rangle$ der Erwartungswert der Trainingsdaten bei gegebenem $x$:

$$\langle t|x\rangle = \sum_{t\in T} t\, p(t|x)$$

Da wir die Verallgemeinerungsfähigkeit des Netzes für alle Daten ermitteln wollen, betrachten wir den Mittelwert von (4.38) über alle Datenmengen [GBD92]:

$$\begin{aligned}
E_T[(y(x) - \langle t|x\rangle)^2] &= E_T[((y(x) - E_T[y(x)]) + (E_T[y(x)] - \langle t|x\rangle))^2] \\
&= E_T[(y(x) - E_T[y(x)])^2] + E_T[(E_T[y(x)] - \langle t|x\rangle)^2] \\
&\quad + 2E_T[(y(x) - E_T[y(x)])(E_T[y(x)] - \langle t|x\rangle)] \\
&= E_T[(y(x) - E_T[y(x)])^2] + (E_T[y(x)] - \langle t|x\rangle)^2 \\
&\quad + 2E_T[y(x) - E_T[y(x)]](E_T[y(x)] - \langle t|x\rangle) \\
&= \underbrace{E_T[(y(x) - E_T[y(x)])^2]}_{\text{Varianz}} + \underbrace{(E_T[y(x)] - \langle t|x\rangle)^2}_{\text{Bias}^2}
\end{aligned}$$

(Der gemischte Term verschwindet, da der Erwartungswert der Differenz zum Erwartungswert Null ist.)

Der Fehler einer trainierten Karte läßt sich demnach aufspalten in zwei Komponenten:

- Der *Bias* mißt den mittleren Abstand zwischen den Trainingsdaten und den vom Netz prädizierten Werten.

[1] Bishop bezieht sich dort auf überwachtes Lernen, d. h. für jeden Eingabevektor $x$ gibt es genau ein $h(x)$, das die optimale Lösung für das Netz ist. Die selbstorganisierenden Karten arbeiten unüberwacht – so sind schon allein durch Drehungen oder Spiegelungen der Karte verschiedene, gleich gute Lösungen gegeben. Dieser Unterschied ist jedoch für die Betrachtungen in diesem Kapitel unerheblich. Aus Gründen der Anschaulichkeit übernehme ich hier Bishops Sichtweise, weil überwacht trainierte Netze besser zu vergleichen sind. Der Vergleich selbstorganisierender Karten ist hingegen nicht trivial. [KL96]

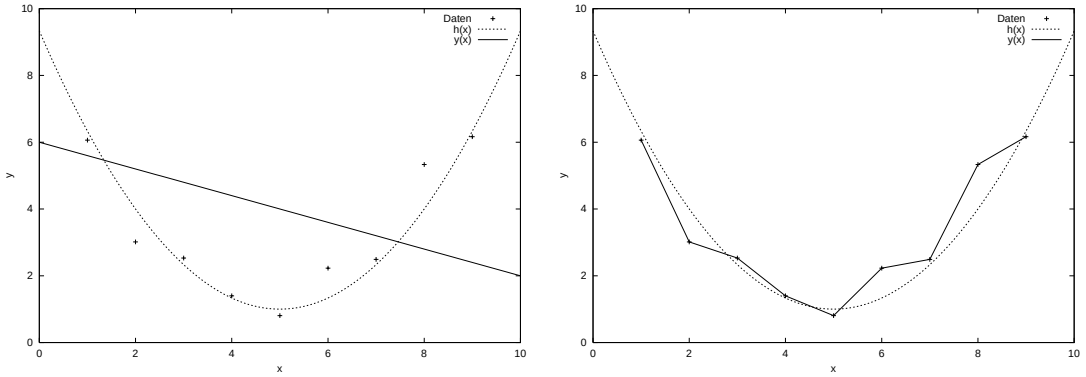



Abbildung 4.5: Bias und Varianz: Im linken Bild ist das Netz wenig abhängig von den Trainingsdaten – im Extremfall überhaupt nicht. Dann ist die Varianz Null, aber der Bias sehr hoch. Das rechte Bild zeigt den anderen Extremfall: Das resultierende Netz ist sehr abhängig von den Trainingsdaten, die Varianz ist also sehr hoch; dafür ist der Bias klein, da sich das Netz sehr genau anpaßt.

- Die *Varianz* mißt die Empfindlichkeit des Ergebnisses für verschiedene Datenmengen.

Man beachte, daß diese Werte jeweils für ein bestimmtes $x$ gelten. Dies kann man natürlich beheben, indem man $x$ herausintegriert. Das ergibt die folgenden Formeln:

$$\text{Varianz} = \int E_T[(y(x) - E_T[y(x)])^2]p(x)\,dx \tag{4.39}$$

$$\text{Bias}^2 = \int (E_T[y(x)] - \langle t|x\rangle)^2 p(x)\,dx \tag{4.40}$$

Es gibt also zwei Extrema, zwischen denen man sich bewegt (vgl. Abb. 4.5):

- Man kann als Modell irgendeine beliebige, feste Funktion $y(x)$ wählen. Da sie überhaupt nicht von unseren Trainingsdaten abhängt, ist die Varianz bezüglich der Wahl von $T$ natürlich Null. Dafür ist der Bias sehr hoch, da das Modell völlig willkürlich ist und weitab von den Daten sein kann.

- Andererseits kann man das Modell sehr flexibel wählen, so daß es sich genau den Trainingsdaten anpaßt. Dann ist der Bias sehr klein; an den Datenpunkten ist er sogar Null, da
$$E_T[y(x)] = E_T[h(x) + \epsilon] = h(x) = \langle t|x\rangle$$
gilt. Dafür ist die Varianz sehr groß; je nach Wahl der Trainingsdaten kommen sehr verschiedene Mappings $y(x)$ heraus. Es ergibt sich
$$E_T[(y(x) - E_T[y(x)])^2] = E_T[(y(x) - h(x))^2] = E_T[\epsilon^2],$$
also genau die Varianz des Rauschens der Daten.

Das optimale Modell ist ein Kompromiß zwischen diesen beiden Extrema.

### 4.5.2 Minimierung von Bias und Varianz

Das Ziel ist es, sowohl Bias als auch Varianz zu minimieren. Eine Möglichkeit dafür ist, mehr Daten zu benutzen. Dies ist aber nicht immer möglich und verursacht auch Speicher- und Rechenzeitprobleme.

Oft gibt es wählbare Parameter, die das Ergebnis des Netzes in bezug auf Minimierung von Bias und Varianz beeinflussen. Im Falle des GTM-Algorithmus gibt es etwa $\sigma$, die Flexibilität der Basisfunktionen. Für ein großes $\sigma$ ist die Varianz klein, aber der Bias groß (man spricht hier von *Underfitting*). Ist $\sigma$ klein, dann ist das Netz sehr flexibel; es kann sich den Trainingsdaten genau anpassen, was zu einem kleinen Bias führt. Die Varianz ist jedoch sehr groß. Darunter leidet die Verallgemeinerungsfähigkeit für eine andere, unabhängig gewonnene Datenmenge. Dieses Phänomen heißt *Overfitting*.

Man stellt bei einer solchen übertrainierten Karte auch fest, daß $\beta^{-1}$ bei jedem Lernschritt gesunken ist und nun einen sehr kleinen Wert hat. Das Rauschen der Daten wird als immer kleiner angesehen – am Ende viel kleiner, als es wirklichkeitsgemäß sinnvoll ist. Die Visualisierung einer solchen Karte zeigt sehr deutlich die Diskretisierung in einzelne Blobs, zwischen denen die Wahrscheinlichkeitsdichte extrem klein ist. Die Blobs haben sich so genau wie möglich den vorhandenen Trainingsdaten angepaßt.

Dies ist unerwünscht, denn das Ziel ist nicht, die gegebenen Trainingsdaten genau zu lernen, sondern vielmehr, den zugrundeliegenden Generator möglichst gut zu modellieren.

Eine Möglichkeit, dieses Overfitting zu lindern, ist die Einführung von Straftermen, die allzu stark verzerrte Karten verhindern. In unserem Beispiel ist dies die Gewichtsregulierung $\lambda$ (siehe Abschnitt 4.1.5). Diese bewirkt eine Glättung der Karte; die Topologie des ursprünglichen Raumes bleibt besser erhalten. Das allzu genaue Anpassen an die Trainingsdaten kann sie jedoch nicht ganz verhindern. Auch darf man die Regulierungskonstante nicht zu groß wählen, um das Ergebnis nicht zu sehr zu verfälschen.

### 4.5.3 Early Stopping

Eine weitere Methode, das Overfitting zu verhindern, heißt *Early Stopping* ([Bis95], S. 343). Dabei werden die gegebenen Daten in drei Teile gespalten:

- die Trainingsdaten, die zum Lernen benutzt werden,
- die Validierungsdaten, um über das Ende des Lernens zu entscheiden, und
- die Testdaten, um die Qualität des Ergebnisses zu beurteilen.

Während des Lernens wird die Likelihood der Trainingsdaten stets ansteigen – der EM-Algorithmus garantiert dies. Auch auf der Validierungsmenge wird die Likelihood zunächst steigen. Irgendwann jedoch beginnt die GTM-Karte, die Artefakte und das Rauschen der Trainingsdaten mitzulernen. Dann sinkt die Likelihood der Validierungsdaten, und dies ist auch der richtige Zeitpunkt, um das Training zu beenden.

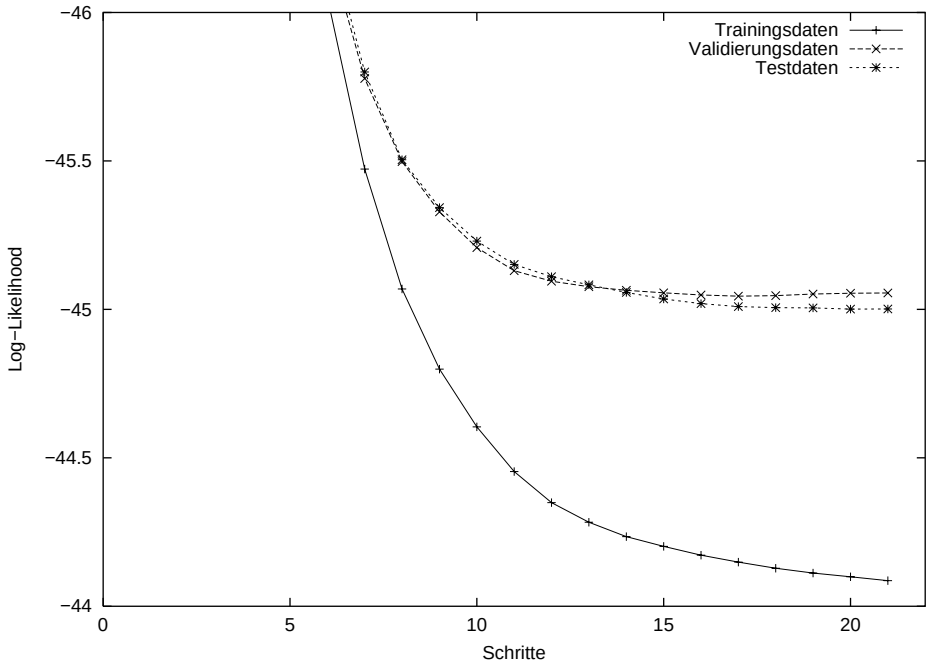



Abbildung 4.6: Early Stopping: Nach dem 17. Lernschritt hat die Log-Likelihood der Validierungsdaten ein Maximum erreicht. Dieses Netz nimmt man daher als dasjenige mit den besten Generalisierungseigenschaften an.

Die dritte Menge der Testdaten wird dann benutzt, um das Ergebnis zu bewerten. Weder die Likelihood der Trainingsdaten noch die der Validierungsdaten (auf deren Likelihood das Netz ja optimiert wurde) sind dafür geeignet. Dazu brauchen wir noch eine dritte Menge, die von den beiden anderen unabhängig ist.

Dieses Verfahren ist nur anwendbar, wenn genügend Daten zur Verfügung stehen, um sich die Aufspaltung leisten zu können. Unsere $A_{bis}$-Daten sind jedoch so reichlich bemessen, daß dies kein Problem darstellt. Bei knapperen Meßdaten kann man sich mit dem Verfahren der *Cross-Validation* helfen; mehr dazu findet sich beispielsweise in [Bis95] (S. 372).

## 4.6 Visualisierung

### 4.6.1 Meßdaten im latenten Raum

Die hochdimensionalen Pegeldaten im latenten Raum darzustellen, ist nicht ganz trivial. In der Regel liegen sie nicht auf der Mannigfaltigkeit $y(x, W)$, da sie ja verrauscht sind. Um eine Datenmenge zu visualisieren, muß man sie zunächst auf den latenten Raum abbilden.

In [Sve98] werden dazu zwei Möglichkeiten angegeben:

- Man kann für einen Datenvektor $t_n$ den Modalwert im latenten Raum bestimmen:

$$x_n^{mode} = \operatorname*{argmax}_{x_k} p(x_k|t_n) \tag{4.41}$$

  Dabei ist $p(x_k|t_n)$ die Responsibility des $k$-ten Blobs für den Datenvektor $t_n$ (siehe Gleichung 4.9).

  Dies entspricht dem üblichen Vorgehen bei der SOM, das „Gewinnerneuron“ zu bestimmen, das dem Vektor am nächsten liegt, und ihn darauf abzubilden.

- Die Alternative ist der gewichtete Mittelwert im latenten Raum:

$$x_n^{mean} = \sum_{k=1}^{K} x_k p(x_k|t_n) \tag{4.42}$$

Die Mittelwert-Abbildung hat den Vorteil, daß sie nicht auf die $K$ Gitterpunkte im latenten Raum beschränkt ist, sondern den gesamten latenten Raum ausnutzt. Allerdings wird ein Datenpunkt, für den mehrere $x_k$ (womöglich auf der Karte weit voneinander entfernt) eine hohe Responsibility übernehmen, auf deren Mittelwert abgebildet, was sehr verkehrt sein kann. Andererseits kann aber auch der Modalwert bei einer multimodalen Verteilung verkehrt sein.

In der für diese Arbeit geschriebenen GTM-Software sind beide Abbildungen vorgesehen; für die Erzeugung der Bilder in Kapitel 7 wurde jedoch stets der Mittelwert-Darstellung $x^{mean}$ der Vorzug gegeben.

### 4.6.2 Wahrscheinlichkeitsdichte

Um Hotspots auf der Karte zu finden, muß man die Wahrscheinlichkeitsdichte im latenten Raum darstellen können. Die nächstliegende Möglichkeit hierzu ist eine Karte der Wahrscheinlichkeitsdichte selbst:

$$x \mapsto p(y(x, W)|W, \beta) \tag{4.43}$$

Diese Karte ist jedoch mit Nachteilen behaftet: Da die Wahrscheinlichkeitsdichte in Gaußblobs verteilt ist, zeichnet sich oft das Gitterraster der Blobs sehr deutlich auf der Karte ab.

Außerdem ist die Wahrscheinlichkeitsdichte im latenten Raum gleichförmig vorgegeben (siehe Gleichung 4.3). Wenn die Gaußblobs im Pegelraum genügend weit auseinander liegen, wird man auf der Wahrscheinlichkeitsdichtekarte auch wenig Struktur sehen. Nur wo sich mehrere Gaußblobs im Pegelraum überschneiden, ist die Wahrscheinlichkeitsdichte signifikant größer, was als Hotspot gedeutet werden kann.

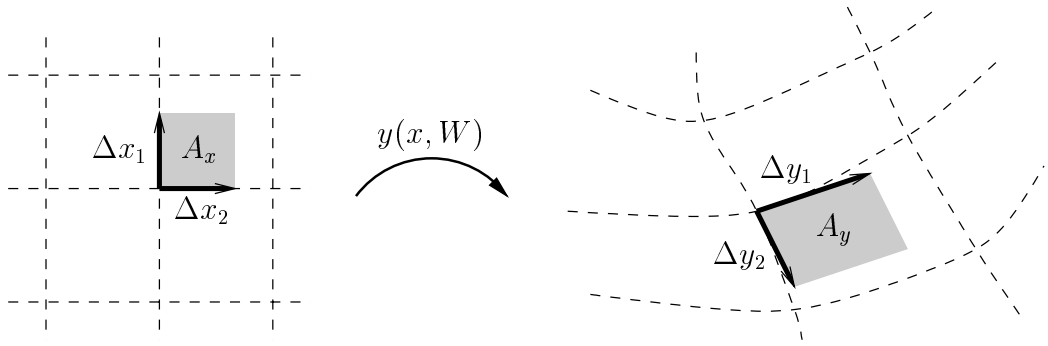



Abbildung 4.7: Magnification Factors: Berechnet wird die Fläche $A_y$ im Bildraum im Verhältnis zur Fläche $A_x$ im Urbildraum.

### 4.6.3 Magnification Factors

Eine andere Visualisierungsmethode, die nicht durch das Gitterraster der Blobs beeinträchtigt wird, sind die *Magnification Factors* (Vergrößerungsfaktoren). Sie sind inspiriert durch das biologische Vorbild: Im somatosensorischen Cortex des menschlichen Gehirns nehmen die Bereiche der Finger, der Lippen oder der Zunge viel mehr Platz ein als andere, so daß dort die Berührungsreize feiner unterschieden werden können. [BGPW96]

Kohonen übertrug ein Maß für eine solche Verzerrung auf den SOM-Algorithmus [Koh95]. Da dieser ein diskretisiertes Neuronengitter verwendet, ist auch die Funktion der Magnification Factors diskretisiert. Im Prinzip besteht es darin, zu jeder zwischen vier benachbarten Neuronen aufgespannten Fläche die Größe der Bildfläche im Datenraum zu berechnen.

Der GTM-Algorithmus hat demgegenüber den Vorteil, daß er eine kontinuierliche Mannigfaltigkeit modelliert. Deswegen läßt sich nicht nur für die Gaußzentren (die ja den Neuronen der SOM entsprechen), sondern für die ganze Mannigfaltigkeit $y(x), x \in \mathbb{R}^L$ eine Formel für den Magnification Factor herleiten. [Sve98]

Für die Herleitung nehmen wir zunächst eine zweidimensionale Karte an ($L = 2$), da dies sowohl der häufigste als auch der anschaulichste Fall ist. Die Verallgemeinerung auf beliebige $L$ ist kanonisch.

Wir interessieren uns für ein Rechteck der Fläche $A_x = \Delta x_1 \Delta x_2$ im latenten Raum und die Bildfläche $A_y$ auf der Mannigfaltigkeit im Datenraum. $A_y$ ist in linearer Näherung ein Parallelogramm, das von den Vektoren $\Delta y_1$ und $\Delta y_2$ aufgespannt wird (siehe Abbildung 4.7).

Da wir die $\Delta x_l$ gegen Null gehen lassen, ist diese lineare Näherung korrekt, denn es gilt für $l = 1, 2$:

$$\lim_{\Delta x_l \to 0} \frac{\Delta y_l}{\Delta x_l} = \frac{\partial y_l(x)}{\partial x_l} \tag{4.44}$$

Dann ist das Quadrat der Fläche

$$A_y^2 = |\Delta y_1|^2 |\Delta y_2|^2 - (\Delta y_1 \Delta y_2^T)^2$$

**Beweis:** Gegeben seien zwei Vektoren $a$, $b$, die im Winkel $\alpha$ zueinander stehen. Dann ist das Quadrat der Fläche des von ihnen aufgespannten Parallelogrammes

$$\begin{aligned} A^2 &= (|a||b| \sin\alpha)^2 \\ &= |a|^2 |b|^2 \sin^2\alpha \\ &= |a|^2 |b|^2 (1 - \cos^2\alpha) \\ &= |a|^2 |b|^2 - (|a||b| \cos\alpha)^2 \\ &= |a|^2 |b|^2 - \|a \cdot b\|^2 \end{aligned}$$

■

Mit $A_x = \Delta x_1 \Delta x_2$ und Übergang zum Infinitesimalen gemäß (4.44) ergibt sich die folgende Formel für den Magnification Factor:

$$\frac{dA_y}{dA_x} = \sqrt{\left|\frac{\partial y}{\partial x_1}\right|^2 \left|\frac{\partial y}{\partial x_2}\right|^2 - \left(\frac{\partial y}{\partial x_1}\frac{\partial y}{\partial x_2}^T\right)^2} \tag{4.45}$$

Nun fehlt nur noch eine Formel für die partiellen Ableitungen. Aus Gleichung 4.6 ergibt sich jedoch sofort

$$\frac{\partial y_d}{\partial x_l} = \sum_{m=1}^{M} \psi_{lm} w_{md} \tag{4.46}$$

mit

$$\psi_{lm} = \frac{\partial \phi_m}{\partial x_l} = \begin{cases} -\exp\left(-\frac{\|x-\mu_m\|^2}{2\sigma^2}\right)\frac{x_l - \mu_{ml}}{\sigma^2} & \Longleftrightarrow m \le M_{NL} \\ 1 & \Longleftrightarrow m = M_{NL} + l,\ l = 1, \ldots, L \\ 0 & \Longleftrightarrow m = M_{NL} + L + 1 = M \end{cases}$$

Dies läßt sich auch zusammengefaßt in Matrixschreibweise ausdrücken. Die Jacobimatrix $J \in \mathbb{R}^{L \times D}$, die die partiellen Ableitungen der Funktion $y$ enthält,

$$J_{ld} = \frac{\partial y_d}{\partial x_l},$$

ist nach Formel 4.46

$$J = \Psi W,$$

wobei $\Psi \in \mathbb{R}^{L \times M}$ die Matrix der $\psi_{lm}$ ist.

Es läßt sich leicht nachprüfen, daß Gleichung 4.45 identisch ist zu

$$\frac{dA_y}{dA_x} = \sqrt{\det(JJ^T)}$$

In [Sve98] wird gezeigt, daß diese Formel für beliebiges $L$ gilt.

Dazu wird zunächst dargelegt, daß das Volumen des von den $\frac{\partial y}{\partial x_l}$ (den Zeilen der Jacobimatrix) aufgespannten Parallelepipeds gleich der Determinante von $\hat{J} = J\hat{M}$ ist. Dabei ist $\hat{M} \in \mathbb{R}^{D \times L}$ eine Orthonormalbasis des von $J$ aufgespannten Unterraumes, die sich mit dem Gram-Schmidt-Verfahren finden läßt.

Dann läßt sich schließlich mit Hilfe der Eigenschaften der Determinante und orthonormaler Matrizen zeigen, daß

$$\det \hat{J} = \sqrt{\det(\hat{J}\hat{J}^T)} = \sqrt{\det(JJ^T)}$$

ist.

# 5 Eigene Arbeit

Ein Ziel der Arbeit ist, den GTM-Algorithmus an die Besonderheiten der $A_{bis}$-Daten anzupassen.

Zunächst werden die Erweiterungen zur Behandlung unvollständiger Daten dargestellt – in Abschnitt 5.1 die Erweiterung des GTM-Algorithmus, in Abschnitt 5.2 eine alternative Hauptkomponentenanalyse.

Anschließend werden die algorithmischen Möglichkeiten, mit der Dämpfung $\lambda$ umzugehen, behandelt: in Abschnitt 5.3 alternative Metriken, um die Dämpfung herauszuprojizieren, in Abschnitt 5.4 ein vereinfachtes (semilineares) dreidimensionales Modell.

Die Log-Likelihood ist zur Beurteilung des Lernerfolges nur bedingt geeignet, da sie vom Datenmodell und von der Variante des GTM-Modells abhängt. Somit ist sie zum Vergleich verschiedener Modelle nicht verwendbar; daher werden hier andere Gütemaße betrachtet. In Abschnitt 5.5 wird ein Maß für die Trennung der Cluster vorgestellt, in Abschnitt 5.6 das topographische Produkt und andere Maße für die Topologieerhaltung selbstorganisierender Karten.

Schließlich folgt Abschnitt 5.7 über die Vorverarbeitung der $A_{bis}$-Daten. Darin wird ein Modul für LocAbis [Hei00] beschrieben, das $A_{bis}$-Dateien einliest, die Gespräche über Intervalle mittelt und abspeichert. Außerdem wird das anschließende Mischen der Daten behandelt.

Die GTM-Software wird in Kapitel 6 dargestellt.

## 5.1 Der GTM-Algorithmus mit fehlenden Daten

Bei den meisten unserer Datensätze fehlen einige Werte, und zwar aus mehreren Gründen:

- Oft sind einige Stationen aus der Nachbarschaftsliste zu schwach, um empfangen zu werden, denn das Handy muß die BSIC dekodieren können, um eine Feldstärke zu melden.

  Die Feldstärke kann zu klein sein, weil der Sender sehr weit entfernt ist, oder weil er durch Abschattung oder Dämpfung geschwächt ist.

- Ein Sender kann nur dann gemeldet werden, wenn seine BSIC erkannt wurde.

Vor allem zu Beginn des Gespräches ist es üblich, daß sehr wenige Nachbarn gemeldet werden. Das Handy braucht einige Zeit, um alle Frequenzen zu überprüfen und die BSICs zu dekodieren.

Ist die BSIC eines Nachbarn einmal dekodiert, so bleibt dieser Nachbar für einige Zeit mit dem jeweils aktuellen Pegelwert auf der Frequenz in der Liste stehen. Wenn es jedoch einige Zeit nicht gelingt, erneut die BSIC zu dekodieren, so verschwindet er wieder.

Auch kann ein Sender zwar einen hohen Pegelwert aufweisen, aber durch Interferenz gestört sein, wenn eine andere BTS auf derselben Frequenz sendet. Dies kann ebenfalls die Dekodierung der BSIC verhindern.

- Jeder $A_{bis}$-Vektor enthält nur Platz für sechs Nachbarfeldstärken. Deshalb können auch dann, wenn mehrere Nachbarn einen hohen Pegelwert aufweisen und ihre BSICs erkannt wurden, einige von diesen fehlen.

Durch Mittelung über Calls oder Zeitintervalle lassen sich diese Probleme lindern. Wenn z. B. mehr als sechs Nachbarn hohe Pegelwerte aufweisen, kommt es oft vor, daß zu verschiedenen Phasen des Gespräches verschiedene Nachbarn gemeldet werden. Nach der Mittelung kann der resultierende Datensatz durchaus mehr als sechs Werte enthalten.

Ganz beheben läßt sich das Problem so freilich nicht. Deswegen sollte der Algorithmus Techniken enthalten, vernünftig mit fehlenden Werten umzugehen.

Es gibt für fehlende Werte zwei verschiedene Modellannahmen, die hier in Betracht kommen [GJ94]:

**Missing at random:** Daß ein Wert eines Datensatzes fehlt, ist völlig zufällig und hat nichts mit den gemeldeten Werten des Datensatzes oder mit dem Wert, der gemeldet würde, zu tun. Dies bedeutet, daß in der Tatsache, daß die Komponente fehlt, keine Information enthalten ist.

In diesem Fall kann man die fehlenden Komponenten einfach ignorieren und mit den restlichen Werten arbeiten. Dies wird in Abschnitt 5.1.1 ausgeführt.

**Abgeschnittene Daten:** Der Wert fehlt, weil er zu schwach ist und außerhalb des Wertebereiches liegt.

Vorgehensweisen für diesen Fall werden in Abschnitt 5.1.2 beleuchtet.

In den folgenden Abschnitten werden zunächst die beiden Modellannahmen genauer behandelt. Anschließend gehe ich auf die Wahl eines der beiden Modelle ein.

### 5.1.1 Zufällig fehlende Werte

Der EM-Algorithmus ist nicht nur für Modelle mit latenten (versteckten) Variablen brauchbar, sondern auch für fehlende Daten. Diese werden dabei genauso behandelt wie

die latenten Variablen: sie werden herausintegriert (siehe Formel 4.2). Dies wird vorgeschlagen in [Sve98], S. 87f. Sei der Datensatz $t$ gespalten in die gemessenen Werte $t^o$ und die fehlenden Werte $t^m$. Wir betrachten dann

$$p(t^o|W,\beta) = \iint p(t^o,t^m|x,W,\beta)p(x)\,dx\,dt^m$$

Wenn der latente Vektor $x$ gegeben ist, sind die einzelnen Komponenten des Datenvektors $t$ unabhängig voneinander, da als einzige zufällige Komponente das Gaußsche Rauschen mit Varianz $\beta^{-1}$ bleibt, das auf alle Komponenten von $t$ unabhängig wirkt. Deswegen kann man die Wahrscheinlichkeitsdichte auseinanderziehen:

$$\begin{aligned} p(t^o|W,\beta) &= \int p(t^o|x,W,\beta)p(x) \int p(t^m|x,W,\beta)\,dt^m\,dx \\ &= \int p(t^o|x,W,\beta)p(x)\,dx \end{aligned}$$

Daraus folgt, daß die fehlenden Werte einfach weggelassen und nur die gemessenen Dimensionen betrachtet werden. (Hier endet der Ansatz in [Sve98].) Die Wahrscheinlichkeitsdichte bedingt durch die latenten Variablen ist dann:

$$p(t_n^o|x_k,W,\beta) = \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left\{-\frac{\beta}{2}\sum_{\substack{d=1\\ t_{nd}\,\text{obs.}}}^{D}(t_{nd}-y(x_k,W))^2\right\} \tag{5.1}$$

Natürlich läuft die Summe nur über die observierten Komponenten des Datenvektors.

Die Log-Likelihood einer Testdatenmenge ergibt sich analog zu Formel 4.5:

$$\ell = \sum_{n=1}^{N}\log\left(\frac{1}{K}\sum_{k=1}^{K}p(t_n^o|x_k,W,\beta)\right)$$

Als partielle Ableitung nach dem Gewicht $w_{md}$ ergibt sich

$$\frac{\partial\ell}{\partial w_{md}} = \sum_{\substack{n=1\\ t_{nd}\,\text{obs.}}}^{N}\sum_{k=1}^{K} r_{kn}\left(-\frac{\beta}{2}\right)2\left\{t_{nd}-\sum_{m'=1}^{M}\phi_{km'}w_{m'd}\right\}\phi_{km} \tag{5.2}$$

mit

$$r_{kn} = \frac{p(t_n^o|x_k,W,\beta)}{\sum_{k'=1}^{K}p(t_n^o|x_{k'},W,\beta)}$$

Setzt man Gleichung 5.2 gleich Null und multipliziert die geschweifte Klammer aus, so erhält man

$$\forall m,d \qquad \sum_{k=1}^{K}\phi_{km}\left(\sum_{\substack{n=1\\ t_{nd}\,\text{obs.}}}^{N} r_{kn}\right)\sum_{m'=1}^{M}\phi_{km'}w_{m'd} = \sum_{k=1}^{K}\phi_{km}\sum_{\substack{n=1\\ t_{nd}\,\text{obs.}}}^{N} r_{kn}t_{nd}$$

Gegenüber der einfachen Update-Formel des gewöhnlichen GTM-Algorithmus (Gleichung 4.12) enthält diese Formel zwei Änderungen:

- Auf der rechten Seite steht nicht mehr einfach $RT$, das Produkt der beiden Matrizen, sondern es wird nur über diejenigen $n$ aufsummiert, für die $t_{nd}$ beobachtet wurde. $RT$ wird also ersetzt durch $\widetilde{RT}$ mit

$$\widetilde{rt}_{kd} = \sum_{\substack{n=1 \\ t_{nd} \text{ obs.}}}^{N} r_{kn} t_{nd}$$

- Auf der linken Seite ist die Aufsummierung aller $r_{kn}$, die in Formel 4.12 die Matrix $G$ ergab, nun auch noch von $d$ abhängig. Wir brauchen also eine ganze Familie von Diagonalmatrizen $G_d, d = 1, \ldots, D$ mit

$$(G_d)_{kk} = \sum_{\substack{n=1 \\ t_{nd} \text{ obs.}}}^{N} r_{kn}$$

Dies bedeutet, daß wir in jedem Schritt nicht mehr nur eine, sondern $D$ $M \times M$-Matrizen invertieren müssen.

Für jeden Spaltenvektor $w_d$ der Gewichtsmatrix ergibt sich die Updateformel

$$w_d = (\Phi^T G_d \Phi)^{-1} \Phi^T \widetilde{RT}_d \tag{5.3}$$

### 5.1.2 Abgeschnittene Daten

Eine Möglichkeit, abgeschnittene Daten zu behandeln, ist eine Fallunterscheidung in eine abgeschnittene Gaußverteilung und einen Peak. Oberhalb eines gegebenen $t_0^d$ ist die Wahrscheinlichkeitsdichte eine Gaußfunktion wie in Formel 4.1. Die gesamte Wahrscheinlichkeit für Werte kleiner als $t_0^d$ sammelt sich dort in einem Dirac-Peak.

$$p(t_n|x, W, \beta) = \prod_{d=1}^{D} \begin{cases} \sqrt{\frac{\beta}{2\pi}} \exp\left\{-\frac{\beta}{2}(t_{nd} - y_d(x, W))^2\right\} & \Longleftrightarrow t_{nd} > t_0^d \\ \Phi\left(\frac{t_0^d - y_d(x,W)}{\beta^{-1/2}}\right) \delta(t_0^d) & \text{sonst} \end{cases} \tag{5.4}$$

Dabei ist $\delta$ das Dirac-Delta und $\Phi$ das Integral über die Gaußfunktion:

$$\Phi(t) = \int_{-\infty}^{t} \varphi(\tau)\, d\tau$$
$$\varphi(\tau) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{\tau^2}{2}\right\}$$

Daraus ergibt sich als Ableitung der Log-Likelihood (4.5) nach einem Gewicht $w_{md}$:

$$\begin{aligned}\frac{\partial \ell}{\partial w_{md}} &= \sum_{n=1}^{N} \frac{1}{\sum_{k'} p(t_n|x_{k'}, W, \beta)} \\ &\quad \cdot \sum_{k=1}^{K} \begin{cases} p(t_n|x_k, W, \beta)\beta(t_{nd} - y_{kd})\phi_{km} & \Longleftrightarrow t_{nd} > t_0^d \\ -\sqrt{\beta}\,\varphi\left(\sqrt{\beta}(t_0^d - y_{kd})\right)\delta(t_0^d)\phi_{km} & \text{sonst} \end{cases} \\ &= \sum_{n,k} r_{kn}\phi_{km} \begin{cases} \beta(t_{nd} - y_{kd}) & \Longleftrightarrow t_{nd} > t_0^d \\ -\sqrt{\beta}\,\frac{\varphi\left(\sqrt{\beta}(t_0^d - y_{kd})\right)}{\Phi\left(\sqrt{\beta}(t_0^d - y_{kd})\right)} & \text{sonst} \end{cases} \end{aligned} \tag{5.5}$$

Durch diese Fallunterscheidung ist die Update-Formel nicht mehr vermittels einfacher Matrixoperationen lösbar. Mit der Approximation

$$\frac{\varphi(x)}{\Phi(x)} \approx -x \tag{5.6}$$

fällt jedoch die Fallunterscheidung weg. Die Formel ist dann identisch zu der, für die die fehlenden Komponenten der Daten jeweils durch $t_0^d$ ersetzt wurden.

Diese Approximation ist vor allem für $x < 0$, also $y_d > t_0^d$, anwendbar. Sie ist jedoch bislang die effizienteste Möglichkeit, mit abgeschnittenen Daten umzugehen. Dieses Vorgehen entspricht dem, für einen fehlenden Wert einen möglichst plausiblen Schätzwert $t_0^d$ einzusetzen.

Nun bleibt nur noch ein Wert für $t_0^d$ zu wählen. Dazu gibt es die folgenden Modellannahmen:

- Ein nicht gemeldeter Pegel kann als schwach angenommen werden. Diese Vorstellung ist insbesondere bekräftigt, wenn weniger als sechs Nachbarn gemeldet wurden.

  Dieses Modell wurde in [Qua97] verwendet, was insofern statthaft ist, als die dort betrachtete Zelle nur sechs Nachbarn hatte. Somit konnte kein Nachbarpegel wegen eines bereits gefüllten $A_{bis}$-Datensatzes wegfallen. Deswegen wurde für einen nicht gemeldeten Wert stets eine 0 eingesetzt.

  Da die 0 noch im Wertebereich der RxLev-Werte liegt, ist es sinnvoller, stattdessen den Wert $-1$ zu benutzen. (Selbst dies könnte – die Annahme eines schwachen Pegels vorausgesetzt – noch zu hoch sein, da es auch für noch schwächere Werte möglich sein kann, eine BSIC zu dekodieren, wobei dann eine 0 gemeldet wird [siehe Tabelle 3.1]. Wird jedoch überhaupt kein Wert gemeldet, so ist der Pegel selbst dafür zu schwach.)

- Wenn bereits sechs Werte in einem $A_{bis}$-Vektor stehen, so kann man über einen Sender, der nicht darunter auftaucht, lediglich die Aussage treffen, daß sein Pegel vermutlich schwächer als der schwächste gemeldete Pegel (genannt $t_{min}$) sein muß.

  Folglich gab es auch die Überlegung, für eine nicht gemeldete Komponente in den Vektor den Wert $t_{min} - 1$ einzutragen. Dies hat allerdings den Nachteil, daß $t_{min} - 1$

stark davon abhängt, was für Werte in den anderen Komponenten stehen. Abhängig davon kann dann der Abstand zweier Vektoren, die an demselben Ort aufgezeichnet wurden, sehr viel größer sein, als es plausibel ist.

### 5.1.3 Wahl eines Modells für die fehlenden Daten

Die Schwierigkeit besteht darin, daß wir es offenbar mit einer Mischung der verschiedenen Modelle zu tun haben. Wenn eine Komponente fehlt, weil der entsprechende Sender schwach ist, dann ist dies ein klassisches Beispiel für abgeschnittene Daten. Wenn jedoch eine Komponente fehlt, weil der Sender durch Interferenz gestört ist, ist über seine wahre Feldstärke nichts bekannt. Sind hingegen die sechs Plätze des $A_{bis}$-Datensatzes schon anderweitig belegt, dann ist laut Modell der Wert $t_{min} - 1$ plausibel.

Wenn man unterscheiden könnte, warum eine Vektorkomponente fehlt, wäre es möglich, die Fälle gesondert zu behandeln. Dies ist jedoch mit letzter Gewißheit nicht möglich.

Deshalb wurden mit allen drei Paradigmen (missing at random, $-1$ und $t_{min} - 1$) Karten trainiert, um das Ergebnis zu beurteilen. Dabei muß man sich dessen bewußt sein, daß man immer einen unvermeidbaren Fehler in Kauf nehmen muß, zumal keine der Modellannahmen zutrifft – so wie auch die Annahme einer gleichmäßigen Dämpfung $\lambda$ nicht zutrifft. Vielmehr haben wir es mit einer Mischung verschiedener Phänomene zu tun, die zu fehlenden Datenkomponenten führen und die man nicht sicher voneinander trennen kann.

Wie in Abschnitt 7.4.2 ausgeführt wird, ist die Annahme, die Komponenten fehlten zufällig (so daß in der Tatsache, daß der Wert fehlt, keine Information enthalten ist), nicht haltbar. Der beste Lernerfolg wurde mit dem Paradigma $-1$ erzielt.

## 5.2 Die Hauptkomponentenanalyse mit fehlenden Daten

In [Row97] wird für die Hauptkomponentenanalyse (siehe Abschnitt 4.3.2) ein EM-Algorithmus vorgeschlagen. Dieser hat gegenüber den herkömmlichen Methoden – etwa der Eigenwertanalyse der Kovarianzmatrix oder der Singulärwertzerlegung [PTVF92] – unter anderem den Vorteil, auch bei fehlenden Daten anwendbar zu sein.

### 5.2.1 Ein EM-Algorithmus für die PCA

Diese Darstellung folgt weitgehend [Row97], ebenso die Notation: Die Einheitsmatrix nennen wir $I$. $x \sim \mathcal{N}(\mu, \Sigma)$ bedeutet, daß der Vektor $x$ normalverteilt ist mit Mittelwert $\mu$ und Kovarianzmatrix $\Sigma$, d. h. ([Zei96], S. 1049):

$$p(x) = \sqrt{\frac{\det \Sigma^{-1}}{(2\pi)^k}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

$\mathcal{N}(\mu, \Sigma)|_x$ bedeutet Auswertung dieser Normalverteilung an einer Stelle $x$.

Die Hauptkomponentenanalyse hat zum Ziel, denjenigen Unterraum des Datenraumes zu finden, in dem die meiste Varianz der Daten wohnt. Sei $y$ ein $p$-dimensionaler Datenvektor. Der Einfachheit halber und ohne Beschränkung der Allgemeinheit nehmen wir an, der Mittelwert der Daten sei Null.

Die Hauptkomponentenanalyse kann als linear-Gaußsches Modell aufgefaßt werden:

$$y = Cx + v \qquad x \sim \mathcal{N}(0, I) \qquad v \sim \mathcal{N}(0, R) \tag{5.7}$$

Ein standardnormalverteilter $k$-dimensionaler latenter Vektor $x$ wird über eine Transformationsmatrix $C \in \mathbb{R}^{p \times k}$ linear in den $p$-dimensionalen Datenraum projiziert. $C$ spannt also den Raum der ersten $k$ Hauptkomponenten der Daten auf. Hinzu kommt ein Vektor Gaußschen Rauschens $v$.

Da die Komponenten von $x$ und von $v$ jeweils gaußverteilt und voneinander unabhängig sind, ergibt sich für $y$ ebenfalls eine Gaußverteilung:

$$y \sim \mathcal{N}(0, CC^T + R) \tag{5.8}$$

Nun kann man auch eine Wahrscheinlichkeitsdichte für den latenten Vektor $x$ bei Beobachtung eines Datenvektors $y$ angeben. Dazu bedient man sich der Bayesschen Formel:

$$\begin{aligned} P(x|y) &= \frac{P(y|x)P(x)}{P(y)} \\ &= \frac{\mathcal{N}(Cx, R)|_y \mathcal{N}(0, I)|_x}{\mathcal{N}(0, CC^T + R)|_y} \qquad (5.9) \\ &= \mathcal{N}(\beta y, I - \beta C)|_x \qquad (5.10) \end{aligned}$$

mit

$$\beta = C^T (CC^T + R)^{-1}$$

Es ist unbedingt notwendig, die Kovarianzmatrix des Rauschens $R$ einzuschränken; wenn man sie beliebig wählen könnte, könnte man einfach $C = 0$ setzen und die gesamte Varianz der Daten durch Rauschen erklären.

Fordert man jedoch zum Beispiel, daß $R$ eine Diagonalmatrix ist, so ergibt sich als Modell eine Faktorenanalyse. Die Hauptkomponentenanalyse entsteht hingegen, indem man das Rauschen gegen Null gehen läßt:

$$R = \lim_{\epsilon \to 0} \epsilon I \tag{5.11}$$

Das bewirkt, daß die Likelihood eines Punktes $y$ lediglich von seinem quadratischen Abstand zu $Cx$ abhängt, seinem rekonstruierten Vektor aus dem zugehörigen $x$, der in dem von $C$ aufgespannten Unterraum liegt. Diese Likelihood wird maximiert, wenn $C$ gerade den Unterraum der $k$ ersten Hauptkomponenten aufspannt. In Formeln bedeutet dies, daß Gleichung 5.10

$$P(x|y) = \mathcal{N}(\beta y, I - \beta C)|_x \qquad \beta = \lim_{\epsilon \to 0} C^T (CC^T + \epsilon I)^{-1}$$

konvergiert zu

$$P(x|y) = \mathcal{N}((C^TC)^{-1}C^Ty, 0)|_x \quad (5.12)$$
$$= \delta(x - (C^TC)^{-1}C^Ty) \quad (5.13)$$

Da das Rauschen Null geworden ist, kollabiert die Normalverteilung zu einem Dirac-Peak.

Daraus ergibt sich der folgende EM-Algorithmus für die Hauptkomponentenanalyse:

**E-Schritt:** Es muß die Verteilung für die latenten Variablen berechnet werden. Für gegebene Daten $Y \in \mathbb{R}^{p\times n}$ und eine Transformationsmatrix $C \in \mathbb{R}^{p\times k}$ berechnet man die Matrix $X \in \mathbb{R}^{k\times n}$ der latenten Vektoren, die Sitz eines der in (5.13) erwähnten Dirac-Peaks sind:

$$X = (C^TC)^{-1}C^TY \quad (5.14)$$

**M-Schritt:** Im M-Schritt muß unter Annahme des soeben berechneten $X$ diejenige Parametermatrix $C$ berechnet werden, die die Likelihood der Daten $Y$ maximiert.

Dazu muß die Matrix $C$ gefunden werden, die jedes der berechneten $x$ möglichst nahe an das zugehörige $y$ abbildet. Auch hier wird die Pseudoinverse benutzt:

$$C^{neu} = YX^T(XX^T)^{-1} \quad (5.15)$$

Für $k = 1$, wenn wir nur eine Hauptkomponente suchen, kann man den Algorithmus mit der folgenden physikalischen Analogie veranschaulichen: Wir haben einen frei drehbaren Stab, der auf dem Ursprung fixiert ist und der unsere Hauptkomponente symbolisiert. Sei eine beliebige Orientierung für den Stab gewählt, so bewirkt eine Iteration des EM-Algorithmus, daß wir zuerst (im E-Schritt) jeden Datenpunkt auf den Stab projizieren. Im M-Schritt wird jeder Punkt mit seiner Projektion durch eine Sprungfeder verbunden. Wenn wir dann den Stab loslassen, dann wird er die Orientierung annehmen, die für die momentane Projektion der Daten die niedrigste Energie in den Strungfedern besitzt; die Energie entspricht also der Log-Likelihood. In der nächsten Iteration projizieren wir die Daten erneut, minimieren wieder den Fehler, und so fort.

### 5.2.2 Fehlende Daten in der EM-PCA

Im E-Schritt wurde bislang für jeden Datenpunkt $y$ dasjenige $x^*$ gesucht, das den Abstand $\|Cx^* - y\|$ minimiert. Wenn nun in $y$ einige Komponenten fehlen, so muß dieses Problem erweitert werden: Nun müssen für jedes $y$ zusätzlich noch die fehlenden Komponenten optimal bestimmt werden; $y$ wird vervollständigt zu dem $y^*$, für das mit dem nach wie vor ebenfalls zu bestimmenden $x^*$ der Rekonstruktionsfehler $\|Cx^* - y^*\|$ minimiert wird.[1]

Der M-Schritt wird wie gehabt ausgeführt, wobei zuvor in $Y$ die ermittelten Werte der $y^*$ eingetragen werden müssen.

[1] Dies kann man z. B. lösen durch QR-Zerlegung [PTVF92].

Dieser Algorithmus findet zwar nicht die Eigenvektoren und Eigenwerte, aber eine Basis $C$ für den Unterraum, in dem die Varianz der Daten am größten ist. Anschließend kann man die Daten auf diesen Unterraum projizieren und dort eine herkömmliche PCA durchführen, die dann sehr viel weniger aufwendig und nicht mit fehlenden Daten gestraft ist.

Dieselbe Idee, einen EM-Algorithmus für die PCA zu entwerfen, findet sich auch in [TB99]. Dort wird jedoch nicht vorgeschlagen, den Algorithmus tatsächlich anzuwenden, sondern es wird vor allem auf die Vorteile des Wahrscheinlichkeitsdichtemodells eingegangen.

## 5.3 Entfernung der Dämpfung durch Projektion

Wenn an demselben Ort ein Gespräch im Auto und eins auf der Straße geführt wird, so werden die Pegelwerte sich wegen der unterschiedlichen Dämpfung stark unterscheiden (siehe Abschnitt 3.2.2). Es wäre jedoch wünschenswert, den beiden Gesprächen einen kleinen Abstand zuzuordnen, da sie im Ortsraum nahe beieinander stattgefunden haben.

In [Qua97] wird auf Abstandsmaße eingegangen, die dieses Problem behandeln. Es werden zwei Methoden vorgeschlagen, die Dämpfung herauszurechnen: Abstand zum Server („Salami-Metrik“) und Means-Metrik.

### 5.3.1 Der Abstand zum Server

Wenn zwei Gespräche im Prinzip die gleichen Pegel empfangen, jedoch ein verschiedenes $\lambda$ haben, so bedeutet die Modellannahme, daß dieses $\lambda$ von allen Pegeln abgezogen wird. Demnach werden die Abstände der Pegelwerte voneinander konstant bleiben.

Dies motiviert das Abstandsmaß, das die Distanz aller Pegel zum Serverpegel berechnet und erst dann die euklidische Metrik benutzt:

$$d(x,y) = \sqrt{\sum_{d=2}^{D}((x_d - x_1) - (y_d - y_1))^2} \tag{5.16}$$

Dies entspricht einer Verschiebung der Vektoren entlang der Richtung $(1, 1, \ldots, 1)$ – also der $\lambda$-Richtung – und einer Projektion auf die Hyperebene $x_1 = 0$, auf der dann die Abstände bestimmt werden. (Diese Verschiebung und das Abschneiden an der Hyperebene $x_1 = 0$ erzeugt ein Gebilde, das einer Salami ähnelt; in [Qua97] heißt diese Metrik daher „Salami-Metrik“.)

Diese Methode zeichnet die Serverkomponente $x_1$ aus, da sie die einzige ist, die garantiert einen Meßwert enthält. Da aber diese Sonderbehandlung sonst nicht plausibel ist, schlägt [Qua97] auch eine Variante vor, die etwas symmetrischer ist.

### 5.3.2 Die Means-Methode

Anstatt die Datenpunkte auf die Hyperebene $x_1 = 0$ zu projizieren, kann man sie auch auf die Hyperebene $\sum_d x_d = 0$ projizieren, die genau orthogonal zur $\lambda$-Achse liegt.

Dazu muß von jeder Komponente des Vektors der Durchschnitt aller Vektorkomponenten abgezogen werden. Es ist klar, daß dann die Gleichung der Hyperebene erfüllt wird.

Das daraus folgende Abstandsmaß (die *Means-Methode*) ist dann

$$d(x, y) = \sqrt{\sum_{d=1}^{D}((x_d - \bar{x}) - (y_d - \bar{y}))^2} \tag{5.17}$$

mit

$$\bar{x} = \frac{1}{D}\sum_{d=1}^{D} x_d \qquad \bar{y} = \frac{1}{D}\sum_{d=1}^{D} y_d$$

Dadurch werden Verzerrungen, die durch die Auszeichnung der Serverkomponente entstehen, verhindert.

## 5.4 Ein vereinfacht dreidimensionales GTM-Modell

### 5.4.1 Das semilineare Modell

Üblicherweise ist der latente Raum der GTM-Karte von kleiner Dimension, meistens zweidimensional. Im Prinzip ist die Dimension des latenten Raumes beliebig, aber einerseits läßt sich das Ergebnis bei großem $L$ nicht mehr gut visualisieren, und andererseits ist die Laufzeit des Algorithmus exponentiell in $L$. Die Anzahl der Gaußblobs und der Basisfunktionen vervielfacht sich mit jeder latenten Dimension.

Es gibt aber auch vereinfachte Modelle höherdimensionaler Karten. Von Tipping und Bishop stammt eine probabilistische Hauptkomponentenanalyse (PPCA) [TB99], die fast identisch ist mit dem Modell von Roweis [Row97], das in Abschnitt 5.2 dargestellt ist.

Sie weist einige Parallelen zu GTM auf: Die Daten werden in einen latenten Raum projiziert, es wird ein Wahrscheinlichkeitsdichtemodell aufgestellt, und die Parameter werden mit einem EM-Algorithmus trainiert. Der Unterschied besteht darin, daß das Mapping linear ist; das Ergebnis ist eine Hauptkomponentenanalyse der Daten. Gegenüber der herkömmlichen Berechnung der PCA hat dieses Modell jedoch den Vorteil, daß es schneller konvergiert und auch leicht erweiterbar ist, z. B. in Hinblick auf fehlende Daten.

Diese PPCA ist so verwandt mit dem GTM-Algorithmus, daß sich die beiden Modelle recht einfach verbinden lassen [BSW98, Sve98]. Diese Verbindung nennt man das *semilineare GTM-Modell*. Es wird so genannt, weil die latenten Variablen zum Teil linear, zum Teil – wie bei GTM – diskretisiert und als Linearkombination von Basisfunktionen in den Datenraum abgebildet werden.

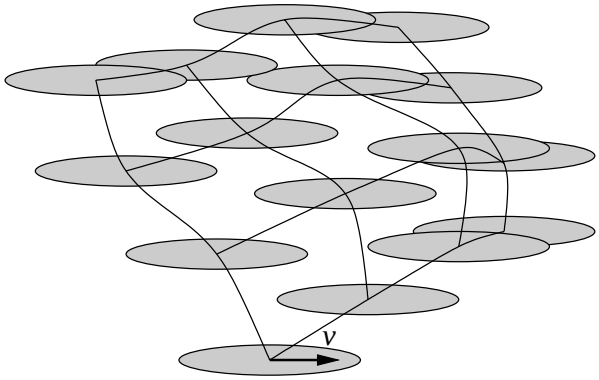



Abbildung 5.1: Das semilineare GTM-Modell: Hier eine zweidimensionale Karte ($L = 2$) mit einer zusätzlichen linearen Dimension ($Q = 1$). Die Gaußblobs sind nicht mehr rund, sondern in Richtung der linearen Komponente (des Vektors $v$) gestreckt.

Eine andere Sichtweise dieses Modelles ist die, daß die GTM-Karte niedrigdimensional bleibt – es gibt weiterhin ein beispielsweise zweidimensionales Gitter von Gaußblobs. Zusätzlich enthält das Wahrscheinlichkeitsmodell jedoch noch Varianz entlang der linearen Achsen. Sei $V \in \mathbb{R}^{D \times Q}$, wobei $Q$ die Anzahl der linearen Komponenten ist. $V$ enthalte als Spalten die Richtungsvektoren dieser linearen Achsen, skaliert entsprechend der Varianz entlang dieser Achsen.

Unsere Gaußblobs sind dann nicht mehr rund mit Varianz $\beta^{-1}$, sondern in diese Richtungen gestreckt (siehe Abbildung 5.1). Dadurch verändert sich die Wahrscheinlichkeitsverteilung; sie enthält nicht mehr den quadratischen Abstand zwischen Datenpunkt und Gaußblob (Gleichung 4.1), sondern man muß nun die Mahalanobis-Distanz [Mah36] benutzen:

$$p(t|W, V, \beta) = \frac{(2\pi)^{-D/2}}{K\sqrt{\det C}} \sum_{k=1}^{K} \exp\left\{-\frac{1}{2}(t - y(x_k, W))C^{-1}(t - y(x_k, W))^T\right\} \tag{5.18}$$

Dabei ist

$$C = \beta^{-1} I + VV^T \tag{5.19}$$

### 5.4.2 Der semilineare Lernalgorithmus

Diese Erweiterung läßt sich elegant in den GTM-Lernalgorithmus einbauen [Sve98]:

- Der E-Schritt besteht nach wie vor aus der Berechnung der Responsibilities (vgl. Formel 4.10), nur muß natürlich für die Wahrscheinlichkeitsdichte Formel 5.18 benutzt werden.

- Der M-Schritt für die Gewichtsmatrix $W$ bleibt, wie er ist. Es schließt sich jedoch noch der M-Schritt für die linearen Komponenten in $V$ an.

Wie in [Sve98] (S. 86) ausgeführt wird, besteht der M-Schritt für die linearen Komponenten zunächst aus der Berechnung der gewichteten Kovarianzmatrix der Daten:

$$S = \frac{1}{N} \sum_{n,k} r_{kn}(t_n - y(x_k, \tilde{W}))^T (t_n - y(x_k, \tilde{W}))$$

Dabei ist $\tilde{W}$ die soeben aktualisierte Gewichtsmatrix.

Eine Eigenwertanalyse dieser Matrix $S$ ergibt die Eigenvektorenmatrix $U \in \mathbb{R}^{D \times Q}$ und die $Q$-dimensionale Diagonalmatrix $\Lambda$, die die $Q$ größten Eigenwerte $\lambda_1, \ldots, \lambda_Q$ enthält. Dann ist die Maximum Likelihood-Lösung für $V$ gegeben durch

$$V = U(\Lambda - \beta^{-1} I)^{1/2}$$

Der neue Wert für $\beta$ ist schließlich die durchschnittliche Varianz der Daten in den restlichen $D - Q$ Dimensionen, die nicht von den linearen Komponenten abgedeckt wird.

$$\beta^{-1} = \frac{1}{D - Q} \sum_{d=Q+1}^{D} \lambda_d$$

### 5.4.3 Anwendung des semilinearen Modells auf unser Problem

Wie kann uns dieses semilineare Modell nützen? Wir brauchen zwar keinen hochdimensionalen latenten Raum, aber die Dimensionsanalyse aus Abschnitt 4.3.1 bestätigt, daß unsere Daten dreidimensional sind. Es ist eine grundlegende Modellannahme, daß die Feldstärkedaten nicht nur vom Ort des Gesprächs abhängen, sondern in der dritten Dimension von der Dämpfung $\lambda$ (siehe Abschnitt 3.2.2).

Da $\lambda$ von jeder Komponente des Feldstärkevektors abgezogen wird, entspricht es einer linearen Verschiebung entlang der Hauptachse $(1, 1, \ldots, 1)$. Gerade eine solche lineare Varianz wird im semilinearen GTM-Modell eingefügt. Auch ist es sehr einfach, die lineare Komponente herauszuprojizieren, indem man die Positionen der herkömmlichen Gaußblobs betrachtet.

Auf diese Weise kann man also die dritte, lineare $\lambda$-Dimension modellieren, ohne auf die Möglichkeit einer zweidimensionalen Darstellung verzichten zu müssen. Diese Variante hat gegenüber den Projektionsmethoden aus Abschnitt 5.3 den Vorteil, daß die Dämpfungsachse durch den Lernalgorithmus optimiert wird. Dies betrifft nicht nur die Richtung (die resultierende Achse zeigt zwar in den richtigen Quadranten, jedoch nicht genau in die Richtung $(1, 1, \ldots, 1)$), sondern auch die Länge, die der Standardabweichung in $\lambda$-Richtung angepaßt wird.

Eine Alternative hierzu ist, stattdessen die Modellannahme zu akzeptieren, sich den Lernschritt für $V$ sparen und den Vektor auf der $(1, 1, \ldots, 1)$-Richtung festhalten.

## 5.5 Das Clustertrennungsmaß

Im Rahmen unserer Meßaktion haben wir an verschiedenen Stellen innerhalb der Zelle selbst Telefonate geführt. Ein Ziel des Trainings ist, die verschiedenen Gespräche an einem Ort auf der Karte nahe beieinander zu legen, aber auch die Gespräche an verschiedenen Orten sauber voneinander zu trennen.

Dies zu überprüfen erfordert ein quantitatives Maß, mit dem verschiedene Karten in dieser Hinsicht objektiv verglichen werden können. In [DB79] wurde ein Maß für die Trennung von Clustern eingeführt, das auf unser Problem angewandt werden kann.

Seien die Daten aus dem Raum $M$. $M$ muß ein metrischer Raum sein, d. h. wir benötigen eine Metrik $d : M \times M \to \mathbb{R}$. Eine Metrik hat die folgenden Eigenschaften:

1. $d(x, x) = 0$
2. $d(x, y) = d(y, x) \neq 0$ für $x \neq y$
3. $d(x, z) \leq d(x, y) + d(y, z)$ (Dreiecksungleichung)

(Daraus folgt, daß $d(x, y) \geq 0$.)

In [DB79] wird zunächst ein Maß für die Dispersion (Streuung) eines Clusters motiviert:

$$S_i = \left( \frac{1}{T_i} \sum_{j=1}^{T_i} |X_j - A_i|^q \right)^{1/q} \tag{5.20}$$

Dabei ist $T_i$ die Anzahl der Punkte $X_j$ in Cluster $i$, $A_i$ ihr Schwerpunkt und $q$ ein Parameter; ist etwa $q = 1$, dann ist $S_i$ der durchschnittliche Abstand der Punkte vom Schwerpunkt, für $q = 2$ ist $S_i$ die Standardabweichung des Abstandes. Für die vorliegende Arbeit wurde $q = 2$ gewählt.

Des weiteren benötigen wir eine Matrix, die die Abstände der Clusterschwerpunkte voneinander enthält:

$$M_{ij} = \left( \sum_{k=1}^{n} |A_{ik} - A_{jk}|^p \right)^{1/p} \tag{5.21}$$

Hierbei gibt $p$ an, welche Norm benutzt wird. Für $p = 2$ haben wir z. B. die euklidische Norm, $p = 1$ ist die „Manhattannorm“.

Als Maß für die Trennung der Cluster $i$ und $j$ wird dann vorgeschlagen:

$$R_{ij} = \frac{S_i + S_j}{M_{ij}} \tag{5.22}$$

Das gesamte Clustertrennungsmaß schließlich ist dann der Durchschnitt über alle Cluster der Trennung vom nächsten Nachbarcluster:

$$\bar{R} = \frac{1}{N} \sum_{i=1}^{N} \max_{j \neq i} R_{ij} \tag{5.23}$$

Dieses $\bar{R}$ gilt es zu minimieren. $\bar{R} = 0$ bedeutet perfekte Trennung der Cluster ohne jedwede Streuung.

## 5.6 Das topographische Produkt

Das topographische Produkt ist ein Maß für die Nachbarschaftserhaltung einer selbstorganisierenden Karte. In erster Linie war es für den SOM-Algorithmus gedacht, kann aber genausogut auch auf GTM angewendet werden.

Das topographische Produkt wurde in [BP92] eingeführt. Im Prinzip ist es das Verhältnis der Abstände der Nachbarn eines Neurons auf der Karte und im Datenraum. Die folgende Beschreibung ist [BP92] entnommen, jedoch auf unsere Terminologie übertragen.

### 5.6.1 Herleitung des topographischen Produktes

Wir beginnen mit einer Notation für den $j$-ten Nachbarn eines latenten Vektors $x_k, k \in \{1, \ldots, K\}$. Diesen nennen wir $n_j^L(k)$ (wobei das $L$ für „latenter Raum“ steht).

$$\begin{aligned} n_1^L(k): \qquad d^L(k, n_1^L(k)) &= \min_{k' \in \{1,\ldots,K\} \setminus \{k\}} d^L(k, k') \\ n_2^L(k): \qquad d^L(k, n_2^L(k)) &= \min_{k' \in \{1,\ldots,K\} \setminus \{k, n_1^L(k)\}} d^L(k, k') \\ &\vdots \end{aligned}$$

Dabei ist $d^L(k, k') = \|x_k - x_{k'}\|$ der euklidische Abstand zweier Vektoren $x_k$ und $x_{k'}$ im latenten Raum, d. h. auf der Karte.

Ganz analog definieren wir die Nachbarn ihrer Bildvektoren im Datenraum. Wir nennen $n_j^D(k)$ den $j$-ten Nachbarn von $y_k = y(x_k, W)$ im hochdimensionalen Datenraum.

$$\begin{aligned} n_1^D(k): \qquad d^D(k, n_1^D(k)) &= \min_{k' \in \{1,\ldots,K\} \setminus \{k\}} d^D(k, k') \\ n_2^D(k): \qquad d^D(k, n_2^D(k)) &= \min_{k' \in \{1,\ldots,K\} \setminus \{k, n_1^D(k)\}} d^D(k, k') \\ &\vdots \end{aligned}$$

Dabei ist $d^D(k, k') = \|y_k - y_{k'}\|$ der euklidische Abstand im Datenraum.

Anschließend werden die folgenden Quotienten definiert:

$$Q_1(k, j) = \frac{d^D(k, n_j^L(k))}{d^D(k, n_j^D(k))} \qquad Q_2(k, j) = \frac{d^L(k, n_j^L(k))}{d^L(k, n_j^D(k))}$$

Diese sind genau dann 1, wenn der $j$-te Nachbar im latenten Raum gleich dem im Datenraum ist. Um Verzerrungen und den Einfluß verschiedener Magnification Factors auszugleichen, wird nun das Produkt dieser Werte bis zu einer Ordnung $i$ gebildet:

$$P_3(k,i) = \left( \prod_{j=1}^{i} Q_1(k,j) Q_2(k,j) \right)^{\frac{1}{2i}}$$

Dieses Maß detektiert nun nur noch Nachbarschaftsverletzungen ab einer Größenordnung von $i$ Nachbarn; selbst wenn die näheren Nachbarn in einer anderen Reihenfolge stehen sollten, kürzen sich ihre Abstände doch heraus.

Das topographische Produkt ist dann der Durchschnitt von $\log P_3$ über alle latenten Vektoren und Nachbarschaftsordnungen:

$$\begin{aligned} P &= \frac{1}{K(K-1)} \sum_{k=1}^{K} \sum_{i=1}^{K-1} \log P_3(k,i) \qquad (5.24) \\ &= \frac{1}{K(K-1)} \sum_{k=1}^{K} \sum_{i=1}^{K-1} \frac{1}{2i} \sum_{j=1}^{i} \log \left( Q_1(k,j) Q_2(k,j) \right) \end{aligned}$$

### 5.6.2 Anpassung an unser Problem

Das topographische Produkt war das erste Topologieerhaltungsmaß, das für das SOM-Verfahren vorgeschlagen wurde. Später wurde es kritisiert [VDM94, Kiv96], weil es die Verteilung der Trainingsdaten im Datenraum nicht berücksichtigt (siehe Abb. 5.2). Eine Faltung der Karte im Datenraum – die als Nachbarschaftsverletzung erkannt wird – kann verkehrt sein, weil sie eine höherdimensionale Datenmenge auszuschöpfen versucht, sie kann aber auch korrekt sein, weil die Datenmenge in einer ebensolchen Faltung im Datenraum liegt.

Es ist jedoch möglich, das Problem zu umgehen. Anstatt die Nachbarschaftserhaltung der Abbildung

$$\text{Pegelraum} \longrightarrow \text{Karte}$$

zu messen, benutzen wir die Abbildung

$$\text{Zelle} \longrightarrow \text{Pegelraum} \longrightarrow \text{Karte},$$

die uns vermittels der Prädiktions- oder Blackbox/Scankarten gegeben ist. Während die Daten im Pegelraum auf nichtlineare Weise eingebettet sind, liegen sie in der Zelle trivialerweise auf einer linearen, zweidimensionalen Ebene. In der Zelle gibt es keine Faltungen oder Schleifen, die die Karte nachbilden müßte.

Das originale topographische Produkt betrachtet die Abbildung zwischen den latenten Vektoren $x_k$ (bzw. den Neuronen einer SOM) und ihren Bildern $y_k$. Wir betrachten die Abbildung zwischen den Unterrechtecken der Prädiktionskarte (s. Abb. 4.4) und den Bildern der zugehörigen $y_k^{pred}$ auf der GTM-Karte. Diese haben nichts mehr mit den $y_k$, den Bildern der latenten Vektoren, zu tun – was das Gütemaß aber nicht stört.

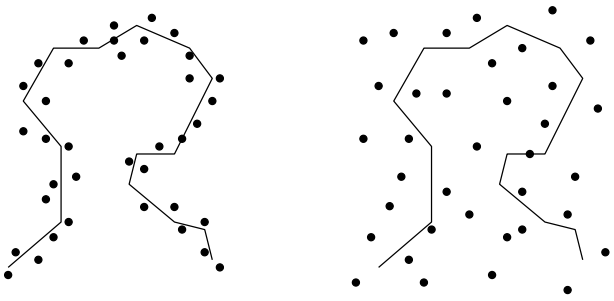


Abbildung 5.2: Das linke Netz ist gut trainiert, es paßt sich der Datenmenge genau an. Das rechte Netz ist schlecht: Es versucht, eine zweidimensionale Menge eindimensional auszuschöpfen, was zu Topologieverletzungen führt. Das topographische Produkt kann diese beiden Fälle nicht unterscheiden, da sich nicht die Netze, sondern nur die trainierten Daten unterscheiden. In beiden Fällen wird diese Schleife als Topologieverletzung gewertet.

### 5.6.3 Andere Topologieerhaltungsmaße

Aufgrund der Mängel des topographischen Produkts wurden noch mancherlei andere Maße für das SOM-Vefahren entwickelt.

In [VDM94] wurde die „topographische Funktion“ eingeführt, die prüft, wie weit zwei Neuronen einer SOM, deren rezeptive Felder im Datenraum benachbart sind, auf der Karte auseinander liegen. Dieses Maß hat jedoch den Nachteil, daß es keine skalare Zahl ist. Vielmehr muß man den Verlauf einer Kurve betrachten. Das erschwert den Vergleich zweier topographischer Funktionen verschiedener Karten. [Kiv96]

In [Kiv96] wird als Maß der „topographische Fehler“ vorgeschlagen. Dieser zählt die Anzahl der Datenpunkte, deren nächstes und zweitnächstes Neuron auf der Karte nicht benachbart liegen. Er betrachtet jedoch leider nicht das *Ausmaß* der Nachbarschaftsverletzung. [KL96]

In [KL96] wird als Gütemaß der Abstand jedes Datenpunktes vom nächsten $y_k$ plus der Länge des kürzesten Weges von $y_k$ zum zweitnächsten $y_{k'}$ auf dem Gitter der Codebook-Vektoren propagiert. Dieses Maß hat jedoch für unser Problem zwei Nachteile:

- Der Abstand des nächsten Vektors $y_k$ und des zweitnächsten $y_{k'}$ kann bei SOM durch Betrachtung aller Wege auf dem Gitter zwischen diesen beiden Codebook-Vektoren ermittelt werden.

  Natürlich kann man auch bei einer GTM-Karte so vorgehen. Da diese beiden Vektoren jedoch auf der kontinuierlichen $L$-dimensionalen Mannigfaltigkeit im $\mathbb{R}^D$ liegen, wäre es wünschenswert, den kürzesten Weg auf der Mannigfaltigkeit zu benutzen und nicht den entlang des Gitters der Codebook-Vektoren. Dazu müßte die Geodätische zwischen den beiden Vektoren berechnet werden, was den Rahmen dieser

Arbeit sprengen würde. (Vermutlich ist dies nur mit numerischen Methoden möglich.)

- Der Abstand jedes Datenpunktes vom nächsten $y_k$ (der den Quantisierungsfehler der Karte messen soll) ist abhängig davon, wie wir die Dämpfung und die fehlenden Komponenten des Datenvektors behandeln. So liefern die Abstände mit fehlenden Komponenten (5.1), die Means-Metrik (5.17) oder die Mahalanobis-Distanz (5.18) nicht vergleichbare Werte. Aus diesem Grunde sind verschiedene Karten, die aus verschiedenen Modellen hervorgegangen sind, anhand dieses Maßes nicht vergleichbar.

Vor diesem Hintergrund erscheint es nicht als Nachteil, sondern als Vorteil, daß das topographische Produkt die Datenpunkte nicht betrachtet, denn so ist es unabhängig vom verwendeten Datenmodell. Da wir das Hauptproblem des topographischen Produktes durch die Koppelung mit der Prädiktionsabbildung behandelt haben, ist es das Maß, das am geeignetsten erscheint, den Lernerfolg unserer Netze zu überprüfen.

## 5.7 Vorverarbeitung der Gespräche

Wie schon in [Qua97] dargelegt wurde, ist es in unserem Kontext unerläßlich, die Gespräche im Vorfeld durch Mittelung zusammenzufassen. Einerseits ist die Streuung der einzelnen $A_{bis}$-Vektoren sehr groß. Durch Mittelung können insbesondere Schwankungen, die durch Kurzzeit-Fading entstanden sind, entfernt werden. Außerdem hängt die Laufzeit und der Speicherbedarf linear von der Datenmenge ab; diese können also durch Zusammenfassung der $A_{bis}$-Vektoren deutlich verbessert werden. Schließlich kann man das Problem der fehlenden Daten mindern, da zu verschiedenen Zeiten des Gesprächs manchmal verschiedene Sender gemeldet werden, so daß der gemittelte Vektor durchaus mehr als sechs Nachbarn enthalten kann.

### 5.7.1 Mittelung über disjunkte Intervalle

Wie auch in [Qua97] dargelegt, ist die Mittelung über ganze Gespräche nicht sinnvoll. Dabei würde aus jedem Gespräch genau ein Vektor erzeugt, so daß der Unterschied zwischen einem kurzen und einem langen Gespräch verwischt würde. Dies würde die Verkehrsdichteinformation stark verfälschen.

Deshalb ist es besser, über Intervalle zu mitteln. Bei der Wahl der Intervallgröße ist jedoch verschiedenes zu beachten:

- Um die Laufzeit des Algorithmus und die Vollständigkeit der Vektoren zu verbessern, ist eine möglichst große Intervallgröße sinnvoll.

- Andererseits wird die Intervallgröße nach oben beschränkt durch bewegte Calls. Ein Handy, das mit 30 $^{m}/_{s}$, also 108 $^{km}/_{h}$ bewegt wird, legt in 24 s (was einer Intervallgröße von 50 entspricht) schon 720 m zurück. Dies ist bei einem Zelldurchmesser von 5 bis 7 km durchaus signifikant.

  Im allgemeinen sind nur wenige so schnell bewegte Calls zu erwarten. Daher wurde als Intervallgröße 50 gewählt. Wenn jedoch eine Zelle eine Autobahn enthält, kann sich ein kleinerer Wert lohnen. Deshalb wurde in Abschnitt 7.4.1 für eine solche Zelle die Intervallgröße auf 20 gesetzt.

### 5.7.2 Mittelungsmethode

In [Qua97] wurde zur Mittelung der arithmetische Mittelwert gewählt. Dies hat jedoch den Nachteil, daß nicht gemeldete Feldstärken – die als Null aufgefaßt wurden – das Ergebnis stark verfälschen könnten. Inzwischen wurden viele Varianten zur Zusammenfassung von Gesprächen untersucht [Sch00]. So kann man zum Beispiel den Mittelwert durch Fitten einer „abgeschnittenen Gaußverteilung“ bilden. Bei der Ortszuordnung hingegen scheint sich zur Zusammenfassung der Durchschnitt der zehn größten Werte des Intervalls zu bewähren.

Um den Aufwand zu reduzieren, wurde an dieser Stelle der Median der gemessenen Daten gewählt. Der Median ist unempfindlich gegenüber Ausreißern. Er behandelt die Peaks am Rand des Wertebereiches (bei 0 und bei 63) auf kanonische Art und Weise, und er ist einfacher zu ermitteln als das Fitten an eine abgeschnittene Gaußkurve. Das Argument aus [Qua97] (S. 53), daß der Median angesichts des hohen Dämpfungsanteils unbrauchbare Werte liefere, ist insofern belanglos, als die Dämpfung $\lambda$ für ein Gespräch als konstant angenommen wird.

### 5.7.3 Mindestzahl von Nachbarn

Manche Handys sind beim Melden von Nachbarsendern äußerst nachlässig. Da diese Daten kaum aussagekräftig sind, wurden Vektoren, die nicht mindestens drei Werte enthielten (den Server und zwei Nachbarn), verworfen.

### 5.7.4 Implementierung der Mittelung

$A_{bis}$-Dateien können von LocAbis [Hei00] zuverlässig eingelesen werden, so daß es nicht nötig war, einen eigenen Parser zu schreiben. Auch ist LocAbis so objektorientiert angelegt, daß es möglich war, ein Modul einzubauen, das die einzelnen Gespräche in Intervallen zusammenfaßt und als T-Vektor-Dateien abspeichert.

Das T-Vektor-Format enthält in jeder Zeile die Pegelwerte des Servers und aller Nachbarn, dahinter zwei reservierte Zahlfelder, die für $t_{min}$ und *BsRed* (s. S. 11) vorgesehen waren, von der GTM-Software jedoch nicht benötigt werden. Anschließend kann noch

ein optionaler Buchstabe folgen, der den Spot angibt, dem das Gespräch entstammt (a = Spot 1, b = Spot 2 und so fort).

Das LocAbis-Modul, das diese Dateien erzeugt, heißt „AggregateLocator“ und wird mit dem Parameter `-map aggregate` aktiviert. Zusätzlich zu den herkömmlichen Kommandozeilenoptionen von LocAbis akzeptiert dieser Modus drei weitere Parameter:

- `-aggregate (Intervallgröße)` stellt die Länge der zusammenzufassenden Intervalle ein. (Default: das gesamte Gespräch)
- `-least (Mindestzahl)` gibt die Mindestzahl an empfangenden Pegeln (inklusive Server) an, den ein Vektor enthalten muß, um abgespeichert zu werden. (Default: 1, d. h. alle Vektoren werden abgespeichert)
- `-spot (Buchstabe)` übergibt einen Buchstaben, der die abgespeicherten Vektoren einem Spot zuordnet. (Default: kein Buchstabe)

Um zum Beispiel aus den Massendaten einer BTS eine T-Vektor-Datei zu erzeugen, genügt ein Befehl wie:

```
LocAbis -o masse41a_18.t -n /home/robin/da/eudat/praed/Euskirchen
 -s 33711 020013 -map aggregate e41a_18.abn -i off -aggregate 50 -least 3
```

Durch diesen Befehl wird die Datei `e41a_18.abn` eingelesen, die Gespräche mit Intervallgröße 50 zusammengefaßt und die T-Vektor-Datei `masse41a_18.t` erzeugt.

Für die Verarbeitung eines Gesprächs, das einem bekannten Spot entstammt, sollte man LocAbis zusätzlich den Buchstaben des Spots übergeben:

```
LocAbis -o callse41a.t -n /home/robin/da/eudat/praed/Euskirchen
 -s 33711 020013 -map aggregate /home/robin/da/eudat/calls/c01.ab1
 -i off -aggregate 50 -least 3 -spot a
```

### 5.7.5 Mischen der Daten

Falls beim Training der Daten Early Stopping verwendet wird (siehe Abschnitt 4.5.3), werden die Daten in Trainings-, Validierungs- und Testdaten gespalten. Wenn man dann etwa die $A_{bis}$-Daten, die an einem Tag in einer Zelle aufgezeichnet wurden, trainieren möchte, wird das Netz anhand der Daten vom Vormittag trainiert, anhand der Daten vom Mittag validiert und anhand der Daten vom Nachmittag beurteilt. Diese können jedoch wesentlich verschieden voneinander sein. Deshalb ist es nicht ratsam, die chronologische Reihenfolge beizubehalten.

Die einfachste Möglichkeit, solche Tendenzen aus den Daten zu entfernen, ist, die Datensätze zu mischen. Dazu habe ich ein kleines C-Programm namens *mischezeilen* geschrieben, das eine T-Vektor-Datei einliest, die Zeilen zufällig vermischt und abspeichert.

# 6 Die GTM-Software

Im Rahmen der Diplomarbeit wurde der GTM-Algorithmus inklusive der auf das Problem zugeschnittenen Varianten in C++ implementiert. In diesem Kapitel möchte ich auf interessante Aspekte und die Handhabung der Software eingehen. Die Klassenstruktur und die Dateiformate werden in Anhang A dargestellt.

Zu Beginn wird in Abschnitt 6.1 die graphische Benutzeroberfläche der Anwendung *xgtm* dargestellt. Es folgen die nicht-graphischen Hilfsprogramme *gtmtopng* und *execplaene*.

## 6.1 Die graphische Benutzeroberfläche von *xgtm*

Die Anwendung *xgtm* dient dazu, eine GTM-Karte auf eine Datenmenge zu trainieren oder aus einer Datei zu laden, das Ergebnis graphisch darzustellen oder abzuspeichern. Dazu ist mit Hilfe der Widget-Bibliothek *Qt* [Dal99] eine X11-Benutzeroberfläche entstanden, die im folgenden dargestellt wird.

### 6.1.1 Die Fenster

Das Hauptfenster von *xgtm* enthält eine Menüzeile, einige Knöpfe und eine Statuszeile. Des weiteren stellt es die GTM-Karte dar, projiziert auf zwei wählbare Dimensionen. Für $A_{bis}$-Daten ist das natürlich selten sinnvoll, da durch die Projektion viel Struktur verloren geht und nur wenig zu erkennen ist. Für niedrigdimensionale Testdatenmengen ist diese Darstellung jedoch hilfreich.

Durch das *Fenster*-Menü lassen sich zwei Fenster zur Darstellung der GTM-Karte öffnen oder schließen (siehe Abschnitt 4.6):

**Magnification Factors:** Dieses Fenster stellt in Graustufen die Magnification Factors der Karte dar. Schwarz sind niedrige, weiß hohe Faktoren.

**Wahrscheinlichkeitsdichte:** Dieses Fenster stellt in Graustufen die Wahrscheinlichkeitsdichte $p(y(x, W)|W, \beta)$ dar. Schwarz bedeutet hohe, weiß niedrige Wahrscheinlichkeit.

Die Skalen sind gegensätzlich zueinander angeordnet, da ein niedriger Magnification Factor einer hohen Wahrscheinlichkeitsdichte entspricht. So bedeutet in beiden Fenstern ein dunkler Fleck einen Hotspot, und die Darstellungen lassen sich gut vergleichen.

Ansonsten haben die beiden Fenster die gleichen Eigenschaften. Sie stellen den latenten Raum dar. Wenn im Fenster ein Punkt $x$ angeklickt wird, wird der zugehörige Datenvektor $y(x, W)$ im Hauptfenster markiert (natürlich projiziert auf die dort gewählten Dimensionen) und in der Legende des Fensters angezeigt, dazu der Funktionswert $\frac{dA_y}{dA_x}(x)$ bzw. $p(y(x, W)|W, \beta)$. Die Skala justiert sich bei jedem Maldurchgang so, daß die Extremwerte auf die Farben Schwarz und Weiß abgebildet werden. Es gibt aber auch einen Menüpunkt, der die Skalen abschätzt, indem die Werte eines kleineren Samples von Pixeln betrachtet werden.

Des weiteren können in den Fenstern die Datenpunkte auf der Karte dargestellt werden. Man hat die Wahl zwischen den Trainingsdaten, den Validierungsdaten und den Testdaten (sofern Early Stopping verwendet wurde; s. Abschnitt 4.5.3); darüber hinaus kann man auch eine vierte Datenmenge, die unabhängig von diesen ist und einer anderen Datei entstammt, anzeigen lassen, die sogenannten „Maldaten“. Diese heißen so, weil sie hauptsächlich zu ebendiesem Zweck dienen, in den Fenstern angezeigt zu werden.[1] Dies können etwa Daten sein, die ausschließlich den bekannten, selbsterzeugten Hotspots entstammen.

### 6.1.2 Die Menüs

Im Dateimenü finden sich die folgenden Menüpunkte:

**Neue GTM** Dieser Punkt öffnet ein Fenster, in dem man die Trainingsdaten und die Parameter einer GTM-Karte angeben kann (siehe Abbildung 6.1). Die Karte wird dann mit den gegebenen Parametern initialisiert.

**Speichern** Speichert die aktuelle GTM-Karte in einer Datei.

**Laden** Lädt eine GTM-Karte aus einer Datei.

**Beenden** Beendet *xgtm*.

Das Training-Menü enthält die Befehle für das Trainieren der Karte und einige statistische Algorithmen:

**Start** Startet das Training der GTM-Karte.

**Stop** Unterbricht das Training der GTM-Karte (auch mit der ESC-Taste erreichbar).

**Schritt** Vollführt einen einzelnen Trainingsschritt.

**Parameter ändern** Dieser Punkt öffnet ein Fenster, in dem man einige Parameter der GTM-Karte manipulieren kann.

[1] Die andere Aufgabe der Maldaten ist die Berechnung des Clustertrennungsmaßes (siehe Abschnitt 5.5).

**Lade Prädiktionskarte** Lädt Prädiktionsdaten. Diese werden für die Berechnung des topographischen Produkts benötigt.

**Topographisches Produkt** Berechnet das topographische Produkt der aktuellen Karte.

**Dimensionplot** Erzeugt eine von Gnuplot lesbare Datei, die ein Histogramm gemäß Dimensionsalgorithmus (Abschnitt 4.3.1) enthält.

**Clusterseparierung der Maldaten ausgeben** Gibt den Wert der Clustertrennung (Abschnitt 5.5) für die Cluster in den Maldaten aus.

Im Ansicht-Menü befinden sich die Optionen für die Darstellung der Daten in den Fenstern:

**Gezeigte Dimensionen** Wählt zwei Dimensionen des Datenraums aus, die im Hauptfenster dargestellt werden.

**Aktualisiere Fenster** Bringt die Fenster, die die Karte und die Daten darstellen, auf den neuesten Stand.

**Schätze Skala** Schätzt die Graustufenskalen des Magnification Factors- und Wahrscheinlichkeitsdichte-Fensters ab.

**Daten zeigen** Hier kann man wählen, welche Daten in den Fenstern dargestellt werden sollen: Trainingsdaten, Validierungsdaten, Testdaten, Maldaten oder keine.

**Hotspots farbig** Stellt ein, ob die Daten, die zu einem Hotspot gehören, farbig dargestellt werden.

**nur Hotspots malen** Stellt ein, ob ein Datenpunkt nur dann gezeichnet wird, wenn er einem Hotspot entstammt.

**Maldaten laden** Lädt eine Datendatei, die die Maldaten enthält.

Das Fenster-Menü schließlich enthält je einen Menüpunkt, um das Magnification Factors- beziehungsweise Wahrscheinlichkeitsdichte-Fenster zu öffnen oder zu schließen.

### 6.1.3 Die Benutzung der graphischen Oberfläche

Zu Beginn kann man entweder eine abgespeicherte GTM-Karte laden (Menüpunkt Datei/Laden) oder eine neue Karte initialisieren (Datei/Neue GTM).

Wählt man den letzteren Menüpunkt, so erscheint der in Abbildung 6.1 abgebildete Dialog. Zunächst wählt man die Datei der T-Vektor-Daten, die man lernen möchte. (Erst dann wird der „OK“-Knopf benutzbar.) Dann stellt man die Parameter der GTM-Karte sowie des Lernverfahrens ein. Die Parameter für die semilineare Karte sind natürlich nur dann verfügbar, wenn $Q > 0$ ist.

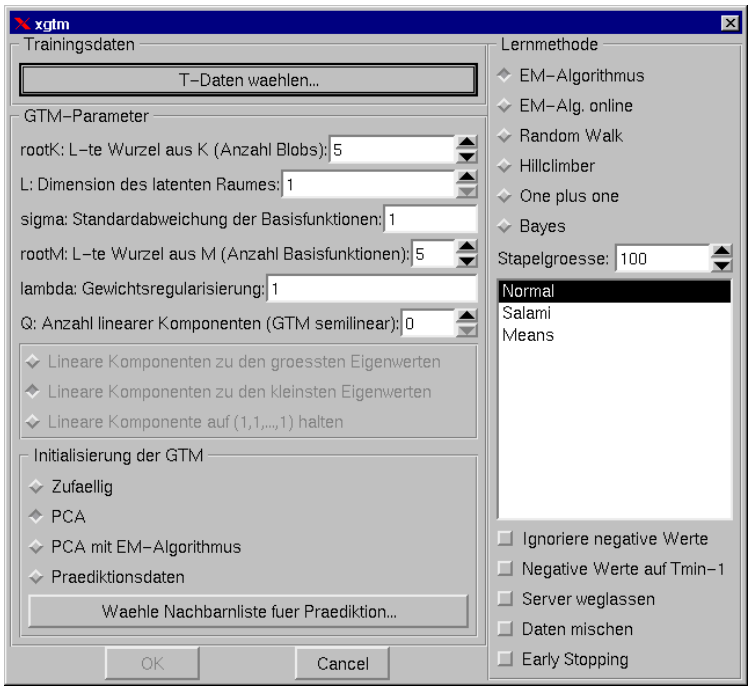



Abbildung 6.1: Dialogfenster für die Parameter einer neuen GTM-Karte

Auf der rechten Seite des Dialogfensters befinden sich die Parameter für das Training und die Behandlung der Datenmenge. Für die Daten kann man eine Metrik wählen (siehe Abschnitt 5.3), man kann die Behandlung fehlender Komponenten einstellen, die Serverkomponente ignorieren und Early Stopping aktivieren.

Ist Early Stopping desaktiviert, dann gibt es keine Validierungs- und Testdaten; alle Daten werden als Trainingsdaten benutzt. Ist es jedoch angeschaltet, dann werden die Daten in diese drei Mengen gedrittelt (siehe Abschnitt 4.5.3).

## 6.2 Die nicht-graphischen Anwendungen

### 6.2.1 Bilder erzeugen mit *gtmtopng*

Mit *gtmtopng* kann man eine GTM-Datei einlesen und eine PNG-Bilddatei erzeugen, die die Magnification Factors oder Wahrscheinlichkeitsdichte darstellt, genau wie die Fenster von *xgtm*. Es gibt die folgenden Parameter:

- `-g (GTM-Datei)`: Die GTM-Karte, die dargestellt werden soll. Ein obligatorisches Argument.

- `-m[Maldatendatei]`: Mit dem Parameter `-m` ohne Argument werden die Trainingsdaten in das Bild eingetragen. Wird jedoch ein Dateiname angegeben, so werden die in dieser Datei enthaltenen Daten dargestellt.
- `-w`: Es wird eine Wahrscheinlichkeitsdichtekarte erzeugt (Default ist Magnification Factors).
- `-s (min:max)`: Direkte Angabe eines Minimum- und Maximumwertes für die Graustufenskala. Default ist Bestimmung dieser Werte durch Schätzung anhand eines Samples.
- `-k (Körnung)`: Auflösung des Bildes. Default ist 10, d. h. das Bild besteht aus Blöcken von 10×10 Pixeln.
- `-o (Ausgabedatei)`: Dateiname für die PNG-Bilddatei. Default ist der Name der GTM-Datei. An diesen Namen wird das Suffix „_skala-min_skala-max.png“ angehängt.

### 6.2.2 Automatisches Training mit *execplaene*

Ein Plan ist eine Datei, die die Werte für die Variablen und Parameter für eine GTM-Karte und ein Training enthält. Mit *execplaene* können einer oder mehrere Pläne eingelesen und ausgeführt werden. Das Kommandozeilenformat ist einfach

`execplaene Plan [Pläne...]`.

Es empfiehlt sich, die Ausführung eines Planes, die je nach Variante des Algorithmus einige Zeit dauern kann, mit Hilfe der Unix-Shell in den Hintergrund zu schicken, bei der tcsh etwa mit:

```
execplaene plan >& plan.out &
```

Das Dateiformat der Pläne wird in Abschnitt A.2.3 spezifiziert.

# 7 Ergebnisse

## 7.1 Synthetische Daten

Bevor das Verfahren auf echte Daten angewandt wird, ist es sinnvoll, es an synthetischen Daten auszuprobieren. Dies dient einerseits dazu, zu prüfen, ob der Algorithmus prinzipiell in der Lage ist, ein solches Problem zu lösen; andererseits gilt es nachzuweisen, daß er fehlerfrei implementiert ist.

Deswegen sollten die synthetischen Daten zwar genügend realistisch sein – damit man die Performance des Verfahrens beurteilen kann –, aber nicht zu sehr: Wenn man alle Schmutzeffekte berücksichtigt, läßt sich nicht beurteilen, ob ein schlechtes Ergebnis an den Daten oder an dem Verfahren liegt.

### 7.1.1 Erzeugung der synthetischen Daten

Von 1997 bis 1999 lief das EU-Projekt *ARNO* (Algorithms for Radio Network Optimization, Esprit-Projekt 23243). Dabei ging es um die Plazierung von Antennen in einem GSM-Netz. Verschiedene Datenbeispiele wurden dafür von France Telekom CNET zur Verfügung gestellt.

Die Software, die für dieses Projekt geschrieben wurde [ZHM00], ist in der Lage, eine Zellstruktur zu erzeugen und an jedem Pixel in einer Zelle die Feldstärke jeder Station zu berechnen. Um synthetische Daten für das vorliegende Problem zu erzeugen, habe ich ein Modul geschrieben, das dem Abgreifen von $A_{bis}$-Daten in einer Zelle entspricht.

Die Größe der Nachbarschaftsliste wurde auf 8 gesetzt. Zunächst wurden 200 Datensätze über die ganze Zelle gleichverteiltes Hintergrundrauschen erzeugt, dann an dreizehn über die Zelle verteilten Punkten je 100 Datensätze. Insgesamt gibt es also 1500 Datensätze, womit das Programm zügig umgehen kann.

Es wurde bewußt eine recht niedrige Menge an Hintergrunddaten gewählt, um die Herausbildung von Hotspots zu erleichtern. Die Hotspots wurden entlang dem Rand der Zelle verteilt, so daß eine ringförmige Struktur zu erwarten ist (s. Abb. 7.1); außerdem sollte auf der Karte die Reihenfolge der Spots erhalten bleiben.

Es wurden jeweils die Feldstärke des Servers und der sechs stärksten Nachbarsender an diesem Punkt in den Vektor geschrieben, versehen mit Gaußschem Rauschen mit der Standardabweichung 3. Dazu kam eine gleichverteilt zufällige Dämpfung $\lambda$ zwischen 0 und 10, die von jeder Komponente abgezogen wurde. Die Werte wurden auf den RxLev-Wertebereich zwischen 0 und 63 eingeschränkt.

Abbildung 7.1: Synthetische Daten: Die Form der Zelle und die Lage der Hotspots

| Nr | Farbe | Position | Feldstärkevektor (RxLev) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | rot | (83,746) | 63 | 28 | 3 | 40 | 7 | 0 | 18 | 0 | 3 |
| 2 | gelb | (93,741) | 48 | 39 | 1 | 26 | 3 | 1 | 13 | 2 | 1 |
| 3 | blau | (100,735) | 42 | 40 | 11 | 21 | 12 | 7 | 13 | 9 | 6 |
| 4 | hellblau | (112,735) | 38 | 33 | 13 | 18 | 14 | 12 | 17 | 14 | 10 |
| 5 | magenta | (121,739) | 33 | 28 | 8 | 13 | 12 | 7 | 16 | 29 | 6 |
| 6 | grau | (132,738) | 29 | 13 | 14 | 9 | 20 | 9 | 14 | 24 | 7 |
| 7 | dunkelgrün | (139,735) | 29 | 10 | 19 | 9 | 22 | 12 | 12 | 23 | 10 |
| 8 | dunkelcyan | (145,730) | 30 | 6 | 24 | 10 | 22 | 13 | 13 | 19 | 12 |
| 9 | dunkelrosa | (140,714) | 32 | 10 | 29 | 11 | 18 | 22 | 15 | 16 | 15 |
| 10 | dunkelgelb | (140,707) | 36 | 10 | 30 | 15 | 14 | 27 | 13 | 14 | 21 |
| 11 | dunkelrot | (129,704) | 32 | 16 | 23 | 11 | 19 | 23 | 11 | 14 | 20 |
| 12 | dunkelblau | (119,709) | 31 | 17 | 20 | 11 | 13 | 19 | 10 | 8 | 16 |
| 13 | hellgrau | (108,714) | 31 | 27 | 17 | 13 | 21 | 16 | 12 | 8 | 15 |

Tabelle 7.1: Synthetische Daten: Positionen und Feldstärkevektoren der Hotspots

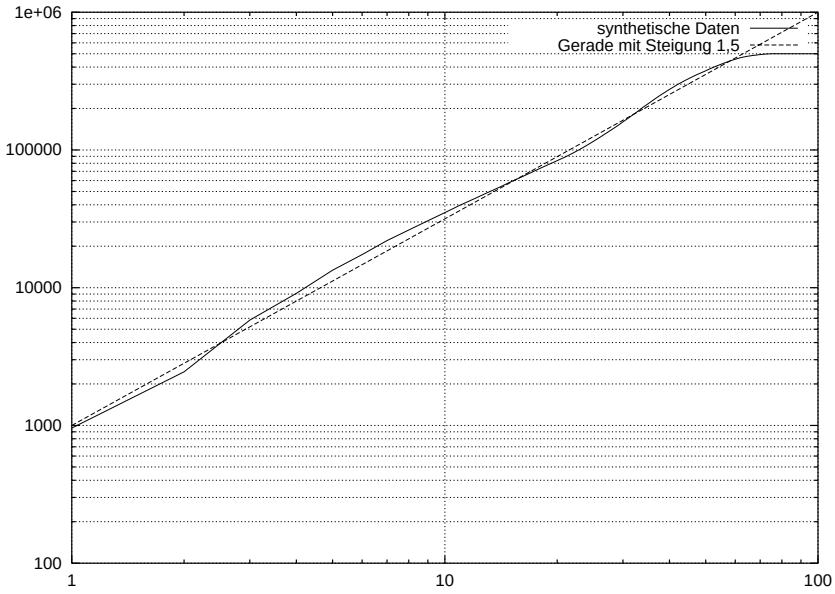



Abbildung 7.2: Dimension der synthetischen Daten

### 7.1.2 Die Dimension der synthetischen Daten

Der Dimensionsalgorithmus aus Abschnitt 4.3.1 liefert das in Abbildung 7.2 dargestellte Ergebnis. Obwohl die Zelle zweidimensional ist und über die hinzugefügte Dämpfung $\lambda$ eine dritte Dimension hinzugekommen sein müßte, beträgt die Steigung des kumulierten Histogramms nur etwa 1,5.

Die Hauptkomponentenanalyse der Daten (siehe Tabelle 7.2) scheint dies zu bestätigen. Schon der zweite Eigenwert ist deutlich kleiner als der erste, zum dritten Wert ist es wieder ein großer Sprung. Außerdem fällt auf, daß erst der dritte Eigenvektor (mit Ausnahme zweier Komponenten) in den ersten Quadranten $(1, 1, \ldots, 1)$ zeigt. Bei den gemessenen, nichtsynthetischen Daten (und auch in [Qua97]) stellte sich stets heraus, daß der erste Eigenvektor in den ersten Quadranten zeigte, der ja der Richtung der Dämpfung entspricht. Es scheint demnach, daß die auf die Werte addierte Dämpfung zwischen 0 und 10 noch hinter der Realität zurückbleibt.

| Eigenvektor | | | | | | | | | Eigenwert |
|---|---|---|---|---|---|---|---|---|---|
| 0,326 | 0,593 | -0.396 | 0,454 | -0,192 | -0,237 | 0,215 | -0,148 | -0,133 | 879,61 |
| -0,226 | -0,013 | -0,252 | -0,105 | 0,405 | -0,508 | 0,063 | 0,539 | -0,397 | 517,38 |
| 0,584 | -0,386 | 0,338 | 0,464 | 0,170 | 0,045 | 0,201 | 0,322 | -0,072 | 358,66 |

Tabelle 7.2: Hauptkomponentenanalyse der synthetischen Daten: Die ersten drei Eigenvektoren und ihre Eigenwerte

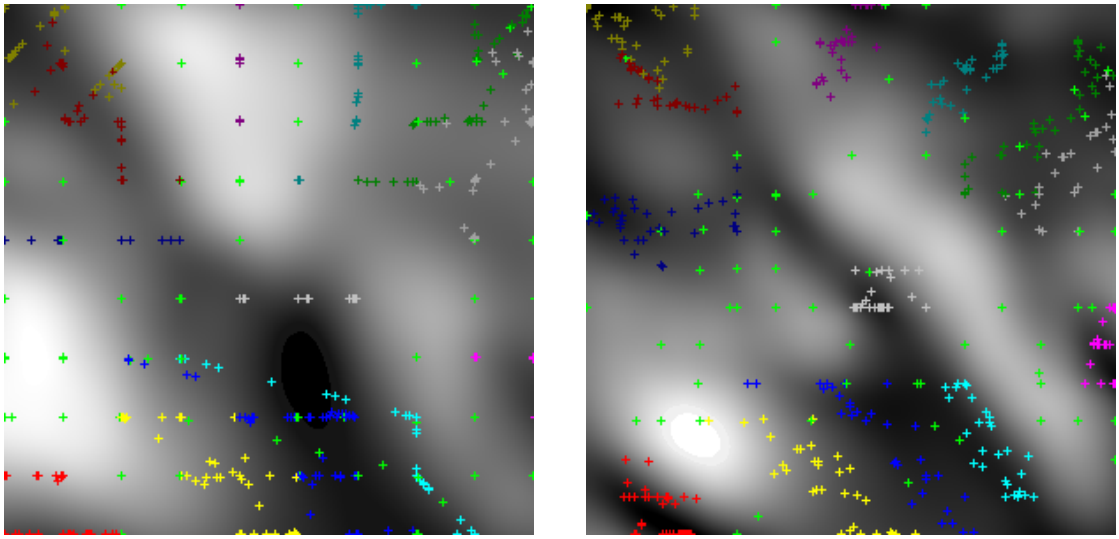


Abbildung 7.3: GTM-Karten für die synthetischen Daten, links ohne, rechts mit Projektion der Daten vermittels Mean-Metrik

### 7.1.3 Anwendung des GTM-Verfahrens auf die synthetischen Daten

Wie bei einem so einfachen Modell nicht anders zu erwarten ist, hat das GTM-Verfahren wenig Probleme, die synthetischen Daten zu lernen. Die Topologie, d. h. die Lage der Spots zueinander sowie die beiden Löcher in der Zellenmitte und oben links, werden fast bei allen Varianten korrekt wiedergegeben.

Meist zeigt sich auch, daß durch die zufällige Dämpfung der Datensätze die Spots in Fäden gezogen werden; auch die semilineare Karte, die doch zur Behandlung der Dämpfung dienen sollte, zeigt dieses Phänomen. Recht gut behandeln kann man es jedoch mit den speziellen Abstandsmaßen, die die $\lambda$-Komponente herausprojizieren, namentlich mit der Means-Metrik (siehe Abschnitt 5.3.2). Wo das Problem der Fadenbildung dennoch auftritt, liegt es offenbar daran, daß einige Komponenten der gedämpften Trainingsvektoren an die Unterkante des Wertebereiches anstoßen.

Abbildung 7.3 zeigt typische GTM-Karten für die synthetischen Daten. Es ist der zweidimensionale latente Raum dargestellt. In Graustufen sieht man die Magnification Factors; schwarz bedeutet einen kleinen, weiß einen großen Magnification Factor. Die Gebiete der Hotspots sind also schwarz gefärbt. Die eingezeichneten Punkte sind die Projektionen der Daten in den latenten Raum nach Formel 4.42. Die Hintergrunddaten, die über die Zelle gleichverteilt sind, sind hellgrün markiert, die Daten aus den Hotspots gemäß dem Schlüssel aus Tabelle 7.1.

Es läßt sich feststellen, daß die Hotspots klar voneinander getrennt werden, vor allem dort, wo die Pegelwerte recht distinguiert sind. Wenn die Werte der Spots hingegen so ähnlich sind, daß die Unterschiede in der Größenordnung des Gaußschen Rauschens liegen (etwa bei Spots 6 und 7), trennen sie sich natürlich nicht. Stets sind auch die Regionen

| | | $A_{bis}$-Daten | | | T-Vektoren | | |
|---|---|---|---|---|---|---|---|
| Spot-Nr. | Farbe | Spot | gesamt | % | Spot | gesamt | % |
| 1 | rot | 7103 | 27247 | 26,1 | 147 | 584 | 25,2 |
| 2 | gelb | 6376 | 35133 | 18,1 | 134 | 733 | 18,3 |
| 3 | blau | 1892 | 15127 | 12,5 | 43 | 325 | 13,2 |
| 4 | hellblau | 3342 | 21906 | 15,3 | 69 | 452 | 15,3 |
| 5 | magenta | 3983 | 34152 | 11,7 | 78 | 705 | 11,1 |
| Summe | | 22696 | 133565 | 17,0 | 471 | 2799 | 16,8 |

Tabelle 7.3: Anzahl der verwertbaren Daten in Bergheim. Angegeben sind für jeden Spot die Anzahl der Zeilen in den $A_{bis}$-Dateien, einerseits der selbsterzeugten Hotspot-Gespräche, andererseits des gesamten in der Zelle gelaufenen Verkehrs. Die Prozentzahl gibt also den Verkehrsanteil an. Der rechte Teil der Tabelle enthält die Zahlen nach der Zusammenfassung der Daten zu T-Vektor-Dateien.

ohne Hotspots als weiße Flächen zu erkennen, die Mitte der Zelle sowie der obere linke Teil (vgl. Abb. 7.1).

## 7.2 Die Meßaktionen

### 7.2.1 Bergheim

Im Rahmen der Vorgängerarbeit [Qua97] wurden in einer Zelle in Bergheim $A_{bis}$-Daten aufgezeichnet. Die Zelle hatte sechs Nachbarsender. Sowohl diese als auch der Server waren omnidirektionale Antennen, d. h. sie strahlten gleichmäßig in alle Richtungen.

Es wurden über die Zelle verteilt fünf Verkehrsschwerpunkte erzeugt, die jeweils etwa ein Sechstel des Verkehrsaufkommens stellten (siehe Tabelle 7.3). Dabei zeichnete sich Spot 3 dadurch aus, daß er sehr nah beim Sender lag. Spot 2 befand sich sehr weit entfernt von der BTS. Die trainierten Daten bestanden aus den Daten dieser fünf selbsterzeugten Spots und den währenddessen in der gesamten Zelle aufgezeichneten Hintergrunddaten.

Damals gelang es, von den fünf Hotspots die Spots 1 und 2 sicher zu erkennen, Spot 3 etwas schwächer. Spot 4 war auf der Karte nicht auffindbar, Spot 5 zerfiel in mehrere Komponenten. ([Qua97], S. 65ff.)

### 7.2.2 Euskirchen

Am 18., 19. und 20. April 2000 wurde in zwei Zellen bei Euskirchen eine umfangreiche Meßaktion durchgeführt. Die beiden Sender befinden sich an demselben Sendemast (nebst einer dritten, nicht betrachteten BTS). Der Sender *Euskirchen 4/1a* hat neun Nachbarn,

**Euskirchen 4/1a**

| Spot | Farbe | T-Vek. |
|---|---|---|
| 1 | rot | 11 |
| 2 | gelb | 33 |
| 3 | blau | 38 |
| 4 | hellblau | 21 |
| 5 | magenta | 48 |
| 6 | grau | 34 |
| -[1] | hellgrün | 1 |
| 7 | dunkelgrün | 38 |
| 8 | dunkelcyan | 36 |
| 9 | dunkelrosa | 26 |
| 11 | dunkelgelb | 44 |
| 12 | dunkelrot | 44 |

**Euskirchen 4/3**

| Spot | Farbe | T-Vek. |
|---|---|---|
| 9 | rot | 5 |
| 10 | gelb | 39 |
| 13 | blau | 57 |
| 14 | hellblau | 45 |
| 15 | magenta | 28 |
| 16 | grau | 31 |
| 17 | dunkelgrün | 41 |
| 18 | dunkelcyan | 31 |
| 19 | dunkelrosa | 25 |
| 20 | dunkelgelb | 36 |
| 21 | dunkelrot | 43 |
| 22 | dunkelblau | 34 |
| 23 | hellgrau | 43 |

Tabelle 7.4: Die Meßpunkte der Meßaktion Euskirchen. Angegeben sind die Farben der Punkte in den GTM-Karten und die Anzahl der T-Vektoren, die nach Aggregation mit Intervallgröße 50 aus den Gesprächen erzeugt wurden. An Spot 9, der in der Nähe der Zellengrenze liegt, gab es Gespräche über beide Server.

*Euskirchen 4/3* zwölf. Beide sind direktionale Antennen; Euskirchen 4/1a strahlt nach Nordosten, Euskirchen 4/3 nach Westen.

Die Straßen und Wege der Zellen wurden flächendeckend mit einem Meßfahrzeug abgefahren, um Blackbox-Daten zu sammeln. Außerdem wurden an verschiedenen Orten innerhalb der Zellen Gespräche geführt, allerdings nicht in dem Umfang wie bei der Bergheim-Messung (siehe Tabelle 7.4). Diese Gespräche wurden nicht im Hinblick auf eine Hotspot-Erkennung geführt, sondern es galt hauptsächlich, sie mit Hilfe der Prädiktionskarten zu lokalisieren [Sch00].

Für das GTM-Verfahren besteht die Aufgabe hauptsächlich darin, die Cluster voneinander zu trennen, und weniger darin, sie als Verkehrsschwerpunkte zu erkennen, da sie nicht als solche erzeugt wurden. Außerdem gilt es, die Topologieerhaltung anhand dieser Spots zu überprüfen. Dazu stehen die gesamten $A_{bis}$-Daten zur Verfügung, die an den drei Tagen in diesen Zellen aufgezeichnet wurden. Nach Zusammenfassung der Gespräche mit Intervallgröße 50 ergab dies in der Zelle Euskirchen 4/1a 21930 T-Vektoren, und der Zelle Euskirchen 4/3 15932 T-Vektoren.

In der Zelle Euskirchen 4/1a befanden sich 11 Spots, in der daneben liegenden Zelle Euskirchen 4/3 13 Spots. Die Lage der Spots ist in Abbildung 7.4 dargestellt.

[1] Call 40; fand in der Nähe von Spot 7 statt

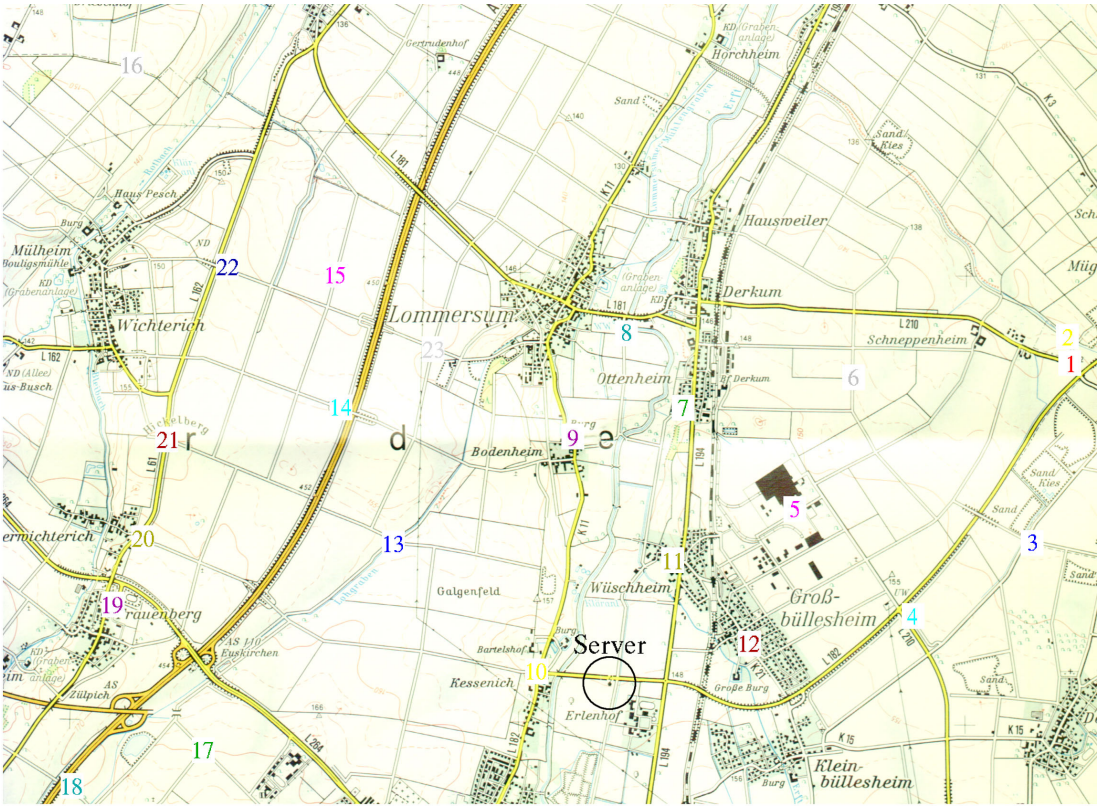



Abbildung 7.4: Karte des Euskirchener Meßgebietes. Ausschnitt entspricht 9,6 km × 7,1 km. Darstellung auf Grundlage der Topographischen Karte 1:50 000 des Landes Nordrhein-Westfalen mit freundlicher Genehmigung des Landesvermessungsamtes NRW vom 12.10.2000, Az.: S 1582/2000

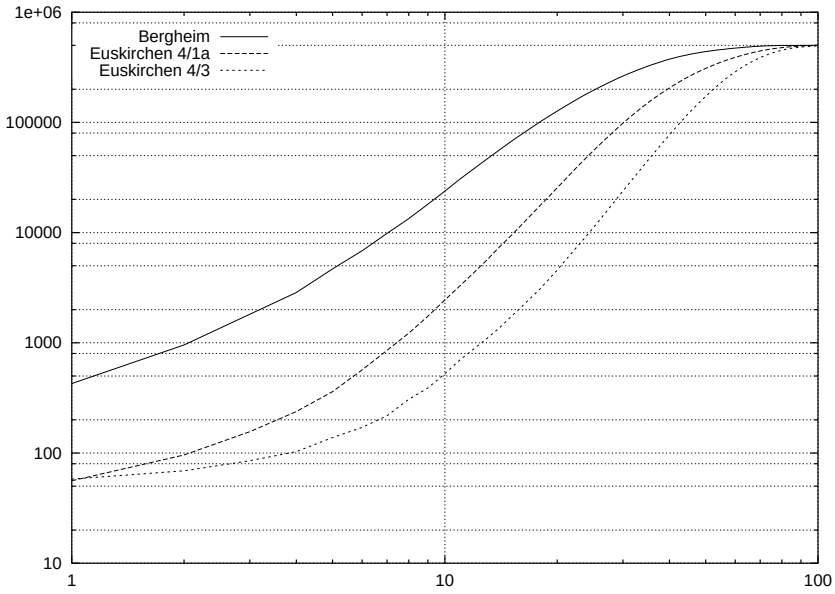



Abbildung 7.5: Intrinsische Dimension der Bergheimer Daten und der beiden Euskirchener Zellen.

## 7.3 Statistische Untersuchungen der Meßdaten

### 7.3.1 Dimension

Das Ergebnis des Dimensionsalgorithmus aus Abschnitt 4.3.1 ist in Abbildung 7.5 dargestellt.

Es zeigt sich, daß die Daten aus Bergheim ungefähr zweidimensional, die aus Euskirchen hingegen etwa dreidimensional sind. Eine mögliche Erklärung dafür ist, daß die Zelle in Bergheim nur sechs Nachbarn hatte, die Euskirchener Zellen hingegen neun (4/1a) bzw. zwölf (4/3) Nachbarn. Da der Pegelraum also eine höhere Dimension hat, ist eine etwas größere intrinsische Dimension der Meßdaten ganz folgerichtig.

Insgesamt bestätigt dies jedoch die Annahme, daß die Daten durch zwei latente Variablen und die Dämpfung beschrieben werden können.

### 7.3.2 Hauptkomponentenanalyse

Die Hauptkomponentenanalyse der Meßdaten ist in den Tabellen 7.5 und 7.6 dargestellt. Allgemein läßt sich feststellen, daß der erste Eigenvektor der Kovarianzmatrix durchweg in den ersten Quadranten, d. h. in Richtung der Dämpfung $\lambda$, zeigt (mit Ausnahme einer Komponente in Zelle Euskirchen 4/3, die sehr selten empfangen wurde).

| | Eigenvektor | | | | | | | Eigenwert |
|---|---|---|---|---|---|---|---|---|
| 1. EV | 0,475 | 0,091 | 0,391 | 0,504 | 0,290 | 0,505 | 0,142 | 752,97 |
| 2. EV | -0,833 | 0,204 | 0,370 | 0,167 | 0,244 | 0,106 | 0,171 | 316,97 |
| 3. EV | 0,237 | 0,354 | 0,551 | -0,243 | -0,272 | -0,437 | 0,437 | 299,63 |
| $\bar{t}$ | 34,46 | 2,26 | 12,08 | 14,54 | 2,75 | 19,43 | 2,55 | |

Tabelle 7.5: Hauptkomponentenanalyse der Bergheimer Daten: Die ersten drei Eigenvektoren und ihre Eigenwerte sowie der Mittelwert der Daten

**Euskirchen 4/1a**

| $\bar{t}$ | 1. EV | 2. EV | 3. EV |
|---|---|---|---|
| 33,06 | 0,409 | -0,324 | 0,087 |
| 9,83 | 0,338 | 0,254 | -0,310 |
| 19,86 | 0,472 | -0,416 | -0,287 |
| 13,10 | 0,365 | 0,419 | 0,554 |
| 10,14 | 0,348 | 0,230 | -0,184 |
| 4,39 | 0,153 | 0,250 | -0,386 |
| 1,36 | 0,071 | 0,115 | -0,035 |
| 16,92 | 0,264 | -0,452 | 0,465 |
| 6,75 | 0,310 | 0,363 | 0,201 |
| 6,44 | 0,219 | -0,135 | -0,259 |
| | 1869,19 | 1492,04 | 982,26 |

**Euskirchen 4/3**

| $\bar{t}$ | 1. EV | 2. EV | 3. EV |
|---|---|---|---|
| 34,91 | -0,460 | 0,084 | -0,081 |
| -0,10 | -0,008 | -0,038 | 0,040 |
| 10,98 | -0,214 | 0,309 | -0,611 |
| 16,02 | -0,422 | -0,530 | 0,026 |
| 17,56 | -0,150 | 0,544 | 0,241 |
| 4,19 | -0,134 | -0,216 | 0,195 |
| 23,46 | -0,479 | -0,044 | 0,315 |
| 1,65 | 0,035 | 0,123 | 0,122 |
| 15,63 | -0,465 | 0,158 | -0,283 |
| 4,22 | -0,002 | 0,309 | 0,099 |
| 4,81 | -0,252 | -0,046 | 0,061 |
| 5,69 | -0,043 | 0,333 | 0,175 |
| 11,49 | -0,106 | 0,150 | 0,532 |
| | 1755,07 | 1243,83 | 1179,94 |

Tabelle 7.6: Hauptkomponentenanalyse der Euskirchener Daten: Mittelwert und die ersten drei Eigenvektoren sowie ihre Eigenwerte

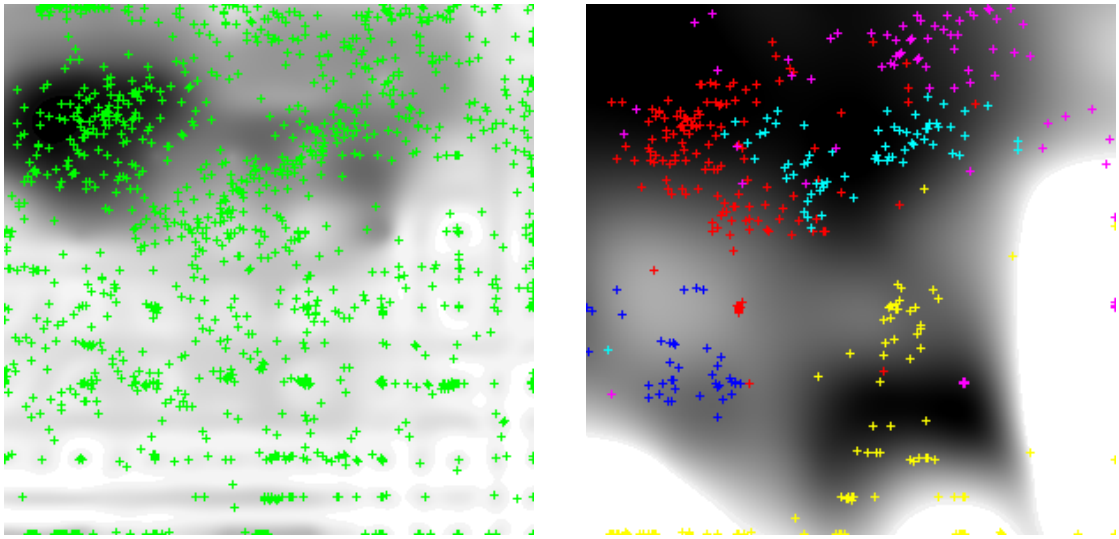


Abbildung 7.6: Einfache GTM-Karte mit Bergheim-Daten. Links eine Wahrscheinlichkeitsdichtekarte mit allen Trainingsdaten. Rechts eine Magnification Factors-Karte, in die nur die fünf Hotspots eingetragen sind.

## 7.4 Ergebnisse des GTM-Algorithmus

### 7.4.1 Der einfache Algorithmus

#### Bergheim

Abbildung 7.6 zeigt das Ergebnis des einfachen GTM-Algorithmus ohne semilineare Erweiterung, angewandt auf die Bergheim-Daten. Spot 2 (gelb), der weitab von den anderen aufgezeichnet wurde, und Spot 3 (blau), der sehr nah am Server lag, sind deutlich von den übrigen getrennt. Auf der Magnification Factors-Karte sind sie als dunkle Flecken sichtbar. Die drei anderen Spots sind anhand der Magnification Factors nicht zu trennen, wohl aber auf der Wahrscheinlichkeitsdichtekarte. Vor allem Spot 1 und Spot 4 sind hier als dunkle Hotspots erkennbar und auch gut getrennt, Spot 5 ist etwas schwächer. Spots 2 und 3 fallen hingegen nicht durch hohe Wahrscheinlichkeitsdichte auf.

Dieses Ergebnis überrascht insofern nicht, als tatsächlich Spot 1 und 4 stärkere Hotspots waren als Spots 3 und 5 (siehe Tabelle 7.3). Nur Spot 2 fällt aus der Reihe. Dieser wird auf der Wahrscheinlichkeitsdichtekarte wohl deswegen nicht erkannt, weil er stärker gestreut wird als die anderen Spots, was vermutlich daran liegt, daß er weit weg vom Sender lag.

#### Euskirchen

Auch bei den Euskirchener Daten werden die serverfernen Spots sehr stark gestreut, wohingegen das Ergebnis in der Nähe des Senders besser ist.

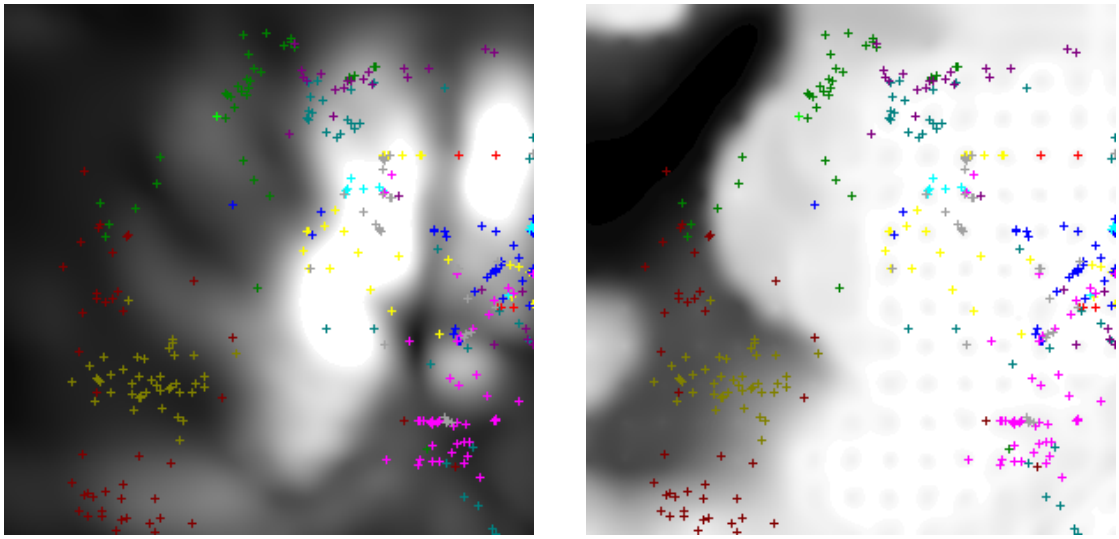


Abbildung 7.7: Einfache GTM-Karte von Euskirchen 4/1a. Links Magnification Factors, rechts Wahrscheinlichkeitsdichte. Jeweils eingetragen die Punkte der 11 Spots.

Auf der Karte der Zelle Euskirchen 4/1a (Abbildung 7.7) werden die Spots 11 (dunkelgelb), 12 (dunkelrot) und 7 (dunkelgrün) gut getrennt. Spot 12 ist in zwei Hälften gespalten, wobei am unteren Bildrand die Gespräche im Fahrzeug, am linken Bildrand jene im Freien dargestellt sind. Etwas verstreuter liegen die Spots 5 (magenta), 8 (dunkelcyan) und 9 (dunkelrosa). Spots 8 und 9 weisen sehr ähnliche Pegelwere auf und werden stets sehr nah beieinander plaziert. Insgesamt bleiben die servernahen Spots kompakter als die entfernteren. Das kann entweder daran liegen, daß dort distinguiertere Pegelwerte herrschen (vor allem für den Server und die beiden Nachbarn, die an demselben Mast hängen), oder daran, daß dort die Meßaktion glatter ablaufen konnte, weil es keine Probleme mit Handover gab. Weit entfernt vom Server gab es viele Gespräche, die wegen eines Handovers zu einer anderen Zelle abgebrochen werden mußten; demzufolge wurde eine größere Anzahl von Gesprächen geführt, was das Rauschen der Daten erhöhte.

Auf der Wahrscheinlichkeitsdichtekarte tun sich hauptsächlich zwei dunkle Flecken am linken Rand hervor, die offenbar den Orten Lommersum/Ottenheim (oben) und Großbüllesheim/Wüschheim (unten, mit Spots 11 und 12) entsprechen. Ein dritter, schwächerer Verkehrsschwerpunkt liegt bei der großen Halle bei Spot 5.

Ein gutes Ergebnis für Zelle Euskirchen 4/3 ist in Abbildung 7.8 dargestellt. Diese Karte wurde mit dem normalen GTM-Algorithmus erzeugt, wobei jedoch die Intervallgröße der Zusammenfassung der Calls nur 20 betrug. Folglich gab es 29615 Datensätze, die zu je einem Drittel als Trainings-, Validierungs- und Testdaten benutzt wurden. Die Topologie ist recht gut erhalten. Fast alle Spots werden auf die richtige Stelle abgebildet, bis auf Spot 19 (dunkelcyan), der über das gesamte Netz verstreut liegt. Allerdings ist bei den meisten anderen Spots die Dispersion sehr groß, auch wenn sie an der richtigen Stelle

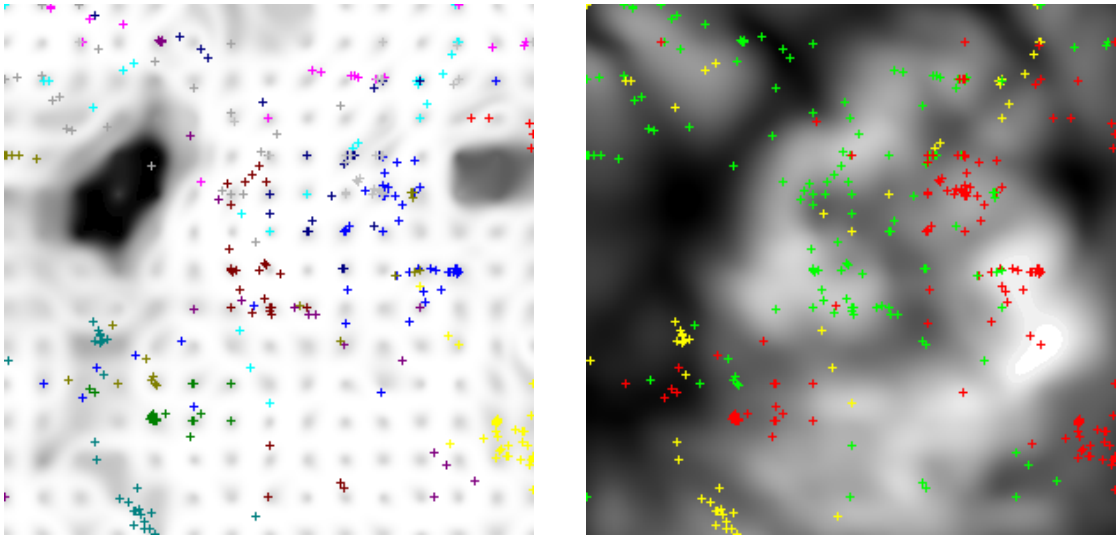


Abbildung 7.8: GTM-Karte Euskirchen 4/3. Aggregationsintervall ist 20. Links Wahrscheinlichkeitsdichte mit Spots, rechts Magnification Factors. Dort Spots östlich der Autobahn rot, an der Autobahn (14 und 18) gelb, westlich grün.

liegen. Sehr kompakt bleiben nur die Spots 9, 10, 17 und 23, alle anderen weisen mehr oder weniger viele Ausreißer auf oder werden in zwei Teile gespalten. Letzteres liegt auch hier daran, daß die Gespräche zum Teil im Meßfahrzeug, zum Teil außerhalb geführt wurden und somit eine verschiedene Dämpfung aufweisen. So besteht bei Spot 18 der Teil am unteren Bildrand aus den Gesprächen im Freien, der obere Teil aus Gesprächen im Fahrzeug. Der Dämpfungsunterschied beträgt etwa 10 dB.

Die Wahrscheinlichkeitsdichtekarte zeigt zwei dunkle Flecken, die den Orten Wichterich (links) und Lommersum (rechts) zu entsprechen scheinen.

Das rechte Bild von Abbildung 7.8 zeigt die Magnification Factors derselben Karte. Dazu wurden wiederum die Daten aus den selbsterzeugten Hotspots eingetragen, wobei die Punkte östlich der Autobahn rot, an der Autobahn (14 und 18) gelb und westlich der Autobahn grün gefärbt sind. Trotz der geringen Datenbasis ist die Vermutung naheliegend, daß der schwarze Streifen in der Mitte der Karte der Autobahn A1 entspricht.

### 7.4.2 Fehlende Daten

Beim Training der obigen Karten wurden fehlende Komponenten stets als $-1$ behandelt. Alternativ dazu ist es auch möglich, fehlende Komponenten als $t_{min} - 1$ zu trainieren oder als unbekannt zu betrachten und herauszuintegrieren (siehe Abschnitt 5.1). Dieser Abschnitt stellt die Ergebnisse dieser Methoden dar.

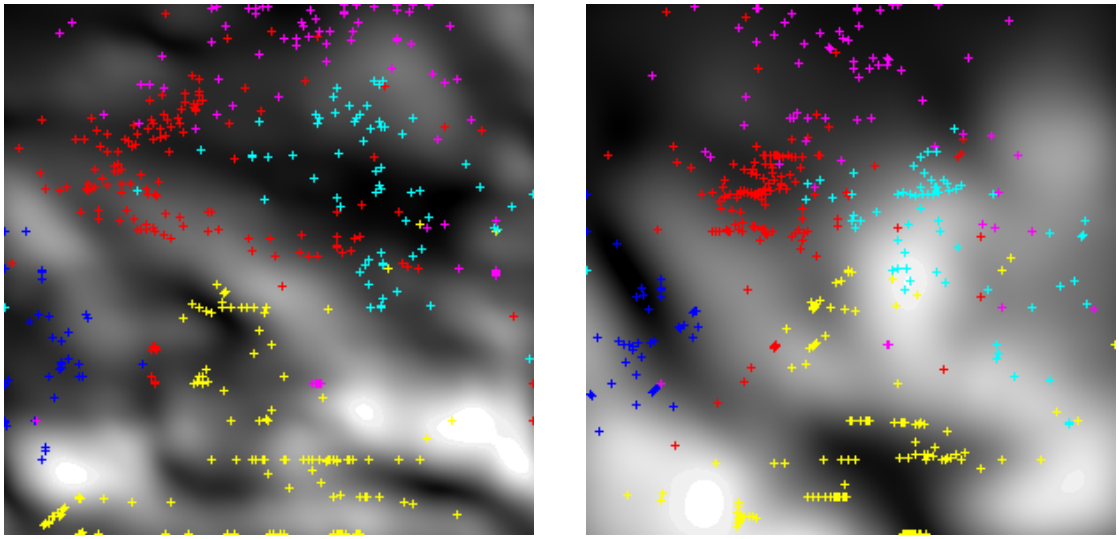


Abbildung 7.9: GTM-Karten der Bergheim-Daten. Fehlende Komponenten wurden von der linken Karte als $t_{min} - 1$ gelernt, rechts herausintegriert.

### Bergheim

Bei den Daten aus Bergheim sind die Ergebnisse mit den obigen vergleichbar (siehe Abbildung 7.9). Die Spots werden recht gut getrennt, sie liegen aber etwas verstreuter.

Es ist erstaunlich, daß die Ergebnisse für Bergheim sehr ähnlich sind. Dies liegt wohl daran, daß die Pegelwerte dort allgemein recht schwach sind, so daß der Unterschied zwischen $t_{min} - 1$ und $-1$ klein ist. Ähnliches gilt, wenn eine Komponente ignoriert wird, die dort, wo sie vorhanden ist, meist sehr kleine Werte aufweist.

### Euskirchen

Bei den Euskirchener Daten ist dies anders. Aufgrund der größeren Nachbarschaftsliste besteht zwischen $-1$ und $t_{min} - 1$ oder Herausintegrieren ein erheblicher Unterschied; denn der kleinste von nur sechs gemeldeten Werten kann noch erheblich größer als $-1$ sein.

Die mit $t_{min} - 1$ oder Herausintegrieren trainierten Karten zeigen jedoch allesamt schlechte Ergebnisse (eines ist in Abbildung 7.14 links dargestellt). Die Spots liegen sehr stark verstreut, eine Trennung ist kaum möglich (außer bei sehr servernahen Spots).

Dies liegt daran, daß der Wert $t_{min} - 1$, der einer fehlenden Komponente zugewiesen wird, sehr stark abhängig ist von den Werten der anderen Komponenten. Dadurch kann sich zwischen Datenvektoren, die an derselben Position aufgezeichnet wurden, ein sehr großer Abstand ergeben, was eine große Dispersion der Daten bewirkt.

Das schlechte Ergebnis des Herausintegrierens belegt, daß die Annahme, man wüßte *nichts* über fehlende Komponenten, die diesem Paradigma zugrundeliegt, hier nicht ver-

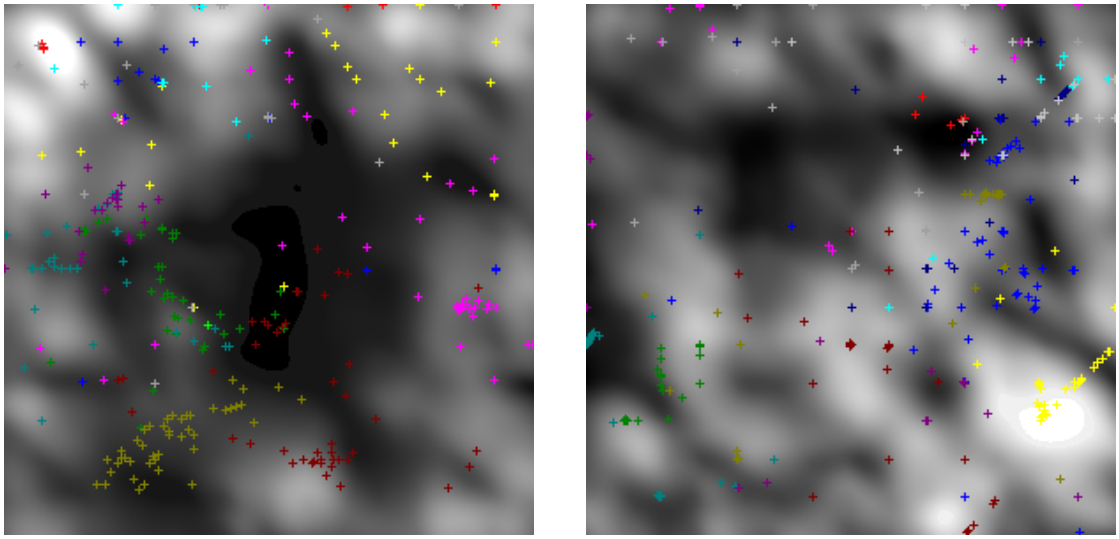


Abbildung 7.10: GTM-Karten von Euskirchen. Die Dämpfung wurde mit der Means-Metrik herausprojiziert. Links Euskirchen 4/1a, rechts Euskirchen 4/3.

tretbar ist. Die Werte der observierten Komponenten haben großen Einfluß auf die Wahrscheinlichkeitsverteilung der fehlenden Komponenten.

### 7.4.3 Ergebnisse mit Behandlung der Dämpfung

Es wurden zwei Möglichkeiten vorgestellt, die Dämpfung $\lambda$ zu behandeln: Entweder man projiziert sie heraus (siehe Abschnitt 5.3), oder man benutzt eine semilineare Karte (siehe Abschnitt 5.4).

#### Projektion der Dämpfung

Die Ergebnisse mit Projektion der Dämpfung vermittels der Means-Methode sind in Abbildung 7.10 dargestellt. Es läßt sich feststellen, daß die Dispersion der Spots weniger stark ist. Beispielsweise ist in Zelle Euskirchen 4/1a der senderferne Spot 2 relativ kompakt geblieben. Auch Spot 14 in Zelle Euskirchen 4/3, der in Abbildung 7.8 sehr stark gestreut ist, zeigt hier nur wenige Ausreißer.

Dafür ist die Topologie bei dieser Methode schlechter erhalten als bei den Karten aus Abb. 7.7 und 7.8. In Zelle Euskirchen 4/1a werden Spots 3 und 4 in die obere linke Ecke, nahe Spot 9 plaziert. Auch in Zelle Euskirchen 4/3 liegen etwa Spots 9, 19 und 20 sehr verkehrt.

Offenbar projiziert die Means-Methode zwar die Dämpfung heraus, so daß die Spots kompakter abgebildet werden, doch dabei wird die Dimension der Daten reduziert, so daß Information verlorengeht. Dies birgt die Gefahr, daß weit voneinander entfernte Spots unverhältnismäßig nahe gerückt werden.

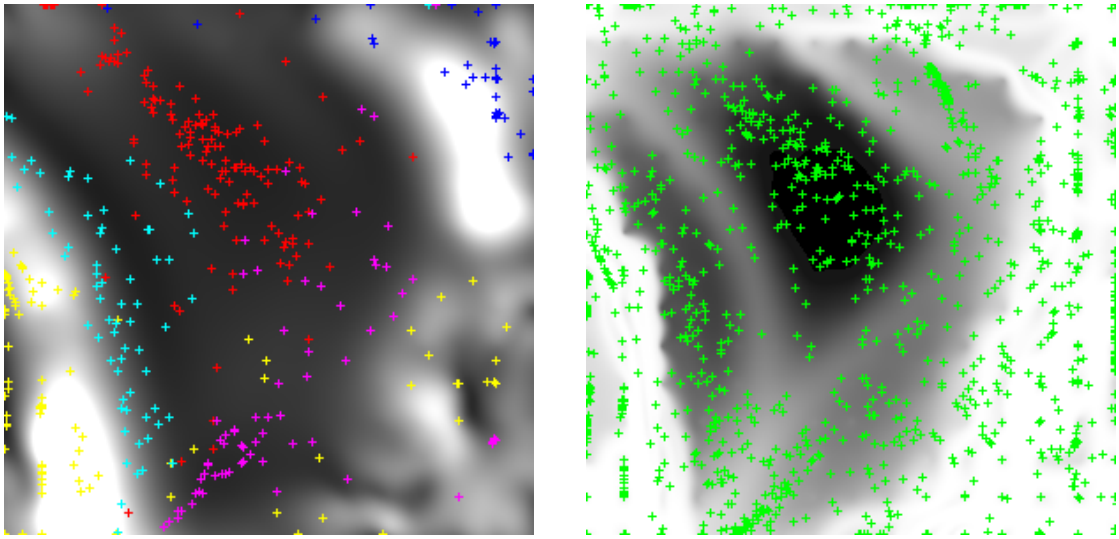


Abbildung 7.11: Bergheim GTM-Karte semilinear; die lineare Achse zeigt in die Richtung $(-4.82, -0.31, -2.62, -3.71, -0.17, -4.82, -0.35)$.

#### Die semilineare Karte

Wie Abbildung 7.11 zeigt, funktioniert die semilineare Karte für die Bergheim-Daten gut. Der lineare Vektor zeigt in den richtigen Quadranten, und die Werte der Komponenten sind vergleichbar mit den mittleren Pegeldaten der Nachbarn (vgl. Tabelle 7.5), was darauf hinweist, daß die lineare Achse tatsächlich in die $\lambda$-Richtung zeigt.

Die Trennung der Spots gelingt ebenso gut wie mit dem gewöhnlichen GTM-Algorithmus (vgl. Abbildung 7.6). So wie dort sind auf der Magnification Factors-Karte Spots 2 und 3 deutlich von den übrigen Spots getrennt (der serverferne Punkt 2 weist wiederum einige Ausreißer auf); Spots 1, 4 und 5 lassen sich auf der Wahrscheinlichkeitsdichtekarte klar trennen.

Für die Euskirchener Daten funktioniert die semilineare Karte weniger gut. Das Ergebnis für Zelle Euskirchen 4/1a ist in Abbildung 7.12 dargestellt. Auch hier richtet sich die lineare Achse selbständig in Richtung der Dämpfung aus, d. h. in den ersten Quadranten mit deutlicher Korrelation zu den Feldstärkewerten der Nachbarn (vgl. Tabelle 7.6). Die Topologie ist zwar in groben Zügen erhalten, doch die Spots sind sehr stark gestreut. Vor allem der sendernahe Spot 12 (dunkelrot) fällt durch ungewöhnlich große Dispersion auf.

### 7.4.4 Ergebnisse der Gütemaße

#### Clustertrennung

Die GTM-Karten mit den besten Clustertrennungen sind in Abbildung 7.13 dargestellt. Es läßt sich feststellen, daß sie in der Tat ihren Zweck erfüllen; die meisten Spots liegen verhältnismäßig kompakt. In der Zelle Euskirchen 4/1a zum Beispiel sind sogar die

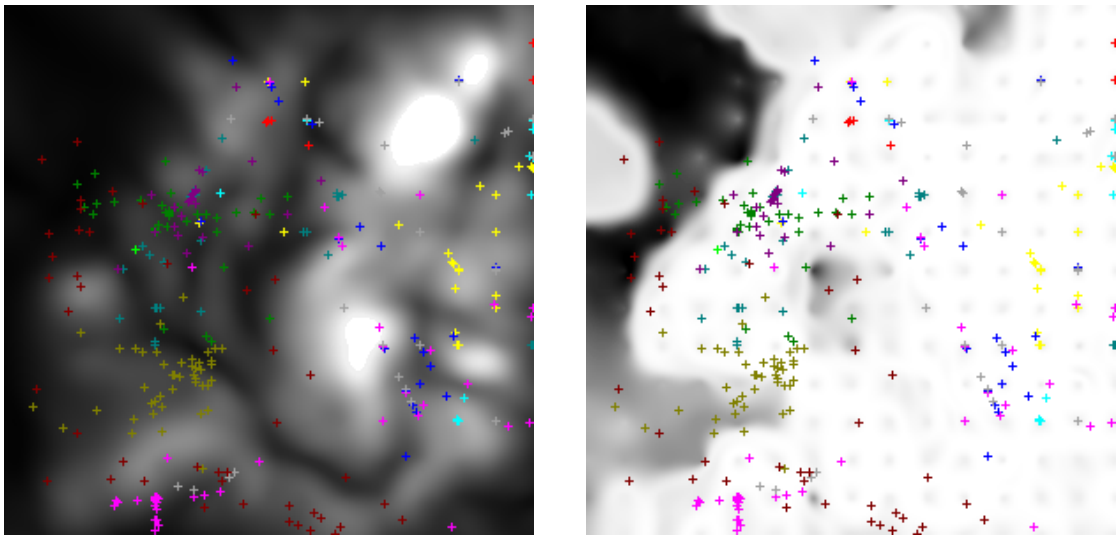


Abbildung 7.12: Semilineare GTM-Karte von Euskirchen 4/1a. Semilineare Achse nach $(-4.78, -1.23, -4.68, -1.12, -2.26, -0.39, -0.02, -4.87, -0.44, -1.31)$.

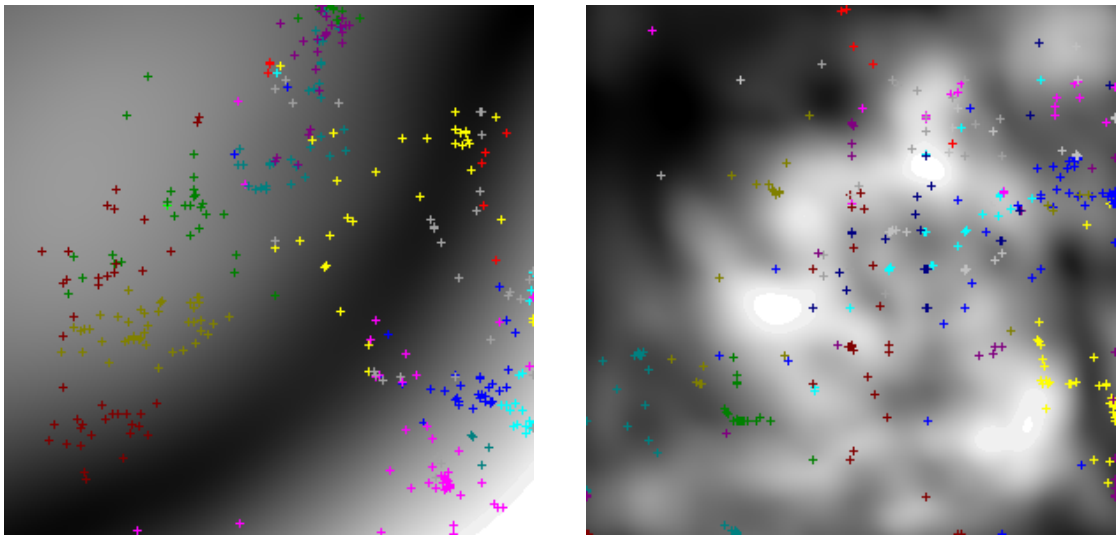


Abbildung 7.13: GTM-Karten mit der besten Clustertrennung. Die linke Karte (Euskirchen 4/1a) behandelt fehlende Komponenten als $t_{min} - 1$. Die rechte Karte (Euskirchen 4/3) benutzt die Means-Metrik und wurde mit PCA initialisiert.

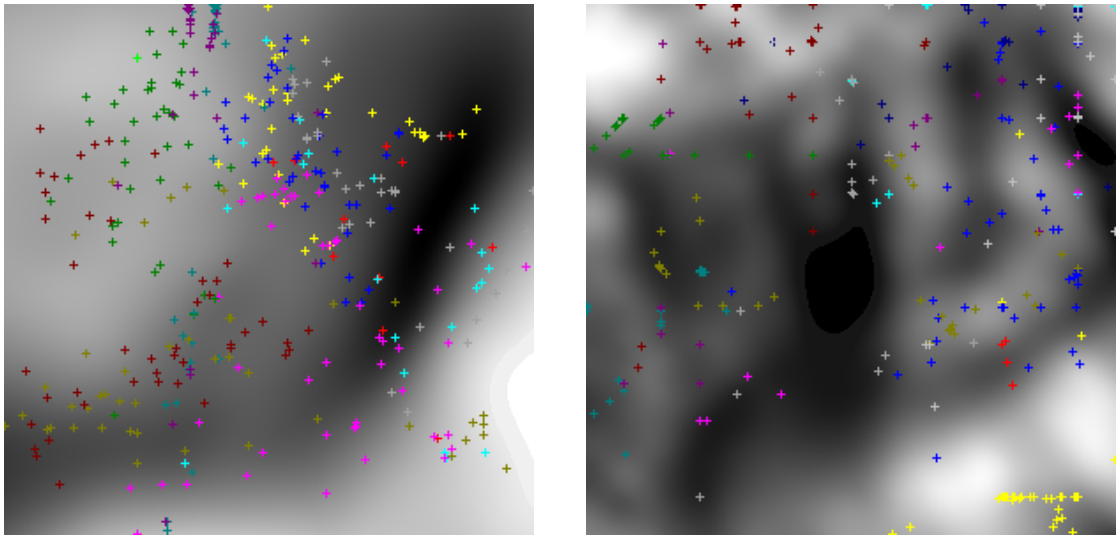


Abbildung 7.14: GTM-Karte mit dem besten topographischen Produkt. Links Euskirchen 4/1a, rechts Euskirchen 4/3.

Spots 3 und 4, die sonst große Probleme bereiten, sehr wenig verstreut. Auch bei dem serverfernen Spot 2 hält sich die Dispersion in Grenzen.

Diese Karte wurde allerdings mit $\sigma = 4$ traininiert, d. h. die Mannigfaltigkeit ist sehr starr. Das ist sehr schön an den Magnification Factors abzulesen, die kaum eine Struktur enthalten. Dieses Bild ist ein Paradebeispiel für Underfitting.

Ähnlich gut ist die Clustertrennung auf der Karte für Euskirchen 4/3 (rechtes Bild). Die Spots liegen verhältnismäßig kompakt, mit $\sigma = 2$ ist auch kein Underfitting festzustellen; aber die Topologieerhaltung ist schlechter als etwa bei der Karte aus Abbildung 7.8. Zum Beispiel liegt Spot 21 (dunkelrot) am unteren Rand, zwischen Spots 10 und 17, sehr verkehrt, ebenso wie Spot 9 am oberen Bildrand.

### Topographisches Produkt

Bei allen trainierten Karten war das topographische Produkt negativ, was auf eine zu niedrige latente Dimension hinweist [BP92]. Die Karten mit dem jeweils betragsmäßig kleinsten topographischen Produkt (siehe Abschnitt 5.6) sind in Abbildung 7.14 dargestellt.

Für die Zelle Euskirchen 4/1a ist die Karte mit dem besten topographischen Produkt eine, die fehlende Komponenten als $t_{min} - 1$ behandelt; dementsprechend sind die Spots über das Netz verstreut. Die gewählte Karte für Zelle Euskirchen 4/3 benutzt die Means-Metrik und ist mit Hauptkomponentenanalyse initialisiert. Auch hier ist jedoch die Topologie stark verfälscht, vor allem bei den serverfernen Punkten westlich der Autobahn.

## 7.5 Evaluierung und Diskussion der Ergebnisse

### 7.5.1 Diskussion der GTM-Varianten

Die stabilsten Ergebnisse für alle Datenmengen wurden erzeugt mit dem einfachen GTM-Algorithmus. Mit diesem war es möglich, die fünf Spots der Daten aus Bergheim zu trennen, aber auch bei den Euskirchener Daten Karten mit plausibler Topologie zu erzeugen. Die Ortschaften im Meßgebiet wurden stets als deutliche Hotspots erkannt. Sogar die Autobahn deutete sich auf einer GTM-Karte an.

Für die Standardabweichung der Gaußschen Basisfunktionen $\sigma$ (siehe Formel 4.7) ergab sich als Optimum ein Wert um 2, d. h. das Doppelte des Abstandes zweier benachbarter Basisfunktionen. Ein Wert unter etwa 1,5 bewirkte allzu flexible Karten mit sehr stark schwankenden Magnification Factors, die sehr zu Topologieverletzungen neigten. Für $\sigma >$ 2,5 hingegen wird die Struktur der Daten nur wenig von der Karte erfaßt.

Für die Gewichtsregulierungskonstante $\lambda$ hat sich ebenfalls ein Wert um 2 bewährt. Die Wahl von $\lambda$ erübrigt sich jedoch, wenn Bayessches Lernen verwendet wird (siehe Abschnitt 4.2.5).

Wie bereits in Abschnitt 7.4.2 ausgeführt wurde, empfiehlt sich zur Behandlung fehlender Komponenten das Training als $-1$. Die alternativen Methoden, $t_{min} - 1$ einzusetzen oder sie zu ignorieren, liefern für die Bergheimer Daten zwar nur wenig schlechtere Ergebnisse, doch für die Euskirchener Testdaten sind die Resultate unbrauchbar.

Bei der Beurteilung der Projektion der Dämpfung vermittels der Means-Methode hält sich die größere Konzentration der Cluster durch Verminderung des Dämpfungseffektes mit dem Informationsverlust durch die Projektion die Waage. Da insgesamt ohne diese Projektion die Topologie besser erhalten blieb, erscheint es empfehlenswert, darauf zu verzichten.

Das semilineare Modell war stets in der Lage, die Richtung der Dämpfung zu erkennen und die lineare Achse darauf auszurichten. Bei den Bergheimer Daten konnte eine Trennung der Hotspots erreicht werden, die sogar noch etwas deutlicher war als bei dem einfachen GTM-Algorithmus. Für die Euskirchener Daten hingegen muß eine stärkere Streuung der Spots konstatiert werden, die wohl damit zusammenhängt, daß die Richtung der linearen Achse stark mit dem Mittelwert der Pegeldaten korreliert ist. Die schwachen Pegel stoßen schnell auf die Unterkante des Wertebereiches. Dadurch ergibt sich eine Verfälschung des Ergebnisses. Bislang leiden alle Versuche, die Dämpfung zu behandeln, unter diesem Problem der fehlenden Komponenten.

Aus diesen Gründen liefert der GTM-Algorithmus ohne Behandlung der Dämpfung die stabilsten Ergebnisse.

### 7.5.2 Diskussion der Gütemaße

Abbildung 7.15 zeigt die Werte der Gütemaße für alle trainierten Karten der Zelle Euskirchen 4/3. Es läßt sich zwischen Clustertrennung und topographischem Produkt keine Korrelation feststellen.

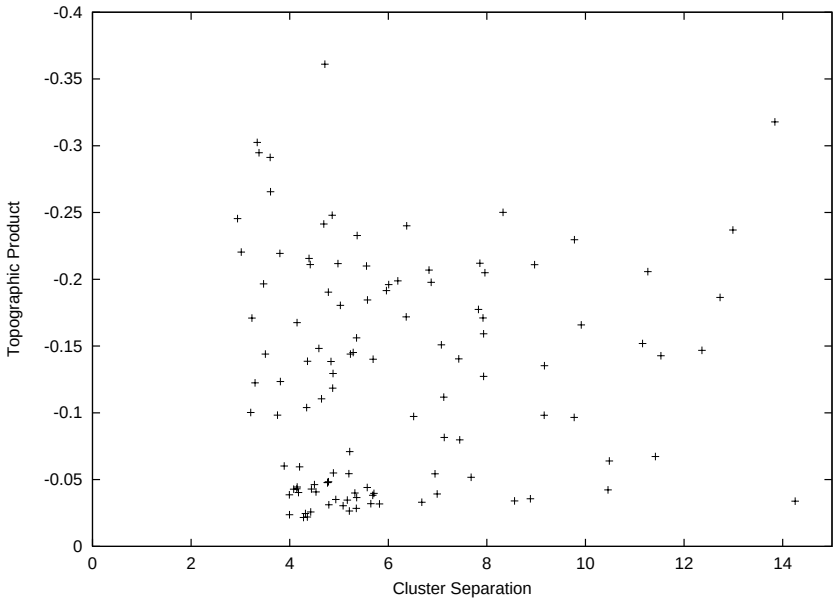



Abbildung 7.15: Werte der Gütemaße für alle trainierten Netze der Zelle Euskirchen 4/3.

Das schlechte Ergebnis des topographischen Produkts liegt vermutlich daran, daß es von Prädiktionskarten abhängig ist. Zwar sind die gemessenen Scankarten deutlich genauer als die berechneten Karten, doch nichtsdestoweniger unterscheiden auch sie sich oft deutlich von den Daten eines Handys, da sie über eine Dachantenne aufgezeichnet wurden.

Um die Scandaten erfolgreich mit den Pegeldaten gewöhnlicher Telefonate zu vergleichen, ist ein aufwendiges Fehlermodell notwendig [Sch00], das bei der Berechnung des topographischen Produkts nicht berücksichtigt werden konnte. Desweiteren wurde auch hierbei die Bounding Box der Zelle verwendet und die Lücken der Scankarten mit dem Mittelwert der Umgebung geschlossen (siehe Abschnitt 4.4.3), was einen zusätzlichen Fehler bewirkt.

Das Maß der Clustertrennung liefert überzeugendere Ergebnisse. Die Trennung der Cluster ist allerdings nicht das alleinige Ziel des Trainings. Für ein brauchbares Ergebnis sollte auch die Topologie ausreichend plausibel erhalten sein. Im übrigen ist das Clustertrennungsmaß von der Verfügbarkeit klassifizierter Daten, d. h. eigens aufgezeichneter Spots, abhängig.

# 8 Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde der GTM-Algorithmus in C++ implementiert. Es zeigte sich, daß er prinzipiell geeignet ist, die Verkehrsschwerpunkte einer Mobilfunkzelle zu erkennen.

Der GTM-Algorithmus ist aufgrund seiner wahrscheinlichkeitstheoretischen Fundierung vielseitig erweiterbar. In der Arbeit wurden Varianten entwickelt, um die speziellen Probleme der gegebenen $A_{bis}$-Daten zu adressieren: unvollständige Daten und die Verrauschung durch Dämpfungseffekte.

- Das *missing data*-Modell, welches in [Sve98] nur angedeutet wurde, wurde formalisiert und implementiert. Leider brachte es keine Verbesserungen. Dies überrascht an sich nicht, da bei der gegebenen Problemstellung die Prämisse „missing at random“ wahrscheinlich nicht ausreichend erfüllt ist.

  Die bislang besten Ergebnisse wurden erzielt, indem für fehlende Komponenten $-1$ eingesetzt wurde. Hier könnte noch ein realistischeres Modell entwickelt werden.

- Die Dämpfungseffekte können vermittels spezieller Metriken herausprojiziert oder durch Verwendung einer semilinearen Karte behandelt werden.

  Beide Methoden sind geeignet, das Rauschen aufgrund von Dämpfungseffekten zu reduzieren, wirken sich jedoch nachteilig auf die Topologieerhaltung der Karte aus. Da diese von eminenter Bedeutung ist, sind beide Verfahren zur Behandlung der Dämpfung wenig empfehlenswert.

Das GTM-Verfahren wurde angewendet auf die Daten, die im Rahmen der Arbeit [Qua97] in Bergheim aufgezeichnet wurden. Das Ergebnis konnte deutlich verbessert werden; im Gegensatz zu damals wurden nun alle fünf Verkehrsschwerpunkte erkannt.

Die Daten, die im Jahre 2000 in Euskirchen aufgezeichnet wurden, enthalten hingegen keine eigens erzeugten Verkehrsschwerpunkte. Die Topologieerhaltung (im Rahmen der Meßdatengenauigkeit) konnte jedoch anhand der selbstgeführten Gespräche nachgewiesen werden. Auch wurden die Ortschaften des Meßgebietes als Verkehrsschwerpunkte erkannt. Sogar eine Autobahn zeichnete sich auf einer Karte ab; da jedoch von der Autobahn nicht genügend Gesprächsmeßdaten vorliegen, läßt sich diese Interpretation nicht zweifelsfrei belegen.

Um die Qualität der erzeugten Karten beurteilen zu können, wurden zwei quantitative Maße herangezogen: ein Maß für die Clustertrennung und ein Maß für die Topologieerhaltung (das topographische Produkt).

- Das Clustertrennungsmaß aus [DB79] wurde implementiert und auf die Daten aus den gegebenen Spots angewendet. Es ist für seinen Zweck geeignet; möglicherweise könnte man es noch verbessern, indem man die Magnification Factors in die Metrik einbezieht oder ein spezielles Abstandsmaß auf der zweidimensionalen GTM-Karte im Pegelraum entwickelt.

- Das topographische Produkt wurde vermittels der Prädiktionskarten auf das Problem angepaßt und implementiert. Visuelle Inspektion der anhand dieses Maßes ermittelten Karten zeigte jedoch, daß es in der vorliegenden Form nicht in der Lage ist, die Topologieerhaltung korrekt zu quantifizieren.

  Andere Topologieerhaltungsmaße, die erwogen wurden, sind nicht verwendbar, da sie von dem gewählten Datenmodell abhängen. Dadurch sind die Werte verschiedener Karten nicht vergleichbar.

Weitere Forschung über geeignete Topologieerhaltungsmaße wäre sehr lohnenswert, denn wenn das Verfahren auf unbekannte Daten angewandt werden soll, ist ein Maß, das eine gute Karte von einer schlechten zuverlässig unterscheiden kann, unbedingt von Nutzen.

# Literaturverzeichnis


[Anl98] Anlauf, Joachim K.: *Abschlußbericht zum Projekt Ortsbezogene Analyse von $A_{bis}$-Meßdaten.* Technischer Bericht, Institut für Informatik, Rheinische Friedrich-Wilhelms-Universität, Bonn, November 1998.

[BGPW96] Bauer, Hans-Ulrich, Theo Geisel, Klaus Pawelzik und Fred Wolf: *Selbstorganisierende neuronale Karten.* Spektrum der Wissenschaft, Seiten 38–47, April 1996.

[Bil98] Bilmes, Jeff A.: *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models.* Technischer Bericht TR-97-021, International Computer Science Institute, Berkeley, April 1998.

[Bis95] Bishop, Christopher M.: *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[BP92] Bauer, Hans-Ulrich und Klaus Pawelzik: *Quantifying the Neighborhood Preservation of Self-Organizing Feature Maps.* IEEE Transactions on Neural Networks, 3(4):570–579, Juli 1992.

[BS95] Barabasi, A. L. und H. E. Stanley: *Fractal Concepts in Surface Growth.* Cambridge University Press, 1995.

[BSW96] Bishop, Christopher M., Markus Svensén und Christopher K.I. Williams: *GTM: a Principled Alternative to the Self-Organizing Map.* In: Malsburg, C. von der, W. von Seelen, J. C. Vorbrüggen und B. Sendhoff (Herausgeber): *Proceedings of ICANN96, International Conference on Artificial Neural Networks*, Band 1112 der Reihe *Lecture Notes in Computer Science*, Seiten 165–170, Berlin, 1996. Springer-Verlag.

[BSW98] Bishop, Christopher M., Markus Svensén und Christopher K.I. Williams: *Developments of the generative topographic mapping.* Neurocomputing, 21:203–224, 1998.

[Dal99] Dalheimer, Matthias Kalle: *Programming with Qt.* O'Reilly Verlag, Köln, 1999.

[DB79] Davies, David L. und Donald W. Bouldin: *A Cluster Separation Measure.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Pam I-1(2):224–227, April 1979.

[DLR77] Dempster, A. P., N. M. Laird und D. B. Rubin: *Maximum likelihood from incomplete data via the EM algorithm.* Journal of the Royal Statistical Society, Series B, 39(1):1–38, 1977.

[Ehr86] Ehrenberg, Andrew S. C.: *Statistik oder der Umgang mit Daten.* VCH Verlagsgesellschaft, Weinheim, erste Auflage, 1986.

[GBD92] Geman, Stuart, Elie Bienenstock und René Doursat: *Neural Networks and the Bias/Variance Dilemma.* Neural Computation, 4(1):1–58, 1992.

[GJ94] Ghahramani, Zoubin und Michael I. Jordan: *Learning from incomplete data.* Technischer Bericht AIM 1509, Massachusetts Institute of Technology, Cambridge, Dezember 1994. ftp://publications.ai.mit.edu.

[Hei00] Heinen, Frank: *Erweiterung und Optimierung des Ortszuordnungsverfahrens in GSM-Netzen mit Hilfe von $A_{bis}$-Daten.* Diplomarbeit, Institut für Informatik, Rheinische Friedrich-Wilhelms-Universität, Bonn, Juli 2000.

[Jol86] Jolliffe, I. T.: *Principal Component Analysis.* Springer-Verlag, Berlin, 1986.

[Kiv96] Kiviluoto, Kimmo: *Topology preservation in self-organizing maps.* In: *IEEE International Conference on Neural Networks (ICNN'96), vol. 1*, Seiten 294–299, Piscataway, New Jersey, Juni 1996.

[KL96] Kaski, Samuel und Krista Lagus: *Comparing Self-Organizing Maps.* In: Malsburg, C. von der, W. von Seelen, J. C. Vorbrüggen und B. Sendhoff (Herausgeber): *Proceedings of ICANN96, International Conference on Artificial Neural Networks*, Band 1112 der Reihe *Lecture Notes in Computer Science*, Seiten 809–814, Berlin, 1996. Springer-Verlag.

[KO98] Kiviluoto, Kimmo und Erkki Oja: *S-Map: A Network with a Simple Self-Organization Algorithm for Generative Topographic Mappings.* In: Jordan, Michael I., Michael J. Kearns und Sara A. Solla (Herausgeber): *Advances in Neural Information Processing Systems*, Band 10. The MIT Press, 1998.

[Koh82] Kohonen, Teuvo: *Self-organized formation of topologically correct feature maps.* Biological Cybernetics, 43:59–69, 1982.

[Koh95] Kohonen, Teuvo: *Self-Organizing Maps.* Springer-Verlag, Berlin, 1995.

[Ler01] Lersch, Martin: *Unterstützung der Ortszuordnung in GSM-Netzen durch Meßdaten.* Diplomarbeit, Institut für Informatik, Rheinische Friedrich-Wilhelms-Universität, Bonn, 2000/01.

[Mah36] Mahalanobis, P. C.: *On the Generalized Distance in Statistics.* Proc. Nat. Inst. Sci. Calcutta, 12:49–55, 1936.

[Mes66] Meschkowski, Herbert: *Mathematisches Begriffswörterbuch.* Bibliographisches Institut, Mannheim, zweite Auflage, 1966.

[MP92] Mouly, Michel und Marie-Bernadette Pautet: *The GSM System for Mobile Communications.* Cell & Sys, Palaiseau, 1992.

[NH98] Neal, R. M. und G. E. Hinton: *A view of the EM algorithm that justifies incremental, sparse, and other variants.* In: Jordan, M. I. (Herausgeber): *Learning in Graphical Models*, Seiten 355–368. Kluwer Academic Publishers, Dordrecht, 1998.

[PTVF92] Press, William H., Saul A. Teukolsky, William T. Vetterling und Brian P. Flannery: *Numerical Recipes in C.* Cambridge University Press, zweite Auflage, 1992.

[Qua97] Quanz, Holger: *Erkennung von Verkehrsschwerpunkten in Mobilfunknetzen auf Grundlage von $A_{bis}$-Meßdaten.* Diplomarbeit, Institut für Informatik, Rheinische Friedrich-Wilhelms-Universität, Bonn, Dezember 1997.

[Row97] Roweis, Sam: *EM Algorithms for PCA and SPCA.* In: *Neural Information Processing Systems 10*, Seiten 626–632, 1997.

[Sch00] Schmitt, Jörg: *Ein Fehlermodell zur Ortszuordnung von Handys in GSM-Netzen mit Hilfe von $A_{bis}$-Daten.* Diplomarbeit, Institut für Informatik, Rheinische Friedrich-Wilhelms-Universität, Bonn, September 2000.

[Str94] Stroustrup, Bjarne: *Design und Entwicklung von C++.* Addison-Wesley, Bonn, 1994.

[Sve98] Svensén, Johan Fredrik Markus: *GTM: The Generative Topographic Mapping.* Doktorarbeit, Aston University, Birmingham, April 1998.

[TB99] Tipping, Michael E. und Christopher M. Bishop: *Probabilistic Principal Component Analysis.* Journal of the Royal Statistical Society, Series B, 61(3):611–622, 1999.

[Tom99] Tomasi, Carlo: *Mathematical Methods for Robotics and Vision.* Technischer Bericht CS 205, Stanford University, 1999.

[VDM94] Villmann, Th., R. Der und Th. Martinetz: *A New Quantitative Measure of Topology Preservation In Kohonen's Feature Maps.* In: *IEEE International Conference on Neural Networks (ICNN'94)*, Seiten 645–648, Piscataway, New Jersey, Juni 1994.

[Whi91] WHITE, G. D.: *The Audio Dictionary.* University of Washington Press, zweite Auflage, 1991.

[Zei96] ZEIDLER, E. (Herausgeber): *Teubner - Taschenbuch der Mathematik.* B. G. Teubner, Stuttgart, Leipzig, 1996.

[ZHM00] ZIMMERMANN, JÖRG, ROBIN HÖNS und HEINZ MÜHLENBEIN: *The Antenna Placement Problem for Mobile Radio Networks: An Evolutionary Approach.* In: *8th International Conference on Telecommunication Systems - Modeling and Analysis*, Seiten 358–366, Nashville, März 2000.

# A Software-Referenz

## A.1 Die Klassenstruktur

### A.1.1 Datenklassen

Die Basisklassen für Daten heißen *Daten* und *Datensatz*. Der Algorithmus benötigt die Möglichkeit, auf die Daten in Form der Matrix $T$ zuzugreifen (siehe z. B. Gleichung 4.12). Deshalb sind die Daten in Matrizen abgespeichert, die Mitglieder der Klasse *Daten* sind. Die Klasse *Datensatz* enthält einen Verweis auf die Matrixzeile, in der er abgespeichert ist.

Für die Matrizen benutze ich die Matrix-Klassenbibliothek *Newmat09*[1]. Sie ist einfach in der Handhabung, ausreichend effizient und enthält alle nötigen Algorithmen, z. B. zur Eigenwertberechnung, zur QR-Zerlegung und zur Singulärwertzerlegung.

#### T-Vektor-Daten

Jeweils von diesen abgeleitet sind die Klassen *TVektorDatensatz* und *TVektorDaten*. Diese sind speziell auf unsere Daten zugeschnitten; sie enthalten einen Parser für T-Vektor-Dateien, die speziellen Metriken für die Behandlung der Dämpfung und die Möglichkeit, fehlende Komponenten auf $t_{min} - 1$ zu setzen sowie die Serverkomponente wegzulassen.

#### Matrixdaten

Die Klassen *MatrixDaten* und *MatrixDatensatz* sind ebenfalls von *Daten* bzw. *Datensatz* abgeleitet. Diese bieten die Möglichkeit, Datenmengen direkt als Matrizen in den Plan- oder GTM-Dateien zu definieren. Dies ist bequem, um den Algorithmus an einfachen Aufgaben auszuprobieren und zu prüfen, ob er korrekt implementiert ist. Inzwischen hat dies jedoch stark an Bedeutung verloren.

### A.1.2 GTM-Klassen

#### GTM

Die *GTM*-Klasse enthält alle Parameter einer GTM-Karte: $K$, $L$, $D$, $M$, *gridXdim* ($= \sqrt[L]{K}$) und *gridFIdim* ($= \sqrt[L]{M}$) – also jeweils die Kantenlängen der Einheitswürfel

[1] *Newmat09* ist unter http://webnz.com/robert frei erhältlich.

der Gaußblobs und der Basisfunktionen –, $\beta$, $\sigma$, $\lambda$, die Log-Likelihoods der Trainings-, Validierungs- und Testdaten (die in der Software Urteildaten heißen, um einen eindeutigen Anfangsbuchstaben zu haben), einen Zeiger auf die zu trainierenden *Daten* sowie die Matrizen *Weights*, $X$, *My* (die Zentren der Gaußschen Basisfunktionen $\mu_m$ im latenten Raum; siehe Formel 4.7), $\Phi$ (die Aktivierungen der Basisfunktionen, $Y$, $\Delta$ (die Abstände der Daten von den Gaußblobs) und $R$ (die Responsibilities).

Die GTM-Klasse enthält darüber hinaus Methoden zum Laden und Speichern von GTM-Dateien, die Berechnung der Magnification Factors, die Abbildung vom latenten Raum in den Datenraum und umgekehrt (siehe dazu Abschnitt 4.6.1), die Initialisierung mit PCA oder Prädiktionskarten sowie das Training mit EM-Algorithmus oder Bayesschem Lernen.

#### Semilineare GTM

Von *GTM* abgeleitet ist die Klasse *GTMsemilin*, die das semilineare Modell (siehe Abschnitt 5.4) bereitstellt. Diese Klasse enthält zusätzlich die Anzahl der linearen Achsen $Q$ sowie die Matrix der linearen Vektoren $V$ und die daraus resultierende Kovarianzmatrix $C$ (siehe Gleichung 5.19). Initialisierungs- und Trainingsfunktionen sind überladen und an das semilineare Modell angepaßt worden.

Da sowohl *GTM* als auch *GTMsemilin* keine mit *new* allozierten Elemente enthalten, kann man sie einfach mit dem Standard-Zuweisungsoperator kopieren. Leider ergibt sich dabei das Problem, daß, wenn man einem GTM-Objekt eine Instanz von *GTMsemilin* zuweist, die semilineare Struktur verloren geht und ein Objekt der Basisklasse *GTM* entsteht. Um dies zu verhindern, habe ich die Funktion *new_clone()* geschrieben, die eine Kopie der Instanz zurückliefert.[2]

### A.1.3 Training-Klassen

#### GTM-Trainer

Der Lernprozeß der GTM-Karte wird von der Klasse *GTM_Trainer* überwacht. Hier kann man verschiedene Lernmodi wählen (von denen jedoch nur EM-Algorithmus und Bayes sinnvoll sind)[3], eine Abbruchbedingung festlegen sowie die aktuelle Karte oder die beste, d. h. die mit der größten Log-Likelihood der Validierungsdaten (von der stets eine Kopie behalten wird), abspeichern.

Des weiteren enthält der *GTM_Trainer* die Prädiktionskarten, die für die Berechnung des topographischen Produkts benötigt werden.

[2] Diese Funktion nutzt die 1992 verabschiedete Regeländerung des C++-Standards aus, daß überladene virtuelle Funktionen, die einen Zeiger oder eine Referenz auf die Basisklasse zurückliefern, in der abgeleiteten Klasse einen Verweis auf die Unterklasse zurückliefern dürfen. Dies ist die einzige Stelle, an der der Ergebnistyp einer Funktion in der Unterklasse nicht exakt mit dem in der Basisklasse übereinstimmen muß. ([Str94], S. 373-375)

[3] Verschiedene Versuche mit Online-EM-Algorithmus, Random Walk, Hillclimber oder 1+1-Verfahren haben sich als unnötig oder unbrauchbar herausgestellt.

#### Plan

Um das Training eines Netzes bequemer zu gestalten, habe ich die Klasse *Plan* geschrieben. Diese liest aus einer Plan-Datei (siehe Abschnitt A.2.3) die Parameter einer GTM-Karte ein und führt das Training in der *execute()*-Funktion aus. Man kann aber auch mehrere Karten mit einem Plan trainieren, indem man Schleifen über verschiedene Werte von $\lambda$, $\sigma$, $K$ oder $M$ bildet.

### A.1.4 Parameterklassen

Es gibt zwei Klassen für Parameter: eine für die GTM-Karte und eine für die Daten und das Training.

#### GTM-Parameter

Die Klasse *GTMparam* enthält *rootK*, *rootM*, $L$, $Q$, $\lambda$, $\sigma$, den Pfad zu den Prädiktionskarten, eine Initialisierungsmethode (zufällig, PCA, PCA mit EM-Algorithmus gemäß Abschnitt 5.2 und Prädiktionskarten gemäß Abschnitt 4.4.3) sowie die Initialisierung der semilinearen Karte (lineare Komponenten auf die ersten Hauptkomponenten, auf die letzten Hauptkomponenten oder auf die $(1, 1, \ldots, 1)$-Richtung festgelegt).

#### Lernparameter

Die Klasse *Lernparameter* enthält einen Lernmodus, eine Stapelgröße (nur für den obsoleten Hillclimber nötig), eine Metrik zur Behandlung der Dämpfung (siehe Abschnitt 5.3) und einige Boolesche Variablen, die angeben, ob nicht empfangene Komponenten ignoriert oder auf $t_{min} - 1$ gesetzt werden sollen, ob die Serverkomponente weggelassen werden soll, ob die Datensätze gemischt werden sollen (dies ist in der Regeln nicht notwendig, da die Eingabedaten schon vorher gemischt wurden; siehe Abschnitt 5.7.5) und ob Early Stopping verwendet werden soll. Für das Early Stopping enthält die Klasse auch den Bruchteil der Daten, der für die Validierungs- und Testmenge reserviert werden soll.

Diese Parameterklassen können an die Konstruktoren von *GTM* bzw. *TVektorDaten* übergeben werden; dadurch werden lange Parameterlisten der Konstruktoren vermieden.

### A.1.5 Hilfsklassen

#### Abstandsklassen

Das Modell enthält verschiedene Methoden, den Abstand zwischen einem Datensatz und einem Gaußblob zu bestimmen. Für den gewöhnlichen GTM-Algorithmus ist der quadratische Abstand zu berechnen (Gleichungen 4.1 und 4.13), für das semilineare GTM-Modell braucht man den Abstandsvektor $t_n - y(x_k, W)$ (Gleichung 5.18). Üblicherweise

ist dies der gewöhnliche euklidische Abstand, bei fehlenden Daten muß man jedoch die nicht vorhandenen Komponenten ignorieren (siehe Gleichung 5.1). Zur bequemen Unterscheidung dieser beiden Modelle dienen die beiden Metrikklassen *Abstand* und *AbstandIN* (für „Ignore Negative“). Die Klasse *Daten* enthält eine solche Metrikklasse, so daß das jeweils korrekte Abstandsmaß benutzt wird.

#### Prädiktionskarten und Distanzmatrizen

Prädiktionskarten dienen zu zwei Zwecken: zur Initialisierung der GTM-Karte und für das topographische Produkt. Die Klasse *Praediktion* enthält für die Initialisierung Funktionen, um die Bounding Box der Zelle zu finden und die $y_k^{pred}$ zu berechnen. Des weiteren enthält sie die Berechnung des topographischen Produkts, wofür zusätzlich die Hilfsklasse *Distanzmatrix* gebraucht wird.

#### FileIO

Um bequem auf Dateien zugreifen zu können, habe ich eine Klasse namens *FileIO* geschrieben, die es erlaubt, Variablen in Dateien zu schreiben oder aus diesen zu lesen. Diese können vom Typ int, unsigned, double, string oder matrix sein. Das Format ist einfach „Typ Name=Wert“. Für eine Matrix ist es etwas komplizierter: „matrix Name(Zeilen,Spalten)=(Werte...)“. Es läßt sich also leicht nach einer Variable durchsuchen. Die Reihenfolge des Abspeicherns spielt dabei keine Rolle, sofern keine Variable mehrmals vorkommt.

Mit Hilfe dieser kleinen Klasse kann man sehr bequem alle Variablen einer GTM-Instanz in eine Datei schreiben. Auch die Plan-Dateien werden mit dieser Klasse eingelesen.

### A.1.6 Klassen der graphischen Benutzeroberfläche

#### Fensterklassen

Die Klasse *GTMApplication* implementiert das Hauptfenster der Anwendung *xgtm*. In ihr werden die Knöpfe, die Statuszeile und die Menüs vereinbart.

Das Fenster, in dem die GTM-Karte im Datenraum dargestellt wird (projiziert auf zwei Dimensionen), ist in der Klasse *GTMPaint* implementiert. Es enthält die Möglichkeit, zwei Dimensionen zur Darstellung auszuwählen, sowie einen Punkt zu markieren, der im Wahrscheinlichkeitsdichte- oder Magnification Factors-Fenster angeklickt worden ist.

Diese Fenster sind in der Klasse *MapWindow* implementiert. Ein *MapWindow* ist ein geteiltes Fenster, das je ein Objekt *MapShow* (der eigentliche Darstellungsbereich des latenten Raums) und *MapLegende* (zur Angabe der Graustufenskala und der Werte eines angeklickten Punktes) enthält.

Klickt man einen Punkt $x$ in einem solchen Fenster an, so wird einerseits der Feldstärkevektor $y(x, W)$ in der Legende angezeigt, außerdem die Wahrscheinlichkeitsdichte

bzw. der Magnification Factor an diesem Punkt. Zusätzlich wird im Hauptfenster der angeklickte Punkt markiert.

Die Klasse *MapShow* benutzt die Klasse *Maler*, die ein Bitmap mit einer Visualisierung der GTM-Karte enthält. Auf diese Klasse *Maler* greift auch die Anwendung *gtmtopng* zurück (siehe Abschnitt 6.2.1). Sie stellt den Magnification Factor der Karte in Graustufen dar. Zusätzlich kann eine Datenmenge in das Fenster gezeichnet werden.

#### Visualisierungsklassen

Die Klasse *Maler* enthält die Möglichkeit, die Größe und Feinheit der Darstellung zu wählen, die Graustufenskala zu schätzen oder explizit zu setzen sowie den Puffer als PNG-Bilddatei abzuspeichern.

Von *Maler* abgeleitet ist die Klasse *MalerWD*, die lediglich den Unterschied hat, daß statt des Magnification Factors die Wahrscheinlichkeitsdichte dargestellt wird und daß die Graustufenskala umgekehrt ist (da einer hohen Wahrscheinlichkeitsdichte ein kleiner Magnification Factor entspricht).

Die Optionen der Darstellung der Klasse *Maler* sind in der Klasse *MalOptionen* abgelegt. Darin wird angegeben, welche Datenmenge eingetragen werden soll (keine, Trainings-, Validierungs-, Urteil- oder Maldaten), des weiteren, ob die Daten aus den selbsterzeugten Hotspots farbig dargestellt werden sollen, und ob die Hintergrunddaten, die keinem Hotspot entstammen, weggelassen werden sollen.

#### Dialogklassen

Schließlich gibt es drei Hilfsklassen für Dialoge.

*NewGTMDialog* beschreibt das Dialogfenster, das erscheint, wenn man den Menüpunkt „Neue GTM“ wählt (siehe Abbildung 6.1). Die Klasse *NewParDialog* enthält das Dialogfenster für die spätere Änderung der Parameter. Beide Fenster enthalten eine Instanz der Klasse *LernButtons*, die die Knöpfe zur Wahl des Lernalgorithmus beschreibt.

## A.2 Die Dateiformate

### A.2.1 Eingabedaten

Die T-Vektor-Dateien, die die GTM-Software einlesen kann, enthalten in jeder Zeile einen Datensatz. Dieser besteht aus den RxLev-Werten des Servers und aller Nachbarn. Für einen nicht empfangenen Nachbarn wird $-1$ eingetragen. Daran schließen sich zwei reservierte Zahlfelder für $t_{min}$ und BsRed an; der *AggregateLocator* (siehe Abschnitt 5.7) trägt jedoch hier stets Nullen ein. Am Schluß des Datensatzes kann noch ein Buchstabe stehen, der die Spotnummer angibt; $a$ steht für Spot 1, $b$ für Spot 2 und so fort.

Dieser Buchstabe wird einerseits benutzt für das Clustertrennungsmaß (siehe Abschnitt 5.5; dieses kann alle Buchstaben von *a* bis *z* unterscheiden), andererseits für die farbige Darstellung in den Visualisierungsfenstern. Dort stehen jedoch nur 13 Farben für die Spots von *a* bis *m* zur Verfügung (siehe Tabellen 7.1 oder 7.4).

### A.2.2 Prädiktionskarten

Die Prädiktionskarten können in demselben asc-Format eingelesen werden, das auch LocAbis lesen kann. Eine Datei enthält die Prädiktionskarte eines Senders für das ganze Gebiet. Sie beginnt mit einem solchen Header:

```
NCOLS 745
NROWS 460
XLLCORNER 24050
YLLCORNER 182310
CELLSIZE 1
```

Die ersten beiden Zeilen geben die Größe der Prädiktionskarte an. Die drei weiteren Zeilen sind für die GTM-Software irrelevant und werden überlesen. Es folgt eine Matrix der Prädiktionswerte, gegeben in dBm (wenn es berechnete Prädiktionskarten sind) oder als RxLev-Wert (wenn es Blackbox- oder Scankarten sind). Die Software kann automatisch zwischen den Formaten unterscheiden und die dBm-Werte gemäß Tabelle 3.1 umrechnen.

Der Konstruktor der Klasse *Praediktion* benötigt als Parameter eine Nachbarschaftsliste. Dies ist eine Datei, die lediglich die Dateinamen der Prädiktionskarten des Servers und der Nachbarn in der richtigen Reihenfolge enthält.

### A.2.3 Pläne

Ein Plan enthält alle Variablen, die zum Training einer GTM-Karte benötigt werden. Die Reihenfolge, in der diese Variablen stehen, ist unerheblich. Sie wirkt sich lediglich auf die Geschwindigkeit des Einlesens aus; vorteilhaft ist es natürlich, wenn die Felder in der Reihenfolge stehen, in der sie gesucht werden.

Ein Beispiel für einen Plan ist in Abbildung A.1 dargestellt.

Er beginnt mit der Versionsnummer des Planes. Es folgen die Lernparameter (siehe Abschnitt A.1.4); dies sind alle Variablen, die mit dem Präfix „Lern“ beginnen. Die meisten dieser Variablen sind selbsterklärend; bei Variable *Lernmodus* steht 0 für den EM-Algorithmus und 5 für Bayessches Lernen. Bei *LernLambdaMetrik* bedeutet 0 gewöhnliche euklidische Metrik, 1 Abstand zum Server („Salami-Metrik“) und 2 Means-Metrik (siehe Abschnitt 5.3).

Es folgt die Angabe der zu lernenden Daten.

```
int Plan_Version=100
int Lernparamversion=104
int Lernmodus=5
int Lernstapelgroesse=0
int LernIgnoreNegative=0
int LernServerWeglassen=0
int LernEarlyStopping=1
int LernNegAufTmin=0
int LernMischen=0
double LernValidAbzwack=0.333333333333
double LernUrteilAbzwack=0.333333333333
int LernLambdaMetrik=2
int Datentyp=2
string TDatendatei=/home/robin/da/eudat/mixede41a.t
int Datendimension=0
unsigned AnzahlTraining=0
unsigned AnzahlValidierung=0
unsigned AnzahlUrteil=0
string Maldatendatei=/home/robin/da/eudat/callse41a.t
unsigned rootK=15
unsigned L=2
unsigned rootM=15
unsigned Q=0
double begin_sigma=1
double end_sigma=5.1
double step_sigma=1
double lambda=2
int Semilin_Init=1
int Initialisierung=1
string Nachbarschaftsliste=/home/robin/da/eudat/channel/1x1/e41a.nbl
string Savefile=/home/robin/da/eures/meanse41
string Synopsisfile=/home/robin/da/eures/means41.synopsis
```

Abbildung A.1: Ein Beispielplan

- Datentyp 2 bedeutet T-Vektor-Daten. Es muß die Variable *TDatendatei* folgen, die den Pfad der Datei enthält. Optional kann man die Dimension der Daten und die Anzahl der Datensätze angeben; dies ist aber nicht notwendig, da diese Werte auch aus der Datei selbst bestimmt werden können.

  Außerdem kann noch eine getrennte Datei für die „Maldaten“ angegeben werden. Wenn dies getan wird, wird während des Trainings die Clustertrennung der Maldaten berechnet und mitprotokolliert.

- Datentyp 3 bedeutet Matrixdaten. Es folgen Matrizen mit den Trainings-, Validierungs- und Urteildaten.

Anschließend enthält der Plan die Parameter der GTM-Karte: *rootK* ( $\sqrt[L]{K}$, wobei $K$ die Anzahl der Gaußblobs ist), $L$ (die latente Dimension), *rootM* ( $\sqrt[L]{M}$; $M$ ist die Anzahl der Basisfunktionen), $Q$ (die Anzahl der linearen Achsen; $Q = 0$ bedeutet eine gewöhnliche GTM-Karte, $Q > 0$ eine semilineare), *sigma* (die Standardabweichung der Gaußschen Basisfunktionen) und *lambda* (die Gewichtsregulierungskonstante).

Für *rootK*, *rootM*, *sigma* und *lambda* kann man stattdessen auch Schleifen angeben, um mit einem Plan mehrere Karten zu trainieren. Dazu stellt man der jeweiligen Variable die Präfixe *begin_* , *end_* und *step_* vor, um Anfangswert, Endwert und Schrittweite festzulegen. Das Beispiel in Abbildung A.1 erzeugt zum Beispiel fünf Karten mit $\sigma = 1$, 2, 3, 4 und 5.

Des weiteren muß man die Initialisierungsmethode wählen (0 = zufällig, 1 = PCA, 2 = PCA mit EM-Algorithmus, 3 = Prädiktionskarten), sowie die Initialisierungsmethode für die semilineare Karte, sofern nötig (0 = lineare Komponenten zu den kleineren Eigenvektoren, 1 = lineare Komponenten zu den größeren Eigenvektoren, 2 = lineare Komponente auf $(1, 1, \ldots, 1)$ festlegen).

Schließlich gibt man die *Nachbarschaftsliste* der Prädiktionskarten an, das *Savefile* und das *Synopsisfile*, in dem eine kurze Statistik des Ergebnisses abgespeichert wird.

Das *Savefile* ist der Rumpf des Dateinamens, in dem die trainierte GTM-Karte abgespeichert wird. Der vollständige Dateiname wird folgendermaßen zusammengesetzt:

*(Savename),(Planname),K(rootK)M(rootM)L(lambda)S(sigma).gtm*

Die Log-Datei, in der nach jedem Lernschritt die aktuellen Werte der Log-Likelihood notiert werden, hat denselben Namen, nur mit dem Suffix *.log*.

### A.2.4 GTM

Die GTM-Dateien werden ebenfalls mit der Klasse *FileIO* geschrieben und gelesen. Demzufolge ist auch hier die Reihenfolge der Variablen im Grunde unerheblich. Da GTM-Dateien jedoch große Matrizen enthalten, ist es ratsam, beim Speichern die Reihenfolge einzuhalten, in der die Variablen gesucht werden.

Zu Beginn einer GTM-Datei befinden sich die *Lernparameter*. Die Namen und Aufgaben der Variablen sind identisch zu denen in den Plan-Dateien. Es folgen einige Felder, die der

*GTM_ Trainer* in die Datei schreibt: Eine Versionsnummer, die Anzahl der Lernschritte und das Abbruchkriterium der Stagnation, d. h. die Anzahl der Schritte, nach denen das Training beendet wird, wenn die Likelihood der Validierungsdaten nicht steigt.

Es folgen die Variablen der zu lernenden Daten, die ebenfalls identisch zu denen in den Plan-Dateien sind.

Dahinter befinden sich die Werte der *GTM* selbt. Zu Beginn steht eine Versionsnummer, die bei einem sich mit der Zeit entwickelnden Dateiformat sehr nützlich ist, um die Kompatibilität sicherzustellen. Sie wird aber auch zur Unterscheidung der gewöhnlichen von der semilinearen Variante verwendet. Die aktuelle Versionsnummer ist 103 für eine gewöhnliche *GTM* und 113 für eine *GTMsemilin*.

Der Versionsnummer folgt die Angabe der Zeit, zu der die Datei abgespeichert wurde. Dahinter stehen die GTM-Parameter $K$, $L$, $D$, $N$, $M$, *gridXdim*, *gridFIdim*, *sigma*, *beta*, *lambda* und *betaAnfang*. Letzteres ist der Wert, mit dem $\beta$ initialisiert wurde.

Es folgt die Initialisierungsmethode sowie die Variablen *beta_ may_ fall* (üblicherweise 1; man kann auch ein Absinken von $\beta$ verbieten, aber für den EM-Algorithmus ist dies nicht notwendig) und *ever_ same_ data* (obsolet, stets 1).

Dem schließen sich die Matrizen *Weights*, *X*, *My*, *Phi* und *YBlobs* an. Die Größe der GTM-Datei wird von *Phi*, der Matrix der Aktivierungen der Basisfunktionen $\Phi \in \mathbb{R}^{K \times M}$, dominiert.

Im Falle einer semilinearen Karte (Versionsnummer größer als 110) folgen zum Schluß die Variable $Q$ (Anzahl der linearen Achsen), die Matrix $V$ (die linearen Vektoren) und die Initialisierung der semilinearen Karte ((0 = lineare Komponenten zu den kleineren Eigenvektoren, 1 = lineare Komponenten zu den größeren Eigenvektoren, 2 = lineare Komponente auf $(1, 1, \ldots, 1)$ festlegen).

# B Abbildungsverzeichnis

Dieser Anhang enthält die Namen, Parameter und Werte aller Karten, die in dieser Arbeit dargestellt sind.

Angegeben sind

- das Lernverfahren (EM-Algorithmus oder Bayes),
- die Behandlung fehlender Daten ($-1$, $t_{min} - 1$ oder ignorieren),
- die Metrik (Euklid oder Means),
- die Initialisierungsmethode (PCA, PCA mit EM-Algorithmus oder Prädiktion),
- die Anzahl der Trainings-, Validierungs- und Testdatensätze,
- die Anzahl der Lernschritte,
- die Log-Likelihood der Trainings-, Validierungs- und Testdaten,
- die inverse Varianz der Gaußblobs $\beta$,
- die Clustertrennung der bekannten Spots,
- das topographische Produkt der Karte,
- die GTM-Parameter $K$, $L$, $D$, $M$, $\sigma$ und $\lambda$ (wobei $\lambda$ beim Bayesschen Lernen vom Algorithmus optimiert wird) und
- die Visualisierungsmethode und Graustufenskala.

**Abb. 7.3, S. 70, links:** synthres/autolam,autolam1,K10dM10dL2.000000S2.000000.gtm
Bayes, MD:$-1$, Euklid, Init PCA, $N_{train} = 750$, $N_{val} = 375$, $N_{test} = 375$
17 Lernschritte, LLH $T = -19.019540$, $V = -21.666299$, $U = -20.061282$
$\beta = 0.586509$, ClusterSep $= 1.745586$, ToProd $= -0.272690$
$K = 100$, $L = 2$, $D = 9$, $M = 103$, $\sigma = 2$, $\lambda = 7.929485$
Magnification Factors-Karte Skala 391.257 bis 2956.05

**Abb. 7.3, S. 70, rechts:** synthres/auto,meansauto_1,K15dM15dL2.000000S2.000000.gtm
Bayes, MD:$-1$, Means, Init PCA, $N_{train} = 500$, $N_{val} = 500$, $N_{test} = 500$
16 Lernschritte, LLH $T = -18.550472$, $V = -20.952098$, $U = -20.117118$
$\beta = 0.490675$, ClusterSep = 1.238052, ToProd = $-0.031049$
$K = 225$, $L = 2$, $D = 9$, $M = 228$, $\sigma = 2$, $\lambda = 48.287712$
Magnification Factors-Karte Skala 301.742 bis 2002.41

**Abb. 7.6, S. 76:** bmres/abaybm1,bmbayes1,K15dM8dL2.000000S2.000000.gtm
Bayes, MD:$-1$, Euklid, Init PCA, $N_{train} = 1390$, $N_{val} = 696$, $N_{test} = 696$
20 Lernschritte, LLH $T = -21.382418$, $V = -21.628105$, $U = -21.777619$
$\beta = 0.108620$, ClusterSep = 1.942738, ToProd = $-0.518295$
$K = 225$, $L = 2$, $D = 7$, $M = 67$, $\sigma = 2$, $\lambda = 2.040109$
Links Wahrscheinlichkeitsdichtekarte Skala 3.62653e-10 bis 3.13661e-08
Rechts Magnification Factors-Karte Skala 1000 bis 2500

**Abb. 7.7, S. 77:** eures/abayese413,abayes41_3,K15dM15dL2.000000S2.000000.gtm
Bayes, MD:$-1$, Euklid, Init Prädikt., $N_{train} = 7310$, $N_{val} = 7310$, $N_{test} = 7310$
21 Lernschritte, LLH $T = -34.263742$, $V = -34.533972$, $U = -34.474807$
$\beta = 0.038367$, ClusterSep = 7.886835, ToProd = $-0.136798$
$K = 225$, $L = 2$, $D = 10$, $M = 228$, $\sigma = 2$, $\lambda = 28.163364$
Links Magnification Factors-Karte Skala 0 bis 5000
Rechts Wahrscheinlichkeitsdichtekarte Skala 1.81682e-17 bis 7.25796e-13

**Abb. 7.8, S. 78:** eures/a20bayese432,a20bayes432,K15dM15dL2.000000S2.000000.gtm
Bayes, MD:$-1$, Euklid, Init Prädikt., $N_{train} = 9871$, $N_{val} = 9872$, $N_{test} = 9872$
19 Lernschritte, LLH $T = -44.210810$, $V = -44.618914$, $U = -44.741911$
$\beta = 0.039697$, ClusterSep = 3.753610, ToProd = $-0.098272$
$K = 225$, $L = 2$, $D = 13$, $M = 228$, $\sigma = 2$, $\lambda = 5.160844$
Links Wahrscheinlichkeitsdichtekarte Skala 6.26351e-21 bis 1.38776e-16
Rechts Magnification Factors-Karte Skala 552.111 bis 7022.81

**Abb. 7.9, S. 79, links:** bmres/schleifbm,schleifbm2,K15dM15dL1.500000S2.000000.gtm
EM-Alg., MD:$t_{min} - 1$, Euklid, Init PCA, $N_{train} = 1390$, $N_{val} = 696$, $N_{test} = 696$
13 Lernschritte, LLH $T = -19.914501$, $V = -21.036131$, $U = -21.011479$
$\beta = 0.178224$, ClusterSep = 2.142077, ToProd = $-0.636221$
$K = 225$, $L = 2$, $D = 7$, $M = 228$, $\sigma = 2$, $\lambda = 1.5$
Magnification Factors-Karte Skala 94.508 bis 2038.85

**Abb. 7.9, S. 79, rechts:** bmres/schleifbm,schleifbm3,K15dM15dL2.500000S2.500000.gtm
EM-Alg., MD:Ign., Euklid, Init EM-PCA, $N_{train} = 1390$, $N_{val} = 696$, $N_{test} = 696$
11 Lernschritte, LLH $T = -18.722394$, $V = -20.132906$, $U = -19.975691$
$\beta = 0.267316$, ClusterSep = 1.653989, ToProd = $-0.331296$
$K = 225$, $L = 2$, $D = 7$, $M = 228$, $\sigma = 2.5$, $\lambda = 2.5$
Magnification Factors-Karte Skala 313.045 bis 2348.05

**Abb. 7.10, S. 80, links:** eures/meanse41,means41_1,K15dM15dL2.000000S2.000000.gtm
Bayes, MD:−1, Means, Init PCA, $N_{train} = 7310$, $N_{val} = 7310$, $N_{test} = 7310$
73 Lernschritte, LLH $T = -32.479912$, $V = -32.990230$, $U = -32.906097$
$\beta = 0.061441$, ClusterSep $= 5.904133$, ToProd $= -0.082105$
$K = 225$, $L = 2$, $D = 10$, $M = 228$, $\sigma = 2$, $\lambda = 14.262013$
Magnification Factors-Karte Skala 352.929 bis 4280.15

**Abb. 7.10, S. 80, rechts:** eures/meanse43,means43_3,K15dM15dL2.000000S2.000000.gtm
Bayes, MD:−1, Means, Init Prädikt., $N_{train} = 5310$, $N_{val} = 5311$, $N_{test} = 5311$
83 Lernschritte, LLH $T = -42.835604$, $V = -43.743983$, $U = -43.744823$
$\beta = 0.049383$, ClusterSep $= 5.209101$, ToProd $= -0.026527$
$K = 225$, $L = 2$, $D = 13$, $M = 228$, $\sigma = 2$, $\lambda = 4.484541$
Magnification Factors-Karte Skala 1259.3 bis 5398.84

**Abb. 7.11, S. 81:** bmres/schleifbm,schleifbm5,K15dM15dL1.500000S2.000000.gtm
EM-Alg., MD:−1, Euklid, Init PCA, $N_{train} = 1390$, $N_{val} = 696$, $N_{test} = 696$
13 Lernschritte, LLH $T = -22.082853$, $V = -22.422272$, $U = -22.841072$
$\beta = 0.156343$, ClusterSep $= 1.946543$, ToProd $= -0.939641$
$K = 225$, $L = 2$, $D = 7$, $M = 228$, $\sigma = 2$, $\lambda = 1.5$
Links Magnification Factors-Karte Skala 0 bis 1500
Rechts Wahrscheinlichkeitsdichtekarte Skala 2.91844e-14 bis 2.08809e-07

**Abb. 7.12, S. 82:** eures/schleifTP41seml_praed,freeseml41_0,K15dM15dL1.500000S2.000000.gtm
EM-Alg., MD:−1, Euklid, Init Prädikt., $N_{train} = 7310$, $N_{val} = 7310$, $N_{test} = 7310$
19 Lernschritte, LLH $T = -34.305090$, $V = -34.713075$, $U = -34.578083$
$\beta = 0.053513$, ClusterSep $= 5.790732$, ToProd $= -0.292468$
$K = 225$, $L = 2$, $D = 10$, $M = 228$, $\sigma = 2$, $\lambda = 1.5$
Links Magnification Factors-Karte Skala 0 bis 5000
Rechts Wahrscheinlichkeitsdichtekarte Skala 3.68393e-19 bis 3.26174e-12

**Abb. 7.13, S. 82, links:** eures/fine41,fine41_3a,K21dM11dL2.000000S4.000000.gtm
Bayes, MD:−1, Euklid, Init Prädikt., $N_{train} = 7310$, $N_{val} = 7310$, $N_{test} = 7310$
62 Lernschritte, LLH $T = -34.312996$, $V = -34.614230$, $U = -34.540127$
$\beta = 0.042735$, ClusterSep $= 3.396644$, ToProd $= -0.151195$
$K = 441$, $L = 2$, $D = 10$, $M = 124$, $\sigma = 4$, $\lambda = 0.055770$
Magnification Factors-Karte Skala 16775.1 bis 212810

**Abb. 7.13, S. 82, rechts:** eures/abayese433,abayes43_3,K15dM15dL2.000000S2.000000.gtm
Bayes, MD:−1, Euklid, Init Prädikt., $N_{train} = 5310$, $N_{val} = 5311$, $N_{test} = 5311$
24 Lernschritte, LLH $T = -44.374363$, $V = -45.100785$, $U = -45.105452$
$\beta = 0.037507$, ClusterSep $= 2.942375$, ToProd $= -0.245472$
$K = 225$, $L = 2$, $D = 13$, $M = 228$, $\sigma = 2$, $\lambda = 19.777715$
Magnification Factors-Karte Skala 450.881 bis 6531.7

**Abb. 7.14, S. 83, links:** eures/fine41,fine41_4a,K21dM11dL2.000000S3.000000.gtm
Bayes, MD:$t_{min}-1$, Euklid, Init Prädikt., $N_{train} = 7310$, $N_{val} = 7310$, $N_{test} = 7310$
59 Lernschritte, LLH $T = -30.160841$, $V = -30.547212$, $U = -30.514516$
$\beta = 0.106040$, ClusterSep $= 5.671946$, ToProd $= -0.044016$
$K = 441$, $L = 2$, $D = 10$, $M = 124$, $\sigma = 3$, $\lambda = 0.455998$
Magnification Factors-Karte Skala 4206.68 bis 26311.7

**Abb. 7.14, S. 83, rechts:** eures/meanse43,means43_2,K15dM15dL2.000000S3.000000.gtm
EM-Alg., MD:$-1$, Means, Init PCA, $N_{train} = 5310$, $N_{val} = 5311$, $N_{test} = 5311$
25 Lernschritte, LLH $T = -43.252456$, $V = -44.188562$, $U = -44.224840$
$\beta = 0.055676$, ClusterSep $= 4.357428$, ToProd $= -0.021991$
$K = 225$, $L = 2$, $D = 13$, $M = 228$, $\sigma = 3$, $\lambda = 2$
Magnification Factors-Karte Skala 2501.14 bis 7613.4