

DIPLOMSTUDIENGANG GEODÄSIE

DIPLOMARBEIT

von

Ribana Roscher

geboren am 30.05.1984 in Erlabrunn

Lernen linearer probabilistischer diskriminativer Modelle für die
semantische Bildsegmentierung

Bonn 2008

Betreuer:

Prof. Dr.-Ing. Wolfgang Förstner

Ing. Filip Korč

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN
INSTITUT FÜR GEODÄSIE UND GEOINFORMATION
PROFESSUR FÜR PHOTOGRAMMETRIE

Diplomaufgabe

für Frau cand. geod. Ribana Roscher

Lernen linearer probabilistischer diskriminativer Modelle für die semantische Bildsegmentierung

Das Ziel der Diplomarbeit ist die Untersuchung linearer Modelle bei der semantischen Bildsegmentierung. Diese Arbeit behandelt das Thema unter einem probabilistischen Gesichtspunkt mit besonderem Schwerpunkt auf dem diskriminativen Training.

Für eine große Anzahl von Verteilungen im Kontext der Klassifikation sind die a posteriori Wahrscheinlichkeiten durch eine normierte Exponentialtransformation linearer Funktionen der Merkmalsvektoren gegeben. Diese Arbeit betrachtet die Maximum Likelihood Schätzung, d.h., das Lernen, der darin enthaltenen unbekanntem Modellparameter, welche ein konvexes Optimierungsproblem ohne Nebenbedingungen darstellt.

Dieser Ansatz soll auf die semantische Bildsegmentierung angewandt werden. Für diesen Zweck wird das trainierte Klassifikationsmodell in ein Markov Random Field (MRF) eingebunden, welches die Interaktion zwischen benachbarten Klasseninstanzen zulässt. Zusätzliche MRF Parameter werden durch eine Kreuzvalidierung bestimmt, wobei der Klassifikationsfehler auf einem Trainingsdatensatz minimiert wird. Die semantische Segmentierung erfolgt durch Loopy Belief Propagation nach Pearl.

Die Leistungsfähigkeit der Klassifikation wird anhand der drei im Bereich Computer Vision weit verbreiteten Datensätze demonstriert, die Bilder realer Objekte beinhalten: (i) der MSRC 23-Klassen Datensatz, (ii) der 7-Klassen Corel Datensatz und (iii) der 7-Klassen Sowerby Datensatz.

Prof. Wolfgang Förstner

Betreuer: W. Förstner, F. Korč

Ausgegeben am: 15. August 2008

Abgabetermin: 15. Februar 2009

Abgegeben am:

Bonn, den 19. Dezember 2008

Erklärung

Hiermit erkläre ich eidesstattlich, dass ich die vorgelegte Arbeit selbstständig und ohne unzulässige Hilfe angefertigt habe. Es wurden keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ribana Roscher

Inhaltsverzeichnis

Notation	IX
1 Einleitung	1
1.1 Motivation	1
1.2 Aufgabenstellung	4
1.3 Verwandte Arbeiten	5
1.4 Aufbau der Arbeit	6
2 Lernen linearer probabilistischer diskriminativer Modelle	9
2.1 Modelle in der Klassifikation	9
2.2 Lineare diskriminative Modelle	12
2.3 Probabilistische Modelle	16
2.4 Logistische Regression	19
3 Merkmalsextraktion	25
3.1 Superpixel	27
3.2 Merkmale	29
4 Segmentierung unter Verwendung von Markov Random Fields	43
4.1 Aufbau	44
4.2 Energiefunktion	45
4.3 Lernen der Parameter	47
4.4 Inferenz	48
5 Umsetzung und Implementierung	59
5.1 Allgemeines	59
5.2 Superpixel- und Merkmalsextraktion	65
5.3 Datenreduktion	66
5.4 Trainieren der Diskriminantenparameter	69
5.5 Maximum a posteriori Schätzung mit dem linearen probabilistischen diskriminativen Modell	70
5.6 Segmentierung mit Markov Random Fields	71
5.7 Evaluierung	71

6	Ergebnisse	77
6.1	Anwendung auf die synthetischen Datensätze	77
6.2	Anwendung auf die realen Datensätze	80
7	Zusammenfassung und Ausblick	105
7.1	Zusammenfassung	105
7.2	Ausblick	105
A	Ableitungen	109
1.1	Ableitung der a posteriori Wahrscheinlichkeiten nach den Aktivierungen .	109
1.2	Ableitung der Kreuzentropie-Fehlerfunktion nach den Parametervektoren	110
B	Arbeitsfluss	112
C	Tabellen	113
3.1	Variation der Schrittweite	113
3.2	Variation der Variation der Merkmale	114
3.3	Variation der Punktzahl	115
D	Konfusionsmatrizen	116
4.1	Sowerby Datensatz	116
4.2	Corel Datensatz	118
4.3	MSRC Datensatz	120
	Abbildungsverzeichnis mit Referenzen	125
	Literatur	129
	Stichwortverzeichnis	130

Notation

Symbol	Bedeutung
a	Aktivierung
C	Clique
\mathcal{C}	Menge aller Klassen
D	Länge der Eingangsvektoren (Länge der Merkmalsvektoren + 1)
g	Wert der Diskriminantenfunktion
\mathbf{H}^+	Pseudoinverse der Hessematrix
I	Anzahl der Bilder in einem Datensatz
\mathbf{I}	Einheitsmatrix
\mathcal{I}	Menge aller Bilder
K	Anzahl der Klassen
\mathbf{K}	Konfusionsmatrix
KE	Klassifikationserfolg
M	Dimension des Merkmalsraumes, Länge der Merkmalsvektoren
N	Anzahl der Pixel im Bild
\mathcal{R}	Entscheidungsregion
s	Schrittweite
t	Zielvektor
\mathbf{T}	Matrix aller Zielvektoren
\mathbf{w}	Diskriminante, Parametervektor
w_0	Bias (negativer Schwellwert) der Diskriminante
\mathbf{W}	Matrix aller Diskriminanten
x	Variablenknoten mit Zustandswert
\mathbf{x}	Vektor aller Variablenknoten
\mathbf{X}	klassifiziertes Bild
y	Merkmal
\mathbf{y}	Merkmalsvektor
\mathbf{y}	erweiterter, homogener Merkmalsvektor
\mathbf{Y}	Matrix aller erweiterten Merkmalsvektoren
Z	Normalisierungskonstante, Partition Function
$\mathbf{1}$	Vektor bestehend aus Einsen
$\delta(\cdot, \cdot)$	Deltafunktion

ϕ	transformierter Merkmalsvektor
Φ	Matrix der transformierten Merkmalsvektoren
\propto	ist proportional zu
Δ	Nabla-Operator, Gradient
\ln	natürlicher Logarithmus
$f(\cdot)$	Transformationsfunktion der Diskriminanten
$f_m(\cdot)$	Faktorknoten des m-ten Variablenknotens
$F(\cdot)$	Kreuzentropie
$E(\cdot)$	Energie
$\mathcal{N}(m)$	Menge der Variablenknoten
$\mathcal{M}(n)$	Menge der Faktorknoten
$\psi(\cdot)$	Potentialfunktion
$p(\cdot)$	Wahrscheinlichkeit
$q_{(\cdot) \rightarrow (\cdot)}$	Variable-zu-Faktor Nachricht
$r_{(\cdot) \rightarrow (\cdot)}$	Faktor-zu-Variable Nachricht
$\text{rand}(\cdot, \cdot)$	zufällig gewählte Werte im Intervall $[\cdot, \cdot]$
$\sigma(\cdot)$	Sigmoidfunktion
$\{\cdot\}$	Menge
$\prod_C(\cdot)$	Produkt über alle Cliques C
$\sum_{\mathbf{x}}(\cdot)$	Summe über alle Zustände von \mathbf{x}
$\sum_{\mathbf{x} \setminus x}(\cdot)$	Summe über alle Zustände von \mathbf{x} außer x
$[\cdot, \cdot]$	abgeschlossenes Intervall
$ \cdot $	Betrag eines Vektors
$\angle(\cdot, \cdot)$	durch zwei Vektoren eingeschlossener Winkel

1 Einleitung

Die visuelle Wahrnehmung eines Bildes erfolgt für den Menschen unbewusst unter Einbeziehung einer Fülle von Erfahrungen. Bei der Betrachtung einer Szene unterteilt er sie in Objekte und ordnet ihnen eine sinnvolle Bedeutung zu. Autonome Computersysteme müssen diese Erfahrungen erst im Laufe eines Lernprozesses erlangen. Für diese Systeme besteht eine Bildszene aus einer Ansammlung von Pixeln mit bestimmten Eigenschaften, jedoch ohne Objektbezug. Diese Arbeit setzt an diesem Punkt an und stellt ein Verfahren vor, das die für sich allein betrachteten, bedeutungslosen Pixel in einen semantischen Zusammenhang bringt.

1.1 Motivation

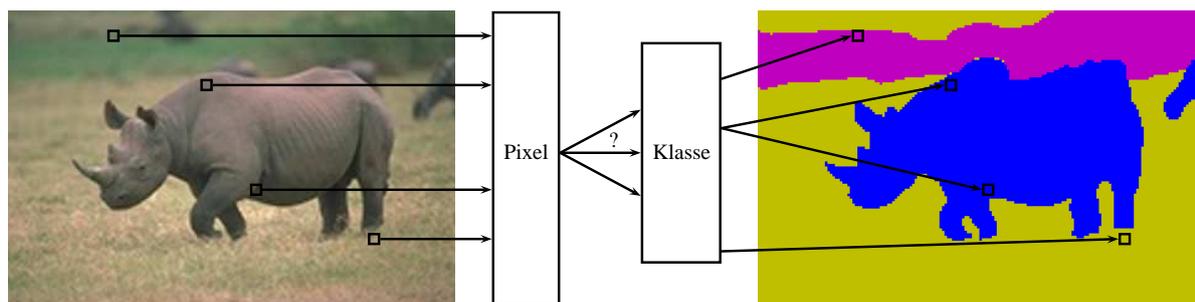


Abbildung 1.1: Die aus einem Bild stammenden Pixel können einer Klasse zugeordnet werden, die aus einem oder mehreren Objekten besteht. Durch die Zuordnung entsteht ein semantisch segmentiertes Bild.

Abbildung 1.1 stellt das soeben beschriebene Prinzip grafisch dar. Um ein semantisch segmentiertes Bild zu erhalten, ordnen wir den aus dem Bild stammenden Pixeln eine Klasse zu, zum Beispiel „Nashorn“, die ein oder mehrere Objekte beinhaltet, zum Beispiel „Horn“, „Ohr“ oder „Haut“.

Abbildung 1.2a-1.2c zeigt Bilder aus den in dieser Arbeit verwendeten Datensätzen. Trotz der Verschiedenartigkeit der Bilder haben wir keine Probleme, die Objekte in diesen Bildern zu erkennen. Dies gelingt durch das Erkennen der Eigenschaften und dem Zuordnen zu Objekten, die uns bereits bekannt sind. Die Zuordnung gelingt selbst dann, wenn Objektteile verdeckt sind, die Beleuchtung oder der Betrachtungswinkel variiert



(a) Bild einer Pflanze aus dem MSRC Datensatz



(b) Bild eines Flusspferdes aus dem Corel Datensatz



(c) Bild einer Landstraße aus dem Sowerby Datensatz

Abbildung 1.2: Die Bilder aus den in dieser Arbeit verwendeten Datensätzen enthalten verschiedenartige Objekte, die wir als Mensch ohne Probleme erkennen können.

oder einige Eigenschaften wie die Farbe oder die Form anders sind. Diese Problematik stellt eine große Herausforderung für autonome Computersysteme dar.

Eine Methode, die sich mit der Zuordnung von Objekten beschäftigt, ist die Bildsegmentierung. Sie ist ein Gebiet des Computer Vision (engl. für Maschinelles Sehen) und kann auf zwei Weisen erfolgen. Während die konturenbasierte Segmentierung mit Hilfe der Kantendetektion erfolgt, nutzt die regionenbasierte Zuordnung weitaus mehr Eigenschaften des Bildes, wie Farbe, Textur, Helligkeit oder auch die Pixelposition. Pixel mit ähnlichen Eigenschaften werden zu einer Region gruppiert. Die *semantische Bildsegmentierung* geht noch einen Schritt weiter und gruppiert Pixel mit Eigenschaften, die denen eines gelernten Objektes am ähnlichsten sind.

Die Abbildungen 1.3a und 1.3b zeigen zwei Motive, bei denen die konturenbasierte und auch die regionenbasierte Segmentierung ohne semantischen Bezug andere Ergebnisse liefern würde, als eine semantische Segmentierung. Auch wenn ein Objekt mehrere Regionen mit sich stark unterscheidenden Merkmalen besitzt, können diese einem Objekt zugeordnet werden, da auf Grund vorhandener Daten bereits bekannt ist, dass all diese Merkmale zu einem Objekt gehören.

Die semantische Bildsegmentierung findet zum Beispiel in vielen Bereichen des täglichen Lebens Anwendung. So wurden in den letzten Jahren zunehmend Roboter entwickelt, die das Leben erleichtern sollen. In Abbildung 1.4a befüllt ein Roboter eine Spülmaschine mit Geschirr, welches er zuvor von einem Tisch abgeräumt hat¹. Bisher ist er noch von Hand gesteuert. Die Entwicklung zu einem autonomen Verhalten ist jedoch bereits

¹Video hierzu unter http://anybots.com/videos.html#monty_does_dishes_title



(a) Leopard



(b) Strandball

Abbildung 1.3: Der Leopard und der Strandball sind Objekte mit mehreren Regionen. Aus diesem Grund liefert eine konturen- oder regionenbasierte Segmentierung andere Ergebnisse als eine semantische Bildsegmentierung.

aktuelles Forschungsthema und beinhaltet auch das Erkennen von Objekten mit Hilfe von Kameras, durch das er Geschirr von Nicht-Geschirr unterscheiden und entsprechend darauf reagieren kann.

Ein weiteres Anwendungsgebiet ist die 3D-Objekterfassung aus Punktwolken. In diesem Bereich können die semantischen Bildinformationen bereits Vorinformationen über die gescannten Objekte liefern. Zum Beispiel sind Häuserfassaden in einer Punktwolke nur schwer von Fenstern trennbar. In Bildern sind beide Objekte jedoch deutlich unterscheidbar. Mit Hilfe der Bildinformationen können den Punkten Bildmerkmale zugewiesen werden. Im Zuge einer 3D-Rekonstruktion mit Hilfe von geometrischen Objekten können für jedes Objekt auch tatsächlich nur die Punkte verwendet werden, die zu ihm gehören.

Die semantische Bildsegmentierung kann auch im Bereich des Anti-Terror Anwendung finden. Abbildung 1.4b demonstriert eine Maßnahme zur Wahrung der Sicherheit am Flughafen. Sie zeigt die Monitore eines Röntgenprüfgeräts für Reisegepäck am Suvarnabhumi International Airport in Bangkok. Die Kontrolle erfolgt durch Menschen. Da der Arbeitstakt meist sehr hoch und die Arbeit monoton und ermüdend ist, können Fehler auftreten. Die semantische Bildsegmentierung eignet sich sehr gut zur Unterstützung

der Erkennung von gefährlichen Gegenständen und kann somit zur Fehlerverminderung beitragen.



(a) Roboter beim Einräumen einer Spülmaschine



(b) Röntgenprüfgerät für Reisegepäck im Suvarnabhumi International Airport in Bangkok

Abbildung 1.4: Die Einsatzgebiete der semantischen Bildsegmentierung reichen vom täglichen Leben bis zur Überwachung am Flughafen, da überall semantische Bildinhalte benötigt werden.

1.2 Aufgabenstellung

Ziel dieser Arbeit ist das Lernen eines linearen probabilistischen diskriminativen Modells für die semantische Bildsegmentierung. Dabei soll ein Programm entwickelt werden, welches folgende Bestandteile enthält: die Merkmalsextraktion, die Vorverarbeitung der Daten, das Lernen der Parameter des linearen probabilistischen diskriminativen Modells, die Segmentierung und das Evaluieren. Die Segmentierung der Bilder erfolgt mit Hilfe von Markov Random Fields. Der Klassifikator soll anhand dreier im Bereich Computer Vision weit verbreiteter Datensätze bewertet werden. Diese sind (i) der MSRC 23-Klassen Datensatz, (ii) der 7-Klassen Corel Datensatz und (iii) der 7-Klassen Sowerby Datensatz.

1.3 Verwandte Arbeiten

In dieser Arbeit widmen wir uns der Aufgabe der Bildsegmentierung. Dazu wählen wir ein lineares probabilistisches diskriminatives Modell und stützen uns dabei auf die Betrachtungen von (Bishop, 2006, Kapitel 4) und Duda u. a. (2001, Kapitel 5). Mit diesem Ansatz können wir direkt die Wahrscheinlichkeiten der Klassen für jedes Pixel bestimmen.

Die Bestimmung der Parameter des Modells stellt ein konvexes Optimierungsproblem dar. Konvex bedeutet, dass wir eine von den Parametern abhängige Funktion minimieren, deren zweite Ableitung stets positiv ist. Diese Funktion besitzt somit ein einziges, globales Minimum. Die Minimierung lösen wir mit dem Gradientenabstiegsverfahren. Unter anderem verwenden wir das Exact Line Search Verfahren nach Boyd und Vandenberghe (2004, Kapitel 9, Seite 464) und stellen dieses Verfahren anderen gegenüber. Die Initialisierung der Parameter richtet sich nach Alpaydin (2008, Kapitel 10, Seite 222).

In den letzten Jahren wurde in vielen Arbeiten der Ansatz verfolgt, Bildsegmente statt Pixel für die Segmentierung zu verwenden. Diese Bildsegmente sind so genannte Superpixel. Die Idee rührte daher, dass Pixel in einem Segment oft die gleiche Klassenzugehörigkeit besitzen. Dieser Ansatz hat den Vorteil, dass ganze Regionen statt einzelner Pixel klassifiziert werden können. Zum Beispiel verwenden He und Zemel (2008) eine Implementation nach Shi und Malik (2000) und bilden den Mittelwert und Histogramme über alle Merkmale in einem Superpixel. Diesen Ansatz verfolgen wir in dieser Arbeit und analysieren die Verwendung der Superpixel. Dabei variieren wir die Merkmalsextraktion für die Superpixel und stellen die Ergebnisse einer Extraktion aus einfachen Rechteckumgebungen gegenüber.

In dieser Arbeit verwenden wir die in der Literatur häufig genutzten Merkmale. He und Zemel (2008), Shotton u. a. (2006) und He u. a. (2004) verwenden die Größe, die Form, die Position, die Farbe, die Textur und Kontextmerkmale für Superpixel beziehungsweise für Rechteckumgebungen. Einige Merkmale aus diesen Gruppen implementieren wir sowohl für Superpixel als auch für Rechteckumgebungen und führen mit diesen die Segmentierung durch. Die Ergebnisse stellen wir zusammen und diskutieren sie.

Ein besonderes Augenmerk in dieser Arbeit liegt dabei auf den Texturen und den Farbmerkmalen, die Malik u. a. (2001), Leung und Malik (2001) und Durupinar (2005) aus-

föhrlich betrachten. Leung und Malik (2001) stellen ein Verfahren zum Lernen von Texturen vor, mit dem sie Merkmale aus Texturen extrahieren und die Texturen durch einige wenige markante Merkmale charakterisieren. Wir verfolgen diesen Ansatz bis zur Extraktion der Merkmale und verwenden diese als unsere Texturmerkmale. Für die Extraktion der Texturen benutzen wir die Leung-Malik Filterbank, die aus 48 Filtern besteht.

Durupinar (2005) richtet seine Aufmerksamkeit auf die Extraktion von Merkmalen aus dem HSV-Farbraum und stellt einen Vergleich zur Extraktion durch Anwendung einer Filterbank an. Er kommt zu dem Schluss, dass Texturen unter verschiedenen Beleuchtungen nicht als ähnlich erkannt werden. Unter diesem Gesichtspunkt kombinieren wir beide Ansätze und extrahieren die Texturen nur aus dem Sättigungs- und dem Farbtonkanal des HSV-Farbraums, um eine Unabhängigkeit von der Beleuchtung zu erreichen.

Ein Möglichkeit der Bildsegmentierung ist die Verwendung von Markov Random Fields. Sie liefern die Wahrscheinlichkeit der Segmentierung eines Bildes durch einen probabilistischen, generativen Ansatz. Potts (1952) formuliert dafür eine Energiefunktion, die Bishop (2006, Kapitel 8) weiter ausführt und zu der Wahrscheinlichkeit für die Segmentierung in Bezug setzt. Des Weiteren beschreibt MacKay (2003, Kapitel 16, Kapitel 24) in seinem Buch die Anwendung der Markov Random Fields bei Bildern durch Loopy Belief Propagation nach Pearl. Diese Methoden greifen wir in der Arbeit auf und setzen sie um.

1.4 Aufbau der Arbeit

Kapitel 2 erläutert das Lernen der Parameter in einem linearen probabilistischen diskriminativen Modell. Darin gehen wir genauer auf die verwendete Art der Klassifikation ein und erläutern die Schätzung der Parameter mit dem Gradientenabstiegsverfahren. Kapitel 3 betrachtet die Extraktion der Merkmale, mit denen die Parameter des Modells gelernt werden. Wir zeigen Möglichkeiten auf, wie Merkmale extrahiert werden können, und erläutern diejenigen genauer, die in dieser Arbeit verwendet werden. Kapitel 4 erläutert die Grundlagen der Markov Random Fields. Es geht auf den Aufbau der Felder, die Inferenz und die Problematik bei der Anwendung auf Bilder ein. Insbesondere betrachtet es die Loopy Belief Propagation, die wir zur Segmentierung des Bildes verwenden.

Kapitel 5 betrachtet die Umsetzung der Verfahren und das für die Arbeit entwickelte Programm, um in Kapitel 6 die Ergebnisse vorzustellen, die damit erzielt wurden.

Kapitel 7 schließt die Arbeit mit einer Zusammenfassung und gibt einen Ausblick auf mögliche Weiterführungen oder Alternativen.

2 Lernen linearer probabilistischer diskriminativer Modelle

Die semantische Bildsegmentierung erfolgt durch eine Klassifikation, das heißt durch die Zuordnung der Pixel \mathbf{y} zu einer von K vorgegebenen diskreten Klassen \mathcal{C}_k mit $k = 1, \dots, K$. Die Zuordnung erfolgt durch ein Klassifikationsverfahren, dem ein parametrisches Modell zugrunde liegt, welches wir auf verschiedene Weisen lernen können.

Das folgende Kapitel geht zunächst näher auf die Klassifikation ein, um anschließend die Methoden des Lernens eines Modells zu erläutern. Besonderer Schwerpunkt liegt auf dem linearen probabilistischen diskriminativen Modell, welches wir in dieser Arbeit verwenden.

2.1 Modelle in der Klassifikation

Für die Klassifikation können wir je nach Anforderung und zur Verfügung stehenden Daten verschiedene Modelle lernen. Die nächsten Abschnitte erläutern die Klassifikation im Allgemeinen. Dabei gehen sie genauer auf die Beschreibung und Notation der Größen und die verschiedenen Arten der Klassifikation ein, um im folgenden Abschnitt das in dieser Arbeit gewählte Modell genauer betrachten zu können.

2.1.1 Beschreibung der Größen

Für die *Klassifikation* sind die Pixel \mathbf{y} durch M -dimensionale *Merkmalsvektoren* repräsentiert und in einer $(M \times N)$ -dimensionalen Matrix

$$\mathbf{Y} = [y_1 \cdots y_n \cdots y_N] \quad (2.1)$$

zusammengefasst. Die Anzahl der Pixel ist N . Die Merkmalsvektoren beschreiben die Eigenschaften der Pixel und werden in Kapitel 3 näher erläutert. Sie sind die Trainingsdaten, mit denen wir das Modell lernen, und die Testdaten, die wir klassifizieren.

Die Voraussetzung für die Klassifikation ist, dass sich alle Klassen ausschließen, sodass jeder Merkmalsvektor ausschließlich zu einer Klasse gehört. Der M -dimensionale Merkmalsraum, in denen die Merkmalsvektoren liegen, besteht aus Entscheidungsregionen, die durch $(M - 1)$ -dimensionale Entscheidungsgrenzen, so genannte Hyperflächen, abgeschlossen sind. Die Hyperflächen sind lineare Funktionen der Merkmalsmatrix \mathbf{Y} . Alle Bilder, deren Merkmalsvektoren wir durch lineare Hyperflächen exakt trennen können, bezeichnen wir als *linear separierbar*.

Der Zielvektor \mathbf{t} (engl. target vector) ist der Vektor mit reellen Werten, der die Klassenzugehörigkeit x , im Folgenden Label genannt, angibt. Der Vektor besitzt die Länge K und nimmt im Falle der Klasse \mathcal{C}_k an allen Stellen den Wert 0 an, außer an der Stelle k , an der er den Wert 1 annimmt. Für $x = 2$ und $K = 5$ beispielsweise ergibt sich folgender Zielvektor:

$$x = 2 \quad \iff \quad \mathbf{t} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (2.2)$$

Obwohl ein Label und ein Targetvektor dasselbe angeben, ist es in den meisten Fällen aus rechentechnischen Gründen sehr viel einfacher, mit den Zielvektoren zu arbeiten. Das Ergebnis der Klassifikation nach der Eingabe der Merkmalsmatrix \mathbf{Y} ist eine $(K \times N)$ -dimensionale Matrix \mathbf{T} , deren Spalten die Zielvektoren \mathbf{t}_n sind.

2.1.2 Methoden der Klassifikation

Im Folgenden wollen wir die Klassifikation anhand von Bishop (2006, Kapitel 4) und Duda u. a. (2001, Kapitel 5) näher diskutieren. Dazu verwenden wir das *Bayestheorem*

$$p(\mathcal{C}|\mathbf{Y}) = \frac{p(\mathbf{Y}|\mathcal{C})p(\mathcal{C})}{p(\mathbf{Y})}. \quad (2.3)$$

Die Annahme über die Wahrscheinlichkeit einer Klasse, bevor die Beobachtungen bekannt sind, ist in der a priori Wahrscheinlichkeit $p(\mathcal{C})$ repräsentiert. Die bedingte Wahrscheinlichkeit $p(\mathbf{Y}|\mathcal{C})$ gibt an, wie wahrscheinlich die Merkmale \mathbf{Y} bei einer gegebenen Klasse \mathcal{C} sind. Bei bekanntem \mathbf{Y} und variablem \mathcal{C} wird $p(\mathbf{Y}|\mathcal{C}) = L(\mathcal{C})$ zur Likelihood-

funktion $L(\mathcal{C})$. Die a posteriori Wahrscheinlichkeiten $p(\mathcal{C}|\mathbf{Y})$ geben die Unsicherheit der Klassen an. Der Nenner $p(\mathbf{Y})$ dient als Normalisierungskonstante und garantiert den Wahrscheinlichkeitscharakter der linken Seite des Bayestheorems (2.3). Die a posteriori Wahrscheinlichkeiten² liegen nach dem Normalisieren im Bereich $[0,1]$ und die Spalten summieren sich zu 1.

Die Klassifikation gliedert sich in zwei Phasen, der Trainings- und der Entscheidungsphase. In der Trainingsphase lernen wir mit Hilfe der Trainingsdaten das Modell für $p(\mathcal{C}|\mathbf{Y})$, um dann anhand dessen optimale Entscheidungen zu treffen. Wir können diese zwei Phasen jedoch auch zusammenfassen, indem wir eine Funktion lernen, die den Merkmalsvektor direkt in eine Entscheidung transformiert. Solch eine Funktion ist eine so genannte *Diskriminante*.

Es existieren drei grundlegende Ansätze zur Klassifikation:

Diskriminantenfunktion Das einfachste Modell verwendet die Berechnung von linearen Diskriminanten, die den Raum in Entscheidungsregionen teilen und einen Merkmalsvektor direkt einer Klasse zuordnen. Bei diesem Ansatz spielen Wahrscheinlichkeiten keine Rolle.

Diskriminatives Modell Dieses Modell berechnet die bedingten Wahrscheinlichkeiten $p(\mathcal{C}|\mathbf{Y})$ und trifft mit diesen optimale Entscheidungen. Die bedingten Wahrscheinlichkeiten werden direkt modelliert, zum Beispiel durch die Repräsentation des Modells durch Parameter und anschließender Optimierung.

Generatives Modell Dieses Modell berechnet die bedingten Wahrscheinlichkeiten $p(\mathcal{C}|\mathbf{Y})$ mit einem generativen Ansatz durch Modellierung der klassenbedingten Wahrscheinlichkeiten $p(\mathbf{Y}|\mathcal{C})$ der Merkmale und den a priori Wahrscheinlichkeiten $p(\mathcal{C})$. Mit Hilfe des Bayestheorems ergeben sich die a posteriori Wahrscheinlichkeiten, mit denen anschließend optimale Entscheidungen getroffen werden.

²Im Folgenden verwenden wir je nach Kontext $p(\cdot)$ als Wahrscheinlichkeit oder als Wahrscheinlichkeitsdichte. Die Wahrscheinlichkeiten eines Eingangsvektors \mathbf{x} besitzen die Eigenschaften einer Wahrscheinlichkeitsdichte $p(\mathbf{x})$, da sie jedoch diskret sind, bezeichnen wir sie im folgenden als Wahrscheinlichkeiten.

2.2 Lineare diskriminative Modelle

Bisher kennen wir drei Ansätze um die a posteriori Wahrscheinlichkeiten für die Klassifikation zu erhalten. In dieser Arbeit verwenden wir den zweiten Ansatz, das heißt wir ermitteln die Wahrscheinlichkeiten direkt ohne den „Umweg“ über das Bayestheorem. Die diskriminativen Modelle haben den Vorteil, dass wir keine a priori Informationen kennen müssen. Wir verwenden lineare Modelle, da sie einfach in der Anwendung sind, auf Grund ihrer geringen Parameterzahl auch mit wenig Daten gut zu lernen sind und in den meisten Anwendungen trotz ihrer geringen Ordnung zufriedenstellende Ergebnisse liefern.

Im Falle eines linearen Modells, das heißt das Modell ist linear in dem Merkmalsvektor \mathbf{y} , ergibt sich für ein Klassenpaar \mathcal{C}_1 und \mathcal{C}_2 eine Diskriminante $g(\mathbf{y})$, die wir zur Klassifikation verwenden:

$$g(\mathbf{y}) = \mathbf{w}^T \mathbf{y} + w_0. \quad (2.4)$$

Dies ist eine Hyperebene, die den Merkmalsraum in die Entscheidungsregionen \mathcal{R}_1 und \mathcal{R}_2 teilt. Im Folgenden betrachten wir die Bestandteile der Diskriminantenfunktion (2.4) anhand der Abbildung 2.1 näher, um sie anschließend auf das Mehrklassen-Problem verallgemeinern zu können. Die Abbildung zeigt eine Diskriminante g und einen Merkmalsvektor \mathbf{y} im Zweiklassen-Problem im dreidimensionalen Raum.

- Bias w_0 (negativer Schwellwert)

Der Bias-Parameter gibt den Ort der Diskriminante an. Nehmen wir einen Punkt $g(\mathbf{y}) = 0$ auf der Entscheidungsfläche an, dann gilt für die normalisierte Distanz vom Ursprung:

$$\mathbf{w}^T \mathbf{y} = |\mathbf{w}| |\mathbf{y}| \cos \angle(\mathbf{w}, \mathbf{y}), \quad (2.5)$$

$$\frac{\mathbf{w}^T \mathbf{y}}{|\mathbf{w}|} = |\mathbf{y}| \cos \angle(\mathbf{w}, \mathbf{y}) \quad (2.6)$$

$$= -\frac{w_0}{|\mathbf{w}|}. \quad (2.7)$$

- Merkmalsvektor \mathbf{y}

Der Merkmalsvektor \mathbf{y} besteht aus M Merkmalen $\{y_1, \dots, y_M\}$. Er wird der Klasse \mathcal{C}_1 zugeordnet, wenn $g(\mathbf{y}) \geq 0$, also genau dann, wenn das Punktprodukt $\mathbf{w}^T \mathbf{y}$

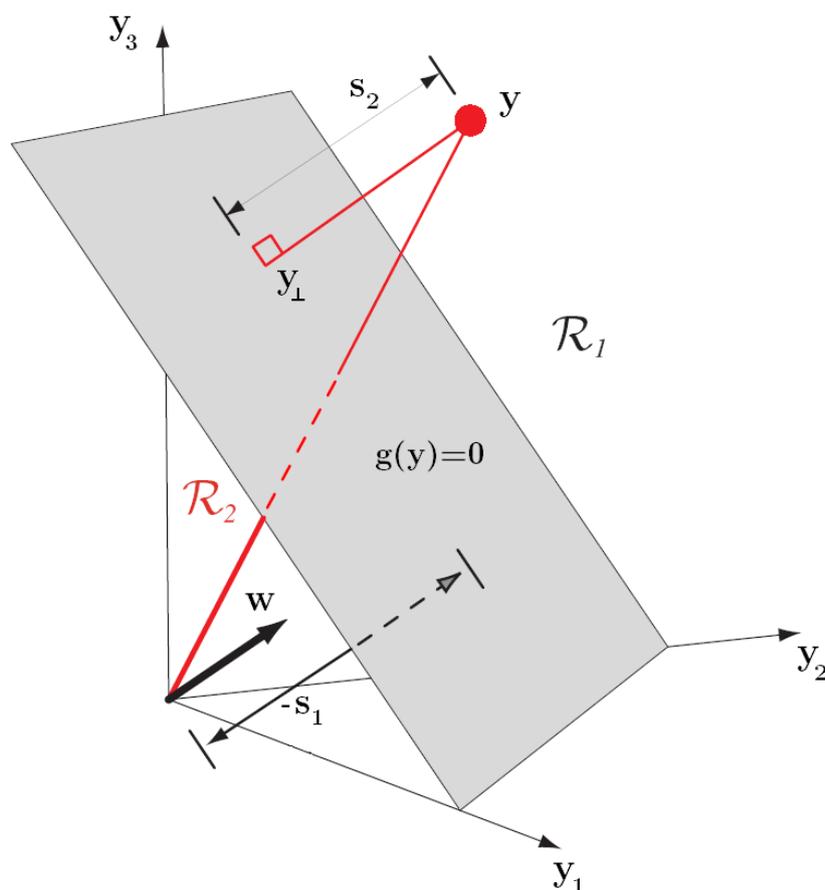


Abbildung 2.1: Die Diskriminante g im Zweiklassen-Problem im dreidimensionalen Raum ordnet den Merkmalsvektor \mathbf{y} einer Klasse zu.

größer als der Bias w_0 ist. \mathbf{y} wird der Klasse \mathcal{C}_2 zugeordnet, wenn $g(\mathbf{y}) \leq 0$. Die Entscheidungsfläche liegt bei $g(\mathbf{y}) = 0$.

- Gewichtsvektor \mathbf{w}

Der Gewichtsvektor \mathbf{w} gibt die Orientierung der Diskriminante an. Befinden sich \mathbf{y}_1 und \mathbf{y}_2 auf der Entscheidungsfläche, dann gilt

$$\mathbf{w}^T \mathbf{y}_1 + w_0 = \mathbf{w}^T \mathbf{y}_2 + w_0 \quad (2.8)$$

oder umgestellt

$$\mathbf{w}^T (\mathbf{y}_1 - \mathbf{y}_2) = 0. \quad (2.9)$$

Das bedeutet, dass der Vektor \mathbf{w} stets orthogonal auf einem Vektor steht, der sich auf der Entscheidungsfläche befindet. Da \mathbf{y} bei $g(\mathbf{y}) \geq 0$ der Klasse \mathcal{C}_1 zugeordnet wird, folgt, dass der Vektor \mathbf{w} in die Region \mathcal{R}_1 zeigt.

- Diskriminantenwert $g(\mathbf{y})$

Die Diskriminantenwerte $g(\mathbf{y})$ gibt den Abstand s_2 der \mathbf{y} -Werte zur Entscheidungsgrenze an. Nehmen wir einen beliebigen Merkmalsvektor \mathbf{y} an, dessen Projektion auf die Entscheidungsfläche \mathbf{y}_\perp ist, dann gilt für den Abstand des Punktes \mathbf{y} zu \mathbf{y}_\perp :

$$s_2 = \frac{\mathbf{w}^T \mathbf{y}}{|\mathbf{w}|} - \frac{-w_0}{|\mathbf{w}|} \quad (2.10)$$

$$= \frac{g(\mathbf{y})}{|\mathbf{w}|}. \quad (2.11)$$

Bei der Mehrklassen-Klassifikation gibt es verschiedene Methoden, den Merkmalsvektor \mathbf{y} einer Klasse \mathcal{C}_k zuzuordnen. Zum Beispiel können wir das Klassifikationsproblem auf K Zweiklassen-Probleme reduzieren. Das k -te Problem lösen wir mit einer Diskriminantenfunktion, die die Punkte den Regionen \mathcal{R}_k oder Nicht- \mathcal{R}_k zuordnet. Dieser Klassifikator wird *one-versus-the-rest* genannt. Im Gegensatz dazu benötigt der *one-versus-one* Klassifikator $\frac{K(K-1)}{2}$ Diskriminantenfunktionen, eine für jedes Klassenpaar.

Beide Varianten können zur Bildung von unbestimmten Regionen führen. Um die Entstehung solcher Regionen zu verhindern, verwenden wir einen Klassifikator mit K Diskriminanten:

$$g_k(\mathbf{y}) = \mathbf{w}_k^T \mathbf{y} + w_{k0}. \quad (2.12)$$

Wir ordnen den Merkmalsvektor \mathbf{y} der Klasse \mathcal{C}_k zu, falls $g_k(\mathbf{y}) > g_j(\mathbf{y})$ für alle $k \neq j$. Die Entscheidungsfläche liegt bei $g_k(\mathbf{y}) = g_j(\mathbf{y})$ und ist definiert durch:

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{y} + (w_{k0} - w_{j0}) = 0. \quad (2.13)$$

Die Entscheidungsregionen eines solchen Klassifikators sind konvex und einfach-verbunden, so dass alle Punkte auf einer Verbindungslinie zwischen zwei in der Region liegenden Punkten auch wieder in dieser liegen.

Wir können die Diskriminantenfunktionen in Matrix-Vektor-Schreibweise zusammenfassen und erhalten den $(K \times 1)$ -dimensionalen Vektor

$$g(\mathbf{y}) = \mathbf{W}^T \mathbf{y} \quad (2.14)$$

mit:

$$\begin{aligned} k\text{-te Spalte der Parametermatrix } \mathbf{W} &: \mathbf{w}_k = \begin{bmatrix} w_{k0} \\ \mathbf{w}_k \end{bmatrix}, \\ \text{Erweiterter Merkmalsvektor } \mathbf{y} &: \mathbf{y} = \begin{bmatrix} y_0 \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{y} \end{bmatrix}. \end{aligned}$$

Die Parametermatrix \mathbf{W} hat nun die Dimension $D \times K$ und der erweiterte Merkmalsvektor $D \times 1$ mit $D = M + 1$. Die Vektoren \mathbf{w}_k und \mathbf{y} sind homogene Größen, da sie frei skalierbar sind. Wir setzen $y_0 = 1$, es kann jedoch auch ein beliebig anderer Faktor gewählt werden. Im Falle von mehreren Merkmalsvektoren fassen wir diese zu der $(D \times N)$ -dimensionalen Matrix \mathbf{Y} zusammen, so dass sich die Diskriminantenwerte zu der $(K \times N)$ -dimensionalen Matrix \mathbf{G} zusammenfügen.

Wir haben die Möglichkeit die Diskriminantenfunktionen auf zwei Weisen anzupassen. Zunächst können wir die Merkmalsvektoren nichtlinear transformieren, indem wir zum Beispiel Basisfunktionen $\phi(\mathbf{Y}) = \Phi$, verwenden:

$$\mathbf{G}(\mathbf{Y}) = \mathbf{W}^T \Phi. \quad (2.15)$$

Das bedeutet, dass wir die Merkmalsvektoren in einen anderen Merkmalsraum transformieren, in dem sie möglicherweise besser trennbar sind. In dieser Arbeit transformieren wir die Merkmalsvektoren nicht, allerdings gelten die nachfolgenden Ausführungen für transformierte Merkmalsvektoren im gleichem Maße. Im Folgenden nehmen wir an, dass

$$\Phi = \mathbf{Y}. \quad (2.16)$$

Wir können zusätzlich die linearen Funktionen von \mathbf{W} mit einer nichtlinearen Funktion $f(\cdot)$ transformieren:

$$\mathbf{G}(\mathbf{Y}) = f(\mathbf{W}^T \Phi). \quad (2.17)$$

Die Funktion $f(\cdot)$ wird *Aktivierungsfunktion* genannt. Obwohl diese nichtlinear ist, sind die Diskriminanten $f^{-1}(G(\mathbf{Y})) = \mathbf{W}^T \Phi$ lineare Funktionen von Φ . Auf Grund der Analogie zu den nicht transformierten Diskriminanten aus Formel (2.14) werden diese Modelle als *generalisierte lineare Modelle* bezeichnet.

Die Bedeutung der nichtlinearen Transformation diskutieren wir im folgenden Abschnitt näher.

2.3 Probabilistische Modelle

Bisher können wir mit Hilfe von linearen Diskriminanten die Merkmalsvektoren \mathbf{Y} den vorgegebenen Klassen \mathcal{C} zuordnen. Wir wissen allerdings noch nichts über die Wahrscheinlichkeit für eine Klasse bei gegebenen Merkmalsvektoren. Damit wir eine probabilistische Aussage über die Klassenzugehörigkeit machen können, transformieren wir die lineare Funktion von \mathbf{W} mit einer nichtlinearen Funktion $f(\cdot)$, wie in Formel (2.17) angegeben:

$$G(\mathbf{Y}) = f(\mathbf{W}^T \Phi).$$

Die transformierte Diskriminante muss für den probabilistischen Charakter zwei Eigenschaften aufweisen: Alle Werte müssen im Intervall $[0,1]$ liegen und die Spalten müssen sich zu 1 summieren. Die Transformation, die dies ermöglicht ist die *Softmax-Transformation*.

Im Folgenden betrachten wir diese näher, um zu verstehen, wie der probabilistische Charakter entsteht und welche Bedeutung die Ergebnisse für diese Arbeit haben.

Wir betrachten die Softmax-Transformation zunächst nach Bishop (2006, Kapitel 4, Seite 205 ff.) anhand des Zweiklassen-Problems eines generativen Modells mit einem Merkmalsvektor Φ_n . Das Prinzip des generativen Modells ist die Modellierung der klassenbedingten Wahrscheinlichkeiten $p(\Phi_n|\mathcal{C})$ und der a priori Wahrscheinlichkeiten $p(\mathcal{C})$, um die a posteriori Wahrscheinlichkeiten $p(\mathcal{C}|\Phi_n)$ mit Hilfe des Bayestheorems berechnen zu können. Die a posteriori Wahrscheinlichkeit der Klasse \mathcal{C}_1 können wir wie folgt schreiben:

$$p(\mathcal{C}_1|\Phi_n) = \frac{p(\Phi_n|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\Phi_n|\mathcal{C}_1)p(\mathcal{C}_1) + p(\Phi_n|\mathcal{C}_2)p(\mathcal{C}_2)}. \quad (2.18)$$

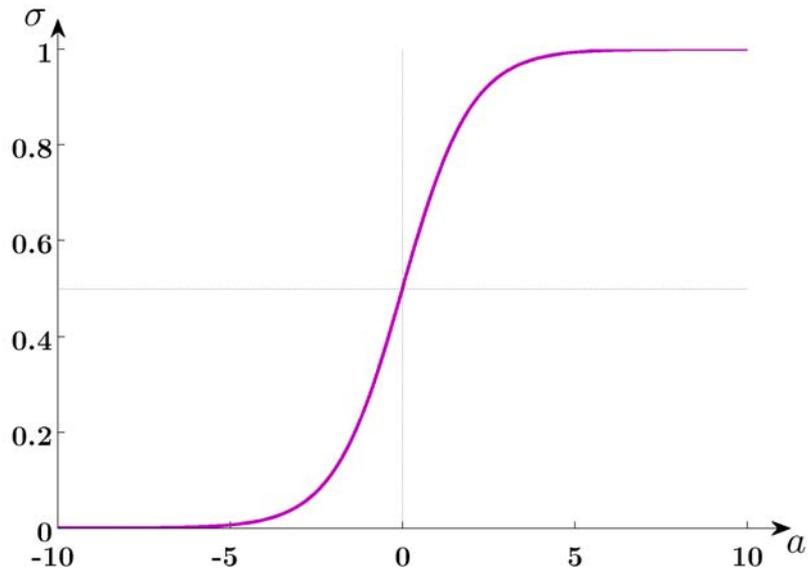


Abbildung 2.2: Die Sigmoid Funktion $\sigma(a)$ nach Formel (2.20) transformiert alle eingehenden Werte a in einen Bereich $(0,1)$. Der Anstieg der Sigmoid Funktion folgt der logistischen Differentialgleichung $\sigma' = \sigma(1 - \sigma)$.

Nun substituieren wir

$$a_n = \ln \frac{p(\Phi_n | \mathcal{C}_1) p(\mathcal{C}_1)}{p(\Phi_n | \mathcal{C}_2) p(\mathcal{C}_2)} \quad (2.19)$$

und erhalten folgende Funktion:

$$p(\mathcal{C}_1 | \Phi_n) = \frac{1}{1 + \exp(-a_n)} = \sigma(a_n). \quad (2.20)$$

Diese Funktion wird *Sigmoid-Funktion* genannt, wobei „Sigmoid“ S-gekrümmt bedeutet. Die Sigmoid-Funktion, wie in Abbildung 2.2 dargestellt, gehört zu den *Squashing-Funktionen*, da sie alle realen Funktion in ein finites Intervall transformiert.

Für das Mehrklassen-Problem gilt für die a posteriori Wahrscheinlichkeit für die Klasse \mathcal{C}_k äquivalent zu Formel (2.18):

$$p(\mathcal{C}_k | \Phi_n) = \frac{p(\Phi_n | \mathcal{C}_k) p(\mathcal{C}_k)}{\sum_j p(\Phi_n | \mathcal{C}_j) p(\mathcal{C}_j)}. \quad (2.21)$$

Mit der Aktivierung

$$a_{kn} = \ln p(\Phi_n | \mathcal{C}_k) p(\mathcal{C}_k) \quad (2.22)$$

ergibt sich für (2.21):

$$p(\mathcal{C}_k | \Phi_n) = \frac{\exp(a_{kn})}{\sum_j \exp(a_{jn})}. \quad (2.23)$$

Erinnern wir uns an das Problem am Anfang des Abschnittes. Wir wollen Diskriminanten in ein Intervall $[0,1]$ transformieren und die Spalten sollen sich zu 1 summieren. Die a posteriori Wahrscheinlichkeiten besitzen genau diese Eigenschaften, sodass wir unser lineares diskriminatives Modell definieren können:

$$p(\mathcal{C}_k | \Phi_n) = g_{kn} = \sigma(a_{kn}) \quad (2.24)$$

mit den Aktivierungen:

$$a_{kn} = \mathbf{w}_k^T \Phi_n. \quad (2.25)$$

Damit erhalten wir den gewünschten Wahrscheinlichkeitscharakter. Die Normalisierung hat zur Folge, dass ein $\exp(a_k)$ linear abhängig von den anderen wird, da $\exp(a_k) = 1 - \sum_{j \setminus k} \exp(a_j)$ gilt. Daher sind nicht alle Parameter bestimmbar, sondern nur $(D-1) \cdot K$.

Wir erhalten also ein Modell, welches durch die Softmax-Transformation der linearen Funktionen der Merkmalsvektoren die a posteriori Wahrscheinlichkeiten der Merkmalsvektoren erhält, ohne die Bayesformel anzuwenden. Der Begriff Softmax bedeutet nach Bishop (1995, Kapitel 6, Seite 238), dass die Aktivierungsfunktionen eine abgeschwächte Version des *Maximums-Aktivierungsmodells* sind, bei dem die Einträge mit dem größten Input das Ergebnis +1 und die anderen Inputs das Ergebnis 0 erhalten.

Dieses Modell ist als *Logistische Regression* bekannt, da die Sigmoid-Funktion als Lösung der logistischen Differentialgleichung

$$\sigma' = \sigma(1 - \sigma) \quad (2.26)$$

hervorgeht.

Die Ergebnisse des Modells dienen in dieser Arbeit als Eingabewerte für das Markov Random Field mit dem wir uns in Kapitel 4 beschäftigen. Wir verwenden sie dort als klassenbedingte Wahrscheinlichkeiten.

Nachdem wir das Modell näher betrachtet haben, gilt es nun die Parameter für die Hyperflächen zu lernen, mit denen die Merkmalsvektoren getrennt werden.

2.4 Logistische Regression

Im letzten Abschnitt haben wir gezeigt, dass die Aktivierung a_{kn} durch Formel (2.25) gegeben ist. Die a posteriori Wahrscheinlichkeiten ergeben sich durch eine Softmax-Transformation:

$$p(\mathcal{C}_k|\boldsymbol{\phi}_n) = g_{kn} = \sigma(a_{kn})$$

Der folgende Abschnitt befasst sich mit dem Lernen der Parameter \mathbf{W} des linearen probabilistischen diskriminativen Modells. Dazu gehen wir zunächst auf die Schätzung der Parameter ein. Das Lernverfahren stellt ein konvexes Optimierungsproblem dar. Auf Grund dessen nennt dieser Abschnitt Optimierungsverfahren für die Parameter und geht insbesondere auf das Gradientenabstiegsverfahren ein, welches wir in dieser Arbeit verwenden.

Die Schätzung der Parameter \mathbf{W} erfolgt durch die *Maximum Likelihood Schätzung*. Die Likelihoodfunktion bilden wir mit den Zielvektoren $\{\mathbf{t}_n\}$ für die Merkmalsvektoren $\{\boldsymbol{\phi}_n\}$. Der zu einer Klasse \mathcal{C}_k gehörende Zielvektor ist ein aus Nullen bestehender Vektor, der an der k -ten Stelle den Eintrag 1 hat. Alle Zielvektoren aneinandergereiht ergeben die $(K \times N)$ -Zielmatrix \mathbf{T} . Die Likelihoodfunktion ist nach Bishop (2006, Kapitel 4, Seite 209) gegeben durch:

$$p(\mathbf{T}|\mathbf{W}) = \prod_{n=1}^N \prod_{k=1}^K p(\mathcal{C}_k|\boldsymbol{\phi}_n)^{t_{kn}} = \prod_{n=1}^N \prod_{k=1}^K g_{kn}^{t_{kn}}. \quad (2.27)$$

Sie gibt die Wahrscheinlichkeit der Zielmatrix \mathbf{T} bei gegebenen Parametern \mathbf{W} an. Wir erhalten die Likelihoodfunktion durch Multiplikation aller a posteriori Wahrscheinlichkeiten der den Merkmalsvektoren zugeordneten Klassen \mathcal{C}_k .

Der negative Logarithmus ist die so genannte *Kreuzentropie*, die als Fehlerfunktion verwendet wird:

$$F(\mathbf{W}) = -\ln p(\mathbf{T}|\mathbf{W}) = -\sum_{n=1}^N \sum_{k=1}^K t_{kn} \ln g_{kn}. \quad (2.28)$$

Bei diesen Funktionen zeigt sich deutlich der Vorteil in der Benutzung der Zielvektoren statt der Label. Funktion 2.28 multipliziert die Wahrscheinlichkeit der Klassen, die wir aus dem transformierten Modell mit Formel (2.23) erhalten, mit den uns bekannten Zielvariablen. Alle Wahrscheinlichkeiten für Klassen mit dem Wert 1 in der Zielmatrix fließen komplett in die Fehlerfunktion ein und Wahrscheinlichkeiten mit dem Wert 0 in der Zielmatrix werden ignoriert. Die Fehlerfunktion wird maximal, wenn die Wahrscheinlichkeiten für die „richtigen“ Klassen maximal werden. Sie stellt eine Hyperfläche im $(D \cdot K)$ -dimensionalen Raum dar, wobei jede Dimension für einen Parameter steht. Für diese Fläche wollen wir das Minimum finden, da an dieser Stelle die Parameter für die Diskriminanten in der Art gewählt werden, dass sie die Merkmale optimal trennen.

Das Minimum erhalten wir durch Nullsetzen der ersten Ableitung. Der Gradient der Fehlerfunktion bezüglich eines Parametervektors \mathbf{w}_j ist nach Anhang 1.2

$$\nabla_{\mathbf{w}_j} F(\mathbf{W}) = \sum_{n=1}^N (g_{jn} - t_{jn}) \boldsymbol{\Phi}_n, \quad (2.29)$$

wobei

$$\nabla_{\mathbf{w}_j} F(\mathbf{W}) = \frac{\partial F(\mathbf{W})}{\partial \mathbf{w}_j}. \quad (2.30)$$

Diese Fehlerfunktion formt im $(D \cdot K)$ -dimensionalen Raum eine so genannte Fehlerfläche $F(\mathbf{W})$. Die Minimierung kann nicht analytisch gelöst werden, sondern stellt ein konvexes Optimierungsproblem dar, bei dem das Minimum der Fehlerfläche iterativ gesucht wird:

$$\widehat{\mathbf{W}} = \arg \min_{\mathbf{W}} F(\mathbf{W}|\mathbf{T}). \quad (2.31)$$

Abbildung 2.3 zeigt einen Querschnitt durch die Fehlerfläche der Trainingsbilder des Sowerby Datensatzes. Die konvexe Eigenschaft wird hier besonders deutlich. Die Optimierung kann mit dem *Newton-Raphson* Abstiegsverfahren (Bishop, 2006, Kapitel 4, Seite 207 ff.) oder mit dem *Gradientenabstiegsverfahren* (Boyd und Vandenberghe, 2004, Kapitel 9, Seite 464) erfolgen. Beides sind iterative Verfahren, die ein Update entgegen-

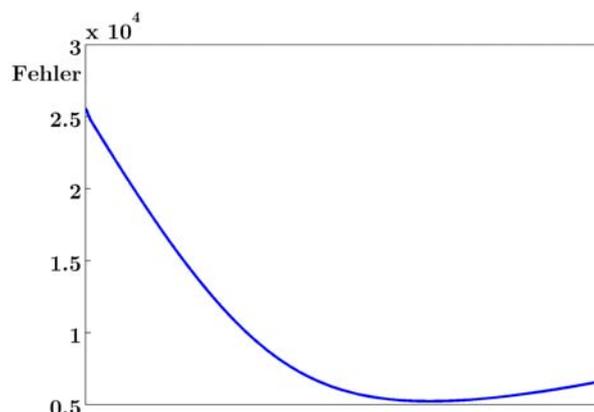


Abbildung 2.3: Die konvexe Eigenschaft der Fehlerfläche wird besonders deutlich, wenn wir einen Querschnitt von dieser erstellen. Die Ordinatenachse zeigt die Größe des Fehlers.

gesetzt des Gradienten an die Parameter anbringen. Der Unterschied zwischen beiden Verfahren ist die Wahl der Schrittweite s :

$$\mathbf{W}^{(new)} = \mathbf{W}^{(old)} - s \nabla F(\mathbf{W}) \quad \text{Gradientenabstiegsverfahren,} \quad (2.32)$$

$$\mathbf{W}^{(new)} = \mathbf{W}^{(old)} - \mathbf{H}^+ \nabla F(\mathbf{W}) \quad \text{Newton-Raphson Abstiegsverfahren.} \quad (2.33)$$

Das Newton-Raphson Abstiegsverfahren verwendet eine dynamische Schrittweite, die durch die Pseudoinverse der Hessematrix \mathbf{H}^+ festgelegt wird. Die Hessematrix der Fehlerfunktion lautet nach Bishop (2006, Kapitel 4, Seite 210)

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} F(\mathbf{W}) = \sum_{n=1}^N y_{kn} (I_{kj} - y_{nj}) \quad (2.34)$$

mit I_{kj} als (k,j) -ten Eintrag der Einheitsmatrix. Wir verwenden die Pseudoinverse, da nur $(D-1) \cdot K$ Parameter bestimmt werden können. Die Matrix weist somit einen Rangdefekt auf und ist folglich singulär.

Die Verwendung der Hessematrix ist sehr nützlich, da sie die Krümmung der Fläche berücksichtigt. Nehmen wir beispielsweise eine Parabel an, so ist das Minimum bei einer kleinen Krümmung noch weit entfernt und wir können ein großes Update anbringen. Nähern wir uns dem Minimum, wird die Krümmung größer und das Verfahren bringt ein kleines Update an die Parameter an. Das Newton-Verfahren neigt somit weniger dazu, das Minimum zu überspringen. In dieser Arbeit verfolgen wir das Newton-Verfahren den-

noch nicht weiter, da wir die Auswirkungen der Wahl der Schrittweite näher betrachten wollen.

Die Wahl der Schrittweite ist mit Bedacht zu wählen, da bei einer ungünstigen Wahl das Minimum schlecht oder erst nach vielen Iterationen gefunden werden kann. Die folgenden Erläuterungen zeigen die hauptsächlichen Probleme, die bei der Lösung eines konvexen Optimierungsproblems mit dem Gradientenabstiegsverfahren und schlechter Wahl der Schrittweite auftreten können.

Abbildung 2.4 zeigt das Problem der Oszillation, welches auftreten kann, wenn die Schrittweite zu groß gewählt wird. Dargestellt ist ein Querschnitt der Fehlerfläche, so dass die Abszissenachse die Parameterwerte \mathbf{W} und die Ordinatenachse den Wert der Fehlerfunktion F angibt. Die Länge der schwarzen Pfeile gibt den Wert des Produktes der Schrittweite mit der Ableitung an. Der Schritt ist umso weiter, je größer die Ableitung ist.

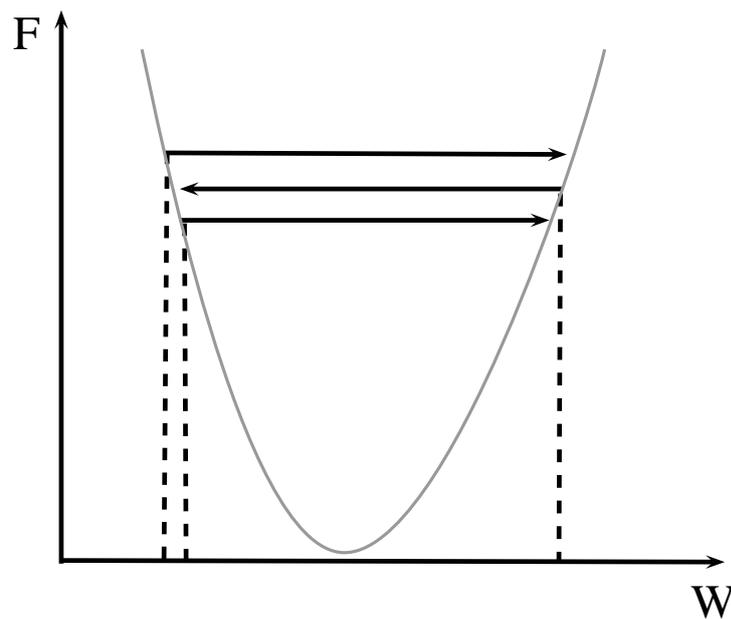


Abbildung 2.4: Die Abbildung zeigt einen Querschnitt der Fehlerfläche bei einem konvexen Optimierungsproblem und verdeutlicht das Problem bei einer zu großen Wahl der Schrittweite. Die Abszissenachse gibt die Parameterwerte \mathbf{W} und die Ordinatenachse den Wert der Fehlerfunktion F an. Die Länge der schwarzen Pfeile gibt den Wert des Produktes der Schrittweite mit der Ableitung an.

Durch eine zu groß gewählte Schrittweite wird das Minimum in jeder Iteration übersprungen. Im schlechtesten Fall wird das Minimum nie gefunden, wenn das Verfahren zwischen zwei oder mehreren Standpunkten hin- und herspringt und nicht tiefer in die Fehlerfläche vordringt. Dies tritt häufig auf, wenn der Gradient der Fehlerfläche sehr groß ist. Ein weiteres Problem stellt eine zu klein gewählte Schrittweite dar. In diesem Fall wird das Minimum gefunden, die Laufzeit steigt jedoch enorm. Es gilt somit eine optimale Wahl der Schrittweite zu wählen.

Unter Beachtung der genannten Probleme kann das Gradientenabstiegsverfahren angepasst werden, um den gewünschten Lernerfolg zu erzielen. Das Verfahren besitzt zwei Stellschrauben, die das Ergebnis maßgeblich beeinflussen: die Wahl der initialen Parameter und die Wahl der Schrittweite.

Die Wahl der initialen Parameter legt fest, wo das Gradientenabstiegsverfahren startet und somit wie weit das Minimum entfernt liegt. Außerdem können wir die Art der Initialisierung wählen, das heißt wie unterschiedlich die Werte untereinander sind (ob sie zum Beispiel alle nahe bei Null liegen).

Für die Wahl der Schrittweite gibt es keine allgemein gültige gute Wahl, sodass wir diese entsprechend des Sachverhaltes anpassen müssen. Eine bewährte Technik ist die zunächst willkürliche Wahl einer Schrittweite und anschließendes sukzessives Anpassen dieser Schrittweite. Eine Erhöhung dieser bewirkt einerseits, dass das Minimum schnell erreicht wird, es kann jedoch auch bewirken, dass es übersprungen wird und das Verfahren oszilliert. Eine Reduktion der Schrittweite führt dazu, dass das Minimum nicht so schnell übersprungen wird und das Verfahren nicht so schnell oszilliert. Es kann jedoch auch bedeuten, dass die Laufzeit enorm steigt.

Mit Hilfe dieses Kapitels können wir nun ein lineares probabilistisches diskriminatives Modell durch Schätzung der Parameter der Diskriminanten lernen, indem wir ein konvexes Optimierungsproblem zum Beispiel mit dem Gradientenabstiegsverfahren lösen. Wir erhalten mit den beschriebenen Methoden die a posteriori Wahrscheinlichkeiten, die wir in Kapitel 4 für die Segmentierung mit den Markov Random Fields benötigen. Mit den bisher genannten Verfahren können wir bereits eine Klassifikation der Bilder durchführen, indem wir das Argument der maximalen a posteriori Wahrscheinlichkeit wählen. Die Markov Random Fields bieten allerdings den entscheidenden Vorteil, die Homogenität eines Bildes zu fordern.

Das folgende Kapitel setzt sich zunächst mit der Wahl der Merkmale auseinander, die die Form der Fehlerfläche beeinflussen. Ihre Wahl ist neben dem Modell ausschlaggebend für den Erfolg der Klassifikation.

3 Merkmalsextraktion

Im letzten Kapitel sind wir von beliebigen Merkmalsvektoren ausgegangen. Dieses Kapitel betrachtet nun die Wahl der Merkmale, sodass diese möglichst gut im Merkmalsraum getrennt werden können. Ziel ist es, Merkmale für jede Klasse in der Art zu wählen, dass sie sich durch ihre Merkmale von allen anderen Klassen unterscheidet. Dadurch können wir Diskriminanten finden, die die Kreuzentropie minimieren.

Die in dieser Arbeit verwendeten Merkmale beinhalten Informationen über die Farbe, die Position, die Größe und die Form von Regionen, die Textur und den Kontext des Bildes. Sie sind in Tabelle 3.1 mit ihrer Dimension aufgelistet und werden anschließend näher betrachtet.

Wir haben die Möglichkeit, die Merkmale ohne Umgebung, aus einer Rechteckumgebung oder aus einem Superpixel zu extrahieren. Im Folgenden betrachten wir jedoch nur zwei Gruppen, die Merkmale aus einer Rechteckumgebung und die Superpixelmerkmale. Die Merkmale ohne Umgebung ordnen wir denen aus der Rechteckumgebung zu. Dies ist möglich, da die Texturmerkmale schon durch die Faltung aus einer Rechteckumgebung extrahiert werden und eine weitere Mittelung überflüssig erscheint. Außerdem sind die pixelweisen Positionsmerkmale mit den gemittelten aus einer Rechteckumgebung identisch und die Mittelung der Kontextmerkmale ist nicht sinnvoll. Darauf gehen wir in Kapitel 3.2 näher ein.

Die Merkmale in den Superpixeln erhalten wir entweder durch Mittelung oder durch Histogrammbildung über die darin enthaltenen Pixelmerkmale. Alle Pixel in einem Superpixel erhalten die gleichen Merkmale. Somit ergeben sich für die Merkmale aus einer Rechteckumgebung und aus einem Superpixel die in Tabelle 3.2 aufgeführten Dimensionen.

Wir werden im Folgenden zunächst auf die Superpixel eingehen und anschließend die Merkmale im Einzelnen erläutern.

Merkmal	Dimension		
	pixelweise	Rechteckumgebung	Superpixel
Farbe			
Mittelwert RGB		3	3
Mittelwert Lab		3	3
Mittelwert HSV		3	3
Mittelwert Grau		1	1
Standardabweichung RGB		3	3
Standardabweichung Lab		3	3
Standardabweichung HSV		3	3
Standardabweichung Grau		1	1
Histogramm RGB (5 Bins)			15
Histogramm Lab (5 Bins)			15
Histogramm HSV (5 Bins)			15
Histogramm Grau (5 Bins)			5
Position			
Pixelkoordinaten	2		
Mittelwert Pixelkoordinaten			2
Größe und Form			
Fläche			1
Exzentrizität			1
Textur			
Leung Malik	96		
Filterantworten (HS Kanal)			
Mittelwert Leung Malik			96
Filterantworten (HS Kanal)			
Kontextmerkmale			
Häufigste Farbwerte	9		9

Tabelle 3.1: Die verwendeten Merkmale beinhalten Informationen über die Farbe, die Position, die Größe und die Form von Regionen, die Textur und den Kontext des Bildes. Die Dimension gibt die Länge der jeweiligen Merkmalsvektoren an. Merkmale ohne Eintrag verwenden wir nicht.

Umgebung	Merkmale	Dimension
Rechteck	alle	127
Superpixel	ohne Histogramme	129
Superpixel	ohne Mittelwerte	169
Superpixel	alle	179

Tabelle 3.2: Wir können die Merkmale unterteilen in: Merkmale aus einer Rechteckumgebung und Merkmale aus einem Superpixel und letztere wiederum in die Extraktion durch Mittelung oder Histogrammbildung.

3.1 Superpixel

Für die meisten Merkmale verwenden wir zusätzlich zur Information des Pixels y_n die Information ihrer Nachbarn. Der Grund stützt sich auf die Tatsache, dass in der Natur eher homogene Regionen vorkommen als solche mit vielen ausgeprägten Merkmalsübergängen. Abrupte Übergänge treten bei Grenzen von Regionen auf oder durch Rauschen. Die Verwendung einer rechteckigen Umgebung eines Pixels hat den Vorteil, dass das Rauschen unterdrückt wird, hat jedoch den Nachteil, dass die Informationen an Kanten verwischen. Am Beispiel eines leicht verrauschten Grauwertbildes bedeutet dies Folgendes: Ist ein Pixel ein Ausreißer und bilden wir über ihn und seine Umgebung das Grauwertmittel, so fällt dieser je nach Größe der Umgebung mehr oder weniger gering ins Gewicht. Bilden wir allerdings das Mittel über die Umgebung eines Pixel, welches sich auf dem Übergang von einer hellen zu einer dunklen Region befindet, dann erhalten wir im Mittel ein mittelgraues Merkmal. Dies spiegelt weder die eine noch die andere Region wider. Um dieses Problem zu verhindern, verwenden wir so genannte *Superpixel*, die das Bild in homogene Regionen teilen. Das Bild wird vorsegmentiert und alle Pixel in den Superpixeln erhalten dieselben Merkmale.

Für die Erstellung der Superpixel verwenden wir den Normalized Cut (NCut)-Algorithmus nach Shi und Malik (2000)³. Abbildung 3.1a-3.1b zeigt ein Bild aus dem Sowerby Datensatz, die gefundenen Superpixelgrenzen und die farbig dargestellten Superpixel.

Für die Erstellung der Superpixel konvertieren wir zunächst das Bild in ein Grauwertbild und wenden anschließend den NCut-Algorithmus an. Dieser berechnet die Ähnlichkeit

³Der Quellcode kann unter <http://www.cis.upenn.edu/~jshi/software/> heruntergeladen werden.

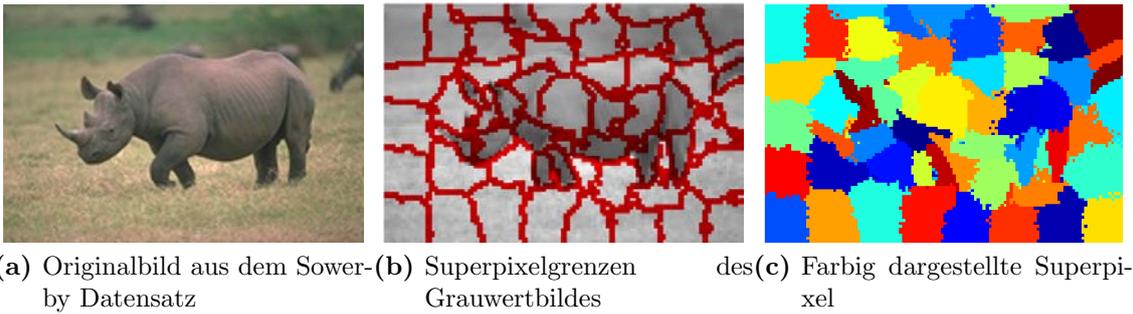


Abbildung 3.1: Die Erstellung der Superpixel eines Beispielbilds aus dem Sowerby Datensatz erfolgt mit dem NCut-Code nach Shi und Malik (2000).

zwischen den Grauwerten und teilt das Bild sukzessiv in immer kleinere, möglichst homogene Regionen, bis die gewünschte Superpixelanzahl erreicht ist.

Die Ähnlichkeiten aller Pixel sind in einer Matrix repräsentiert. Da der Speicher, den die Entwicklungsumgebung zur Verfügung stellt, begrenzt ist, verkleinern wir die Bilder auf eine Pixelanzahl von $N_P = 10000$. Dazu berechnen wir einen Skalierungsfaktor

$$s = \sqrt{\frac{N_P}{N_R \cdot N_C}}, \quad (3.1)$$

der die maximale Pixelanzahl ausnutzt und zusätzlich die Proportion von Zeilenanzahl N_R und Spaltenanzahl N_C des Bildes erhält. Wir multiplizieren ihn mit der tatsächlichen Bildgröße um die Maße für das verkleinerte Bild zu erhalten.

Die Anzahl der Superpixel in einem Bild richtet sich nach der Bildgröße. Die Größe soll in etwa der Hälfte einer (25×25) -Rechteckumgebung entsprechen, sodass wir die Anzahl N_S wie folgt berechnen können:

$$N_S = \frac{1}{2} \cdot \frac{N_R \cdot N_C}{25^2 \cdot s^2}. \quad (3.2)$$

Im Zuge der Implementation müssen wir beachten, dass mit diesem Verfahren Superpixel der Größe 0 Pixel auftreten können.

Mit Hilfe der Superpixel können wir eine regionenbasierte Segmentierung durchführen, da anders als bei der Rechteckumgebung die Pixel in einem Superpixel die gleichen

Merkmale erhalten. Die Merkmale können wir über Mittelung oder Histogrammbildung extrahieren.

3.2 Merkmale

Die Wahl der Merkmale stellt eine schwierige Entscheidung dar, da die Eignung je nach Datensatz und Aufgabenstellung stark variieren kann. Im Grunde genommen gilt, dass wir nie genug Merkmale extrahieren können, denn je mehr Merkmale eine Klasse beschreiben, desto besser ist sie von anderen Klassen unterscheidbar. Die Anzahl und Art der Merkmale ist jedoch meist durch den Speicher und die Laufzeiten begrenzt. Diese Arbeit verwendet Merkmale aus den in der Literatur meist verwendeten Gruppen: Größe und Form von Regionen, Position, Farbe, Textur und Kontextmerkmale (siehe He und Zemel (2008), Shotton u. a. (2006), He u. a. (2004), Malik u. a. (2001), Leung und Malik (2001) und Durupinar (2005)).

Die folgenden Abschnitte erläutern die verwendeten Merkmale aus diesen Gruppen näher.

3.2.1 Farbmerkmale

Die Farbmerkmale sind sehr intuitiv, da sie die Originaldaten repräsentieren. Je nach Wahl des Farbraumes verwenden wir andere Grundgrößen, die Informationen über die Farbe enthalten. Der folgende Abschnitt erläutert diejenigen Farbräume, die wir in dieser Arbeit verwenden.

RGB-Farbraum

Der RGB-Farbraum besteht aus den drei Primärfarben Rot R , Grün G und Blau B , die jeweils eine Achse im Bereich $[0,255]$ in diesem Raum darstellen. Die Bilder unserer Datensätze sind RGB-Bilder. Für die Umrechnungen in andere Farbräume skalieren wir das Intervall auf $[0,1]$.

HSV-Farbraum

Der HSV-Farbraum enthält die drei Grundgrößen Farbton H , Sättigung S und Helligkeit V . Anders als im RGB-Farbraum besitzt der HSV-Raum nur einen Farbkanal. Dieser

ist im Gegensatz zu den RGB-Farben unabhängig von der Beleuchtung. Wir können sie aus den RGB-Werten berechnen⁴ und wie folgt definieren:

- Farbton (engl. Hue)

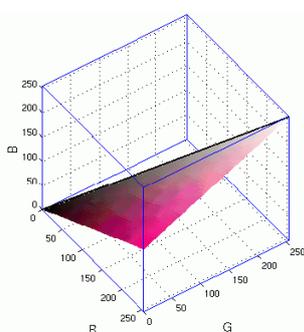
Der Farbton wird durch einen Farbwinkel im Bereich $[0^\circ, 360^\circ]$ repräsentiert, wobei 0° Rot, 120° Grün und 240° Blau entsprechen.



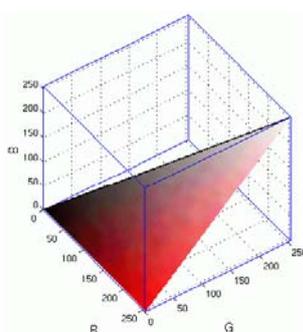
Wir können den Farbton wie folgt berechnen:

$$H = \begin{cases} \left(0 + \frac{G-B}{\max(R,G,B)-\min(R,G,B)} \right) \cdot 60, & \text{falls } R = \max(R,G,B) \\ \left(2 + \frac{B-R}{\max(R,G,B)-\min(R,G,B)} \right) \cdot 60, & \text{falls } G = \max(R,G,B) \\ \left(4 + \frac{R-G}{\max(R,G,B)-\min(R,G,B)} \right) \cdot 60, & \text{falls } B = \max(R,G,B) \end{cases} \quad (3.3)$$

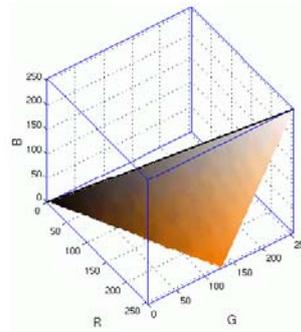
Die Beziehung des Farbtons zum RGB-Farbraum ist in Abbildung 3.2a-3.2c dargestellt.



(a) Farbton = 330°



(b) Farbton = 0°



(c) Farbton = 30°

Abbildung 3.2: Eine Ebene mit gleichem Farbton im RGB-Würfel liegt in einem Winkel zur neutralen Achse, durch den der Farbton beschrieben ist. (a) zeigt Pink mit $H = 330^\circ$, (b) zeigt Rot mit $H = 0^\circ$, was gleichzeitig auch $H = 360^\circ$ ist und (c) zeigt Orange mit $H = 30^\circ$. Auf der Ebene ist der Farbton konstant, die Sättigung und die Helligkeit variieren jedoch.

⁴Umrechnung nach: <http://www.farbtipps.de/farbsysteme/hsv/>

- Sättigung (engl. Saturation)

Die Sättigung gibt die Intensität des Farbtons durch einen Wert im Bereich $[0,1]$ an, also wie rein ein Farbton im Vergleich zu Grau ist. Der Wert 0 repräsentiert einen Grauton, 0,5 eine ungesättigte Farbe und 1 eine gesättigte Farbe. Wir können ihn berechnen, indem wir den minimalen RGB-Wert durch den maximalen Wert teilen und von 1 abziehen:

$$S = 1 - \frac{\min(R,G,B)}{\max(R,G,B)}. \quad (3.4)$$

Die Beziehung der Sättigung zum RGB-Würfel ist in Abbildung 3.3a-(b) dargestellt.

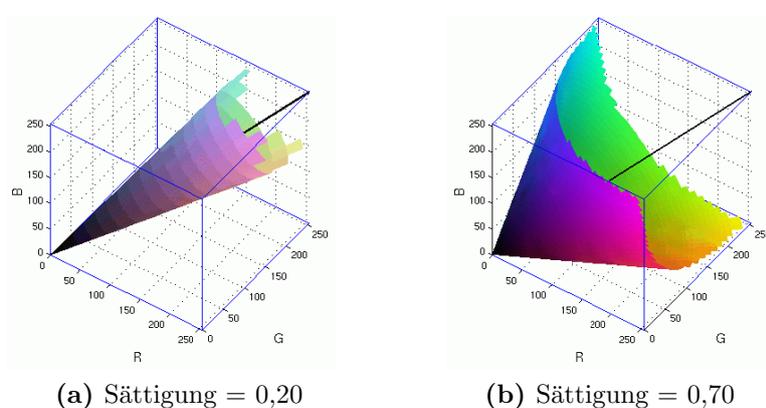


Abbildung 3.3: Die Flächen mit gleicher Sättigung im RGB-Raum bilden Kegel. Je geringer die Sättigung, desto spitzer ist der Kegel. (a) zeigt eine geringe und (b) eine hohe Sättigung. Je spitzer der Kegel, desto blasser werden die Farben, weil er näher an der neutralen Achse mit allen Grautönen liegt (Verbindung von $(0,0,0)$ und $(255,255,255)$).

- Helligkeit (engl. Value)

Sie gibt die Dunkelstufe in einem Bereich $[0,1]$ an, wobei 0 Schwarz und 1 volle Helligkeit bedeuten. Je dunkler der Farbton, desto weniger Sättigung kann eine Farbe erreichen. Aus diesem Grund ergibt sich die typische Zylinderform des HSV-Farbraums. Die Helligkeit können wir wie folgt berechnen:

$$V = \max(R,G,B). \quad (3.5)$$

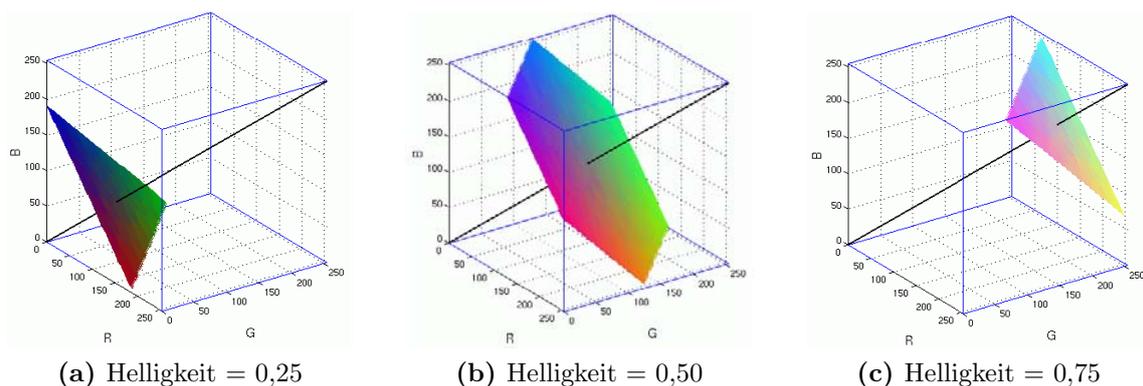


Abbildung 3.4: Farben mit gleicher Helligkeit beschreiben Ebenen im HSV-Farbraum. Eine Ebene mit gleicher Helligkeit und $R + G + B = const.$ liegt im RGB-Würfel senkrecht zur neutralen Achse. Diese Ebenen verdeutlichen das formale Prinzip, entsprechen jedoch der menschlichen Wahrnehmung nicht sehr gut. Wir nutzen deshalb die Berechnung nach Formel 3.5.

Lab-Farbraum

Der Lab-Farbraum ist auf der Gegenfarbentheorie aufgebaut. Diese besagt, dass es keine Mischfarben wie „gelbliches Blau“ oder „rötliches Grün“ gibt, also dass sich die Farben Gelb und Blau sowie die Farben Rot und Grün gegenseitig ausschließen. Des Weiteren gelten Schwarz und Weiß als Gegenfarben. Anschaulich kann man sich diesen Effekt folgendermaßen vorstellen: Betrachtet man einen gelbes Objekt auf schwarzem Untergrund und anschließend einen weißen Hintergrund, entsteht ein blaues Nachbild. Dies tritt bei allen zusammengehörenden Gegenfarben ein. Abbildung 3.5 zeigt die drei Gruppen der Gegenfarben.

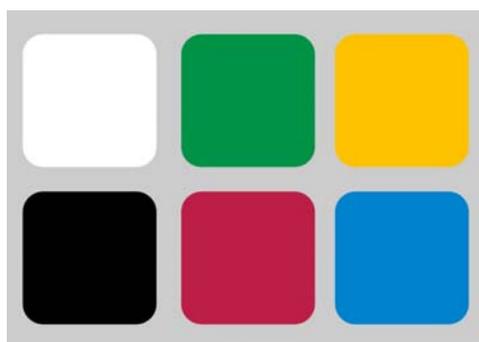


Abbildung 3.5: Die drei Gruppen der Gegenfarben sind Weiß-Schwarz, Grün-Rot und Gelb-Blau.

Der Lab-Farbraum enthält folgende Komponenten:

- L (Lumineszenz)
Sie gibt die Helligkeit im Bereich $[0,1]$ an, also die Information zu den Gegenfarben Schwarz-Weiß. Schwarz hat den Wert 0 und Weiß den Wert 1.
- a (Rot-Grün)
Die Position auf der Rot-Grün Achse im Bereich $[-150,100]$ wird im Lab-Farbraum mit a bezeichnet.
- b (Gelb-Blau)
Die Position auf der Gelb-Blau Achse im Bereich $[-100,150]$ wird im Lab-Farbraum mit b bezeichnet.

Die Umrechnung⁵ der RGB-Werte in die Lab-Werte erfolgt über die XYZ-Werte, die so genannte virtuelle RGB-Werte sind:

$$X = 2,36460R - 0,51515G + 0,00520B, \quad (3.6)$$

$$Y = -0,89653R + 1,42640G - 0,01441B, \quad (3.7)$$

$$Z = -0,46807R + 0,08875G + 1,00921B. \quad (3.8)$$

Die Lab-Farbwerte können wir mit diesen Werten und dem Weißpunkt (X_n, Y_n, Z_n) berechnen, der alle RGB-Farben in der CIE-Normtafel zu je einem Drittel repräsentiert. Der Weißpunkt kann je nach Beleuchtungsart andere Werte annehmen. Wir wählen für ihn den D50-Normwert nach den Deutschen Normen (1979) mit $(X_n = 0,3457, Y_n = 0,3585, Z_n = 0,2958)$ und erhalten

$$L = 116 \cdot \sqrt[3]{\frac{Y}{Y_n}} - 16, \quad (3.9)$$

$$a = 500 \cdot \left(\sqrt[3]{\frac{X}{X_n}} - \sqrt[3]{\frac{Y}{Y_n}} \right), \quad (3.10)$$

$$b = 200 \cdot \left(\sqrt[3]{\frac{Y}{Y_n}} - \sqrt[3]{\frac{Z}{Z_n}} \right). \quad (3.11)$$

Dieser Farbraum ist gegenüber dem RGB-Farbraum so konzipiert, dass Abstände zwischen zwei Farbkoordinaten der Wahrnehmung des Menschen entsprechen. Abbildung

⁵Umrechnung nach <http://wapedia.mobi/de/Lab-Farbraum#8>.

3.6 zeigt den Lab-Farbraum, wie er in der Praxis verwendet wird. Er ist im Gegensatz zum RGB-Raum kein Würfel sondern besitzt unter Vernachlässigung der nicht wahrnehmbaren Farben eine ungleichmäßige Form.

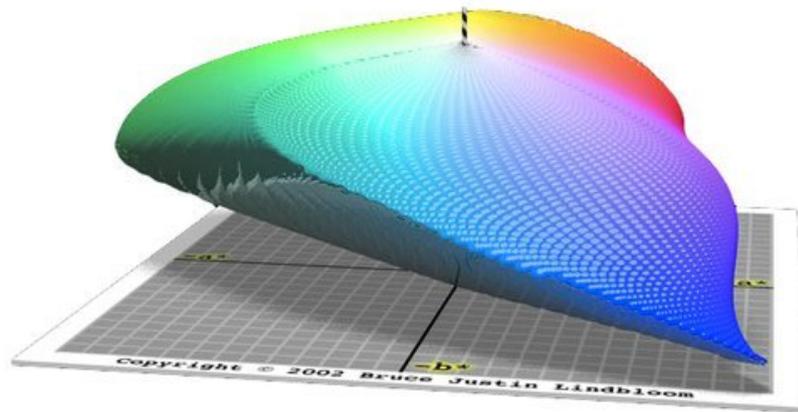


Abbildung 3.6: Der Lab Farbraum besitzt unter Vernachlässigung der nicht wahrnehmbaren Farben eine ungleichförmige Form. Er besteht aus der Lumineszenz-, der Rot-Grün- und der Gelb-Blau-Achse.

Zusätzlich zu den bisher genannten Farbräumen verwenden wir die Grauwerte der Bilder. All diese Farbräume haben zunächst keine Vorteile gegenüber einander, da sie alle die Merkmale desselben Farbeindrucks enthalten. Dadurch können also keine neuen Informationen gewonnen werden. Die Transformation in andere Farbräume kann jedoch bewirken, dass die Farbinformationen in dem einen Farbraum nicht separierbar sind, in einem anderen Farbraum jedoch sehr gut.

Verwendete Merkmale

Aus der Farbinformation an jedem Pixel können wir Merkmale extrahieren. Dazu gehören die Farbinformationen Φ_c aus den Farbräumen selbst und Merkmale aus einer

Umgebung U , wie der Mittelwert Φ_{mu} , die Standardabweichung Φ_{std} und die Histogrammwerte Φ_{hist} . Wir können diese wie folgt berechnen:

$$\Phi_{mu} = \frac{1}{N_U} \sum_{\text{Pixel} \in U} \Phi_c \quad \text{Mittelwert,} \quad (3.12)$$

$$\Phi_{std} = \sqrt{\frac{1}{N_U} \sum_{\text{Pixel} \in U} (\Phi_c - \Phi_{mu})^2} \quad \text{Standardabweichung,} \quad (3.13)$$

$$\Phi_{hist} = \frac{N_B}{N_U} \quad \text{Histogramm.} \quad (3.14)$$

Die Anzahl der in der Umgebung enthaltenen Pixel ist N_U und die Anzahl der Pixel in der Säule (engl. Bin) B des Histogramms ist N_B .

3.2.2 Positionsmerkmale

Die Positionsmerkmale beschreiben das Vorkommen einer Klasse im Bild. „Himmel“, „Erdboden“, „Gras“ und „Wasser“ können gut durch deren Position im Bild unterschieden werden. „Himmel“ befindet sich in der Regel im oberen Teil des Bildes, während „Erdboden“, „Gras“ und „Wasser“, meist nur im unteren Bereich des Bildes zu finden sind.

Auf Grund dessen führen wir ein Positionsmerkmal ein. Dies besteht aus den auf 1 normierten Positionen im Pixelkoordinatensystem. Durch die Normierung wird gewährleistet, dass die Position bildgrößen- und seitenverhältnisunabhängig ist.

3.2.3 Texturmerkmale

Texturen sind visuelle Eigenschaften der Oberfläche, die wir als Struktur wahrnehmen. Die Beschaffenheit wird vor allem durch die Musterung erkennbar. Die Textur ist neben der Farbe eine der wichtigsten Charakteristika, die die Oberfläche eines Objekts beschreiben. Die Bedeutung der Texturen wird klar, wenn Objekte, die durch ihre Farbe nicht zu unterscheiden sind, auf Grund ihrer Textur voneinander getrennt werden können. Mit Hilfe von Texturen können wir zum Beispiel sehr gut zwischen einem Schachbrett mit sich wiederholenden schwarz-weißen Quadraten, einem Zebra mit schwarz-weißen Streifen oder einer weißen Kuh mit schwarzen Flecken unterscheiden (Abbildung 3.7a-3.7c).



Abbildung 3.7: Objekte, die durch die Farbmerkmale nicht unterscheidbar sind, können durch Texturmerkmale sehr gut getrennt werden.

Die folgenden Abschnitte betrachten die Extraktion der Texturmerkmale nach Malik u. a. (2001) mittels Filterung und stellt eine Erweiterung nach Leung und Malik (2001) vor. Diesen Ansatz implementieren wir in dieser Arbeit nicht. Wir betrachten ihn trotz dessen näher, da er vielversprechend zur weiteren Merkmalsextraktion eingesetzt werden.

Verwendete Filterbank

Nach Malik u. a. (2001) ist jede Textur durch Merkmale charakterisiert, die durch Filterung hervorgehoben werden können. Im Folgenden beschreiben wir den Aufbau der Filterbank und anschließend deren Verwendung.

Für die Filterung der Bilder verwenden wir die in Abbildung 3.8 dargestellte *Leung Malik (LM) Filterbank*⁶. Sie besteht aus 48 Multiskalen- und Multiorientierungsfiltern $\{F_t\}$ der Größe 49×49 Pixel. Zu den Filtern gehören die ersten und zweiten Ableitungen eines Gaußfilters in sechs Orientierungen und drei Skalen, zusammen also 36, acht Laplacian of Gaussian (LOG) Filter und vier Gaußfilter.

Die Filterbank enthält drei verschiedene Arten von Filtern, die wir im Folgenden mit Schenk (1999, Seite 110) näher betrachten:

- Gaußfilter

Die Gaußfilter sind radialsymmetrische Glättungsfilter und dienen zur Rauschun-

⁶Download unter: <http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html>

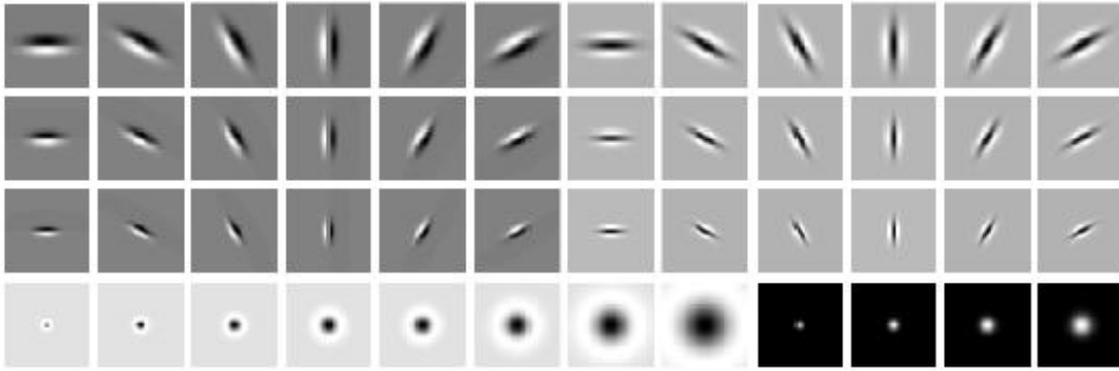


Abbildung 3.8: Die Leung Malik Filterbank besteht aus 48 Multiskalen- und Multi-orientierungsfilttern. Dazu gehören die ersten und zweiten Ableitungen eines Gaußfilters in sechs Orientierungen und drei Skalen, zusammen also 36, acht Laplacian of Gaussian (LOG) Filter und vier Gaußfilter.

terdrückung. Kleine Strukturen gehen verloren, während grobe Struktur erhalten bleiben. Der Faltungskern ergibt sich nach

$$G(N_r, N_c, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{N_r^2 + N_c^2}{2\sigma^2}\right) \quad (3.15)$$

mit der Standardabweichung σ . N_r und N_c geben die Pixelposition im Faltungskern an. In der Filterbank sind Gaußfilter mit den Standardabweichungen $\sigma = \{\sqrt{2}, 2, 2\sqrt{2}, 4\}$ enthalten.

- Laplacian of Gaussian Filter

Diese Filter sind Kantendetektoren und werden durch Anwendung des Laplace-Operators Δ auf den Gaußfilter konstruiert. Die Anwendung des Laplace-Operators auf Formel (3.15) liefert folgenden kontinuierlichen Faltungskern:

$$\Delta G(N_r, N_c, \sigma) = \frac{\partial^2 G(N_r, N_c, \sigma)}{\partial x^2} + \frac{\partial^2 G(N_r, N_c, \sigma)}{\partial y^2} \quad (3.16)$$

$$= \frac{1}{\sigma^4} \left(\frac{N_r^2 + N_c^2}{2\sigma^2} + 2 \right) \exp\left(-\frac{N_r^2 + N_c^2}{2\sigma^2}\right). \quad (3.17)$$

Diesen Faltungskern diskretisieren wir. In der Filterbank sind Laplacian of Gaussian Filter der Standardabweichungen σ und 3σ enthalten.

- Erste und zweite Ableitungen eines Gaußfilters

Die ersten und zweiten Ableitungen der Gaußfilter entstehen auf den drei Skalen

$\sigma = \{\sqrt{2}, 2, 2\sqrt{2}\}$ mit einem Dehnungsfaktor von 3, also zum Beispiel $\sigma_x = \sigma$ und $\sigma_y = 3\sigma_x$.

Filterung der Bilder

Der folgende Abschnitt erläutert die Verwendung der soeben vorgestellten Filterbank nach Malik u. a. (2001) und zeigt deren Nutzen für die Texturanalyse.

Auf Grund der hohen Rechenintensität verkleinern wir zu große Bilder zunächst auf 150×150 Pixel. Anschließend transformieren wir sie in den HSV-Kanal, da dieser die Intensitätsinformation im Helligkeits- und Sättigungskanal vom Farbton trennt. Die Informationen über Beleuchtung und Schatten befinden sich alle im Helligkeitskanal. Die Idee hierfür stammt aus Durupinar (2005). Er richtete dort seine Aufmerksamkeit auf die Extraktion von Merkmalen aus dem HSV-Farbraum und vergleicht sie mit der Extraktion durch Anwendung einer Filterbank. Er kommt zu dem Schluss, dass Texturen unter verschiedenen Beleuchtungen nicht als ähnlich erkannt werden. Unter diesem Gesichtspunkt kombinieren wir den Ansatz von Malik u. a. (2001) und Durupinar (2005) und extrahieren die Texturen nur aus dem Sättigungs- und dem Farbtonkanal um eine Unabhängigkeit von der Beleuchtung zu erreichen. Wir erhalten somit die Merkmalsvektoren Φ_{hs} .

Die bearbeiteten Bilder falten wir mit allen Schichten der Filterbank, wodurch wir so genannte *Filterantworten* erhalten. Die Matrix der Filterantworten besteht aus $N_R \cdot N_C$ Merkmalsvektoren Φ_{fr} der Dimension $T \times 1$, wobei T die Anzahl der Filter ist. Jeder Merkmalsvektor charakterisiert einen um einen Pixel y_n zentrierten Bildausschnitt der Größe des Filters F_t . Die Merkmalsvektoren können untereinander sehr ähnlich sein, da es sich bei Texturen meist um wiederholte Strukturen handelt, die im Bild immer wieder auftreten.

Um die Matrix der Filterantworten mit der Größe des Bildes zu erhalten, müssen wir ein Verfahren für die Randfortsetzung finden. Das bedeutet, dass wir eine Textur mit einer homogenen Fläche anfügen. Bei einer Faltung eines Grauwertbildes mit einem Gaußfilter zum Beispiel führt dies zu einer Verdunklung des Bildes. Alternativ könnten wir das Bild gespiegelt anfügen, was bei nicht-symmetrischen Strukturen zu Problemen führt, oder das Bild zyklisch fortsetzen. Letzteres ist problematisch, wenn Regionen am Rand schmal sind und mit Strukturen anderer Regionen fortgeführt würden. Da für



Abbildung 3.9: Dieses Bild aus dem Sowerby Datensatz skalieren wir auf die Größe 150×150 Pixel, falten es mit der Filterbank und clustern die Filterantworten, um die dominanten Strukturmerkmale aus Abbildung 3.10a und 3.10b zu erhalten.

die Faltung keine neuen Strukturen oder Strukturen anderer Regionen mit einbezogen werden sollen, wählen wir die Fortsetzung mit Nullen.

Nach Leung und Malik (2001) gibt es einige wenige dominante Merkmale und alle anderen sind verrauschte Abbilder von ihnen. Die dominanten Merkmale bilden Clusterzentren und können mit dem *k-means Clusteralgorithmus* (Bishop (2006)) extrahiert werden. Sie werden *Strukturmerkmale* (engl. *Textons*) genannt und gelten als Charakteristika einer Textur. Abbildung 3.9 zeigt ein Bild aus dem Sowerby Datensatz mit den 25 dominantesten Strukturmerkmalen des Farbton- und des Sättigungskanals, die in Abbildung 3.10a und 3.10b dargestellt sind. Die Strukturmerkmale können wir visualisieren, indem wir zunächst jeden Filter in der Filterbank vektorisieren, die Strukturmerkmale mit der Pseudoinversen der vektorisierten Filterbank multiplizieren und sie anschließend wieder zu einer Matrix umordnen. Die visualisierten Strukturmerkmale stellen eine Ansammlung von Filterantworten dar, die charakteristisch für das Bild sind.

Der Großteil der visualisierten Strukturmerkmale weist eine Orientierung auf, die durch überwiegend schräg verlaufende Texturen im Bild entstehen. Vereinzelt treten kreisrunde Strukturmerkmale auf. Alle sind durch einen hohen Kontrast geprägt, was bedeutet, dass der Anteil von stark strukturierten Regionen und ausgeprägten Kanten im Bild sehr hoch ist und homogene Regionen kaum auftreten.



(a) Dominante Strukturmerkmale des Farbtonkanals (b) Dominante Strukturmerkmale des Sättigungskanals

Abbildung 3.10: Die 25 dominantesten Strukturmerkmalen des Farbton- und des Sättigungskanals des Bildes aus Abbildung 3.9 visualisieren wir, indem wir zunächst jeden Filter in der Filterbank vektorisieren, die Strukturmerkmale mit der Pseudoinversen der vektorisierten Filterbank multiplizieren und anschließend wieder zu einer Matrix umordnen.

Wir können die Filterantworten den Strukturelementen im Merkmalsraum zuordnen, die ihnen am nächsten liegen, wodurch eine so genannte *Texton-Karte* entsteht. Diese Zuordnung stellt eine Klassifikation durch Clustering der Texturen dar. Die Texton-Karte enthält die Texturklassen, wobei je eine Klasse eine andere Farbe besitzt. Abbildung 3.11 zeigt die Texton-Karte des Bildes aus Abbildung 3.9 mit den dominanten Strukturmerkmalen aus den Abbildungen 3.10a und 3.10b.

Aus der Abbildung 3.11 ist ersichtlich, dass Strukturen wie die Fahrspuren, die Wiese, der Himmel und die Büsche gut als Textur erkannt werden. Die Vegetation jedoch zeichnet sich im oberen Bereich des Bildes durch viele verschiedene Texturen aus, sodass dort kaum größere Regionen mit gleichen Texturen vorkommen.

Nach Leung und Malik (2001) können wir die Textur einer Region durch ein oder mehrere Histogramme beschreiben. Die Histogramme können wir aus den Trainingsdaten lernen. Dazu klassifizieren wir mehrere Trainingsbilder zu einer Texton-Karte, wie oben beschrieben, und tragen die Anzahl der dominanten Strukturelemente jeder Klasse in einem Histogramm auf. Die Histogramme aus mehreren Regionen der gleichen Klasse ähneln sich meist stark und können auf einige wenige typische Histogramme reduziert werden.



Abbildung 3.11: Die Texton-Karte des Bildes aus Abbildung 3.9 entsteht durch Zuordnung der Pixel zu dem Strukturmerkmal, welches im Merkmalsraum am nächsten liegt. Sie enthält die Texturklassen, wobei je eine Klasse eine andere Farbe besitzt.

Die Histogramme der Testdaten erstellen wir an jeder Stelle eines Pixels aus dessen Umgebung. Zunächst falten wir das Bild mit der Filterbank, klassifizieren die Umgebung des Pixels zu einer Texton-Karte und berechnen daraus das Histogramm über die Strukturelemente. Dieses vergleichen wir mit allen gelernten Histogrammen und ordnen es dem ähnlichsten zu. Statt die Filterantworten als Merkmale zu verwenden, können wir auch die Label dieser ähnlichsten Histogramme als Merkmale wählen.

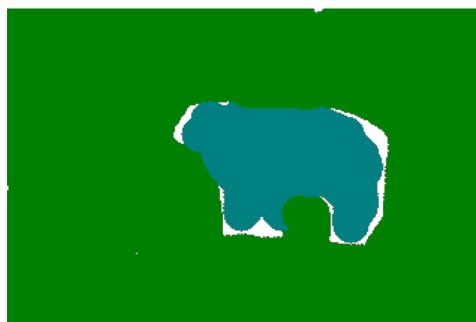
Diesen Ansatz verfolgen wir in dieser Arbeit nicht. Statt der Reduktion auf repräsentative Strukturelemente verwenden wir alle Filterantworten und überlassen die Trennung den Diskriminanten. Der Nachteil ist, dass die Cluster der Filterantworten für eine Klasse ungünstig im Merkmalsraum liegen können. Ungünstig heißt, dass wir sie nicht durch eine lineare Hyperebene von den anderen Clustern trennen können. Dies tritt zum Beispiel auf, wenn ein Cluster einer Klasse zwischen zwei Clustern einer anderen Klasse liegt. Wir wählen den Ansatz der Diskriminanten dennoch, da einerseits keine Informationen durch eine Vorklassifizierung der Filterantworten verloren gehen und wir andererseits auf eine „harte“ Zuweisung zu einer Klasse verzichten und die jeweilige Wahrscheinlichkeit verwenden.

3.2.4 Kontextmerkmale

Die Kontextmerkmale verwenden das ganze Bild statt nur einer kleinen Umgebung eines Pixels. Dies hat den Vorteil, dass sich Regionen in den Gesamtkontext des Bildes einordnen können. Die Abbildungen 3.12a und 3.12b zeigen ein Beispiel, bei dem wir die Kontextmerkmale sehr gut einsetzen können.



(a) Schaf auf einer Wiese



(b) Annotation des Schafes auf der Wiese mit blau als die Klasse „Schaf“, grün als die Klasse „Gras“ und weiß als die Klasse „void“

Abbildung 3.12: Durch die Kontextmerkmale können wir das Schaf in den Gesamtkontext des Bildes einordnen. Das Schaf erhält als Kontextmerkmale die Farbmerkmale der Wiese und andersherum.

Wir ermitteln zunächst die Farbmerkmale jedes Pixels. Dazu gehören die RGB-, HSV- und Lab-Werte. Anschließend erstellen wir für jedes Merkmal ein Histogramm mit zehn Säulen, sodass wir zehn Histogramme mit den Häufigkeiten jedes Farbmerkmals erhalten. Allen Pixeln im Bild, die den häufigsten Wert eines Histogramms besitzen, ordnen wir den zweithäufigsten Wert zu. Allen anderen weisen wir den häufigsten Wert als Merkmal zu. Wir führen dies für alle zehn Histogramme durch. Für das Bild aus Abbildung 3.12a würde dies bedeuten, dass alle Pixel mit der Klasse „Schaf“ die Merkmale der Klasse „Gras“ erhalten und andersherum.

Mit den in diesem Kapitel vorgestellten Merkmalen kann nun das lineare probabilistische lineare Modell gelernt und das Bild segmentiert werden. Die Segmentierung erfolgt mit Markov Random Fields, mit denen sich das nächste Kapitel befasst.

4 Segmentierung unter Verwendung von Markov Random Fields

Mit Hilfe der letzten beiden Kapitel können wir nun Merkmale aus den Bildern extrahieren und diese mittels eines linearen probabilistischen diskriminativen Modells trennen. Die a posteriori Wahrscheinlichkeiten der Pixel für jede Klasse erhalten wir über die Softmax-Transformation der Diskriminanten, wie in Kapitel (2.3) beschrieben. Nun können wir mit diesen Informationen die Bildsegmentierung durch die Klassifikation der Pixel beziehungsweise Superpixel durchführen. Wir könnten bereits mit dem linearen probabilistischen diskriminativen Modell eine Klassifikation durchführen, allerdings bieten die Markov Random Fields Vorteile, die wir im Folgenden näher betrachten.

Im Falle der pixelweisen Klassifikation mit den Merkmalen aus einer Rechteckumgebung oder bei der regionenbasierten Klassifikation mit sehr kleinen Regionen kann es auf Grund von Ausreißern passieren, dass das klassifizierte Bild verrauscht ist, das heißt, dass sich vereinzelte Label von ihrer Umgebung unterscheiden. Die Wahrscheinlichkeit eines Labels für eine Klasse \mathcal{C}_k ist umso höher, je mehr Label derselben Klasse sich in der direkten Umgebung befinden. Dies stützt sich wieder auf die Annahme, dass homogene Regionen häufiger auftreten und somit wahrscheinlicher sind, als abrupte Übergänge oder Kanten. Dieses Prinzip steht hinter den Markov Random Fields, wodurch sie sehr gut für die Segmentierung der Bilder geeignet sind. Sie nutzen einerseits die a posteriori Wahrscheinlichkeiten für jede Klasse, berücksichtigen jedoch auch, dass homogene Regionen wahrscheinlicher sind.

Das folgende Kapitel zeigt zunächst den Aufbau der Markov Random Fields, erläutert danach die Möglichkeiten der Inferenz, um anschließend die Loopy Belief Propagation nach Pearl zu erläutern, die wir in dieser Arbeit für die Segmentierung der Bilder verwenden.

4.1 Aufbau

Ein *Markov Random Field* ist ein ungerichtetes graphisches Modell und besteht aus Beobachtungen, in unserem Fall die Menge⁷ der Merkmalsvektoren Φ , denen je eine verborgene (engl. hidden) Variable $x = x_n$ aus der Menge $\mathcal{X} = \{x_1, \dots, x_N\}$ zugeordnet ist. Diese Variablen lernten wir bereits in Kapitel 2 als Label kennen. Letztere sind nicht beobachtet und somit nicht bekannt. Diese Variablen können wie in unserem Fall von einem anderem Typ, wie zum Beispiel einer Klasse, oder vom gleichen Typ, also einem Pixelwert, sein. Sie können außerdem eine andere Dimension annehmen, sodass mehrere unbeobachtete Variablen auf eine Beobachtung kommen. In dieser Arbeit nehmen wir je eine unbeobachtete Variable pro Beobachtung an, die die Zugehörigkeit zu einer Klasse angibt. Alle Variablen formen auf Grund der Markoveigenschaft ein Markov Modell, das durch ein Verbund von Knoten und Kanten gekennzeichnet ist. Die Markoveigenschaft besagt, dass die Wahrscheinlichkeit für einen Knoten x , wenn alle anderen Knoten im Feld \mathcal{X} gegeben sind, genauso groß ist, wie die Wahrscheinlichkeit, falls nur seine Nachbarn \mathcal{U}_x gegeben sind:

$$p(x|\mathcal{X}) = p(x|\mathcal{U}_x). \quad (4.1)$$

Der Verbund der Knoten ist durch weiche, das heißt ungerichtete Beziehungen zwischen den Knoten charakterisiert, die wir im Folgenden anhand der Abbildung 4.1 näher betrachten.

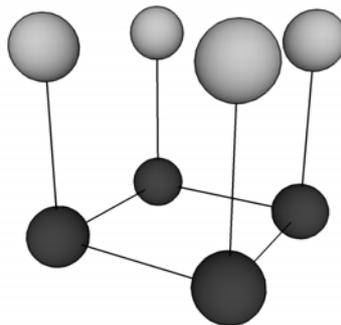


Abbildung 4.1: Ein Markov Random Field mit Labeln \mathcal{X} (dunkelgrau) und Merkmalen Φ (hellgrau) ist durch weiche, also ungerichtete Beziehungen zwischen den Knoten gekennzeichnet.

⁷Achtung: In diesem Kapitel bezeichnen wir mit Φ nicht die Matrix, sondern die Menge der Merkmalsvektoren.

Abbildung 4.1 zeigt die Merkmale Φ (gezeichnet in hellgrau) und die unbeobachteten Label \mathcal{X} eines Markov Random Fields. Sie sind als Knoten dargestellt, die jeweils eine Beobachtung beziehungsweise einen unbeobachteten Zustand repräsentieren. Im Folgenden kann ein Label den Wert $x = \{1, \dots, K\}$ annehmen. Wenn sich x im Zustand k befindet, wird die Beobachtung der Klasse \mathcal{C}_k zugeordnet. Die Knoten $\{x_n, x_m\}$, mit n und m als Indizes benachbarter Knoten, sind miteinander verbunden und formen *Cliquen*. Alle Knoten einer Clique sind komplett miteinander verbunden, das heißt sie sind so miteinander verknüpft, dass keine Knoten zwischen ihnen liegen. In Abbildung 4.1 ist die maximale Cliquengröße zwei.

Die Cliquen $\{x_n, \phi_n\}$ stellen die Verbindung zwischen beobachteten und unbeobachteten Knoten dar. Ohne diesen Bezug zu den Merkmalen wäre das Markov Random Field ein ideales Modell, so dass auf Grund der bevorzugten Homogenität immer die Felder mit nur einem Zustand die höchste Wahrscheinlichkeit erhalten würden. Wir könnten somit nichts über die Wahrscheinlichkeit von Merkmalen bei einem gegebenen Modell aussagen. Das ideale Modell liefert lediglich die a priori Wahrscheinlichkeiten der Knoten.

4.2 Energiefunktion

Das Ziel der Anwendung eines Markov Random Fields ist der Rückschluss auf die wahrscheinlichsten Label bei gegebenen Merkmalen. Dazu benötigen eine Energiefunktion, die der folgende Abschnitt näher erläutert.

Um auf die Wahrscheinlichkeit $p(\mathcal{X}|\Phi)$ aller Label \mathcal{X} bei den gegebenen Merkmalen Φ schließen zu können, wenden wir die Bayesformel an:

$$p(\mathcal{X}|\Phi) = \frac{p(\Phi|\mathcal{X})p(\mathcal{X})}{p(\Phi)}. \quad (4.2)$$

Da $p(\Phi)$ nur als Normalisierungskonstante fungiert, gilt

$$p(\mathcal{X}|\Phi) \propto p(\Phi|\mathcal{X})p(\mathcal{X}) \quad (4.3)$$

und wegen $p(\mathcal{X}, \Phi) = p(\Phi|\mathcal{X})p(\mathcal{X})$ somit näherungsweise

$$p(\mathcal{X}|\Phi) \propto p(\mathcal{X}, \Phi). \quad (4.4)$$

Da wir die wahrscheinlichsten Label suchen, genügt es statt der bedingten Wahrscheinlichkeiten die gemeinsamen zu berechnen. Gesucht ist nun die gemeinsame Wahrscheinlichkeit der Merkmale Φ und der Label \mathcal{X} . Wir können die Beziehungen, die in Abbildung 4.1 dargestellt sind, in einer *Energiefunktion* nach Bishop (2006, Kapitel 8, Seite 389) und Potts (1952) zusammenfassen und für Knoten mit den Zustandswerten $x = \{1, \dots, C\}$ wie folgt formulieren:

$$E(\mathcal{X}, \Phi) = - \sum_{n,m} \beta \cdot \delta(x_n, x_m) + \sum_n \eta \cdot \log(p(\phi_n | x_n)). \quad (4.5)$$

Dieses Modell wird *Potts Modell* genannt und stellt eine Generalisierung des Ising Modells dar, bei dem die unbeobachteten Zustandsknoten die Werte $x = \{-1, 1\}$ annehmen können. Der erste Term (*binary energy*) mit der Konstanten β beschreibt die Energie zwischen x_n und dessen Nachbarn x_m . Er beinhaltet die Deltafunktion δ , die im Falle gleicher Klassenzugehörigkeit 1 und im Falle unterschiedlicher Klassenzugehörigkeit der Nachbarn 0 ist. Falls β negativ ist, nimmt der Term einen niedrigen Wert an, wenn β positiv ist, einen hohen. Der zweite Term mit der Konstanten η beschreibt den Bezug zu den Merkmalen, wobei $p(\phi_n | x_n)$ die a posteriori Wahrscheinlichkeiten der Merkmale für jede Klasse sind, die wir aus der Klassifikation mit dem linearen probabilistischen diskriminativen Modell erhalten haben. Dies können wir annehmen, da in dem Modell die a priori Wahrscheinlichkeiten für die Klassen gleich wahrscheinlich sind. Somit gilt mit der Bayesformel (4.2), dass die a posteriori Wahrscheinlichkeiten gleich der klassenbedingten Wahrscheinlichkeiten sind:

$$p(\mathcal{X} | \Phi) = \frac{1}{Z} p(\Phi | \mathcal{X}) \cdot \mathbf{1}. \quad (4.6)$$

Die Normalisierungskonstante Z dient dazu, dass sich die Wahrscheinlichkeiten $p(\mathcal{X} | \Phi)$ zu 1 summieren. Sie ist identisch mit dem Nenner der Bayesformel (4.2).

Der Zusammenhang zwischen der Energiefunktion und der Wahrscheinlichkeit wird durch folgende Ausführungen deutlich. Die Energiefunktion ohne Merkmalsbezug lautet:

$$E(\mathcal{X}) = - \sum_{n,m} \beta \cdot \delta(x_n, x_m). \quad (4.7)$$

Wir führen das *Hammersley-Clifford Theorem* nach Clifford (1990) (zit. n. Bishop (2006, Kapitel 8, Seite 387)) ein und erhalten

$$\psi_C = \exp(-E(\mathcal{X}_C)), \quad (4.8)$$

wobei \mathcal{X}_C die Clique angibt, die aus x_n und x_m besteht. Die *Potentialfunktion* ψ_C enthält die nicht normierte Energie der in der Clique enthaltenen unbeobachteten Knoten. Das Potenzieren garantiert stets positive Werte. Die Wahrscheinlichkeit $p(\mathcal{X})$ können wir mit dem Produkt über die Potentialfunktionen aller Cliques, normiert mit Z , berechnen:

$$p(\mathcal{X}) = \frac{1}{Z} \prod_C \psi_C(\mathcal{X}_C). \quad (4.9)$$

Fügen wir nun noch den Bezug zu den Merkmalen durch die klassenbedingten Wahrscheinlichkeiten $p(\Phi|\mathcal{X})$ hinzu, gilt der Zusammenhang:

$$p(\mathcal{X}, \Phi) = p(\mathcal{X}) \cdot p(\Phi|\mathcal{X}) \quad (4.10)$$

$$= \frac{1}{Z} \prod_C \psi_C(\mathcal{X}_C) \cdot p(\Phi|\mathcal{X}) \quad (4.11)$$

$$= \frac{1}{Z} \prod_C \exp(-E(\mathcal{X}_C)) \cdot p(\Phi|\mathcal{X}). \quad (4.12)$$

Die Maximierung der Wahrscheinlichkeit ist somit äquivalent zu der Minimierung der Energiefunktion. Da es meist schwierig ist die Normierungskonstante Z zu berechnen, genügt es bei Fragestellungen nach den wahrscheinlichsten Labeln, nur die Energiefunktion zu berechnen.

4.3 Lernen der Parameter

Der letzte Abschnitt befasste sich mit der Energiefunktion und der Bedeutung der Parameter $\theta = \{\beta, \eta\}$. Diese Parameter können durch Maximierung der Likelihoodfunktion $L(\theta)$ bestimmen:

$$\exp(L(\boldsymbol{\theta})) = \prod_{\mathcal{X}} p(\mathcal{X}, \Phi|\boldsymbol{\theta}), \quad (4.13)$$

$$L(\boldsymbol{\theta}) = \sum_{\mathcal{X}} \log(p(\mathcal{X}, \Phi|\boldsymbol{\theta})). \quad (4.14)$$

Die optimalen Parameter können wir durch das Argument des Maximums der Likelihood Funktion bestimmen:

$$\hat{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}). \quad (4.15)$$

Anders als beim Lernen der Parameter des linearen Modells verwenden wir zum Lernen dieser Parameter die Validierung. Das bedeutet, dass wir einige Parameterwerte testen und deren Klassifikationserfolg unter ihrer Verwendung ermitteln. Der Unterschied zur Minimierung des konvexen Optimierungsproblems ist, dass wir mit diskreten Parameterwerten arbeiten statt mit einer kontinuierlichen Fehlerfläche. Auf Grund dessen ist die Bestimmung der Markov Random Field Parameter nicht so präzise, wie die des linearen Modells.

Wir wählen den Parameter β für alle Cliques und den Parameter η für alle klassenbedingten Wahrscheinlichkeiten gleich. Beide Parameter stellen ein Gewicht dar, sodass das Verhältnis entscheidend dafür ist, ob die Homogenität eine große oder eher eine kleine Rolle im Gegensatz zu den klassenbedingten Wahrscheinlichkeiten spielt. Die verwendeten Werte und die Ergebnisse betrachten wir in Kapitel 6 genauer.

4.4 Inferenz

Die folgenden Abschnitte betrachten die Inferenz näher. Inferenz bedeutet den Rückschluss auf die wahrscheinlichsten Label bei gegebenen Merkmalen. Dies erreichen wir durch die Berechnung aller Randwahrscheinlichkeiten $p(x|\Phi)$. Das wahrscheinlichste Labeling ergibt sich durch die Wahl der höchsten Wahrscheinlichkeit an jedem Knoten x . Der Abschnitt geht auf zwei Berechnungsmöglichkeiten ein, wobei wir eine von ihnen, die Loopy Belief Propagation, genauer betrachten. Wir verwenden sie in dieser Arbeit zur Segmentierung der Bilder.

4.4.1 Bestimmung aller Randwahrscheinlichkeiten

Die intuitivste Lösung ist die Bestimmung der Randwahrscheinlichkeit für einen Knoten x durch Summieren über alle Klassen \mathcal{X} außer x selbst:

$$p(x|\Phi) = \sum_{\mathcal{X} \setminus x} p(\mathcal{X}|\Phi). \quad (4.16)$$

Die Formel zeigt, dass wir die Randwahrscheinlichkeit $p(x_n = k|\Phi_n)$ erhalten, indem wir alle gemeinsamen Wahrscheinlichkeiten $p(\mathcal{X}|\Phi)$ addieren, die die Klasse $x_n = k$ enthalten. Führen wir dies für alle K Klassen durch, können wir eine Aussage über die Wahrscheinlichkeiten für jede Klasse $x_n = k$ unabhängig von den anderen Variablen tätigen. Somit vereinigen sich die Informationen aller Label der anderen Knoten in diesem.

Beispiel

Gegeben seien zwei Knoten $\{x_1, x_2\}$ und die Merkmale $\{\phi_1, \phi_2\}$ mit gemeinsamen Wahrscheinlichkeiten $p(x_1 = k, x_2 = j|\phi_1, \phi_2)$. Diese können wir in einer Matrix zusammenfassen, dargestellt in Tabelle 4.1.

$p(x_2 \phi_2)$			
0,4	$x_2 = 1$	0,1	0,3
0,6	$x_2 = -1$	0,4	0,2
		$x_1 = 1$	$x_1 = -1$
		0,5	0,5
			$p(x_1 \phi_1)$

Tabelle 4.1: Die Berechnung der Randwahrscheinlichkeit zweier Knoten $\{x_1, x_2\}$ mit den Merkmalen $\{\phi_1, \phi_2\}$ können wir anschaulich in einer Tabelle schreiben.

Die Randwahrscheinlichkeit berechnen wir durch Summieren der gemeinsamen Wahrscheinlichkeiten, die je an einer Stelle in der Matrix eingetragen sind. Je nach gesuchter Randwahrscheinlichkeit summieren wir jeweils über die Spalten (gesucht: $p(x_1|\phi_1)$) oder die Zeilen (gesucht: $p(x_2|\phi_2)$), sodass wir eine Aussage über die Wahrscheinlichkeit einer Klasse an einem Knoten machen können. Der Vektor der gesuchten Randwahrscheinlichkeit hat die Dimension $K \times 1$ mit K Klassen.

Die Lösung der Randwahrscheinlichkeit aus dem oben genannten Beispiel ist:

$$p(x_1|\phi_1) = [0,5, 0,5]^T,$$
$$p(x_2|\phi_2) = [0,4, 0,6]^T.$$

Die Anzahl der Zeilen und Spalten ist durch die Anzahl der Klassen gegeben. Die Dimensionalität der Matrix ist durch die Anzahl der Knoten festgelegt, die an der gemeinsamen Wahrscheinlichkeit teilnehmen. Sie wächst also linear mit der Anzahl der Knoten. Im Falle von drei Knoten ergäbe sich beispielsweise ein Würfel.

Dieses Verfahren stellt die „Straight-Forward“-Lösung der Bestimmung von den wahrscheinlichsten Klassen dar. Es gibt jedoch eine elegantere Lösung mit vielen Vorteilen, die wir im folgenden Abschnitt betrachten.

4.4.2 Message Passing

Die Berechnung der Wahrscheinlichkeit der Label in einem Markov Random Field ist durch den extrem hohen Rechenaufwand begrenzt, da wir alle möglichen Zustandskombinationen von \mathcal{X} bestimmen müssen. In einem Bild der Größe 200×200 Pixel mit $K = 4$ möglichen Klassenzugehörigkeiten wären dies $4^{200 \cdot 200} \approx 10^{24000}$ Kombinationen. Um diesen hohen Rechenaufwand zu umgehen, wird das *Message Passing* genutzt. Dies ist eine einfache, effiziente Methode, mit der wir komplizierte und aufwendige Berechnungen schnell und exakt lösen können. Der nächste Abschnitt erläutert zunächst das Message Passing, damit wir im nachfolgenden Abschnitt die Loopy Belief Propagation betrachten können, die das Message Passing verwendet.

(MacKay, 2003, Kapitel 16, Seite 242) erklärt das Prinzip des Message Passings mit Hilfe einer Gruppe von Soldaten, die sich durchzählen sollen. Da sie sich im Nebel befinden, können sie sich nicht alle gegenseitig sehen. Um die Aufgabe zu lösen, stellen sie sich alle in einer Reihe auf, der hinterste und der vorderste Mann beginnen bei 1 zu zählen und lassen ihren Vorder- beziehungsweise Hintermann immer 1 auf die Anzahl addieren, bis ein Soldat von beiden Seiten eine Zahl erhält, die er nur noch miteinander und mit 1 addieren muss. Dieses Beispiel zeigt, dass eine lokale Kommunikation genügt, um eine globale Information zu erhalten.

Pearl (1982) formuliert nach diesem Prinzip einen Algorithmus für baumartige graphische Modelle. Der einfachste graphische Baum ist die Markovkette, auf die wir obiges Beispiel direkt anwenden können. Dazu führen wir nach MacKay (2003) *Faktorknoten* $f_m(\mathcal{X}_m|\Phi_m)$ ein, die die Bündelung zweier Knoten $\mathcal{X}_m = \{x_{m_1}, x_{m_2}\}$ mit ihren Merkmalen $\Phi_m = \{\phi_{m_1}, \phi_{m_2}\}$ darstellt. Die Wahrscheinlichkeit eines Labelings ausgedrückt mit Faktorknoten lautet

$$p(\mathcal{X}|\Phi) = \frac{1}{Z} \prod_{m=1}^M \exp(-E_m(\mathcal{X}_m, \Phi_m)) = \frac{1}{Z} \prod_{m=1}^M f_m(\mathcal{X}_m, \Phi_m), \quad (4.17)$$

wobei der Faktorknoten durch

$$f_m(\mathcal{X}_m, \Phi_m) = \exp(-E_m, \Phi_m) \quad (4.18)$$

$$= \exp(-b \cdot \delta(x_{m_1}, x_{m_2})) \cdot \exp(-\eta) p(\phi_{m_1}|x_{m_1}) p(\phi_{m_2}|x_{m_2}) \quad (4.19)$$

definiert ist. Die Wahrscheinlichkeit eines Labelings $\mathcal{X} = \{x_1, \dots, x_N\}$ bestehend aus N Knoten mit den Merkmalen $\Phi = \{\phi_1, \dots, \phi_N\}$ ist nach Formel (4.17) als Produkt von M Faktorknoten definiert:

$$p(\mathcal{X}|\Phi) = \frac{1}{Z} \prod_{m=1}^M f_m(\mathcal{X}_m, \Phi_m),$$

$$p^*(\mathcal{X}|\Phi) = \prod_{m=1}^M f_m(\mathcal{X}_m, \Phi_m) \quad (4.20)$$

Beispiel

Gegeben sei eine Markovkette mit drei Knoten $\mathcal{X} = \{x_1, x_2, x_3\}$, die wir der Übersichtlichkeit halber ohne Merkmalsbezug betrachten.

Die Markovkette besteht aus fünf Teilmengen \mathcal{X}_m : $\mathcal{X}_1 = \{x_1\}$, $\mathcal{X}_2 = \{x_2\}$, $\mathcal{X}_3 = \{x_3\}$, $\mathcal{X}_4 = \{x_1, x_2\}$ und $\mathcal{X}_5 = \{x_2, x_3\}$. Die gemeinsame Wahrscheinlichkeit der Kette berechnet sich daher mit folgenden Faktorknoten:

$$p(\mathcal{X}) = \frac{1}{Z} f_1(x_1) f_2(x_2) f_3(x_3) f_4(x_1, x_2) f_5(x_2, x_3), \quad (4.21)$$

$$p^*(\mathcal{X}) = f_1(x_1) f_2(x_2) f_3(x_3) f_4(x_1, x_2) f_5(x_2, x_3) \quad (4.22)$$

Eine Funktion der Form (4.20) können wir als *Faktorgraph* mit runden Knoten und eckigen Faktorknoten darstellen. Auch hier betrachten wir der Übersichtlichkeit halber die nächsten Ausführungen ohne Datenbezug.

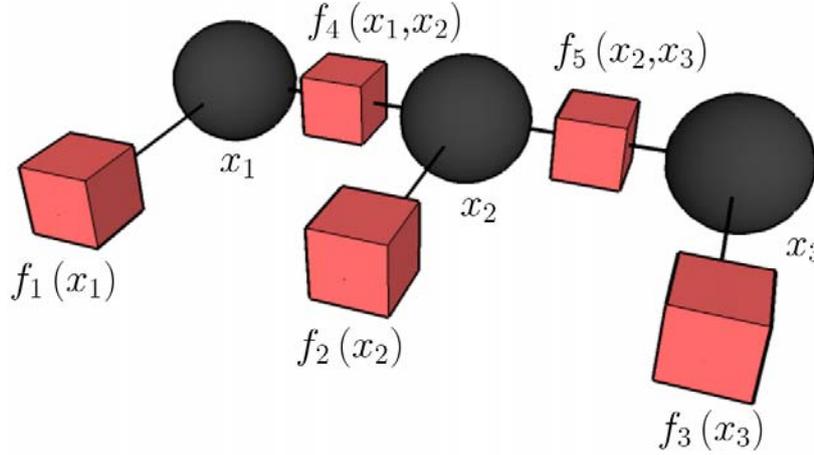


Abbildung 4.2: Der Faktorgraph des Beispiels aus Formel (4.22) mit runden Variablenknoten $\{x_1, x_2, x_3\}$ und eckigen Faktorknoten $\{f_1, f_2, f_3, f_4, f_5\}$ besteht aus den Teilmengen $\mathcal{N}(1) = \{1\}$, $\mathcal{N}(2) = \{2\}$, $\mathcal{N}(3) = \{3\}$, $\mathcal{N}(4) = \{1, 2\}$, $\mathcal{N}(5) = \{2, 3\}$ und $\mathcal{M}(1) = \{1, 4\}$, $\mathcal{M}(2) = \{2, 4, 5\}$, $\mathcal{M}(3) = \{3, 5\}$.

Mit Hilfe dieses Graphs können wir die Formel für die Bestimmung der Randwahrscheinlichkeit eines beliebigen Knotens x verdeutlichen. Wir führen zunächst zwei Mengen, bestehend aus Indizes ein: Die Menge der Indizes derjenigen Variablenknoten $\mathcal{N}(m)$, mit denen der Faktorknoten f_m direkt verbunden ist, und die Menge der Indizes der Faktorknoten $\mathcal{M}(n)$, mit denen der Variablenknoten x_n direkt verbunden ist. Weiterhin benötigen wir zwei verschiedene Formeln: eine zur Berechnung der Nachricht q eines Knotens x_n zu einem Faktor f_m und eine zur Berechnung der Nachricht r von einem Faktor f_m zu einem Knoten x_n .

Variable-zu-Faktor:

$$q_{n \rightarrow m}(x_n) = \prod_{m' \in \mathcal{M}(n) \setminus m} r_{m' \rightarrow n}(x_n) \quad (4.23)$$

Faktor-zu-Variable:

$$r_{m \rightarrow n}(x_n) = \sum_{n' \in \mathcal{N}(m) \setminus n} \left(f_m(\mathcal{X}_m) \prod_{n' \in \mathcal{N}(m) \setminus n} q_{n' \rightarrow m}(x_{n'}) \right) \quad (4.24)$$

Ein Faktorknoten, der gleichzeitig ein Blattknoten ist, sendet die Nachricht

$$r_{m \rightarrow n}(x_n) = f_m(x_n), \quad (4.25)$$

da $\mathcal{N}(m) \setminus n$ eine leere Teilmenge und somit ein leeres Produkt mit dem Ergebnis 1 ist. Diese Nachrichten sind die Likelihoodfunktionen, also die klassenbedingten Wahrscheinlichkeiten.

Ein Variablenknoten, der zusätzlich ein Blattknoten ist, sendet die Nachricht

$$q_{n \rightarrow m}(x_n) = 1, \quad (4.26)$$

da die Teilmenge $\mathcal{M}(n) \setminus m$ leer ist.

Die Initialisierung der Nachrichten kann auf zwei Weisen erfolgen:

Methode 1

Äquivalent zu der Soldatenkette initialisieren wir alle Blattknoten gemäß Formel (4.25) und (4.26). Eine Nachricht senden wir dann und nur dann an den nächsten Knoten, wenn alle Nachrichten bereits eingegangen sind, auf die diese Nachricht am aktuellen Knoten aufbaut. Die Nachrichten fließen solange durch den Baum – eine in jede Richtung entlang der Kanten –, bis alle Nachrichten erzeugt wurden.

Methode 2

Eine zweite Möglichkeit ist das Initialisieren aller Variablen mit 1 gemäß Formel (4.26). Die Nachrichten updaten wir anschließend mit (4.23) und (4.24), so dass das Ergebnis schrittweise gegen das der ersten Methode konvergiert.

In dieser Arbeit verwenden wir die zweite Methode. Die bisherigen Überlegungen basieren auf der Annahme von baumartigen graphischen Modellen. Die Markovfelder der Bilder sind jedoch nicht baumartig. Mit der Anwendung des Message Passings auf Markovfelder mit Zyklen beschäftigt sich der nächste Abschnitt.

4.4.3 Loopy Belief Propagation

Das soeben beschriebene Message Passing eignet sich sehr gut für baumartige graphische Modelle, kann jedoch nicht unmittelbar auf Markov Random Fields angewendet werden.

Der folgende Abschnitt stellt die *Loopy Belief Propagation* vor, mit der wir das Message Passing auf Markovfelder mit Zyklen (Abbildung 4.3) anwenden können.

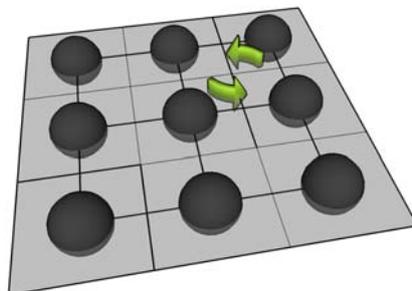


Abbildung 4.3: Ein Markov Random Field auf Bilder angewandt enthält Loops (Zyklen), da sie keine baumartige Struktur aufweisen.

Das Problem der Markov Random Fields stellen Zyklen dar, da in einem Feld weder Anfang noch Ende von Ketten vorhanden sind. Bei dem Beispiel der Soldaten zeigt sich dieses Problem, wenn sich die Soldaten in einem Kreis aufstellen und niemand weiß, wer anfangen soll zu zählen und wann aufgehört werden soll. Pearl (1988) schlug seinen Algorithmus aus Abschnitt 4.4.2 als Approximation für solche Loopy Graphs vor, da wie auch in baumartigen Modellen die Regeln für das Versenden von Nachrichten, (4.23) und (4.24), lokal und somit anwendbar sind. Für einige Modelle kann der Algorithmus konvergieren, es gibt allerdings keine Garantie dafür.

Auf Grund der Zyklen im Graph können wir die Nachrichten viele Male herumschicken, weswegen wir eine *Versendevorschrift* (engl. *Passing Schedule*) einführen, die angibt wann welche Nachrichten wohin verschickt werden. Diese Versendevorschrift enthält die Regeln, dass neue Nachrichten alte ersetzen und sich die neu ankommenden Nachrichten auch nur mit den zuletzt angekommenen zusammensetzen. Wie in Methode 1 im letzten Abschnitt (4.4.2) bereits erläutert, können wir Nachrichten nur dann verschicken, wenn am Knoten alle notwendigen Nachrichten eingegangen sind. Auf Grund der Zyklen ohne Wurzelknoten erfordert dieser Sachverhalt ein initiales Versenden von Nachrichten.

In dieser Arbeit verwenden wir einen iterativen Flooding Schedule, der zunächst alle Nachrichten an den Variablenknoten nach links verschickt, anschließend die Randwahrscheinlichkeiten an jedem Knoten neu berechnet, um danach die Nachrichten nach oben,

rechts und unten zu verschicken. Wir teilen das Markov Random Field somit in einzelne Zweierketten, die aus folgenden Mengen bestehen, wie in Abbildung 4.4 dargestellt:

$$\begin{aligned} \mathcal{N}(m) : \quad \mathcal{N}(1) &= \{1\}, & \mathcal{M}(n) : \quad \mathcal{M}(1) &= \{1,3\}, \\ \mathcal{N}(2) &= \{2\}, & \mathcal{M}(2) &= \{2,3\}. \end{aligned} \quad (4.27)$$

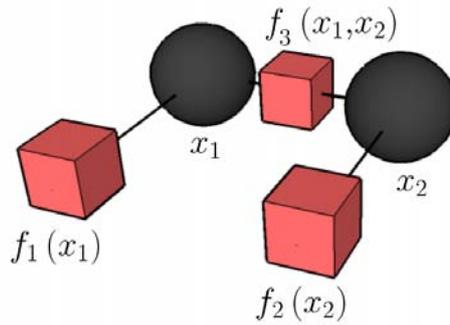


Abbildung 4.4: Der Faktorgraph, der in dieser Arbeit verwendeten Versandevorschrift, besteht aus Zweierketten.

Somit ergeben sich folgende Nachrichten, um die Randwahrscheinlichkeiten für den Knoten x_2 zu erhalten:

Vorwärtsnachricht:

$$r_{1 \rightarrow 1}(x_1 | \phi_1) = f_1(x_1 | \phi_1), \quad (4.28)$$

$$q_{1 \rightarrow 3}(x_1 | \phi_1) = r_{1 \rightarrow 1}(x_1 | \phi_1), \quad (4.29)$$

$$r_{3 \rightarrow 2}(x_2 | \phi_2) = \sum_{x_1} f_3(x_1, x_2 | \phi_1, \phi_2) \cdot q_{1 \rightarrow 3}(x_1 | \phi_1), \quad (4.30)$$

Rückwärtsnachricht:

$$r_{2 \rightarrow 2}(x_2 | \phi_2) = f_2(x_2 | \phi_2). \quad (4.31)$$

Durch Multiplikation der Vorwärts- und der Rückwärtsnachricht am Knoten x_2 erhalten wir die Randwahrscheinlichkeit

$$p(x_2 | \phi_2) = f_1(x_1 | \phi_1) f_2(x_2 | \phi_2) \sum_{x_1} f_3(x_1, x_2 | \phi_1, \phi_2). \quad (4.32)$$

Die Knoten x_1 und x_2 stehen stellvertretend für alle Anfangsvariablenknoten, von denen eine Nachricht ausgeht, und die Zielvariablenknoten, die die Nachrichten erhalten. Die Abbildungen 4.5a-4.5f zeigen anschaulich das gleichzeitige Verschicken der Nachrichten in einem Markov Random Field. Die eingefärbten Knoten sind diejenigen, bei denen aktuell eine Nachricht existiert, die anschließend in Pfeilrichtung verschickt wird. Zur besseren Veranschaulichung zeigt der dunkelrote Knoten den Weg einer Nachricht an. Abbildung 4.5d gibt die resultierende Vorwärtsnachricht und Abbildung 4.5f die resultierende Rückwärtsnachricht an, die wir miteinander multiplizieren, um die Randwahrscheinlichkeit am dunkelroten Knoten zu erhalten.

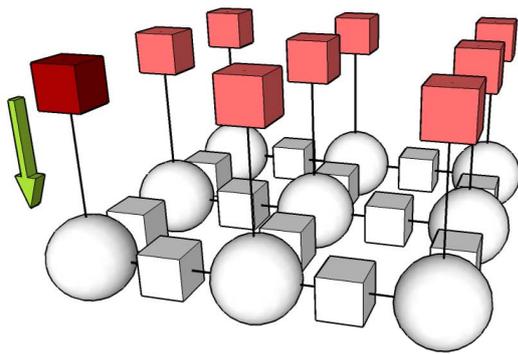
Die Abbildungen 4.5a-4.5f zeigen einen Teil der in dieser Arbeit verwendeten Versendevorschrift. Wir verschicken die eingegangenen Faktor-zu-Variablen Nachrichten an alle Variablenknoten, um sie dort mit den eingegangenen Faktor-zu-Variablen Nachrichten zu multiplizieren. Die vollständige Versendevorschrift enthält das Verschicken nach links, anschließend nach oben, dann nach rechts und nach unten. Das Verschicken in alle Richtungen verläuft äquivalent zu Abbildung 4.5a-4.5f.

Nach jeder Iteration bestimmen wir das Argument des Maximums der Randwahrscheinlichkeiten und erhalten das segmentierte Bild:

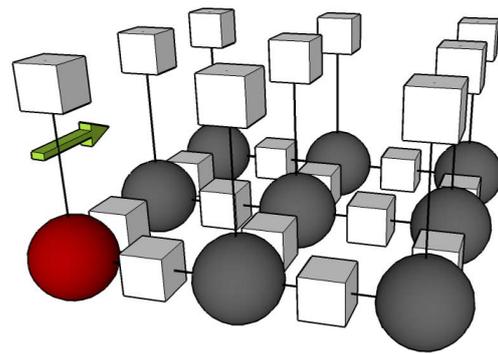
$$\mathcal{X} = \arg \max_{\mathcal{X}} p(\mathcal{X} | \Phi). \quad (4.33)$$

Das Verfahren führen wir solange aus, bis es gegen ein festes Labeling konvergiert.

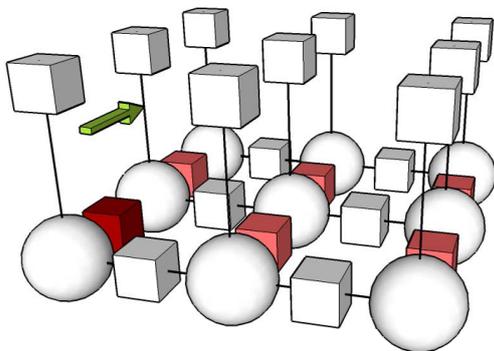
Mit Hilfe der letzten drei Kapitel können wir nun ein lineares probabilistisches diskriminatives Modell lernen, durch das wir die a posteriori Wahrscheinlichkeiten der Pixel für jede Klasse erhalten. Mit diesen Informationen können wir die Segmentierung des Bildes unter Verwendung der Markov Random Fields durchführen. Die folgenden Kapitel erläutern die Implementation der vorgestellten Verfahren in dem für diese Arbeit erstellten Programm und die damit erzielten Ergebnisse bei mehreren synthetischen und bei drei realen Datensätzen.



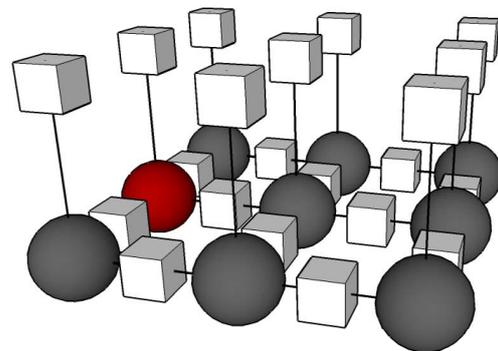
(a) Versenden der Nachrichten von den Faktorzur den Variablenknoten



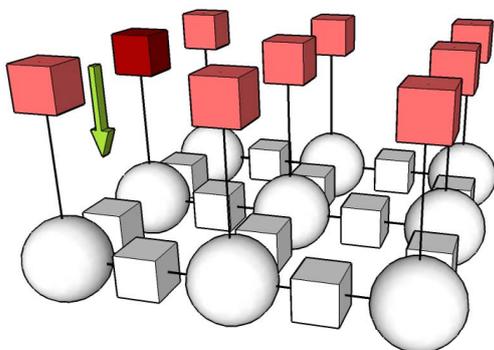
(b) Versenden der Nachrichten von den Variablen- zur den Faktorknoten



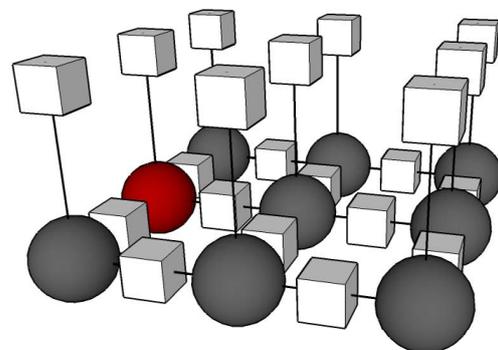
(c) Versenden der Nachrichten von den Faktorzur den Variablenknoten



(d) Resultierende Vorwärtsnachrichten



(e) Versenden der Nachrichten von den Faktorzur den Variablenknoten



(f) Resultierende Rückwärtsnachrichten

Abbildung 4.5: Hier dargestellt ist ein Teil der in dieser Arbeit verwendeten Versendevorschrift. Die eingefärbten Knoten sind diejenigen, bei denen aktuell eine Nachricht existiert, die anschließend in Pfeilrichtung verschickt wird. Der dunkelrote Knoten gibt den Weg einer Nachricht an. (a)-(d) demonstrieren das Verschicken, um die Vorwärtsnachrichten zu erhalten, und (e) und (f) das Verschicken, um die Rückwärtsnachrichten zu erhalten. Beide Nachrichten multiplizieren wir, um die Randwahrscheinlichkeit am dunkelroten Knoten aus (d) beziehungsweise (f) zu erhalten.

5 Umsetzung und Implementierung

Ein Ziel dieser Arbeit ist die Entwicklung eines Programms, welches als Eingabe einen Datensatz mit Trainings- und Testdaten erhält, mit den Trainingsdaten ein lineares probabilistisches diskriminatives Modell lernt, mit Hilfe von Markov Random Fields semantisch segmentiert und die Ergebnisse evaluiert.

Wir nutzen die Entwicklungsumgebung Matlab, da in ihr bereits viele Methoden der Linearen Algebra und der Bildverarbeitung zur Verfügung stehen. Es wurde mit der Version R2007a gearbeitet, in der erstmalig eine Bibliothek mit Basic Linear Algebra Subroutines (BLAS) enthalten ist. BLAS sind in Fortran geschriebene, für den Prozessor optimierte Routinen der Linearen Algebra, die auf Grund des optimalen Speicherzugriffs eine sehr schnelle Laufzeit haben. Dieses Kapitel beschreibt nun die Umsetzung der in den vorherigen Kapiteln beschriebenen Verfahren.

5.1 Allgemeines

Die folgenden Abschnitte erläutern zunächst die Optimierung aller Berechnungen durch die Verwendung von Blockmatrizen, anschließend die verwendete Ordnerstruktur, durch die der Benutzer leicht eigene Datensätze einbinden kann, und zuletzt das Initialisierungs- und das Hauptprogramm.

5.1.1 Aufteilung in Blockmatrizen

Die Umsetzung erfolgt auf Grund der großen Datenmengen mit Blockmatrizen, da schon Matrizenmultiplikationen mit diesen enormen Dimensionen nicht zu bewältigen sind. Ein Block entspricht einem Bild, sodass wir für jedes Bild i die Matrizen Φ_i , G_i und T_i verwenden. Die Bilder betrachten wir vektorisiert, sodass sich für diese Größen für alle Bilder \mathcal{I} zusammen die Dimensionen in Tabelle 5.1 ergeben.

K ist die Anzahl der Klassen, $N_R \cdot N_C = N$ die Anzahl der Pixel eines Bildes, I die Anzahl der Bilder und D die Dimension der erweiterten Merkmalsvektoren, die der Dimension M des Merkmalsraumes plus 1 entspricht.

Matrix	Dimensionen
Φ	$K \times (N_R \cdot N_C \cdot I)$
\mathbf{Y}	$1 \times (N_R \cdot N_C \cdot I)$
\mathbf{G}	$K \times (N_R \cdot N_C \cdot I)$
\mathbf{T}	$K \times (N_R \cdot N_C \cdot I)$
\mathbf{W}	$D \times K$

Tabelle 5.1: Die Dimensionen der Matrizen Φ , \mathbf{Y} , \mathbf{G} , \mathbf{T} und \mathbf{W} berechnen sich mit der Anzahl der Klassen K , der Anzahl der Pixel eines Bildes $N_R \times N_C = N$, der Anzahl der Bilder I und der Dimension der erweiterten Merkmalsvektoren D .

Das Prinzip der Blockmatrizen betrachten wir zunächst anhand der Berechnung der a posteriori Wahrscheinlichkeiten. Ein Bild i besteht aus den Pixeln $\{y_{ni}\}$, denen die Merkmalsvektoren ϕ_{ni} zugeordnet sind:

$$\mathbf{Y}_i = \left[y_{1i}, \dots, y_{ni}, \dots, y_{Ni} \right] \leftarrow \left[\phi_{1i}, \dots, \phi_{ni}, \dots, \phi_{Ni} \right]. \quad (5.1)$$

Ausgehend von Formel (2.24) können wir die Matrix der a posteriori Wahrscheinlichkeiten \mathbf{G} für jedes Bild i einzeln berechnen:

$$\left[\underbrace{\begin{bmatrix} g_{111} & \cdots & g_{1N1} \\ \vdots & \ddots & \vdots \\ g_{K11} & \cdots & g_{KN1} \end{bmatrix}}_{\text{Bild 1}} \cdots \underbrace{\begin{bmatrix} g_{11I} & \cdots & g_{1NI} \\ \vdots & \ddots & \vdots \\ g_{K1I} & \cdots & g_{KNI} \end{bmatrix}}_{\text{Bild I}} \right] =$$

$$f \left(\mathbf{W}^T \left(\underbrace{\left[\begin{bmatrix} 1 \\ \phi_{111} \\ \vdots \\ \phi_{M11} \end{bmatrix}, \dots, \begin{bmatrix} 1 \\ \phi_{1N1} \\ \vdots \\ \phi_{MN1} \end{bmatrix} \right]}_{\text{Bild 1}} \cdots \underbrace{\left[\begin{bmatrix} 1 \\ \phi_{11I} \\ \vdots \\ \phi_{M1I} \end{bmatrix}, \dots, \begin{bmatrix} 1 \\ \phi_{1NI} \\ \vdots \\ \phi_{MNI} \end{bmatrix} \right]}_{\text{Bild I}} \right) \right) \quad (5.2)$$

$$\left[\underbrace{\begin{bmatrix} \mathbf{G}_1 \end{bmatrix}}_{\text{Bild 1}} \cdots \underbrace{\begin{bmatrix} \mathbf{G}_I \end{bmatrix}}_{\text{Bild I}} \right] = f \left(\mathbf{W}^T \left[\underbrace{\begin{bmatrix} \Phi_1 \end{bmatrix}}_{\text{Bild 1}} \cdots \underbrace{\begin{bmatrix} \Phi_I \end{bmatrix}}_{\text{Bild I}} \right] \right). \quad (5.3)$$

Die a posteriori Wahrscheinlichkeiten \mathbf{G}_i für das Bild i können wir durch eine Transformation des Produkts der Parametermatrix \mathbf{W}^T und der Merkmalsvektoren Φ_i erhalten. Da die Parametermatrix keine problematische Dimension besitzt, müssen wir sie nicht in Blockmatrizen unterteilen. Wir speichern alle Blockmatrizen und laden sie wieder, wenn sie zur Berechnung benötigt werden.

5.1.2 Ordnerstruktur

In diesem Abschnitt erläutern wir die Ordnerstruktur. Diese dient einerseits dazu, weitere Datensätze einfach einzubinden und andererseits die Blöcke der Matrizen für einen schnellen Zugriff abzuspeichern, um sie bei Bedarf laden zu können. Alle Datensätze und Programme sind in folgender Ordnerstruktur gespeichert, wobei `%C%` die Klassenanzahl und `%Ds%` den Namen des Datensatzes angeben:

```

/
├─ data
│   ├── imageSets
│   │   └─ %C%_%Ds%
│   ├── ind-im-db
│   │   ├── images
│   │   │   └─ %C%_%Ds%
│   │   ├── annotations
│   │   │   └─ %C%_%Ds%
│   ├── ind-ft-db
│   │   ├── images
│   │   │   └─ %C%_%Ds%_%Merkmal%
│   │   ├── annotations
│   │   │   └─ %C%_%Ds%_%Merkmal%
│   │   ├── results
│   │   │   └─ %C%_%Ds%
│   │   └─ discriminant analysis
│   ├── matrices
│   │   └─ %C%_%Ds%
│   │       └─ orig
│   └─ %Source%

```

Die Bilder des Datensatzes sind in `\ind-im-db\images\C%Ds%` und die Annotationen in `\ind-im-db\annotations\C%Ds%` gespeichert. Die Annotationen sind die wahren gelabelten Bilder. Die Merkmale und die Superpixelmatrizen sind im Ordner `\ind-ft-db\images\C%Ds%\Merkmal%` im Unterordner mit dem Namen des Merkmals beziehungsweise mit dem Namen `superpixel matrices` gespeichert.

Der Ordner `\imageSets` enthält die `.txt` Dateien mit den Trainings- und den Testdaten. Sie enthalten zeilenweise die Strings der Dateinamen ohne Dateieindung.

Die Matrizen Φ_i , G_i und T_i für jedes Bild i sind im Ordner `\matrices\C%Ds%\orig` gespeichert. Wir kopieren Φ_i zu Beginn der Optimierung mit dem Gradientenabstiegsverfahren in den Ordner `\matrices\C%Ds%`, um damit G_i und T_i mit der aktuellen Parametermatrix W berechnen und in diesem Ordner abspeichern zu können. Die Matrizen laden, updaten und speichern wir erneut nach jeder Iteration.

Die Programmdateien liegen im Ordner `\%Source%`. Der Übersichtlichkeit halber sind sie in mehrere Unterordner unterteilt. Die zwei wichtigsten Dateien, die Hauptdatei zum Ausführen des Programms `run_classify.m` und die Initialisierungsdatei `init.m`, liegen im Hauptverzeichnis `\data`. Die folgenden zwei Abschnitte erläutern diese näher.

5.1.3 Initialisierung

Die Initialisierung erfolgt durch die Datei `init.m`. In dieser Datei befinden sich alle Stellgrößen für das Programm, die notwendigen Informationen über den Datensatz und die Pfade, die gegebenenfalls geändert werden können. Die eingegebenen Werte sind in einem globalen Struct `h` gespeichert, die das ganze Programm über die Informationen von Funktion zu Funktion weiterträgt. Tabelle 5.2 zählt die zu wählenden Elemente auf und erläutert deren Bedeutung.

Alle Pfade sind wie in Abschnitt 5.1.2 beschrieben angelegt, können jedoch wahlweise geändert werden. Nach der Initialisierung legt die Datei `mk_folders.m` alle gewählten Pfade an.

Element	Datentyp	Beschreibung
<code>h.s_dbIn</code>	String	Pfad zur Datenbank der Bilder (<code>\ind-im-db</code>)
<code>h.s_dbOut</code>	String	Pfad zur Datenbank der Merkmale (<code>\ind-ft-db</code>)
<code>h.s_is</code>	String	Pfad zu <code>train.txt</code> und <code>val.txt</code>
<code>h.s_dsIn</code>	String	Eingabename des Datensatzes
<code>h.s_dsOut</code>	String	Ausgabename des Datensatzes
<code>h.s_imftypeIn</code>	String	Dateiendung der Bilder
<code>h.s_annoftypeIn</code>	String	Dateiendung der Annotationen
<code>h.s_imftypeOut</code>	String	Dateiendung der zu speichernden Bilder
<code>h.s_annoftypeOut</code>	String	Dateiendung der zu speichernden Annotationen
<code>h.input</code>	String	Pfad zu den Blockmatrizen (<code>\matrices</code>)
<code>h.c_learningRate</code>	Double	Schrittweite des Gradientenabstiegsverfahrens
<code>h.c_ftBlSz</code>	Integer	Größe einer Rechteckumgebung
<code>h.c_ftSpSz</code>	Integer	durchschnittliche Größe eines Superpixels
<code>h.c_Nrep</code>	Integer	Ausdünnungsfaktor (jeder wievielte Pixel)
<code>h.cs_ft_b</code>	Cellstring	verwendete Merkmale mit Rechteckumgebungen
<code>h.cs_ft_s</code>	Cellstring	verwendete Merkmale mit Superpixeln
<code>h.c_b</code>	Integer	Parameter β des Markov Random Fields
<code>h.c_eta</code>	Integer	Parameter η des Markov Random Fields
<code>h.classes</code>	Cellstring	Klassen des Datensatzes
<code>h.C</code>	Integer	Anzahl der Klassen

Tabelle 5.2: Die Elemente der globalen Variablen `h` bestehen aus allen Stellgrößen für das Programm, den notwendigen Informationen über den Datensatz und den Pfaden, die gegebenenfalls geändert werden können.

5.1.4 Hauptprogramm

Das Hauptprogramm `run_classify.m` durchläuft mehrere Funktionen, von denen einige je nach Bedarf auskommentiert werden können. Zu den Funktionen gehören

- `s_ex()`
Diese Funktion extrahiert die Superpixel aus jedem Bild und speichert sie im Ordner `\ind-ft-db\images\%C%_Ds%\superpixel matrices` ab.
- `ft_ex()`
Diese Funktion extrahiert die Merkmale aus jedem Bild und speichert sie in den Ordnern `\ind-ft-db\images\%C%_Ds%\%C%_Ds%_Merkmal%` ab.
- `getDataMatrices()`
Diese Funktion erstellt die Matrizen Φ_i , \mathbf{Y}_i und \mathbf{T}_i für jedes Bild i und speichert sie in dem Ordner `\matrices\%C%_Ds%` ab.
- `h.w = learnDiscriminants()`
Diese Funktion lernt die Diskriminanten und speichert die Parametermatrix \mathbf{W} in dem globalen Struct `h`.
- `classify()`
Diese Funktion klassifiziert die Testbilder mit der Parametermatrix \mathbf{W} und speichert die Ergebnisbilder in den Ordner `\ind-ft-db\results\%C%_Ds%`.
- `evl()`
Diese Funktion evaluiert die Ergebnisse, erstellt eine Konfusionsmatrix und speichert diese in einer `.txt` Datei im Ordner `\ind-ft-db\results\%C%_Ds%` ab.
- `sumProduct()`
Diese Funktion verschiebt die klassifizierten Bilder in den Ordner `\ind-ft-db\results\%C%_Ds%\discriminant analysis`, wendet die Loopy Belief Propagation auf diese an, und speichert die Ergebnisbilder im Ordner `\ind-ft-db\results\%C%_Ds%` ab.

5.1.5 Arbeitsfluss

Das Schema im Anhang in Abbildung B.1 zeigt den Arbeitsfluss des Programms, in denen die eben genannten Funktionen und weitere Subroutinen eingebettet sind. Der Arbeitsfluss besteht aus den Unterprogrammen der Vorverarbeitung, der Merkmalsextraktion, der Datenaufbereitung, der Trainingsphase, der Segmentierung und der Evaluierung. Die blauen und braunen Felder sind weitere, speziellere Unterprogramme, die grünen Felder geben die Verwendung des Trainingsdatensatzes (train) oder des Testdatensatzes (test) an und die violetten Felder enthalten den verwendeten Quellcode.

Die folgenden Abschnitte erläutern die wichtigsten Programme des Arbeitsflusses und die darin enthaltenen Algorithmen in Pseudocode, wie sie in der Arbeit implementiert sind.

5.2 Superpixel- und Merkmalsextraktion

Der erste Schritt nach der Initialisierung ist die Extraktion der Superpixel und der Merkmale mit den Funktionen `s_ex()` und `ft_ex()`. Diese speichern die Dateien im Ordner `\ind-ft-db\images\%C%\%Ds%` ab, sodass wir die Dateien nur einmal erstellen müssen und sie bei erneuter Ausführung des Hauptprogramms auskommentieren können.

Für die Superpixelextraktion verkleinern wir die Bilder auf eine maximale Pixelanzahl von $N_P = 10000$. Dies ist ein guter Kompromiss zwischen Laufzeit und Genauigkeit des Verfahrens. Zusätzlich transformieren wir die Bilder in Grauwertbilder, da dies das NCut-Programm nach Shi und Malik (2000) als Eingabe verlangt. Die Matrix, welche alle Superpixel enthält, ist indiziert, das heißt die Superpixel sind beginnend mit 1 fortlaufend nummeriert mit je einer Nummer für alle Pixel in einem Superpixel. Die Matrix vergrößern wir auf die Originalgröße des Bildes mit der Nearest-Neighbor Interpolation und speichern sie als `.mat` Datei ab (Pseudocode 5.1).

Die Merkmalsextraktion durch die Funktion `ft_ex()` erfolgt für alle Bilder durch die Merkmalsfunktionen, die in den Ordnern `\src feature extraction\%Merkmal%` liegen (Pseudocode 5.2). Die Funktion ruft die Merkmalsfunktionen auf, die in der Funktion `init()` nicht auskommentiert sind, und wendet sie auf jedes Bild an. Die Merkmalsfunktionen, die mit Superpixeln arbeiten, rufen die dazugehörige Superpixelmatrix auf und wei-

Funktion 5.1 : Superpixelextraktion(s_ex.m)

Daten : Bilder \mathcal{I} , durchschnittliche Größe der Superpixel A_{sp} **Ergebnis** : Indizierte Superpixelmatrizen $\{S_i\}$ **Beginn** **für alle** Bilder Y_i mit $i \in \mathcal{I}$ **tue** Lies i-tes Bild Y_i ein; Verkleinere Bild Y_i auf maximale Pixelanzahl $N_P = 10000$; Konvertiere Bild Y_i in Grauwertbild $Y_{gray,i}$; Bestimme Anzahl der Segmente N_S mit A_{sp} ; Segmentiere mit NCut()-Algorithmus: $S_i = \text{NCut}(Y_{gray,i})$; Vergrößere S_i mit Nearest-Neighbor Interpolation auf Originalgröße von Y_i ; **Speichere** Indizierte Superpixelmatrix S_i ;**Ende**

sen allen Pixeln in einem Superpixel dasselbe Merkmal zu. Die entstehenden Merkmalsmatrizen speichert die Funktion im Ordner `\ind-ft-db\images\%C%_%Ds%_%Merkmal%` ab.

Beide Funktionen speichern die Matrizen in der Größe der Bilder ab, sodass die Superpixelmatrizen die Größe $N_R \times N_C \times 1$ und die Merkmalsmatrizen die Größe $N_R \times N_C \times N_{\phi_f}$ haben. Die Merkmalsvektoren des f-ten Merkmals mit der Länge N_{ϕ_f} befinden sich in der dritten Dimension. Im weiteren Verlauf verwenden wir die Merkmale in der Matrix Φ_i vektorisiert.

Der folgende Abschnitt erläutert Möglichkeiten der Datenreduktion, da wir auf Grund des enormen Datenaufkommens nicht alle Merkmalsvektoren verwenden können. Das Gradientenabstiegsverfahren zum Beispiel benötigt bei der Benutzung aller Merkmalsvektoren des Corel Datensatzes mehrere Minuten für eine Iteration. Das Verfahren selbst konvergiert sehr langsam, sodass wir mit bis zu 1000 Iterationen rechnen müssen. Aus diesem Grund und wegen der schlechten Konvergenzeigenschaften auf Grund der Beschaffenheit der Fehlerfläche (siehe Abschnitt 6.2.3) bei zu vielen Daten reduzieren wir die Daten durch eine Ausdünnung.

5.3 Datenreduktion

Die Datenreduktion verwenden wir für die Ausdünnung der Trainingsdaten, da wir für das Trainieren des linearen probabilistischen diskriminativen Modells nicht notwendiger-

Funktion 5.2 : Merkmalsextraktion(ft_ex.m)

Daten : Bilder \mathcal{I} , Merkmale $\{f_i\}$ bestehend aus Rechteckmerkmalen $\{f_{bi}\}$ und Superpixelmerkmalen $\{f_{si}\}$

Ergebnis : Merkmalsmatrizen $\{\Phi_{bi}\}$ der Rechteckmerkmale und $\{\Phi_{si}\}$ der Superpixelmerkmale

Beginn

```

%% Extraktion der Merkmale aus Rechteckumgebung
für alle Merkmale  $\in \{f_{bi}\}$  tue
  für alle Bilder  $\mathbf{Y}_i$  mit  $i \in \mathcal{I}$  tue
    Lies  $i$ -tes Bild  $\mathbf{Y}_i$  ein;
    Rufe Funktion für Extraktion des Merkmals auf;
    Extrahiere Merkmal  $f_{bi}$ ;
    Speichere Merkmalsmatrix  $\Phi_{bi}$ ;

%% Extraktion der Merkmale aus Superpixelumgebung
für alle Merkmale  $\in \{f_{si}\}$  tue
  für alle Bilder  $\mathbf{Y}_i$  mit  $i \in \mathcal{I}$  tue
    Lies  $i$ -tes Bild  $\mathbf{Y}_i$  ein;
    Lies  $i$ -te Superpixelmatrix  $\mathbf{S}_i$  ein;
    Rufe Funktion für Extraktion des Merkmals auf;
    Extrahiere Merkmal  $f_{si}$ ;
    Speichere Merkmalsmatrix  $\Phi_{si}$ ;

```

Ende

weise alle Daten verwenden müssen. Die Reduktion ist in der Funktion `GetDataMatrices()` implementiert (Pseudocode 5.3).

Ab einer gewissen Anzahl geht die Verwendung von vielen Daten zu Lasten der Laufzeit, ohne bessere Ergebnisse zu liefern. Es gilt somit, einen Kompromiss zwischen Laufzeit und Trainierbarkeit des Modells zu finden. Im Falle von Bildern als Trainingsdaten sind in der Regel genug Daten vorhanden, sodass wir die Datenreduktion problemlos anwenden können.

Für die Datenreduktion lesen wir zunächst die Matrix Φ_i des i -ten Bildes ein. Wir dünnen anschließend alle Punkte einer Klasse im Merkmalsraum aus, indem wir repräsentative Punkte suchen. Die Anzahl der repräsentativen Punkte ist

$$N_{rep} = \frac{N_k}{N^*}. \quad (5.4)$$

Diese unterteilen wir in

$$N_{mu} = \frac{1}{20} \cdot N_{rep}, \quad (5.5)$$

$$N_{rand} = \frac{19}{20} \cdot N_{rep}. \quad (5.6)$$

Von der Gesamtanzahl N_k aller Merkmalsvektoren einer Klasse wählen wir jeden n^* -sten Merkmalsvektor und erhalten die Gesamtanzahl der repräsentativen Merkmalsvektoren N_{rep} . Diese Anzahl untergliedert sich in die zufällig ausgewählten Merkmalsvektoren mit der Anzahl N_{rand} und der Anzahl, wie oft wir den Mittelpunkt N_{mu} verwenden. Die jeweilige Anzahl der Merkmalsvektoren fungiert als Gewicht, da der Mittelpunkt der Merkmale höher gewichtet sein soll, als die zufällig ausgewählten Merkmalsvektoren.

Durch dieses Verfahren ist gewährleistet, dass die Anzahl der repräsentativen Merkmalsvektoren abhängig von der Gesamtanzahl N_k in jeder Klasse ist und dass wir aus allen Bildern für jede Klasse repräsentative Merkmalsvektoren verwenden.

Funktion 5.3 : Bestimmung repräsentativer Punkte(getRepPoints.m)

Daten : Merkmalsmatrizen $\{\Phi_i\}$, Wert, jeder wievielte Pixel verwendet wird n^*

Ergebnis : Ausgedünnte Merkmalsmatrizen $\{\Phi_{rep,i}\}$

Beginn

```

für alle Merkmalsmatrizen  $\Phi_i$  mit  $i \in \mathcal{I}_{Train}$  tue
  Lade  $\Phi_i$ ;
  Initialisiere  $\{\Phi_{rep,i}\} \leftarrow []$ ;
  für alle  $\Phi_{ki}$  mit  $k \in \mathcal{C}$  und  $i \in \mathcal{I}$  tue
    Bestimme Anzahl der Merkmalsvektoren in  $\Phi_{ki}$ ;
    Berechne  $N_{rep}, N_{mu}$  und  $N_{rand}$ ;
    Berechne Mittelpunkt der Merkmalsvektoren in  $\Phi_{ki}$ ;
    Schreibe diesen  $N_{mu}$  mal in  $\Phi_{rep,i}$ ;
    Bestimme  $N_{rand}$  zufällige Punkte aus  $\Phi_{ki}$ ;
    Schreibe diese in  $\Phi_{rep,i}$ ;
  zurück  $\Phi_{ki}$ 
  Füge alle  $\Phi_{ki}$  aneinander;
Speichere  $\Phi_i$ ;

```

Ende

5.4 Trainieren der Diskriminantenparameter

Die Diskriminantenparameter lernen wir mit der logistischen Regression und dem Gradientenabstiegsverfahren, wie in Kapitel 2.4 beschrieben:

$$\widehat{\mathbf{W}} = \arg \min_{\mathbf{W}} F(\mathbf{W}|\mathbf{T}). \quad (5.7)$$

Die optimalen Parameter sind demnach die, bei denen die Fehlerfunktion F minimal ist. Für die Bestimmung der Parameter verwenden wir das Programm `learnDiscriminants()` (Pseudocode 5.4).

Die Initialisierung der Parameter \mathbf{W} für das Gradientenabstiegsverfahren erfolgt nach Alpaydin (2008, Kapitel 10, Seite 222) mit zufälligen Werten im Intervall $[-0,01, 0,01]$. Das garantiert, dass die Aktivierungen einen kleinen Wert erhalten und die Sigmoidfunktion nicht gesättigt ist. Dies macht Abbildung 2.2 deutlich. Liegen die Aktivierungen a_{nk} in dem Bereich $[-5,5]$, so liegen die Wahrscheinlichkeiten nahe 0,5 und die Parameter können aktualisiert werden. Liegen die Aktivierungen jedoch in hohen Größenordnungen, so ist die Ableitung der Sigmoidfunktion nahe Null und es können keine Updates an die Parameter angebracht werden.

Das Verfahren wird nach 800 Iterationen abgebrochen, da wir nicht garantieren können, dass es konvergiert und automatisch abbricht. Den Normalisierungsfaktor $\frac{10000}{N \cdot I}$ wählen wir, damit die Schrittweite unabhängig von der Anzahl aller Datenpunkte $N \cdot I$ ist und eine numerische Stabilität auf Grund der sehr kleinen Zahlen gewährleistet ist.

Mit dem synthetischen Datensatz testen wir einen weiteren Ansatz, bei dem die Schrittweite automatisch gewählt wird. Dieser Gradientenabstieg ist unter dem Exact Line Search bekannt (Pseudocode 5.5). Anders als bei dem oben beschriebenen Algorithmus testet dieser verschiedene Schrittweiten bevor er sie ausführt. Er wählt diejenige Schrittweite, bei der die resultierende Kreuzentropie am kleinsten ist. In dieser Arbeit verwenden wir die Schrittweiten $s = \{10^{-8}, 10^{-7}, \dots, 10^{-1}, \frac{1}{2} \cdot 10^{-8}, \frac{1}{2} \cdot 10^{-7}, \dots, \frac{1}{2} \cdot 10^{-1}\}$.

Funktion 5.4 : Gradientenabstiegsverfahren(learnDiscriminants.m)

Daten : Parameter \mathbf{W} , Schwellwert ν , Schrittweite s , Merkmalsmatrizen $\{\Phi_i\}$, a posteriori Wahrscheinlichkeiten $\{G_i\}$, wahre Targetmatrizen $\{\tilde{T}_i\}$

Ergebnis : Parameter \mathbf{W}

Beginn

```

solange  $|s\nabla F(\mathbf{W})| > \nu$  oder  $Iteration < 800$  tue
  für alle Bilder  $\mathbf{Y}_i$  mit  $i \in \mathcal{I}_{Train}$  tue
    Lade  $\Phi_i, G_i, \tilde{T}_i$ ;
    Berechne Ableitungen der Fehlerfunktion:  $\nabla F_i(\mathbf{W})$ ;
    Berechne Schrittrichtung: Summe über alle  $\nabla F_i(\mathbf{W})$ ;
    Normalisiere Schrittrichtung:  $\nabla F(\mathbf{W}) \leftarrow \frac{10000}{N \cdot I} \cdot \nabla F(\mathbf{W})$ ;
    Wähle Schrittweite  $s$ ;
    Berechne Update:  $\mathbf{W} \leftarrow \mathbf{W} - s\nabla F(\mathbf{W})$ ;
  zurück  $\mathbf{W}$ 
  für alle Bilder  $\mathbf{Y}_i$  mit  $i \in \mathcal{I}_{Train}$  tue
    Lade  $\Phi_i$ ;
    Berechne neue a posteriori Wahrscheinlichkeiten:
     $G_i \leftarrow \text{Sigmoid}(\mathbf{W}^T \Phi_i)$ ;
    Speichere  $G_i$ ;

```

Ende

5.5 Maximum a posteriori Schätzung mit dem linearen probabilistischen diskriminativen Modell

Nachdem wir die Parameter gelernt haben, können wir mit Hilfe der Funktion `classify()` die a posteriori Wahrscheinlichkeiten für die Verwendung der Markov Random Fields bereitstellen und die Testbilder klassifizieren (Pseudocode 5.6). Wir verwenden die Formel

$$p(\mathcal{C}|\Phi) = \mathbf{G}(\mathbf{Y}) = \sigma(\mathbf{W}^T \Phi). \quad (5.8)$$

Die Klassifikation erfolgt durch die Bestimmung des Arguments des Maximums der a posteriori Wahrscheinlichkeiten. Durch Umordnen auf die Originalgröße des Bildes entsteht ein gelabeltes Bild, welches wir im Ordner `\ind-ft-db\results\%C%_Ds%` abspeichern.

Funktion 5.5 : Exact Line Search(P_exactLineSearch.m)**Daten** : Parameter \mathbf{W} , Schwellwert ν , Schrittweiten $\{s_j\}$ **Ergebnis** : Parameter \mathbf{W} **Beginn**

```

solange  $|s\nabla F(\mathbf{W})| > \nu$  oder  $Iteration < 800$  tue
  für alle Bilder  $\mathbf{Y}_i$  mit  $i \in \mathcal{I}_{Train}$  tue
    Lade  $\Phi_i, \mathbf{G}_i, \tilde{\mathbf{T}}$ ;
    Berechne Ableitungen der Fehlerfunktion:  $\nabla F_i(\mathbf{W})$ ;
    Berechne Schrittrichtung: Summe über alle  $\nabla F_i(\mathbf{W})$ ;
    Normalisiere Schrittrichtung:  $\nabla F(\mathbf{W}) \leftarrow \frac{10000}{N \cdot I} \cdot \nabla F(\mathbf{W})$ ;
    für alle  $s_j \in \{s\}$  tue
      Wähle Schrittweite:  $s_j$ ;
      Berechne Update:  $\mathbf{W} \leftarrow \mathbf{W} - s_j \nabla F(\mathbf{W})$ ;
      Berechne Kreuzentropie  $F_j$ ;
    Wähle  $s_j$  zu kleinstem  $F_j$ ;
    Wähle Update zur Schrittweite  $s_j$ ;
    zurück  $\mathbf{W}$  für alle Bilder  $\mathbf{Y}_i$  mit  $i \in \mathcal{I}_{Train}$  tue
      Lade  $\Phi_i$ ;
      Berechne neue a posteriori Wahrscheinlichkeiten:
       $\mathbf{G}_i \leftarrow \text{Sigmoid}(\mathbf{W}^T \Phi_i)$ ;
      Speichere  $\mathbf{G}_i$ ;

```

Ende

5.6 Segmentierung mit Markov Random Fields

Nachdem die a posteriori Wahrscheinlichkeiten der Pixel für jede Klasse bestimmt sind, können wir nun diese in das Markov Random Field einbinden und die Bildsegmentierung durchführen. Wir verwenden die Loopy Belief Propagation mit dem in Abschnitt 4.4.3 beschriebenen Verfahren (Pseudocode 5.7).

Durch das Drehen des Bildes um 90° können wir immer denselben Teilalgorithmus des Verschiebens nach links anwenden.

5.7 Evaluierung

Um die Leistungsfähigkeit des Klassifikators zu testen, betrachten wir mehrere Evaluierungen. Zum einen betrachten wir die Auswirkungen verschiedener Schrittweite an-

Funktion 5.6 : Bestimmung der a posteriori Wahrscheinlichkeiten und Klassifikation mit linearem probabilistischem diskriminativem Modell(classify.m)

Daten : Parametermatrix \mathbf{W} , Merkmalsmatrizen $\{\Phi_i\}$

Ergebnis : Klassifizierte Bilder $\{\mathbf{X}_i\}$, a posteriori Wahrscheinlichkeiten $\{\mathbf{G}_i\}$

Beginn

für alle $\mathbf{Y}_i \in \mathcal{I}_{Test}$ **tue**

 Lade Φ_i ;

 Bestimme a posteriori Wahrscheinlichkeiten \mathbf{G}_i :

$\mathbf{G}(\mathbf{Y}_i) = \text{Sigmoid}(\mathbf{W}^T \Phi(\mathbf{Y}_i))$;

Speichere A posteriori Wahrscheinlichkeiten \mathbf{G}_i ;

 Bestimme Klassenzugehörigkeiten: $\mathbf{T}_i \leftarrow \mathbf{X}_i = \arg \max_{\mathbf{x}_i} \mathbf{G}_i$;

 Ordne \mathbf{X}_i auf Größe $N_{Ri} \times N_{Ci}$ um;

Speichere Klassifiziertes Bild \mathbf{X}_i ;

Ende

hand des Klassifikationserfolges. Desweiteren variieren wir die Verwendung verschiedener Merkmale und die Anzahl der verwendeten Pixel. All diese Betrachtungen benötigen die Berechnungen des Klassifikationserfolges KE. Dieser setzt die Anzahl der richtig zugeordneten Pixel zu der Gesamtanzahl aller klassifizierten Pixel ins Verhältnis:

$$\text{KE} = \frac{\sum \text{richtig zugeordnete Pixel}}{\sum \text{klassifizierte Pixel}}. \quad (5.9)$$

Wir betrachten zusätzlich einen detaillierten Klassifikationserfolg, indem wir so genannte *Konfusionsmatrizen* \mathbf{K} erstellen. Diese geben den Klassifikationserfolg jeder Klasse an und zeigen auf, welche Klasse prozentual als was klassifiziert wurde. In ihr stehen die relativen Häufigkeiten aller möglichen Kombinationen von ermittelter Klasse und tatsächlicher Klasse. Die Zeilen sind mit ihrer Summe normiert und die Angaben in Prozent. Die Prozente der richtig klassifizierten Pixel befinden sich auf der Diagonalen und die der falsch zugeordneten in den restlichen Zellen der Matrix. Die Berechnung eines Eintrages erfolgt durch die Anzahl aller Pixel, die der Klasse \mathcal{C}_k zugeordnet wurden, wenn $\tilde{x} = \mathcal{C}_j$ gilt, normiert mit der Summe aller als \mathcal{C}_k klassifizierten Pixel:

$$K_{kj} = \frac{\sum (x = k, \tilde{x} = j)}{\sum (x = k)}. \quad (5.10)$$

Die Evaluierung erfolgt durch die Funktion `evl()` (siehe Pseudocode 5.8).

Die Konfusionsmatrix speichern wir im Ordner `\ind-ft-db\results\%C%_%Ds%` in einer `.txt` Datei. Diese Datei enthält zusätzlich die verwendeten Merkmale, die Klassenbezeichnungen, die absoluten akkumulierten Werte und den Klassifikationserfolg.

Dieses Kapitel stellte die wichtigsten Algorithmen in Pseudocode zusammen. Das folgende Kapitel diskutiert die Ergebnisse, die mit diesen erreicht werden konnten.

Funktion 5.7 : Segmentierung mit Loopy Belief Propagation(SumProduct.m)**Daten** : A posteriori Wahrscheinlichkeiten $\{\mathbf{G}_i\}$, Parameter $\boldsymbol{\theta} = \{\beta, \eta\}$ **Ergebnis** : A posteriori Wahrscheinlichkeiten $\{\mathbf{G}_i\}$, segmentierte Bilder $\{\mathbf{X}_i\}$ **Beginn**Initialisiere: Parameter $\boldsymbol{\theta} = \{\beta, \eta\}$;Initialisiere: Schwellwert $\nu = 1$;**für alle** $\mathbf{X}_i \in \mathcal{I}_{Test}$ **tue** Initialisiere: $\hat{\mathbf{X}}_i = -\mathbf{X}_i$; Lade a posteriori Wahrscheinlichkeiten \mathbf{G}_i ; **solange** $\frac{1}{\#Pixel\ in\ \mathbf{X}_i} \cdot \sum_x \delta(\mathbf{X}_i, \hat{\mathbf{X}}_i) < \nu$ **oder** *Iteration* < 20 **tue** $\hat{\mathbf{X}}_i = \mathbf{X}_i$;

%% Nachrichten nach links verschicken

 Bestimme Werte für Faktorknoten mit gegebenen Parametern $\boldsymbol{\theta} = \{\beta, \eta\}$; Berechne und normalisiere Randverteilung: $\mathbf{G}_i \leftarrow p(\mathbf{X}_i | \Phi_i)$; **zurück** A posteriori Wahrscheinlichkeiten \mathbf{G}_i ;

%% Nachrichten nach oben verschicken

 Drehe \mathbf{G}_i um 90° im Uhrzeigersinn;

Wiederhole Verschicken der Nachrichten nach links;

zurück A posteriori Wahrscheinlichkeiten \mathbf{G}_i ;

%% Nachrichten nach rechts verschicken

 Drehe \mathbf{G}_i um 90° im Uhrzeigersinn;

Wiederhole Verschicken der Nachrichten nach links;

zurück A posteriori Wahrscheinlichkeiten \mathbf{G}_i ;

%% Nachrichten nach unten verschicken

 Drehe \mathbf{G}_i um 90° im Uhrzeigersinn;

Wiederhole Verschicken der Nachrichten nach links;

zurück A posteriori Wahrscheinlichkeiten \mathbf{G}_i ;

%% Drehe a posteriori Wahrscheinlichkeiten in Ausgangslage

zurück Drehe \mathbf{G}_i um 90° im Uhrzeigersinn; **zurück** A posteriori Wahrscheinlichkeiten \mathbf{G}_i ; Bestimme Klassenzugehörigkeiten: $\mathbf{T}_i \leftarrow \mathbf{X}_i = \arg \max_{\mathbf{X}} \mathbf{G}_i$; Ordne \mathbf{X}_i auf Größe $N_{ri} \times N_{ci}$ um; **Speichere** Segmentiertes Bild \mathbf{X}_i ;**Ende**

Funktion 5.8 : Evaluierung der Testbilder(evl.m)

Daten : Klassifizierte Bilder $\{\mathbf{X}_i\}$, Annotierte Bilder $\{\widetilde{\mathbf{X}}_i\}$

Ergebnis : Konfusionsmatrizen $\{\mathbf{K}\}$

Beginn

Initialisierung: $\mathbf{K} \leftarrow 0$;

für alle \mathbf{X}_i mit $i \in \mathcal{I}_{Test}$ **tue**

 Lade klassifiziertes Bild $\widetilde{\mathbf{X}}_i$;

 Lade annotiertes Bild $\widetilde{\mathbf{X}}_i$;

für alle $x_{ni} \in \mathcal{X}_i$ mit $i \in \mathcal{I}_{Test}$ **tue**

wenn x_{ni} gleich \tilde{x}_{ni} **dann**

 Akkumuliere 1 zum x-ten Eintrag auf der Hauptdiagonalen der
 Konfusionsmatrix \mathbf{K}

sonst

 Akkumuliere 1 an der Stelle (x, \tilde{x}) der Konfusionsmatrix \mathbf{K}

Normalisiere \mathbf{K} durch Dividieren jeder Zeile durch die Summe der Zeile und
wandle in Prozent durch Multiplikation mit 100 um;

Speichere Konfusionsmatrix \mathbf{K} ;

Ende

6 Ergebnisse

Das in dieser Arbeit gelernte Modell wenden wir zum einen auf einen synthetisch generierten Datensatz im zweidimensionalen Raum und zum anderen auf drei im Bereich Computer Vision weit verbreitete Datensätze an, die Bilder realer Objekte beinhalten: (i) der MSRC 23-Klassen Datensatz, (ii) der 7-Klassen Corel Datensatz und (iii) der 7-Klassen Sowerby Datensatz. Anhand der synthetischen Daten können wir sehr gut erkennen, wie die Diskriminanten geschätzt wurden, die a posteriori Wahrscheinlichkeiten verteilt sind und wie die Diskriminanten bei nicht separierbaren Daten im Merkmalsraum liegen. Nach dieser Betrachtung wenden wir das Modell auf Datensätze mit realen Objekten an und evaluieren das Verfahren durch Erstellung einer Konfusionsmatrix.

Die folgenden Abschnitte beurteilen die in dieser Arbeit gewählten Verfahren, die Merkmale, die Datensätze und diskutieren die erzielten Ergebnisse. Alle Ergebnisse beziehen sich auf die in Kapitel 5 beschriebene Implementation in Matlab Version R2007a, ausgeführt auf einem 1,73 GHz Intel Pentium M Prozessor mit 1,0 GB RAM und dem 32-Bit-Betriebssystem Microsoft Windows Vista.

6.1 Anwendung auf die synthetischen Datensätze

Die Datensätze bestehen aus einem beliebigen synthetisch erzeugten Merkmal im zweidimensionalen Raum. Die Anzahl der Klassen beträgt $K = \{2, \dots, 5\}$ und die Merkmale sind entweder linear separierbar oder nicht. Die Generierung der synthetischen Daten erfolgte durch Verrauschen der Merkmale um K definierte Mittelpunkte.

6.1.1 Linear separierbare Datensätze

Zunächst betrachten wir den Fall linear separierbarer Daten mit drei Klassen.

Abbildung 6.1a zeigt den synthetisch generierten Datensatz. Abbildung 6.1b und 6.1c zeigen zwei Ergebnisse für die geschätzten Diskriminanten bei unterschiedlichem Abbruchkriterium des Gradientenabstiegsverfahrens. In diesem Fall wurde der Exact Line Search verwendet, da dieser durch die geringe Datenanzahl am schnellsten zu einem Er-

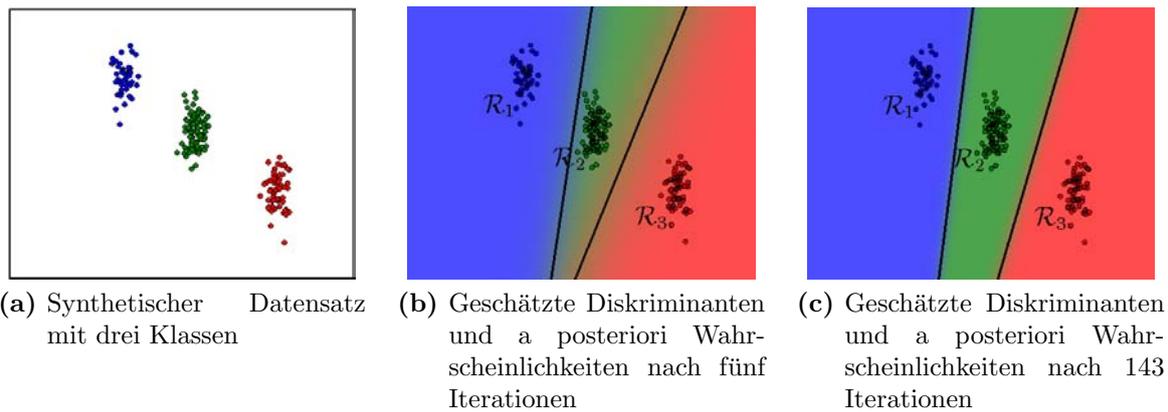


Abbildung 6.1: Der synthetische Datensatz mit drei Klassen liefert nach fünf Iterationen das Ergebnis aus (b), bei dem alle Merkmale richtig zugeordnet sind und nach 155 Iterationen das Ergebnis aus (c), bei dem die Kreuzentropie kleiner 10^{-2} ist.

gebnis gelangt. Der Vorteil besteht darin, dass der Algorithmus die optimale Schrittweite in jeder Iteration selbst bestimmt und wir diese Stellgröße nicht manuell wählen müssen.

Abbildung 6.1b zeigt das Ergebnis nach fünf Iterationen, dann wenn die geschätzten Targets mit den wahren übereinstimmen, also sobald alle Merkmale richtig zugeordnet wurden. Abbildung 6.1c zeigt das Ergebnis nach 143 Iterationen, wenn die Kreuzentropie kleiner 10^{-2} ist. An diesem Beispiel sieht man deutlich den Unterschied zwischen der Klassifikation mit Diskriminantenfunktionen und den diskriminativen Modellen. Letztere optimieren die Lage der Diskriminanten durch Minimierung der Kreuzentropie und somit der Maximierung der a posteriori Wahrscheinlichkeiten, während es bei der ersten Methode genügt, dass alle Merkmale richtig zugeordnet sind. Es ist jedoch auch ersichtlich, dass der Rechenaufwand durch den Gradientenabstieg sehr groß ist, da bereits bei einem separierbaren, niedrig dimensionalen, einfachen Datensatz 143 Iterationen benötigt werden, um die optimale Lage der Diskriminanten zu bestimmen.

Abbildung 6.2a zeigt den synthetischen Datensatz mit fünf Klassen. An diesem Beispiel wird erneut die Schwierigkeit deutlich, das Minimum der Fehlerfläche zu finden. Das Verfahren benötigt mit dem Exact Line Search 344 Iterationen, um das erste Mal alle Merkmale richtig zuzuordnen. Obwohl die Kreuzentropie in jeder Iteration kleiner wird, kann es passieren, dass die Anzahl der falsch zugeordneten Merkmale erneut über 0 steigt. Anhand dessen ist ersichtlich, dass das primäre Ziel der Klassifikation mit linearen diskriminativen probabilistischen Modellen nicht die Maximierung der richtigen

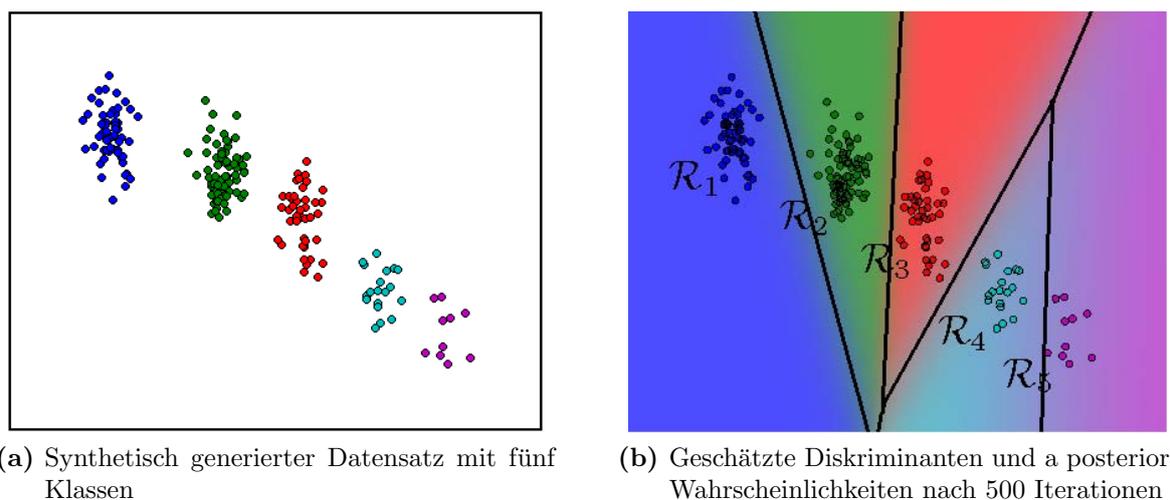


Abbildung 6.2: Der synthetische Datensatz mit drei Klassen liefert nach 500 Iterationen das Ergebnis aus (b) mit einem Kreuzentropiewert von $F = 17,39$.

Zuordnungen, sondern die Maximierung der a posteriori Wahrscheinlichkeiten ist, da dies den höheren Klassifikationserfolg bei Testdaten verspricht. Somit ist das Verfahren in gewissen Grenzen robust gegen Ausreißer. Der Wert der Kreuzentropie liegt nach 500 Iterationen bei $F = 17,39$.

Abbildung 6.3 stellt diesen Sachverhalt grafisch dar. Die Ordinateneinheit vernachlässigen wir, da hier lediglich der Verlauf der Kurve von Interesse ist. Die Grafik zeigt, dass die Kreuzentropiefehlerkurve (rot) gleichmäßig abfällt, während die Kurve über die Anzahl der falschen Zuordnungen (blau) vor allem am Anfang und am Ende schwankt. Die beiden vergrößerten Ausschnitte zeigen, dass die Maximierung der richtigen Zuordnungen nicht exakt mit der Minimierung der Kreuzentropie einhergeht.

In allen Fällen von linear separierbaren Merkmalen mit $K = \{2, \dots, 5\}$ Klassen konnten wir die Diskriminanten bestimmen, die die Kreuzentropie minimierten und alle Merkmale richtig zuordneten.

6.1.2 Nicht linear separierbare Merkmale

In diesem Abschnitt verwenden wir Beispiele mit nicht linear separierbaren Merkmalen. Wir arbeiten wieder mit dem Exact Line Search für das Gradientenabstiegsverfahren.

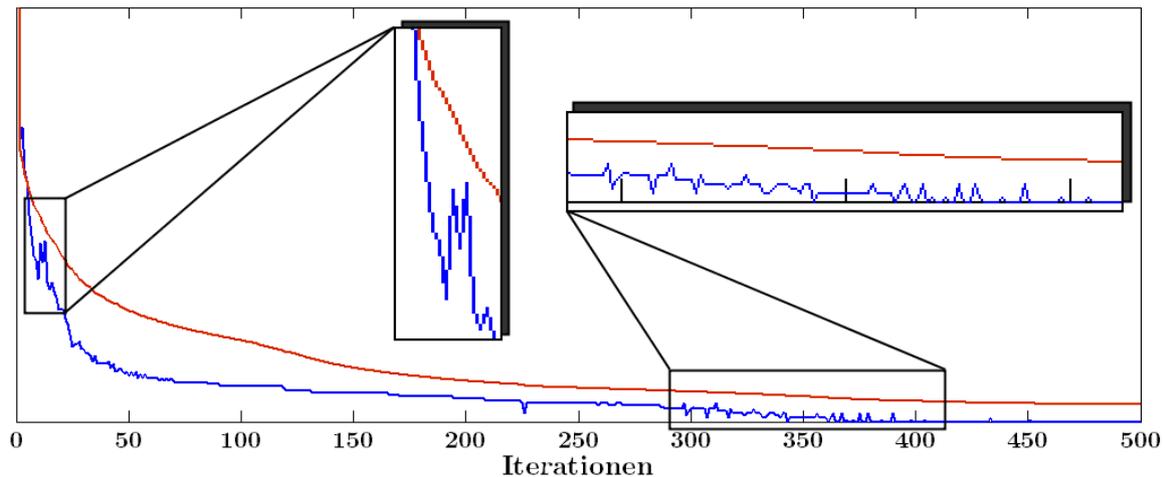


Abbildung 6.3: Die Maximierung der richtigen Zuordnungen geht nicht exakt mit der Minimierung der Kreuzentropie einher. Aufgetragen ist die Anzahl der falschen Zuordnungen (blaue Kurve) und der Wert der Kreuzentropie (rote Kurve) gegen die Anzahl der Iterationen zu dem Beispiel mit fünf Klassen aus 6.2a.

Abbildung 6.4a zeigt einen nicht linear separierbaren Datensatz mit drei Klassen und das Ergebnis mit den geschätzten Diskriminanten.

Die geschätzten Diskriminanten können im Falle eines nicht linear separierbaren Datensatzes die Merkmale nicht linear trennen. Sie streben allerdings gegen ein Minimum und trennen die Merkmale in der Art, dass die Kreuzentropie minimal wird. Abbildung 6.5 zeigt die Anzahl der falschen Zuordnungen und den Wert der Kreuzentropie aufgetragen gegen die Anzahl der Iterationen. Beide Größen fallen sehr schnell und konvergieren dann langsam gegen ihr Minimum.

Im nächsten Abschnitt verwenden wir reale Datensätze, deren Merkmale in einem hochdimensionalen Merkmalsraum liegen. Der Abschnitt diskutiert die drei Datensätze, die verwendeten Merkmale und Algorithmen und die erreichten Ergebnisse durch die Betrachtung der Evaluierung mit den Konfusionsmatrizen.

6.2 Anwendung auf die realen Datensätze

Das Modell wenden wir auf drei im Bereich Computer Vision weit verbreitete Datensätze an, die Bilder realer Objekte beinhalten: (i) den MSRC 23-Klassen Datensatz, (ii)

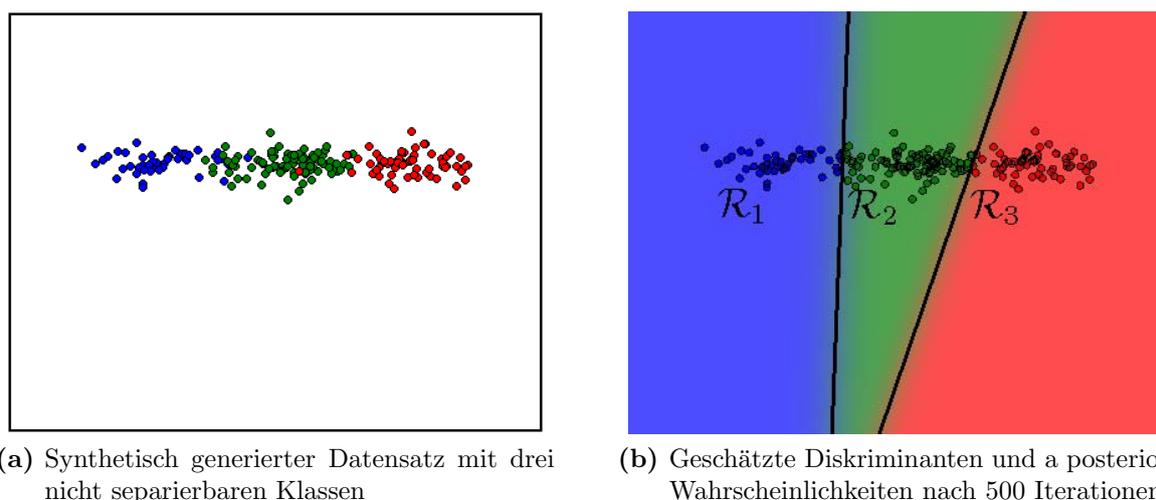


Abbildung 6.4: Der synthetische Datensatz mit drei nicht separierbaren Klassen liefert nach 500 Iterationen das Ergebnis aus (b) mit einem Kreuzentropiewert von $F = 42,64$.

den 7-Klassen Corel Datensatz und (iii) den 7-Klassen Sowerby Datensatz. Die folgenden Abschnitte stellen zunächst die Datensätze vor, beurteilen danach die verwendeten Merkmale und Verfahren und stellen anschließend die Klassifikationsergebnisse zusammen.

6.2.1 Verwendete Datensätze

Datensatz (i) ist der Microsoft Research Cambridge Datensatz⁸, der Farbbilder aus Innen- und Außenaufnahmen enthält. Der Datensatz beinhaltet 591 Bilder, bestehend aus 23 Klassen: „Gebäude“, „Gras“, „Baum“, „Kuh“, „Pferd“, „Schaf“, „Himmel“, „Berg“, „Flugzeug“, „Wasser“, „Gesicht“, „Auto“, „Fahrrad“, „Blume“, „Schild“, „Vogel“, „Buch“, „Stuhl“, „Straße“, „Katze“, „Hund“, „Körper“ und „Boot“. 491 Bilder verwenden wir zum Trainieren und die übrigen 100 zum Testen. Der Datensatz enthält Pixel, die mit „void“ gelabelt sind. Sie kennzeichnen entweder Label, die zu keiner der aufgeführten Klassen gehören oder entstehen dadurch, dass die Bilder schwach annotiert (engl. weakly labeled) sind. Diese verwenden wir nicht zum Trainieren. Von der Evaluierung schließen wir sie aus.

⁸Download unter http://research.microsoft.com/vision/cambridge/recognition/MSRC_ObjCategImageDatabase_v2.zip

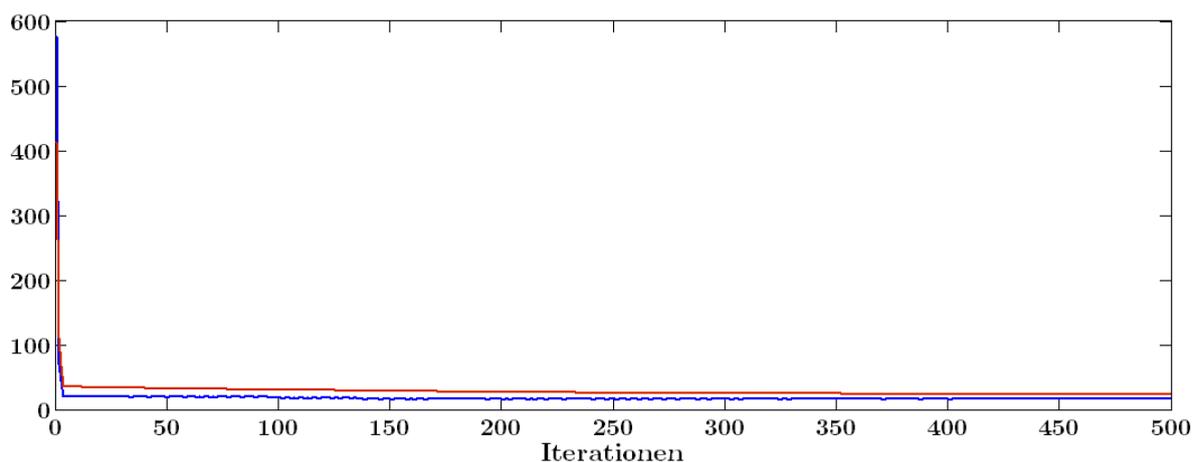


Abbildung 6.5: Die Anzahl der falschen Zuordnungen (blaue Kurve) und der Wert der Kreuzentropie (rote Kurve) sind gegen die Anzahl der Iterationen zu dem Beispiel mit drei Klassen aus Abbildung 6.4a aufgetragen. Die Werte können im Falle eines nicht linear separierbaren Datensatzes nicht gegen 0 konvergieren. Sie streben gegen ein Minimum.

Datensatz (ii) ist der Corel Datensatz⁹, der Farbbilder von wild lebenden afrikanischen und arktischen Tieren enthält. Der Datensatz besteht aus 100 Bildern mit 7 Klassen: „Nashorn/Nilpferd“, „Polarbär“, „Wasser“, „Schnee“, „Vegetation“, „Erdboden“ und „Himmel“. Von diesem verwenden wir 90 Bilder zum Trainieren und 10 zum Testen.

Datensatz (iii) ist der Sowerby Datensatz¹⁰, der Farbbilder mit ländlichen und vorstädtischen Motiven enthält. Er besteht aus 104 Bildern mit ebenfalls 7 Klassen: „Himmel“, „Vegetation“, „Fahrbahnmarkierung“, „Straße“, „Gebäude“, „Verkehrsgegenstände“ und „Auto“. Wir verwenden 90 Bilder zum Trainieren und die übrigen 14 zum Testen. In Tabelle 6.1 sind die verwendeten Datensätze mit den wichtigsten Eigenschaften zusammengefasst.

6.2.2 Beurteilung der Datensätze

Der MSRC Datensatz umfasst bei 23 Klassen lediglich 591 Bilder. Es ist umso schwieriger Merkmale zu finden, die im Merkmalsraum trennbar sind, je mehr Klassen der Datensatz besitzt. Auf Grund der Vielfalt der Klassenobjekte sind für diesen Datensatz 591 Bilder zu wenig, um die Variationen der Klassenobjekte zu erfassen. Die große Klassenanzahl

⁹Download unter http://www.cs.toronto.edu/%7Ehexm/data/corel_subset.mat

¹⁰Download unter http://www.cs.toronto.edu/%7Ehexm/data/sowerby_mod.mat

Datensatz	Anzahl Bilder			Anzahl der Klassen	Bildgröße [Pixel]
	Gesamt	Training	Testen		
Corel	100	90	10	7	240×360
Sowerby	104	90	14	7	192×288
MSRC	591	491	100	23	240×320 und 213×320

Tabelle 6.1: Zur Beurteilung des Klassifikators verwenden wir: (i) den MSRC 23-Klassen Datensatz, (ii) den 7-Klassen Corel Datensatz und (iii) den 7-Klassen Sowerby Datensatz.

bedarf weitaus mehr Merkmale, als die in dieser Arbeit verwendeten, und auch bei mehr Merkmalen würde der Trainingsprozess eine erhebliche Zeit in Anspruch nehmen. Das Problem bei langwierigen Trainingsprozessen ist, dass mit jeder Variation der Stellgrößen erneut der komplette Trainingsprozess durchlaufen werden muss und der Umfang des Datensatzes somit zu Lasten der Optimierung der Stellgrößen fällt.

Die Bilder des Sowerby Datensatzes sind 64×94 Pixel groß, das heißt sie sind niedrig aufgelöst. Obwohl wir die Bilder auf die dreifache Größe von 192×288 Pixel vergrößern, nehmen die Klassen „Fahrbahnmarkierung“, „Verkehrsgegenstände“ und „Auto“ manchmal nur ein paar wenige Pixel ein. Dies hat vor allem Nachteile für die Extraktion der Texturmerkmale, da die Filter eine Größe von 49×49 Pixel haben und die Merkmale für die soeben genannten Klassen durch die Filterantworten umgrenzender Klassen verwischt werden. Auffällig bei diesen Bildern ist die Gleichmäßigkeit der Art der Aufnahme, das heißt für alle Bilder herrschen nahezu gleiche Lichtverhältnisse und der Aufbau der Bilder gleicht sich stark. Ein Vorteil des Datensatzes sind somit die relativ geringen Variationen der Klassenobjekte, sodass wir davon ausgehen können, dass die gelernten Diskriminanten, falls sie die Trainingsdaten gut separieren, die Merkmale der Testbilder auch gut trennen können.

Die Schwierigkeit des Corel Datensatzes besteht in dem Finden von Merkmalen, die nahezu identische Objekte trennen. Das Problem zeigt sich deutlich bei den zwei Klassen „Polarbär“ und „Schnee“ und bei den Klassen „Nashorn“ und „Erdboden“. Die Klassen besitzen fast identische Farbmerkmale und kaum Textur.

Abbildung 6.6a und 6.6b zeigen zwei Bilder, bei denen die Tiere sehr stark ihrem Hintergrund ähneln. Bei diesen Bildern ist es sehr schwierig, linear separierbare Merkmale zu finden.



(a) Starke Ähnlichkeit der Klassen „Polarbär“ und „Schnee“



(b) Starke Ähnlichkeit der Klassen „Nashorn“ und „Erdboden“

Abbildung 6.6: Einige Klassen des Corel Datensatzes ähneln sich sehr stark, sodass es schwierig ist, linear separierbare Merkmale zu finden.

6.2.3 Beurteilung der gewählten Verfahren

Lineares probabilistisches diskriminatives Modell

In dieser Arbeit verwenden wir ein lineares probabilistisches diskriminatives Modell. Für unsere Anwendung ist dies eine sehr gute Wahl, da wir die a posteriori Wahrscheinlichkeiten direkt modellieren können und sie als klassenbedingte Wahrscheinlichkeiten für die Markov Random Fields verwenden können. Da wir das Bayestheorem nicht anwenden müssen, benötigen wir keine Informationen über die a priori Wahrscheinlichkeiten. Das hat vor allem einen praktischen Vorteil, da die a priori Wahrscheinlichkeiten meist nicht bekannt sind. Die Wahl der linearen Modelle stellt sich als gute Wahl heraus. Sie sind einfach in der Anwendung, besitzen eine geringe Parameterzahl und liefern trotz ihrer geringen Ordnung gute Ergebnisse.

Merkmalsextraktion

In dieser Arbeit verwenden wir die in der Literatur meist genutzten Gruppen von Merkmalen – Größe und Form von Regionen, Position, Farbe, Textur und Kontextmerkmale. Trotz der relativ kleinen Auswahl der Merkmale und der wenigen Bilder ist die Laufzeit für die Extraktion sehr hoch. Tabelle 6.2 stellt die Laufzeiten der Merkmalsextraktion zusammen.

Die Laufzeit variiert je nach Bildanzahl, Bildgröße, Superpixel- und Klassenanzahl. Fast 90% der Laufzeit verursacht die Faltung mit der Filterbank für die Extraktion der Texturmerkmale. Trotz Verkleinerung der Bilder auf 150×150 Pixel dauert die Faltung

	Datensatz	Anzahl der Bilder	Laufzeit	Laufzeit pro Bild
Superpixel	Corel	100	27 Min.	16 Sek.
	Sowerby	104	51 Min.	29 Sek.
	MSRC	591	1 Std. 32 Min.	9 Sek.
Merkmale aus Superpixel	Corel	100	2 Std. 38 Min.	95 Sek.
	Sowerby	104	2 Std. 31 Min.	87 Sek.
	MSRC	591	1 Tag 43 Min.	151 Sek.
Merkmale aus Rechteckumgebung	Corel	100	12 Std. 8 Min.	437 Sek.
	Sowerby	104	12 Std. 14 Min.	423 Sek.
	MSRC	591	3 Tage 3 Std. 21 Min.	447 Sek.

Tabelle 6.2: Die Laufzeiten der Merkmalsextraktion variieren je nach Bildanzahl, Bildgröße, Superpixel- und Klassenanzahl.

sehr lange, da 49 Filter zur Anwendung kommen. Eine Laufzeitverbesserung könnten wir dadurch erzielen, dass wir die Fouriertransformierten des Bildes und des Filters multiplizieren. Dies ist identisch mit einer Faltung. Diesen Ansatz setzen wir jedoch in dieser Arbeit nicht um, er stellt allerdings eine sinnvolle Verbesserung dar. Die Faltung müssen wir für jedes Bild nur einmal durchführen, auch wenn sich die Größe der Rechteckumgebung oder die Anzahl der Superpixel ändern, da wir die Filterantworten als Merkmal verwenden. Unter diesem Gesichtspunkt ist die Laufzeit der Extraktion akzeptabel.

Gradientenabstiegsverfahren

Für das Gradientenabstiegsverfahren versuchen wir verschiedene Ansätze. Ein Ansatz ist der Exact Line Search nach Boyd und Vandenberghe (2004, Kapitel 9, Seite 464), der verschiedene Schrittweiten testet, bevor er sie anwendet. Ein anderer Ansatz ist die dynamische Anpassung der Schrittweite während des Abstiegs. Nach der Wahl einer großen initialen Schrittweite testen wir nach jedem Schritt, ob die Fehlerfunktion einen geringeren Wert als zuvor erreicht. Falls dies der Fall ist, rechnen wir mit derselben Schrittweite weiter, ansonsten machen wir den Schritt rückgängig und wählen eine kleinere Schrittweite. Nach jedem erfolgreichen Schritt nach vorn setzen wir die Schrittweite wieder auf den großen initialen Wert. Dies bezweckt einerseits, dass wir bei jedem Schritt testen konnten, welche Schrittweite zur tiefsten Stelle in der näheren Umgebung führt, und dass das Verfahren durch das Zurücksetzen auf den großen Wert möglichst schnell konvergiert. Ein weiterer Ansatz war eine feste, manuell gewählte Schrittweite.

Alle drei Verfahren konvergieren gegen ein Minimum, jedoch mit unterschiedlicher Effizienz. Alle Varianten bis auf die manuelle Wahl der Schrittweite haben einen sehr hohen Rechenaufwand, unter anderem auch deswegen, da das Gradientenabstiegsverfahren schon im Allgemeinen sehr langsam konvergiert. Wir verwenden für den synthetischen Datensatz den Exact Line Search, da er am schnellsten das Minimum findet und für die realen Datensätze die manuelle Wahl der Schrittweite, da bei einer zu hohen Pixelzahl der Exact Line Search und die dynamische Schrittweite eine inakzeptable Laufzeit haben. Der Nachteil des Verfahrens ist die mühselige Suche nach der besten Schrittweite, da wir hierfür bis zu 800 Iterationen je Test durchlaufen müssen. Die Schrittweite kann je nach Wahl der Merkmale und nach den Eigenschaften des Datensatzes schwanken, sodass wir keine Garantie für eine gute oder sogar die beste Wahl der Schrittweite geben können.

Wir brechen das Gradientenabstiegsverfahren nach 800 Iterationen ab, um den Zeitaufwand in Grenzen zu halten. In keinem Fall der Tests mit realen Datensätzen konvergiert das Verfahren in weniger als 800 Iterationen.

Tabelle C.1 zeigt die Tests, bei denen wir die Schrittweiten $s = \{10^{-8}, 10^{-7}, 10^{-6}\}$ beim Sowerby Datensatz unter Verwendung von jedem 1000sten Pixel testen. Bei der Schrittweite 10^{-7} mit maximal 800 Iteration ist die Kreuzentropie am kleinsten und der Klassifikationserfolg am größten.

Datenreduktion

Die Datenreduktion erfolgt für die Trainingsdaten wie in Kapitel 5.3 beschrieben. Im Folgenden führen wir einen Test durch, der die Anzahl der Pixel variiert, um festzustellen, wieviele Punkte notwendig sind um, einen hohen Klassifikationserfolg bei einer geringen Laufzeit zu erzielen. Dazu führen wir die Segmentierung für verschiedene Anzahlen von verwendeten Pixeln durch. Der Test läuft auf dem Sowerby Datensatz mit allen Superpixelmerkmalen, einer Schrittweite von 10^{-7} und maximal 800 Iterationen für das Gradientenabstiegsverfahren. Die Ergebnisse stehen aufgelistet in Tabelle C.3.

Die Tabelle zeigt, dass die Prozentzahl der falsch zugeordneten Merkmale fällt, je mehr Pixel wir verwenden. Für die Vergleichbarkeit führen wir eine normalisierte Kreuzentropie ein, welche die Kreuzentropie dividiert durch die verwendete Pixelanzahl ist. Die normalisierte Kreuzentropie zeigt deutlich, dass der Fehler nach 800 Iteration umso kleiner ist, je mehr Pixel wir verwenden. Obwohl die normalisierte Kreuzentropie mit kleiner

werdender Pixelanzahl steigt, hat dies bis zu einer gewissen Grenze keinen Einfluss auf den Klassifikationserfolg. Erst bei jedem 10000sten verwendeten Pixel zeigt sich eine Verschlechterung des Klassifikationserfolges. Der Datensatz garantiert immer noch einen sehr hohen Erfolg, auch wenn bis zu 37% der Merkmale falsch zugeordnet sind. Die Tabelle zeigt somit, dass die Diskriminanten bereits bei einer geringen verwendeten Pixelanzahl gut liegen, sodass es nicht notwendig ist, übermäßig viele Pixel zu verwenden.

Abbildung 6.7 zeigt den Verlauf der Kreuzentropie bei allen verwendeten Pixelanzahlen aufgetragen gegen die Anzahl der Iterationen. Der Verlauf aller Graphen ähnelt sehr stark. Wie im vorherigen Absatz beschrieben, ist die Kreuzentropie umso höher, je weniger Pixel wir verwenden.

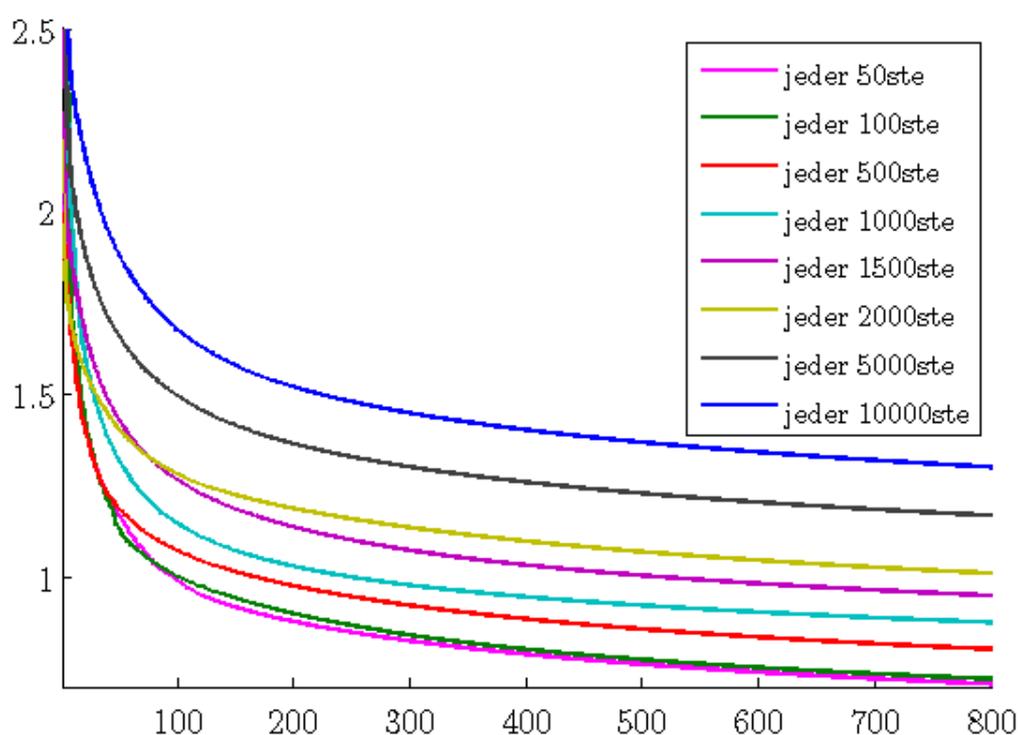


Abbildung 6.7: Der Graph zeigt die Kreuzentropie bei allen verwendeten Pixelanzahlen aufgetragen gegen die Anzahl der Iterationen. Die Kreuzentropie ist umso höher, je weniger Punkte wir verwenden.

Auf Grund der in Abschnitt 6.2.2 erläuterten Vorteile des Sowerby Datensatzes liegt die Mindestanzahl der Pixel für einen möglichst hohen Klassifikationserfolg sehr niedrig. Beim MSRC Datensatz, bei dem die Vielfalt der Klassen sehr hoch ist, oder beim Corel Datensatz, bei dem sich die Klassen sehr stark ähneln, ist es von Vorteil, mehr Pixel zu

verwenden. Aus diesem Grund verwenden wir für die Erstellung der Konfusionsmatrizen für den MSRC Datensatz jedes 5000ste Pixel und für den Corel Datensatz jedes 1000ste Pixel.

Markov Random Fields

Wir verwenden in dieser Arbeit die Markov Random Fields, um die Bilder semantisch zu segmentieren. Dies hat den Vorteil, dass die segmentierten Bilder ausschließlich homogene Regionen enthalten. Die Anwendung zeigt allerdings, dass das verwendete, vergleichsweise einfache Modell kaum besseren Klassifikationsergebnisse liefert, als die Segmentierung mit dem linearen probabilistischen diskriminativen Modell. Meist verbessert oder verschlechtert sich der Klassifikationserfolg um weniger als 1%. Dies liegt allerdings nicht in den Markov Random Fields selbst begründet. Es liegt vielmehr daran, dass wir ein sehr einfaches Modell mit einem relativ einfachen Versendeplan für die Loopy Belief Propagation verwenden. Eine komplexere Modellierung der Beziehungen in einem Feld würde vermutlich zu besseren Ergebnissen führen.

Die Bestimmung der Parameter im Markov Random Field erfolgt durch Validierung. Das heißt wir testen einige Parameter, betrachten den Klassifikationserfolg und wählen die Parameter mit dem größten Erfolg. Da nur das Verhältnis der Parameter relevant ist, halten wir $\eta = 10$ fest und variieren die Gewichtung der Homogenität β . Tabelle 6.3 fasst die Ergebnisse zusammen.

β	η	Klassifikationserfolg
10000	10	83,76
1000	10	83,76
100	10	83,76
10	10	83,76
1	10	83,77
0,1	10	83,77
0,01	10	83,78
0,0010	10	83,78

Tabelle 6.3: Die Tabelle listet die Variation der Markov Random Field Parameter im Sowerby Datensatz auf. Ohne die Markov Random Field Anwendung liegt der Klassifikationserfolg bei $KE = 83,47\%$.

Die Tabelle zeigt, dass sich die Ergebnisse fast garnicht voneinander unterscheiden. Das bedeutet, dass die Anwendung der Markov Random Fields Einfluss auf die Homogenität

hat, in unserem Fall jedoch kaum die Wahl der Parameter. Wir wählen für alle weiteren Tests und für die Erstellung der Konfusionsmatrizen die Werte $\beta = 0,01$ und $\eta = 10$.

Validierung mit Konfusionsmatrizen

Wir verwenden in dieser Arbeit eine Validierung mit Konfusionsmatrizen zur Beurteilung der Klassifikationserfolge der einzelnen Klassen der Datensätze. Eine Alternative ist die Kreuzvalidierung. Sie verwendet für die Bewertung des Datensatzes die Trainings- und Testdaten in mehreren Kombinationen. Es gibt verschiedene Möglichkeiten diese zu wählen. Entweder werden alle Kombinationen der Bilder validiert oder alle Kombinationen von festen Gruppen. Für einen Datensatz mit 100 Bildern und zehn Testdaten wären dies zehn verschiedene Kombinationen, damit alle Bilder einmal validiert werden. Dies hat den Vorteil, dass wir die Ergebnisse mitteln können und nicht nur eine Kombination, sondern den ganzen Datensatz bewerten können.

In dieser Arbeit verwenden wir die Kreuzvalidierung nicht, da sie zu zeitaufwendig ist. Auf Grund dessen können wir jedoch nur etwas über die Leistungsfähigkeit des Klassifikators bei einer zufällig gewählten Kombination von Test- und Trainingsdaten aussagen.

Die Wahl der Testdaten erfolgt zufällig. Dies hat den Vorteil, dass die Auswahl der Daten sehr schnell geht. Es hat jedoch auch den Nachteil, dass es passieren kann, dass Klassen mit wenig Daten ungenügend trainiert oder getestet werden können.

6.2.4 Beurteilung der Merkmale

Superpixel und Rechteckumgebung

Für die Extraktion der Merkmale können wir entweder eine Rechteckumgebung oder eine Superpixelumgebung wählen. Der Vorteil der Superpixel ist die Verwendung von homogenen Regionen. Entgegen dessen können in einer Rechteckumgebung starke Inhomogenitäten auftreten. Da die Merkmale in den Superpixeln alle gleich sind, klassifizieren wir regionenbasiert, wenn wir ausschließlich Superpixelmerkmale verwenden. Unter Verwendung von Rechteckumgebungen klassifizieren wir jedes Pixel einzeln.

Die Extraktion der Superpixel ist sehr rechenintensiv, sobald die Bilder sehr groß sind, wobei die Größe durch den Speicher von Matlab auf 145×145 Pixel begrenzt ist. Um die Rechenzeit zu verringern, verkleinerten wir die Bilder auf eine Pixelanzahl von 10000 Pixel.

Um allen Pixeln im Superpixel die gleichen Merkmale zuzuweisen, haben wir verschiedene Möglichkeiten. Wir verwenden den Mittelwert und ein Fünf-Säulen-Histogramm. Das Histogramm hat den Vorteil, dass eventuelle Homogenitätsschwankungen immer noch repräsentiert und nicht herausgemittelt werden. Tabelle 3.1 stellt Tests mit den verschiedenen Merkmalen zusammen. Diese lässt jedoch keinen sicheren Schluss zu, welches Merkmal die beste Segmentierung liefert. Es ist nur ersichtlich, dass eine Segmentierung ohne Mittelwertmerkmale sehr viel schlechtere Ergebnisse liefert. Es ist erkennbar, dass Mittelwertmerkmale sowie die Kombination von Histogramm- und Mittelwertmerkmalen gute Ergebnisse liefern. Für die Erstellung der Konfusionsmatrizen verwenden wir die Kombination, da diese die geringste Kreuzentropie liefert.

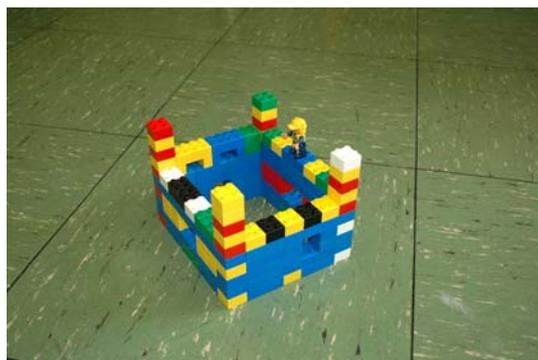
Farbmerkmale

Farbinformationen sind auf Grund ihrer Einfachheit und ihres meist großen Informationsgehalts sehr gut als Merkmale geeignet. In dieser Arbeit verwenden wir verschiedene Farbräume. Durch die Transformation in andere Farbräume können wir zwar keine Informationen hinzugewinnen, allerdings können Merkmale, die in einem Raum schlecht separierbar sind, in einem anderen Raum sehr gut trennbar sein.

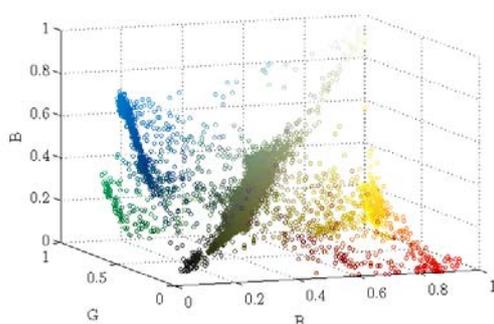
Abbildung 6.8a zeigt ein Bild mit Vordergrund (Legoburg) und Hintergrund (Boden), welches im RGB-Raum nicht linear trennbar ist. Abbildung 6.8b zeigt, dass die Pixel des grau-grünen Bodens ungefähr in der Mitte des Raumes liegen und von den Pixeln der bunten Legosteine umschlossen werden. Durch die Lage kann keine Ebene gefunden werden, die die Merkmale der Legoburg von denen des Bodens trennt. Abbildung 6.8c zeigt die Lage der RGB-Pixel im HSV-Raum. Die Pixel sind der Übersichtlichkeit halber in ihren RGB-Werten eingefärbt. Die Pixel des Bodens liegen in einem wenig gesättigten Bereich, während die bunten Legoburgpixel in einem hohen Sättigungsbereich liegen. Auf Grund dessen können die Merkmale in diesem Raum sehr leicht getrennt werden.

Einen großen Vorteil bietet der HSV-Raum, da durch ihn Informationen unabhängig von der Helligkeit gewonnen werden können. Wir nutzen dies bei der Texturanalyse, indem wir nur den Farbton- und den Sättigungskanal der Bilder verwenden.

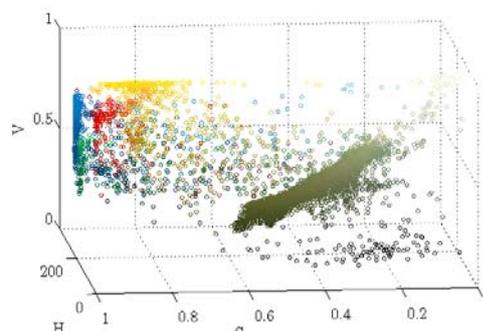
Klassen wie „Vegetation“, „Himmel“ und „Straße“ sind zum Beispiel durch ihre Farbwerte sehr gut unterscheidbar. Allerdings können wir Klassen wie „Baum“ und „Gras“ nur sehr schwer unterscheiden.



(a) RGB-Bild



(b) Pixel im RGB-Raum



(c) Pixel im HSV-Raum

Abbildung 6.8: Dieses Beispiel zeigt ein Bild mit Vordergrund (Legoburg) und Hintergrund (Boden), welches im RGB-Raum nicht linear separierbar ist, im HSV-Raum jedoch sehr gut. Der Boden umfasst das dichte grau-grüne Cluster und die Legoburg alle anderen farbigen Pixel.

Neben dem Mittelwert oder dem Histogramm aus einer Umgebung extrahieren wir die Standardabweichung. Mit ihr können wir gut Regionen mit homogenen Farbmerkmalen von denen mit einem großen Farbmerkmalspektrum unterscheiden. Die Farbmerkmale decken einen weiten Bereich der Farbräume ab, sodass diese als einzige Merkmale die zur Verfügung stehenden Informationen ausschöpfen.

Texturmerkmale

Als Texturmerkmale verwenden wir die Filterantworten für jedes Pixel. Der Vorteil der Texturmerkmale ist, dass sie Informationen über die Struktur unabhängig von der Helligkeit liefern. Die Merkmale sind sehr gut für Klassen mit markanten Strukturen geeignet, wie „Gras“ und „Schaf“. Sie können allerdings nicht zwischen Klassen mit homogenen Regionen unterscheiden, wie zum Beispiel „Himmel“, „Straße“ und „Schnee“.

Da die Filtergröße mit 49×49 Pixel sehr groß in Bezug auf die Bilder ist, gibt es einerseits Probleme am Rand eines Bildes und bei Klassenregionen mit kleiner Ausdehnung. Bei einer Faltung wird die Umgebung eines Pixels im Bild mit einbezogen. Bei diesen Filtern benötigen wir zu allen Seiten je 24 Pixel zur Berechnung der Filterantwort an der Stelle des Pixels. Die Größe der Bilder im Sowerby Datensatz beträgt allerdings 94×64 Pixel, sodass effektiv nur 46×26 Pixel mit tatsächlich vorhandener Bildinformation gefaltet werden. Auf Grund dessen vergrößern wir die Bilder auf eine Größe von 150×150 Pixel, da dies einen Kompromiss zwischen Laufzeit und Anwendbarkeit des Verfahrens darstellt. Die Regionen von Klassen, die eine kleine Ausdehnung haben, wie zum Beispiel „Fahrbahnmarkierung“ aus dem Sowerby Datensatz, können auch nach der Bildvergrößerung keine Filterantworten liefern, die ihre Struktur widerspiegeln, da der Einfluss von Strukturen angrenzender Regionen größer ist, als der der eigenen. Daraus folgt, dass die Texturmerkmale nur für große Regionen geeignet sind.

Positionsmerkmale

Die Positionsmerkmale sind sehr gut für Bilder geeignet, in denen sich der Horizont ungefähr in der Bildmitte befindet. Eine Klasse wie „Himmel“ befindet sich in diesem Fall immer in der oberen Bildhälfte und Klassen wie „Wasser“, „Erdboden“ und „Schnee“ befinden sich in der unteren Hälfte. Dieses Merkmal kann jedoch keine Informationen über Klassen liefern, die gleichermaßen oft an allen Stellen in der realen Welt vorkommen können. Beispiele hierfür sind die Klassen „Vogel“ und „Flugzeug“.

Ein Nachteil dieses Merkmals ist außerdem, dass es nicht die Größe und die Nähe des Objekts vor der Kamera einbeziehen kann. Nahe Objekte sind größer als gleich große, weit entfernte und nehmen somit mehr Bildfläche ein. Das bedeutet, dass der Bereich der Positionen sehr viel größer ist.

Ein weiterer Nachteil ist, dass die Positionsmerkmale nicht die Neigung der Kamera bei der Aufnahme mit einbeziehen können. Einige Bilder sind komplett unter oder über dem Horizont aufgenommen. Dies hat zur Folge, dass die Merkmale von Bildern mit dem Horizont in der Bildmitte von denen ohne Horizont „verwischt“ werden. Das Merkmal kann somit nicht zwischen guten und schlechten Positionen unterscheiden.

Kontextmerkmale

Durch die Kontextmerkmale können wir berücksichtigen, dass es Klassen gibt, die häufig zusammen auftreten. Zum Beispiel befinden sich Schafe und Kühe oft auf der Wiese und

Autos auf der Straße. Selten hingegen befinden sich Autos im oder am Wasser oder Schafe auf der Straße. Als Merkmale verwendeten wir die Farbwerte der häufigsten Nachbarn.

Der Nachteil dieser Merkmale ist, dass sie nur die häufigsten beziehungsweise die zweithäufigsten Merkmale verwenden. Die Abbildungen 3.12a und 3.12b besitzen beispielsweise im RGB-Raum drei Farbcluster – weiß, schwarz und grün. Die grünen Pixel erhalten die Farbwerte der weißen Pixel und andersherum. Die schwarzen Pixel erhalten zwar auch die Farbwerte der grünen Pixel, jedoch würde dieses Prinzip nicht mehr funktionieren, wenn das Schaf mehr Bildfläche einnehmen würde. In diesem Fall erhielten die schwarzen Pixel die Farbmerkmale der weißen Pixel. Weitere Nachteile sind, dass die Kontextmerkmale nur so gut sind, wie die Farbmerkmale selbst auch, und wir nicht mehr als zwei Klassen einbeziehen können. Die Kontextmerkmale sind demnach nur gut geeignet, wenn sich wenige Klassen im Bild befinden und entweder die Bildproportionen der Klassen stimmen oder nicht mehr als zwei Farbcluster im Bild existieren.

6.2.5 Segmentierte Bilder und Evaluierungsergebnisse

Die Evaluierung erfolgt durch die Erstellung der Konfusionsmatrizen und der Betrachtung der Klassifikationserfolge. Die Zeilen der Konfusionsmatrizen sind mit ihrer Summe normiert und die Angaben in Prozent. Die Prozente der richtig klassifizierten Pixel befinden sich auf der Diagonalen und die der falsch zugeordneten in den restlichen Zellen der Matrix. Die Ordinatenachse gibt die wahren Label an und die Abszissenachse die detektierten. Zur besseren Veranschaulichung sind die Felder der Matrix unterlegt, wobei das Feld umso dunkler ist, je höher der Klassifikationserfolg ist. Leere Felder haben den Wert 0,0.

Die Konfusionsmatrizen erstellen wir für den Klassifikationserfolg mit den Superpixelmerkmalen und den Rechteckmerkmalen. Dabei unterscheiden wir zwischen der Klassifikation mit dem linearen probabilistischen diskriminativen Modell und der Klassifikation mit Markov Random Fields. An dieser Stelle greifen wir die Konfusionsmatrix mit Superpixelmerkmalen unter Anwendung der Loopy Belief Propagation heraus und diskutieren die Einträge. Alle Konfusionsmatrizen sind im Anhang D aufgeführt.

Die folgenden Abschnitte stellen zunächst die Klassifikationserfolge, dann die Konfusionsmatrizen und anschließend einige Beispiele der klassifizierten Bilder zusammen. Wir

beurteilen sie anhand dessen, um etwas über die Leistungsfähigkeit des Klassifikators aussagen zu können.

Allgemeine Betrachtungen

Alle Segmentierungen führen wir mit dem Gradientenabstiegsverfahren, einer Schrittweite von 10^{-7} und höchsten 800 Iterationen durch. In allen Fällen konvergiert das Verfahren nach 800 Iterationen noch nicht vollständig. Die Anzahl der Iterationen stellt jedoch einen guten Kompromiss zwischen Laufzeit und Optimierung dar. Für den Corel und den Sowerby Datensatz verwenden wir jeden 1000sten Punkt und für den MSRC Datensatz jeden 5000sten Punkt. Für die Datensätze ergeben sich die Klassifikationserfolge, die in Tabelle 6.4 zusammengefasst sind.

Datensatz	Merkmale	Klassifikationserfolg	
		ohne LBP[%]	mit LBP[%]
Corel	Rechteckumgebung	58,0	57,9
	Supapixel	53,8	53,8
Sowerby	Rechteckumgebung	82,9	83,1
	Supapixel	84,6	85,1
MSRC	Rechteckumgebung	30,1	30,1
	Supapixel	35,5	35,5

Tabelle 6.4: Die Tabelle führt die erreichten Klassifikationserfolge der Datensätze mit Supapixelmerkmalen mit Anwendung der Loopy Belief Propagation auf.

Die Tabelle zeigt, dass der Sowerby Datensatz sehr gute, der Corel Datensatz gute und der MSRC für den Klassenumfang akzeptable Ergebnisse liefert. Die Supapixel liefern in zwei von drei Fällen bessere Ergebnisse, jedoch nicht stark signifikant. Beim Corel Datensatz sind die Ergebnisse unter Verwendung der Supapixel sogar schlechter. Ein möglicher Grund ist, dass sich die Merkmale zwischen mehreren Klassen stark ähneln und somit viele Supapixel mehrere Klassen enthalten.

Die Anwendung der Markov Random Fields liefert keine deutliche Verbesserung der Klassifikationserfolge.

Sowerby Datensatz

Der Sowerby Datensatz erreicht den größten Klassifikationserfolg in dieser Arbeit, wie Abbildung 6.9 zeigt. Klassen wie „Himmel“, „Straße“ und „Vegetation“ werden sehr gut erkannt. Ein Grund hierfür ist, dass die Merkmale der Klassen sehr gut separierbar sind und sie kaum in den Bildern variieren. Dadurch können wir den Klassifikator gut trai-

nieren. Die soeben genannten Klassen nehmen den größten Teil des Bildes ein, sodass die Texturen kaum durch Einfluss anderer Klassen verwischt sind. Wie erwartet, werden Klassen mit einem geringen Bildanteil gar nicht erkannt. Dazu gehören „Fahrbahnmarkierung“, „Gebäude“, „Verkehrsgenstand“ und „Gebäude“. Es ist außerdem sehr auffällig, dass sie nie, also auch nicht fälschlicherweise detektiert werden. Die Fahrbahnmarkierung zum Beispiel wird oft als „Vegetation“ oder „Straße“ erkannt. Der Grund hierfür könnte sein, dass die Texturen auf Grund der Größe der Filter fast nur Merkmale der Straße enthalten. Im Merkmalsraum sind die Klassen somit kaum separierbar. Auch die anderen, mit wenig Erfolg klassifizierten Klassen werden fast ausschließlich den Klassen „Vegetation“ und „Straße“ zugeordnet.

Himmel	93,1	4,6		2,3			
Vegetation	4,8	85,4		9,7			
Fahrb.-mark.		60,2		39,8			
Straße		9,3		90,7			
Gebäude	2,4	54,9		42,7			
Straßenobj.	26,1	63,4		10,5			
Auto		81,0		19,0			
	Himmel	Vegetation	Fahrb.-mark.	Straße	Gebäude	Straßenobj.	Auto

Abbildung 6.9: Die Abbildung zeigt die Konfusionsmatrix des Sowerby Datensatzes mit Superpixelmerkmalen mit Anwendung der Loopy Belief Propagation. Der Klassifikationserfolg liegt bei $KE = 85,12\%$.

Die Beispiele aus Abbildung 6.10 zeigen in den ersten beiden Spalten die Segmentierungen, die mit Superpixeln sehr gute und mit den Rechteckmerkmalen gute Ergebnisse liefern. Es ist ersichtlich, dass die Rechteckmerkmale mehr Probleme mit Objekten wie der Klasse „Baum“ haben, bei denen der Hintergrund hindurchscheint. An diesen Stellen wird oft die Klasse „Straße“ statt „Vegetation“ detektiert.

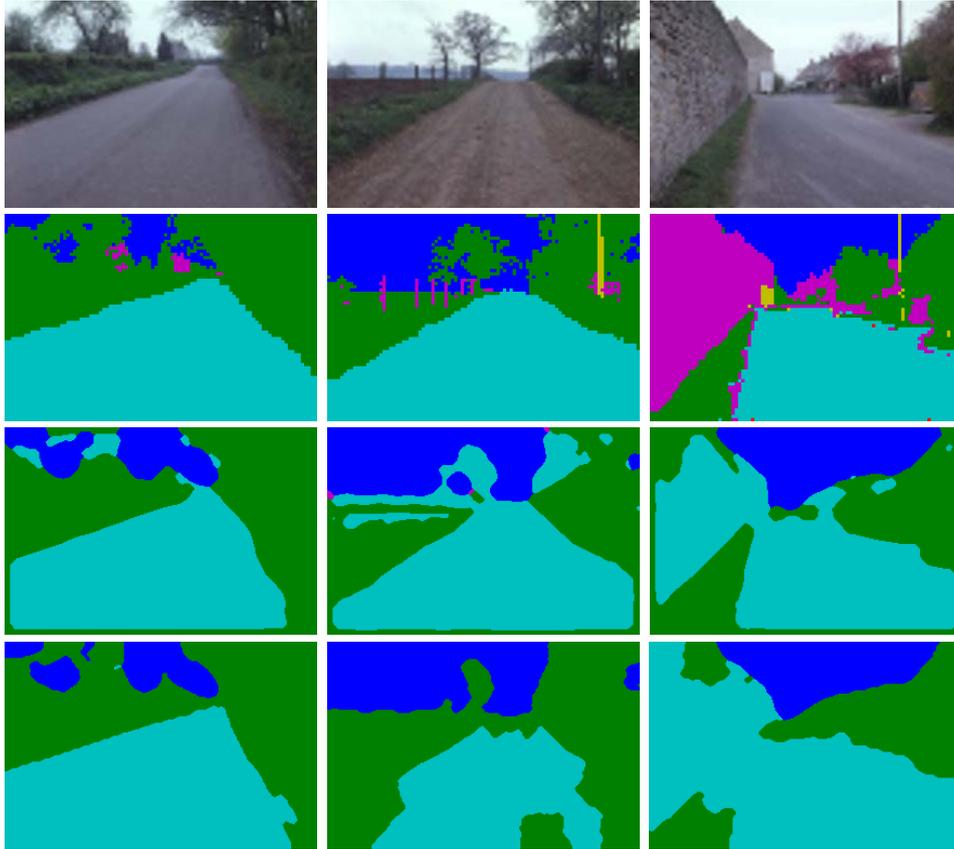


Abbildung 6.10: Die Abbildung zeigt einige Ergebnisse der segmentierten Bilder aus dem Sowerby Datensatz. In der ersten Zeile befinden sich die Originalbilder, in der zweiten die Annotationen, in der dritten die Ergebnisse mit Rechteckmerkmalen und in der vierten die Ergebnisse mit Superpixelmerkmalen.

In der zweiten Spalte sieht man einen negativen Effekt der Superpixel. Dort wird ein Teil der Straße falsch klassifiziert. Wenn ein Superpixel eine relativ große Fläche des Bildes einnimmt und falsch klassifiziert wird, dann kann die Auswirkung auf den Klassifikationserfolg sehr viel größer sein, als wenn nur ein paar wenige Pixel durch ihre Rechteckumgebung falsch klassifiziert werden.

Die dritte Spalte zeigt ein Beispiel, bei dem beide Verfahren schlecht klassifizieren. Die Klasse „Gebäude“ wird nicht erkannt und stattdessen der Klasse „Straße“ zugeordnet. Wie schon aus der Konfusionsmatrix ersichtlich, wird „Gebäude“ nie erkannt, obwohl sie im Trainingsdatensatz vertreten ist und auch nach Tabelle 6.5 in den Testdaten sehr oft vorkommt.

Klasse	Farbe der Annotation	Anzahl Pixel pro Klasse
Himmel		137.808
Vegetation		286.722
Fahrbahnmarkierung		1.197
Straße		310.797
Gebäude		33.894
Verkehrsgegenstand		3.348
Auto		378

Tabelle 6.5: Die Tabelle listet die Klassen des Sowerby Datensatzes mit ihrer Farbe in der Annotation und die Anzahl der Pixel pro Klasse im Testdatensatz auf.

Corel Datensatz

Der Corel Datensatz ist ein schwieriger Datensatz, da die Merkmale der Klassen sich sehr ähnlich sind und somit eine Segmentierung mit den verwendeten Merkmalen ungenügende Ergebnisse liefert. Die Klassen, die sehr gut erkannt wurden, sind „Nashorn/Nilpferd“, „Schnee“ und „Erdboden“. Weniger gute Ergebnisse lieferte die Klasse „Wasser“ und schlechte Ergebnisse lieferte die Klasse „Vegetation“. Die Klassen „Himmel“ und „Polarbär“ wurden nie beziehungsweise fast nie erkannt.

Wider den Erwartungen wird die Klasse „Polarbär“ nicht am häufigsten der Klasse „Schnee“ zugeordnet, sondern der Klasse „Erdboden“. Das liegt daran, dass der Erdboden ein sehr weites Spektrum an Textur und Farbe einnimmt und die Polarbären meist nicht ganz weiß sind. Auf Grund der großen Variation der Klassen „Wasser“ und „Erdboden“ werden diese am häufigsten fälschlicherweise anderen zugeordnet. Auffällig ist, dass die Klasse „Schnee“ in allen Fällen falsch als „Wasser“ klassifiziert wird. Es liegt die Vermutung nahe, dass die Trainingsdaten schlecht gewählt sind und die Trainingsdaten nicht repräsentieren. Diese Ergebnisse zeigen, dass die Art und Menge der verwendeten Merkmale für diesen Datensatz nicht günstig sind. Die Klassenobjekte „Polarbär“ und „Nashorn/Nilpferd“ nehmen häufig Merkmale ihres Hintergrundes an und verschwinden somit häufig davor.

Die Beispiele aus dem Corel Datensatz aus Abbildung 6.12 zeigen in der ersten Spalte ein gutes Ergebnis mit dem Superpixelmerkmalen, in der zweiten Spalte ein gutes Ergebnis mit Rechteckmerkmalen und in der letzten Spalte eine Segmentierung, die mit beiden Merkmalsarten schlechte Ergebnisse liefert. Anders als beim Sowerby Datensatz sind bei diesem Datensatz die Unterschiede zwischen Rechteck- und Superpixelmerkmalen sehr

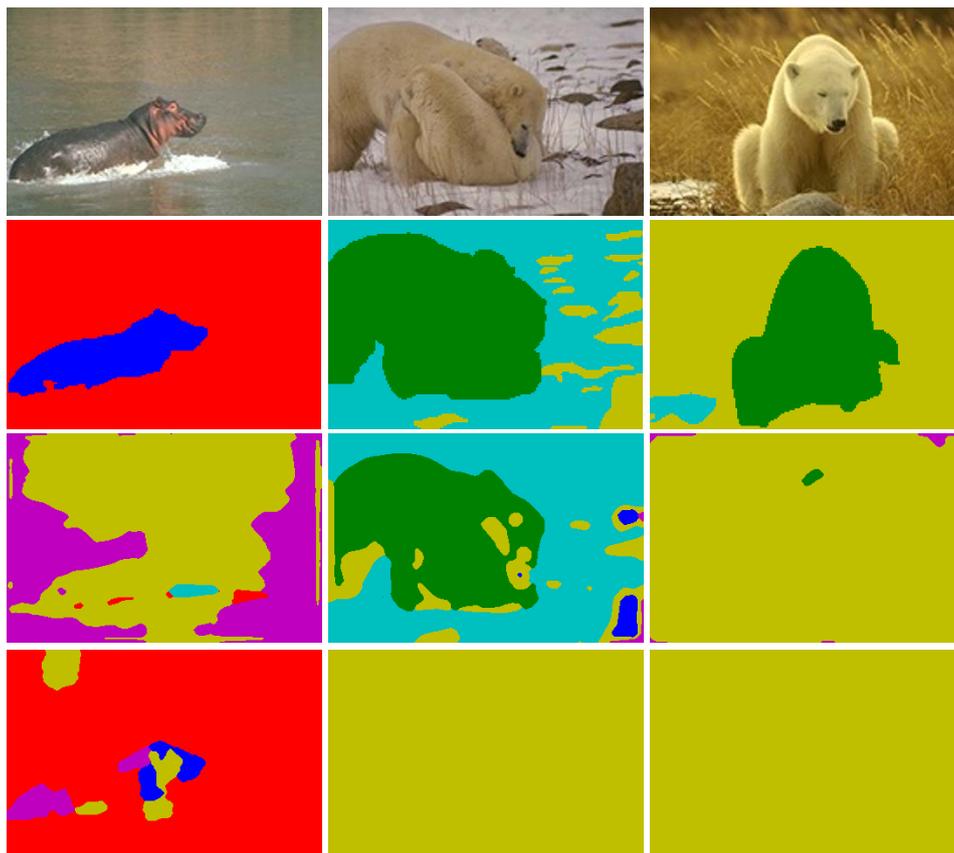


Abbildung 6.12: Die Abbildung zeigt einige Ergebnisse der segmentierten Bilder aus dem Corel Datensatz. In der ersten Zeile befinden sich die Originalbilder, in der zweiten die Annotationen, in der dritten die Ergebnisse mit Rechteckmerkmalen und in der vierten die Ergebnisse mit Superpixelmerkmalen.

Laufzeit sonst zu hoch ist. Aus diesem Grund liegt keine Evaluierung mit allen Pixeln vor. Die Verwendung aller Pixel würde mit hoher Wahrscheinlichkeit jedoch bessere Ergebnisse liefern.

Einige Klassen wurden nie richtig klassifiziert. Die Gründe hierfür mögen klassenspezifisch etwas schwanken, jedoch sind die Hauptgründe, dass sich viele Klassen zu stark ähneln und die Variationen innerhalb der Klasse zu groß sind. Die Klassen „Fahrrad“ und „Boot“ decken zum Beispiel einen sehr großen Teil des Farbspektrums ab, sodass diese nicht durch die Farbmerkmale voneinander und von anderen Klassen unterscheidbar sind. Viele Klassen weisen kaum oder sich stark ändernde Texturen auf, sodass die Texturen zwischen verschiedenen Klassen nicht trennbar sind oder keine Texturcluster im Merkmalsraum entstehen. Texturcluster treten vor allem dann auf, wenn sich die Struk-

Klasse	Farbe der Annotation	Anzahl Pixel pro Klasse
Nashorn/Nilpferd		75.164
Polarbär		122.404
Wasser		148.172
Schnee		237.392
Vegetation		95.852
Erdboden		169.732
Himmel		15.284

Tabelle 6.6: Die Tabelle listet die Klassen des Corel Datensatzes mit ihrer Farbe in der Annotation und die Anzahl der Pixel pro Klasse im Testdatensatz auf.

tur der Oberfläche oft wiederholt und es einige weniger dominante Strukturmerkmale gibt. Da sich oft mehr als zwei Klassen im Bild befinden, tragen auch die Kontextmerkmale in diesem Fall zur linearen Separierbarkeit nicht bei. Dadurch ist klar, dass die verwendeten Merkmale für diesen Datensatz zu wenig sind.

Die Beispiele des MSRC Datensatzes aus Abbildung 6.14 zeigen in der ersten Spalte eine Segmentierung mit guten Ergebnissen der Superpixelmerkmale, in der zweiten Spalte eine gute Segmentierung beider Merkmalsgruppen und in der letzten Spalte eine schlechte Segmentierung. Wie beim Corel Datensatz sind auch hier die Unterschiede zwischen den Rechteck- und den Superpixelmerkmalen sehr groß. Auf Grund der großen Klassenanzahl gibt es kaum gute Ergebnisse. In den meisten Fällen wird gut segmentiert, jedoch nicht gut semantisch segmentiert. Dies wird am Beispiel aus der dritten Spalte besonders deutlich. In diesem Bild werden homogene Regionen als eine Klasse erkannt, jedoch nicht mehrere Regionen als die Klasse „Zeichen“.

Auffällig sind die Bildränder der Segmentierungen mit den Rechteckmerkmalen aus der dritten Spalte. Dies liegt vermutlich an der Bildfortsetzung mit Nullen wegen der Faltung für die Texturmerkmale. Diese verwischen am Bildrand. Eine andere Bildfortsetzung, zum Beispiel gespiegelt, könnte eventuell bessere Ergebnisse liefern.

Klasse	Farbe der Annotation	Anzahl Pixel pro Klasse
Gebäude		448.432
Gras		1.545.116
Baum		332.507
Kuh		71.776
Pferd		0
Schaf		45.516
Himmel		436.767
Berg		90.067
Flugzeug		77.300
Wasser		247.750
Gesicht		66.846
Auto		198.146
Fahrrad		65.720
Blume		172.530
Zeichen		98.747
Vogel		41.766
Buch		167.190
Stuhl		102.596
Straße		424.516
Katze		83.866
Hund		83.882
Körper		66.143
Boot		19.022

Tabelle 6.7: Die Tabelle listet die Klassen des MSRC Datensatzes mit ihrer Farbe in der Annotation und die Anzahl der Pixel pro Klasse im Testdatensatz auf.

Gebäude	26,8	23,0	11,8			23,9	4,3	1,6		0,3				0,3	7,9								
Gras	2,7	87,4	3,2			4,1				0,1				0,2	2,3								
Baum	21,2	26,6	35,2			2,0	0,1	0,3					0,7		13,8								
Kuh	1,2	75,2	4,3			6,1		0,1	0,4						1,9			10,8					
Pferd																							
Schaf		90,0				9,0		1,0															
Himmel	0,5	20,7	1,1			57,8		1,9							18,0								
Berg		49,4	0,6			41,9		5,1							3,0								
Flugzeug	36,0	21,7	14,9			10,3									17,1								
Wasser	0,6	34,7	4,0			44,7	0,3	5,7		0,6			7,5	0,1	1,8				0,1				
Gesicht	35,2	24,7	9,1	0,9		4,3	0,3	4,5		2,1	0,7		18,0						0,2				
Auto	25,9	16,9	35,3			8,3		5,7	1,4	0,1					6,5								
Fahrrad	42,0	28,0	22,6			0,2									7,2								
Blume	6,7	46,2		6,1	0,8	2,5			11,9		0,5		20,2					5,1					
Zeichen	38,9	14,5	11,1			20,3		13,0							2,3								
Vogel	4,6	73,2	5,4			0,1		1,5							15,2								
Buch	9,4	44,2		12,3	0,2	3,7		1,1	3,4	19,1			6,5										
Stuhl	20,3	53,7	3,2	0,2		3,9		6,5		2,5	0,4	0,6	1,4		7,4								
Straße	15,1	34,2	6,5			26,2	0,7	5,1		0,3	0,3		1,0	0,6	10,1								
Katze	14,0	16,4	37,4			5,6		4,6					15,1	2,6	4,4								
Hund	12,9	24,6	3,2			8,4		0,5		8,8			30,5	10,8	0,2								
Körper	11,8	30,1	17,2	2,3		6,0	9,7	2,3	0,5	0,9	3,7		11,1	0,1				3,1	1,1				
Boot	23,3	37,1	3,6			23,5							12,5										
	Gebäude	Gras	Baum	Kuh	Pferd	Schaf	Himmel	Berg	Flugzeug	Wasser	Gesicht	Auto	Fahrrad	Blume	Zeichen	Vogel	Buch	Stuhl	Straße	Katze	Hund	Körper	Boot

Abbildung 6.13: Die Abbildung zeigt die Konfusionsmatrix des MSRC Datensatzes mit Superpixelmerkmalen mit Anwendung der Loopy Belief Propagation. Der Klassifikationserfolg liegt bei $KE = 35,51\%$.

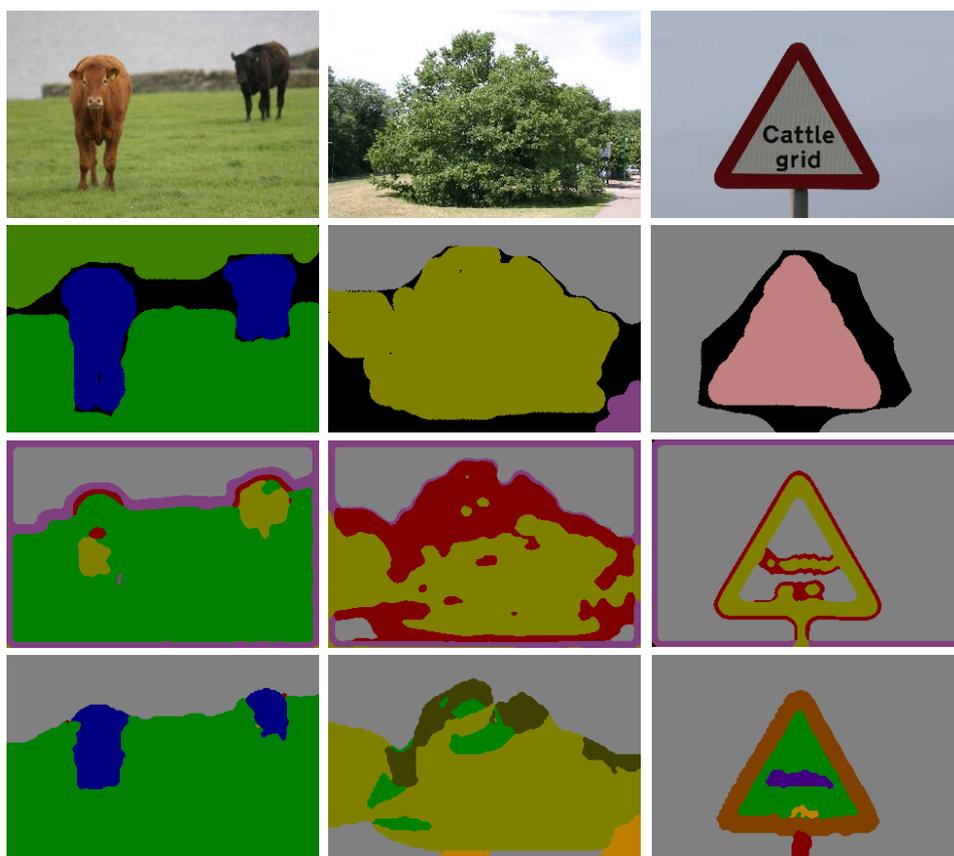


Abbildung 6.14: Die Abbildung zeigt einige Ergebnisse der segmentierten Bilder aus dem MSRC Datensatz. In der ersten Zeile befinden sich die Originalbilder, in der zweiten die Annotationen, in der dritten die Ergebnisse mit Rechteckmerkmalen und in der vierten die Ergebnisse mit Superpixelmerkmalen.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

In dieser Arbeit konnten wir ein lineares probabilistisches diskriminatives Modell lernen, indem wir die Parameter in einer Maximum Likelihood Schätzung bestimmten. Die Segmentierung der Bilder unter Verwendung der Markov Random Fields mit Loopy Belief Propagation setzten wir um. Wir entwickelten ein vollständiges Programm, welches einen Datensatz als Eingabe erhält und alle Bilder semantisch segmentiert. Das Programm umfasst die Merkmalsextraktion, das Lernen der Parameter des linearen probabilistischen diskriminativen Modells durch Optimierung mit dem Gradientenabstiegsverfahren, die Segmentierung durch Loopy Belief Propagation und die Evaluierung der Ergebnisse. Die Leistungsfähigkeit der Klassifikation demonstrierten wir anhand der drei Datensätze mit Bildern realer Objekte: (i) der MSRC 23-Klassen Datensatz, (ii) der 7-Klassen Corel Datensatz und (iii) der 7-Klassen Sowerby Datensatz. Die Ergebnisse sind unter Berücksichtigung der Verfahren und des Umfangs der Merkmale zufriedenstellend.

Die Arbeit betrachtet den kompletten Ablauf einer Klassifikation und bildet somit einen soliden Grundbaustein für die Segmentierung mit einem linearen probabilistischen diskriminativen Modell. Sie bietet die Möglichkeit das Programm an jedem Punkt beliebig zu erweitern.

7.2 Ausblick

Die semantische Segmentierung der Bilder durch den gelernten Klassifikator erfolgte im Falle von gut trennbaren Merkmalen der Klassen stets gut. Der Klassifikator stößt jedoch an einigen Stellen an seine Grenzen. Zum Beispiel stellt es ein großes Problem für die Klassifikation dar, wenn Objekte Merkmale anderer Objekte annehmen. Dies ist der so genannte Camouflage- oder Tarneffekt, den sich zum Beispiel Tiere zu Nutzen machen, um sich vor Feinden zu schützen. Dieses Problem tritt sehr häufig im Corel Datensatz auf, da dieser aus Bildern von wild lebenden afrikanischen und arktischen Tieren besteht. Abbildung 7.1a und 7.1b zeigen zwei Beispiele, bei denen dieses Problem

besonders deutlich wird. Es ist sehr schwierig, die Merkmale für die Klasse „Cd“ oder „Tee“ zu lernen, die sich von den Klassen „Nahrungsmittel“ und „Blumen“ unterscheiden.



(a) Tarnung der Klasse „Cd“



(b) Tarnung der Klasse „Tee“

Abbildung 7.1: Die Tarnung der Klassen „Cd“ mit den Merkmalen der Klasse „Nahrungsmittel“ und die Tarnung der Klasse „Tee“ mit den Merkmalen der Klasse „Blume“ stellt ein großes Problem in der Bildsegmentierung dar. Die Objekte nehmen die Merkmale anderer Objekte an und können somit falsch klassifiziert werden.

Allein durch Bildinformationen scheint es nahezu unmöglich, diese Klassen zu unterscheiden. Eine mögliche Lösung könnte die Hinzunahme von 3D-Informationen darstellen, sodass nicht mehr nur Bilder, sondern 3D-Oberflächen klassifiziert werden.

Ein weiteres Problem stellen Klassen dar, die gleiche Objekte enthalten. Abbildung 7.2a-7.2c zeigt drei Objekte aus dem MSRC Datensatz, die Fenster enthalten. Die Klasse „Fenster“ gibt es nicht als alleinige Klasse, sodass das Objekt zu verschiedenen Klassen gehören kann.

Eine Lösung ist, die Label in den Kontext des Bildes einzuordnen. Werden die Klassen „Auto“ und „Flugzeug“ im Bild detektiert und umschließen die Label der Klasse „Auto“ die des Flugzeugs, ist es sehr wahrscheinlich, dass das Fenster der falschen Klasse zugeordnet wurde. Dieser Problemstellung widmet sich zum Beispiel Shotton u. a. (2006), indem er ein diskriminatives Modell entwickelte, welches die Informationen über die Position, die Farbe, die Form, die Textur und den Kontext der gelabelten Regionen in ein Conditional Random Field einbettet.

Das Conditional Random Field bestimmt die a posteriori Wahrscheinlichkeiten der Label bei gegebenem Bild und erlaubt, die Merkmale mit einzubeziehen. Ähnlich dem Markov

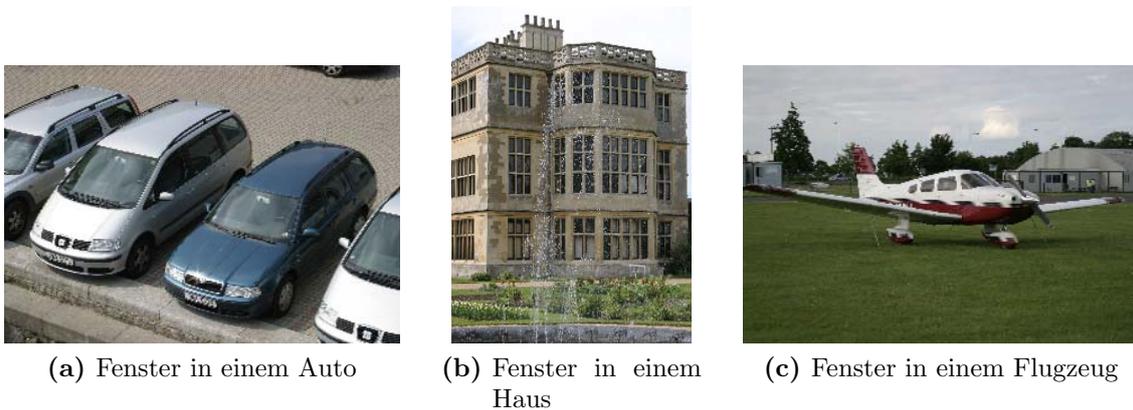


Abbildung 7.2: Das Objekt „Fenster“ kann zu verschiedenen Klassen, wie zum Beispiel „Auto“, „Gebäude“ oder „Flugzeug“ gehören.

Random Field betrachtet es die Nachbarschaften der Pixel und lässt deren Informationen zum Beispiel in Form eines Potts-Modells einfließen. Die Erweiterung des Markov Random Fields, welches wir in dieser Arbeit verwendeten, auf ein Conditional Random Field stellt eine sinnvolle und vielversprechende Erweiterung dar. Die Conditional Random Fields können zusätzlich auf höhere Ordnungen erweitert werden und so noch bessere Ergebnisse erzielen, wie Kohli u. a. (2008) zeigt.

Das lineare probabilistische diskriminative Modell lernten wir mit verschiedenen Merkmalen wie Farbe, Textur, Position und Form von Superpixeln und dem Kontext. Statt die Filterantworten als Merkmale zu verwenden, könnten wir Histogramme für Texturen erstellen, wie in Leung und Malik (2001) beschrieben. Das Verfahren dazu beschrieben wir in Kapitel 3.2.3. Für das Lernen der Texturen stehen bereits große Datenbanken frei zur Verfügung, die viele klassenspezifische Variationen abdecken. Die Erstellung der Histogramme der Trainingsdaten würde allerdings so mit externen Daten und nicht allein mit den zur Verfügung stehenden erfolgen.

Das Trainieren des Modells erfolgte mit dem Gradientenabstiegsverfahren. Eine bessere Alternative stellt das Newton-Verfahren dar, welches wir in dieser Arbeit nicht verwendeten. Das Newton Verfahren hat den Vorteil, dass die Schrittweite durch die Hessematrix bestimmt ist und es somit schneller konvergiert. Dadurch verringert sich die Laufzeit, was bei Tests zur Entwicklung von Modellen sehr nützlich ist.

Die Betrachtungen dieser Arbeit umfassen alle notwendigen Schritte für das Lernen eines Klassifikators, sodass mit ihr und dem dazu entwickelten Programm ein Grundbaustein

für weiterführende Entwicklungen im Bereich der semantischen Bildsegmentierung gelegt ist.

A Ableitungen

1.1 Ableitung der a posteriori Wahrscheinlichkeiten nach den Aktivierungen

Gegeben sind die a posteriori Wahrscheinlichkeiten

$$p(\mathcal{C}_k|\phi) = g_k(\phi) \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

mit den Aktivierungen a_j

$$a_k = \ln p(\phi|\mathcal{C}_k) p(\mathcal{C}_k) = \mathbf{w}_k^T \phi. \quad (\text{A.1})$$

Für die erste Ableitung für $j = k$ wird zunächst der Nenner umgeschrieben:

$$g_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} = \frac{\exp(a_k)}{\sum_{j \neq k} \exp(a_j) + \exp(a_k)} \quad (\text{A.2})$$

und danach abgeleitet

$$\frac{\partial g_k}{\partial a_k} = \frac{\exp(a_k) \left(\sum_{j \neq k} \exp(a_j) + \exp(a_k) \right) - \exp(a_k) \exp(a_k)}{\left(\sum_j \exp(a_j) \right)^2} \quad (\text{A.3})$$

$$= \frac{\exp(a_k)}{\sum_j \exp(a_j)} - \frac{\exp(a_k)^2}{\left(\sum_j \exp(a_j) \right)^2} \quad (\text{A.4})$$

$$= g_k(1 - g_k). \quad (\text{A.5})$$

Für die erste Ableitung für $j \neq k$ wird erneut zunächst der Nenner umgeschrieben:

$$g_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} = \frac{\exp(a_k)}{\sum_{l \neq j} \exp(a_l) + \exp(a_j)} \quad (\text{A.6})$$

und danach abgeleitet

$$\frac{\partial g_k}{\partial a_j} = - \frac{\exp(a_k) \exp(a_j)}{\left(\sum_{l \neq j} \exp(a_l) + \exp(a_j) \right)^2} \quad (\text{A.7})$$

$$= - \frac{\exp(a_k) \exp(a_j)}{\left(\sum_j \exp(a_j) \right)^2} \quad (\text{A.8})$$

$$= -g_k g_j. \quad (\text{A.9})$$

Die Ableitungen von g_k nach allen Aktivierungen a_j lauten damit allgemein:

$$\frac{\partial g_k}{\partial a_j} = g_k (I_{kj} - g_j) \quad (\text{A.10})$$

mit I_{kj} als Einträge der Einheitsmatrix.

1.2 Ableitung der Kreuzentropie-Fehlerfunktion nach den Parametervektoren

Die erste Ableitung können wir wie folgt erhalten: Zunächst wird die Kettenregel angewandt:

$$\frac{\partial F}{\partial \mathbf{w}_j} = \sum_{n=1}^N \frac{\partial F}{\partial g_{jn}} \frac{\partial g_{jn}}{\partial a_{jn}} \frac{\partial a_{jn}}{\partial \mathbf{w}_j}. \quad (\text{A.11})$$

Für die Bestandteile von (A.11) erhalten wir unter Verwendung (A.10) von folgenden Ableitungen:

$$\frac{\partial F}{\partial g_{jn}} = \sum_{n=1}^N -\frac{t_{jn} (1 - t_{jn})}{g_{jn} (1 - g_{jn})} \quad (\text{A.12})$$

$$\frac{\partial g_{jn}}{\partial a_{jn}} = \sum_{n=1}^N g_{jn} (1 - g_{jn}) \quad (\text{A.13})$$

$$\frac{\partial a_{jn}}{\partial \mathbf{w}_j} = \frac{\partial (\mathbf{w}_j^T \boldsymbol{\phi}_n)}{\partial \mathbf{w}_j} = \sum_{n=1}^N \boldsymbol{\phi}_n. \quad (\text{A.14})$$

Für den Gradienten ∇F ergibt sich durch Multiplikation aller Bestandteile

$$\nabla F = \sum_{n=1}^N \left(-\frac{t_{jn} (1 - t_{jn})}{g_{jn} (1 - g_{jn})} \right) g_{jn} (1 - g_{jn}) \boldsymbol{\phi}_n, \quad (\text{A.15})$$

$$= \sum_{n=1}^N -t_{jn} (1 - g_{jn}) + (1 - t_{jn}) g_{jn} \boldsymbol{\phi}_n, \quad (\text{A.16})$$

$$= \sum_{n=1}^N (g_{jn} - t_{jn}) \boldsymbol{\phi}_n. \quad (\text{A.17})$$

B Arbeitsfluss

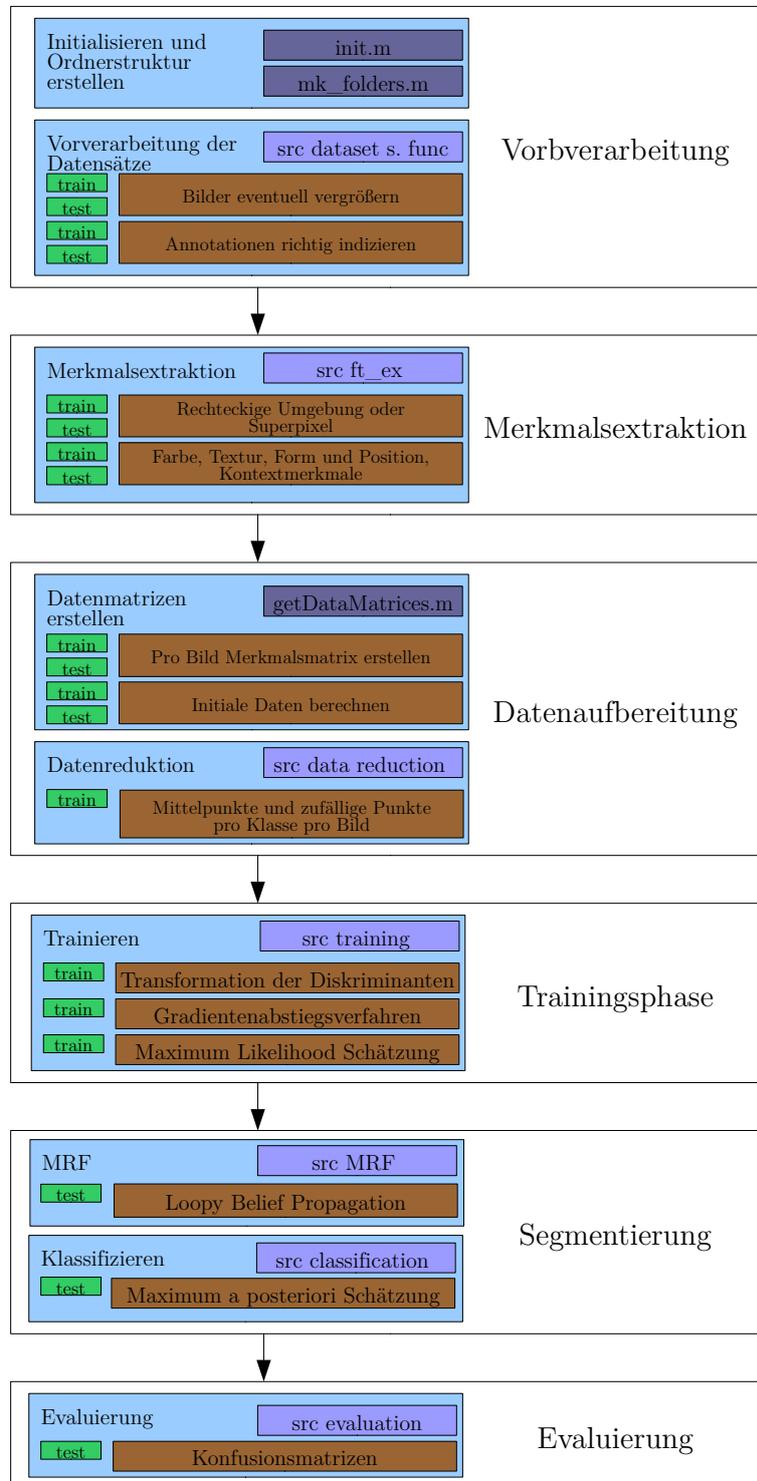


Abbildung B.1: Der Arbeitsfluss ist unterteilt in die Unterprogramme der Vorverarbeitung, der Merkmalsextraktion, der Datenaufbereitung, der Trainingsphase, der Segmentierung und der Evaluierung.

C Tabellen

3.1 Variation der Schrittweite

Schrittweite	Superpixel	Kreuzentropie	Falsche Targets		Klassifikationserfolg		Laufzeit [min]
			#	[%]	Ohne LBP [%]	Mit LBP [%]	
10^{-8}		6928	2446	42,0	64,9	65,0	3046
		6935	2377	40,8	65,2	65,2	3424
	x	7094	2484	42,7	63,6	66,5	2864
	x	6955	2203	37,8	66,1	66,5	2929
10^{-7}		5398	1462	25,1	83,5	83,8	2982
		5404	1492	25,6	83,3	83,6	3030
	x	5291	1439	24,7	83,8	84,1	2905
	x	5156	1326	22,7	86,2	86,6	2836
10^{-6}		11098	2280	39,1	67,4	67,5	3955
		11047	2223	38,2	71,8	71,9	2999
	x	15617	2391	41,1	63,5	63,9	3060
	x	12045	2136	36,7	68,8	69,1	2921

Tabelle C.1: Die Schrittweite des Gradientenabstiegsverfahren liefert bei 10^{-7} die besten Klassifikationserfolge. Die Tests führten wir mit dem Sowerby Datensatz und jedem 1000sten Pixel durch.

3.2 Variation der Variation der Merkmale

Superpixel	Merkmale	Kreuzentropie	Falsche Targets #	Klassifikationserfolg		Laufzeit
				Ohne LBP [%]	Mit LBP [%]	
x	Mittelwerte	5393	1481	25,43	83,6	37 Min.
x		5251	1429	24,54	84,8	42 Min.
x	Histogramme	7623	2793	47,96	56,4	41 Min.
x		7667	2791	47,92	57,3	41 Min.
x	alle	5291	1439	24,7	84,1	48 Min.
x		5156	1326	22,7	86,6	47 Min.
	alle	5398	1462	25,1	83,8	50 Min.
		5404	1492	25,6	83,6	51 Min.

Tabelle C.2: Die Tabelle zeigt mögliche Variationen der verwendeten Merkmale beim Sowerby Datensatz mit einer Schrittweite von 10 – –7 und jedem 1000sten verwendeten Pixel.

3.3 Variation der Punktzahl

Verwendete Pkt.	Kreuz- entropie	Normalisierte Kreuzentropie	Falsche Targets #	Klassifikationserfolg		Laufzeit
				Ohne LBP [%]	Mit LBP [%]	
Jeder 50ste	69640	0,69	17613	17,6	86,8	148 Min.
Jeder 100ste	36557	0,72	9683	19,2	85,2	115 Min.
Jeder 500ste	8897	0,83	2442	22,7	84,1	58 Min.
Jeder 1000ste	5248	0,90	1425	24,5	84,8	40 Min.
Jeder 2000ste	3415	1,02	990	29,5	84,7	37 Min.
Jeder 5000ste	2209	1,17	693	36,6	84,5	35 Min.
Jeder 10000ste	1818	1,30	625	44,6	79,2	36 Min.
Jeder 20000ste	1562	1,33	542	46,2	78,0	32 Min.

Tabelle C.3: Die Tabelle zeigt die erzielten Ergebnisse bei Variation der verwendeten Punktzahl beim Sowerby Datensatz mit einer Schrittweite von 10^{-7} .

D Konfusionsmatrizen

4.1 Sowerby Datensatz

Himmel	95,9	1,6	2,5			
Vegetation	5,7	84,5	9,8			
Fahrb.-mark.	18,0	53,3	28,7			
Straße		13,6	86,4			
Gebäude	4,9	51,3	43,8			
Straßenobj.	36,9	42,7	20,4			
Auto	2,1	57,7	40,2			
	Himmel	Vegetation	Fahrb.-mark.	Straße	Gebäude	Straßenobj.

- (a) Konfusionsmatrix mit Rechteckmerkmalen ohne Anwendung der Loopy Belief Propagation: Klassifikationserfolg KE = 82,87%

Himmel	95,5	1,8	2,6	0,1		
Vegetation	5,5	84,2	10,3			
Fahrb.-mark.	23,1	45,8	31,2			
Straße		13,6	86,4			
Gebäude	4,5	50,9	44,6			
Straßenobj.	36,1	43,2	20,7			
Auto	2,1	57,1	40,7			
	Himmel	Vegetation	Fahrb.-mark.	Straße	Gebäude	Straßenobj.

- (b) Konfusionsmatrix mit Rechteckmerkmalen mit Anwendung der Loopy Belief Propagation: Klassifikationserfolg KE = 83,07%

Abbildung D.1: Konfusionsmatrizen des Sowerby Datensatzes

Himmel	93,5	4,3		2,2			
Vegetation	4,9	86,6		8,5			
Fahrb.-mark.		67,5		32,5			
Straße		9,3		90,7			
Gebäude	2,6	54,9		42,5			
Straßenobj.	26,3	62,8		10,9			
Auto	0,3	74,3		25,4			
	Himmel	Vegetation	Fahrb.-mark.	Straße	Gebäude	Straßenobj.	Auto

- (c) Konfusionsmatrix mit Superpixelmerkmalen ohne Anwendung der Loopy Belief Propagation: Klassifikationserfolg KE = 84,63%

Himmel	93,1	4,6		2,3			
Vegetation	4,8	85,4		9,7			
Fahrb.-mark.		60,2		39,8			
Straße		9,3		90,7			
Gebäude	2,4	54,9		42,7			
Straßenobj.	26,1	63,4		10,5			
Auto		81,0		19,0			
	Himmel	Vegetation	Fahrb.-mark.	Straße	Gebäude	Straßenobj.	Auto

- (d) Konfusionsmatrix mit Superpixelmerkmalen mit Anwendung der Loopy Belief Propagation: Klassifikationserfolg KE = 85,12%

Abbildung D.1: Konfusionsmatrizen des Sowerby Datensatzes

4.2 Corel Datensatz

Nash./Nilpf.	45,6		23,5	1,6	7,5	21,9	
Polarbär		4,2		16,3	1,1	78,4	
Wasser	0,3		55,9	29,8	12,1	1,9	
Schnee		0,1		80,2	1,4	18,3	
Vegetation	8,0	11,0	48,2	16,2	13,8	2,7	
Erdboden	1,3		15,9		0,7	82,0	
Schnee			100,0				
			Nash./Nilpf.	Polarbär	Wasser	Schnee	Vegetation
							Erdboden
							Schnee

(a) Konfusionsmatrix mit Rechteckmerkmalen ohne Anwendung der Loopy Belief Propagation: Klassifikationserfolg KE = 58,0%

Nash./Nilpf.	45,9		23,3	1,5	8,2	21,2	
Polarbär		3,8		16,7	1,1	78,4	
Wasser	0,2		55,9	29,9	12,3	1,7	
Schnee				80,5	1,3	18,2	
Vegetation	7,8	10,1	48,8	17,1	13,7	2,6	
Erdboden	1,3		16,2		0,8	81,8	
Schnee			100,0				
			Nash./Nilpf.	Polarbär	Wasser	Schnee	Vegetation
							Erdboden
							Schnee

(b) Konfusionsmatrix mit Rechteckmerkmalen mit Anwendung der Loopy Belief Propagation: Klassifikationserfolg KE = 57,9%

Abbildung D.2: Konfusionsmatrizen des Corel Datensatzes

4.3 MSRC Datensatz

Gebäude	20,3	3,1	9,0	1,9		1,8	11,5	0,4	0,7	10,0	0,8	9,7			0,6	2,8	2,6	15,9	0,7	1,9	5,4	1,1	
Gras	3,5	86,7	2,3	0,6		0,2	2,4		0,8	1,0						0,7		1,5	0,1		0,1		
Baum	7,0	22,9	29,6	3,3		2,7	6,4			0,8	0,4	1,6	0,5	2,5	0,2	0,4	12,1	1,2	1,8	0,1	0,4	6,0	
Kuh	10,2	32,4	0,2	27,8			6,9	0,3		1,3	0,5	0,7	3,2	0,6	8,7		1,5	3,8	1,0	1,0			
Pferd																							
Schaf	29,7	33,3	8,2	0,1		3,4		4,1								0,1		21,1					
Himmel	1,1	0,5	0,1			5,9	63,7			17,0	0,1	0,4	1,2	0,3	0,1	0,1	0,4					0,1	
Berg	10,7	11,4					24,8		3,5	11,7	1,4							34,1				2,3	
Flugzeug	12,6	19,5	14,0	0,2		8,0	0,2		2,1	0,9	0,1	4,6	4,9	1,3	0,1		10,9			0,7	2,2	0,1	
Wasser	8,9	5,4	13,5	0,2		3,6	20,1	0,7	0,1	28,5	1,6	2,8	0,6	2,4			1,0		8,7	0,8		0,3	0,8
Gesicht	5,4	1,7	2,9	14,6		15,2	4,4			0,1	4,5	8,8		2,2	4,2		13,3		12,6	0,2	0,3	6,6	3,1
Auto	9,4	0,3	18,1	6,8		0,2	12,3		0,1	13,8	4,6	9,3	4,3	3,6	0,7	0,2	1,4	7,6	4,9	0,5	0,9	1,0	
Fahrrad	18,9	0,9	9,8	0,5		7,5	2,9		0,4	0,1	0,8	6,3		6,1	1,3		10,7	8,9	1,7		1,3	22,0	
Blume	13,2	5,4	12,4	11,6		0,9	8,9	0,2	2,1		10,2		0,3	18,2			7,9		6,4			1,7	0,5
Zeichen	14,1	13,4	14,9	4,0		0,1	15,0	0,1		0,6	17,2	2,9			3,5	1,5	8,7		3,9			0,1	
Vogel	18,3	19,9	12,4	3,1		0,8	13,6	1,5	0,4	4,7		1,1		1,0			11,6		10,4			1,1	
Buch	5,7	3,1	4,6	12,3		2,2	4,0	0,3	1,5	5,2	2,6	0,8	9,3	7,7	3,5	0,4	19,0		14,7	0,3	2,8		
Stuhl	22,4	9,5	3,0	4,3		0,2	3,9			8,5		1,2		2,4			14,6	2,7	15,5	1,0		9,5	1,5
Straße	23,9	0,4	0,5	6,1		8,4	8,7	0,8	0,5	6,6	0,6	5,3	0,2	0,5	0,2		1,8	1,2	29,9		0,4	1,7	2,2
Katze	7,0	1,0	3,6	4,5		11,9	1,6	0,3	4,2	4,9	1,0	11,6	12,5	4,9	1,3		6,6		22,6	0,7			
Hund	8,4	2,3	4,4	10,2		2,3	17,2	4,3		0,9	1,9	1,0	1,7	0,3	2,7		3,4	0,1	25,7	5,1	1,6	1,6	4,9
Körper	13,7	4,4	2,2	14,5		0,1	1,1	4,5		7,0	9,3	9,8	7,7	1,5			3,6	1,5	15,9	1,3		0,8	0,9
Boot	26,8	9,3	0,2	8,5		0,3	9,2			4,4	12,1		2,6	4,9			8,3	6,3	2,0	0,2		4,8	

(a) Konfusionsmatrix mit Rechteckmerkmalen ohne Anwendung der Loopy Belief Propagation: Klassifikationserfolg KE = 30,1%

Abbildung D.3: Konfusionsmatrizen des MSRC Datensatzes

Gebäude	20,7	2,9	8,9	1,9		1,8	11,5	0,3	0,7	9,9	0,8	9,8			0,7		2,7	2,5	16,3	0,7	1,8	5,2	1,1
Gras	3,5	87,0	2,2	0,5		0,1	2,4		0,8	1,0						0,7		1,5	0,1		0,1		
Baum	6,9	23,1	29,9	3,1		2,7	6,4			0,8	0,4	1,6	0,5	2,5	0,2	0,3	12,0	1,2	1,8	0,1	0,4	6,1	
Kuh	10,8	33,2		27,8			7,0	0,1		1,0	0,3	0,5	3,2	0,5	8,6		1,4	3,7	0,9	1,0			
Pferd																							
Schaf	29,3	34,9	8,0	0,1		3,4		4,0											20,4				
Himmel	1,0	0,5	0,1			5,9	63,9			17,0	0,1	0,4	1,1	0,3	0,1	0,1	0,4		9,1				0,1
Berg	10,6	11,4					24,7		3,5	11,7	1,3								34,6				2,3
Flugzeug	12,5	20,2	14,0	0,2		8,0	0,2		2,0	0,9		4,4	4,7	1,2	0,2		11,2		17,3		0,7	2,2	0,1
Wasser	8,9	5,4	13,7	0,2		3,4	20,1	0,7	0,1	28,8	1,5	2,9	0,6	2,4			1,0		8,7	0,7		0,2	0,8
Gesicht	5,5	1,4	3,0	14,9		15,8	4,4			0,1	4,4	8,8		2,1	4,1		12,9		12,9	0,1	0,2	6,7	2,9
Auto	9,5	0,2	18,1	6,9		0,1	12,3		0,1	14,1	4,8	9,2	4,1	3,7	0,7	0,1	1,3	7,5	4,9	0,5	0,9	1,0	
Fahrrad	19,7	0,9	9,7	0,5		7,5	2,8		0,3	0,1	0,7	6,6		6,0	1,0		10,6	8,3	1,4		1,4	22,7	
Blume	13,0	5,2	12,3	11,8		0,8	8,9	0,2	2,2		10,4		0,1	18,4			7,9		6,5			1,8	0,4
Zeichen	14,5	12,9	14,8	3,9			15,0	0,1		0,5	17,5	2,8			3,7	1,5	9,0		3,8				
Vogel	17,6	20,9	12,4	3,0		0,7	13,7	1,2	0,3	4,7		1,1		1,2			11,2		11,0			1,0	
Buch	5,9	2,9	4,8	12,2		2,2	3,8	0,3	1,4	5,1	2,5	0,7	9,3	7,8	3,5	0,5	19,5		14,6	0,3	2,6		
Stuhl	22,7	9,8	3,1	4,4		0,2	3,8			8,5		1,2		2,2			14,5	2,6	15,5	0,9		9,3	1,4
Straße	24,0	0,4	0,5	6,1		8,3	8,9	0,8	0,5	6,5	0,5	5,3	0,2	0,5	0,2		1,8	1,1	30,1		0,4	1,7	2,1
Katze	6,8	0,8	3,4	4,3		11,8	1,5	0,2	4,3	5,1	0,8	11,8	12,6	4,8	1,1		6,6		23,3	0,8			
Hund	8,5	2,3	4,8	10,5		2,2	17,3	4,2		0,8	1,8	0,9	1,6	0,3	2,7		3,4	0,1	26,0	4,9	1,5	1,5	4,7
Körper	14,0	4,2	2,2	14,6		0,2	1,0	4,3		6,9	9,2	10,0	7,5	1,6			3,4	1,5	16,5	1,2		0,8	0,9
Boot	27,6	9,4	0,3	8,3		0,2	9,4			4,0	12,4		2,4	4,7			9,1	6,1	1,9	0,2		4,0	
	Gebäude	Gras	Baum	Kuh	Pferd	Schaf	Himmel	Berg	Flugzeug	Wasser	Gesicht	Auto	Fahrrad	Blume	Zeichen	Vogel	Buch	Stuhl	Straße	Katze	Hund	Körper	Boot

(b) Konfusionsmatrix mit Rechteckmerkmalen mit Anwendung der Loopy Belief Propagation: Klassifikationserfolg KE = 30,1%

Abbildung D.3: Konfusionsmatrizen des MSRC Datensatzes

Gebäude	26,9	22,9	11,8			23,7	4,3	1,8		0,4				0,3	7,8								
Gras	2,7	87,0	3,4			4,1				0,1				0,3	2,3								
Baum	21,2	26,6	35,2			2,0	0,1	0,3		0,1			0,7		13,8								
Kuh	1,3	74,4	4,6			6,0		0,2	0,5						2,1			11,0					
Pferd																							
Schaf		89,7				9,0		1,2															
Himmel	0,5	20,2	1,2			58,1	0,1	2,0							17,9								
Berg		48,3	1,2			42,9		4,8							2,8								
Flugzeug	35,2	21,7	15,2			10,7									17,1								
Wasser	0,7	34,4	4,1			44,9	0,3	5,8		0,5			7,5	0,1	1,7				0,1				
Gesicht	35,0	24,6	9,2	1,2		4,5	0,3	4,6		2,2	0,7		17,6						0,3				
Auto	25,5	17,0	35,1			8,4		5,8	0,1	1,4	0,1				6,5								
Fahrrad	41,4	27,8	23,0			0,2									7,5								
Blume	6,6	45,9		6,2	0,9	2,6		11,9		0,5			20,2					5,2					
Zeichen	38,3	14,5	11,4			19,9		13,2	0,1					2,4				0,1	0,2				
Vogel	4,7	72,8	5,4			0,2		1,8						15,1									
Buch	9,4	44,1		12,2	0,2	3,8		1,1	3,5	19,0			6,7										
Stuhl	19,9	53,6	3,2	0,3		3,9		6,6		2,5		0,4	0,6	1,6	7,4								
Straße	15,1	34,0	6,5			26,3	0,7	5,0		0,3	0,3		1,0	0,7	10,1								
Katze	13,8	16,3	37,4			5,7		4,7					15,1	2,7	4,4								
Hund	12,8	24,5	3,4			8,5		0,5		8,9			30,3	10,8	0,2								
Körper	11,8	30,1	17,4	2,4		6,0	9,6	2,3	0,5	0,9	3,6		11,0	0,2				3,2	1,1				
Boot	23,3	36,1	4,1			23,3							13,2										
	Gebäude	Gras	Baum	Kuh	Pferd	Schaf	Himmel	Berg	Flugzeug	Wasser	Gesicht	Auto	Fahrrad	Blume	Zeichen	Vogel	Buch	Stuhl	Straße	Katze	Hund	Körper	Boot

(c) Konfusionsmatrix mit Superpixelmerkmalen ohne Anwendung der Loopy Belief Propagation: Klassifikationserfolg $KE = 35,5\%$

Abbildung D.3: Konfusionsmatrizen des MSRC Datensatzes

Gebäude	26,8	23,0	11,8			23,9	4,3	1,6		0,3				0,3	7,9				
Gras	2,7	87,4	3,2			4,1				0,1				0,2	2,3				
Baum	21,2	26,6	35,2			2,0	0,1	0,3					0,7		13,8				
Kuh	1,2	75,2	4,3			6,1		0,1	0,4						1,9			10,8	
Pferd																			
Schaf		90,0				9,0		1,0											
Himmel	0,5	20,7	1,1			57,8		1,9							18,0				
Berg		49,4	0,6			41,9		5,1							3,0				
Flugzeug	36,0	21,7	14,9			10,3									17,1				
Wasser	0,6	34,7	4,0			44,7	0,3	5,7		0,6			7,5	0,1	1,8				0,1
Gesicht	35,2	24,7	9,1	0,9		4,3	0,3	4,5		2,1	0,7		18,0						0,2
Auto	25,9	16,9	35,3			8,3		5,7	1,4	0,1					6,5				
Fahrrad	42,0	28,0	22,6			0,2									7,2				
Blume	6,7	46,2		6,1	0,8	2,5		11,9		0,5			20,2					5,1	
Zeichen	38,9	14,5	11,1			20,3		13,0							2,3				
Vogel	4,6	73,2	5,4			0,1		1,5							15,2				
Buch	9,4	44,2		12,3	0,2	3,7		1,1	3,4	19,1			6,5						
Stuhl	20,3	53,7	3,2	0,2		3,9		6,5		2,5		0,4	0,6	1,4	7,4				
Straße	15,1	34,2	6,5			26,2	0,7	5,1		0,3	0,3		1,0	0,6	10,1				
Katze	14,0	16,4	37,4			5,6		4,6					15,1	2,6	4,4				
Hund	12,9	24,6	3,2			8,4		0,5		8,8			30,5	10,8	0,2				
Körper	11,8	30,1	17,2	2,3		6,0	9,7	2,3	0,5	0,9	3,7		11,1	0,1				3,1	1,1
Boot	23,3	37,1	3,6			23,5							12,5						

(d) Konfusionsmatrix mit Superpixelmerkmalen mit Anwendung der Loopy Belief Propagation: Klassifikationserfolg KE = 35,5%

Abbildung D.3: Konfusionsmatrizen des MSRC Datensatzes

Abbildungsverzeichnis mit Referenzen

1.1	Zuordnung der Pixel zu einer Klasse am Beispiel eines Bildes aus dem Corel Datensatz	1
1.2	Bilder aus den in dieser Arbeit betrachteten Datensätze mit verschiedenartigen Motiven	2
1.3	Leopard: http://de.wikipedia.org/wiki/Bild:Luipaard_kop.jpg und Strandball: http://www.beatportal.com/uploads/news/1206071535_beachball.jpg	3
1.4	Roboter: http://anybots.com/dishwasher2.jpg , Röntgenprüfgerät: http://upload.wikimedia.org/wikipedia/commons/6/61/Luggage_screening_at_VTBS.jpg	4
2.1	Diskriminante im dreidimensionalen Raum: Duda(2001, Kapitel 5, Seite 217), bearbeitet	13
2.2	Sigmoid Funktion	17
2.3	Die konvexe Eigenschaft der Fehlerfläche wird besonders deutlich, wenn wir einen Querschnitt von dieser erstellen. Die Ordinatenachse zeigt die Größe des Fehlers.	21
2.4	Die Abbildung zeigt einen Querschnitt der Fehlerfläche bei einem konvexen Optimierungsproblem und verdeutlicht das Problem bei einer zu großen Wahl der Schrittweite. Die Abszissenachse gibt die Parameterwerte \mathbf{W} und die Ordinatenachse den Wert der Fehlerfunktion F an. Die Länge der schwarzen Pfeile gibt den Wert des Produktes der Schrittweite mit der Ableitung an.	22
3.1	Die Erstellung der Superpixel eines Beispielbilds aus dem Sowerby Datensatz erfolgt mit dem NCut-Code nach Shi und Malik (2000).	28
3.2	Farbton im HSV Farbraum: http://www.verwaltung.uni-mainz.de/edv/projekte/doku/gimp/grokking/node52.html	30
3.3	Sättigung im HSV Farbraum: http://www.verwaltung.uni-mainz.de/edv/projekte/doku/gimp/grokking/node52.html	31
3.4	Helligkeit im HSV Farbraum: http://www.verwaltung.uni-mainz.de/edv/projekte/doku/gimp/grokking/node52.html	32
3.5	Gegenfarben: http://de.wikipedia.org/w/index.php?title=Bild:Opponent_colors.svg&filetimestamp=20060415005330	32

3.6	Lab Farbraum: http://www.heise.de/foto/Mehr-Bits-fuer-Farbe--/zoom/106500/13	34
3.7	Schachbrett: http://weed-e.de/Galerie/schachbrett.jpg , Zebra: http://upload.wikimedia.org/wikipedia/commons/b/b8/N2_zebra.jpg , Kuh: http://www.weltexpress.info/pics/schwarz-bunte-kuh_01.jpg	36
3.8	Leung-Malik Filterbank: http://www.robots.ox.ac.uk/~vgg/research/texclass/figs/with/lmfilters.jpg	37
3.9	Bild aus dem Sowerby Datensatz, anhand dessen wir die Strukturmerkmale des Farbton- und des Sättigungskanals erstellen	39
3.10	Dominante Strukturmerkmale der Farbton- und Sättigungskanals	40
3.11	Texton-Karte	41
3.12	Bild eines Schafes auf der Wiese aus dem MSRC Datensatzes zur Verdeutlichung der Kontextmerkmale	42
4.1	Aufbau eines Markov Random Field	44
4.2	Aufbau eines Faktorgraphs	52
4.3	Markov Random Fields mit Loops (Zyklen)	54
4.4	Zweierkette als Faktorgraph	55
4.5	Versendeplan der Nachrichten im Markov Random Field	57
6.1	Synthetischer Datensatz mit drei Klassen	78
6.2	Synthetischer Datensatz mit fünf Klassen	79
6.3	Anzahl der falschen Zuordnungen und der Wert der Kreuzentropie aufgetragen gegen Anzahl der Iterationen für den synthetischer Datensatz mit fünf Klassen	80
6.4	Synthetischer Datensatz mit drei nicht separierbaren Klassen	81
6.5	Anzahl der falschen Zuordnungen und der Wert der Kreuzentropie aufgetragen gegen Anzahl der Iterationen des synthetischen Datensatzes mit drei nicht separierbaren Klassen	82
6.6	Starke Ähnlichkeit zwischen Klassen in Bildern des Corel Datensatzes	84
6.7	Kreuzentropie bei verschiedenen Pixelanzahlen	87
6.8	Legoburg: Institut für Geodäsie und Geoinformation Bonn, Professur für Photogrammetrie; Verteilung der Pixel im RGB und HSV Raum	91
6.9	Konfusionsmatrix des Sowerby Datensatzes mit Superpixelmerkmalen mit Anwendung der Loopy Belief Propagation	95
6.10	Ergebnisse der segmentierten Bilder aus dem Sowerby Datensatz	96

6.11	Konfusionsmatrix des Corel Datensatzes mit Superpixelmerkmalen mit Anwendung der Loopy Belief Propagation	98
6.12	Ergebnisse der segmentierten Bilder aus dem Corel Datensatz	99
6.13	Konfusionsmatrix des MSRC Datensatzes mit Superpixelmerkmalen mit Anwendung der Loopy Belief Propagation	102
6.14	Ergebnisse der segmentierten Bilder aus dem MSRC Datensatz	103
7.1	Tarnproblem	106
7.2	Fenster in Objekten von verschiedenen Klassen aus dem MSRC Datensatz	107
B.1	Der Arbeitsfluss ist unterteilt in die Unterprogramme der Vorverarbeitung, der Merkmalsextraktion, der Datenaufbereitung, der Trainingsphase, der Segmentierung und der Evaluierung.	112
D.1	Konfusionsmatrizen des Sowerby Datensatzes	116
D.1	Konfusionsmatrizen des Sowerby Datensatzes	117
D.2	Konfusionsmatrizen des Corel Datensatzes	118
D.2	Konfusionsmatrizen des Corel Datensatzes	119
D.3	Konfusionsmatrizen des MSRC Datensatzes	120
D.3	Konfusionsmatrizen des MSRC Datensatzes	121
D.3	Konfusionsmatrizen des MSRC Datensatzes	122
D.3	Konfusionsmatrizen des MSRC Datensatzes	123

Literatur

- Alpaydin, E. (2008). *Maschinelles Lernen*. Oldenbourg, R.
- Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, USA.
- Bishop, C. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.
- Boyd, S. und L. Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.
- Clifford, P. (1990). Markov random fields in statistics. In: D. J. A. W. E. G. R. Grimmett (Hrsg.), *A Volume in Honour of J.M. Hammersley*, S. 19–32. Oxford University Press.
- Duda, R., P. Hart und D. Stork (2001). *Pattern classification (Second Aufl.)*. Wiley New York.
- Durupinar, F. (2005). Classification of Textures under Various Lighting and Viewing Conditions. In: *Turkish Symposium on Artificial Intelligence & Neural Networks*.
- He, X. und R. Zemel (2008). Latent topic random fields: Learning using a taxonomy of labels. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, S. 1–8.
- He, X., R. Zemel und M. Carreira-Perpinan (2004). Multiscale Conditional Random Fields for Image Labeling. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Volume 2*. IEEE Computer Society; 1999.
- Kohli, P., L. Ladicky und P. Torr (2008). Robust higher order potentials for enforcing label consistency. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. CVPR.
- Leung, T. und J. Malik (2001). Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. *International Journal of Computer Vision* 43(1), S. 29–44.
- MacKay, D. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

-
- Malik, J., S. Belongie, T. Leung und J. Shi (2001). Contour and Texture Analysis for Image Segmentation. *International Journal of Computer Vision* 43(1), S. 7–27.
- Normen, D. (1979). DIN 5033: Farbmessung.
- Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In: *Proceedings of the AAAI National Conference on AI*, S. 133–136.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers.
- Potts, R. (1952). Some generalized order-disorder transformations. In: *Proceedings of the Cambridge Philosophical Society*, Volume 48, S. 106–109.
- Schenk, T. (1999). *Digital Photogrammetry*, Volume 1. Volume I, TerraScience.
- Shi, J. und J. Malik (2000). Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8), S. 888–905.
- Shotton, J., J. Winn, C. Rother und A. Criminisi (2006). TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-class Object Recognition and Segmentation. In: *Computer Vision – ECCV 2006*, Volume 3951 of *Lecture Notes in Computer Science*, S. 1–15. Springer.

Stichwortverzeichnis

- Aktivierungsfunktion, 16
- Bayestheorem, 10
- Clique, 45
- Diskriminante, 11
- Dominante Strukturmerkmale, 39
- Energiefunktion, 46
- Faktorgraph, 52
- Faktorknoten, 51
- Farbraum
 - HSV-Farbraum, 29
 - Lab-Farbraum, 32
 - RGB-Farbraum, 29
- Filterantworten, 38
- Filterbank, 36
- Generalisierte lineare Modelle, 16
- Gradientenabstiegsverfahren, 20
 - Wahl der Schrittweite, 22
- Homogene Größen, 15
- Klassifikation, 9
 - Diskriminantenfunktionen, 11
 - Diskriminatives Modell, 11
 - Generatives Modell, 11
- Konfusionsmatrix, 72
- Kontextmerkmale, 42
- Kreuzentropie, 19
- Label, 10
- Linear separierbar, 10
- Logistische Regression, 18
- Loopy Belief Propagation, 54
- Markov Random Field, 44
- Maximum Likelihood Schätzung, 19
- Maximums-Aktivierungsmodells, 18
- Merkmalsvektoren, 9
- Message Passing, 50
- Newton-Raphson Abstiegsverfahren, 20
- one-versus-one Klassifikator, 14
- One-versus-the-rest Klassifikator, 14
- Positionsmerkmale, 35
- Semantische Bildsegmentierung, 2
- Sigmoid-Funktion, 17
- Softmax-Transformation, 16
- Squashing-Funktionen, 17
- Superpixel, 27
- Texton-Karte, 40
- Texturmerkmale, 35
- Versendevorschrift, 54
- Zielvektor, 10

