Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III

Bachelorarbeit

Klassifikation sequentieller 3D-Laserscans mit Conditional Random Fields

Wadim Kehl

Erstgutachter: PD Dr. Volker Steinhage
Zweitgutachter: Dr. Kristian Kersting
Bonn, 30. September 2010

Danksagung

Ich möchte mich recht herzlich bei Herrn PD Dr. Volker Steinhage sowie Jens Behley für die ausgezeichnete Betreuung bedanken. Ich danke auch Herrn Dr. Kristian Kersting für das Zweitgutachten. Ausserdem geht ein Dank an Johannes, Florian und Dominik für die amüsante Zeit in Büro A111.

Bedanken möchte ich mich auch beim Fraunhofer FKIE, dass nicht nur die Scandaten, sondern auch einen Computer zur vollen Nutzung bereitstellte.

Schlussendlich grüße und danke ich meiner Familie, insbesondere Vika und Arno, und meiner Freundin Anna für die mentale Unterstützung und das Korrekturlesen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Nutzung von 3D-Laserscannern (auch 3D-Laserentfernungsmessern) zur Abtastung und dreidimensionalen Rekonstruktion der Umgebung kann in vielen Anwendungsszenarien gewinnbringend eingesetzt werden. In manchen Domänen möchte man jedoch nicht nur die Umgebung geometrisch modellieren, sondern sie auch verstehen bzw. interpretieren. Das heißt in diesem Kontext, man möchte Objekte in der Welt identifizieren und diese damit in Klassen wie bspw. 'Mensch', 'Gebäude' oder 'Fahrzeug' unterteilen. Der Prozess der Annotierung von Daten mit (semantischen) Klassen wird Klassifikation genannt und ist nicht nur in der Robotik, sondern auch in zahlreichen anderen Gebieten wie bspw. der Computerlinguistik oder der Genforschung von großem Interesse.

Ein Beispiel ist die Erstellung von lasergestützten Umgebungsmodellen um die SLAM-Aufgabe von Robotern positiv zu beeinflussen. Der Vorteil im Gegensatz zu Umgebungsmodellen ohne Laserunterstützung ist dabei insbesondere, dass Laserdaten Informationen über die Distanz der abgetasteten Objekte liefern und damit das Umgebungsmodell robust erweitern. Der Nachteil ist, dass die Menge an 3D-Laserdaten meist zu hoch ist, um sie vollständig zu registrieren und das die Registrierung der Laserdaten in dynamischen Szenen (d.h. Objekte sich in der Szene zu verschiedenen Zeitpunkten an verschiedenen Orten befinden können) sehr problematisch sein kann. Eine Klassifikation der Daten hilft dem System, statische Landmarken in seinem Umgebungsmodell zu finden. Statische klassifizierte Objekte wie Hauswände, Schilder, Hydranten, Bordsteinkanten oder Laternenpfähle können dann genutzt werden, Scandaten korrekt zu transformieren um sie geometrisch richtig in das Umgebungsmodell einzugliedern. Auch die Klassifikation des Bodens in Asphalt oder Gras kann zur besseren Lokalisation genutzt werden [Wurm et al., 2009].

Ein anderes Anwendungsgebiet ist die Rekonstruktion dreidimensionaler Modelle von städtischen oder ländlichen Umgebungen durch luft- oder bodengetragene Laserscanner. Die Modelle können für die Stadtplanung, für präzise Navigation oder auch als Unterstützung bei Naturunglücken oder geologischen Diensten verwendet werden. Eine Klassifikation luftgestützter Aufnahmen kann dazu genutzt werden, erwünschte Objekte (wie bspw. Häuser, Straßen, Vegetation) und unerwünschte Objekte (Autos, Menschen) in den Daten zu identifizieren und voneinander zu separieren [Shapovalov et al., 2010]. Ferner können auf identifizierten Häusern Strukturen wie Schornsteine, Gauben, Dachflächen oder

Satellitenschüssel erkannt werden, um die Rekonstruktion beliebig detailliert zu gestalten [Behley and Steinhage, 2009], [Sampath and Shan, 2010]. Bei terrestrischen Aufnahmen können, aufgrund der geänderten Perspektive, Straßen und Häuser besser modelliert werden. Fassaden benötigen bspw. für die Rekonstruktion die Erkennung von Türen und Fenstern [Brenner, 2005], die mit Laserdaten fein abgetastet werden können und damit auch eine Identifikation erlauben.

Weiter ist Klassifikation nützlich bei der akkuraten Wahrnehmung bei mobilen autonomen und teil-autonomen Systemen. Ein populärer Event war die DARPA Urban Challenge, bei der (mit verschiedenen Sensoren gestützte) Fahrzeuge autonom durch abgesperrte urbane Areale navigieren mussten. Klassifikation war hier wichtig zur Erkennung der anderen Teilnehmer und Identifikation des befahrbaren Untergrundes [Urmson et al., 2008], [Montemerlo et al., 2008]. In einem realen Szenario kommen zusätzliche Anforderung wie die Erkennung von Passanten und verschiedener Fahrzeugtypen. Da autonome Fahrzeuge sich an die anderen Teilnehmer im Verkehr anpassen müssen, hilft die Klassifikation der anderen Verkehrsteilnehmer als LKWs, Busse und Fahrradfahrer, die Trajektorien aufgrund der verschiedenen Dynamiken besser zu antizipieren. Ausserdem helfen Laserdaten auch bei der Interaktion von autonomen Systemen mit Objekten. Besonders der aufkeimende Bereich der Haushaltsrobotik muss Objekte präzise manipulieren können um Kühlschränke zu öffnen oder Elektronikgeräte an Steckdosen anzuschließen. Hier hilft Klassifikation durch die Erkennung von Griffen, Kabeln und Dosen [Blodow et al., 2009], [Rusu et al., 2009b].

Bei der Nutzung von 3D-Laserscannern entstehen dichte dreidimensionale Punktwolken. Solche Punktwolken können unter Umständen zu sehr großen Datenmengen anwachsen und damit rechnerisch unhandbar werden. Ausserdem können Daten redundant sein, wenn die abgetasteten Punkte sehr nahe beieinander liegen. Andererseits können aufgrund schlechter Abtastungen einige Areale in den Punktwolken aber auch spärlich besetzt sein. Ein weiterer Faktor ist das natürliche Rauschen bei der Aufnahme der Laserdaten. Diese Überlegungen führen dazu, dass die Daten geometrisch so gespeichert werden sollten, dass redundante Daten entfallen, spärliche Informationen nicht verloren gehen und dass ein gewisses Maß an Rauschen die Daten nur schwach verfälscht.

Ein besonderer Aspekt existiert im Zusammenhang mit zeitlichen Abfolgen von Laserscans (auch: Laserabtastungen). In vielen Bereichen werden sukzessiv neue Abtastungen sofort nach der alten durchgeführt und dies hat zur Folge, dass sich ein Teilraum der vorher abgetasteten Szene auch in der neuen Abtastung befindet. Dieser Mehrwert an Information kann dann dazu genutzt werden, die modellierte Umgebung zu verfeinern oder aber auch die Klassifikation zu verbessern. Dies setzt natürlich auch voraus, dass die Daten in einer geometrischen Ordnung vorliegen, sodass die zeitliche Zuordnung performant und auch zuverlässig durchgeführt werden kann. Problematisch kann dieser Ansatz werden, wenn die Umwelt dynamisch ist (d.h. sich bewegende Objekte enthält) oder aber der Laserscanner bzw. die Datenerhebung starkem Rauschen unterworfen ist. In diesen Fällen wäre ein Rückgriff auf zeitlich weit zurückliegende Daten problematisch, da diese mitunter falsch in die zeitlich aktuelle Abtastung verrechnet werden könnten.

Diese Arbeit thematisiert die robuste Klassifikation von 3D-Laserscandaten unter Berücksichtigung intelligenter Datenverwaltung, aussagekräftiger Merkmale und die Eingliederung von zeitlichen Zusatzinformationen. Im vorliegenden Szenario der Arbeit bewegt sich eine Trägerplattform mit einem montierten 3D-Laserscanner durch ein urbanes Areal und erzeugt zeitlich unmittelbar aufeinanderfolgende Abtastungen der Umgebung. Das Ziel ist die Klassifikation von Autos, Boden und Vegetation in den resultieren dreidimensionalen Punktwolken. Aufgrund der negativen Aspekte eines zu großen Zeitfensters wird eine Beschränkung der zeitlichen Zusatzinformationen auf den Spezialfall der unmittelbaren bzw. sequentiellen Abfolge definiert.

1.1 Verwandte Arbeiten

Nach der Einführung in die möglichen Anwendungsgebiete der Klassifikation als auch der Herausforderung durch große Datenmengen sowie geometrischer Zuordnung zeitlich zusammenhängender Scans, folgt jetzt eine Zusammenfassung verwandter Arbeiten bezüglich der Klassifikation der Daten an sich. Für die Klassifikation von 3D-Laserscandaten ist die Nutzung von probabilistischen graphischen Modellen der aktuelle Stand der Forschung (siehe Kapitel 2.2. für eine genaue Erläuterung). Eine wichtige Variante eines ungerichteten graphischen Modells sind Conditional Random Fields (siehe Kapitel 2.2.2.), welche eine bedingte Wahrscheinlichkeitsverteilung repräsentieren. Die Nutzung dieser Modelle, auch in Kombination mit Datenreduktion, werden in der Robotik, in der Computer Vision als auch in der Computergraphik seit mehreren Jahren untersucht. Für die Bachelorarbeit relevante Publikationen werden nun im Einzelnen vorgestellt und nachher in Bezug zur eigenen Arbeit gesetzt.

- In [Anguelov et al., 2005] haben die Autoren die Associative Markov Networks (als Spezialfall der Conditional Random Fields) erstmalig dazu genutzt, Laserscandaten von Außen- und Innenaufnahmen zu klassifizieren und haben auf den positiven Einfluss des Nachbarschaftskontextes für das Ergebnis hingewiesen. Darüber hinaus zeigten sie den Vorteil des Ansatzes gegenüber Support Vector Machines auf. Support Vector Machines galten bis dahin als Stand der Forschung zur Klassifikation verschiedenster Daten. Als Merkmale kamen verschiedene Deskriptoren zum Einsatz, welche die lokale Umgebung eines Punktes beschreiben, darunter auch Spin-Images (siehe Kapitel 3.2).

- In der Arbeit von [Triebel et al., 2006] wurden Laserscans von Hauswänden klassifiziert. Die Autoren haben ausserdem eine adaptive kd-Baumstruktur genutzt, um die Datenmenge informationsbewahrend zu verringern. Die genutzten Merkmale waren wieder Beschreibungen der lokalen Verteilung um einen Punkt.

- In [Lim and Suter, 2007] wurden, durch Berechnung von geometrischen Stützregionen, Laserpunkte und ihre Nachbarn für den Graphen selektiv bestimmt um die Datenmenge für das Lernen und die Inferenz zu verringern.

- Die Autoren von [Rusu et al., 2009a] klassifizierten im Kontext der Haushaltsrobotik abgetastete Objekte auf einem Tisch. Nachdem die Laserdaten der Tischplatte aus dem Scan entfernt wurden, konnten die Objekte geclustert werden. Das Hauptaugenmerk der Arbeit lag in der Entwicklung eines aussagekräftigen und schnell zu berechnenden Merkmals.

Darüber hinaus wurde die Standardformulierung des Associative Markov Networks auch abgeändert, um weitere kontextuelle Informationen verwenden zu können.

- Bei [Munoz et al., 2008] wurde das Modell um Richtungsabhängigkeit erweitert. Die Kantenpotentiale wurden in der neuen Formulierung nach Richtung im Raum unterschieden und damit konnten lineare Objekte wie Stromleitungen, Baumstämme oder Pfähle besser erkannt werden. Ausserdem wurde das Lernen alternativ mit dem Subgradientenverfahren und die Inferenz mit GraphCut-Algorithmen durchgeführt.

- Weiter haben die Autoren in [Munoz et al., 2009b] die Formulierung auf Cliquen höherer Ordnung erweitert. Diese Cliquen wurden für geclusterte Punktmengen erstellt und erlaubten eine schnellere Inferenz gegenüber normalen paarweisen Netzen bei vergleichbarer Klassifikationsrate. In einer weiteren Arbeit [Munoz et al., 2009a] wurde dieser Ansatz mit einem weiteren Lernansatz getestet, bei welchem nicht die besten log-linearen Parameter ermittelt wurden, sondern ein Decision-Tree binärer Schwellwertfunktionen. Die Ergebnisse waren eine weitere Verbesserung der Klassifikationsgenauigkeit.

- Die Arbeit [Shapovalov et al., 2010] beschäftigt sich mit Associative Markov Networks, bei denen die Kantenpotentiale nicht assoziativ sind und damit auch das Lernen dieser Potentiale für unterschiedliche Labelpaare möglich ist. Dadurch ist es den Autoren gelungen, die Glättungsfehler des Netzwerks zu verringern und damit die Klassifikationsgenauigkeit ebenfalls zu verbessern.

Neben Verbesserungen an der Struktur und den Potentialfunktionen der Conditional Random Fields wurden auch Erweiterungen auf räumlich-temporale Datenströme diskutiert.

- So haben die Autoren von [Chen and Tang, 2007] solch ein Modell genutzt, um Zwischenbilder eines Films miteinander zu verknüpfen. Damit konnte ein mit Gaußschem Rauschen versetzter Film besser entrauscht werden als mit anderen modernen Verfahren.

- Eine weitere Anwendung findet sich in den Arbeiten [Douillard et al., 2007] und [Douillard et al., 2010]. Die Autoren verwenden ein solches bedingtes Modell, um auf zusammenhängenden Bild- und Laserdaten über die Zeit Objekte zu erkennen.

Durch die hinzugekommene temporale Glättung konnte die Klassifikation signifikant verbessert werden.

Zusammenfassend sind Conditional Random Fields das aktuelle Verfahren der Wahl zur robusten Klassifikation von 3D-Laserscandaten. Die Literatur hat dabei auch durchgehend gezeigt, dass eine intelligente Datenverwaltung und -reduktion sowie aussagekräftige Merkmale ebenfalls wichtig sind, um performant und auch robust zu klassifizieren. Weiter wurde gezeigt, dass das Rahmenwerk flexibel erweiterbar ist und dass nicht nur räumliche, sondern auch zeitliche Informationen gewinnbringend in die Klassifikation integriert werden können. Ausserdem erlauben diese Modelle auch alternative Verfahren zum Lernen und zur Inferenz.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die robuste und performante Klassifikation von zeitlich aufeinanderfolgenden 3D-Laserscans. Ausgehend vom heutigen Stand der Forschung sowie den sehr positiven Ergebnissen aus der Literatur werden für diese Aufgabe Associative Markov Networks [Taskar et al., 2004] gewählt und methodisch an das vorliegende Problem angepasst.

Dazu wird das probabilistische Rahmenwerk derart erweitert, dass sequentielle (d.h. zeitlich unmittelbar aufeinanderfolgende) Abtastungen die Klassifikation beeinflussen. Dabei werden zwei verschiedene Ansätze verfolgt und im Vergleich zur Standardformulierung auf die Fragestellung der Performanz und Genauigkeit der Klassifikation untersucht.

Ein weiterer Aspekt der Arbeit ist die Untersuchung der alternativen Varianten zum Lernen der Modellparameter sowie zur Inferenz in diesen Modellen. Dabei sind Fragestellungen wie Zeit- und Speicheraufwand als auch die Ergebnisse der Verfahren von großer Relevanz.

Um eine robuste Klassifikation zu ermöglichen, sollten zum einen aussagekräftige Merkmale für die Daten verwendet werden und zum anderen müssen die Daten auch so verwaltet werden, dass wichtige Informationen erhalten bleiben und die zeitlich-geometrische Einordnung der Scandaten funktionieren kann. Daher wird in dieser Arbeit eine angepasste Version der Spin-Images besprochen und eine Transformation der Scandaten in eine Voxelstrukur durchgeführt.

1.3 Aufbau der Arbeit

Diese Arbeit geht im Grundlagenkapitel zuerst auf das Problem der Klassifikation von Scandaten ein und erläutert die Motivation einer probabilistischen Modellierung. Ebenfalls wird kurz auf grundlegende Sachverhalte des maschinellen Lernens eingegangen und warum

Punkte in einem Scan nicht isoliert betrachtet klassifiziert werden sollten. Anschließend werden Markov-Netzwerke als ungerichtete graphische Modelle eingeführt und die spezielle Variante der Associative Markov Networks. Für diese gibt es Standardformulierungen sowohl für die Inferenz als auch für das Lernen der Modellparameter.

Im nächsten Abschnitt wird erklärt, wie mit den großen Datenmengen mittels Voxelgittern effizient umgegangen werden kann. Weiter wird gezeigt, wie mit der Voxelstruktur eine abgeänderte Variante der Spin-Images als Merkmalsdeskriptor performant berechnet wird. Ferner wird für das Lernen und die Inferenz jeweils eine weitere Methode vorgestellt.

Das vierte Kapitel beschäftigt sich dann mit der Formulierung sequentieller Associative Markov Networks. Unter verschiedenen Annahmen wird dann jeweils die probabilistische Modellierung verändert und die Auswirkung auf das Lernen und die Inferenz angesprochen.

Dem folgt die Evaluation der verschiedenen Lernverfahren sowie der Untersuchung der sequentiellen Modelle im Vergleich zur Standardformulierung. Abschließend wird im letzten Kapitel eine Zusammenfassung und Vorschläge für weitere Verbesserungen und folgende Arbeiten gegeben.

Kapitel 2

Grundlagen

Dieser Abschnitt erläutert das vorliegende Szenario der bewegten Trägerplattform mit montiertem Laserscanner und das damit einhergehende Klassifikationsproblem. Dieses Problem wird probabilistisch formuliert und eine mögliche Lösung dafür mittels überwachten Lernens präsentiert.

Aufgrund der Ergebnisse aus der Literatur wird zur Modellierung der Wahrscheinlichkeitsverteilung für die Klassifikation ein spezielles ungerichtetes graphisches Modell genutzt. Daher werden erst generell Markov-Netzwerke eingeführt. Davon ausgehend werden dann Conditional Random Fields besprochen und die spezielle Variante der Associative Markov Networks erklärt. Im Anschluss wird darauf eingegangen, wie in Associative Markov Networks Inferenz betrieben werden kann und wie in solchen Netzwerken überwachtes Lernen ermöglicht wird.

2.1 Problemstellung und Lösungsansatz

Häufig begegnet man in vielen Bereichen dem Problem, in Datensätzen Strukturen zu erkennen und diese adäquat identifizieren zu müssen. Dazu zählen beispielsweise die Erkennung von Personen oder Gesichtern in Bildern, d.h. die Klassifikation zusammenhängender Pixelregionen. Weiter die semantische Analyse von gesprochenen oder geschriebenen Texten, wobei Wortgruppen identifiziert und im Verbund klassifiziert werden, sodass ein System Sätze in ihrer Bedeutung erfassen kann. Ein weiteres Beispiel ist auch das Auffinden von Erbkrankheiten in Gensequenzen durch Klassifikation von Gengruppen. Im hier vorliegenden Szenario bewegt sich eine Trägerplattform durch ein vorgegebenes urbanes Areal und tastet simultan mit Hilfe des montierten 3D-Laserentfernungsmessers die Umgebung ab. Dieser Laserentfernungsmesser dreht sich dabei um seine eigene Achse, während er Laserstrahlen aussendet, die reflektierten Laserstrahlen detektiert und anhand der vergangenen Zeit zwischen Aussendung und Detektion die Länge der Strahlen und damit die Distanz zum abgetasteten Objekt ermittelt. Zudem wird auch die Stärke der Rückstrahlung gemessen, auch Remission genannt, welche sich je nach abgetastetem Objekt stark unterscheiden kann. Beispielsweise hat ein Fahrzeug mit schwarzer Lackierung einen

(a) (b)

Abbildung 2.1: (a) Velodyne LIDAR HDL-64E (Quelle: Produktbeschreibung des Herstellers), (b) das genutzte System auf dem Gelände der FKIE. Neben dem Velodyne HDL-64E Laserscanner wurde noch ein rotierender SICK-Laserscanner (links von den Notebooks) mitgeführt, dessen Daten jedoch nicht in dieser Arbeit verwendet wurden. Zur genaueren Positionsbestimmung wurde differentielles GPS verwendet. Die Antenne der Basisstation ist im Hintergrund zu sehen.

sehr geringen Remissionswert wohingegen Objekte mit hoher Reflektion (Schilder, Fahrradreflektoren) hohe Remissionwerte zurückliefern. Um die reflektierten Strahlen räumlich richtig einordnen zu können, verfügt die Trägerplattform außerdem meist über eine Hilfssensorik zur eigenen Positionsbestimmung. Das Endergebnis dieser Abtastung ist eine dreidimensionale Punktwolke wobei die Plattform sich in der Mitte dieser Punktwolke befindet. In der Ausarbeitung wird der Kürze halber 'Laserentfernungsmesser' synonym als 'Scanner' und die resultierende Punktwolke als 'Scan' bezeichnet.

Für diese Arbeit lagen Scandaten der FKIE [1] vor, die mittels des Laserscanners Velodyne LIDAR HDL-64E aufgenommen worden sind. Dieser Scanner liefert pro Sekunde mehr als 1,3 Millionen Abtastungen und damit eine sehr hoch aufgelöste Repräsentation der Umgebung. Die genutzte Trägerplattform war das Kettenfahrzeug 'Longcross' der Firma QinetiQ und die Hilfssensorik der Plattform war ein inertiales Navigationssystem (INS) der Firma Oxford Technical Solutions. Das INS verrechnet die Position (GPS), die Beschleunigung (Inertialsensoren) und die Rotation (Gyroskop) der Trägerplattform, und minimiert dabei die Fehler bei der eigenen Ortsbestimmung aufgrund der inhärenten Ungenauigkeiten der einzelnen Sensoren. Das Resultat dieser Verrechnung ist eine relativ präzise 6-dimensionale Position im Raum, bestehend aus den kartesischen Koordinaten (im WGS84-System) und

[1] Fraunhofer-Institut für Kommunikation, Informationsverarbeitung und Ergonomie

der Rotation der Trägerplattform in Nick-, Roll- und Gierwinkeln (engl. Roll-Pitch-Yaw). Da der Velodyne-Scanner mit bis zu 15 Hz um seine Achse rotieren kann, pro Abtastung 64 Laserstrahlen aussendet und damit der Begriff des Scans relativ ist, wird ein Scan $\mathcal{S} = \{p_1, ..., p_N\}$ hier definiert als die Menge der Punkte, die bei einer vollen Umdrehung des Scanners gewonnen wurde. Abbildung 2.1 zeigt den Scanner und das Gesamtsystem.

Das Ziel ist es nun, die gewonnene Menge der Scanpunkte mit semantischen Informationen zu versehen. In diesem Kontext heißt das, zu entscheiden, welche Objektklasse einem Punkt zugeordnet wird. Eine *Objektklasse* kann dabei abstrakt (bspw. 'Ebene', 'Kugel', 'Boden') oder aber recht spezifisch ('Baum', 'Hauswand', 'Fahrzeug') definiert sein. Daraus resultiert eine grobe Segmentierung der Umgebung in definierte Objektklassen und diese kann bspw. für weitere Probleme der Robotik wie 3D-Rekonstruktion oder SLAM hinzugezogen werden. Ein exemplarischer Scan mit manueller Segmentierung in mehrere Objektklassen ist in Abbildung 2.2 auf der nächsten Seite zu ersehen.

Die Menge der möglichen Objektklassen kann dabei beliebig groß sein. Allerdings steigt mit der Anzahl der möglichen Objektklassen auch die Komplexität der Zuordnung. Dies ist zum einen dann der Fall, wenn Klassen sich in ihrer Struktur ähneln und dadurch schwer zu unterscheiden sind. Im Sinne der geometrischen Struktur wäre das beispielsweise die nicht einfache Unterscheidung von 'Gehweg' und 'Fahrradweg'. Zum anderen steigt mit der Anzahl der Objektklassen aber auch meist der Berechnungsaufwand der genutzten Methoden die diese Zuordnung durchführen.

Aufgrund dieser Überlegungen wurde in der Arbeit der Kontext auf drei Objektklassen eingeschränkt (wobei die Ground Truth selbst mehr Klassen beinhaltet), nämlich 'Fahrzeug', 'Boden' als genereller Untergrund und 'Vegetation' als Sammelbegriff für Büsche und Sträucher. Die Annahme hierbei ist, dass drei Objektklassen ausreichend sein sollten, um den Klassifikator zu evaluieren und dass sich die drei Objektklassen in ihrer geometrischen Struktur ausreichend stark unterscheiden. Um der Nomenklatur der gängigen Literatur zu folgen, wird fortan Objektklasse synonym als *Label* bezeichnet und die Zuweisung eines Labels zu einem Punkt als 'Labeling' oder 'Klassifikation'.

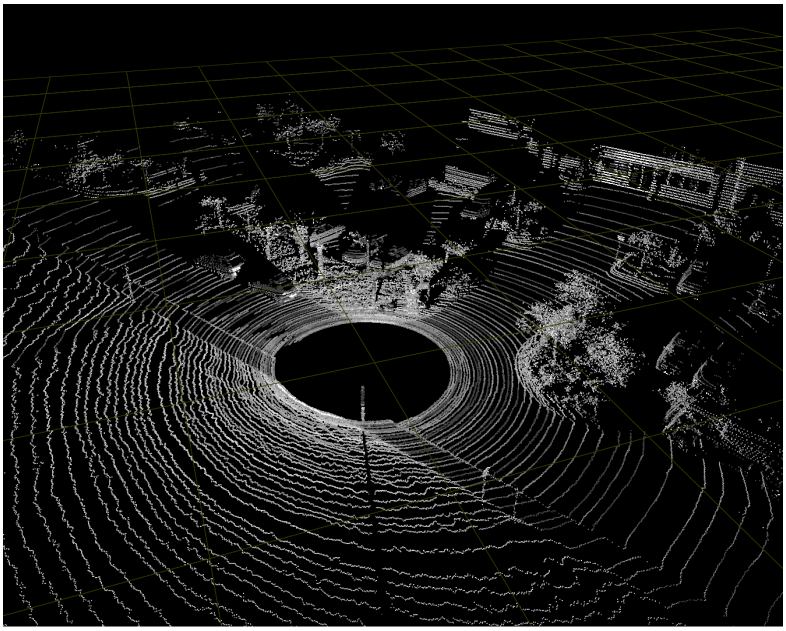
2.1.1 Klassifikation

Bevor über den Begriff der Klassifikation gesprochen werden kann, soll geklärt werden, wie ein Klassifikator generell definiert ist. Darauf aufbauend wird kurz diskutiert, warum Klassifikation selten eindeutig sein kann.

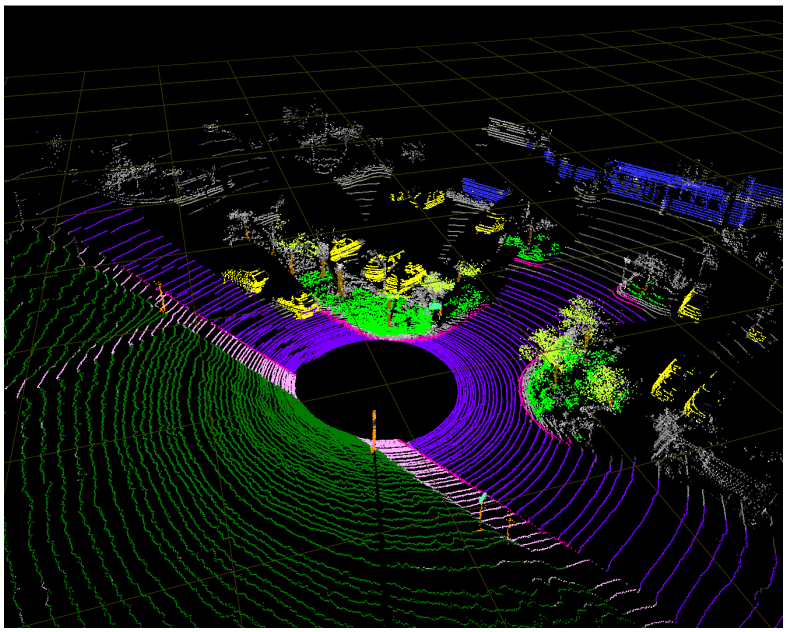
Definition 2.1.1 (Klassifikator)**.** *Sei $\mathcal{S} = \{p_1, \cdots, p_N\}$ eine Punktmenge mit N Punkten und $\mathcal{L} = \{k_1, \cdots, k_K\}$ die K-elementige Menge der möglichen Objektklassen. Ferner existiert für jeden Punkt p_i ein dazugehöriges Label $y_i \in \mathcal{L}$. Dann ist die Abbildung*

$$F : \mathcal{S} \to \mathcal{L},\ p_i \mapsto y_i \tag{2.1}$$

ein Klassifikator, welcher jedem Punkt p_i das Label $F(p_i) = y_i$ mit einem Wert aus $\mathcal{L}$ belegt.



(a)



(b)

Abbildung 2.2: Ein Scan ohne (a) und mit (b) manueller Segmentierung. Die unterschiedlichen Helligkeitsstufen bei (a) repräsentieren den Grad der Remission. Je stärker ein Objekt den Laserstrahl zurückstrahlt, umso heller wird der Abtastpunkt bei der Visualisierung. Bei (b) kann man durch die Segmentierung klar Objektklassen wie Autos (gelb), Straße (lila), Vegetation (grün) oder auch Gebäude (blau) erkennen. Die Plattform befindet sich dabei in der Mitte des Scans und bildet, bedingt durch den vertikalen Abtastwinkel des Scanners, um sich herum eine Lücke in der Punktwolke.

Solch eine Funktion, die eine Eingabe sofort auf genau ein Klassenlabel abbildet, wird auch *diskriminante Funktion* genannt [Bishop, 2006]. Wünschenswert wäre daher ein Klassifikator F, der diese Belegung immer korrekt ausführt. Das Problem ist aber, dass Klassifikatoren in den wenigsten Fällen eine eindeutige und korrekte Antwort zurückliefern können. Dies hat mehrere Ursachen, von denen hier die wichtigsten kurz aufgeführt werden.

Oft sind die zu klassifizierenden Daten in der Erhebung einem natürlichen Rauschen unterworfen. Bei den Scandaten tritt dies beispielsweise dadurch in Erscheinung, dass Velodyne für seinen Scanner eine Distanzgenauigkeit von kleiner als zwei Zentimeter und damit eine obere Schranke für die Präzision in der Abtastung angibt. Auch die Ungenauigkeiten der Hilfssensorik verstärken das Problem des natürlichen Rauschens.

Ein weiterer Grund ist der Informationsgehalt bzw. die Aussagekraft der Daten. Je nach Anwendungsdomäne erfordern diese eine Vorverarbeitung, in welcher jedem Datum im Datensatz ein beschreibendes, möglichst informatives Merkmal zugewiesen wird. Das Bestimmen angemessener Merkmale hat signifikanten Einfluss auf das Ergebnis der Klassifikation und wird später noch einmal aufgegriffen.

Ein weiterer wichtiger Aspekt betrifft den Klassifikator selbst. In der Literatur finden sich unzählige Ansätze und Diskussionen mit verschiedensten Klassifikatormodellen und Parametrisierungen, wobei die Conditional Random Fields für probabilistische Klassifikation von 3D-Laserscans (siehe nächster Abschnitt) immer sehr gute (und meist die komparativ besten) Ergebnisse liefern. Es wäre unmöglich, in dieser Arbeit auch nur auf einen Bruchteil der möglichen Modelle einzugehen. Es sei an dieser Stelle aber angemerkt, dass all diese Klassifikatoren auch Grenzen haben, in welchen sie zuverlässige Aussagen liefern können. Versucht man daher Daten mit ungeeigneten Klassifikatoren zu bestimmen, versagen diese Modelle oft und zeigen dem Anwender die Grenzen des jeweiligen Verfahrens auf. Ein Beispiel wäre hier die Separation komplexer, nicht-linear zusammenhängender Daten mit einem linearen Klassifikator, welcher mit Hyperebenen keine gute Trennung erreichen würde. Der in dieser Arbeit genutzte Ansatz der Conditional Random Fields bzw. Associative Markov Networks hat sich in der Domäne des 3D-Scanlabelings jedoch im Vergleich zu anderen etablierten Verfahren als besonders geeignet erwiesen und findet daher hier Anwendung. Außerdem wird die Nutzung der Conditional Random Fields für die Klassifikation in den nächsten Abschnitten motiviert.

Zusammengefasst ist also Klassifikation fast immer mit Unsicherheit verbunden, und daher wird im Folgenden das Klassifikationsproblem in eine probabilistische Formulierung überführt.

2.1.2 Probabilistische Modellierung

Die grundlegende Idee bei der probabilistischen Modellierung ist es, die Punkte bzw. deren Merkmale und ihre Labels als Zufallsvariablen zu definieren und sie in ein probabilistisches Rahmenwerk zu setzen. Seien also ein Wahrscheinlichkeitsmaß P und Zufallsvaria-

blen $p_1, ..., p_N, y_1, ..., y_N$ gegeben, wobei $y_1, ..., y_N$ als diskrete Zufallsvariablen über dem Raum der Labels $\mathcal{L}$ definiert sind.

Eine Möglichkeit wäre es, eine Verbundwahrscheinlichkeit $P(p_1, ..., p_N, y_1, ..., y_N)$ zwischen diesen Variablen zu modellieren. Solch ein Modell würde also auch wissen, wie wahrscheinlich eine Belegung für die Variable p_i ist, wenn es das Label bzw. die Variable y_i als Evidenz erhält, also $P(p_i|y_i)$. Da es mit solch einem Modell möglich wäre, Anfragen in beide Richtungen zu stellen und damit synthetisch auch neue Abtastungen p_j zu generieren, werden solche Modelle auch *generative probabilistische Modelle* genannt.

Anfragen heißt hier, dass man Evidenzen über eine Teilmenge der Variablen besitzt und mit deren Hilfen die Belegung der unbekannten Variablen ermittelt. Anfragen in beide Richtungen bedeutet dabei, dass eine Teilmenge der Punkte aber auch eine Teilmenge der Labelings als Evidenz gegeben sein kann. Das Problem mit solchen Modellen ist jedoch, dass der Aufwand zum Finden einer Verteilung über die Punkte $p_1, ..., p_N$ aus frequentistischer Sicht extrem hoch sein kann und eine sehr große Menge an Lerndaten erfordern kann, um die Verteilung akkurat zu erfassen. Darüber hinaus kann auch der Fall eintreten, dass über die Verteilung der p_i gar keine Aussage getroffen werden kann, weil diese keine greifbare Struktur besitzt bzw. mathematisch nicht klar zu beschreiben ist [Bishop, 2006]. Ein Beispiel wäre im vorliegenden Szenario das Lernen einer Verteilung von Autopunkten und die Wahrscheinlichkeit des Auftretens dieser in bestimmten Konstellationen. Würde das System eine stark und dicht befahrene Hauptstraße entlang fahren und eine Verteilung der Autopunkte lernen, dann wäre diese gelernte Verteilung für eine ruhige Nebenstraße sofort ungültig. Im Grunde verstärkt jeder mögliche, nicht vorhersehbare Einfluss die Problematik eines solchen Schätzers.

Die andere probabilistische Variante ist die Modellierung einer bedingten Wahrscheinlichkeit $P(y_1, ..., y_N|p_1, ..., p_N)$ mit den Punkten als Evidenz. Dieses Modell beschreibt die Wahrscheinlichkeit eines bestimmten Labelings, gegeben den Punkten und erlaubt so einseitige Anfragen. Solche Modelle werden *diskriminative probabilistische Modelle* genannt und diese Arbeit wird sich eines solchen Modells bedienen. Der Grund ist, dass solche Modelle nicht so komplex wie ihre generativen Gegenstücke sind, eine solche bedingte Verteilung häufig leichter zu modellieren ist und solche Modelle die für die Klassifikationsaufgabe benötigten Anfrage- bzw. Inferenzmechanismen ebenfalls erlauben [Bishop, 2006].

Die diskriminative probabilistische Klassifikationsaufgabe ist es nun, bei gegebenen Punkten $p_1, ..., p_N$ *möglichst optimal* die dazugehörigen Labels zu bestimmen, d.h. eine Belegung für $y_1, ..., y_N$ zu finden, sodass die bedingte Wahrscheinlichkeit $P(y_1, ..., y_N|p_1, ..., p_N)$ maximiert wird. Das Finden des besten Labelings ist jetzt also eine Bestimmung der *Maximum-A-Posteriori*-Belegung (MAP)!

Wie eben erwähnt, bietet ein Punkt für sich allein jedoch außer seiner Position (x, y, z) im Raum, seiner Entfernung zum Laserscanner und seiner Remission keine weitere Information, die Aufschluss über seine Objektklassenzugehörigkeit geben könnte. Es ist daher

notwendig, Möglichkeiten zu finden, dreidimensionale Punkte und ihre Umgebung strukturiert beschreiben zu können.

Ein sogenannter *Merkmalsdeskriptor* D berechnet für einen Punkt p_i einen dazugehörigen *Merkmalsvektor* $D(p_i) = \mathbf{x}_i$. Diese Deskriptoren können sich stark in ihrer Methodik und im Informationsgehalt der erzeugten Merkmale unterscheiden. Die Merkmale zweier Punkte sollten sich dabei optimalerweise genau dann ähneln, wenn die Punkte zur selben Objektklasse gehören. Diese Thematik wird im dritten Kapitel näher behandelt. An dieser Stelle sei jedoch schon zum Verständnis erwähnt, dass jedem Punkt im Scan ein solches Merkmal zugewiesen wird, damit dieser Punkt möglichst eindeutig in Bezug zu seiner Objektklasse beschrieben wird.

Die Menge aller Labels $\{y_1, ..., y_N\}$ und die Menge aller Merkmalsvektoren $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ werden zu zwei Vektoren $\mathbf{y}$ und $\mathbf{x}$ zusammengefasst. Durch Ersetzung der Punkte durch ihre Merkmale lässt sich das Problem nun schreiben als $P(\mathbf{y}|\mathbf{x})$ und mit der probabilistischen Modellierung ergibt sich für den Klassifikator daher:

$$F(\mathcal{S}) := \underset{\mathbf{y}}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{x}) \tag{2.2}$$

Der Klassifikator soll also bei einem Scan dasjenige MAP-Labeling wählen, dass die bedingte Wahrscheinlichkeit bei gegebenen Punktmerkmalen maximiert.

Es wurde an dieser Stelle noch keinerlei Aussage über die Modellierung der Wahrscheinlichkeitsverteilung getroffen. Bevor nämlich auf diese Modellierung eingegangen wird, soll zuerst der Ansatz des überwachten Lernens eingeführt und erläutert werden.

2.1.3 Überwachtes Lernen, Modellselektion und Inferenz

Ein oft genutzter Ansatz im Bereich des maschinellen Lernens ist das überwachte Lernen [Bishop, 2006]. Die Grundidee ist, ein gegebenes Modell auf repräsentativen Daten (der sogenannten Ground Truth) lernen zu lassen um durch die Konditionierung anschließend auf unbekannten Daten Klassifikationshypothesen erstellen zu können. In der Regel haben diese Modelle dabei eine Gewichtung bzw. Parametrisierung, die sich im Laufe der Lernphase an die Struktur des Problems anpasst. Da es aber eine große Menge an Klassifikatoren und ein Kontinuum an Parametrisierungen gibt, muss entschieden werden, welches Modell schlussendlich für die Klassifikation gewählt wird. Dieses Problem, auch Modellselektion genannt, ist im Bereich des maschinellen Lernens von großer Bedeutung und wird hier auch kurz erläutert. Ist das der Situation bezüglich beste Modell gewählt, wird dieses auf unbekannten Daten angewendet um eine (möglichst korrekte) Klassifikation zu erreichen.

Die Problematik wird also in eine *Lern-* und eine *Inferenzphase* unterteilt. Beide Phasen besitzen dabei voneinander getrennte Datenmengen, auf denen das Modell agiert, um sicherzustellen, dass das trainierte Modell später auch auf unbekannten Daten robust arbeitet. Dies wird später bei der Evaluation des Modells im fünften Kapitel näher besprochen.

Die Mengen werden dabei sinngemäß je nach Phase *Lern- bzw. Trainingsmenge* und *Test- bzw. Evaluationsmenge* genannt.

Lernen

Im Kontext der Klassifikation von 3D-Laserscans bedeutet das, dass eine Trainingsmenge aus Punkten besteht, deren Labeling bekannt ist. Sei also der Vektor $\hat{\mathbf{y}}$ das korrekte Labeling zu einer gegebenen Punktmenge $\mathcal{S}$, respektive dem gegebenen Vektor $\mathbf{x}$ der Merkmalsvektoren. Ausserdem sei P_θ ein Wahrscheinlichkeitsmaß und θ die Parametrisierung der zugrunde liegenden Verteilung. Dann lässt sich das Lernproblem schreiben als:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, P_\theta(\hat{\mathbf{y}}|\mathbf{x}) \tag{2.3}$$

Das heißt, das Lernen ist das Finden der optimalen Parameter θ^*, welches die bedingte Wahrscheinlichkeit maximiert. Solch eine Bestimmung ohne vorherige Informationen über die Verteilung wird auch *Maximum-Likelihood-Estimation* (MLE) genannt. Da zu jedem Zeitpunkt das korrekte Labeling bekannt ist und damit das Lernverfahren in eine konkrete Richtung gesteuert wird, wird dieser Vorgang überwachtes Lernen genannt.

Das Problem, dass diesem Prinzip inhärent gegeben ist, ist die Spezialisierung des Modells auf die Trainingsmenge, auch *Überanpassung* oder *over-fitting* genannt. Im Laufe des Lernverfahrens wird das Modell nämlich darauf konditioniert, die gegebene Trainingsmenge im Rahmen der eigenen Möglichkeiten so exakt wie möglich zu prädizieren. Sollte aber der Fall eintreten, dass die Trainingsmenge die gesamte Datenmenge nicht vollständig repräsentiert, wird der Klassifikator bei anders strukturierten Daten versagen. Ein Beispiel im Kontext der 3D-Klassifikation wäre das Lernen von Fahrzeugen, welche sich alle geometrisch ähneln und die Form eines PKWs hätten. Würde dem Klassifikator nun eine Punktwolke in Form eines Busses gegeben, wäre eine korrekte Klassifikation als Fahrzeug kaum möglich weil der Klassifikator solch ein Fahrzeug nie gelernt hat.

Die gewünschte Eigenschaft lautet hier also *Generalisierung* und ist eine Forderung der Modellselektion. Das gelernte Modell soll auf unterschiedlich strukturierten Szenen bzw. Punktwolken robust klassifizieren. Erreicht werden kann dies dadurch, dass die Trainingsmenge jede mögliche Strukturierung der Daten enthält, die später auch in der Klassifikation auftauchen könnte. Sicherstellen könnte man dies, indem man die Trainingsmenge sehr groß wählt. Der Nachteil ist der Aufwand, der dann mit dem Lernen auf sehr großen Mengen einhergeht. Und selbst dann ist keine Gewissheit darüber gegeben, ob nicht doch statistisch signifikante Informationen in der Trainingsmenge fehlen. Eine weitere Möglichkeit ist die *Regularisierung* der Parameter des Modells. Dabei achtet man während des Lernvorgangs darauf, dass die Parameter sich innerhalb gewisser Grenzen bewegen. Im Evaluationsteil wird auf beide dieser Gedankengänge näher eingegangen. Wichtig ist die Bemerkung, dass das Problem des over-fitting immer präsent ist und die Spezialisierung systembedingt auch nie vollständig erlischt.

Inferenz

Die zweite Phase ist die Inferenz. In dieser Phase besitzt man ein vorher konditioniertes Modell und möchte nun auf der Testmenge überprüfen, wie gut sich das Modell auf unbekannten Daten verhält. Durch die vorher durchlaufene Lernphase besitzt man also eine Parametrisierung θ^*, aber dafür einen neuen Vektor $\mathbf{x}$ von Merkmalsvektoren zusammen mit einem Labelvektor $\mathbf{y}$ unbekannter Belegung. Das Ziel in der Inferenzphase lässt sich mathematisch schreiben als:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} P_{\theta^*}(\mathbf{y}|\mathbf{x}) \tag{2.4}$$

und damit als das Ermitteln der optimalen MAP-Belegung $\mathbf{y}^*$, gegeben einer Modellparametrisierung und den Punktmerkmalen.

Bei dem Vergleich des Inferenzausdrucks mit dem des probabilistischen Klassifikators ist sofort erkennbar, dass diese sich bis auf das θ^* gleichen. Nun lässt sich aber der Klassifikator mit einer zusätzlichen Parametrisierung θ^* als genau diese Inferenz beschreiben und erlaubt die folgende Formulierung:

$$F_{\theta^*}(\mathcal{S}) := \underset{\mathbf{y}}{\operatorname{argmax}} P_{\theta^*}(\mathbf{y}|\mathbf{x}) \tag{2.5}$$

Damit wurde der Klassifikationsbegriff hinreichend definiert und die Frage nach einer guten, dem Maß P_{θ^*} zugrundeliegenden, parametrisierten Verteilungsmodellierung eröffnet.

2.1.4 Kollektive Klassifikation

Die Verteilung für P_θ kann beliebig gewählt werden. Eine Möglichkeit wäre es, eine Normalverteilung anzunehmen, wobei θ aus einem Mittelwertsvektor und einer Kovarianzmatrix für jedes Label bestehen würde. Dadurch würde man eine multimodale Normalverteilung erhalten, wobei ein Modus dann genau einem Label entsprechen würde. Solch eine Verteilung macht sehr starke Annahmen in Bezug auf die Unabhängigkeit der Daten untereinander. Dabei ist die hier erwähnte Normalverteilung nur eine mögliche Variante der Modellierung. Dieser Ansatz der Unabhängigkeit der Merkmale, gegeben der Klassen, wird bei der (naiven) *Bayes-Klassifikation* ausgenutzt. Ein solcher Klassifikator hat sich in der Literatur bei verschiedensten Problemen ausgezeichnet, gerade wenn diese Unabhängigkeitsannahmen in einem bestimmten Maße auf die Daten der Domäne zutreffen [Bishop, 2006].

Solche starken Annahmen sind für das gewählte Szenario aber nicht haltbar. Wenn ein Scanpunkt zu einer bestimmten Klasse gehört, dann steigt intuitiv auch die Wahrscheinlichkeit, dass benachbarte Punkte ebenfalls dieses Label besitzen. Geometrisch gesehen liegen diese abgetasteten Punkte unter Umständen auf einem gemeinsamen Objekt, bspw. einem Auto, und damit ist das hervorgebrachte Argument der Unabhängigkeit der Punkte untereinander nicht mehr gegeben. Diese Art der Klassifikation mit Nachbarschaftskontext wird auch *kollektive Klassifikation* genannt.

Der Vorteil einer solchen Modellierung ist der Informationsgewinn durch die Nachbarschaft, der bei der Entscheidung über das Labeling eines Punktes mit einfließt. Der Klassifikator hat während seiner Entscheidungsfindung für jeden Punkt eine Vermutung. Diese Vermutung, auch *Belief* genannt, drückt aus, wie sicher der Klassifikator sich für das ein oder andere Labeling eines Punktes ist. Beispielsweise kann ein starker, aber falscher Belief eines Punktes für ein bestimmtes Label durch seine Nachbarn überstimmt werden und damit die Klassifikation in die richtige Richtung lenken. Damit wird im übertragenen Sinne eine geometrische Glättung im Labeling erreicht und Ausreißer werden (zumindest in Grenzen) mit der richtigen Objektklasse versehen. Andererseits kann eine falsch gelabelte Nachbarschaft den Belief eines Punktes auch ungünstig verändern. Wie stark die Nachbarschaft Einfluss ausüben sollte, sollte dabei als ein zusätzlicher Parameter gegeben sein und im Lernvorgang bestimmt werden.

Gesucht ist also ein Modell, dass eine Wahrscheinlichkeitsverteilung repräsentiert und dabei gleichzeitig die Möglichkeit bietet, solche Abhängigkeiten in das Modell zu integrieren. Probabilistische graphische Modelle wie beispielsweise Markov-Netzwerke und ihre Varianten bieten für solche Probleme ein nützliches Rahmenwerk. Im Folgenden werden aber graphische Modelle zuerst allgemein beschrieben, bevor im Anschluss auf Markov-Netzwerke fokussiert wird.

2.2 Probabilistische graphische Modelle

Probabilistische graphische Modelle sind Werkzeuge zur Gestaltung bzw. Repräsentation von Verbundwahrscheinlichkeiten und bedingter Unabhängigkeiten als Graphstrukturen. Dabei bilden die Knoten die Zufallsvariablen (ZVs) dieser Verteilungen und Kanten zwischen den Knoten modellieren Abhängigkeiten der ZVs untereinander. Daher werden in diesem Kontext Zufallsvariablen und Knoten synonym für den selben Begriff verwendet. In der Literatur werden dabei grob zwei Arten von Netzwerken (oder Netzen) unterschieden: gerichtete Netzwerke (sog. *Bayes-Netze*) und ungerichtete Netzwerke (sog. *Markov-Netze*). Jeweils ein Exemplar ist in Abbildung 2.3 zu ersehen. Wichtig ist hierbei die Beobachtung, dass die beiden Netzwerke in der Abbildung die selbe Verbundverteilung, jedoch unterschiedliche Unabhängigkeitsannahmen kodieren.

Bayes-Netze modellieren einseitige Einflüsse von Zufallsvariablen und suggerieren dies auch durch die gerichteten Kanten. Die Berechnung der Wahrscheinlichkeit im abgebildeten Bayes-Netz für $P(B,C) = \sum_A P(A,B,C) = \sum_A P(B|A) \cdot P(C|A) \cdot P(A)$ verläuft über die bedingte Wahrscheinlichkeit des Elternknotens A mit Hilfe von sog. Conditional Probability Tables (CPTs). Diese geben die Wahrscheinlichkeit einer Belegung einer ZV in Abhängigkeit von der Belegung ihrer Eltern an. In Markov-Netzen verläuft diese Wahrscheinlichkeitsberechnung anders, da dort verbundene Knoten wechselwirken. Dadurch ergibt sich auch der Schluss, dass beide Netze bestimmte Verteilungen modellieren können, welche von dem jeweils anderen Netz nicht konstruierbar sind. Bayes-Netze dienen hier

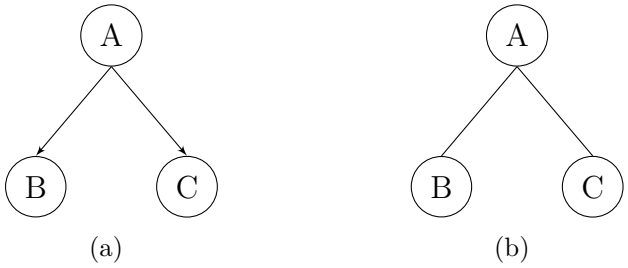


Abbildung 2.3: Graphische Repräsentation einer Wahrscheinlichkeitsverteilung dreier Zufallsvariablen A,B,C als (a) (gerichtetes) Bayes- und (b) als (ungerichtetes) Markov-Netzwerk. Zu beachten ist, dass die beiden vorliegenden Graphen die selbe Verbundverteilung $P(A, B, C)$, nicht aber die selben Unabhängigkeitsannahmen modellieren.

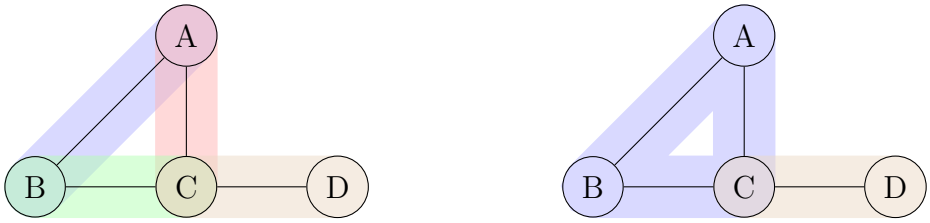
nur dem Vergleich und werden nicht weiter betrachtet. Für eine ausführliche Einführung zu Bayes-Netzen sei auf [Koller and Friedman, 2009] verwiesen.

2.2.1 Markov-Netzwerk

Formal ist ein *Markov-Netzwerk* (auch Markov-Netz, *Markov Random Field*, MRF) ein ungerichteter Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ mit $\mathcal{V}$ als Menge der Zufallsvariablen und $\mathcal{E} = \{(i, j)|i < j, v_i, v_j \in \mathcal{V}\}$ als Menge der ungerichteten Kanten im Graphen. Die Idee einer solchen Modellierung geht auf Ising [Ising, 1924] zurück und wurde erstmals von ihm für Spin-Vorhersagen in der statistischen Physik eingeführt. Zusätzlich besitzt ein MRF eine Parametrisierung θ über einer Menge von Funktionen, den Faktoren θ_c, welche auf Cliquen $c \in Cliques(\mathcal{G})$ definiert sind.

Eine Clique c ist dabei eine Menge von Zufallsvariablen $V_c \subseteq \mathcal{V}$, in welcher jede Variable mit jeder anderen in der Menge durch Kanten verbunden ist. Dabei ist die Größe einer Clique nicht vorgegeben, d.h. es muss nicht zwangsläufig immer die größtmögliche Clique gebildet werden. Der Faktor θ_c weist dann jeder möglichen Belegungskonstellation der Variablen in der Clique c ein nicht-negatives, reellwertiges Potential zu. Solch ein Faktor existiert dann genau für jede einzelne Clique im Graphen. Der Sachverhalt ist in Abbildung 2.4 beispielhaft zu ersehen.

Die Potentiale repräsentieren im Netz dabei keine Wahrscheinlichkeiten im eigentlichen Sinne (so wie es die CPTs bei den Bayes-Netzen tun), sondern sagen aus, wie kompatibel bzw. affin bestimmte Belegungen der Zufallsvariablen innerhalb dieser Clique sind. Dadurch ist eine Modellierung von symmetrischer Interaktion der Variablen untereinander möglich, also ein direkter gegenseitiger Einfluss auf die Belegung unmittelbarer Nachbarn. Hierbei modellieren Cliquen folgerichtig auch Abhängigkeiten, d.h. zwei voneinander unabhängige Variablen liegen nicht in einer gemeinsamen Clique und besitzen damit auch keine gemeinsamen Potentiale.



(a) (b)

Abbildung 2.4: Das gleiche Markov-Netzwerk mit (a) vier paarweisen Cliquen {(A,B),(A,C),(B,C),(C,D)} bzw. (b) zwei maximalen Cliquen {(A,B,C),(C,D)}. Dies impliziert, dass der Graph aus (a) über genau vier Faktoren verfügt wohingegen das Netz aus (b) mit zwei Faktoren parametrisiert ist.

Im Kontext der 3D-Scanklassifikation sind hier Punkte bzw. ihre Merkmale und die dazugehörigen Labels Zufallsvariablen, wobei die Merkmale kontinuierliche Variablen sein können und die Labels diskrete Variablen sind. Bei der Konstruktion des Graphen werden die Merkmalsvariablen mit den dazugehörigen Labelvariablen verbunden. Damit wird der Zusammenhang zwischen Label und Merkmal modelliert. Weiter werden die Labelvariablen geometrischer nächster Nachbarn in gemeinsame Cliquen gesetzt um den Nachbarschaftskontext einzugliedern. Damit besitzt man Merkmalscliquen und zusätzlich Nachbarschaftscliquen mit den jeweils dazugehörigen Potentialen.

Diese lokale Parametrisierung ermöglicht einen sehr hohen Freiheitsgrad, erschwert aber dadurch auch gleichzeitig die Wahl der richtigen Potentiale für eine Clique, da es keinen unmittelbaren und intuitiven Bezug zur Gesamtverteilung der Wahrscheinlichkeiten im Graphen gibt. Ausserdem erzwingt die beliebig reellwertige Parametrisierung auch eine nicht-triviale Normalisierung, die in nur in kleinen Netzen rechnerisch zumutbar ist.

Sei V eine Belegung aller Knoten bzw. Zufallsvariablen im MRF und V_c die Teilmenge der Variablen, die zur Clique c gehören. Dann ist die Wahrscheinlichkeit für V definiert als:

$$P(V) = \frac{1}{Z} \prod_{c \in Cliques(\mathcal{G})} \theta_c(V_c) \tag{2.6}$$

Die Gesamtwahrscheinlichkeit ergibt sich also als Produkt der einzelnen lokalen Potentiale im Graphen mit anschließender Normalisierung $\frac{1}{Z}$. Je kompatibler also die Gesamtbelegung der Variablen zueinander ist, desto höher ist das Produkt und damit auch die Wahrscheinlichkeit. Z wird auch *Partitionsfunktion* genannt und dient, wie eingangs erwähnt, der Überführung des Gesamtpotentials im Graphen in eine Wahrscheinlichkeitsangabe. Die Komplexität von Z steigt dabei (im worst-case) exponentiell in der Menge der Knoten und bildet damit die hauptsächliche Hürde für die Berechnungen, da über alle möglichen Belegungen evaluiert werden muss:

$$Z = \sum_{V} \prod_{c \in Cliques(\mathcal{G})} \theta_c(V_c) \tag{2.7}$$

Anzumerken ist, dass die Summe über alle Belegungen gebildet wird, da in dieser Domäne mit diskreten Variablen gearbeitet wird. Damit ist die Summation über K^N Zustände nötig, wobei N die Anzahl der Variablen ist und K die Anzahl der möglichen Belegungen.

Einschränkungen und Eigenschaften

Eingangs wurde erwähnt, dass Markov-Netze nicht jede beliebige Verteilung modellieren können. Im Folgenden wird also daher eingeschränkt, wann eine Verteilung mit einem MRF überhaupt repräsentierbar ist.

Definition 2.2.1 (Gibbs-Verteilung)**.** *Sei eine Verteilung P_θ mit Variablen $X = \{x_1, ..., x_n\}$ gegeben. Ferner ist P_θ mit einer Menge an Faktoren $\theta = \{\theta_1(D_1), ..., \theta_q(D_q)\}$, $D_i \subseteq X$ parametrisiert, sodass gilt:*

$$P_\theta(x_1, ..., x_n) = \frac{\prod_{i=1}^{q} \theta_i(D_i)}{\sum_{x_1,...,x_n} \prod_{i=1}^{q} \theta_i(D_i)} \tag{2.8}$$

Dann wird die Verteilung P_θ Gibbs-Verteilung genannt.

Diese Gleichung ähnelt aus gutem Grund der Wahrscheinlichkeitsberechnung in einem MRF, denn es gilt folgender Satz [Koller and Friedman, 2009]:

Satz 2.2.1. *Eine Gibbs-Verteilung lässt sich mit einem Markov-Netzwerk faktorisieren, genau dann, wenn die einzelnen Teilmengen D_i im Markov-Netz vollkommene Subgraphen bzw. Cliquen bilden.*

Findet sich also solch eine Darstellung, lässt sich ein Markov-Netz für P_θ konstruieren.

Weitere, sehr wichtige Eigenschaften in einem MRF sind die sogenannten *bedingten Unabhängigkeiten* (auch *Markov-Eigenschaften*). Dabei baut eine Eigenschaft auf der jeweils vorherigen auf. Bevor diese drei Definition erläutert werden, wird der benötigte Begriff der Markov-Hülle kurz eingeführt.

Definition 2.2.2 (Markov-Hülle)**.** *Sei $v_i \in \mathcal{V}$ ein Knoten eines Markov-Netzwerkes. Dann ist die Menge der Nachbarn $ne(i) := \{v_j \in \mathcal{V} | (i,j) \in \mathcal{E}\}$, wobei o.B.d.A. $i < j$, die sogenannte Markov-Hülle (engl. Markov-Blanket).*

Nun folgt die schwächste Definition der Markov-Eigenschaften.

Definition 2.2.3 (Paarweise Markov-Eigenschaft)**.** *Seien $v_i, v_j \in \mathcal{V}$, o.B.d.A $i < j$. Dann gilt: $v_i \perp v_j | \mathcal{V}_{\setminus\{v_i,v_j\}} \Leftrightarrow (i,j) \notin \mathcal{E}$.*

Zwei Variablen sind also unabhängig, gegeben aller anderen Variablen des Netzwerkes, genau dann, wenn sie keine gemeinsame Kante besitzen. Diese Definition lässt sich erweitern auf die Nachbarschaft eines Knotens.

Definition 2.2.4 (Lokale Markov-Eigenschaft)**.** *Sei $v_i \in \mathcal{V}$ ein Knoten mit gegebener Nachbarschaft $ne(i)$. Dann gilt: $v_i \perp \mathcal{V}_{\setminus\{\{v_i\}\cup ne(i)\}} | ne(i)$.*

Eine Variable ist unabhängig vom gesamten restlichen Netz, gegeben ihrer Nachbarschaft (bzw. Markov-Hülle). Die letzte Eigenschaft verallgemeinert die Aussage auf Subgraphen.

Definition 2.2.5 (Globale Markov-Eigenschaft)**.** *Seien $V_i, V_j, V_k \subseteq \mathcal{V}$ Teilmengen von Knoten. Sei ferner V_k eine separierende Teilmenge, d.h. alle Pfade von V_i nach V_j durch V_k laufen. Dann gilt: $V_i \perp V_j | V_k$.*

Man sagt auch, die Menge V_k *blockt* die Informationen bzw. den Einfluss. Ein Graph der diese Eigenschaften erfüllt, wird auch *Independency-Map* (I-Map) genannt, weil er genau die Unabhängigkeiten der Variablen untereinander modelliert.

Die letzte offene Frage ist, wann eine Wahrscheinlichkeitsverteilung in einem MRF mit diesen Eigenschaften konstruiert werden kann. Das sog. *Hammersley–Clifford-Theorem* [Hammersley and Clifford, 1971] besagt, dass ein MRF mit einer zugrundeliegenden Gibbs-Verteilung genau dann die Markov-Eigenschaften erfüllt, wenn die Verteilung ein positives Wahrscheinlichkeitsmaß besitzt. Schränkt man also in obiger Definition eines Markov-Netzes die θ_i so ein, dass sie immer positiv sind, gilt: $\forall i : \theta_i > 0 \Rightarrow P_\theta > 0$ und somit die Markov-Eigenschaften für diese Verteilung.

2.2.2 Conditional Random Field

Eine besondere Gruppe der MRFs sind die bedingten Markov-Netze (auch *Conditional Random Field*, CRF [Lafferty et al., 2001]). Bei diesem Netz wird die Menge der Variablen $\mathcal{V} = X \cup Y$ in zwei Teilmengen unterteilt, wobei die Menge X gegeben bzw. observiert ist und die restliche Menge Y die zu ermittelnden Variablen, auch *Zielvariablen* genannt, darstellt. Damit lässt sich, wie bei Abschnitt 2.1.2 erwünscht, eine bedingte Wahrscheinlichkeitsverteilung $P(Y|X)$ modellieren.

Die Parametrisierung verläuft analog bis auf eine bestimmte Einschränkung der Faktoren. Es gilt: $\forall i : \theta_i(V_i), V_i \not\subseteq X$. Das heißt, Faktoren dürfen nicht ausschließlich auf observierten Variablen definiert sein und damit auch keine Clique im Graphen nur aus observierten Variablen bestehen. Die Schlussfolgerung wäre sonst, dass man ein probabilistisches Modell über X legt, was weder gewollt noch (in den meisten Domänen) möglich ist. Damit sind die Faktoren in einem CRF entweder ausschließlich auf Y oder auf Y **und** X definiert (siehe Abbildung 2.5 zur Veranschaulichung). Die Vermeidung dieser Modellierung ist die Hauptstärke der CRFs, denn diese erlaubt die Verwendung von Informationen, deren Verteilung unbekannt ist. Ausserdem können (und werden in dieser Arbeit) diese Variablen auch kontinuierlich sein und damit Teil einer Verteilung, welche unter Umständen keine einfache parametrische Form besitzt. In Bezug auf das vorliegende Szenario sind die Merkmale also die gegebenen Variablen und die Labels die zu ermittelnden Variablen.

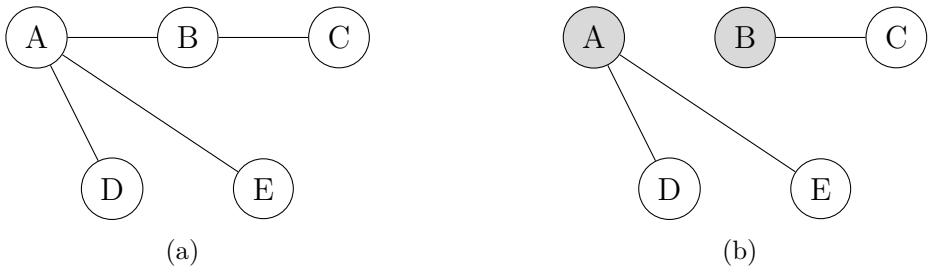


Abbildung 2.5: Visualisierung eines paarweisen CRF. Der Graph in (a) besteht aus mehreren paarweisen Cliquen. Der Graph in (b) hat nun Knoten A und B als Evidenz ($A, B \in X$) gegeben und damit besitzen beide Knoten aufgrund der Annahme keine gemeinsame paarweise Clique mehr.

Durch den veränderten Sachverhalt ändert sich auch die Wahrscheinlichkeitsberechnung, bezogen auf Gleichung (2.6), von $P(V)$ zu $P(Y|X)$ und durch Anwendung der Produkt- und Summenregel ergibt sich:

$$P(Y|X) = \frac{P(Y,X)}{P(X)} = \frac{P(Y,X)}{\sum_Y P(Y,X)} = \frac{1}{Z(X)} \prod_{c \in Cliques(\mathcal{G})} \phi_c(X_c, Y_c) \tag{2.9}$$

wobei X_c, Y_c die Teilmengen der observierten Variablen bzw. Zielvariablen der Clique $c = X_c \cup Y_c$ sind. Die Partitionsfunktion $Z(X)$ ist definiert als:

$$Z(X) = \sum_Y \prod_{c \in Cliques(\mathcal{G})} \phi_c(X_c, Y_c) \tag{2.10}$$

Zu beachten ist hier die veränderte Partitionsfunktion $Z(X)$. Durch die bedingte Modellierung ändert sich mit jeder Änderung einer observierten Variable $x \in X$ der Wert von Z und muss daher dann jedesmal erneut evaluiert werden. Das heißt, die Partitionsfunktion ist nun eine Funktion in X.

2.2.3 Associative Markov Network

Eine spezielle Variante der Conditional Random Fields sind die *Associative Markov Networks* (AMN) [Taskar et al., 2004]. Die Eigenschaften werden im Folgenden einzeln erläutert, da sie aufeinander aufbauen. Bezüglich der Notation sei angemerkt, dass eine observierte Zufallsvariable in normaler Schrift geschrieben ist (bspw. x_i), der dazugehörige Wert, d.h. der Merkmalsvektor, aber fett gedruckt im Text auftaucht (bspw. $\mathbf{x}_i$). Zusammengefasst sind alle Variablen in Mengen X und Y.

Die erste Eigenschaft der AMNs ist die Einschränkung der Potentialfunktionen auf Cliquen mit einer oder zwei Zielvariablen (im Fall der unbedingten Markov-Netze spricht man auch von *Pairwise Markov Networks*). Dabei bilden die Cliquen mit einer Zielvariable und dem

dazugehörigen Faktor ϕ die sogenannten *Knotenpotentiale*. Das Knotenpotential gibt für die Clique an, wie gut das Merkmal zur jeweils gewählten Klasse der Zielvariable passt. Je höher das Potential, desto höher ist intuitiv auch die Kompatibilität zwischen Merkmal und gewähltem Label.

Würde das Netz ausschließlich aus diesen Knotenpotentialen bestehen, d.h. die unmittelbare Abhängigkeit des Labels von dem Merkmal, hätte man eine lokale Klassifikation. Um nun den kollektiven Kontext zu integrieren, gibt es im AMN auch die Cliquen mit zwei Zielvariablen und dem dazugehörigem Faktor ψ für die *Kantenpotentiale*. Solch ein Potential bewertet das Labeling zweier benachbarter Knoten zusammen mit einem gemeinsamen Merkmal und gibt dem Netz so die geforderte Kollektivität.

Zusammen bilden die Knoten- und Kantenpotentiale also das Gesamtpotential im Netzwerk und damit lässt sich die bedingte Wahrscheinlichkeit aus Gleichung (2.9) nun schreiben als:

$$P(Y|X) = \frac{1}{Z(X)} \prod_{i=1}^{N} \phi(x_i, y_i) \prod_{(ij)\in\mathcal{E}} \psi(x_i, x_j, y_i, y_j) \tag{2.11}$$

wobei N die Menge der zu labelnden Punkte ist und damit auch gleichzeitig die Anzahl der Knotenpotentiale. Die Partitionsfunktion Z ist dabei wie folgt:

$$Z(X) = \sum_{Y} \left(\prod_{i=1}^{N} \phi(x_i, y_i) \prod_{(ij)\in\mathcal{E}} \psi(x_i, x_j, y_i, y_j) \right) \tag{2.12}$$

Einer weiteren Bemerkung bedarf der Ausdruck des Kantenpotentials $\psi(x_i, x_j, y_i, y_j)$. Hier sind neben beiden Labels y_i, y_j auch die beiden observierten Zufallsvariablen x_i, x_j zu finden. In der Literatur findet sich auch oft der kombinierte Ausdruck x_{ij} und damit $\psi(x_{ij}, y_i, y_j)$, da in der Verrechnung ein gemeinsames Kantenmerkmal genutzt wird. Die explizite Darstellung zeigt aber jeden Einfluss auf das Potential und wird hier daher präferiert.

Die zweite Eigenschaft der AMNs ist die Überführung des Netzes in ein log-lineares Modell. Solche Modelle besitzen für jede Klasse einen Gewichtsvektor $\mathbf{w}^k$, wobei hier weiter unterschieden wird zwischen *Knotengewichten* $\mathbf{w}_n^k$ und *Kantengewichten* $\mathbf{w}_e^{k,l}$ für zwei Klassen $k, l \in \mathcal{L}$. Das log-lineare Modell nutzt weiterhin eine Definition der Potentiale im logarithmischen Maß. Das Knotenpotential wird definiert als $\log \phi(x_i, y_i) := \mathbf{w}_n^k \cdot \mathbf{x}_i$. Hier wird also das Merkmal des Punktes mit einer Gewichtung versehen, die anhand des jeweils gewählten Labels $y_i = k$ bestimmt wird. Dabei betonen die Gewichte optimalerweise bei unterschiedlichen Klassen auch unterschiedliche Teile der Merkmale, d.h. bestimmte Einträge der Merkmalsvektoren. Stimmen Merkmal und das Label mit dazugehöriger Gewichtung überein, ist das Produkt und damit das Potential groß. Wenn hingegen Merkmal und Label nicht zusammenpassen, sollte das Produkt gegen 0 streben. Analog folgt die Definition für die Kantenpotentiale mit $\log \psi(x_i, x_j, y_i, y_j) := \mathbf{w}_e^{k,l} \cdot \mathbf{x}_{ij}$. Hier wird aber in Abhängigkeit von zwei Labelings $k = y_i$ und $l = y_j$ gewichtet. Dabei ist wieder, wie vorhin erwähnt, auf

das gemeinsame Merkmal $\mathbf{x}_{ij}$ zu achten. Dieses Merkmal kann beispielsweise als Skalarprodukt $\mathbf{x}_{ij} = \mathbf{x}_i^T \cdot \mathbf{x}_j$ der beiden Knotenmerkmale definiert sein, als ein konstanter Wert oder aber jede andere beliebige Variante, die einen Zusammenhang mit Hilfe einer linearen Gewichtung auszudrücken vermag.

Die dritte Eigenschaft der AMNs ist die Einschränkung der Kantengewichte auf $\mathbf{w}_e^{k,l} = 0$ für ungleiche Labels $k \neq l$ und $\mathbf{w}_e^{k,k} \geq 0$ für gleiche Labels. Diese Eigenschaft wird *assoziativ* genannt, denn sie bevorzugt Labelings gleicher Klasse bei den Kantenpotentialen, also $\log \psi(x_i, x_j, y_i, y_j) \geq 0$, bei $y_i = y_j$. Zielvariablen ungleicher Belegung tragen bei gemeinsamer Clique nicht zum Gesamtpotential bei, denn dadurch gilt $\log \psi(x_i, x_j, y_i, y_j) = 0$, für $y_i \neq y_j$. Diese Einschränkung folgt der Idee des *Potts-Modells* [Wu, 1982] als eine Erweiterung des Ising-Modells.

Zusammengefasst können die Potentiale in einem AMN kompakt ausgeschrieben werden, wenn zusätzlich noch ein Indikator $I : \mathcal{L} \times \mathcal{L} \to \{0, 1\}$ für den Vergleich von zwei Labels genutzt wird . Die Funktion $I(l, k)$ ist genau dann 1, wenn $l = k$ und sonst 0. Analog wird der Indikator I für drei Elemente definiert mit $I(l, k, m) = 1 \Leftrightarrow l = k = m$ und sonst 0. Somit lassen sich die Potentiale schreiben als:

$$\log \phi(x_i, y_i) := \sum_{k=1}^{K} \mathbf{w}_n^k \cdot \mathbf{x}_i \cdot I(y_i, k) \tag{2.13}$$

$$\log \psi(x_i, x_j, y_i, y_j) := \sum_{k=1}^{K} \mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij} \cdot I(y_i, y_j, k) \tag{2.14}$$

mit K als die Anzahl der möglichen Objektklassen.

Dieses Modell besitzt somit die Gewichtung als Parametrisierung $\theta = \mathbf{w} = (\mathbf{w}_n, \mathbf{w}_e)$, d.h. den Vektor der Gewichtsvektoren. Schreibt man daher Gleichung (2.11) zusammen mit einer Parametrisierung $\mathbf{w}$ wie folgt um:

$$P_{\mathbf{w}}(Y|X) = \exp(\log(\frac{1}{Z_{\mathbf{w}}(X)} \prod_{i=1}^{N} \phi(x_i, y_i) \prod_{(ij)\in\mathcal{E}} \psi(x_i, x_j, y_i, y_j)))$$

dann ergibt sich durch Anwendung der Logarithmenregeln:

$$P_{\mathbf{w}}(Y|X) = \exp(\sum_{i=1}^{N} \log \phi(x_i, y_i) + \sum_{(ij)\in\mathcal{E}} \log \psi(x_i, x_j, y_i, y_j) - \log Z_{\mathbf{w}}(X)) \tag{2.15}$$

Zu beachten ist hier, dass Z sich nun auch in Abhängigkeit von $\mathbf{w}$ ändert. Dies folgt sofort aus der Definition der Faktoren ϕ, ψ. Somit muss trotz gegebener Menge X, die Funktion Z bei einer Änderung von $\mathbf{w}$ aufgrund geänderter Potentiale erneut berechnet werden.

In vielen Anwendungsbereichen der Stochastik ist es sinnvoll, Berechnungen logarithmiert durchzuführen, da durch die Umwandlung von Produkten zu Summen Vorteile entstehen.

Erstens sind Additionen schneller auszuführen und zweitens sind die Berechnungen numerisch stabiler, da viele Zahlen unterschiedlicher Skalierungen miteinander addiert und nicht multipliziert werden. Dies wird hier ebenfalls durchgeführt und damit folgt für Gleichung (2.15) zusammen mit (2.13) und (2.14):

$$\log P_{\mathbf{w}}(Y|X) = \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbf{w}_n^k \cdot \mathbf{x}_i \cdot I(y_i, k) + \sum_{(ij)\in\mathcal{E}}\sum_{k=1}^{K} \mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij} \cdot I(y_i, y_j, k) - \log Z_{\mathbf{w}}(X) \quad (2.16)$$

und damit die grundlegende Definition der bedingten Wahrscheinlichkeit in einem AMN.

2.3 Inferenz in AMNs

Dieser Abschnitt soll erläutern, wie in einem AMN Inferenz betrieben werden kann. Das bedeutet im vorliegenden Kontext, dass ein auf bekannten Daten konditioniertes Modell mit einer Gewichtung $\theta = \mathbf{w}$ existiert und dass eine Menge X mit Merkmalsvektoren gegeben ist sowie eine dazugehörige Menge Y mit Variablen unbekannter Belegung. Im Sinne der Klassifikation mittels überwachten Lernens, erfolgt die Inferenz in der zeitlichen Abfolge nach dem Lernen. Jedoch nutzen die später erläuterten Lernverfahren Inferenz für Zwischenschritte, und daher sollte dieser Abschnitt zuerst behandelt werden.

Das Ziel ist also das Finden der MAP-Belegung $\mathbf{y}^*$, sodass gilt: $\max P_{\mathbf{w}}(\mathbf{y}|\mathbf{x}) = P_{\mathbf{w}}(\mathbf{y}^*|\mathbf{x})$. Im Allgemeinen ist Inferenz in Markov-Netzwerken aufgrund der Normalisierung NP-hart und damit für große Netze rechnerisch kaum zu bewältigen. Daher werden fast ausschließlich approximative Verfahren für die Inferenz genutzt. Ein Grund für effiziente Inferenz in AMNs ist die Unabhängigkeit der Partitionsfunktion in Bezug auf die Menge Y. Dies ist aufgrund der bedingten Wahrscheinlichkeitsmodellierung gegeben (vgl. Abschnitt 2.2.2). Da X und $\mathbf{w}$ konstant sind, kann die Normalisierung bei der Maximierung von $\log P_{\mathbf{w}}(\mathbf{y}|\mathbf{x})$ ignoriert werden und die Inferenz ist die Lösung von:

$$\max \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbf{w}_n^k \cdot \mathbf{x}_i \cdot I(y_i, k) + \sum_{(ij)\in\mathcal{E}}\sum_{k=1}^{K} \mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij} \cdot I(y_i, y_j, k) \quad (2.17)$$

Die erste Variante der Inferenz als Lineares Programm ist die Standardformulierung von [Taskar et al., 2004] und berechnet die Lösung analytisch. Die zweite Variante, Belief Propagation, arbeitet auf der Graphstruktur und versendet Nachrichten um die MAP-Belegung der Variablen zu ermitteln. Belief Propagation wird im nächsten Kapitel als alternative Inferenzmethode erläutert. Mathematisch hängen diese Methoden aber zusammen und werden dementsprechend auch oft im Zusammenhang untersucht (bspw. [Yanover et al., 2006]). Für die approximative Inferenz werden in der Literatur auch beispielsweise oft Graph-Cut-Algorithmen verwendet, basierend auf der Arbeit von [Boykov et al., 2001]. Im Rahmen der Bachelorarbeit wurde der Fokus auf Belief Propagation als Inferenzverfahren der Wahl gesetzt.

2.3.1 Lineare Programmierung

Ein *Lineares Programm* (LP) ist die Formulierung eines konvexen Optimierungsproblems $\max\{c^T x | Ax \leq b, x \in \mathbb{R}^n\}$ mit einer linearen Zielfunktion und linearen Nebenbedingungen [Boyd and Vandenberghe, 2004]. Dabei ist x der Vektor der Variablen, c der Vektor der Koeffizienten und A und b bilden die Nebenbedingungen des Optimierungsproblems. Eine Untergruppe der LPs sind die sogenannten *Ganzzahligen Linearen Programme* (ILP), welche zusätzlich die Ganzzahligkeit der Variablen ($x \in \mathbb{Z}^n$) fordern und im Allgemeinen NP-harte Probleme darstellen. LPs und ILPs sind gründlich erforscht und es gibt zahlreiche effiziente Verfahren zur Lösung dieser Probleme.

Das Inferenzproblem in einem AMN lässt sich sehr einfach in ein ILP überführen. Die ganzzahligen Variablen im ILP entsprechen aber nicht den diskreten Variablen aus Y in der ursprünglichen Form und das Problem ist im binären Fall ($K = 2$) zwar in polynomieller Zeit exakt lösbar, für $K > 2$ jedoch NP-hart. Durch Relaxation (die Forderung der Ganzzahligkeit wird verworfen) entsteht aber eine beweisbare 2-Approximation zur optimalen Lösung für $K > 2$, welche anschliessend noch u.U. mit geeigneten Verfahren gerundet wird. Ausserdem erhält man im binären Fall weiterhin eine ganzzahlige Lösung [Taskar et al., 2004].

Taskar et al. benutzen im ILP für die Variablen aus Y eine 1-aus-n-Kodierung zur Darstellung des Labelings und damit wird jede Variable $y_i \in \{k_1, ..., k_K\}$ in K binäre Variablen $y_i^1, ..., y_i^K \in \{0, 1\}$ ausgeschrieben. Damit lassen sich die Variablen als Indikatoren auffassen wobei $\sum_{k=1}^{K} y_i^k = 1$ immer gilt, d.h. eine Variable y_i kann immer nur genau ein Label besitzen. Weiter werden für jede Kante im Graphen K^2 neue Variablen eingeführt, also für eine Kante zwischen Knoten i und j die binären Variablen $\{y_{ij}^{k,l} | k, l \in \mathcal{L}, y_{ij}^{k,l} \in \{0, 1\}\}$ als Indikatoren, welche Labels die Knoten der Kante besitzen. Folgerichtig gilt also immer die Bedingung, dass $y_{ij}^{k,l} = y_i^k \cdot y_j^l$. Durch die dritte Einschränkung der AMNs sind ausserdem nur Kanten mit Knoten gleicher Klasse von Belang, d.h. die Variablen $y_{ij}^{k,l}$ mit $k \neq l$ werden nicht betrachtet obwohl sie ebenfalls im ILP als Variablen auftauchen. Weiter lässt sich die vorherige Bedingung linearisieren in zwei Bedingungen $y_{ij}^{k,k} \leq y_i^k$ und $y_{ij}^{k,k} \leq y_j^k$. Damit besitzt das ILP insgesamt $K \cdot N + K^2 \cdot |\mathcal{E}|$ Variablen und das relaxierte LP lautet:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbf{w}_n^k \cdot \mathbf{x}_i \cdot y_i^k + \sum_{(ij) \in \mathcal{E}} \sum_{k=1}^{K} \mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij} \cdot y_{ij}^{k,k} \\
s.t. & \\
\forall i, k: \quad & y_i^k \geq 0 \\
\forall i: \quad & \sum_{k=1}^{K} y_i^k = 1 \\
\forall (ij) \in \mathcal{E}, k: \quad & y_{ij}^{k,k} \leq y_i^k, \quad y_{ij}^{k,k} \leq y_k^k
\end{aligned}
\tag{2.18}
$$

Nach Lösen des relaxierten LPs lassen sich die Labelings und damit die MAP-Belegung aus den (u.U. noch gerundeten, da relaxierten) Indikatorvariablen y_i^k auslesen.

2.4 Überwachtes Lernen mit AMNs

Dieser Abschnitt wird erläutern, wie ein AMN mittels überwachten Lernens effizient trainiert werden kann. Das bedeutet, dass ein Datensatz X mit Merkmalsvektoren gegeben ist und dazu die Labelmenge $\hat{Y}$ mit bekannter Belegung bzw. der korrekte Vektor $\hat{\mathbf{y}}$. Das Ziel ist nun also, ausgehend von den gegebenen Daten, die möglichst beste Gewichtung $\theta^* = \mathbf{w}$ zu bestimmen, sodass $\mathbf{w} = \text{argmax}_\theta P_\theta(\hat{Y}|X)$ gilt. Im Gegensatz zur Inferenz kann hier jedoch nicht wie bei Gleichung (2.17) der Term $Z_\mathbf{w}(X)$ ignoriert werden, da sich die Partitionsfunktion in ihrem Wert während des Lernens aufgrund der Änderung von $\mathbf{w}$ ebenfalls ändert. Damit wäre das Problem in seiner jetzigen Form NP-hart und damit nicht effizient zu lösen.

Daher verwenden [Taskar et al., 2004] in ihrem sogenannten M^3N (maximum margin markov network) die Idee, den Klassifikationsabstand zwischen dem korrekten Labeling und jedem anderen Labeling $\mathbf{y} \neq \hat{\mathbf{y}}$ zu maximieren. Diese Idee ähnelt den Support Vector Machines (SVM, [Muller et al., 2001]), welche ebenfalls versuchen den Abstand zwischen korrektem und falschem Labeling maximal zu halten. Um dies zu erläutern sei noch einmal die Gleichung (2.16) gegeben mit dem korrekten Labeling $\hat{\mathbf{y}}$. Zusätzlich werden wie beim LP (2.18) die Variablen aus $\hat{Y}$ als Indikatoren in binäre Variablen ausgeschrieben. Dann ergibt sich für die Wahrscheinlichkeit:

$$\log P_\mathbf{w}(\hat{Y}|X) = \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbf{w}_n^k \cdot \mathbf{x}_i \cdot \hat{y}_i^k + \sum_{(ij)\in\mathcal{E}}\sum_{k=1}^{K} \mathbf{w}_e^{k,k} \cdot \mathbf{x}_{ij} \cdot \hat{y}_{ij}^{k,k} - \log Z_\mathbf{w}(X) \tag{2.19}$$

Durch geeignetes Umformulieren lassen sich nun die beiden Doppelsummen in eine kompaktere (lineare) Darstellung bringen, sodass gilt:

$$\log P_\mathbf{w}(\hat{Y}|X) = \mathbf{w}\mathbf{X}\hat{\mathbf{y}} - \log Z_\mathbf{w}(X) \tag{2.20}$$

wobei die Matrix $\mathbf{X}$ für jede Klasse und jedes Klassenpaar die Knoten- und Kantenmerkmale dupliziert und in einer gewissen Anordnung hält, $\mathbf{w}$ der Vektor der Gewichtsvektoren ist und $\hat{\mathbf{y}}$ die Variablen in erwähnter 1-aus-n-Kodierung darstellt.

Das Ziel ist nun das Maximieren des Abstandes und damit gilt für ein gegebenes $\mathbf{X}$ und $\mathbf{w}$ folgender Zusammenhang:

$$\begin{aligned} \log P_\mathbf{w}(\hat{Y}|X) - \log P_\mathbf{w}(Y|X) &= \mathbf{w}\mathbf{X}\hat{\mathbf{y}} - \log Z_\mathbf{w}(X) - (\mathbf{w}\mathbf{X}\mathbf{y} - \log Z_\mathbf{w}(X)) \\ &= \mathbf{w}\mathbf{X}(\hat{\mathbf{y}} - \mathbf{y}) \end{aligned} \tag{2.21}$$

aufgrund der gleichen Partitionsfunktion. Anzumerken ist, dass die Partitionsfunktion nicht mehr im Ausdruck auftaucht und daher das Problem nun effizient berechenbar ist.

2.4.1 Quadratische Programmierung

Ein *Quadratisches Programm* (QP) ist, ähnlich einem LP, die Formulierung eines konvexen Optimierungsproblems $\max\{\frac{1}{2}x^TQx + c^Tx | Ax \leq b, x \in \mathbb{R}^n\}$ mit einer quadratischen Zielfunktion und linearen Nebenbedingungen [Boyd and Vandenberghe, 2004]. Dabei ist hier x der Vektor der Variablen, c der Vektor der Koeffizienten und A und b bilden die Nebenbedingungen des Optimierungsproblems. Neu ist hier allerdings die Matrix Q, welche für ein QP die quadratischen Koeffizienten hält. Auch für QPs gibt es, besonders aufgrund der Konvexität der Probleme, effiziente und gut erprobte Verfahren zur Lösung.

Ausgehend von Gleichung (2.21) haben Taskar et al. folgendes QP aufgestellt:

$$\begin{aligned} min \quad & \frac{1}{2}||\mathbf{w}||^2 + \xi \\ s.t. \quad & \\ \forall \mathbf{y} : \quad & \mathbf{wX}(\hat{\mathbf{y}} - \mathbf{y}) \geq N - \hat{\mathbf{y}}_n^T \mathbf{y}_n - \xi \end{aligned} \tag{2.22}$$

Die Zielfunktion ist also ein Regularisierungsterm in $\mathbf{w}$ plus einer eingeführten Schlupfvariable ξ, welche den unseparierbaren Teil der Daten hält und im Idealfall $\xi = 0$ ist. Die Nebenbedingung besagt, dass für alle möglichen Labelings der Abstand $\mathbf{wX}(\hat{\mathbf{y}} - \mathbf{y})$ niemals kleiner sein soll als die Anzahl der falsch klassifizierten Punkte abzüglich des Schlupfes. Dabei steht der Term $N - \hat{\mathbf{y}}_n^T \mathbf{y}_n$ ($\hat{\mathbf{y}}_n$,$\mathbf{y}_n$ sind die Vektoren der Indikatorvariablen für die Knoten) für die Anzahl der falschen Labelings von N Daten, da im Falle einer Übereinstimmung von $\hat{\mathbf{y}}_n$ und $\mathbf{y}_n$ in einem Label, der Wert von $\hat{\mathbf{y}}_n^T \mathbf{y}_n$ um 1 steigt und somit für den Idealfall $N - \hat{\mathbf{y}}_n^T \hat{\mathbf{y}}_n = 0$ gilt.

Das Problem an dieser Formulierung sind jedoch die exponentiell vielen Nebenbedingungen, da jedes mögliche Labeling durch das QP abgedeckt werden muss. Alternativ dazu kann aber die Menge dieser linearen Nebenbedingungen durch Äquivalenzumformungen und mit dem relaxierten LP aus Gleichung (2.18) derart nicht-linear umgeformt werden, sodass das QP folgende Form hat:

$$\begin{aligned} min \quad & \frac{1}{2}||\mathbf{w}||^2 + \xi \\ s.t. \quad & \\ & \mathbf{wX}\hat{\mathbf{y}} - N + \xi \geq \max_{\mathbf{y}} \mathbf{wXy} - \hat{\mathbf{y}}_n^T \mathbf{y}_n \end{aligned} \tag{2.23}$$

und damit das QP als Nebenbedingung selbst die MAP-Inferenz als Unterproblem enthält. Zu beachten ist, dass die Zielfunktion des relaxierten LPs nicht mehr $\mathbf{wX}$ (vgl. Umformung bei Gleichung (2.20) mit LP von Gleichung (2.18)) sondern $\mathbf{wX} - \hat{\mathbf{y}}_n^T$ lautet und $\mathbf{y}$ durch die Relaxation kontinuierlich wird.

Aufgrund der (schwachen) Dualität gilt, dass primale und duale Lösungen eines beschränkten LPs gegenseitige obere und untere Schranken vorgeben und damit das MAP-Unterproblem

durch das Duale des LPs ersetzt werden kann. Zusammen mit den Einschränkungen der Kantengewichte im AMN (vgl. 2.2.3) und kleineren Äquivalenzumformungen ergibt sich damit das folgende QP [Taskar et al., 2004]:

$$
\begin{aligned}
min \quad & \frac{1}{2}||\mathbf{w}||^2 + \xi \\
s.t. \quad & \\
& \mathbf{w}\mathbf{X}\hat{\mathbf{y}} + \xi - \sum_{i=1}^{N} \alpha_i && \geq N \\
& \mathbf{w}_e && \geq 0 \\
\forall i, k: \quad & \alpha_i - \sum_{(ij),(ji)\in\mathcal{E}} \alpha_{ij}^k - \mathbf{w}_n^k \cdot \mathbf{x}_i && \geq -\hat{y}_i^k \\
\forall (ij) \in \mathcal{E}, k: \quad & \alpha_{ij}^k + \alpha_{ji}^k - \mathbf{w}_e^k \cdot \mathbf{x}_{ij} && \geq 0 \\
\forall (ij) \in \mathcal{E}, k: \quad & \alpha_{ij}^k, \alpha_{ji}^k && \geq 0
\end{aligned}
\tag{2.24}
$$

Die zusätzlichen Variablen α_i, α_{ij}^k entspringen dabei aus dem Dualen des relaxierten LPs. Die Nebenbedingungen $\forall i : \sum_{k=1}^{K} y_i^k = 1$ aus Gleichung (2.18) werden dabei zu den Variablen α_i und die restlichen Ungleichungen des LPs sind die Variablen α_{ij}^k aus dem Dualen.

Für den binären Fall ($K = 2$) ist die LP-Relaxation exakt, damit auch das Duale und somit liefert das QP die exakten Gewichte für das Problem. Für $K > 2$ führt die Relaxation zu stärkeren Einschränkungen für $\mathbf{w}$ und damit kann das QP die optimale Lösung nicht mehr erreichen und approximiert. Wenn jedoch das relaxierte MAP-LP trotzdem ganzzahlige Lösungen liefert, dann liefert auch das approximative QP die optimale Gewichtung, da die reellen und die ganzzahligen Nebenbedingungen in diesem Spezialfall zusammenfallen.

Zusammenfassung

Dieses Kapitel erläuterte das Szenario und die Problematik der 3D-Laserscanklassifikation mittels überwachten Lernens. Dabei wurde auf die Lern- und Inferenzphase als auch auf das Thema der Modellselektion eingegangen. Die Idee einer bedingten probabilistischen Modellierung wurde motiviert sowie die Ausnutzung räumlicher Nachbarschaftsinformationen und die daraus resultierende kollektive Klassifikation. Als geeignetes Rahmenwerk wurden dann zuerst Markov-Netze als allgemeine ungerichtete graphisches Modelle und die spezielle Variante der Associative Markov Networks eingeführt. Für diese spezielle Variante wurden dann die Standardformulierungen für das Lernen als ein Quadratisches Programm und die Inferenz als Lineares Programm vorgestellt.

Kapitel 3

Datenrepräsentation und alternative Lern- und Inferenzmethoden

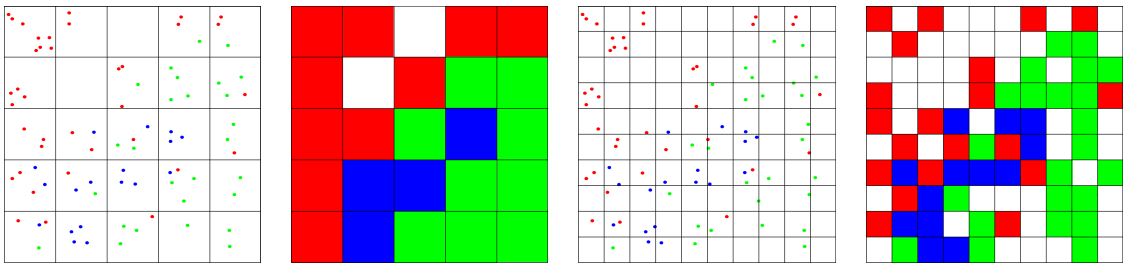
Dieses Kapitel erläutert die wichtigen Aspekte der geometrischen Datenverwaltung und der damit assoziierten, informationsbewahrenden Datenreduktion. Von Interesse ist hierbei auch die Suche nach geometrischen nächsten Nachbarn. Für eine Klassifikation sind die beschreibenden Merkmale von entscheidender Bedeutung und die Arbeit nutzt eine speziell an die die Datenverwaltung angepasste Version der Spin-Images. Zum Schluss wird für das Lernen und die Inferenz in einem Associative Markov Network jeweils eine weitere Variante präsentiert.

3.1 Geometrische Datenverwaltung

Durch die Nutzung von 3D-Laserscannern entstehen dreidimensionale Punktwolken, die sich je nach genutztem Gerät in der Dichte unterscheiden können. Der für diese Arbeit genutzte Laserscanner Velodyne HDL-64E, mit seinen 1,3 Millionen Laserpunkten in der Sekunde, führt zu einem sehr großen Satz an Daten. Eine Teilmenge dieser Daten ist jedoch für die Segmentierung kaum von Relevanz, da sehr nahe beieinander liegende Punkte keinen Mehrwert an Information bieten. Daher wird hier der Ansatz verfolgt, besonders dichte Bereiche zu fusionieren und spärlich besetzte Bereiche zu erhalten und ggf. zu stärken bzw. hervorzuheben.

3.1.1 Voxelgitter

Ein Voxelgitter unterteilt den Raum in uniforme Würfel (Voxel, volumetrischer Pixel), wobei die ideale Größe der Würfel adaptiv an die Szene und die Dichte der Punktmenge angepasst werden kann. Wenn ein Scan mit Punkten vorliegt, wird dieser in ein Voxelgitter überführt. In der Phase des überwachten Lernens, d.h. der Scan besitzt ein vorgegebenes



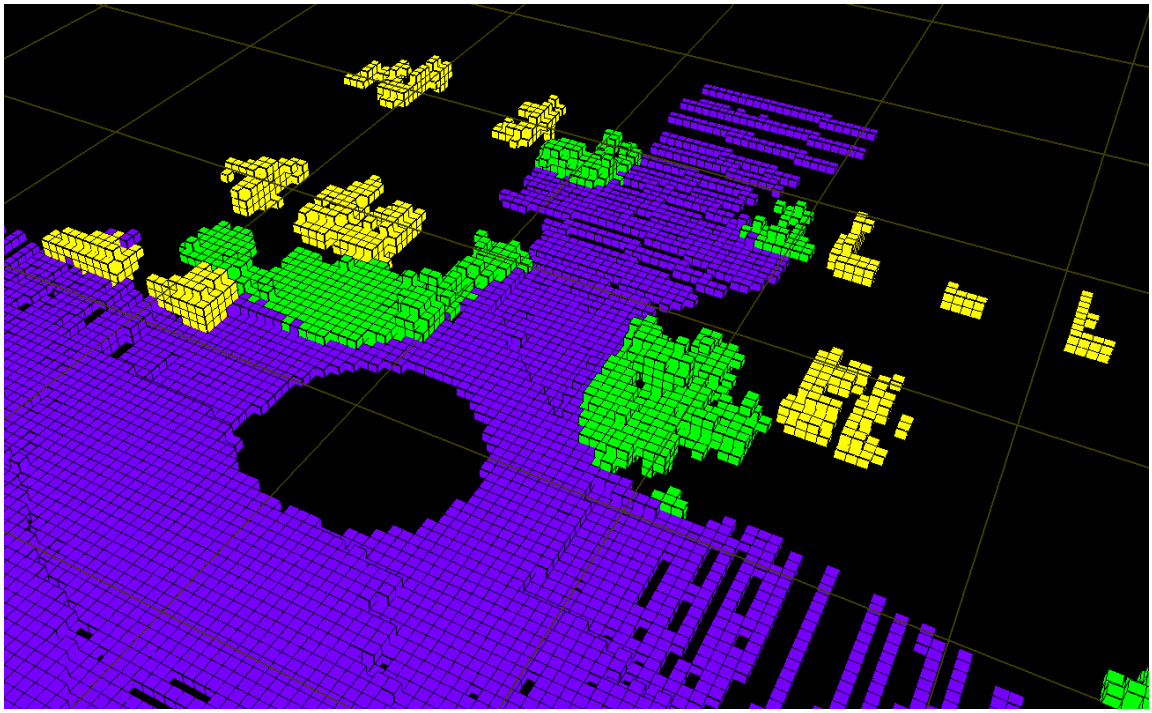
(a) (b) (c) (d)

Abbildung 3.1: Schematische Darstellung der Überführung von Scanpunkten zu Voxeldaten. In (a) ist eine Voxelgröße gewählt und in (b) wird per Mehrheitsentscheid jedem Voxel eine Klasse zugeordnet. Bei einem Klassenschwellwert von zwei ist jedoch das Voxelgitter zu grob und wird daher in (c) für den selben Scan verfeinert. Daraus resultiert eine feinere Voxelgitterstruktur (d).

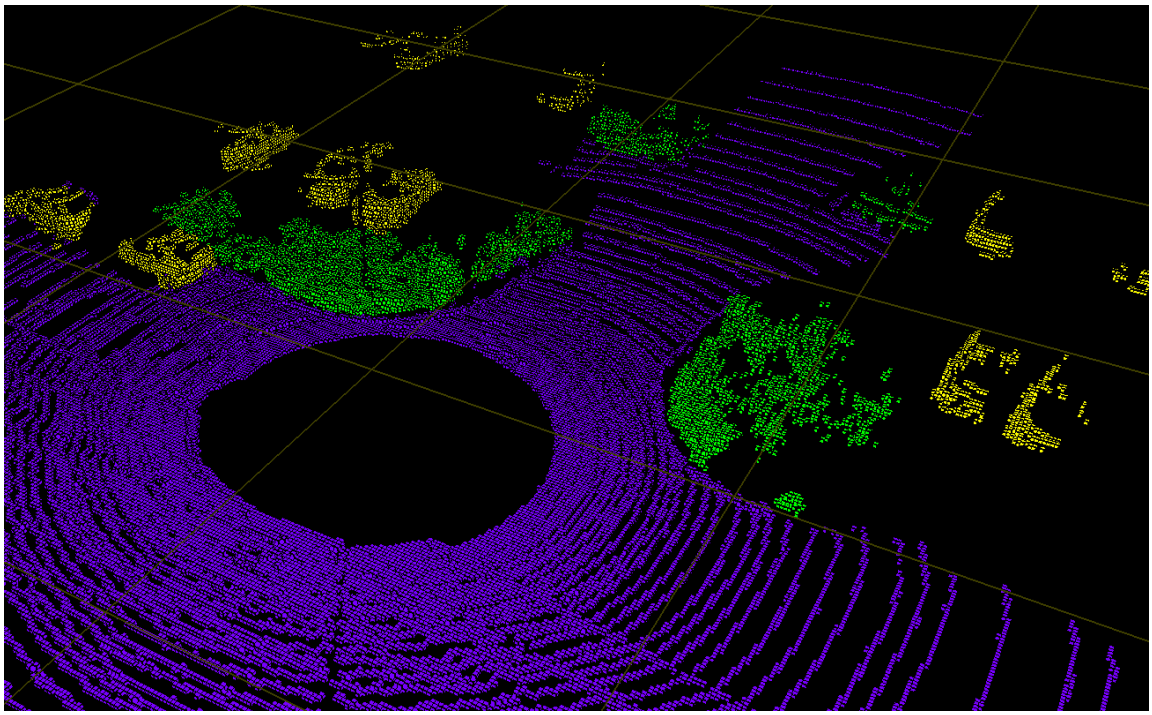
Labeling, kann ein Voxel im Raum als Histogramm über den Labelklassen aufgefasst werden. Für jedes Voxel wird gezählt, wieviele Scanpunkte einer Klasse in das Voxel hineinfallen und anschließend wird dem Voxel per Mehrheitsentscheid die Klasse mit den meisten Punkten zugewiesen. Dabei wird im Voxel die Menge der beinhaltenden Punkte für die spätere Merkmalsberechnung gespeichert. Fallen in ein Voxel mehr Scanpunkte verschiedener Klassen als ein festgelegter Klassenschwellwert, dann ist dies ein Zeichen für eine zu grobe Unterteilung und ein Hinweis, die Voxelgröße adaptiv oder manuell zu verringern. Eine schematische Darstellung findet sich in Abbildung 3.1 und eine Darstellung der genutzten Daten und ihre Unterteilung durch Voxelgitter in Abbildung 3.2.

Liegen nun in einem Voxel viele Punkte der selben Klasse, dann fusioniert ein Voxel durch seine Struktur implizit diese Informationen um Redundanz zu verringern. Wenn hingegen wenige Punkte bzw. nur ein Punkt in einem Voxel landet, dann verstärkt das Voxel automatisch diese Information. Damit werden spärlich besetzte Regionen und dicht besetzte Regionen im Voxelgitter in ihrem Informationsgehalt angepasst. Darüber hinaus wird durch diese Diskretisierung das Rauschen in den Daten abgeschwächt, denn ein leicht verrauschter Punkt landet meist im selben Voxel wie der gleiche Punkt mit geringerem Messfehler. Da der Scan auf eine abstraktere Darstellung gebracht wurde, findet das Lernen und die Inferenz nun auf Voxeldaten, und nicht mehr auf Punktdaten, statt. Wurde in der Inferenzphase dann einem Voxel eine Objektklasse zugewiesen, kann die Klasse auf alle beinhaltenden Punkte übertragen werden.

Die Konstruktion des graphischen Modells arbeitet nun also auf der Voxelstruktur. Bei einer gegebenen Anzahl N Voxeln (und nicht mehr N Punkten) gibt es für jeden Voxel jeweils eine Merkmalsvariable x_i, respektive einen berechneten Merkmalsvektor $\mathbf{x}_i$, und eine Labelvariable y_i. Für jeden Voxel wird dann für das Knotenpotential eine paarweise Clique zwischen x_i und y_i gebildet. Für die Kantenpotentiale wird eine Menge k nächster Nachbarn oder die Menge aller Nachbarn in einem festen Radius für jeden Voxel x_i ermittelt



(a)



(b)

Abbildung 3.2: Überführung des 3D-Laserscans von Abbildung 2.2 in eine grobe (a) (40cm Voxel) und eine feinere (b) (10cm Voxel) Voxelgitterstruktur. Hier sei angemerkt, dass bis auf die drei relevanten Objektklassen (Fahrzeug, Boden, Vegetation) alle anderen Daten aus den Scans entfernt wurden.

(siehe nächster Abschnitt) und für jeden gefunden Nachbarn x_j eine Clique mit beiden Labelvariablen y_i, y_j und Merkmalsvariablen x_i, x_j erstellt. Der resultierende Graph kann dann entweder für das Lernen oder für die Inferenz genutzt werden.

3.1.2 Räumliche Nachbarnsuche

Für die Berechnung der Merkmale und auch bei der Konstruktion des graphischen Modells muss eine Suche nach (nächsten) Nachbarn durchgeführt werden. Je mehr Daten vorhanden und je umfangreicher die Merkmale sind, desto wichtiger ist hier eine geometrische Datenverwaltung, welche effiziente Nachbarnsuche ermöglicht. Da nach Transformation eines Scans in ein Voxelgitter keine einzelnen Punktdaten, sondern nur noch Voxeldaten existieren, werden die Variablen x_i und die dazugehörigen Merkmalsvektoren $\mathbf{x}_i$ jeweils für Voxel definiert. Um leichter argumentieren zu können, wird eine bijektive Funktion $Voxel_{xyz} : \{x_i\} \rightarrow \mathbb{Z}^3$ eingeführt, die eine gegebene Zufallsvariable auf das dazugehörige, geometrisch eindeutige Voxel in absoluten, ganzzahligen Weltkoordinaten abbildet. Ferner gilt für zwei unmittelbar benachbarte Voxel $a, b : ||a - b|| = 1$.

Wenn nun die Kantenlänge l eines Voxel gegeben ist, und r ist der Radius der Nachbarnsuche, dann ist die Menge der benachbarten Variablen $\{x_j\}$ für eine Variable x_i definiert durch:

$$Neighbors_r(x_i) := \left\{ x_j | \frac{r}{l} \geq ||Voxel_{xyz}(x_i) - Voxel_{xyz}(x_j)||_\infty \right\} \tag{3.1}$$

wobei die Norm beliebig gewählt werden kann (hier: Maximumsnorm). Da die Berechnung der Voxelkoordinaten der Nachbarn eine konstante Zeit erfordert und die Funktion $Voxel_{xyz}(x_i)$ bijektiv ist (d.h. es ist keine weitere hierarchische Suche im Voxel nach ZVs notwendig), lässt sich daher in konstanter Zeit die Bestimmung der benachbarten Zufallsvariablen durchführen, sofern die Voxelgitterstruktur vollständig abgespeichert wird.

3.2 Merkmalsvektoren

Von entscheidender Bedeutung für robuste Klassifikation sind die beschreibenden Merkmale für die Daten. Dabei sollten die Merkmale die lokale Umgebung eines Punktes beschreiben, da ein Punkt ausser seinen inhärenten Daten (Position, Distanz zum Scanner, Remission) keine weitere Information bietet. Im Rahmenwerk des Associative Markov Network werden zwei Arten von Merkmalen unterschieden, die Knoten- und die Kantenmerkmale. Das Knotenmerkmal soll dabei das Datum an sich beschreiben während das Kantenmerkmal den Zusammenhang zwischen zwei Daten beschreibt. Als Knotenmerkmal kommt eine abgeänderte, auf Voxeln basierte Variante der Spin-Images zur Anwendung und das Kantenmerkmal wird das Skalarprodukt der Knotenmerkmale.

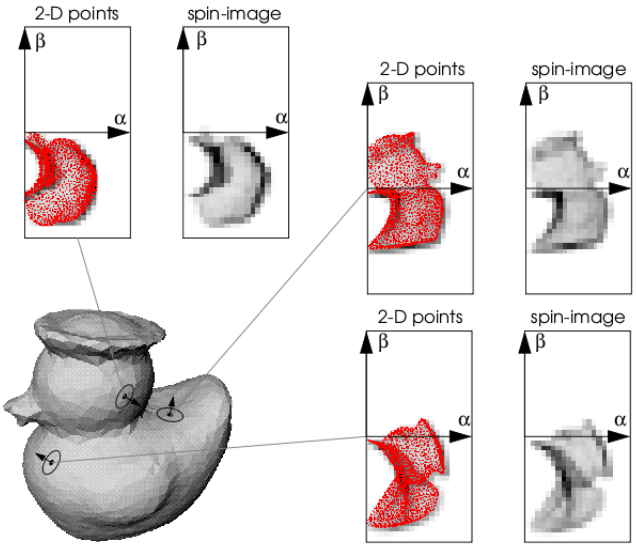


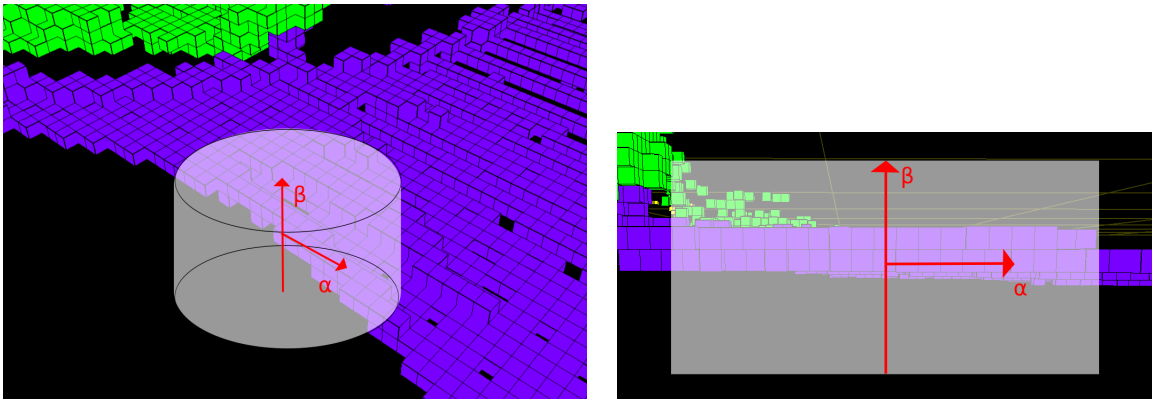
Abbildung 3.3: Exemplarische Spin-Images auf einem 3D-Modell einer Ente. Dabei wurde für drei Punkte das jeweilige Spin-Image berechnet und visualisiert (Abbildung aus [Johnson, 1997]).

3.2.1 Knotenmerkmal

Das genutzte Knotenmerkmal basiert auf der Arbeit von [Johnson, 1997]. Ein *Spin-Image* ist eine projizierte, diskretisierte zweidimensionale Darstellung der Umgebung eines Punktes. Dabei wird eine Funktion, *spin-map* genannt, dazu genutzt, die 3D-Punkte in der Nachbarschaft auf 2D-Koordinaten einer zylindrischen Basis zu überführen. Der Punkt, für den dieses Merkmal berechnet wird, sitzt dabei im geometrischen Zentrum des Zylinders und mit Hilfe einer Normalenschätzung wird die Ausrichtung des Zylinders bestimmt. In dieser zylindrischen Basisdarstellung geben dann zwei Parameter α, β an, auf welcher Kreisbahn sich ein Nachbarpunkt relativ zum Zentrum befindet. Das Merkmal ist durch die Diskretisierung ähnlich einem Histogramm, wobei ein Bin durch die zwei Werte α, β festgelegt wird und die Anzahl der Nachbarpunkte für jedes Bin gezählt wird. Anschließend wird das Spin-Image noch normalisiert. Die Erstellung ist in Abbildung 3.3 dargestellt.

Die Vorteile dieses Merkmals sind, dass metrische Informationen größtenteils erhalten bleiben, dass durch die Diskretisierung das Merkmal eine gewisse Robustheit gegen Rauschen aufweist und dass es sich invariant bezüglich rigider Transformationen, d.h. Rotation und Translation, verhält. Die Nachteile sind, dass die Normalenschätzung anfällig für Rauschen sein kann und damit das Merkmal nicht orthogonal zur (geschätzten) Oberfläche des Punktes berechnet wird und zweitens die optimale Größe des Zylinders sowie der Anzahl der Unterteilungen α, β der beiden Achsen ermittelt werden muss.

Eine neue Variante der Spin-Images sind die hier vorgestellten voxelbasierten Up-Spin-



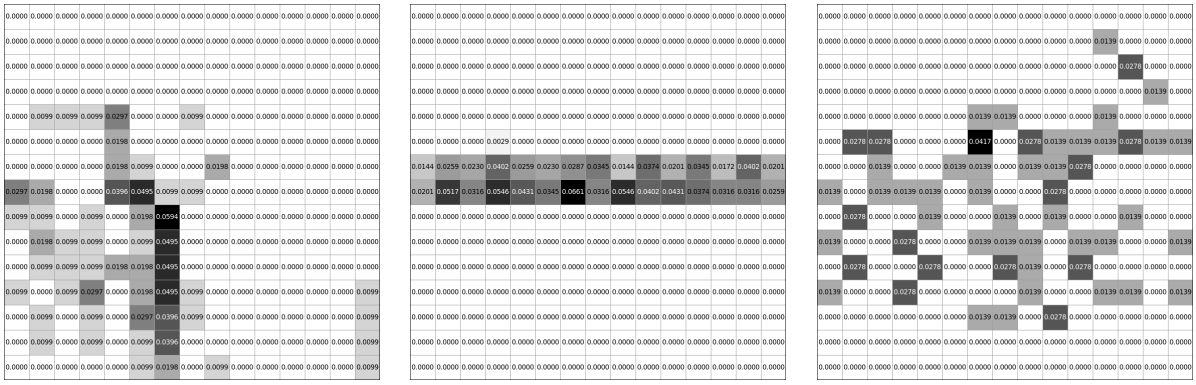
(a) (b)

Abbildung 3.4: Berechnung eines voxelbasierten Up-Spin-Image für ein Bodenvoxel aus (a) isometrischer und (b) orthogonaler Perspektive. Die Koordinaten α, β geben die (diskrete) Position der Kreisbahn eines Nachbarvoxel an. Der Zylinder ist bei dieser Variante immer nach oben ausgerichtet (in Richtung der β-Achse). Durch die zylindrische Projektion werden Voxel auf der selben Kreisbahn in den selben Eintrag des Merkmals verrechnet und dadurch hat das resultierende Merkmal den Radius (und nicht den Durchmesser) des Zylinders als Breite.

Images. Dabei wird nicht die Normale einer Stützregion des Punktes berechnet um den Zylinder auszurichten, sondern der Up-Vektor (d.h. der Vektor der genau nach oben zeigt) als Normale angenommen. Damit haben alle Zylinder eine eindeutige, nach oben zeigende Ausrichtung und die fehlerträchtige Normalenschätzung entfällt. Der Nachteil ist hier jedoch, dass die Rotationsinvarianz teilweise verschwindet und damit bspw. ein Autopunkt an einem Hang aufgrund des schrägen Untergrundes ein anderes Merkmal besitzen würde als auf einer planaren Ebene. Eine weitere Veränderung ist die Nutzung der Voxel, und nicht der Punkte, zur Berechnung des Merkmals. Damit verschwinden die Parameter α, β zur Unterteilung des Raums, denn diese sind nun identisch zur Voxelgröße und dadurch implizit bestimmt durch die Punktdichte und das resultierende Voxelgitter. Ausserdem ist durch die schnelle Nachbarnsuche im Voxelgitter das Merkmal effizient berechenbar, denn es werden in der Größe des Zylinders alle Voxel eingesammelt und die Anzahl der Punkte in den Voxel-Histogrammen pro Bin zusammengerechnet und anschliessend normalisiert. Die Erstellung des Merkmals ist in Abbildung 3.4 dargestellt und exemplarische Merkmale sind in Abbildung 3.5 zu ersehen.

3.2.2 Kantenmerkmal

Als Kantenmerkmal für die Kollektivität im AMN kam das Skalarprodukt von zwei voxelbasierten Up-Spin-Images zur Anwendung: $\mathbf{x}_i^T \cdot \mathbf{x}_j = \sum_{m=1}^{M} x_i^m \cdot x_j^m$. Die Motivation ist, dass zwei sich ähnelnde Merkmalsvektoren ein großes gemeinsames Skalarprodukt haben, da die



(a) (b) (c)

Abbildung 3.5: Visualisierung exemplarischer voxelbasierter Up-Spin-Images für ein Autovoxel (a), ein Bodenvoxel (b) sowie ein Vegetationsvoxel (c). In (a) erkennt man deutlich die Silhouette eines Fahrzeugs und in (b) die planare Gestalt des Bodens. In (c) sieht man, dass Vegetationsvoxel relativ strukturlose Spin-Images besitzen. Die Voxelgröße ist 10cm und die Größe des Zylinders ist $2 \cdot 10cm \cdot 15$ in der Breite und $10cm \cdot 15$ in der Höhe (insgesamt 15x15 Bins). Dabei repräsentieren im Merkmal dunkle Bereiche größere Werte und helle Bereiche kleinere Werte.

korrespondierenden Bins bzw. Einträge in den Merkmalsvektoren ähnlich sind. Unterscheiden sich die Merkmale, dann sind auch die korrespondierenden Einträge unterschiedlich und damit sollte das Produkt kleiner sein. Diese Annahme ist aber nur dann gültig, wenn die Merkmale die Klassen auch gut voneinander separieren können. Wenn die Merkmale wenig aussagekräftig sind und bspw. Boden und Auto ähnlich aussehende Merkmale aufweisen, dann ist das Skalarprodukt eine schlechte Wahl, denn das Produkt wäre relativ groß. Im Laufe der Arbeit zeigten sich die Knotenmerkmale jedoch als aussagestark und damit ist das Skalarprodukt als Kantenmerkmal geeignet. Als Vergleich sei hier wieder auf Abbildung 3.5 verwiesen. Das Skalarprodukt von (a) und (b) wäre relativ klein, da sich nur wenige Bins mit dunkleren Bereichen in beiden Merkmalen überschneiden (hauptsächlich in der Mitte). Ausserdem sollte ein Kantenmerkmal in seiner Dimension nicht zu groß gewählt sein, da die Kantenanzahl überproportional mit der Anzahl der Knoten wächst und somit der Berechnungsaufwand schnell ansteigt.

3.3 Alternative Lern- und Inferenzmethoden

Neben den Standardformulierungen von [Taskar et al., 2004] gibt es in der Literatur weitere Ansätze für das Lernen und die Inferenz in einem AMN. Belief Propagation als ein allgemein gängiges Verfahren wird hier für die Inferenz in Graphen vorgestellt. Für das überwachte Lernen der Parameter wird als Alternative das Subgradientenverfahren erläutert.

3.3.1 Belief Propagation

Basierend auf der Arbeit von [Pearl, 1988] ist *Belief Propagation* (BP) ein oft genutztes Verfahren zur (auch approximativen) Inferenz in beliebigen graphischen Modellen. BP berechnet Nachrichten zwischen verbundenen Knoten im Graphen bis diese Nachrichten auf Evidenzen stoßen. Dadurch ist es möglich, Variablen unbekannter Belegung zu marginalisieren und die maximierende Verbundwahrscheinlichkeit (MAP) zu finden. BP führt bei Bäumen und Polytrees zu exakten Lösungen. Bei zyklischen Graphen muss dieses Verfahren in einer Wiederholschleife ausgeführt werden, daher die Bezeichnung *loopy belief propagation*, und für solche Graphen ist im Allgemeinen keinerlei Konvergenzgarantie gegeben. Dennoch gibt es für zyklische Graphen bei bestimmten Einschränkungen Konvergenzbeweise und BP ist ein aktives Forschungsgebiet mit zahlreichen abgeänderten Varianten, wie bspw. Generalized BP [Yedidia et al., 2000] oder auch Tree-reweighted BP [Wainwright et al., 2003].

Um Belief Propagation für MAP anschaulich zu erklären, soll zuerst die Marginalisierung einer Variable mit BP erläutert werden. Sei nun der Graph (a) aus Abbildung 3.6 gegeben mit Faktoren $\phi(v_i, v_j), \phi(v_j, v_k)$ und die Variable v_i aus diesem Graph soll marginalisiert werden. Die Wahrscheinlichkeit für eine Belegung von v_i lässt sich schreiben als:

$$\begin{aligned} P(v_i) &= \sum_{v_j} \sum_{v_k} \frac{1}{Z} \cdot \phi(v_i, v_j) \cdot \phi(v_j, v_k) \\ &= \frac{1}{Z} \cdot \sum_{v_j} \phi(v_i, v_j) \cdot \sum_{v_k} \phi(v_j, v_k) \\ &= \frac{1}{Z} \cdot \sum_{v_j} \phi(v_i, v_j) \cdot m_{kj}(v_j) \\ &= \frac{1}{Z} \cdot m_{ji}(v_i) \end{aligned} \tag{3.2}$$

und damit als eine Funktion in der Variable selbst. Dabei wird m_{ji} als eine Nachricht von Knoten j nach Knoten i bezeichnet. Diese Nachrichten sind die Zwischenschritte im Sum-Product-Algorithmus[Kschischang et al., 2001], welches das Standardverfahren des BP für die Marginalisierung einer oder mehrerer Variablen darstellt. Eine Nachricht bildet dabei das Produkt aus Marginalisierung der aktuellen Variable mit den Nachrichten (also den verketteten Marginalisierungen) anderer Knoten.

Für den Fall von paarweisen Markov-Netzwerken mit diskreten Variablen berechnet sich eine Nachricht m_{ij} von Knoten j zu Knoten i mit den mit einem Wert belegten Variablen v_i und v_j wie folgt [Jordan and Weiss, 2002]:

$$m_{ji}(v_i) = \sum_{v_j} \left(\phi(v_j) \cdot \psi(v_i, v_j) \cdot \prod_{k \in ne(j) \setminus \{i\}} m_{kj}(v_j) \right) \tag{3.3}$$

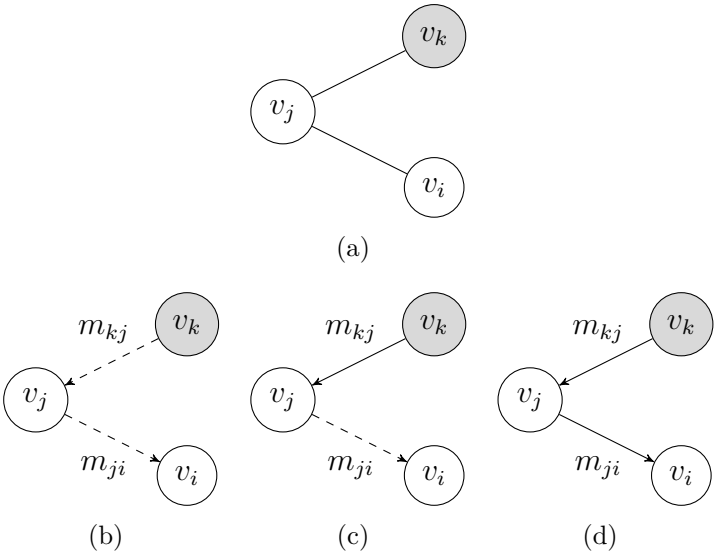


Abbildung 3.6: Nachrichtenberechnung bei BP. Ausgehend vom Markov-Netzwerk (a) soll die Variable v_i marginalisiert werden. Dafür muss die Nachricht m_{ji} berechnet werden (b). Dafür braucht wiederum v_j die Nachricht m_{kj}, welche bei (c) nun vorliegt, da v_k evident ist. Bei (d) kann nun die Information zurück propagiert und m_{ji} berechnet werden.

Für eine Belegung der Variablen v_i, v_j werden also Knoten- und Kantenpotential verrechnet sowie alle Nachrichten zu Knoten j von anderen Knoten außer i. Dies wird dann für jede Belegung von v_j durchgeführt und aufsummiert.

Da für die Berechnung die Nachrichten m_{kj} anfangs noch nicht existieren, werden diese also selbst iterativ oder rekursiv berechnet. Wenn das Verfahren auf Evidenz trifft, wird diese zurück propagiert. Wenn ein Knoten dann alle Nachrichten der Nachbarn erhalten hat, kann er dann ebenfalls Nachrichten versenden. Die einzelnen Schritte sind schematisch in Abbildung 3.6 zu ersehen. Es sei noch angemerkt, dass die Notation hier nicht unterscheidet zwischen gegebenen und zu inferrierenden Variablen ($\{v_i\} = X \cup Y$) und dass im Allgemeinen Belief Propagation von der Evidenz aus startet um möglichst performant zu arbeiten (bezogen auf Abbildung 3.6 wäre der Start des Verfahrens also auf Knoten v_k).

Sind alle Nachrichten propagiert, dann kann die Verteilung einer marginalisierten Variable v_i bestimmt werden wie folgt:

$$P(v_i|X) \propto \phi(v_i) \prod_{j \in ne(i)} m_{ji}(v_i) \tag{3.4}$$

und ist damit also proportional zum Produkt des Knotenpotentials mit allen eintreffenden Nachrichten.

Die Bestimmung der gesuchten MAP-Belegung folgt analog. Anstatt über alle Belegungen zu summieren, wird bei der Berechnung der Nachrichten das Maximum gewählt. Das führt

dazu, dass sich die MAP-Belegung analog zu Gleichung (3.2) schreiben lässt als:

$$\begin{aligned}\underset{v_i,v_j}{\operatorname{argmax}} P(v_i, v_j|v_k) &= \max_{v_i}\max_{v_j} \frac{1}{Z}\cdot\phi(v_i, v_j)\cdot\phi(v_j, v_k)\\ &= \frac{1}{Z}\cdot\max_{v_j}\phi(v_j, v_k)\cdot\max_{v_i}\phi(v_i, v_j)\\ &= \frac{1}{Z}\cdot\max_{v_j}\phi(v_j, v_k)\cdot m_{ij}(v_j)\\ &= \frac{1}{Z}\cdot m_{jk}(v_k)\end{aligned} \tag{3.5}$$

und damit als Funktion in der Evidenz v_k. Dies ist die Grundidee hinter dem Max-Product-Algorithmus zum Finden der MAP-Belegung in einem probabilistischen graphischen Modell. Damit folgt auch analog zu Gleichung (3.3), dass eine Nachricht für den Fall eines paarweisen Markov-Netzwerkes sich schreiben lässt als [Jordan and Weiss, 2002]:

$$m_{ji}(v_i) = \max_{v_j}\left(\phi(v_j)\cdot\psi(v_i, v_j)\cdot\prod_{k\in ne(j)\setminus\{i\}} m_{kj}(v_j)\right) \tag{3.6}$$

Wenn ein Graph über Zyklen verfügt, dann können diese Nachrichten immer weiter oszillieren und damit dazu führen, dass Belief Propagation nicht konvergiert. Daher werden (loopy) BP und ihre Varianten solange ausgeführt, bis Änderungen im Graphen unter einem bestimmten Schwellwert bleiben oder aber die maximale Anzahl an Iterationen überschritten wurde. In der Arbeit [Globerson and Jaakkola, 2007] wird auch bspw. eine Variante des Max-Product-Algorithmus für das MAP-Problem vorgestellt, bei welcher die Konvergenz garantiert wird. In vielen Fällen konvergiert (loopy) BP jedoch auch ohne jegliche Garantie.

3.3.2 Subgradientenverfahren

Das Subgradientenverfahren [Boyd et al., 2003] ist ein Algorithmus zur Minimierung nichtdifferenzierbarer konvexer Funktionen und eine alternative Methode zum Lernen der Modellparameter. Ähnlich dem Verfahren des Gradientenabstiegs für differenzierbare Funktionen, welches entlang des negativen Gradienten durch die Definitionsmenge läuft, nähert man sich beim Subgradientenverfahren iterativ dem Optimum an. Es gibt jedoch signifikante Unterschiede zwischen beiden Methoden. Im Gegensatz zum Gradientenabstieg nutzt man keine exakte oder approximative Liniensuche für die Schrittweite, sondern nutzt vorher festgelegte Größen. Darüber hinaus ist das Subgradientenverfahren kein monotones Abstiegsverfahren, da der Funktionswert des Öfteren auch zunimmt und damit der zuletzt beste Wert gespeichert werden muss.

Definition 3.3.1 (Subgradient)**.** *Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine konvexe Funktion. Dann ist ein Vektor g ein Subgradient von f in x wenn gilt:*

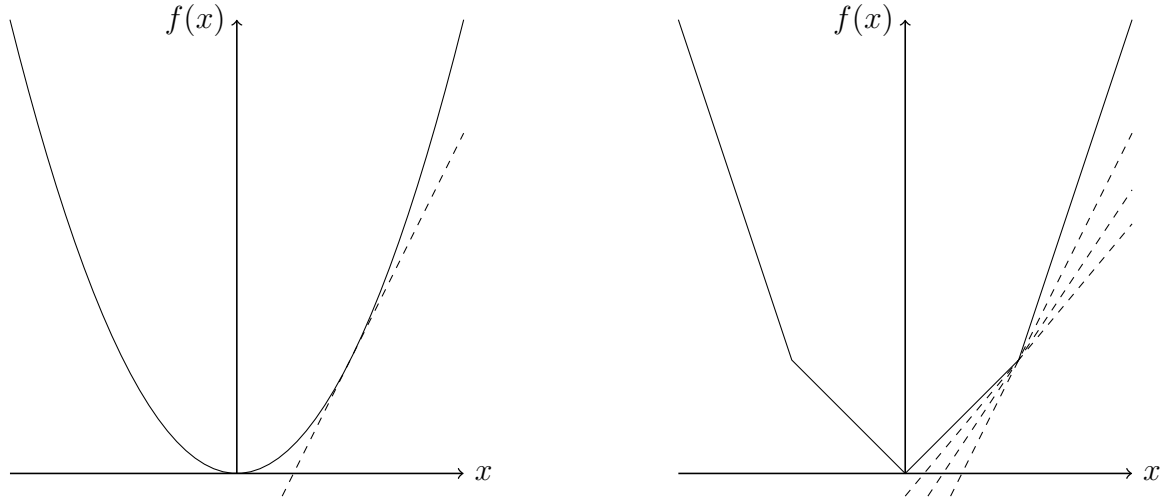


Abbildung 3.7: Gradient und Subgradienten. Die konvexe Funktion in (a) ist differenzierbar und hat an jeder Stelle genau einen (Sub)-Gradienten (visualisiert durch genau eine Tangente). Wenn f nicht-differenzierbar ist (b), dann existieren an den Unstetigkeiten eine Menge von möglichen Tangenten und damit mehrere Subgradienten.

$$\forall y \in \mathbb{R}^n : f(y) \geq f(x) + g^T(y - x)$$

Wichtig ist hierbei, dass es im nicht-differenzierbaren Falle von f an den Unstetigkeiten mehr als einen Subgradienten für x gibt. Daher wird die Menge aller Subgradienten in einem Punkt x auch Subdifferential $S(x) = \{g|g \text{ ist Subgradient in x}\}$ genannt. Ist f differenzierbar, ist das Subdifferential immer einelementig und der Subgradient an Stelle x ist genau der Gradient. Der Sachverhalt wird in Abbildung 3.7 veranschaulicht.

Das Subgradientenverfahren berechnet iterativ an der momentanen Stelle $x^{(k)}$ einen Subgradienten $g^{(k)}$ und läuft entlang diesem eine bestimmte Weite α_k in die negative Richtung. Das heißt, es entsteht eine Folge

$$x^{(k+1)} = x^{(k)} - \alpha_k \cdot g^{(k)} \tag{3.7}$$

und da, wie schon erläutert, das Verfahren nicht zwangsläufig in jedem Schritt minimiert, wird der beste Funktionswert zur Iteration k gespeichert:

$$f_{best}^{(k)} = \min\{f(x^{(1)}), ..., f(x^{(k)})\} \tag{3.8}$$

Es gibt verschiedene Möglichkeiten, die Schrittweite (konstant oder abnehmend) zu wählen und diese haben dann auch dazugehörige Konvergenzgarantien. Der Vorteil an diesem Verfahren ist im Gegensatz zum QP der lineare Speicheraufwand aber der Nachteil der sehr langsamen Konvergenz. Daher gibt es Alternativen wie das Stochastische Subgradientenverfahren [Boyd and Mutapcic, 2007], welches dem Subgradienten ein natürliches

Rauschen zu Grunde legt und damit in der Regel schneller konvergiert. Eine weitere Variante ist das Projizierte Subgradientenverfahren, bei welchem die konvexe Forderung $x \in \mathcal{C}$ dazu führt, dass nach jedem Schritt durch eine (orthogonale) Projektion sichergestellt wird, dass x in der konvexen Menge bleibt.

In den Arbeiten von [Ratliff et al., 2003] und [Ratliff et al., 2007] wird gezeigt, wie für Maximum-Margin-Klassifikatoren ein QP, ähnlich Gleichung (2.24), in eine nicht-differenzierbare Funktion umformuliert werden kann. Da im Optimum der Schlupf ξ des QPs von Zielfunktion und Nebenbedingung gleich ist, lautet die neue Kostenfunktion in $\mathbf{w}$:

$$c(\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}||^2 + \max_{\mathbf{y}}(\mathbf{w}\mathbf{X}\mathbf{y} + \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})) - \mathbf{w}\mathbf{X}\hat{\mathbf{y}} \tag{3.9}$$

wobei λ ein Parameter ist, der Regularisierung gegen Anpassung gewichtet und $\mathcal{L}$ eine sogenannte *loss-Funktion*, welche den Abstand des aktuell gefundenen Labelings zum optimalen Labeling angibt (vgl. $N - \hat{\mathbf{y}}_n^T \mathbf{y}_n$ aus Gleichung (2.22)). Ein Subgradient g dieser Funktion ist [Munoz et al., 2008]:

$$g = \lambda\mathbf{w} + \mathbf{X}\mathbf{y}^* - \mathbf{X}\hat{\mathbf{y}} \tag{3.10}$$

wobei $\mathbf{y}^* = \text{argmax}_{\mathbf{y}}(\mathbf{w}\mathbf{X}\mathbf{y} + \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}))$, also das optimale Labeling bezüglich gegebener Merkmale und Gewichtung. Das bedeutet, dass für jede iterative Berechnung des Subgradienten g ein beliebiges Inferenzverfahren (siehe Abschnitt 2.3) mit der aktuellen Gewichtung $\mathbf{w}$ durchgeführt werden muss. Angefangen bei $\mathbf{w}^{(1)} = \mathbf{0}$, wird das Subgradientenverfahren dann bis zur Konvergenz oder einer maximalen Anzahl Iterationen angewandt.

Der Gradient zur Iteration $k + 1$ wird wie folgt berechnet:

$$\mathbf{w}^{(k+1)} = \mathcal{P}_{\mathcal{C}}[\mathbf{w}^{(k)} - \alpha_k \cdot g^{(k)}] \tag{3.11}$$

wobei die Schrittweite bspw. abnehmend durch $\alpha_k = \frac{1}{k}$ definiert sein kann und $P_{\mathcal{C}}$ die Projektion auf konvexe Beschränkungen ist. Für AMNs ist die Beschränkung der Kantengewichte $\mathbf{w}_e$ derart, dass diese nicht-negativ sein müssen und damit $\mathbf{w}_e \geq \mathbf{0}$ nach jeder Iteration gelten muss.

Zusammenfassung

Insgesamt erläuterte dieses Kapitel zu Anfang die Nutzung von Voxelgittern zur intelligenten Datenverwaltung und -reduktion von 3D-Laserscans. Weiter wurden in diesem Kapitel die in der Arbeit genutzten Merkmale besprochen. Das Knotenmerkmal war eine abgeänderte und auf die Datenverwaltung angepasste Version der Spin-Images und das Kantenmerkmal war das Skalarprodukt der Knotenmerkmale unter der Annahme, dass die Knotenmerkmale sich je nach Objektklasse hinreichend unterscheiden. Darüber hinaus wurde für die Inferenz Belief Propagation als ein oft genutztes Verfahren für Graphen vorgestellt und das Subgradientenverfahren zuerst allgemein besprochen und dann für das vorliegende Minimierungsproblem spezifiziert.

Kapitel 4

Sequenzielle AMN-Modelle

Dieser Abschnitt beinhaltet die grundlegende Idee der sequentiellen Associative Markov Networks und führt zwei mögliche Modelle ein. Darauf aufbauend werden dann die mathematischen Grundlagen dieser Modelle formuliert und Überlegungen angestellt, wie auf diesen Modellen gelernt und inferriert werden kann.

4.1 Grundidee

Bei sensorischen Systemen erhält man oft eine Folge von Abtastungen $\mathcal{S}^{(t)}$ über die Zeit. Im Bezug auf 3D-Laserscanner führt diese sequentielle Abtastung zu Überlappungen in der räumlichen Szene, dargestellt in Abbildung 4.1. Das heißt, ein Teil der Voxelmenge eines Scans ist in einem konsekutiven Scan weiterhin zu finden. Es wäre daher also ratsam, diese temporalen Informationen auszunutzen um das Klassifikationsergebnis positiv zu beeinflussen. Da jedoch die Welt dynamisch ist und sich damit Objekte in der Umgebung bewegen können, ist das Ausnutzen aller vergangenen Scans nicht sinnvoll und würde auch zu falschen Ergebnissen führen. Die Beschränkung auf den jeweils vorherigen Scan minimiert zum einen den Fehler durch sich bewegende Abtastobjekte und zum anderen den Fehler, der sich bei der Berechnung des eigenen Standortes über die Zeit vergrößert. Darüber hinaus verringert sich ebenso der temporale Einfluss falscher Daten bzw. Merkmale und der Einfluss inkorrekter Klassifikation.

Das Ziel dieses Kapitels ist die Formulierung neuer graphischer Modelle, aufbauend auf dem AMN-Rahmenwerk, um temporale (bzw. sequentielle) Informationen in die Ermittlung der Maximum-A-Posteriori-Berechnung einzugliedern. Dazu wird zur besseren Visualisierung der Graph eines Scans zum Zeitpunkt t wie in Abbildung 4.2 abstrahiert. Die Menge aller gegebenen Zufallsvariablen werden zu einer Menge X^t und die zu ermittelnden Zufallsvariablen zu einer Menge Y^t zusammengefasst. Dabei sind Kanten in der Zeit diejenigen, welche Knoten mit einem unterschiedlichen Zeitindex verbinden. Diese temporalen Kanten sind ebenfalls in der Abbildung zu ersehen.

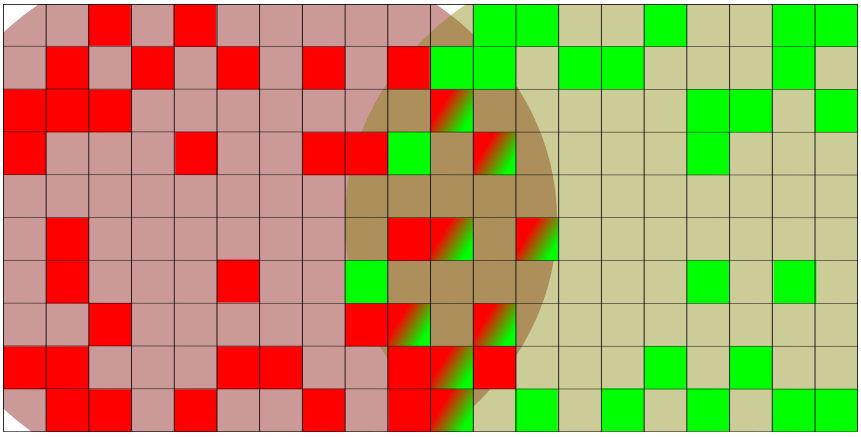


Abbildung 4.1: Schematische Darstellung einer Überlappung zweier Scans $\mathcal{S}^t$ und $\mathcal{S}^{t+\Delta t}$ aus der Vogelperspektive betrachtet. Die beiden ovalen Regionen sind die jeweiligen Abtastbereiche des Scanners zu den zwei Zeitpunkten, wobei eine Farbe einem Zeitpunkt zugeordnet ist. Die hellen, einfarbigen Voxel sind die im jeweiligen Scan auftretenden Daten. Die Voxel mit gradueller Färbung sind dabei die Daten, welche in beiden Scans auftreten und sich damit überlappen.

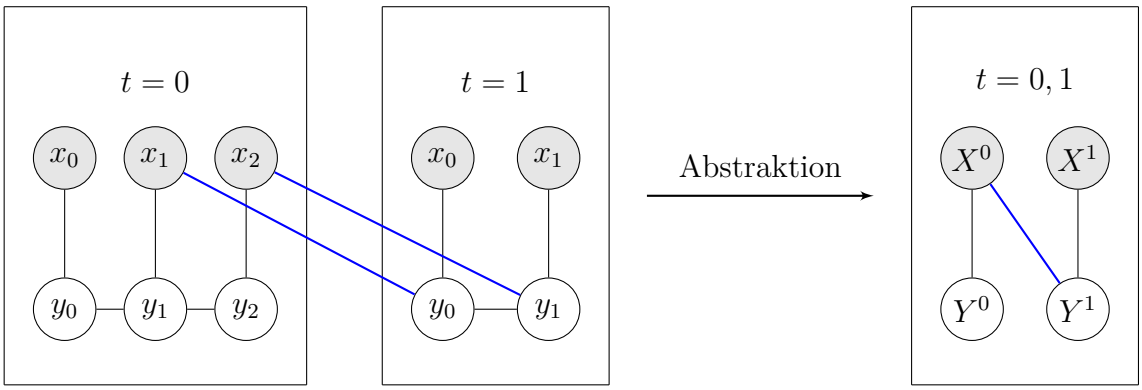


Abbildung 4.2: Abstraktion der Zufallsvariablen von zwei Scans $\mathcal{S}^0 = \{x_0, x_1, x_2\}$ und $\mathcal{S}^1 = \{x_0, x_1\}$ mit eigenem Netzwerk zu den jeweiligen Mengen X^0, X^1, Y^0, Y^1 in einem gemeinsamen Netzwerk. Zu beachten ist, dass Kanten innerhalb der Mengen Y^0, Y^1 in der abstrahierten Darstellung nicht sichtbar sind. Beispielhafte temporale Kanten (blau gefärbt) sind ausserdem ebenfalls zu ersehen. Diese werden hier ausgehend von Zeitpunkt $t = 0$ und $t = 1$ zu einer gemeinsamen Kante in der Abstraktion.

Die zwei nun vorgestellten Modelle gehen für den zeitlichen Kontext unterschiedliche Wege. Das Merkmal-sequentielle AMN hat als Annahme das Wissen über alle Scans und deren Merkmale. Damit kann in der Zeit sowohl die unmittelbare Vergangenheit als auch die unmittelbare Zukunft zur Informationsgewinnung herangezogen werden. Diese Annahme trifft auf online-Szenarien natürlich nicht zu, kann aber für offline-Klassifikation mit allen vorhandenen, d.h. auch zukünftigen, Daten getroffen werden und dient dem Entwurf eines robusten Klassifikators. Zukünftig heißt hier, dass zwei Scans $\mathcal{S}^t, \mathcal{S}^{t+\Delta t}$ existieren und dass, ausgehend von $\mathcal{S}^t$, die Informationen von $\mathcal{S}^{t+\Delta t}$ bekannt sind. Die Annahme kann natürlich auch auf die Vergangenheit eingeschränkt werden und damit kann das Modell dann auch für online-Szenarien genutzt werden. Das zweite Modell, das Label-sequentielle AMN, nutzt das Wissen der Klassifikation des vorherigen Scans und ist damit, bezogen auf den Informationsgewinn durch die Zeit, stärker eingeschränkt als das Merkmal-sequentielle AMN. Trotzdem kann dieses Modell aussagekräftiger als die Standardformulierung sein und ist ebenfalls für online-Szenarien anwendbar.

Beide Modelle werden dabei auf der Voxelstruktur arbeiten und nicht auf den Datenpunkten selbst. Dies liegt daran, dass Voxel sehr einfach die geometrische Zuordnung von Zufallsvariablen über die Zeit erlauben. Die folgenden Ansätze könnten mit Anpassung natürlich auch auf Punkten arbeiten, bedürften aber dann weiteren Überlegungen wie Suche von nächsten Nachbarn oder auch der besten Verrechnungsmethode.

Es soll an dieser Stelle darauf hingewiesen werden, dass das Ziel der Arbeit das Ausnutzen der graphischen Modellierung ist. Es wäre beispielsweise möglich, jeweils zwei sequentielle Scans zu fusionieren und auf diesen die Klassifikation durchzuführen. Dies würde höchstwahrscheinlich ebenfalls zu besseren Resultaten im Vergleich zur zeitlich isolierten Version führen, könnte aber ohne Weiteres nicht in das bestehende graphische Modell integriert werden. Ferner wäre es auch möglich, den temporalen Kontext wie oben erwähnt nicht nur sequentiell, sondern auch über mehrere Scans hinweg einzugliedern. Mit diesen erwähnten Einschränkungen ist nun jedoch der Beitrag dieser Arbeit klar umrissen und kann hinreichend untersucht werden.

4.2 Probabilistische graphische Modellierung

Es wird ausgehend von den gegebenen Mengen der Zufallsvariablen die Verteilung derart formuliert, dass diese möglichst angenehm an beide Modelle angepasst werden kann. Seien X^t die Beobachtungen zu den Zeitpunkten $t \in \{0, \ldots, T\}$ und die Y^t die jeweiligen zusammengefassten Variablen des Labelings. Dann lässt sich die bedingte Wahrscheinlichkeit ausdrücken als:

$$
\begin{aligned}
P(Y^T,\ldots,Y^0|X^T,\ldots,X^0) &= \frac{P(Y^T,\ldots,Y^0,X^T,\ldots,X^0)}{P(X^T,\ldots,X^0)} = \frac{P(Y^{T:0},X^{T:0})}{P(X^{T:0})} \\
&= \frac{P(Y^T|Y^{T-1:0},X^{T:0})\cdot P(Y^{T-1:0},X^{T:0})}{P(X^{T:0})} \\
&= \frac{P(Y^T|Y^{T-1:0},X^{T:0})\cdot P(Y^{T-1}|Y^{T-2:0},X^{T:0})\cdot P(Y^{T-2:0},X^{T:0})}{P(X^{T:0})} \\
&\vdots \\
&= \frac{\prod_{t=1}^{T} P(Y^t|Y^{t-1:0},X^{T:0})\cdot P(Y^0|X^{T:0})\cdot P(X^{T:0})}{P(X^{T:0})}
\end{aligned}
$$

Nun kürzt sich die Verteilung über die Beobachtungen $P(X^{T:0})$ im Bruch heraus. Dies ist erwünscht, da als Annahme keinerlei Aussage über die Verteilung dieser Zufallsvariablen getätigt werden kann (vgl. 2.2.2.). Es ergibt sich für die bedingte Wahrscheinlichkeit:

$$
P(Y^T,\ldots,Y^0|X^T,\ldots,X^0) = \prod_{t=1}^{T} P(Y^t|Y^{t-1:0},X^{T:0})\cdot P(Y^0|X^{T:0}) \tag{4.1}
$$

Nun wird je nach gewähltem sequentiellen Modell diese Gleichung umgeformt und auf die jeweilige Graphstruktur angepasst.

4.3 Merkmal-sequentielles AMN

Das erste hier vorgestellte Modell benutzt die Merkmalsinformationen aus dem letzten und dem nächsten Scan um die Klassifikation zu verbessern. Dieses Netz wird zukünftig als MS-AMN referenziert. Die Einbindung in die Klassifikation geschieht dabei wie folgt:

Es wird im aktuellen Scan überprüft, ob das selbe (im räumlich-globalen Sinne) Voxel im vorherigen oder auch im zukünftigen Scan ebenfalls ein Merkmal aufweist. Bei positivem Befund werden alle verfügbaren Merkmale verrechnet und als neues Merkmal in der Potentialfunktion eingesetzt. Graphisch lässt sich diese Verteilung wie in Abbildung (4.3) visualisieren. Wird die Annahme ausschließlich auf die Vergangenheit eingeschränkt, wird nur rückblickend nach Voxeln gesucht. Bezüglich der Abbildung (4.3) würden die drei Kanten $(Y^0 - X^1)$, $(Y^1 - X^2)$ und $(Y^2 - X^3)$ nicht existieren.

4.3.1 Definition

Für die Gleichung (4.1) ergibt sich durch diese Modellierung:

$$
P(Y^T,\ldots,Y^0|X^T,\ldots,X^0) = \prod_{t=1}^{T-1} P(Y^t|X^{t-1:t+1})\cdot P(Y^0|X^0,X^1)\cdot P(Y^T|X^T,X^{T-1}) \tag{4.2}
$$

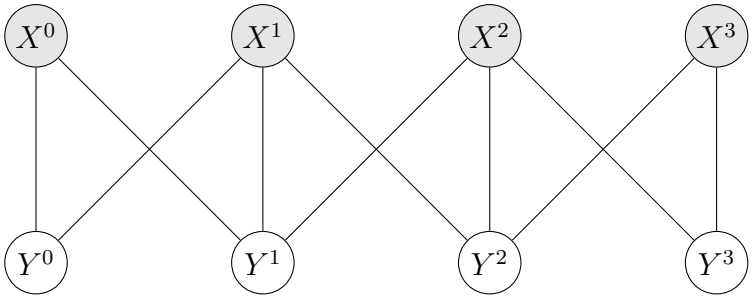


Abbildung 4.3: Abstrahierte graphische Repräsentation der bedingten Wahrscheinlichkeitsverteilung $P(Y^3, \ldots, Y^0 | X^3, \ldots, X^0)$ als MS-AMN über vier Scans. Die Merkmale werden bei dem jeweils vorherigen und nachfolgenden Scan zur Verrechnung hinzugezogen um neue, aussagekräftigere Merkmale zu erhalten.

Damit ist ein Labeling Y^t zum Zeitpunkt t ausschließlich abhängig von X^t und, falls existent, X^{t-1} und X^{t+1}. Mit Vergangenheitseinschränkung entfällt die Menge X^{t+1}.

Um nun den Kontext des MS-AMN einzubinden, wird die Potentialfunktion für die Knoten aus Gleichung (2.13) nun wie folgt erweitert:

$$\log \phi(x_i^t, y_i^t) := \sum_{k=0}^{K} w_n^k \cdot \tau_t(x_i^t) \cdot I(y_i^t, k) \tag{4.3}$$

In der Formulierung findet sich im Vergleich zu der AMN-Knotenpotentialfunktion die weitere Funktion τ_t. Diese Funktion τ_t berechnet zu einem gegebenen Knotenmerkmal x_i^t abhängig von der Zeit t ein neues Merkmal und wird hier wie folgt definiert:

$$\tau_t(x_i^t) := \begin{cases} (x_i^t + \chi_+(x_i^t))/||x_i^t + \chi_+(x_i^t)|| & , t = 0 \\ (x_i^t + \chi_-(x_i^t))/||x_i^t + \chi_-(x_i^t)|| & , t = T \\ (x_i^t + \chi_-(x_i^t) + \chi_+(x_i^t))/||x_i^t + \chi_-(x_i^t) + \chi_+(x_i^t)|| & , sonst \end{cases} \tag{4.4}$$

Die Fallunterscheidung ist notwendig, um die Funktion für die beiden Grenzfälle $t \in \{0, T\}$ wohl zu definieren. Zu erklären sind nun noch die zwei Ausdrücke χ_- und χ_+.

Die Funktion $\chi_\bullet$ gibt für ein gegebenes Merkmal x_i^t zum Zeitpunkt t entweder das Merkmal des räumlich-temporalen Vorgängers oder Nachfolgers zurück oder aber ein sich bezüglich der Merkmalsverrechnung neutral verhaltendes Objekt.

Für diese Arbeit wurde τ_t definiert als die normierte Summe der Merkmale und daher gibt $\chi_\bullet$ als Alternative den 0-Vektor zurück:

$$\chi_\bullet(x_i^t) := \begin{cases} x_j^{t\bullet 1} \in X^{t\bullet 1} & , Voxel_{xyz}(x_i^t) = Voxel_{xyz}(x_j^{t\bullet 1}) \\ \mathbf{0} & , sonst \end{cases} \tag{4.5}$$

wobei die Funktion $Voxel_{xyz}$ zu einer gegebenen Variable das globale und eindeutige Voxel zurückliefert. Dabei ist $Voxel_{xyz}$ nicht mehr zwangsläufig injektiv, da hier zwei Zufallsvariablen auf das selbe Voxel verweisen können. Das $\bullet$-Zeichen wird durch ein Vorzeichen ersetzt und bezeichnet dann die Funktion für die Zukunft$(+)$ oder die Vergangenheit$(-)$. Wird das MS-AMN auf die Vergangenheit eingeschränkt, dann ergibt sich für $\chi_+ := \mathbf{0}$ und damit automatisch das Ausblenden zukünftiger Voxel.

Die Idee dieser Verrechnung ist die Stärkung von Bereichen welche in allen Merkmalen übereinstimmen. Nicht übereinstimmende Merkmalsteile werden nämlich durch die normierte Berechnung in Richtung 0 verschoben. Dadurch werden Fehler wie Ausreißer oder auch Rauschen verringert. Andere Verrechnungen, wie beispielsweise der Mittelwert der Merkmale, wären ebenfalls denkbar.

4.3.2 Lernen und Inferenz

Da bis auf die neue Definition der Knotenpotentialfunktion die Formulierung die der Standard-AMN entspricht, verläuft das Lernen in diesem Falle identisch. Dies ist auch dadurch möglich, da die bedingte Wahrscheinlichkeit von Gleichung (4.2) ausschliesslich die $X^{T:0}$ als gegeben voraussetzt. Dies wird im nächsten Modell nicht der Fall sein. Daher können MS-AMNs auch mit der Formulierung als Quadratisches Programm gelernt werden. Um jedoch später dieses Modell mit allen anderen vergleichbarer zu gestalten, wird das Lernen mittels dem Subgradientenverfahren durchgeführt.

Auch die Inferenz verläuft in diesem Falle analog und bedarf keiner speziellen Aufmerksamkeit.

4.4 Label-sequentielles AMN

Das Label-sequentielle AMN (LS-AMN) nutzt, im Gegensatz zum MS-AMN, die Informationen über das Labeling im letzten Scan um die Klassifikation zu verbessern. Die Annahme ist, dass keinerlei Wissen über vergangene Merkmale sondern nur eine (u.U. fehlerbehaftete) Segmentierung der Welt im vorherigen Zustand zur Verfügung steht. Bevor näher auf die konkreten Details eingegangen wird, wird zuerst anhand des Graphen das Modell erläutert und die bedingte Wahrscheinlichkeitsverteilung hergeleitet.

Die Abbildung 4.4 stellt dar, wie über die Zeit die Menge der Informationen, und damit auch die Menge der Zufallsvariablen, zunimmt. Dies bedeutet, dass zum Zeitpunkt $t = 0$ ausschliesslich die Beobachtungen X^0 und die zu inferrierenden Objekte Y^0 existieren. Zum Zeitpunkt $t = 1$ ist zusätzlich zu der neuen Beobachtung X^1 auch die alte Beobachtung X^0 sowie das vorherige Labeling Y_0 als gegeben vorausgesetzt. Um zu verstehen, warum Y^1 unabhängig von den Beobachtungen von X^0 ist, wird die lokale Markov-Eigenschaft ausgenutzt (vgl. 2.2.1). Diese besagt, dass eine Variable, gegeben ihrer Markov-Hülle, unabhängig

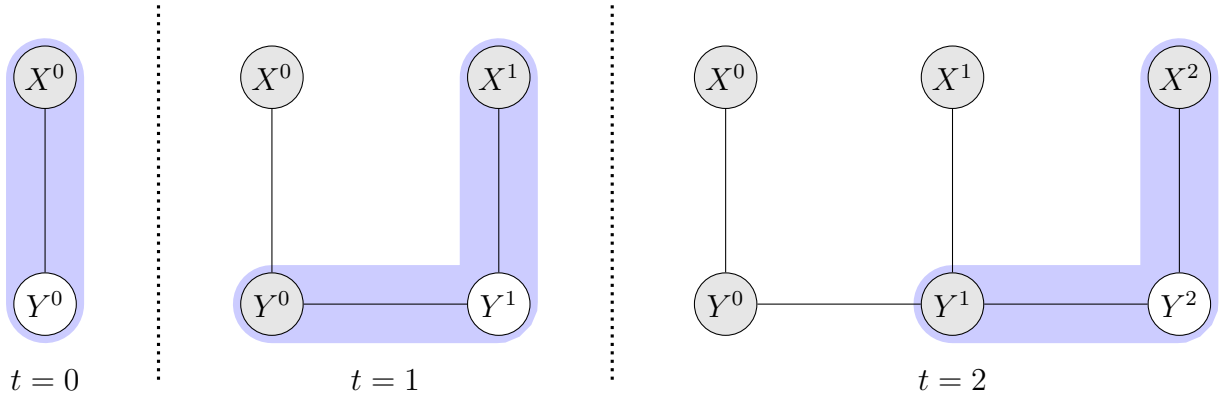


Abbildung 4.4: Konstruktion des bedingten Netzes über die Zeit. Die Menge der Variablen steigt an während gleichzeitig die Vergangenheit als gegeben angenommen wird. Die farbigen Wege visualisieren aufgrund der lokalen Markov-Eigenschaft den einzigen Einfluss auf das Labeling zum Zeitpunkt t.

vom gesamten restlichen Netz ist. Damit sind zum Zeitpunkt $t = 1$ die Abhängigkeiten mittels der Markov-Hülle von Y^1, definiert durch X^1 und Y^0, vollkommen beschrieben. Durch die iterative Konstruktion des Graphen wird die lokale Markov-Eigenschaft induktiv für alle t weitergetragen. Damit ist ersichtlich, dass in die Vergangenheit blickend nur das letzte Labeling Y^{t-1} einen zeitlichen Einfluss ausübt.

4.4.1 Definition

Mit diesem Wissen lässt sich die Gleichung (4.1) vereinfachen zu:

$$P(Y^t, \ldots, Y^0 | X^t, \ldots, X^0) = \prod_{i=1}^{t} P(Y^i | Y^{i-1}, X^i) P(Y^0 | X^0) \tag{4.6}$$

Im Gegensatz zu bspw. [Douillard et al., 2007] oder [Chen and Tang, 2007] werden hier also nicht alle Labels über die Zeit im Verbund inferriert um eine temporale Glättung zu erreichen, sondern nur die Labelvariablen im aktuellen Scan. Dieser Ansatz ist damit eine neue Herangehensweise an das Problem, da die klassifizierte Vergangenheit hier gegeben ist und damit nicht wechselwirkt.

Um diese Informationen in die Standard-Formulierung einzugliedern, wird ein temporaler, log-linearer Faktor τ für ein vergangenes und ein aktuelles Label definiert:

$$\log \tau(y_i^t, y_j^{t-1}) := \sum_{k=0}^{K} \mathbf{w}_t^k \cdot I(y_i^t, y_j^{t-1}, k) \tag{4.7}$$

wobei $Voxel_{xyz}(y_i^t) = Voxel_{xyz}(y_j^{t-1})$. Dieser Faktor gewichtet mit einer neuen Gewichtung $\mathbf{w}_t$ den Indikator für eine bestimmte Klasse genau dann, wenn vergangenes und aktuelles Label für das selbe Voxel übereinstimmen. Stimmen beide Label nicht überein, dann ist das Potential gleich 0. Damit hat die bedingte Wahrscheinlichkeit für ein LS-AMN im Vergleich zu Gleichung (2.11) die folgende Form:

$$P(Y^t|Y^{t-1}, X^t) = \frac{1}{Z(X^t)} \prod_{i=1}^{N^t} \phi(x_i^t, y_i^t) \prod_{(ij)\in\mathcal{E}^t} \psi(x_i^t, x_j^t, y_i^t, y_j^t) \prod_{(ij)\in\Delta^t} \tau(y_i^t, y_j^{t-1}) \tag{4.8}$$

Zu beachten sind hier die neuen zeitlichen Indizes sowie die neue Potentialfunktion τ. Die Menge Δ^t steht für die Menge der temporalen Labelvariablenpaare zum Zeitpunkt t, die beide auf das selbe Voxel verweisen (vgl. Gleichung (4.7)). Ausserdem gleicht im Falle von $t = 0$ das LS-AMN der Standard-AMN-Formulierung, da zu Anfang noch kein Vergangenheitskontext existiert und damit die Menge $\Delta^0 = \emptyset$ ist.

4.4.2 Lernen und Inferenz

Aufgrund der bedingten Modellierung kann ein LS-AMN nicht auf die gewohnte Weise gelernt werden. Das Problem ist nämlich, dass zum Lernen der Knoten- und Kantengewichte alle Labelvariablen inferriert werden müssen wohingegen für das Lernen der temporalen Gewichte eine Teilmenge der Labelvariablen als gegeben vorausgesetzt wird. Durch diesen Umstand muss das Lernen also hierarchisch erfolgen. In einem ersten Schritt werden mit einem Lernverfahren der Wahl die Knoten- und Kantengewichte für alle Scans wie gehabt im Verbund gelernt. Dieser Gewichtsvektor wird ab nun referenziert als $\mathbf{w}^* = (\mathbf{w}_n^* \quad \mathbf{w}_e^*)$ und entspricht der gelernten Gewichtung einer Standard-AMN.

Der zweite Schritt setzt nun sukzessiv eine zeitliche Labelmenge nach der anderen als gegeben voraus und lernt auf dieser die zeitliche Gewichtung. Angefangen beim Zeitpunkt $t = 1$ ist also nun Y^0 gegeben und das graphische Modell wird für die Zeitpunkte $t = 0$ und $t = 1$ konstruiert. Es existieren also die drei entscheidenden Mengen Y^0, X^1, Y^1 (vgl. Abbildung 4.4). Dabei wird hier neben der geometrischen Labelnachbarschaft nun auch die zeitliche Labelnachbarschaft in das Modell integriert und die Wahrscheinlichkeitsverteilung im Sinne von Gleichung (4.8) herbeigeführt. Anschließend wird das Lernen mit dem Subgradientenverfahren für eine bestimmte Anzahl an Iterationen durchgeführt, angefangen bei $\mathbf{w}^0 = (\mathbf{w}_n^* \quad \mathbf{w}_e^* \quad \mathbf{0})$.

Nach dem Durchlauf existiert eine Gewichtung $\mathbf{w}^1 = (\mathbf{w}_n^1 \quad \mathbf{w}_e^1 \quad \mathbf{w}_t^1)$. Im Laufe des Verfahrens wurde aber nicht nur die temporale Gewichtung gelernt, sondern ebenfalls die Knoten- und Kantengewichtung für diesen einen Scan verfeinert. Um dem entgegenzuwirken, wird die Gewichtung der Knoten und Kanten auf den anfangs gelernten Stand gebracht und damit gilt für $\mathbf{w}^1 = (\mathbf{w}_n^* \quad \mathbf{w}_e^* \quad \mathbf{w}_t^1)$. Iterativ werden dann also alle Zeitpunkte nacheinander auf diese Weise durchlaufen um schlussendlich die Gewichtung $\mathbf{w}^T = (\mathbf{w}_n^* \quad \mathbf{w}_e^* \quad \mathbf{w}_t^T) = (\mathbf{w}_n^* \quad \mathbf{w}_e^* \quad \mathbf{w}_t^*)$ zu erhalten.

Da das temporale Potential nicht zwei sich gegenseitig beeinflussbare, zu inferrierende Variablen verrechnet, verhält es sich wie ein Knoten- und nicht wie ein Kantenpotential. Damit entfällt, im Gegensatz zum Kantenpotential, die Forderung der Nichtnegativität und eine konvexe Projektion während des Lernens ist nicht notwendig.

Die Inferenz muss aufgrund der selben Annahmen analog durchgeführt werden. Es existiert die Gewichtung $\mathbf{w}^* = (\mathbf{w}_n^* \quad \mathbf{w}_e^* \quad \mathbf{w}_t^*)$ und ausgehend vom Zeitpunkt $t = 0$ wird der erste Scan wie bei der Standardvariante inferriert. Ab dem Zeitpunkt $t = 1$ wird das Modell dann sukzessiv wie oben beschrieben konstruiert und die restlichen $T - 1$ Scans werden in genau $T - 1$ Inferenzphasen klassifiziert.

Zusammenfassung

Dieses Kapitel führte zwei neue Modellvarianten der AMNs zur Ausnutzung zeitlicher bzw. sequentieller Zusatzinformationen ein. Das erste Modell, MS-AMN, nutzt die Informationen über Merkmale in der Zukunft als auch in der Vergangenheit und verrechnet diese miteinander um eindeutigere Merkmale zu erhalten. Dabei wurde die Formulierung der Knotenpotentiale derart angepasst, dass das Lernen und die Inferenz mit den üblichen Vorgehensweisen bewerkstelligt werden kann. Außerdem kann die Annahme auch auf die Vergangenheit beschränkt werden und beispielsweise in einem online-Verfahren genutzt werden. Die zweite Variante, LS-AMN, nutzt das Wissen über das vergangene Labeling und mit Hilfe einer temporalen, log-linearen Potentialfunktion wurde dieser Kontext in das Modell eingegliedert. Das Lernen und die Inferenz muss jedoch aufgrund der Annahme der gegeben Labelings vergangener Scans in mehreren Schritten erfolgen.

Kapitel 5

Evaluation

Dieses Kapitel präsentiert die Untersuchungsergebnisse, welche im Laufe der Arbeit ermittelt wurden. Zuerst wird die Standardformulierung der Associative Markov Networks bezüglich mehrerer Aspekte betrachtet. Es wird untersucht, ob der Nachbarschaftskontext signifikanten Einfluss ausübt und damit die kollektive Klassifikation der isolierten Klassifikation vorzuziehen ist. Darüber hinaus wird evaluiert, wie sich die Lern- und Inferenzverfahren unterscheiden und sich bei verschiedener Parametrisierungen verhalten. Das Finden der optimalen Parameter ist im Bereich des maschinellen Lernens ein oft erörtertes Problem, da sich die Parameter je nach Daten und Werten stark unterscheiden können und damit meist empirisch bestimmt werden müssen. Daher sind die hier ermittelten Ergebnisse für den vorliegenden Datensatz zwar korrekt, aber nicht allgemeingültig und generell übertragbar. Im zweiten Teil der Evaluation wird bezüglich mehrerer Aspekte untersucht, inwieweit die sequentiellen Modelle einen positiven Einfluss ausüben und sich damit von der Standardformulierung absetzen.

Der für die Evaluation genutzte Datensatz bestand aus einer Trainingsmenge und aus einer Testmenge. Die Trainingsmenge beinhaltete sieben 3D-Laserscans, welche in kurzen zeitlichen Abständen hintereinander aufgenommen worden sind. Die Szene war eine Straße mit Parkplätzen auf der linken Seite (bezogen auf die Fahrtrichtung der Trägerplattform) und darauf abgestellten Fahrzeugen. In regelmäßigen Abständen standen Sträucher entlang der Parkflächen und auf der rechten Seite befand sich durchgehend Vegetation neben der Straße. Die Testmenge bestand ebenfalls aus sieben Scans dieser Straßenaufnahmen, jedoch mit zeitlichem Versatz. Daher ähnelten sich die beiden Mengen in ihrer Struktur.

Für den genutzten Trainings-Datensatz kamen als Knotenmerkmale voxelbasierte Up-Spin-Images (siehe Abschnitt 3.2.1) mit einer Auflösung von 10x10 Einträgen zum Einsatz wobei mit allen sieben Scans zusammengefasst 36.372 Voxel (mit 10 cm Kantenlänge) instantiiert und insgesamt 76.182 Kantenmerkmale ermittelt wurden. Die Beschränkung der Merkmalsauflösung für diese Datenmenge ist bedingt durch den großen Speicheraufwand des QPs. Für die Evaluation musste also ein Trade-Off zwischen Merkmalsgröße und Datenmenge gefunden werden, der das Lernen des QPs als auch ein Vergleich beider Methoden erlaubte. Bei einer gewählten Auflösung von 10x10 Einträgen und einer Voxelkantenlänge von 10 cm wurden also 2 m in der Breite und 1 m in der Höhe für die Merkmalsberechnung

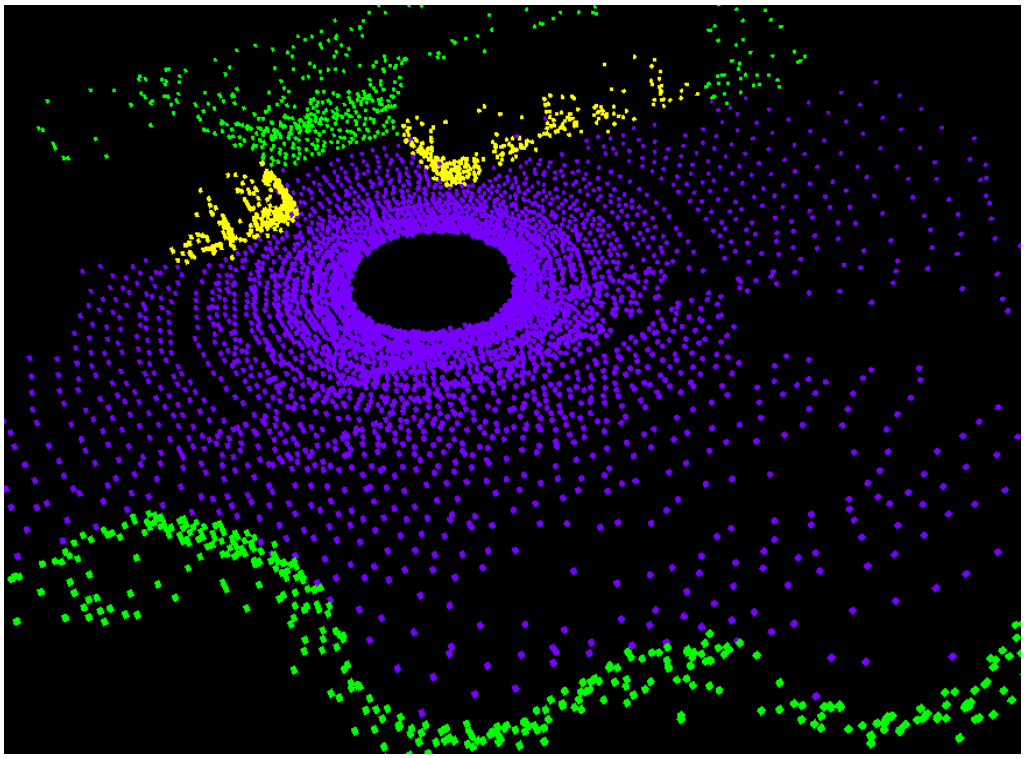
betrachtet. Bei der Nachbarnbestimmung wurden maximal $k = 3$ nächster Nachbarn für ein Voxel gesucht, wobei der Suchradius maximal $r = 1$ m bzw. 10 Voxel betrug. Die Testmenge bestand aus 35.557 Voxeln und es wurden 76.527 Kantenmerkmale berechnet. Die aufsummierte Klassenverteilung sowie eine Aufnahme von einem der sieben Scans der Testmenge sind in Abbildung 5.1 zu ersehen. Der für die Evaluation genutzte Computer war ein Intel Xeon X5550 mit 4 x 2,6 GHz und 12 GB RAM.

5.1 Vergleich für Standard-AMNs

Die vorliegenden Daten wurden in sieben voneinander unabhängige Voxelgitter überführt und anschließend wurden die Knoten- und Kantenmerkmale berechnet. Es wurde ein gemeinsamer Graph konstruiert wobei jedoch keine Kanten zwischen Variablen unterschiedlicher Scans existierten. Bei den Laufzeiten für die Berechnung der Merkmale wurde unterschieden, ob die Voxelgitterstruktur als Matrix-Array (vollständig) oder aber als Listenstruktur (dünn besetzt) abgespeichert wurde. Aus Abbildung 5.2 ist ersichtlich, dass Listenstrukturen die Berechnungen verlangsamten und hauptsächlich zur Evaluation geeignet sind. Sie besitzen nur linearen Speicheraufwand und erlauben damit mehrere simultane Evaluationsinstanzen, benötigen jedoch beim Zugriff auf Elemente der Liste im worst-case linearen zeitlichen Aufwand. Vollständig abgespeicherte Voxelgitter benötigen mehr Speicher (abhängig von der Größe des Scans), sind dafür aber sehr performant in Bezug auf die Berechnung, da Elemente in konstanter Zeit angesprochen werden können. Die Erstellung der Gitterstruktur war dabei in beiden Fällen zeitlich vernachlässigbar.

Die Standardformulierung von Taskar et al. [Taskar et al., 2004] nutzt für das Lernen ein Quadratisches Programm (QP). QPs können mit verschiedenen Methoden minimiert werden, wobei primal-duale Innere-Punkte-Verfahren [Boyd and Vandenberghe, 2004] sich durch gute theoretische Eigenschaften auszeichnen. Diese Verfahren suchen sich einen Pfad zur Nullstelle einer Funktion (ähnlich dem Newton-Verfahren im Mehrdimensionalen), wobei die zu minimierende Funktion hier die Dualitätslücke ist (also die Differenz zwischen primaler und dualer Zielfunktion). Zur Lösung des QPs von Gleichung (2.24) wurde die freie C++-Bibliothek OOQP [Gertz and Wright, 2003] genutzt. OOQP nutzt ein solches Verfahren im Zusammenspiel mit einem Predictor-Corrector-Verfahren nach Mehrotra und Gondzio, welches die Optimierungsrichtung des Pfades verfeinert und damit in der Praxis meist die Konvergenz beschleunigt. OOQP minimiert das Quadratische Programm bis die Dualitätslücke kleiner als 1 ist und hat damit keine vorher festgelegte Anzahl Iterationen wie etwa beim Subgradientenverfahren (siehe Abschnitt 3.3.2).

Als alternative Lernmethode für ein AMN wurde im dritten Kapitel das Subgradientenverfahren vorgestellt und die Ergebnisse für die gegeben Daten werden hier präsentiert. Das Verfahren wurde in C++ implementiert und zeichnete sich durch einen geringen Speicheraufwand aus. Der Nachteil ist die äußerst langsame Konvergenz im Vergleich zur Lösung des QPs mit dem Innere-Punkte-Verfahren mit Mehrotra/Gondzio-Korrektur. Für



(a)

	Absolut	Anteilig (in %)
Fahrzeug	2202	~ 6
Boden	26194	~ 72
Vegetation	7976	~ 22

(b)

	Absolut	Anteilig (in %)
Fahrzeug	2590	~ 7
Boden	26407	~ 72
Vegetation	7560	~ 21

(c)

Abbildung 5.1: Abbildung (a) zeigt einen der sieben Scans der Testmenge. Die Tabelle in (b) zeigt die Klassenverteilung für die Trainingsmenge, also den Anteil der jeweiligen Klasse an den Gesamtdaten. Die Tabelle (c) zeigt den Sachverhalt für die Testmenge. Man kann hier klar erkennen, dass die lilafarbenen Bodenvoxel den größten Teil der Scans ausmachen. Danach folgen die in grüner Farbe gehaltenen Vegetationsvoxel und schlussendlich eine kleine Menge von gelben Fahrzeugvoxeln. Die Tabellen (b) und (c) zeigen, dass beide Mengen ähnlich strukturiert sind.

	Matrix	Liste
Erstellung	0,1 s	$< 0{,}1$ s
Knoten	1,89 s	209,93 s
Kanten	1,23 s	75,22 s

Abbildung 5.2: Zeitlicher Aufwand für die Erstellung der Gitterstruktur als auch für die Berechnung der Merkmale für 36.372 Knoten und 76.182 Kanten in Abhängigkeit der Art der Speicherung.

die Inferenz-Zwischenschritte im Subgradientenverfahren als auch zur Klassifikation der Testdaten wurde die freie C++-Bibliothek libDAI [Mooij, 2010] benutzt, welche verschiedene Algorithmen zur exakten und approximativen Inferenz für diskrete Zufallsvariablen in Graphen bereitstellt. Als Inferenzalgorithmus wurde das in Abschnitt 3.3.1 erläuterte (loopy) Belief Propagation genutzt. Bei der Evaluation wurde neben der Güte der erhaltenen Gewichtung und dem zeitlichen Aufwand auch untersucht, inwiefern die Wahl der Schrittweite als auch die Wahl des Regularisierungsparameters im Subgradientenverfahren einen Einfluss ausübt.

Untersuchung des Einflusses durch die Schrittweite

Die gewählte Schrittweite im Subgradientenverfahren (siehe Abschnitt 3.3.2) hat starken Einfluss auf die Minimierung der Funktion. Dabei sei für die weiteren Erläuterungen auf Abbildung 5.3 verwiesen. Der Regularisierungsparameter war $\lambda = 1$ und es wurden 1.000 Iterationen durchgeführt. Es wurden drei verschiedene Schrittweiten evaluiert, nämlich $\alpha = \frac{1}{k^2}$, $\alpha = \frac{1}{\sqrt{k}}$ sowie $\alpha = \frac{1}{k}$. Der Plot (a) zeigt die zu minimierende logarithmierte Kostenfunktion von Gleichung (3.9) und da das Subgradientenverfahren nicht in jedem Schritt minimiert, wird in Plot (b) das bis zum jeweiligen Zeitpunkt beste Minimum dargestellt. Dazu zeigt Plot (c) die Genauigkeit zur Trainingsmenge mit dem zum Schritt k berechneten $\mathbf{w}^{(k)}$.

Bei einer gewählten Schrittweite von $\alpha = \frac{1}{k^2}$ konnte das Verfahren nicht überzeugen, da die Minimierung relativ schnell zum Erliegen kam. Der Grund dafür war, dass das α bei dieser Wahl mit quadratischer Geschwindigkeit kleiner wird und damit auch die Schritte entlang des negativen Subgradienten immer kleiner wurden. Damit verharrte das Verfahren an einem beliebigen Punkt in der Definitionsmenge und war mit dieser Schrittweite für die vorliegenden Daten nicht zu gebrauchen.

Bei einer gewählten Schrittweite von $\alpha = \frac{1}{k}$ minimierte das Verfahren die Kostenfunktion relativ gut, aber auch hier zeigt sich leicht das Problem der zu kleinen Schritte. Das Verfahren minimiert zwar stetig, die Kurve wird jedoch mit steigender Iterationszahl immer flacher. Dies ist besonders bei Plot (b) zu erkennen.

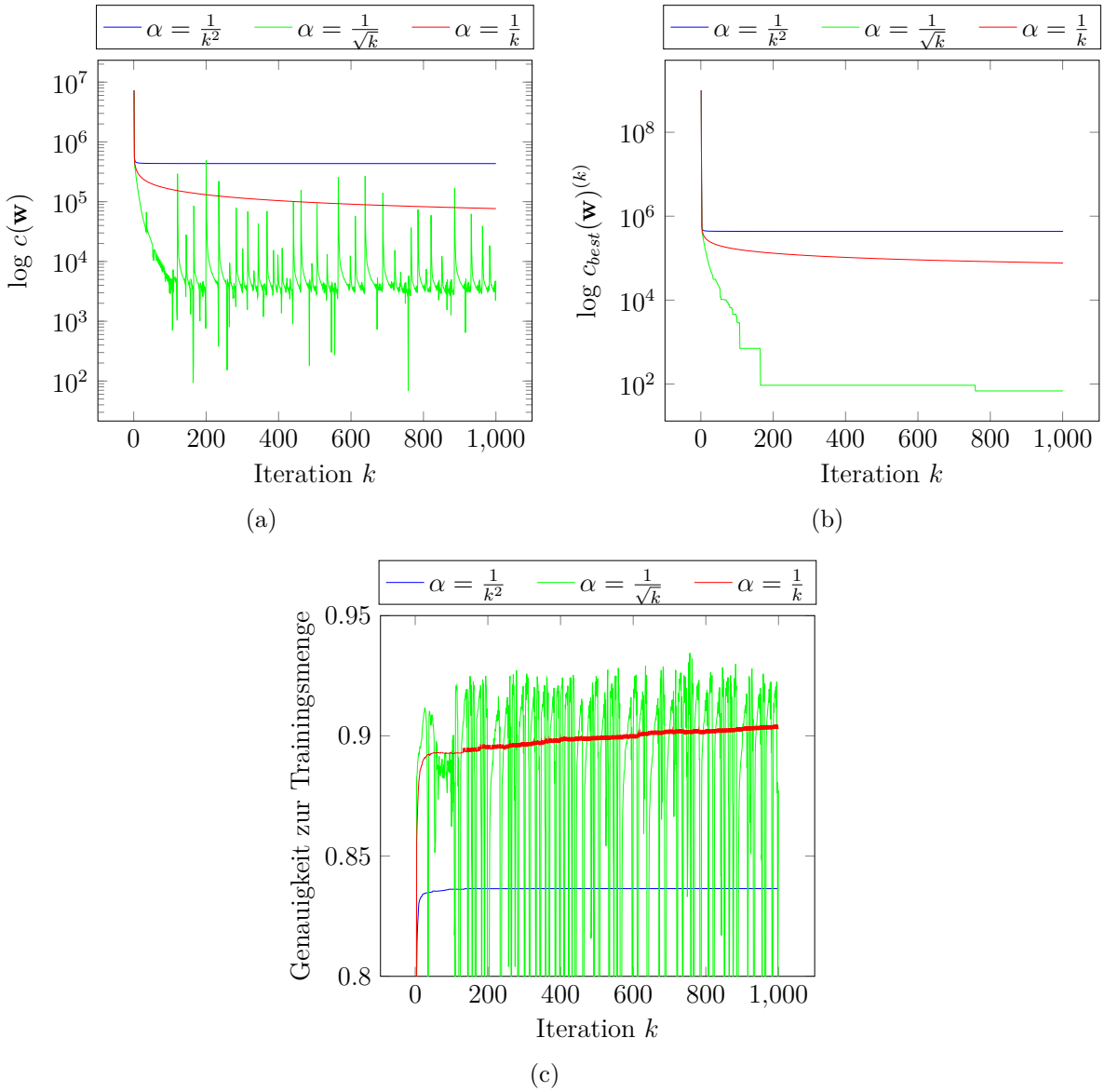


Abbildung 5.3: Konvergenz des Subgradientenverfahrens bei unterschiedlicher Schrittweite. In (a) ist die zu minimierende Kostenfunktion $c(\mathbf{w})$ für den Datensatz zu ersehen. Der Graph in (b) zeigt zur besseren Übersicht $c_{best}(\mathbf{w})^{(k)}$ und damit den zur Iteration k besten Funktionswert. Beide Funktionen sind hier zur logarithmischen Skala abgebildet. Plot (c) zeigt die Genauigkeit zur Trainingsmenge für die in der jeweiligen Iteration erhaltene Gewichtung $\mathbf{w}^{(k)}$.

Bei $\alpha = \frac{1}{\sqrt{k}}$ ergab sich ein interessantes Ergebnis. Durch die sehr großen Schritte, selbst bei einer hohen Iterationszahl, ist das Verfahren bei jeder Iteration sehr weit durch die Definitionsmenge gelaufen. Dies äußert sich in einer starken Oszillation in Plot (a) und (c). Das Verfahren näherte sich in großen Schritten dem Optimum an und wurde durch genau diese großen Schritte wieder gezwungen, weit wegzuspringen. Trotz der vermeintlich 'blinden' Optimierung zeigt sich, dass zielgerichtet minimiert wurde. Beispielsweise sieht man, dass sich eine fallende Regressionskurve in den Plot (a) legen lassen könnte und vor allem in (b) ist offensichtlich, dass das Verfahren bei dieser Schrittweite immer näher an das Optimum kam und sogar das im Vergleich beste Ergebnis erzielte. Insbesondere sei hierbei auf die Spitze bei $k \approx 780$ hingewiesen, welche bei (a) sehr nahe ans Optimum reicht und sich in (c) durch den besten erhaltenen Wert in diesem Verfahren auszeichnet. Eine weitere Beobachtung ist, dass das Oszillieren mit steigender Iterationszahl auch abnahm, besonders ab ca. $k = 900$. Der Grund dafür ist, dass die α-Werte bei hohen Iterationszahlen auch allmählich kleiner wurden und damit das Verfahren weniger stark gesprungen ist.

Das Fazit dieser Untersuchung ist, dass für den genutzten Datensatz die Schrittweiten $\alpha = \frac{1}{k}$ und $\alpha = \frac{1}{\sqrt{k}}$ für die Minimierung geeignet waren. Während Erstere eine eher glatte Optimierung begünstigte, oszillierte bei Letzterer das Verfahren sehr stark. Es ist zu vermuten, dass für den gegebenen Datensatz die Schrittweite $\alpha = \frac{1}{\sqrt{k}}$, bezogen auf den Wertebereich der Merkmals- und Gewichtsvektoren, zu groß war. Dadurch lief das Verfahren bei einer Iteration zu weit durch die Definitionsmenge und verursachte das beobachtete Oszillieren durch sehr starke Funktionswertänderungen.

Untersuchung des Einflusses durch die Regularisierung

In der vorgestellten Formulierung von Gleichung (3.10) hat der Subgradient einen Regularisierungsparameter λ. Dieser Parameter drückt aus, wie stark der Abstand zum optimalen Labeling und die Länge des Gewichtsvektors bei der Minimierung berücksichtigt werden und ist damit als Trade-off zwischen Anpassung und Generalisierung der Modellparameter zu verstehen. Je größer λ ist, desto stärker wird die Länge des Gewichtsvektors bestraft und damit wird die Parametrisierung generalisiert. Der Grund dafür ist, dass einzelne Einträge in $\mathbf{w}$ sich schlechter an Ausreißer und statistisch insignifikante Daten im Datensatz anpassen können, wenn sie bei der Minimierung bei großem λ stärker in Richtung 0 verschoben werden. Damit sind Einträge im Gewichtsvektor nur dann groß, wenn sie durch die Daten gestützt werden. Für die Evaluation wurde zur besseren Visualisierung des Plots (vgl. Abbildung 5.3) eine mit k linear abnehmende Schrittweite gewählt, also $\alpha = \frac{1}{k}$. Während der Auswertung zeigte sich, dass zur besseren Aussagekraft die Zahl der Iterationen auf 1.500 erhöht werden musste. Die nachfolgenden Erläuterungen beziehen sich auf Abbildung 5.4.

Aus Plot (b) lässt sich ersehen, dass mit zunehmender Größe von λ die Länge des Gewichtsvektors bei der Minimierung stärker bestraft wurde. Insgesamt zeigt sich, dass die Minimierung der Gewichtung auch nicht linear erfolgte. Anfänglich wurde die Länge sehr zügig

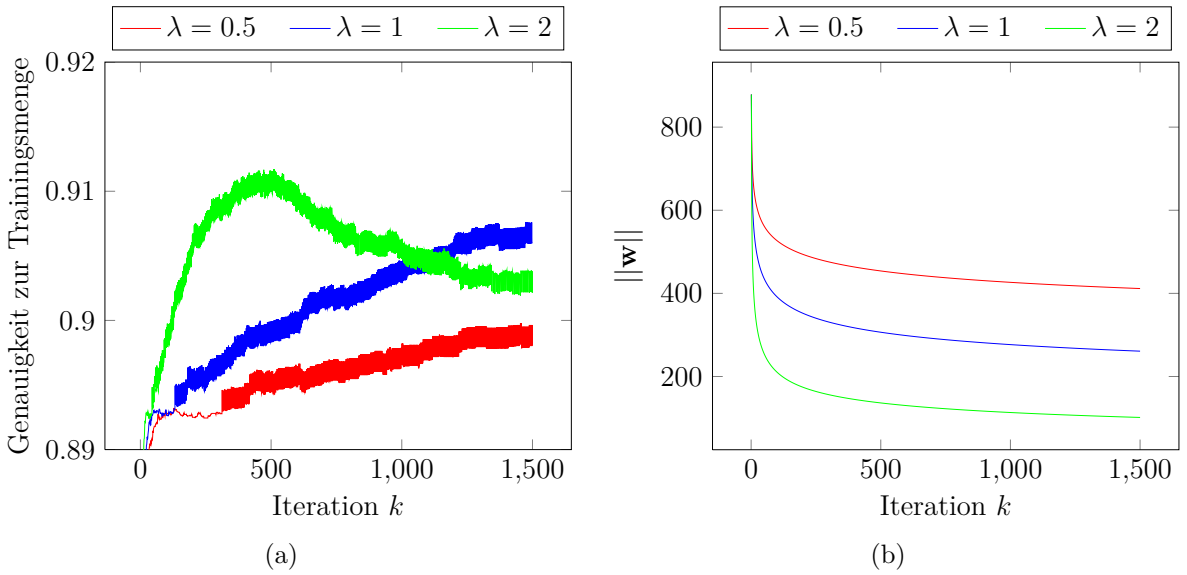


Abbildung 5.4: Einfluss des Regularisierungsparameters λ. Plot (a) zeigt, wie sich die Genauigkeit zur Trainingsmenge im Laufe des Verfahrens je nach gewählter Regularisierung ändert und (b) die Länge des jeweiligen Gewichtsvektors bei jeder Iteration.

minimiert um dann schließlich in weiteren Iterationen nur marginal angepasst zu werden. Dies ist auf die abnehmende Schrittweite und die damit verbundene Abnahme in der Verbesserung zurückzuführen (siehe Abbildung 5.3 für die gewählte Schrittweite $\alpha = \frac{1}{k}$).

Ein differenzierteres Bild ergibt sich bei Betrachtung von (a). Obwohl $\lambda = 2$ die stärkste Regularisierung und damit eigentlich die schlechteste Anpassung an die Trainingsmenge fordert, konnte das Verfahren bei diesem Wert die Gewichtung bei $k \approx 500$ sehr gut an die Trainingsmenge anpassen. Die Vermutung ist, dass bei der Berechnung des Subgradienten (vgl. Gleichung 3.10) der erste Term $\lambda\mathbf{w}$ durch die geringe Länge sehr schnell klein wurde und damit der Subgradient verhältnismäßig stark in die Richtung des Abstandes $\mathbf{X}\mathbf{y}^* - \mathbf{X}\hat{\mathbf{y}}$ zeigte. Durch diese Unverhältnismäßigkeit wurde daher am Anfang des Verfahrens sehr stark der Abstandsterm verringert. Bei fortschreitender Iterationszahl entwickelte sich die Regularisierung jedoch wie erwartet und bei geringerer Länge wurde auch die Genauigkeit zur Trainingsmenge verschlechtert.

Bei Betrachtung von $\lambda = 0,5$ und $\lambda = 1$ zeigt sich auch ein unerwartetes Bild. Zwar dominiert ca. ab $k = 1.000$ die Genauigkeit zur Trainingsmenge von $\lambda = 1$ über $\lambda = 2$, jedoch ist die Genauigkeit von $\lambda = 0,5$ trotz schwacher Regularisierung am schlechtesten. Auch hier ist zu vermuten, dass bei $\lambda = 1$ der Subgradient stärker den Abstand des Labelings betonte aufgrund des kleineren ersten Terms. Zu erwarten ist jedoch, dass die selbe Entwicklung, die sich bei $\lambda = 2$ ab $k \approx 500$ abzeichnete, sich auch bei $\lambda = 1$ ab einer bestimmten Iterationszahl abzeichnet und dann ab einer höheren Anzahl Schritten die Genauigkeit bei $\lambda = 0,5$ die Genauigkeiten der beiden anderen λ-Werten übertrifft.

Ein weitere Betrachtung ist die Klassifikationsgüte der regularisierten Gewichtungen nach 1.500 Iterationen. Für $\lambda = 1$ war die Genauigkeit auf der Trainingsmenge 91,1 % und auf der Testmenge 91 % und damit praktisch gleich. Für $\lambda = 2$ war die Genauigkeit auf der Trainings- und Testmenge bei identischen 90,5 % und für $\lambda = 0,5$ war Trainingsgenauigkeit 90,1 % und die Testgenauigkeit 89,9%. Insgesamt waren die Genauigkeitsunterschiede für ein λ gering und das zeigt, dass beide Mengen ähnlich strukturiert waren und damit die Ground-Truth gut durch die Trainingsmenge repräsentiert wurde.

Das Fazit ist, dass die Regularisierung Einfluss auf die Klassifikation ausüben kann und sich beim Subgradientenverfahren erst ab einer bestimmten Iterationszahl bei der Trainingsgenauigkeit auch bemerkbar macht. Daher muss bei der Wahl einer Regularisierung die Bestimmung einer möglichst optimalen Anzahl Iterationen mit berücksichtigt werden. Bei den vorliegenden Datensätzen konnte jedoch aufgrund der ähnlichen Struktur ein wirklicher Vorteil der Regularisierung nicht gezeigt werden.

5.1.1 Vergleich beider Lernmethoden

Beide Verfahren unterscheiden sich elementar in der Art ihres Aufwands. Das QP erfordert für die Formulierung des Problems mit den gegebenen Daten und einer Merkmalsauflösung der Spin-Images von 10x10 Einträgen (folglich $\mathbf{x}_i \in \mathbb{R}^{100}$) sehr viel Speicherplatz, selbst bei Nutzung dünn-besetzter (sparse) Strukturen. Der große Vorteil ist jedoch die zeitlich effiziente Minimierung des Problems mit dem genutzten primal-dualen Innere-Punkte-Verfahren. Zusammengefasst lässt sich das QP zwar performant minimieren, aber der Umfang der technisch möglichen Datenmenge ist sehr stark durch die Speicherkomplexität eingeschränkt. Im Gegensatz dazu ist das Subgradientenverfahren aufgrund des linearen Speicherbedarfs bestens dazu geeignet, auf sehr großen Datensätzen zu lernen. Der gravierende Nachteil ist aber die sehr lange Minimierungsdauer. Für das Subgradientenverfahren gibt es, abhängig von der gewählten Schrittweite, ausschließlich Garantien für Konvergenz bei $k \to \infty$ [Boyd et al., 2003]. Es gibt auch keine theoretische untere Schranke bezüglich der Konvergenzgeschwindigkeit und ebenso ist unklar, ob numerische Probleme solch eine Konvergenz praktisch überhaupt zulassen.

Die Tabelle von Abbildung 5.5 stellt die ermittelten Daten bezüglich Laufzeit und Speicherbedarf dar. Es ist ersichtlich, dass das QP performant aber sehr speicherlastig minimiert wurde. Das Subgradientenverfahren benötigte nur eine geringe Menge an Speicher, brauchte jedoch sichtlich lange um 1.000 Iterationen durchzuführen. Hierbei wird noch betont, dass die präsentierten Angaben zum Speicherbedarf bezüglich des gesamten Programms zu verstehen sind. Mit dem Subgradientenverfahren hätte also auf der zwölffachen Datenmenge im Vergleich zum QP bei ähnlichem Speicherverbrauch gelernt werden können. Dafür war die Laufzeit des Subgradientenverfahrens um ein Vielfaches höher. Bei den Laufzeiten ist außerdem noch die Angabe der Inferenzzeit angegeben. Das Subgradientenverfahren benötigt zur Berechnung des Subgradienten in jedem Iterationsschritt eine Inferenz mit der momentan gefundenen Gewichtung. Aus der Tabelle ist zu ersehen, dass fast die gesamte

	QP (sparse)	Subgradient
Benötigter Speicher	$\sim$ 7,5 GB	$\sim$ 600 MB
Laufzeit	36 min	$\sim$ 12 Std 50 min
davon Inferenz	N/A	$\sim$ 99,5 %

Abbildung 5.5: Evaluation beider Lernmethoden in Hinblick auf Speicherbedarf und Laufzeit. Zu beachten ist, dass der Speicherbedarf sich auf das gesamte Programm bezieht und damit neben den Voxel- und Merkmalsdaten auch temporäre Strukturen wie die Constraint-Matrix beim QP sowie die Inferenzstrukturen beim Subgradientenverfahren eingerechnet sind. Die Laufzeiten beziehen sich aber ausschließlich auf die Verfahren.

Laufzeit des Verfahrens auf diese Inferenzschritte zurückzuführen ist. Das Minimieren des QPs löst zwar ebenfalls eine MAP-Belegung als Unterproblem, ist aber bezüglich eines Zeitmaßes nicht greifbar, da die Minimierung alle Constraints des QPs zur gleichen Zeit berücksichtigt.

Nun soll untersucht werden, wie gut die beiden erhaltenen Gewichtungen $\mathbf{w}_{QP}$, $\mathbf{w}_{SG}$ die Klassifikation der Testmenge gestalten. Dabei wurde zum Vergleich des Zusatzgewinns durch die Nachbarschaft auch untersucht, wie sich die Klassifikation ohne Nutzung der Kantenpotentiale ändert.

Dargestellt werden die Ergebnisse mit Hilfe von Fehlerraten und Konfusionsmatrizen. Eine Fehlerrate gibt an, wieviel Prozent aller Voxel insgesamt falsch klassifiziert worden sind (1 − Genauigkeit) und eine Konfusionsmatrix lässt sich wie folgt interpretieren: In einer Spalte lässt sich ablesen, wieviele Voxel einer bestimmten Ground-Truth-Klasse einer (anderen) Klasse zugewiesen worden sind. Am Beispiel von Abbildung 5.6 (a) wurden also insgesamt 684 Fahrzeugvoxel richtig erkannt, jedoch in 542 Fällen als Boden und in 1364 Fällen als Vegetation identifiziert. Eine Zeile in der Matrix gibt an, wieviele Voxel aller Klassen einer bestimmten Klasse zugewiesen worden sind. Für die Klasse 'Fahrzeug' gilt, dass 684 Voxel mit 'Fahrzeug'-Label auch wirklich Fahrzeugvoxel waren. Es wurden aber 22 Boden- und 31 Vegetationsvoxel fälschlicherweise als 'Fahrzeug' interpretiert. In der Diagonalen steht für jede Objektklasse die Anzahl der richtig klassifizierten Voxel. Die Recall- und Precisionwerte sind zwei gängige Maße zur Einschätzung der Klassifikationsgüte. Die Precision gibt an, wie hoch der Anteil der richtigen klassifizierten Voxel für eine Klasse 'A' zur Gesamtmenge aller klassifizierten Voxel als Klasse 'A' ist. Der Recall gibt an, wie hoch der Anteil der richtig klassifizierten Voxel für eine Klasse 'A' zur Gesamtmenge der klassifizierten Voxel mit echter Klasse 'A' ist. Mathematisch sind beide Maße definiert als

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

wobei TP (True Positive) die Anzahl der richtig klassifizierten Voxel für Klasse 'A', FP (False Positive) die Anzahl der falsch klassifizierten Voxel als 'A' und FN (False Negative) die Anzahl der falsch klassifizierten Voxel mit echter Klasse 'A' sind.

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	684	22	31	92,8 %
Boden	542	26121	1731	91,9 %
Vegetation	1364	264	5792	78,1 %
Recall	26,4 %	98,9 %	76,7 %	

Fehlerrate: 10,8 %

(a)

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	1015	281	265	65 %
Boden	48	22431	277	98,5 %
Vegetation	1527	3694	7018	57,3 %
Recall	39,1 %	84,9 %	92,8 %	

Fehlerrate: 16,7 %

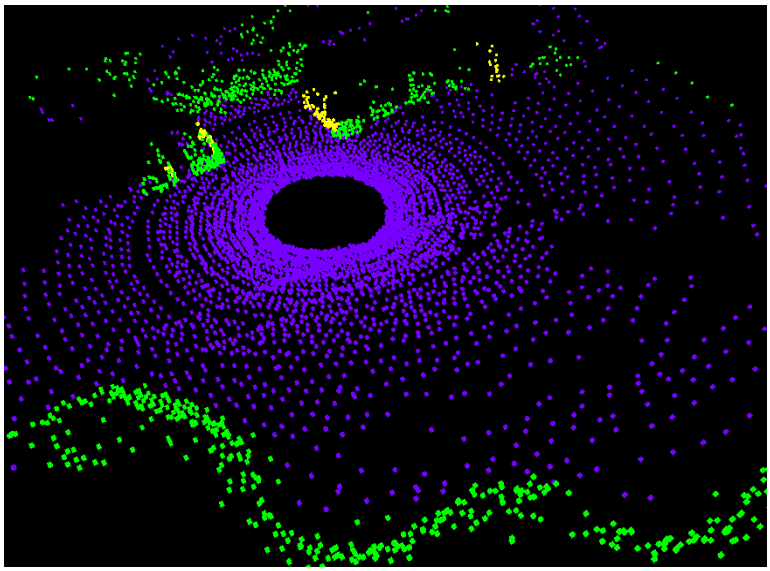
(b)

Abbildung 5.6: Die Konfusionsmatrix mit (a) und ohne (b) Ausnutzung der Kantenmerkmale zur kollektiven Klassifikation für die gelernten Gewichte des QPs.

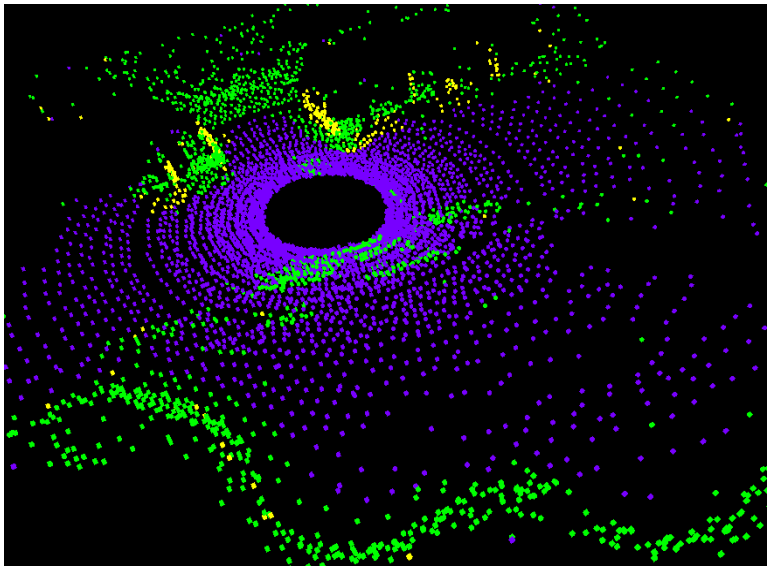
Untersuchung der QP-Gewichtung

In Abbildung 5.6 werden die Konfusionsmatrizen für die Gewichtung $\mathbf{w}_{QP}$, welche über das QP erhalten wurden, dargestellt. Bezüglich der Fehlerraten zur Testmenge lässt sich sofort ersehen, dass die kollektive Klassifikation die Fehlerrate im Vergleich zur isolierten Klassifikation insgesamt um ca. 6 % reduzierte. Bezogen auf den gesamten Datensatz hat sich also die Anzahl der richtig klassifizierten Objekte signifikant erhöht. Das heißt, dass das Minimierungsverfahren auch die Kantengewichte derart trainieren konnte, dass sich der Gesamtfehler verringerte. Auch haben sich die Precision-Werte bis auf die Objektklasse 'Boden' stark verbessert. Bezogen auf (a) ist also im Vergleich zu (b) die Sicherheit gestiegen, mit dem Labeling eines Voxel als 'Fahrzeug' oder 'Vegetation' richtig zu liegen. Die Recall-Werte haben sich jedoch, ausgenommen von 'Boden', teils sehr stark verschlechtert. Dieser Wert ist insbesondere für 'Vegetation' stark gefallen und damit wurden Vegetationsvoxel öfters mit falschen Labels belegt.

Die Argumentation für dieses Ergebnis lässt sich derart gestalten, dass die unausgeglichene Klassenverteilung (Abbildung 5.1) bei der Minimierung des QPs zu einer ungleichen Optimierung führte. Durch die anteilig sehr große Menge der Bodenvoxel wurde das Verfahren gezwungen, für einen möglichst kleinen Zielfunktionswert die Klasse 'Boden' möglichst gut zu prädizieren und die beiden anderen Klassen zu vernachlässigen. Dies zeigt sich besonders dadurch, dass bei den Ergebnissen mit Kantenpotentialen (a) das Kantengewicht für den Boden so groß ist, dass eine fehlerhafte räumliche Glättung durch Kantenpotentiale zu fast 4.000 mehr korrekt klassifizierten Bodenvoxeln führte und damit aber auch zu einer besseren Gesamtklassifikation beitrug. Die höhere Recall- und die niedrigere Precisionrate



(a)



(b)

Abbildung 5.7: Visualisierung der Scan-Klassifikation aus Abbildung 5.1 mit QP-Gewichtung. In (a) wurden Kantenpotentiale genutzt, in (b) nicht. Der Vorteil der räumlichen Glättung ist deutlich zu sehen. Fahrzeugvoxel wurden jedoch sehr schlecht erkannt und meist als Vegetation klassifiziert.

von 'Boden' im Vergleich von (a) zu (b) untermauert dieses Argument. Eine Visualisierung beider Resultate ist in Abbildung 5.7 zu ersehen. Insgesamt erzeugte die kollektive Klassifikation ein räumlich glatteres Ergebnis. Fahrzeugvoxel wurden jedoch sehr schlecht erkannt und insgesamt fast durchgehend als Vegetation klassifiziert (zum Vergleich: die Ground Truth von Abbildung 5.1). Damit hat das Verfahren die Separation von 'Fahrzeug' und 'Vegetation' nicht gut bewerkstelligt.

Der Grund für die schlechte Separation ist, dass die Merkmale beider Klassen sich desöfteren in ihrer Struktur ähnelten. Dafür sei für die folgende Erläuterung auf Abbildung 5.8 verwiesen. Nicht nur die Spin-Image-Merkmale lassen sich gut visualisieren, sondern ebenfalls auch die dazugehörigen Gewichtungen. In der Abbildung erkennt man drei große Quadrate, wobei ein Quadrat die Gewichtung der jeweiligen Klasse darstellt (v.l.n.r.: 'Fahrzeug', 'Boden', 'Vegetation'). Die Einträge haben dabei eine Färbung, die je nach Vorzeichen des Eintrags grau (positiv) oder rot (negativ) ist und je stärker die Färbung ist, umso höher ist auch der Betrag des jeweiligen Eintrags. Ein Eintrag korrespondiert dabei genau mit dem dazugehörigen Bin des Spin-Image-Merkmals. Während des Lernvorgangs wurde ermittelt, inwiefern ein bestimmter Bin mit korrespondierendem Eintrag wichtig für die Erkennung einer bestimmten Klasse ist. Das mittlere Quadrat zeigt die Gewichtung für die Klasse 'Boden' und es ist sofort zu ersehen, dass Bins in der Horizontalen sehr stark aussagten, ob ein Voxel als Boden klassifiziert werden konnte oder nicht. Alle sonstigen Bins wurden negativ bewertet und die korrespondierenden Einträge sind daher rot gefärbt. In den Knotenpotentialfunktionen des AMN werden die Bins mit diesen Einträgen multipliziert und je größer das Produkt, desto höher das Potential für diese bestimmte Klasse. Auch beim Gewicht für 'Fahrzeug' sieht man, dass besonders auf der linken Seite das Lernverfahren die Einträge positiv bewertete und Einträge in der Horizontalen bestrafte. Hier konnte also eine gute Merkmalsunterscheidung zwischen 'Fahrzeug' und 'Boden' erreicht werden. Nun zeigt sich aber, dass sich große Teile der Gewichtung für 'Fahrzeug' und 'Vegetation', besonders oben und unten, überschneiden. Bei diesen Bins ähnelten sich die Merkmale beider Klassen sehr, und das Verfahren konnte daher keine gute Unterscheidung in diesen Merkmalsbereichen erzielen.

Untersuchung der Subgradienten-Gewichtung

Für die Evaluation der Gewichtung $\mathbf{w}_{SG}$ wurden die Parameter auf $\lambda = 1$ und $\alpha = \frac{1}{k}$ gesetzt und 1.000 Iterationen durchgeführt. Die Ergebnisse lassen sich aus Abbildung 5.9 ablesen. Es zeigt sich, dass die gefundene Gewichtung allgemein zu einer leichten Reduktion der Fehlerrate im Vergleich zur QP-Gewichtung führte. Es lässt sich auch beobachten, dass insbesondere die Anzahl der als 'Vegetation' klassifizierten Voxel zunahm und gleichzeitig die Menge der klassifizierten 'Fahrzeug'-Voxel abnahm. Die Tabelle (b) zeigt, dass die Precision für 'Fahrzeug' sich dabei deutlich erhöhte und 'Fahrzeug'-Voxel fast ausschliesslich nur dann identifiziert wurden, wenn sie diese Klasse auch wirklich besaßen. Bei Ausnutzung der Kantenpotentiale konnte darüber hinaus das größte Problem der QP-Gewichtung, nämlich

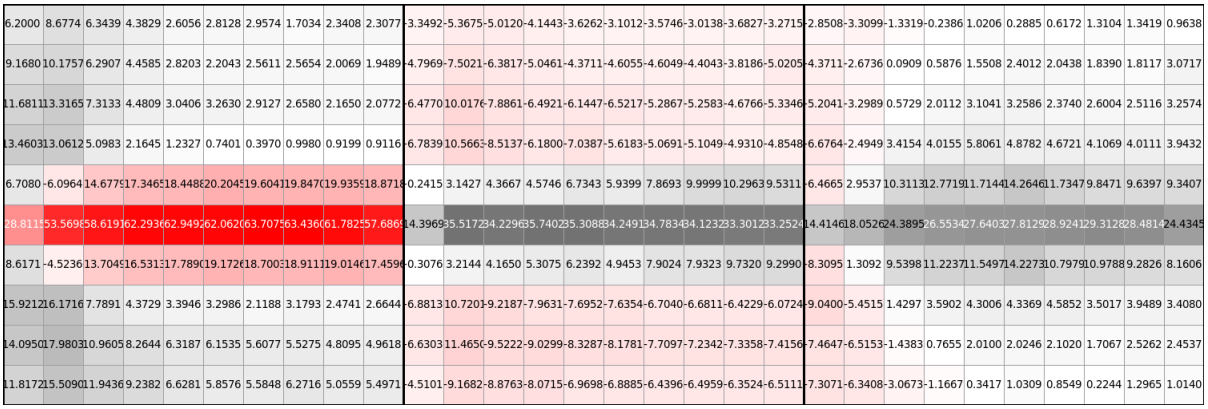


Abbildung 5.8: Visualisierung der erhaltenen QP-Gewichtung. Grau gefärbte Einträge sind positiv, rot gefärbte Einträge sind negativ. Je stärker die Farbe eines Eintrages ist, umso größer ist sein Betrag. Von links nach rechts ist dabei die Gewichtung für 'Fahrzeug', 'Boden' und 'Vegetation' zu sehen.

die signifikante Menge der fälschlich als 'Boden' klassifizierten Voxel, abgeschwächt werden. Ein Großteil dieser Voxel wurden mit der Subgradienten-Gewichtung als 'Vegetation' klassifiziert und aufgrund des höheren Recalls für diese Klasse im Vergleich zu Abbildung 5.6 und ähnlichem 'Boden'-Recall, ist dies der Hauptgrund für die geringere Fehlerrate. Das Subgradientenverfahren konnte also 'Fahrzeug' von 'Vegetation' marginal besser separieren und damit auch die Kantengewichtung ein wenig angemessener bestimmen.

Fazit beider Verfahren

Insgesamt waren beide Gewichtungen, trotz unterschiedlicher Berechnungsweise, ähnlich in ihrer Klassifikationsgüte. Darüber hinaus war der Trainingsfehler in beiden Fällen auch mit dem des Testfehlers vergleichbar. Das führt zu der Überlegung, dass nicht die Verfahren eine bessere Klassifikation verhinderten, sondern die Merkmale, da sie in ihrer Aussagekraft beschränkt waren und damit beide Verfahren limitierten. Um diese Überlegung zu stützen, wurde das Subgradientenverfahren mit den selben Parametern auf Spin-Images mit einer Auflösung von 15x15 Einträgen angewandt. Die Fehlerrate für die Trainings- und die Testmenge reduzierte sich auf $\sim 5\%$ mit Nutzung von Kantenpotentialen und ist damit ein möglicher Hinweis auf diese Annahme. Die Merkmale waren hier nun diskriminativer und das Verfahren konnte besser optimieren.

Somit sind die Minimierung des QPs als auch das Subgradientenverfahren probate Mittel zur Gewichtsberechnung. Die Vorteile des Subgradientenverfahrens sind jedoch die feine Justierung, die die Parameter des Verfahrens erlauben sowie der geringe Speicheraufwand. Je nach Anwendungsgebiet kann die Schrittweite oder aber auch die Regularisierung unterschiedlich gewählt werden um eine gegebene Probleminstanz besser zu lösen. Die lange Lernzeit ist bei Klassifikationsaufgaben meist von untergeordneter Bedeutung, da das

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	429	7	25	93 %
Boden	438	25692	538	96,3 %
Vegetation	1723	708	6997	74,2 %
Recall	16,6 %	97,2 %	88,6 %	

Fehlerrate: 9,4 %

(a)

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	461	10	62	86,5 %
Boden	373	23051	467	96,4 %
Vegetation	1756	3346	7031	57,9 %
Recall	17,8 %	87,3 %	93 %	

Fehlerrate: 16,4 %

(b)

Abbildung 5.9: Die Konfusionsmatrix mit (a) und ohne (b) Ausnutzung der Kantenmerkmale zur kollektiven Klassifikation für die gelernten Gewichte durch das Subgradientenverfahren.

Lernen einmalig und im Vorfeld stattfindet. Ein hoher Speicherverbrauch ist hingegen problematisch, da es das Lernen von großen Datensätzen schwer oder unmöglich macht und damit eine optimale Gewichtungsbestimmung verhindern kann.

5.2 Vergleich der sequentiellen AMN-Formulierungen

Im folgenden soll untersucht werden, inwiefern die beiden sequentiellen AMN-Formulierungen des vierten Kapitels einen Einfluss auf die Klassifikation ausüben. Für diesen Evaluationsteil wurde ausschließlich das Subgradientenverfahren genutzt und aufgrund der Untersuchungsergebnisse des letzten Abschnitts wurde für das bestmögliche Resultat die Schrittweite auf $\alpha = \frac{1}{\sqrt{k}}$ und $\lambda = 1$ gesetzt. Die Anzahl der Iterationen wurde auf 1.000 begrenzt. Die Ergebnisse für die Standardformulierung für diese Parameter sind in Abbildung 5.10 zu sehen.

5.2.1 MS-AMN

Das Merkmal-sequentielle AMN-Modell von Abschnitt 4.3. sucht nach Merkmalsinformationen im vorherigen und zukünftigen Scan an der selben räumlichen Position, um die Merkmale miteinander zu verrechnen und sie dadurch, vor allem in dünn-besetzten Regionen, diskriminativer zu gestalten. Die Idee ist hierbei, strukturlose Informationen, wie

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	1531	314	385	68,7 %
Boden	358	25559	606	96,3 %
Vegetation	701	534	6569	84,1 %
Recall	59,1 %	96,8 %	86,9 %	

Fehlerrate: 7,9 %

(a)

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	244	6	13	92,8 %
Boden	821	25401	2827	87,4 %
Vegetation	1525	1000	4720	69,3 %
Recall	9,4 %	96,2 %	62,4 %	

Fehlerrate: 16,9%

(b)

Abbildung 5.10: Die Konfusionsmatrix mit (a) und ohne (b) Ausnutzung der Kantenmerkmale für die Standard-AMN mit Schrittweite $\alpha = \frac{1}{\sqrt{k}}$.

bspw. Rauschen, aus den Daten zu entfernen und klassentypische Bereiche im Merkmal zu verstärken. In der Trainingsmenge wurden beim Lernen insgesamt 1.627 zeitliche Nachbarn gefunden und in der Testmenge waren es 1.329 zeitliche Paare. Das Suchen nach zeitlichen Nachbarn sowie die Verrechnung der Merkmale betrug dabei weniger als ein Zehntel einer Sekunde. Die Lernzeit mit dem Subgradientenverfahren für das MS-AMN war mit der Lernzeit der Standardformulierung vergleichbar; der Unterschied betrug wenige Sekunden. Die Abbildung 5.11 zeigt beide Konfusionsmatrizen.

Das erste und wichtigste Kriterium ist die Reduktion des Fehlers um 0,6 % mit Kantenpotentialen und eine Reduktion von 1% ohne Nutzung der Kantenpotentiale. Insbesondere ist hier auf die Fehlerrate ohne Kantenpotentiale hinzuweisen. Werden keine Kantenpotentiale genutzt, modelliert das Netzwerk ausschließlich die Merkmalsinformationen eines Voxel zur (isolierten) Klassifikation. Um eine Reduktion von 1% zu erreichen, mussten also die Knotenmerkmale, respektive die Up-Spin-Images, diskriminativer gewesen sein und erlaubten damit eine bessere Separation. Die räumlichen Überlappungen zweier Scans finden oft in den spärlich besetzten Randregionen statt und in diesen Regionen werden auch meist keine räumlichen Nachbarschaften gefunden. Die Voxel in diesen Regionen müssen also fast immer isoliert klassifiziert werden und die zeitliche Merkmalsverrechnung verbesserte diesen Umstand bzw. schwächte die Problematik fehlender räumlicher Nachbarn ab.

Um das Modell weiter zu beleuchten, wurde eine zusätzliche Testmenge evaluiert. Diese Testmenge bestand aus 44.044 Knoten und 92.452 Kanten, wobei der zeitliche Abstand dieser Scans zueinander im Vergleich zur alten Testmenge verringert wurde. Da die Menge der zeitlichen Paare stark von der Scandichte, der Voxelgröße als auch vom zeitlichen und

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	1478	165	256	77,8 %
Boden	361	25697	597	96,4 %
Vegetation	751	545	6707	83,8 %
Recall	57,1 %	97,3 %	88,7 %	

Fehlerrate: 7,3 %

(a)

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	274	9	13	92,6 %
Boden	820	25838	2905	87,4 %
Vegetation	1496	560	4642	69,3 %
Recall	10,6 %	97,8 %	61,4 %	

Fehlerrate: 15,9 %

(b)

Abbildung 5.11: Die Konfusionsmatrix mit (a) und ohne (b) Ausnutzung der Kantenmerkmale für das MS-AMN.

räumlichen Versatz der Aufnahmen abhängig ist, sollte hier also eine Zunahme an zeitlichen Paaren zu erwarten sein. Insgesamt konnten dadurch 8.207 zeitliche Paare gefunden, und damit das Siebenfache zur alten zeitlichen Paarmenge bei nur einem Drittel mehr Knoten. Damit zeigt sich, dass bei einem geringeren zeitlichen und räumlichen Abstand die Zahl der zeitlichen Paare zunimmt. Auch hier zeigte sich klar der Vorteil des sequentiellen Modells. Die Fehlerraten waren mit Nutzung von Kantenpotentialen 8,7 % (AMN) und 8 % (MS-AMN). Insbesondere war hier wieder der Vorteil bei der isolierten Klassifikation zu ersehen, denn die Fehlerrate war mit 17,5 % für das AMN signifikant größer als 16,04 % für das MS-AMN.

Damit zeigt sich der Vorteil des MS-AMN gegenüber der Standardformulierung aufgrund aussagekräftigerer Merkmale. Durch die diskriminativeren Merkmale konnten dann auch folgerichtig die Kantenpotentiale die Klassifikation weiter verbessern. Exemplarisch wird hier noch eine Visualisierung eines Merkmals ohne und mit zeitlicher Verrechnung in Abbildung 5.12 gegeben. Das Bodenvoxel wurde nach Verrechnung eines Nachbarns marginal aussagekräftiger, da in der horizontalen Linie des Merkmals jetzt einzelne Bins einen größeren Wert aufweisen und in der Visualisierung damit dunkler geworden sind. Es muss hierbei noch berücksichtigt werden, dass nur insgesamt 1.329 von 35.557 bzw. 8.207 von 44.044 Voxeln eine zeitliche Merkmalsverrechnung erfuhren und damit der Einfluss auf die Klassifikation durch das MS-AMN nicht sehr groß sein konnte. Dementsprechend ist eine Verbesserung von 1 % bzw. 1,46 % in der Genauigkeit ein bedeutender Unterschied zwischen beiden Modellen.

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0364	0.1273	0.2183	0.1091	0.0546	0.1091	0.2001
0.2364	0.4547	0.2183	0.3456	0.2183	0.2183	0.2546	0.2910	0.4001	0.1819
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0182	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

(a)

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0355	0.1421	0.2309	0.1065	0.0355	0.1065	0.2309
0.2309	0.4262	0.1953	0.3374	0.2131	0.2309	0.2486	0.3196	0.4084	0.1776
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

(b)

Abbildung 5.12: Exemplarisches Merkmal eines Bodenvoxel vor (a) und nach (b) Verrechnung eines zeitlichen Nachbarns. Man kann erkennen, dass das Merkmal in der Horizontalen höherwertigere Bins erhalten hat, da nun einzelne Bins ein weniger dunkler geworden sind. Damit wird das Merkmal insgesamt stärker in diesem Bereich betont.

5.2.2 LS-AMN

Das Label-sequentielle AMN von Kapitel 4.4 nutzt das Wissen über das Labeling aus dem vorherigen Scan, um über ein temporales Potential Einfluss auf die Klassifikation auszuüben. Die Idee ist, das Labeling eines Voxel über die Zeit mitzutragen und damit den Belief für das jeweilige Label zu stärken. Damit kann der Klassifikator über die Zusatzinformationen ein Labeling aufgrund der Erfahrung des letzten Scans favorisieren oder, falls die Gewichtung für die Klasse negativ ist, benachteiligen.

In der Trainingsmenge wurden 520 zeitliche Labelpaare identifiziert und in der ersten Testmenge wurden 323 und in der zweiten Testmenge 2.486 zeitliche Paare gefunden. Damit war die Größe der Überlappung in Bezug zu der Gesamtvoxelanzahl sehr klein. Das Lernen erforderte einen zeitlichen Zusatzaufwand, denn nach der erhaltenen Gewichtung des Standard-AMNs musste sukzessiv ein Graph für den jeweiligen Zeitschritt konstruiert und mit dem Subgradientenverfahren gelernt werden. Die Anzahl der Iterationen wurde dabei für einen Scan auf 100 gesetzt und insgesamt dauerte das Lernverfahren ca. 5 min länger. In jeder Iteration musste für die Berechnung des Subgradienten aus Gleichung 3.10 das $\mathbf{y}^*$ durch eine Inferenz ermittelt werden. Da die Gewichtung für die Knoten und Kanten schon bestimmt worden ist, konnten die vielen Inferenzzwischenschritte effizient gelöst werden. Der Grund dafür war, dass die loopy Belief Propagation aufgrund der schon guten Knoten- und Kantengewichtung jedes Mal schnell konvergierte. Dadurch ergaben sich nur wenige Änderungen im Graphen und große Teile des Netzes behielten ihr Labeling.

Die temporalen Gewichte für die drei Klassen waren gerundet (v.l.n.r.: 'Fahrzeug', 'Boden', 'Vegetation') -2.17, 0.06 sowie 5.85. Somit hat das Verfahren gelernt, dass ein Fahrzeugvo-

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	1074	299	1058	44,2 %
Boden	216	29248	1080	95,8 %
Vegetation	432	731	9906	89,5 %
Recall	62,4 %	96,6 %	82,6 %	

Fehlerrate: 8,7 %

(a)

	Fahrzeug	Boden	Vegetation	Precision
Fahrzeug	1074	297	1014	45 %
Boden	216	29246	1083	95,7 %
Vegetation	432	735	9947	89,5 %
Recall	62,4 %	96,6 %	82,6 %	

Fehlerrate: 8,6 %

(b)

Abbildung 5.13: Die Konfusionsmatrix für die zweite Testmenge für das AMN (a) und das LS-AMN (b).

xel negativ mit seiner Vergangenheit korrelierte und 'Boden' temporal kaum identifiziert werden konnte. Bei 'Vegetation' konnten aber über die Vergangenheit während des Lernens positive Rückschlüsse auf das aktuelle Labeling gezogen werden.

Das Ergebnis für die erste Testmenge war nicht ermutigend. Die Fehlerrate hat sich bei Ausnutzung der temporalen Labelpaare im Vergleich zur Standardformulierung um 0,02 % vergrößert. Hier haben also die temporalen Kanten, aufgrund eines nicht optimalen Labelings des vorherigen Scans, einen negativen Einfluss ausgeübt. Bei der zweiten Testmenge mit mehr temporalen Labelpaaren war das Ergebnis ähnlich. Die Konfusionsmatrizen in Abbildung 5.13 zeigen die Ergebnisse für die zweite Testmenge bezüglich der normalen AMN-Formulierung und dem LS-AMN. Es zeigte sich, dass sich die Fehlerrate nur um insignifikante 0,1 % für das LS-AMN verringert hatte und auch die Recall- und Precisionwerte ebenfalls so gut wie identisch waren.

Insgesamt ließ sich hier kein positiver Mehrwert durch dieses sequentielle Modell finden. Dies könnte entweder an einer zu geringen Anzahl an temporalen Paaren, an einer zu kleinen Trainingsmenge oder aber an nicht aussagekräftigen Merkmalen liegen. Das Problem an diesem Modell ist nämlich, dass auch falsche Labelings aus der Vergangenheit die Klassifikation beeinflussen. Wenn also die Trainingsmenge nicht repräsentativ genug ist oder aber auch die Merkmale nicht diskriminativ genug sind, so macht das LS-AMN Folgefehler. Dies suggeriert, dass dieses Modell methodisch vorteilhaft sein kann, jedoch für eine im Vergleich bessere Klassifikation auf guten bzw. zuverlässigen Informationen arbeiten muss.

5.3 Fazit

Zu Anfang wurde untersucht, wie gut die Minimierung des QPs als auch das Subgradientenverfahren die Gewichtung des Modells lernen. Während das QP mit hohem Speicheraufwand performant minimierte, lernte das Subgradientenverfahren recht langsam aber dafür nicht speicherintensiv. Beide erhaltenen Gewichtungen waren dabei in ihrer Klassifikationsgüte vergleichbar. Das Subgradientenverfahren bietet jedoch eine höhere Flexibilität in Bezug auf das Lernen, da es mehrere Möglichkeiten der Anpassung gibt. Die Schrittweite hatte großen Einfluss auf die Minimierung wohingegen die Regularisierung keinen wirklichen Nutzen für die vorliegenden Daten bot. Als Empfehlung kann man hervorbringen, dass das QP besonders für kleine Datensätze sowie bei gering dimensionierten Merkmalen geeignet ist. Das Subgradientenverfahren erlaubt hingegen das Lernen auf sehr großen Datensätzen. Insgesamt war jedoch der limitierende Faktor bei der Klassifikation die nicht sehr aussagekräftigen Merkmale. Es konnte durch Nutzung größerer Merkmale gezeigt werden, dass die Genauigkeit für die Trainings- und auch die Testmenge signifikant anstieg.

Bei der Evaluation der sequentiellen Modelle zeigte sich ein gemischtes Bild. Das Merkmalsequentielle AMN konnte in der Tat eine empirische Verbesserung der Klassifikation für beide Testmengen erbringen. Die Verrechnung von Merkmalen über die Zeit ist daher ein geeignetes Mittel, die Aussagekraft von Spin-Images und damit die Genauigkeit zu erhöhen. Ausserdem erforderte die Suche nach zeitlichen Nachbarn sowie deren Verrechnung keinen spürbaren zeitlichen Zusatzaufwand. Die Untersuchung des Label-sequentiellen AMNs war jedoch weniger erfolgreich. Es zeigte sich, dass das LS-AMN keinen Vorteil in Bezug auf die Klassifikation bringt, sofern die vergangenen Klassifikationen zu fehlerbehaftet sind. Die Differenzen der Fehlerraten zum Standard-AMN waren vernachlässigbar und damit war das LS-AMN, besonders in Anbetracht des zusätzlichen Aufwandes beim Lernen und der Inferenz, hier die schlechtere Wahl.

Kapitel 6

Zusammenfassung und Ausblick

Diese Arbeit präsentierte das Problem der robusten Klassifikation von sequentiellen 3D-Laserscandaten unter Nutzung von Conditional Random Fields. Dafür wurde zuerst die spezielle Variante der Associative Markov Networks eingeführt und die Standardformulierungen für das Lernen als Quadratisches Programm und das Lernen als Lineares Programm erläutert. Zusätzlich wurden Belief Propagation und das Subgradientenverfahren als alternative Methoden für das Lernen und die Inferenz vorgestellt.

Um dem Problem der großen Datenmengen entgegenzuwirken, wurden die Scandaten in eine abstraktere, voxelbasierte Gitterstruktur überführt. Die Idee dahinter war die informationsbewahrende Reduktion der Datenmenge, wobei redundante Scaninformationen verschwanden und spärlich abgetastete Regionen erhalten geblieben sind. Das Knotenmerkmal wurde als eine an die Voxelstruktur angepasste und nach oben ausgerichtete Version eines Spin-Image und das Kantenmerkmal war das Skalarprodukt zweier Knotenmerkmale.

Dem folgte die Formulierung zweier neue Modelle, aufbauend auf dem Associative Markov Network, um zeitlich-sequentielle Informationen in das Modell zu integrieren und die Klassifikation zu verbessern. Dabei wurde zum einen die Idee verfolgt, Merkmale über die Zeit zu verrechnen um sie eindeutiger in Bezug zu ihrer Objektklasse zu setzen und zum anderem, die Klassifikation bzw. das Labeling des vorherigen Scans auszunutzen um Zusatzinformationen für die Klassifikation des aktuellen Scans zur Verfügung zu stellen.

In der Evaluation wurden dann beide Lernverfahren in Bezug auf Geschwindigkeit, Speicherverbrauch als auch Klassifikationsgüte der erhaltenen Gewichtung miteinander verglichen. Während die Minimierung des QPs performant aber speicherlastig durchgeführt wurde, konnte das Subgradientenverfahren zwar langsam aber dafür mit geringem Speicherverbrauch minimieren. Darüber hinaus wurde untersucht, inwiefern die Wahl unterschiedlicher Werte für die Schrittweite als auch der Regularisierung Einfluss auf das Subgradientenverfahren ausübt. Die Schrittweite hatte wesentlichen Einfluss auf die Minimierung wohingegen verschiedene Regularisierungsparameter aufgrund der zu starken Ähnlichkeit zwischen Trainings- und Testmenge zu keinem signifikanten Unterschied führten.

Bei der Untersuchung der sequentiellen Modelle konnte das Merkmal-sequentielle Modell die Klassifikation merklich verbessern. Durch die Verrechnung der Merkmale zeitlicher

Nachbarn wurden diese diskriminativer gestaltet und damit wurde eine bessere Klassifikationsrate erreicht. Dies zeigte sich vor allem bei Ausschluss der Kantenpotentiale. Das Label-sequentielle Modell konnte jedoch nicht überzeugen, da es trotz eines Zusatzaufwands bezüglich der Klassifikationsgenauigkeit fast identisch zur Standardformulierung war. Die möglichen Ursachen könnten dabei die geringe Anzahl der Überlappungen, ein schlechtes Labeling der vergangenen Scans als auch aussageschwache Merkmale gewesen sein.

Ausblick

Im Rahmen der Bachelorarbeit musste der Umfang der Untersuchungen stark eingeschränkt werden und daher konnten nicht sehr viele Aspekte betrachtet werden.

Die Knotenmerkmale hätten in mehreren Größen evaluiert werden können und insgesamt hätten auch mehr Klassen und weitere Merkmale in die Betrachtung hinzu gezogen werden können. Beispielsweise konnten während der Arbeit die Informationen über Remission und Distanz nicht betrachtet werden.

Die Lernverfahren wurden sowohl durch den Speicherverbrauch als auch aufgrund der benötigten Zeit limitiert. Interessant wäre eine Untersuchung des QPs für größere Datensätze und Merkmale gewesen. Das Subgradientenverfahren war für die Evaluation äußerst zeitintensiv. Hier hätten mehrere Schrittweiten und mehr Iterationen weiteren Aufschluss über die Minimierung und den Klassifikationseinfluss geben können. Ausserdem wäre interessant gewesen, ob sich die Regularisierung auf anders gearteten Trainings- und Testmengen positiv auswirkt.

Bezüglich der sequentiellen Modelle hätte weiter untersucht werden können, wie stark der Einfluss verschiedener Überlappungsgrößen auf die Klassifikation ist. Die Anzahl der zeitlichen Paare ist direkt abhängig von der Dichte der Scans, der Voxelgitterauflösung sowie dem räumlichen und zeitlichen Versatz der Aufnahmen. Hier hätte eine gründlichere Untersuchung unter Umständen Gesetzmäßigkeiten aufzeigen können.

Eine weitere Idee wäre ein hybrides Modell, dass vergangene Merkmale und vergangene Labels gleichzeitig ausnutzt um die Klassifikation zu verbessern. Ferner wäre auch zu überlegen, ob nicht nur direkte Voxelnachbarn in der Zeit betrachtet werden sollten, sondern auch Voxel die im letzten Scan um den zeitlichen Nachbarn herum positioniert waren. Diese könnten dann ebenfalls, mit bspw. einem von der Distanz abhängigen Abschwächungsterm, in die Merkmalsverrechnung einfließen.

Um Rauschen und allgemein Fehler im zeitlichen Kontext zu reduzieren, wurden ausschließlich sequentielle Datenströme betrachtet. Die Frage ist, ob nicht auch über mehrere Zeitpunkte hinweg, dieser Kontext hätte eingebaut werden sollen. Hier hätte man einen Trade-Off zwischen zeitlichem Rauschfehler und zeitlichem Zusatzgewinn für die Klassifikation finden müssen.

Literaturverzeichnis

[Anguelov et al., 2005] Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., and Heitz, G. (2005). Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 169–176.

[Behley and Steinhage, 2009] Behley, J. and Steinhage, V. (2009). Generation of 3D City Models using Domain-Specific Information Fusion. In *Proc. of the International Conference on Computer Vision Systems*, pages 164–173.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Blodow et al., 2009] Blodow, N., Rusu, R. B., Marton, Z. C., and Beetz, M. (2009). Partial view modeling and validation in 3D laser scans for grasping. In *Proc. of the IEEE/RAS International Conference on Humanoid Robots*, pages 459–464.

[Boyd and Mutapcic, 2007] Boyd, S. and Mutapcic, A. (2007). *Stochastic Subgradient Methods*. Lecture Notes to Convex Optimization II, Stanford University.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[Boyd et al., 2003] Boyd, S., Xiao, L., and Mutapcic, A. (2003). *Subgradient Methods*. Lecture Notes to Convex Optimization II, Stanford University.

[Boykov et al., 2001] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.

[Brenner, 2005] Brenner, C. (2005). Building reconstruction from images and laser scanning. *International Journal of Applied Earth Observation and Geoinformation*, 6(3-4):187–198.

[Chen and Tang, 2007] Chen, J. and Tang, C. K. (2007). Spatio-Temporal Markov Random Field for Video Denoising. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[Douillard et al., 2007] Douillard, B., Fox, D., and Ramos, F. (2007). A spatio-temporal probabilistic model for multi-sensor object recognition. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2402–2408.

[Douillard et al., 2010] Douillard, B., Fox, D., and Ramos, F. (2010). Classification and Semantic Mapping of Urban Environments. *The International Journal of Robotics Research*, (in press).

[Gertz and Wright, 2003] Gertz, E. and Wright, S. (2003). Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, 29:58–81.

[Globerson and Jaakkola, 2007] Globerson, A. and Jaakkola, T. (2007). Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. *Proc. of Advances in Neural Information Processing Systems*, pages 553–560.

[Hammersley and Clifford, 1971] Hammersley, J. M. and Clifford, P. (1971). *Markov fields on finite graphs and lattices*. Manuskript.

[Ising, 1924] Ising, E. (1924). *Beitrag zur Theorie des Ferromagnetismus*. Dissertation.

[Johnson, 1997] Johnson, A. (1997). *Spin-images: a representation for 3-D surface matching*. Dissertation.

[Jordan and Weiss, 2002] Jordan, M. I. and Weiss, Y. (2002). *Probabilistic Inference in Graphical Models*. Technical Report.

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

[Kschischang et al., 2001] Kschischang, F. R., Frey, B. J., and Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional Random Fields: Probabilistic Modeling for Segmenting and Labeling Sequence Data. In *Proc. of International Conference on Machine Learning*, pages 282–289.

[Lim and Suter, 2007] Lim, E. H. and Suter, D. (2007). Conditional Random Field for 3D Point Clouds with Adaptive Data Reduction. In *Proc. of International Conference on Cyberworlds*, pages 404–408.

[Montemerlo et al., 2008] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., and Thrun, S. (2008). Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597.

[Mooij, 2010] Mooij, J. (2010). libDAI: A Free and Open Source C++ Library for Discrete Approximate Inference in Graphical Models. *Journal of Machine Learning Research*, 11:2169–2173.

[Muller et al., 2001] Muller, K. R., Mika, S., Ratsch, G., Tsuda, K., and Scholkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201.

[Munoz et al., 2009a] Munoz, D., Bagnell, J., Vandapel, N., and Hebert, M. (2009a). Contextual classification with functional max-margin markov networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 975 – 982.

[Munoz et al., 2008] Munoz, D., Vandapel, N., and Hebert, M. (2008). Directional associative markov network for 3-d point cloud classification. In *International Symposium on 3D Data Processing, Visualization and Transmission*.

[Munoz et al., 2009b] Munoz, D., Vandapel, N., and Hebert, M. (2009b). Onboard contextual classification of 3-D point clouds with learned high-order Markov Random Fields. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 4273–4280.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufman Publishing Inc.

[Ratliff et al., 2007] Ratliff, N., Bagnell, J., and Zinkevich, M. (2007). Online subgradient methods for structured prediction. In *Proc. of the International Conference on Artificial Intelligence and Statistics*.

[Ratliff et al., 2003] Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. (2003). *Subgradient Methods for Maximum Margin Structured Learning*. Technical Report.

[Rusu et al., 2009a] Rusu, R. B., Holzbach, A., Blodow, N., and Beetz, M. (2009a). Fast Geometric Point Labeling using Conditional Random Fields. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7–12.

[Rusu et al., 2009b] Rusu, R. B., Marton, Z. C., Blodow, N., Holzbach, A., and Beetz, M. (2009b). Model-based and learned semantic object labeling in 3D point cloud maps of kitchen environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3601–3608.

[Sampath and Shan, 2010] Sampath, A. and Shan, J. (2010). Segmentation and Reconstruction of Polyhedral Building Roofs From Aerial Lidar Point Clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 48(3):1554–1567.

[Shapovalov et al., 2010] Shapovalov, R., Velizhev, A., and Barinova, O. (2010). Non-associative Markov networks for 3D point cloud classification. In *Proc. of the Symposium on Photogrammetric Computer Vision and Image Analysis*.

[Taskar et al., 2004] Taskar, B., Chatalbashev, V., and Koller, D. (2004). Learning Associative Markov Networks. In *Proc. of the International Conference on Machine Learning*, pages 102–110.

[Triebel et al., 2006] Triebel, R., Kersting, K., and Burgard, W. (2006). Robust 3d scan point classification using associative markov networks. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 2603–2608.

[Urmson et al., 2008] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Geyer, C. M., Kolski, S., Howard, T., Likhachev, M., Pilnick, B., Rajkumar, R., Rybski, P., Miller, N., Nickolaou, J., Singh, S., Snider, J., Stentz, A., Whittaker, W. R., Sadekar, V., Ziglar, J., Seo, Y.-W., Demitrish, D., Litkouhi, B., Zhang, W., and Struble, J. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466.

[Wainwright et al., 2003] Wainwright, M. J., Jaakkola, T., and Willsky, A. (2003). Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *Proc. of the International Workshop on Artificial Intelligence and Statistics*.

[Wu, 1982] Wu, F. Y. (1982). The Potts model. *Reviews of Modern Physics*, 54(1):235–268.

[Wurm et al., 2009] Wurm, K. M., Kuemmerle, R., Stachniss, C., and Burgard, W. (2009). Improving Robot Navigation in Structured Outdoor Environments by Identifying Vegetation from Laser Data. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1217–1222.

[Yanover et al., 2006] Yanover, C., Meltzer, T., and Weiss, Y. (2006). Linear Programming Relaxations and Belief Propagation – An Empirical Study. *Journal of Machine Learning Research*, 7:1887–1907.

[Yedidia et al., 2000] Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2000). Generalized Belief Propagation. In *Proc. of Advances in Neural Information Processing Systems*, pages 689–695.

Eidesstattliche Erklärung der selbstständigen Arbeit

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Bonn, den 30. September 2010.

Wadim Kehl