technische universität
dortmund

Bachelorarbeit

Untersuchung von Poisson- und Multinomialverteilungsbasierten Modellen für die Analyse von Verkehrsdaten

Liudmila Vishnyakova
07. Oktober 2016

Gutachter:
Prof. Dr. Katharina Morik
Nico Piatkowski

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS8)
http://www-ai.cs.uni-dortmund.de

Abstrakt

Die vorliegende Arbeit beschäftigt sich mit zwei probabilistischen graphischen Modellen, Spatio-Temporal Random Fields und Poisson Dependency Network, die für die Vorhersagen des Verkehrsaufkommens eingesetzt werden. Die Arbeit fängt mit der Zusammenfassung der statistischen Grundlagen und theoretischen Ansätzen der beiden Modellen. Die bereits implementierte Modelle werden dann auf die vorher vorbereiteten Datensammlung angewandt. Die Experimente werden mit unterschiedlichen Trainingskonfigurationen durchgeführt. Bei STRF werden mehrere räumliche Strukturen vorgegeben. Am Ende werden die Ergebnisse zusammengefasst und evaluiert. Zentrales Ziel dieser Arbeit ist einen Überblick über die beiden Ansätze zu schaffen und eine Gegenüberstellung hinsichtlich unterschiedlicher Gütemaße aufzustellen. Um die Unterschiede und Gemeinsamkeiten besser darstellen zu können, werden die beiden Analysen auf der gleichen Datenbasis durchgeführt und die Ergebnisse am Ende mit den Testdaten verglichen.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation und Hintergrund 1

1.2 Aufbau der Arbeit 1

1.3 Key Words 2

2 Grundlagen des maschinellen Lernens **3**

2.1 Wahrscheinlichkeitstheorie 3

2.1.1 Notation 3

2.1.2 Verteilungen der Exponentialfamilie 5

2.2 Maschinelles Lernen 5

2.3 Probabilistische graphische Modelle 6

2.3.1 Graphisches Modell 7

2.3.2 Graphische Modelle und Verteilungen der Exponentialfamilie 8

2.4 Maximum-Likelihood-Schätzer 9

2.5 Regularisierung 10

2.6 Optimierungsverfahren 11

2.6.1 Fast iterative shrinkage-thresholding Algorithm 12

2.7 Statistische Inferenz 13

2.8 Gütemaße 13

2.8.1 Gesamtgenauigkeit 14

2.8.2 Genauigkeit und Vollständigkeit 14

2.8.3 Mittlere quadratische Abweichung 15

3 Spatio-Temporal Random Fields **17**

3.1 Graphisches Modell 17

3.2 Modell Optimierung 19

3.2.1 Reparametrisierung und Regularisierung 19

3.2.2 Parameterschätzung 20

3.2.3 Belief Propagation 20

4 Poisson Dependency Network **23**
4.1 Modell . . . 23
4.1.1 Log-lineares Poisson Modell . . . 24
4.2 Modell Optimierung . . . 24
4.2.1 Regularisierung . . . 25
4.2.2 Parameterschätzung . . . 25
4.2.3 Vorhersage . . . 25

5 Datenvorverarbeitung **27**
5.1 Aufbereitung der Daten . . . 27
5.1.1 Behandlung von Inkonsistenzen . . . 27
5.1.2 Behandlung fehlender Werte . . . 28
5.1.3 Behandlung konstanter Werte . . . 28
5.2 Transformation der Daten . . . 28
5.2.1 Diskretisierung . . . 29
5.3 Graphische Darstellung der Daten . . . 30

6 Experimente **35**
6.1 Aufbau der Experimente . . . 35
6.1.1 STRF . . . 35
6.1.2 PDN . . . 36
6.2 Ergebnisse . . . 37
6.2.1 Räumliche Struktur . . . 37
6.2.2 Gütemaße . . . 37

7 Zusammenfassung **47**

A DVD **49**

Abbildungsverzeichnis **51**

Tabellenverzeichnis **53**

Algorithmenverzeichnis **55**

Literaturverzeichnis **58**

Erklärung **58**

Kapitel 1

Einleitung

We are drowning in information but starved for knowledge.

John Naisbitt

1.1 Motivation und Hintergrund

Realitätsnahe Aussagen über das zu erwartende Verkehrsaufkommen sind sehr wichtig, um die möglichen Engpässe einzuschätzen und die Abläufe auf den Straßen zu optimieren. Es gibt viele Verfahren, die herangezogen werden können, um eine Aussage bezüglich der zukünftigen Verkehrslage zu treffen. Im Rahmen dieser Bachelorarbeit werden zwei Ansätze vorgestellt, die es möglich machen das Verkehrsaufkommen anhand von Sensordaten zu modellieren und vorherzusagen. Dabei hängen die Prognosewerte nicht nur von dem aktuellen Verkehrsaufkommen, sondern auch von den räumlichen Abhängigkeiten einzelner Sensoren ab. Beide Ansätze basieren auf den Techniken des maschinellen Lernens, insbesondere auf in dieser Arbeit betrachteten probabilistischen graphischen Modellen, und zielen auf kurzfristige Vorhersagen ab. Zentrales Ziel dieser Arbeit ist einen Überblick über die beiden Ansätze zu schaffen und eine Gegenüberstellung hinsichtlich unterschiedlicher Gütemaße aufzustellen. Um die Unterschiede und Gemeinsamkeiten besser darstellen zu können, werden die beiden Analysen auf der gleichen Datenbasis durchgeführt und die Ergebnisse am Ende mit den Testdaten verglichen.

1.2 Aufbau der Arbeit

Im Kapitel 2 werden die Grundlagen des maschinellen Lernens vorgestellt. Kapitel 3 bietet einen Überblick über den ersten Ansatz, Spatio-temporal Random Field (STRF). STRF basiert auf multivariater Verteilung und erzeugt ein globales Modell, das sowohl räumliche als auch zeitliche Abhängigkeiten zwischen allen Sensoren im Verkehrsnetz repräsentiert. Das Modell ist parametrisiert und ihre Parameter werden mit dem Maximum-Likelihood-Schätzer berechnet. In Kapitel

4 geht es um den zweiten Ansatz: Poisson Dependency Network. Dieses setzt für die Modellierung der Abhängigkeiten eine approximative multivariate Poisson-Verteilung ein. Dabei wird eine bestimmte Menge von lokalen Modellen gebildet, die dann schrittweise optimiert und zu einem globalen Modell zusammengesetzt wird. Kapitel 5 handelt von der zur Evaluation der Verfahren benutzten Datensätze, deren Vorverarbeitung und Visualisierung. Die Frage wie gut die Verfahren die Daten modellieren können, werden in einer Studie untersucht, die in Kapitel 6 vorgestellt wird. Abschließend wird eine Zusammenfassung gegeben.

1.3 Key Words

Maschinelles Lernen, Probabilistische Graphische Modelle, Random Fields, Poisson Regression

Kapitel 2

Grundlagen des maschinellen Lernens

Im diesem Kapitel wird ein Überblick über grundlegende Begriffe und Verfahren aus dem Bereich des maschinellen Lernens gegeben. Der Abschnitt 2.1 bietet eine kurze Einführung in die grundlegende Konzepte und Ideen der Wahrscheinlichkeitstheorie. Es wird eine für die Regressions- oder Klassifikationsanalyse wichtige Klasse der Wahrscheinlichkeitsverteilungen, die Exponentialfamilie, vorgestellt. Auch ein Repräsentant dieser Klasse, die Poisson-Verteilung, wird hier kurz erläutert. Im Abschnitt 2.2 wird das Konzept maschinelles Lernen behandelt. Der Abschnitt 2.3 widmet sich den probabilistischen graphischen Modellen, die das Werkzeug zur kompakten Beschreibung von komplexeren Verteilungen unter Ausnutzen der Struktur zur Verfügung stellen. In den darauf folgenden Abschnitten 2.4 und 2.5 werden die grundlegenden Techniken erklärt, die für das Schätzen und die Regularisierung der Modellparameter benötigt werden. Die hier vorgestellten Konzepte werden Sie in [2, 6, 11] finden.

2.1 Wahrscheinlichkeitstheorie

Die Wahrscheinlichkeitstheorie bildet ein Fundament für die Verfahren im Bereich des maschinellen Lernens. Zusammen mit der Entscheidungstheorie ermöglicht es uns die Vorhersagen anhand der verfügbaren Informationen zu treffen, auch wenn diese nicht vollständig sind [2]. Im Folgenden werden kurz die grundlegende Begriffe und Konzepte der Wahrscheinlichkeitstheorie vorgestellt. Für weitere Informationen sei auf zahlreiche Einführungsbücher zur Statistik verwiesen, wie z. B. [4, 5, 6, 17].

2.1.1 Notation

In der Wahrscheinlichkeitstheorie werden Zufallsexperimente betrachtet. Sei X eine univariate Zufallsvariable und $\mathscr{X} = \{x_1, x_2, ..., x_n\}$, eine Menge von allen möglichen Ergebnissen eines Zufallsexperimentes, ihr Zustandsraum. Die Wahrscheinlichkeitsverteilung gibt die Wahrscheinlichkeit an, dass X den Wert x_i annimmt, und ist als $p(x_i) = p(X = x_i)$, $x_i \in \mathscr{X}$ definiert. Außerdem hat eine Wahrscheinlichkeitsverteilung folgende Eigenschaften $p(x_i) \in [0,1], \forall x_i \in \mathscr{X}$ und

$p(\mathscr{X}) = \sum_{i=1}^{n} p(x_i) = 1$.

In der Arbeit spielen die Wechselwirkungen zwischen einzelnen Sensoren eine große Rolle, deswegen wird jetzt eine multivariate multinomialverteilte Zufallsvariable eingeführt.

Multivariate Zufallsvariable. Eine multivariate Zufallsvariable $\mathbf{X} = (X_1, X_2, ..., X_n)$ ist eine Funktion, die mehrere Variablen zu einem Vektor zusammenfasst. Das Zustandsraum von $\mathbf{X}$ ist ein Kreuzprodukt $\mathscr{X} = \mathscr{X}_1 \otimes \mathscr{X}_2 \otimes \cdots \otimes \mathscr{X}_n$, in unserem Fall $\mathscr{X} = \mathbb{R}^n$.

Die gemeinsame Wahrscheinlichkeitsverteilung einer multivariaten Zufallsvariable ist gegeben durch:

$$p(\mathbf{x}) = p(\mathbf{X} = \mathbf{x}) = p(X_1 = x_1, X_2 = x_2, ..., X_n = x_n)$$

Wenn $|\mathscr{X}_i| > 1$ spricht man von einer Multinomialverteilung.

Bei Zufallsvariablen kann man die gemeinsame Wahrscheinlichkeitsverteilung auch mit Hilfe von bedingter Wahrscheinlichkeit ausdrücken. Betrachten wir hier einfachheitshalber zwei (diskrete) Variablen X_i und X_j, dann ist die gemeinsame Verteilungsfunktion wie folgend definiert

$$p(x_i, x_j) = p(x_j \mid x_i) \cdot p(x_i) = p(x_i \mid x_j) \cdot p(x_j) \tag{2.1}$$

In dieser Arbeit werden die Zufallsvariablen mit großen Buchstaben X,Y,Z gekennzeichnet. Ein konkreter Wert x_i, der eine Zufallsvariable annehmen kann, wird klein geschrieben. Der Vektor $\mathbf{X}$ beschreibt hier eine Menge von Zufallsvariablen und $\mathbf{x}$ einen konkreten Zustand einer multivariaten Zufallsvariable. Beide Zeichen werden fett hervorgehoben.

Randverteilung. Die Wahrscheinlichkeit einer bestimmten Belegung einer Zufallsvariable x_j wird mit Hilfe einer Randverteilung berechnet. Dabei werden die Wahrscheinlichkeiten aller Beobachtungen $\mathbf{x}_{\setminus x_j}$ aufsummiert, bei denen x_j diesen bestimmten Wert annimmt.

$$p(x_j) = \sum_{x_i \in \mathscr{X}_{\setminus \mathscr{X}_j}} p(x_i, x_j) \tag{2.2}$$

Die Berechnung der Randverteilung wird auch als Inferenz bezeichnet.

Unabhängigkeit. Zwei Zufallsvariablen X_i und X_j (kurz $X_i \perp X_j$) sind unabhängig voneinander, wenn gilt

$$p(X_i \mid X_j = x_j) = p(X_i),\ p(X_i = x_i, X_j = x_j) = p(X_i = x_i) p(X_j = x_j) \tag{2.3}$$

Die bedingte Unabhängigkeit $X \perp Y \mid Z$ ist genau dann gegeben, wenn 2.1 gilt. Die letzte Gleichung stellt die Faktorisierung dar.

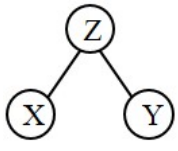

Abbildung 2.1: Graphisches Modell.

$$\begin{aligned} p(X = x \mid Z = z, Y = y) &= p(X = x \mid Z = z) \\ p(Y = y \mid Z = z, X = x) &= p(Y = y \mid Z = z) \\ p(Z = z \mid X = x, Y = y) &= p(Z = z \mid X = x) p(Z = z \mid Y = y) \end{aligned} \tag{2.4}$$

2.1.2 Verteilungen der Exponentialfamilie

Die Wahrscheinlichkeitsverteilungen der Exponentialfamilie spielen in der Datenanalyse eine große Rolle und sind dank ihrer besonderen statistischen Eigenschaften gut für die Schätzung der Parameter geeignet.
Eine Wahrscheinlichkeitsfunktion gehört zu einer Exponentialfamilie, falls gilt

$$p_{\theta,\phi}(x) = \exp\left(\frac{x\theta - b(\theta)}{\phi} + c(x,\phi)\right) \tag{2.5}$$

Dabei ist θ ein Parametervektor, ϕ steuert die Streuung der Verteilung und die anderen Funktionen sind bekannt und von der Verteilung vorgegeben. Die bekanntesten und wichtigsten Verteilungen, die zur Exponentialfamilie gehören, sind die Normal-, Binomial-, Poisson-, und Gammaverteilung.
Wenn man in die Funktion 2.5 folgende Werte einsetzt

- $\theta = log(\lambda)$
- $\phi = 1$
- $b(\theta) = \lambda = \exp(\theta)$
- $c(x,\phi) = -log(x!)$

bekommt man eine Poisson Verteilung raus

$$f_\lambda(x) = \exp\left(\frac{x\log(\lambda) - \lambda}{1} - log(x!)\right) = \frac{1}{x!}\lambda^x \exp(-\lambda), x \in \mathbb{N}_0 \tag{2.6}$$

Die Poisson-Verteilung ist eine diskrete, parametrisierte Wahrscheinlichkeitsverteilung über natürlichen Zahlen. Eine Zufallsvariable X mit der Wahrscheinlichkeitsfunktion 2.6 ist Poisson-verteilt mit Parameter $\lambda > 0$. Die Verteilung gibt die Wahrscheinlichkeit an, x Ereignisse zu beobachten, wenn die erwartete Häufigkeit/Mittelwert der Ereignisse λ ist.
Die Gleichheit des Erwartungswertes und der Varianz

$$E(X) = \lambda = Var(X) \tag{2.7}$$

ist charakteristisch für eine poisson-verteilte Zufallsvariable.

2.2 Maschinelles Lernen

Das maschinelle Lernen spielt heutzutage eine zunehmend wichtige Rolle in der Wirtschaft, Industrie und Forschung. Es ist heute allgegenwärtig. Die Techniken und Algorithmen werden in vielen unterschiedlichen Bereichen eingesetzt. Sie ermöglichen zum Beispiel eine effiziente Web-Suche, geben uns Kaufempfehlungen in diversen Online-Shops, erkennen Spam-Mails in unserem E-Mail-Fach. Was ist aber maschinelles lernen?

Als Lernen bezeichnet man die Anpassung eines Modells an die gegebenen Daten. Man unterscheidet dabei zwischen verschiedenen Lernmethoden im statistischen Lernen, die meist verwendeten Techniken kann man in [2] finden. Die zweite Hervorhebung: Diese werden in zwei großen Kategorien unterteilt in das überwachte (*supervised*) und unüberwachte (*unsupervised*) Lernen.
Gegeben sei ein Datensatz $\mathscr{D} = \{\mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^n\}$ mit N Beobachtungen. Jede Beobachtung $\mathbf{X}^i$ wird durch eine Menge von Merkmalen $\mathbf{X}^i = \{X_1, X_2, \cdots, X_m\}$ beschrieben. Jedes Merkmal kann einen bestimmten Wert $X_i = x_i$, $x_i \in \mathscr{X}_i$ annehmen, der sowohl qualitativ, als auch quantitativ sein kann, der aber aus dem Zustandsraum der zugehörigen Merkmal stammt. Bei der überwachten Lernmethoden, um die es in dieser Arbeit auch geht, setzen sich die Beobachtungen $\mathscr{D} = \{(\mathbf{x_1}, y_1), \cdots, (\mathbf{x_n}, y_n)\}$ aus einem Merkmalsvektor $\mathbf{x} \in \mathscr{X}^m$ und eine Zielvariable $y \in \mathscr{Y}$ zusammen. Die Aufgabe ist es eine Funktion $f(\mathbf{x}) = y$ zu finden, welche zu jeder Beobachtung $\mathbf{x} \in \mathscr{X}^m$ eine möglichst präzise $y \in \mathscr{Y}$ vorhersagen kann. Anders als bei dem überwachten Lernen, ist der Zielvektor beim unüberwachten Lernen vorher nicht bekannt. Das Ziel hier ist es die Ähnlichkeit zwischen verschiedenen Datenobjekten anhand der gegebenen Merkmale zu erkennen und diese Objekte zu gruppieren.
Beim überwachten Lernen unterscheidet man zwischen zwei Problemstellungen: Klassifikation und Regression. Bei der Regression versucht man einen nummerischen Wert $y \in \mathbb{R}$ vorherzusagen, bei Klassifikation ist y nominal.
Der Lernvorgang läuft meistens nach folgenden Schema durch. Bevor man ein Lernverfahren an die Daten anwenden kann, müssen die Daten vorbereitet werden (s. Abschnitt 5 Datenvorverarbeitung). Nach diesem Schritt teilt man den Datensatz in Trainings- $\mathscr{T}_{train}$ und Testset $\mathscr{T}_{test}$. Der Testset wird später für die Bewertung der Funktion benötigt. Die Aufteilung der Daten kann auf unterschiedliche Art und Weise geschehen. Bei der Kreuzvalidierung, z. B. wird der gesamte Datenbestand in n Mengen aufgeteilt. Das Modell wird dabei auf $n-1$ Mengen trainiert und auf der n-te Menge getestet. Dieses Vorgang wird dann n-Mal wiederholt, wobei jede Menge ein mal als Testmenge eingesetzt wird. Anhand der Trainingsdaten wird eine möglichst gute Annäherung $\hat{f}(\mathbf{x})$ an die wahre Funktion $f(\mathbf{x})$ gesucht. Diese Annäherung wird als Modell bezeichnet. Abschließend wird das gelernte Modell an den Testdaten $\mathscr{T}_{test}$ mit Hilfe von unterschiedlichen Gütemaßen (vgl. Anschnitt 2.8) evaluiert. Die Qualität des Modells wird als Mittel der gemessenen Qualitäten bei der Kreuzvalidierung bestimmt.

2.3 Probabilistische graphische Modelle

Probabilistische graphische Modelle (PGM) spielen eine wichtige Rolle in Analyse und Design von Algorithmen. In seinem Buch [10] hat Michael I. Jordan PGM als Ehe zwischen Wahrscheinlichkeits- und Graphentheorie beschrieben. Die Modelle ermöglichen eine kompakte Repräsentation von statistischen Abhängigkeiten mit Knoten und Kanten, sodass die Probleme besser veranschaulicht und mit geringerer Rechenkomplexität gelöst werden können[19]. Aus diesem Grund werden die-

Folgende maximale Cliquen werden hier definiert:
$C_1 = \{v_1, v_2, v_3\}, C_2 = \{v_1, v_5, v_6, v_7\}, C_3 = \{v_1, v_4\}$.
$\mathbf{x}_{C_i}$ bezeichnet eine Belegung der Knoten, die in C_i enthalten sind. Die Wahrscheinlichkeit einer solcher Belegung wird mit Randverteilung 2.2 berechnet.
Die gemeinsame Verteilung lässt sich entsprechend faktorisieren $p(x_1, x_2, \cdots, x_7) = \frac{1}{K}\psi_1(\mathbf{x}_{C_1})\psi_2(\mathbf{x}_{C_2})\psi_3(\mathbf{x}_{C_3})\psi_4(\mathbf{x}_{C_4})$.

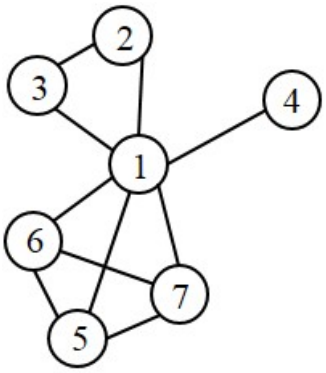

Abbildung 2.2: Graphisches Modell. Jedem Knoten $v_i \in V$ wird ein Wert $x_i \in \mathscr{X}_i$ zugeordnet.

se Modelle sehr oft in Bereichen, die sich mit der Analyse sehr großen Datenmengen befassen, eingesetzt.

2.3.1 Graphisches Modell

Ein PGM wird durch eine parametrisierte multivariate Wahrscheinlichkeitsverteilung p_θ und einen Graph G beschrieben. Dieser Abschnitt fängt mit der Einführung in die Graphentheorie an. Da es sich hier um eine kurze Einführung handelt, wird an dieser Stelle auf einen Einstiegskurs in der Graphentheorie [3] für weitere Informationen verwiesen.
Ein Graph $G = (V, E)$ ist ein Tupel von der endlichen Menge der Knoten $V = V_1, V_2, \ldots, V_m$ und der endliche Menge der Kanten $E \subset V \times V$. Dabei verbindet jede Kante genau zwei Knoten $(V_i, V_j) \subset E$. Für ein gerichteten Graph, spielt die Reihenfolge der Knoten eine Rolle, dabei unterscheidet sich (v_1, v_2) von (v_2, v_1). Bei ungerichteten Graph ist jede Kante ein ungeordnetes Paar, so dass die beide Tupel (v_1, v_2) und (v_2, v_1) äquivalent sind.
Sei $G = (V, E)$ in weiterem Verlauf ein ungerichteter Graph. Für weitere Informationen über gerichtete Graphen sei hier auf [19, Chapter 2.1.1] verwiesen.
Jede Zufallsvariable X_i wird durch einen Knoten $v_i \in V$ und die Abhängigkeitsstruktur zwischen den Zufallsvariablen wird durch die Kantenmenge E repräsentiert. Die Grundidee, die hinter der graphischen Modellen steckt, ist Faktorisierung der Wahrscheinlichkeitsverteilungen gemäß der vorgegebenen Struktur. Im Fall von ungerichteten Modellen, werden die maximale Cliquen betrachtet. Als Clique C wird eine Teilmenge von Knoten definiert, wo jedes Knotenpaar miteinander verbunden ist. $\exists(s,t) \in E$ für $\forall s,t \in C$ und $C \subset V$ (Abb. 2.2). Jede Clique C_i ist ein vollständiger Teilgraph von G, für die auch eine gemeinsame Funktion $\psi_{C_i} : (\otimes_{v_i \in C} \mathscr{X}_i) \longrightarrow \mathbb{R}_+$ definiert wird. $\otimes_{v_i \in C} \mathscr{X}_i$ ist ein Kartesisches Produkt der Zustandsräume von $\forall X_i \in \mathbf{X}_{C_i}$.
Markov-Eigenschaft. Sei $U \subseteq V$ eine Teilmenge von Knoten und $\bar{U} = V - U$. Sei $pa(\mathbf{x}_U)$ die

Menge aller Knoten, die mindestens eine Kante zu dem Knoten aus x_U haben, auch Eltern genannt. Ein PGM hat die Markov-Eigenschaft genau dann, wenn gilt

$$p(\mathbf{x}_U \mid \mathbf{x}_{\bar{U}}) = p(\mathbf{x}_U \mid pa(\mathbf{x}_U)).$$

Gemäß der oben genannten Notationen wird die gemeinsame Wahrscheinlichkeitsverteilung eines ungerichteten graphischen Modells, das auch die Markov-Eigenschaft erfüllt, als ein Produkt von einzelnen Wahrscheinlichkeitsverteilungen definiert

$$p(x_1, x_2, \cdots, x_m) = \prod_{i \in V} p_i(x_i \mid pa(x_i)) \tag{2.8}$$

Bei ungerichteten graphischen Modellen lässt sich die gemeinsame Verteilung mittels der vorher festgelegten Cliquen faktorisieren

$$\begin{aligned} p(x_1, \ldots, x_m) &= \frac{1}{K} \prod_{C \in \mathscr{C}} \psi_C(\mathbf{x}_C) \\ K &= \sum_{\mathbf{x} \in \mathscr{X}} \prod_{C \in \mathscr{C}} \psi_C(\mathbf{x}_C) \end{aligned} \tag{2.9}$$

K ist eine Konstante, die die Funktion normalisieren soll. Dies ist erforderlich, damit die Funktion der Eigenschaften einer Wahrscheinlichkeitsverteilung entspricht. $\mathscr{C}$ ist die Menge aller maximalen Cliquen im Graph. Die ψ_C stellt die Faktorisierung dar.

$$\psi_C(\mathbf{x}_C) = \exp(\langle \theta_C, \phi_C(\mathbf{x}) \rangle)$$

2.3.2 Graphische Modelle und Verteilungen der Exponentialfamilie

Die PGM ermöglichen es die komplexen Wahrscheinlichkeitsverteilungen mittels graphischer Modelle darzustellen. Viele graphische Modelle werden mit der Verteilungsfunktionen exponentieller Familien beschrieben. Diesen Fakt kann man mit dem Prinzip von Shanon Entropie erklären.
Sei $G = (V, E)$ ein Graph wie vorher im Abschnitt 2.3.1 beschrieben ist. Zuerst wird eine Indikatorfunktion ϕ_x für eine feste, aber zufällige Ordnung von Knoten und Kanten in $\mathscr{X}$ definiert. Die Indikatorfunktion sagt aus, welcher Knoten in welchem Zustand sich befindet.

$$\phi_C(\mathbf{x}) = \begin{cases} 1, & \text{wenn } \mathbf{x}_C \in \mathscr{X}_C, \\ 0, & \text{sonst.} \end{cases}$$

Der empirische Moment oder Erwartungswert für $\mathbf{x}$ beschreibt die geschätzte empirische Häufigkeit von einem bestimmten Ereignis

$$\hat{\mu}_{x_C} := \frac{1}{N} \sum_{i=1}^{N} \phi_{x_C}(\mathbf{x}_i). \tag{2.10}$$

Die Wahrscheinlichkeitsverteilungsfunktion p mit einem Erwartungswert $\mathbb{E}[\phi_C(\mathbf{x})]$ heißt nur dann konsistent mit dem Datensatz $\mathscr{D}$, wenn der empirische Moment (2.10) und Erwartungswert übereinstimmen (dieses Ereignis heißt auch *Moment Matching*). Es gibt mehrere Möglichkeiten eine

Verteilung p zu definieren, die die Moment Matching Bedingung erfüllt. Die Grundidee von maximale Entropie ist es eine Verteilungsfunktion p^* zu finden, deren Shanon Entropie $\mathscr{H}(p)$ maximal ist.

$$\mathscr{H}(p) := -\int_{\mathscr{X}} (\log p(x)) p(x) \nu(dx) \tag{2.11}$$

Die Lösung für p^* wird durch die Lösung vom beschränkten Optimierungsproblem gegeben

$$p^* := \arg\max_{p \in \mathscr{P}} \mathscr{H}(p), \text{ wenn } \mathbb{E}_p[\phi_C(\mathbf{x})] = \hat{\mu}_{\mathbf{x}_C}. \tag{2.12}$$

Man kann auch zeigen, dass die optimale Lösung dieser Gleichung die Form einer Verteilung der Exponentialfamilie hat

$$p_\theta(\mathbf{x}) = \exp[\langle \theta, \phi(\mathbf{x}) \rangle - A(\theta)]. \tag{2.13}$$

Mit einer log-Partition Funktion $A(\theta) = \ln K$.

2.4 Maximum-Likelihood-Schätzer

In diesem Abschnitt wird das Grundprinzip der Maximum-Likelihood-Schätzers erklärt. Um ein Modell an den gegebenen Datensatz anzupassen, müssen die Modellparameter der gemeinsamen Verteilung θ anhand der beobachteten Werten geschätzt werden. Das Ziel ist dabei so eine Parameterkombination zu finden, die die beobachteten Werte am besten widerspiegelt. Wenn der Datensatz vollständig ist, kann man die unbekannte Parametern mit der Maximum-Likelihood Methode schätzen lassen.
Die *Likelihood*-Funktion $\mathscr{L}$ nimmt die gemeinsame Wahrscheinlichkeitsverteilung und interpretiert es als Funktion unbekannter Parameter θ für gegebenen Datensatz $\mathscr{D} = \{\mathbf{x}^1, \cdots, \mathbf{x}^N\}$. Der *Likelihood* soll maximiert werden.

$$\mathscr{L}(\theta; \mathscr{D}) := \prod_{\mathbf{x} \in \mathscr{D}} p_\theta(\mathbf{x}). \tag{2.14}$$

Auf Grund von sehr kleinen Wahrscheinlichkeiten der einzelnen Werten neigt dazu, sehr klein zu sein. Die *log-Likelihood*-Funktion ℓ liefert gleiches Ergebnis wie 2.14, weil die Logarithmus-Funktion streng monoton steigend ist und das Maximum sich an der gleichen Stelle befindet. Die Zielfunktion sollte mit Hilfe von Optimierungsverfahren maximiert oder bei dem negativem log-Likelihood minimiert werden.

$$\ell(\theta; \mathscr{D}) = \frac{1}{N} \sum_{i=1}^{N} \log(p_\theta(\mathbf{x}_i)) \tag{2.15}$$

Mit $\hat{\mu}$ 2.10 wird der log Likelihood für die Verteilungen der exponentiellen Familie definiert als

$$\ell(\theta \mid \mathscr{D}) = \langle \theta, \hat{\mu} \rangle - A(\theta). \tag{2.16}$$

Der Nachteil vom Maximum-Likelihood-Schätzer, besteht darin, dass er versucht für jede Zufallsvariable ein Parameter θ_i größer 0 zu finden, obwohl einige Variablen für die Vorhersage irrelevant

sein können. Zu viele Parameter können dann zu einer Überanpassung (*Overfitting*) führen. Von der Überanpassung spricht man, wenn ein Modell zu sehr an die Trainingsdaten angepasst wird, und somit weniger repräsentativ für die Grundgesamtheit ist. Dieses Problem kann man an einen hohen Testfehler erkennen. Dieser Effekt kann durch die Regularisierung 2.5 geschwächt werden. Bei einem unvollständigen Datensatz kann man den Expectation-Maximization (EM) Algorithmus zum Schätzen der Parameters einsetzen. Die Idee von EM-Algorithmus ist es den fehlenden Wert/die fehlende Werte zuerst zu schätzen (*Expectation*), z. B. man setzt für die fehlende Werte den Erwartungswert ein. Danach wird der *Likelihood* maximiert (*Maximization*) und ein neuer Schätzwert für die nächste Iteration eingesetzt. Der Vorgang wird solange wiederholt bis die Konvergenz oder eine maximale Anzahl an Schritten, die vorher festgesetzt wurde, erreicht wird.
Da die ausführliche Beschreibung des EM-Algorithmus den Rahmen der Bachelorarbeit sprengen würde, wurde hier nur die Grundidee des EM-Algorithmus erklärt. Weitere Informationen kann man in [12] finden. In dieser Arbeit werden die Parameter nach der Vorverarbeitung der Daten mit Maximum-log-Likelihood geschätzt.

2.5 Regularisierung

Komplexe Modelle mit vielen Parametern führen oft zur Überanpassung (*Overfitting*). Es wird schwierig eine Funktion zu finden, die die neuen Beobachtungen möglichst gut klassifiziert. Außerdem wird es fast unmöglich diese Parameter auf begrenzten Ressource zu bestimmen und ein gutes Modell in Echtzeit zu lernen. Um ein kompaktes aber robustes Modell zu Erhalten, braucht man die Regularisierung.
Nach der Regularisierung kann eine Vorauswahl an Parametern getroffen werden, die für den Aufbau des Modells nützlich sein könnten. Somit wird die Komplexität des Modells verringert, weil die Variable, die wenig relevante oder keine Informationen liefern, entfernt werden. Ein einfaches Modell lässt sich auch besser auf die Grundgesamtheit anwenden.
Die bekanntesten Regularisierungsmethoden sind Lasso ℓ_1 und Ridge-Regression ℓ_2. Die Kombination von den beiden Methoden wird Elastic-Net genannt.
Die ℓ_1-Regularisierung macht die Daten spärlicher, weil die wenig informativen Merkmale am Ende der Regularisierung entfernt werden. Die kleine Modellparameter werden bestraft und ein Modell eingeschränkt, in dem die Kante zu den Knoten, dessen Parameter 0 ist, entfernt wird. Diese spärliche Lösung hilft einige wenige wichtige Faktoren zu identifizieren.

$$\|\theta\|_1 = \sum_{i=1}^{d} |\theta_i| \tag{2.17}$$

In dieser Arbeit wird die ℓ_1-Regularisierung auch für das Schätzen von räumlichen Struktur eingesetzt [18]. In den meisten Fällen wird die graphische Struktur, die die Abhängigkeiten zwischen Variablen in dem Model erfasst, vorgegeben. Es gibt aber Fälle, wo man nur Daten vorliegen hat und nicht weiß welche Beziehung die zueinander haben, die graphische Struktur eines Modells ist

also unbekannt oder wenig aussagekräftig. Es gibt mehrere Gründe, warum es manchmal besser nicht die gegebene Struktur zu nehmen, sondern die wenig aussagekräftige Struktur zu optimieren oder gar neu schätzen zu lassen. Somit kann man eine bessere Struktur gewinnen, die genau die statistische Abhängigkeitsverhältnisse im gegebenen Datensatz repräsentiert. Auch die Parameterschätzung wird in dem Fall effizienter, dank wenig vorgegebenen Daten.
Es gibt aber auch Nachteile, weil es unzählige Varianten gibt, wie eine Struktur aufgebaut werden kann. Bei den ungerichteten Graphen muss jeder Knoten neu angepasst werden, wenn eine Kante hinzugefügt oder entfernt wird. Jeden Parameter bei jeder Änderung anzupassen erfordert natürlich die Rechenleistung. Bei der ℓ_2-Regularisierung wird die Bestrafung der zu hohen Parameter quadriert

$$\|\theta\|_2 = \sqrt{\sum_{i=1}^{d} \theta_i^2}. \tag{2.18}$$

Die beiden Regularisierungs-Terme (Ausdrucke) 2.17 und 2.18 werden auf die Zielfunktion drauf addiert

$$\min h(\theta) := -\ell(\theta \mid \mathscr{D}) + \lambda_1 \|\theta\|_1 + \frac{\lambda_2}{2}\|\theta\|_2^2. \tag{2.19}$$

Die Regularisierungsparameter $\lambda > 0$ stellt einen möglichst optimalen Kompromiss zwischen der Genauigkeit der Werte und der Empfindlichkeit gegenüber der Ausreißer dar.
Wenn man die beiden Methoden vergleicht, neigt die ℓ_1-Regularisierung mehr dazu, die Parameter mehr zu bestrafen, so dass mehrere davon nach der Regularisierung gleich 0 gesetzt werden. Es ist aber wichtig ein richtiges Maß für die Modell-Komplexität zu finden, weil es auch einen Einfluss auf die Vorhersagegenauigkeit hat.

2.6 Optimierungsverfahren

Die numerischen Optimierungsverfahren sind für die effiziente Berechnung der Zielfunktion, die minimiert bzw. maximiert werden soll, erforderlich. Dabei spielt die Verteilung der Zielfunktion bei der Wahl der Optimierungsverfahren eine große Rolle. Zwei große Klassen numerischer Optimierungsverfahren sind das Gradientenverfahren und Newton-Verfahren.
Der Gradient. Als Gradient bezeichnet man den Vektor der partiellen Ableitungen 1. Ordnung einer Funktion $h(\theta)$ 2.19 nach allen Parametern.

$$\nabla h(\theta) = \begin{pmatrix} \frac{\partial h}{\partial \theta_1} \\ \frac{\partial h}{\partial \theta_2} \\ \vdots \\ \frac{\partial h}{\partial \theta_m} \end{pmatrix} \tag{2.20}$$

Der Gradient $\nabla h(\theta)$ zeigt an der Stelle θ in die Richtung der steilsten Abstiegs.
Hesse-Matrix. Als Hesse-Matrix bezeichnet man eine Matrix der partiellen Ableitungen 2.Ordnung einer Funktion $h(\theta)$.

$$\nabla^2 h(\theta) = \begin{pmatrix} \frac{\partial h}{\partial \theta_1^2} & \cdots & \frac{\partial h}{\partial \theta_1 \partial \theta_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial h}{\partial \theta_m \partial \theta_1} & \cdots & \frac{\partial h}{\partial \theta_m^2} \end{pmatrix} \tag{2.21}$$

Obwohl das Newtonverfahren schneller ist, wird es in dieser Arbeit nicht eingesetzt, weil die Funktion zu viele Parameter hat und die Hesse-Matrix und die für Newton-Verfahren benötigte Invertierung derer schwer zu bestimmen sind. In dieser Arbeit wird das Gradientenverfahren oder das Verfahren des steilsten Abstiegs (*Gradien Descent*) zur Minimierung der Zielfunktion aus 2.19 verwendet. Bei diesem Verfahren wird die Richtung des steilsten Abstiegs von $h(\theta)$ bestimmt. Dabei wird der Gradientenabstieg bei jeder Iteration aktualisiert. Wichtig ist dabei eine gute Schrittweite $\eta \in \mathbb{R}$ zu wählen.

$$\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)} \nabla h(\theta) \tag{2.22}$$

Für weitere Informationen über die numerische Optimierung sei hier auf eine Einführung in die Optimierung von Prof. Dr. Harrach [9] verwiesen.

2.6.1 Fast iterative shrinkage-thresholding Algorithm

Fast iterative shrinkage-thresholding Algorithm (FISTA) ist ein proximales numerisches Gradientenverfahren, welches zur Lösung konvexen Optimierungsprobleme eingesetzt wird [1]. FISTA ist eine Erweiterung vom iterative shrinkage-thresholding Algorithm (ISTA) und wird in der Arbeit bei dem ℓ_1-Regularisierungsproblem eingesetzt.
Das Optimierungsproblem in 2.19 besteht daran, dass $\|\theta\|_1$ 2.17 in $\|\theta\|_1 = 0$ nicht differenzierbar ist.
Das Problem wird in ISTA aufgeteilt, indem die Likelihood-Funktion normal abgeleitet und ℓ_1-Regularisierung mit Hilfe von einem proximalen Operator [14] berechnet wird.

$$\text{prox}_{\alpha \cdot \|\theta^{(t+1)}\|_1}(\theta) = \arg\min_{\theta} \left\{ \|\theta^{(t)}\|_1 + \left(\frac{1}{2\alpha} \|\theta^{(t)} - \theta^{(t+1)}\|_2^2 \right) \right\} \tag{2.23}$$

Dabei bestimmt $\alpha = \gamma\lambda, \alpha > 0$ die Schrittweite der proximalen Gradient.

$$\theta^{(t+1)} := \text{prox}_{\alpha^{(t)} \|\cdot\|_1} \left(\theta^{(t)} - \alpha^{(t)} \nabla - \ell(\theta^{(t)}; \mathscr{D}) \right) \tag{2.24}$$

Das Minimierungsproblem aus 2.23 wird mit Hilfe von *soft thresholding* gelöst.

$$\text{prox}_{\alpha \|\theta\|_1}(\theta) = \begin{cases} \theta - \alpha & \theta \geq \alpha \\ 0 & |\theta| \leq \alpha \\ \theta + \alpha & \theta \leq -\alpha \end{cases} \tag{2.25}$$

ISTA ist langsam und hat in Worst-Case einen Komplexität von $O(1/t)$. FISTA kann diesen Komplexität verringern und ISTA bis zum Gradientenverfahren reduzieren. FISTA baut ihre Berechnungen aufeinander auf, dabei werden die zwei letzten Vorgänger $\theta^{(t+1)} = \theta^{(t)} + \left(\frac{t_k-1}{t_{k+1}}\right)(\theta_t - \theta_{t-1})$ betrachtet.

2.7 Statistische Inferenz

Während des Lernvorgangs stößt man auf verschiedene Probleme der statistischen Inferenz. Zum einen stellt sich die Frage, wie die Modellparameter effizient berechnet werden können, zum anderen wie die Randverteilung $p(\mathbf{x}_U)$ von einer bestimmter Teilmenge $U \subseteq V$ bestimmt wird. Auch die Berechnung der bedingten Wahrscheinlichkeitsverteilungen und der wahrscheinlichsten Belegung der Knoten (*Maximum a posteriori*) stellen eine Herausforderung dar.
Die Randverteilungen und die maximale wahrscheinlichste Zuordnung von allen Variablen kann mit Hilfe von Belief-Propagation ([15]) berechnet werden. Die Vorhersage kann mit dem MAP Schätzer berechnet werden. Ziel der Maximum a posteriori Schätzer(MAP) ist es die wahrscheinlichste Zuordnung von allen Variablen zu finden. MAP kann auch als Regularisierung von Maximum-Likelihood (2.4) angesehen werden, wenn θ auch eine Zufallsvariable ist.
Die Aufgabe der MAP ist bei den fehlenden Werten, die meist wahrscheinliche Wertzuordnung x_i für die Variable X_i zu finden, wenn alle andre Variablen gegeben sind.

$$\mathrm{MAP}(X_i \mid \mathbf{X}_{\setminus X_i}) = \arg\max_{x \in \mathscr{X}_m} p(x) \tag{2.26}$$

Hierbei repräsentiert $\arg\max_{x \in \mathscr{X}_m} f(x)$ den Wert von x, für den $f(x)$ maximal ist. Wichtig ist, dass hier eine gemeinsame Verteilung betrachtet wird und nicht jede einzelne Variable. Diese Marginalisierung der Variable stellt ein Problem dar.
Marginal Maximum a posteriori berechnet eine wahrscheinlichste Wertzuordnung gegeben einer Teilmenge von Variablen. Sei $\mathbf{S}$ eine Teilmenge von $\mathbf{X}$, dann gilt

$$\mathrm{MAP}(\mathbf{S} \mid \mathbf{X}_{\setminus \mathbf{S}}) = \arg\max_{\mathbf{s}} p(\mathbf{s} \mid X_{\setminus \mathbf{S}}). \tag{2.27}$$

Man kann die Performanz durch die Faktorisierung dieser Zufallsvariablen erreichen. Bei einem graphischen Modell kann man die Erreichbarkeit von Knoten ausnutzen und die gemeinsame Verteilung faktorisieren

$$p(\mathbf{x}) = \frac{1}{K} \prod_{s \in V} \psi_s(\mathbf{x}_s) \prod_{(s,t) \in E} \psi_{s,t}(\mathbf{x}_s, \mathbf{x}_t). \tag{2.28}$$

2.8 Gütemaße

In diesem Abschnitt werden die Funktionen zur Beurteilung der Modellqualität zusammengefasst. Eine Bewertungsfunktion sagt aus wie gut ein Modell die Daten beschreibt. Bei der Bewertung der Funktion wird der Zielvektor $\mathbf{y}$ für eine Testmenge $\mathscr{T}_{test}$, die bei dem Training nicht berücksichtigt

		vorhersagte Klasse	
		y_i	$\mathscr{Y}_{\setminus y_i}$
tatsächliche Klasse	y_i	True Positive	False Negative
	$\mathscr{Y}_{\setminus y_i}$	False Positive	True Negative

Mit Hilfe von einer Konfusionsmatrix lassen sich die Modelle bezüglich einer bestimmten Klasse bewerten.

Tabelle 2.1: Konfusionsmatrix

wurde, vorhergesagt. Diese Vorhersagen werden dann mit den tatsächlichen Werte/Klassen verglichen. Ziel ist es dabei ein Modell zu finden, das die Bewertungsfunktionen maximiert.
Die ersten beiden Bewertungskriterien beziehen sich auf die Klassifikation. Bei der Regression wird der mittlere quadratische Fehler verwendet.

2.8.1 Gesamtgenauigkeit

Als Gesamtgenauigkeit oder Accuracy (*acc*) wird eine Rate der korrekt klassifizierten Beispiele bezeichnet. Dabei handelt es sich um eine relative Häufigkeit der korrekt vorhergesagten Beobachtungen.

$$\text{acc} = \frac{\text{korrekte Vorhersagen}}{\text{Alle Vorhersagen}} \tag{2.29}$$

Die Gesamtgenauigkeit ist leicht zu berechnen. Auch die Fehlerrate ε lässt sich leicht daraus ableiten $\varepsilon = 1 - acc$. Ein Problem tritt dann auf, wenn die Beobachtungen nicht gleich verteilt sind und eine Klasse überwiegend in den Beispielen auftritt, z. B. 90% der Beispiele sind vom Klasse A und der Rest vom B. Dann sagt eine höhere Gesamtgenauigkeit von 0.9 nichts über die Genauigkeit der Vorhersagen für die Klasse B aus.

2.8.2 Genauigkeit und Vollständigkeit

Bei der Genauigkeitsquote (*Precision*) wird der Anteil der richtig klassifizierten Beobachtungen einer bestimmten Klasse y_i zu allen als y_i klassifizierten Beobachtungen berechnet. Bei der Sensitivität (*recall*) ermittelt man die Rate aller richtig als y_i klassifizierten Beobachtungen zu allen Beobachtungen, die als y_i klassifiziert wurden.

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \text{ und } \text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{2.30}$$

Je höher Vollständigkeit ist, desto kleiner ist die Genauigkeit. Da die beiden Werte in Konflikt stehen, wird versucht ein harmonisches Mittel von den beiden Maßen zu finden, *F-Measure*.

$$\text{F} = 2\frac{\text{precesion} \cdot \text{recall}}{\text{precesion} + \text{recall}} \tag{2.31}$$

2.8.3 Mittlere quadratische Abweichung

Die mittlere quadratische Abweichung oder Mean Squared Error (*MSE*) ist eine Summe der quadrierten Differenzen zwischen dem tatsächlichen und vorhergesagtem Wert über alle Beobachtungen $\mathscr{T}_{test}$, die am Ende durch die Anzahl der Beobachtungen geteilt wird.

$$\text{MSE}(\mathbf{x}) = \frac{1}{|\mathscr{T}_{test}|} \sum_{i=1}^{|\mathscr{T}_{test}|} (f(x_i) - \hat{y}_i)^2 \tag{2.32}$$

Kapitel 3

Spatio-Temporal Random Fields

In diesem Kapitel wird das Spatio-Temporal Random Field (STRF) Modell vorgestellt [16]. Das STRF gehört zu den probabilistischen graphischen Modellen. Dessen Ziel ist es eine Vorhersage der fehlenden Messung zu jedem Zeitpunkt und an jedem Ort zu ermöglichen.
STRF ist ein diskretes multinomial-verteiltes generatives graphisches Modell. Ein generatives Modell hat eine gemeinsame Verteilung von beobachteten Merkmalen $\mathbf{x}$ und ein gesuchtes Label y, so dass man eine bedingte Wahrscheinlichkeit der bestimmten Wertezuordnung gegeben y ausrechnen kann.

3.1 Graphisches Modell

STRF ist ein ungerichtetes graphisches Modell, das aus mehreren Schichten $G_1, G_2, \ldots, G_T$ besteht. Jede Schicht G_t wird durch einen Graph $G_0 = (V_0, E_0)$ repräsentiert. Jeder Knoten im Graph G_0 entspricht einer Zufallsvariable X_v und beschreibt den Ort, an dem eine Messung durchgeführt wurde und die Kanten repräsentieren die Abhängigkeiten (bzw. Nachbarschaftsbeziehungen) zwischen einzelnen Sensoren.
Die räumliche Abhängigkeiten in G_0 können durch die Topologie des Straßennetzes vorgegeben oder mit Hilfe verschiedener numerischer Optimierungsverfahren bestimmt werden. Beiden Verfahren werden für das Schätzen der räumlichen Struktur verwendet. Bei dem Straßennetz wird ein Sensor mit seinen k nächsten Nachbarn verbunden, wobei bei der Regularisierung die statistischen Abhängigkeiten in Betracht gezogen werden. Der mit der Regularisierung geschätzte Graph muss nicht mit dem Straßennetz übereinstimmen. Es könnte gut sein, dass der Wert bei dem Sensor im Süden der Stadt den Wert in Osten statistisch beeinflusst. Der Graph G_0 dient als Grundlage für alle zeitlichen Schichten und wird nicht über die Zeit verändert.
Zusätzlich zu den räumlichen Abhängigkeiten werden in STRF die zeitlichen Abhängigkeiten betrachtet. Dabei werden einzelne Schichten miteinander zu einem globalen Model G verbunden. Eine Kante $E_{t;t+1} \subset V_t \times V_{t+1}$ für $t = 1, \ldots, T-1$ und $E_{t-1;T} = \emptyset$ repräsentiert die Abhängigkeiten

Das Modell setzt sich aus mehreren Schichten G_t für t=1,...,T zusammen. Der graue Bereich entspricht einer Zeitschicht L_t.

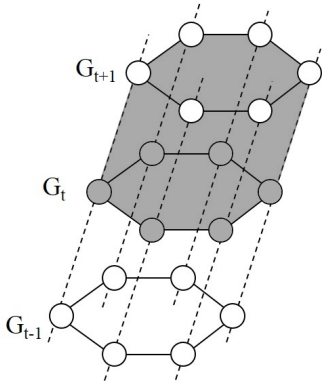

Abbildung 3.1: Globales räumlich-zeitliches Modell

zwischen benachbarten Schichten G_t und G_{t+1}. Der Zeitabschnitt zwischen t und $t+1$ kann beliebig gewählt werden.
Der komplette Graph G wird der Reihe nach $t = 1,2,\ldots,T$ aus einzelnen Schichten und Kanten, die zeitliche Abhängigkeiten repräsentieren, zusammengesetzt. G ist definiert als $G := (V,E)$ für $V := \cup_{t=1}^{T} V_t$ und $E := \cup_{t=1}^{T} \{E_t \cup E_{t;t+1}\}$.
Eine Zeitschicht L_t (s. Abbildung 3.1), Teilgraph von G, besteht aus Knoten V_t und Kanten $E_t \cup E_{t;t+1}$, für alle Zeitpunkte $t = 1,\ldots,T$. Für jede Zeitschicht wird ein binärer Teilvektor (Indikatorvektor) $\phi(X)$ und ein Parametervektor θ definiert:

$$\phi(t,\mathbf{X}) := (\phi_{v=a}(\mathbf{X}_v), \phi_{(v,w)=(a,b)}(\mathbf{X}_v,\mathbf{X}_w) \mid v \in L_t, (v,w) \in L_t, a \in \mathscr{X}_v, b \in \mathscr{X}_w) \quad (3.1)$$

$$\theta(t) := (\theta_{v=a}, \theta_{(v,w)=(a,b)} \mid v \in L_t, a \in \mathscr{X}_v, b \in \mathscr{X}_w) \quad (3.2)$$

Der Indikatorvektor merkt sich die Kanten, die vom Knoten $v \in L_t$ ausgehen und die Gewichtung dieser Kanten.
Wir gehen davon aus, dass die einzelnen Schichten sich nicht überlappen, dann kann man sie zu einer Summe von Skalarprodukten zusammenfassen.

$$\langle \phi(\mathbf{X}), \theta \rangle = \sum_{t=1}^{T} \langle \phi(t,\mathbf{X}), \theta(t) \rangle \quad (3.3)$$

Die Länge der beiden Vektoren $\phi(\mathbf{X})$ und θ hängt vom Aufbau des Graphes G_0 und den Zustandsraum $\mathscr{X}_v$ ab, so dass wir erkennen können, dass hier ein Problem der großen Dimension (Fluch der großen Dimensionen) entsteht und das Modell, so wie es hier vorgestellt wurde, noch auf keinem mobilen Gerät effizient berechnet werden kann. Ein kompakteres Modell mit der gleichen Vorhersagegenauigkeit wäre wünschenswert.

3.2 Modell Optimierung

Die Verteilungsfunktion von X_v stammt aus der Exponentialfamilie(2.1.2) und wird wie folgend definiert:

$$p_\theta(\mathbf{X} = \mathbf{x}) = \exp[\langle \theta, \phi(\mathbf{x}) \rangle - A(\theta)] \tag{3.4}$$

$A(\theta)$ ist eine log partition Funktion

$$A(\theta) := \log \int_{\mathscr{X}} \exp[\langle \theta, \phi(\mathbf{x}) \rangle] \nu(d\mathbf{x}). \tag{3.5}$$

Bei dem Lernvorgang wird jeder Parameter θ mehrmals aufgerufen, so dass es hilfreich wäre für die Berechnung die Parameter in einem Cache zu haben. Allerdings ist θ ein dicht besetzter Vektor, so dass es viel Speicher benötigen wird und der Lehrvorgang sehr viel Zeit in Anspruch nehmen wird.

3.2.1 Reparametrisierung und Regularisierung

Aus diesem Grund wird ein Teilvektor $\theta(t)$, der eine Zeitschicht repräsentiert, durch ein neuen Parametervektor $\mathbf{Z}_{\cdot\mathbf{i}} \in \mathbb{R}^{d'}$ für $i = 1, \ldots, T$ und $d' := d/T$ ersetzt. Der räumliche Graph G_0 und die Anzahl der möglichen Werten im Zustandsraum $\mathscr{X}_v$ bestimmen die Anzahl der Parameter d.

$$\theta(t) = \sum_{i=1}^{t} \frac{1}{t-i+1} \mathbf{Z}_{\cdot\mathbf{i}},\ t = 1, 2, ..., T. \tag{3.6}$$

Parametrisierung führt zur kompakten Darstellung vom Parametervektor. Dabei werden in $\mathbf{Z}_{\cdot\mathbf{i}}$ nur Differenzen zwischen zwei aufeinander folgenden Schichten gespeichert, nicht die tatsächlichen Werte. $\triangle_{(t-1):t} := \theta(t) - \theta(t-1) = \mathbf{Z}_{\cdot\mathbf{t}} - \sum_{i=1}^{t} \frac{1}{(t-i+1)(t-i)} \mathbf{Z}_{\cdot\mathbf{i}}$ repräsentiert den Einfluss von allen vorherigen (alten) Schichten auf die aktuelle Schicht, je weiter eine alte Schicht von den neuen entfernt ist, desto weniger Einfluss hat sie auf die neuen Werte.

$$\theta = \begin{bmatrix} \theta(1) \\ \theta(2) \\ \vdots \\ \theta(T) \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_{\cdot\mathbf{1}} \\ \frac{1}{2}\mathbf{Z}_{\cdot\mathbf{1}} + \mathbf{Z}_{\cdot\mathbf{2}} \\ \vdots \\ \sum_{i=1}^{T} \frac{1}{(t-i+1)(t-i)} \mathbf{Z}_{\cdot\mathbf{i}} \end{bmatrix},\ \mathbf{Z} := \begin{bmatrix} | & | & & | \\ \mathbf{Z}_{\cdot\mathbf{1}} & \mathbf{Z}_{\cdot\mathbf{2}} & \cdots & \mathbf{Z}_{\cdot\mathbf{T}} \\ | & | & & | \end{bmatrix} \tag{3.7}$$

Dabei impliziert $\mathbf{Z}_{\cdot\mathbf{t}} = 0$, dass die Verteilung vom Zeitpunkt t mit der Verteilung vom Zeitpunkt $(t-1)$ übereinstimmt. $\theta(t) = 0$ weist auf eine Gleichverteilung hin. Man erhofft sich dabei ein einfaches Modell, wenn sich die Wahrscheinlichkeiten nicht stark ändern.
Es ist zwar mehr Aufwand aus $\mathbf{Z}$ den tatsächlichen Parametervektor θ zu berechnen. Es lohnt sich trotzdem, weil die spärliche Darstellung von $\mathbf{Z}$ nicht so viel Speicherplatz braucht, so dass es in einem Cache-Speicher abgelegt werden kann und man schnell darauf zugreifen kann.
Die Regularisierung wird für die $\mathbf{Z}$ Matrix dann wie folgt definiert.

$$\|\mathbf{Z}\|_1 := \sum_{j=1}^{d'} \|\mathbf{Z}_{j\cdot}\|_1 \tag{3.8}$$

Hier werden nicht die Parameter selber bestraft, sondern die Differenzen der nacheinander folgenden Schichten. Die Ziele der einzelnen Regularisierungsverfahren wurden in Abschnitt 2.5 dargestellt.

3.2.2 Parameterschätzung

Für die Schätzung der Modellparameter mit der Maximum-Likelihood-Methode wird eine hinreichend große Stichprobe benötigt, die möglichst gut die Grundgesamtheit repräsentiert. Sei $\mathscr{D} := \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N$ ein unabhängiger und identisch verteilter Datensatz, der aus N Fallbeispiel zusammengesetzt wird. Jede multivariate Zufallsvariable $\mathbf{x}^i \in \mathscr{X}$ ist ein Set von Zufallsvariablen, die allen Knoten in im Graph G beschreiben. Für die Regularisierung definiert in 3.8 und $\lambda_1 \geq 0$ gilt dann

$$\min_{\mathbf{Z} \in \mathbb{R}^{d' \times T}} h(\mathbf{Z}) := -\ell(\theta(\mathbf{Z}); \mathscr{D}) + \lambda_1 \|\mathbf{Z}\|_1 \tag{3.9}$$

In der Arbeit [16] wurde die Elastic Net- Regularisierung eingesetzt. Bei der Durchführung der Experimente wurde aber festgestellt, dass ℓ_2 nicht so großen Einfluss auf die Modellparameter hat. Mittlerweile wird die Regularisierung mittels ℓ_1-Regularisierung durchgeführt und das FISTA-Verfahren (vgl. Abschnitt 2.6.1)) zur Lösung des Optimierungsproblems eingesetzt. Die Gleichung 3.9 wird aufgeteilt und die partielle Ableitung von der Likelihood-funktion $f(\mathbf{Z}) = -\ell(\theta(\mathbf{Z}); \mathscr{D})$ unabhängig von der ℓ_1-Regularisierung berechnet

$$\frac{\partial f(\mathbf{Z})}{\partial \mathbf{Z}_{ji}} = \sum_{t=i}^{T} \frac{[\mu(t) - \hat{\mu}(t)]_j}{t - i + 1} \tag{3.10}$$

wobei $i = 1, 2, \cdots, T$ für die Tage steht und $j = 1, 2, \cdots, d'$ von der Anzahl der Zufallsvariablen im Modell abhängt.

3.2.3 Belief Propagation

Wie es schon im Abschnitt 2.2 erwähnt wurde, stellt die Berechnung wahrscheinlichste Wertezuordnung und der Randverteilungen eine Herausforderung dar. Um dieses Problem möglichst effizient zu lösen, setzt man bei der STRF ein Inferenzalgorithmus, Belief Propagation (BP), ein. BP ist eine Message-Passing Algorithmus und kann die Randverteilungen unbeobachteten Knoten gegeben aller andere Knoten effizient und exakt berechnen, wenn der Graph gerichtet, zusammenhängend und kreisfreien ist. Auch wenn kein Baumstruktur vorliegt, könnte das Algorithmus nützlich sein, obwohl dann keine Garantie für die Korrektheit der Ergebnisse gegeben werden kann.

Die STRF Modell ist nicht kreisfrei, sodass eine spezielle Form vom BP Loopy-Belief -Propagation (LBP) eingesetzt wird. Der LBP berechnet nur eine Annäherung an die tatsächliche Randverteilung und muss nicht konvergieren.

Gegeben sei $p_\theta(\mathbf{X}) = \frac{1}{K} \prod_{(s,t) \in E} \psi_{s,t}(\mathbf{x}_s, \mathbf{x}_t)$, die faktorisierte Wahrscheinlichkeitsverteilung. Damit man die $p_\theta(\mathbf{X})$ ausrechnen kann, muss mal erst zwei NP-vollständige Probleme lösen:

- MAP: finde die wahrscheinlichste Wertezuordnung von $\mathbf{x}^* = \arg\max_{\mathbf{x}} \mathbf{p}(\mathbf{x})$
- Randverteilung: finde $p_\theta(x_i) = \sum p_\theta(\mathbf{X}_{\setminus x_i})$

Um die Randverteilung berechnen zu können muss die Wahrscheinlichkeitsfunktion auch normalisiert werden. Bei der Normalisierung wird eine Konstante $K = \sum_{\mathbf{X}} \prod_{(s,t)\in E} \psi_{s,t}(\mathbf{x}_s, \mathbf{x}_t)$ gesucht, sodass $p_\theta(\mathbf{X})$ die Eigenschaften der Wahrscheinlichkeitsverteilung erfühlt. Für die Berechnung der Randverteilungen und Normalisierung wird so genannte Summen-Produkt-Algorithmus (*sum-product-Algorithm*) eingesetzt. Dabei wird der Nachricht mit 1 initialisiert. $\mu_{\alpha \longrightarrow i}(x_i) = 1$, $\mu_{i \longrightarrow \alpha}(x_i) = 1$. dann wird ein Wurzelknoten festgelegt. Zuerst werden die Nachrichten von Blätter zu den Wurzel geschickt $\mu_{i \longrightarrow \alpha}(x_i) = \prod_{\alpha \in \mathcal{N}(i)\setminus\alpha} \mu_{\alpha \longrightarrow i}(x_i)$. Dabei terminiert der Algorithmus wenn alle Nachrichten den Wurzel erreichen. Dann wird ein Nachricht von dem Wurzel an alle Kinder geschickt. Da wir aber nicht garantieren können, dass es ein Wurzel und ein Blatt existiert, muss das Algorithmus angepasst werden, sonst wird er nicht terminieren.
MAP-Wert wird mit max-product-Algorithmus berechnet. Hier wird ein Wert gesucht, der das Produkt maximiert.

Kapitel 4

Poisson Dependency Network

Ein alternativer Ansatz zu STRF, der den Verkehrsfluss modellieren und vorhersagen kann, ist Poisson Dependecy Network (PDN) [8]. PDN ist ein graphisches approximatives multivariates poisson-verteiltes Modell. Die Poisson Verteilung ist besonders gut für die Bestimmung der Häufigkeiten eines Ereignisses über eine bestimmte Zeit geeignet und ist für die natürliche Zahlen größer gleich 0 definiert (vgl. Abschnitt 2.1.2).

4.1 Modell

PDN Modell ist ein Tupel von bedingte lokalen Poisson Verteilungen, welche jeweils eine Zufallsvariable gegeben alle andere repräsentiert. Das globale Modell wird als Produkt von Regressionsmodellen, gestartet von eine log-linearen Modell, schrittweise aufgebaut. Ein diskriminatives Modell ist ein globales Modell, welches aus vielen lokalen zusammengesetzt wird. D.h. jedes $p(y|x)$ wird einzeln trainiert. Wenn wir die räumliche Struktur betrachten, würde man für jeden Ort ein Modell aufbauen müssen, so wie es auch der Fall in PDN ist.
PDN ist ein Tupel (G,P). Wobei $G = (V,E)$ ein gerichteter Graph ist, bei dem jeder Knoten in V eine Zufallsvariable in $\mathbf{X} = x_i$, $x_i \in \mathbf{N}$ repräsentieren, und P die Wahrscheinlichkeitsverteilung ist. Die Abhängigkeitsstruktur der beobachteten Werte in PDN wird nicht explizit mittels Regularisierung geschätzt oder vorher vorgegeben, sondern über die Abhängigkeiten zwischen einzelnen Zufallsvariablen aus dem Modell abgeleitet werden kann. Im Fall von Regressionsbäumen wird die Struktur auf der Basis von Entscheidungsbäumen bestimmt.
Für $\mathbf{X} \setminus \{X_i, X_j\}$ wird eine Kurzschreibweise $\mathbf{X}_{\setminus i,j}$ verwendet. Sei $pa_i(x_i) = \{x_j \in \mathbf{X}_{\setminus i} \mid (x_j, x_i) \in \mathbf{E}\}$ die Menge der Elternknoten von X_i. Die Wahrscheinlichkeitsverteilung wird dann für jede Zufallsvariable folgend definiert:

$$p(x_i \mid \mathbf{x}_{\setminus i}) = p(x_i \mid \mathbf{pa}_i) = \frac{\lambda_i(\mathbf{x}_{\setminus i})^{x_i}}{x_i!} e^{-\lambda_i(\mathbf{x}_{\setminus i})}. \tag{4.1}$$

$\lambda_i(\mathbf{x}_{\setminus i})$ ist eine Funktion, die den Wert von λ_i in Abhängigkeit von allen andren Variablen $\mathbf{X}_{\setminus i}$ bestimmt. Einfachheitshalber wird $\lambda_i(\mathbf{x}_{\setminus i})$ weiter nur als λ_i aufgefasst.
Gemäß [8] wird die gemeinsame Verteilung als Produkt der lokalen poisson-verteilten Modellen definiert

$$p(\mathbf{x}) := \prod_{x_i \in \mathbf{x}} p(x_i \mid \mathbf{x}_{\setminus i}) = \prod_{x_i \in \mathbf{x}} \frac{\lambda_i^{x_i}}{x_i!} e^{-\lambda_i} \tag{4.2}$$

Es sei hier aufmerksam gemacht, dass die Konsistenz von diesen gemeinsamen Verteilung nicht garantiert werden kann. $\prod_{x_i \in \mathbf{x}} p(x_i \mid \mathbf{x}_{\setminus i})$ ist nicht normalisiert. Die gemeinsame Verteilung ist aber erst dann notwendig, wenn der Datensatz unvollständig ist und man den Gibbs Sampling für die Schätzung der gemeinsame Verteilung einsetzen soll.
PDN generalisiert die Dependency Networks für multivariate Poisson-Verteilungen. In der [8] wurden drei unterschiedliche Vorgehensweise vorgestellt, wie man λ modellieren kann. Man könnte das Lernverfahren mit einem konstanten Werten für λ_i's starten, die λ_i's mit der log-linearen Modell oder Poisson Regressionsbäume bestimmen. In dieser Arbeit wird die log-lineare Modell ausführlich erklärt. Für die weiter Ansätze sei hier auf [8] verwiesen.

4.1.1 Log-lineares Poisson Modell

Die Poisson Regression ist ein Spezialfall von verallgemeinerten linearen Modellen. Das verallgemeinerte lineare Modell [13] (GLM) ist ein Modell in der klassischen linearen Regressionsanalyse. Die Grundidee ist eine Funktion zu finden, die den Einfluss der beobachteten Werten auf die Zielvariablen möglichst präzise abbildet (vgl. Abschnitt 2.2). Dabei stammt die Verteilung der Zielvariable aus der Exponentialfamilie, hier ist die Variable poisson-verteilt. Die Erwartungswert λ_i wird über eine Link-Funktion, log, linear modelliert.

$$\begin{aligned} \log(\lambda_i) &= \beta_i + \sum_{j \neq i} \beta_{ij} x_j \\ E[X_i] = \lambda_i &= \exp\left(\beta_i + \sum_{j \neq i} \beta_{ij} x_j\right) \end{aligned} \tag{4.3}$$

4.2 Modell Optimierung

Die log-lineare Modelle neigen öfters dazu die vollständige Modelle zu erzeugen. Beim vollständigen Modell handelt es sich um ein Modell, bei dem jede Zufallsvariable von allen anderen Zufallsvariablen beeinflusst wird. Bei dem Graph, wird dann jeder Knoten mit jeden anderen Knoten verbunden. Diese Vorgehensweise kann zur Überanpassung führen.

4.2.1 Regularisierung

PDN wird am Anfang als ein vollständiges Modell initialisiert. Deswegen ist es wichtig die überflüssige Kanten zu entfernen, um die Komplexität des Modells zu verringern. Die Modellparametern werden wegen der Vergleichbarkeit von beiden Modellen, wie auch bei STRF, mittels ℓ_1-Regularisierung angepasst.

4.2.2 Parameterschätzung

Die Schätzung der Modellparameter (λ) wird über die Maximum-Likelihood-Methode vorgenommen.

$$\ell(\beta_i, \beta_{ij}) = \log \prod_{k=1}^{m} p\left(x_i^{(k)} \mid x_{\setminus i}^{(k)}\right) = \sum_{k=1}^{m} \log p\left(x_i^{(k)} \mid x_{\setminus i}^{(k)}\right) \tag{4.4}$$

Hier wird nach Parametern gesucht, die log-Likelihood Funktion maximieren. Um ein Optimum zu finden, muss die Funktion abgeleitet werden. Die partielle Ableitung

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta_i} &= \tfrac{\partial}{\partial \beta_i} \textstyle\sum_{k=1}^{m} x_i^{(k)} \log\left(\lambda_i^{(k)}\right) - log\left(x_i^{(k)}!\right) - \lambda_i^{(k)} &= \sum_{k=1}^{m} x_i^{(k)} - \lambda_i^{(k)} \\
\frac{\partial \ell}{\partial \beta_{ij}} &= \tfrac{\partial}{\partial \beta_{ij}} \textstyle\sum_{k=1}^{m} x_i^{(k)} \log\left(\lambda_i^{(k)}\right) - log\left(x_i^{(k)}!\right) - \lambda_i^{(k)} &= \sum_{k=1}^{m} \left(x_i^{(k)} - \lambda_i^{(k)}\right) x_j^{(k)}
\end{aligned}$$

Um diese Optimierungsproblem zu lösen, wird auch hier ein Gradientenverfahren angewendet. In Fall vom Poisson Regression wird iteratively-re-weighted least squares (IRLS) Algorithm oft eingesetzt. Um die bessere Vergleichbarkeit der beiden Verfahren STRF und PDN zu ermöglichen, wurde in dieser Arbeit anstatt IRLS das proximative Gradientenverfahren durchgeführt. An dieser Stelle wird für weiteren Informationen über IRLS auf [7] verwiesen.

4.2.3 Vorhersage

Der Einfachheit halber wird angenommen, dass der Datensatz vollständig ist, d.h. keine unbeobachtete Werte enthält.
Um die Vorhersagen über fehlende Werte treffen zu können, wird die probabilistische Inferenz berechnet. Die MAP Inferenz findet die wahrscheinlichste Werte (vgl. Abschnitt **??**). In der Poisson Modell entspricht der MAP-Wert dem λ_i.
Wenn der Datensatz unvollständig ist und man die Eltern von X_i auch nicht kennt, wird für die Berechnung der Randverteilungen ein weiteres Inferenzalgorithmus, Gibbs Sampling, verwendet. Der Gibs Sampler ist besonders für PDN geeignet, weil er auch die Schätzung für alle Werte einzeln ausführt. Man initialisiert den unbekannten Parameter x mit einem beliebigen Wert und wertet das komplette Modell für diesen Wert aus. Nach diesem Schritt werden die Parameter anhand diesen zufällig gewählten Wertes bestimmt. Danach zieht man einen weiteren Wert zufällig heraus und setzt wieder für x ein. Dieser Vorgang wird mehrmals wiederholt, bis man die maximale Anzahl an Schritten erreicht hat oder der Fehler klein genug ist.

Da im Rahmen dieser Arbeit werden die beiden Ansätze STRF und PDN auf einem vollständigen Datensatz modelliert, wurde die Vorgehensweise von Gibbs Sampler nur kurz erklärt, eine ausführliche Beschreibung des Algorithmus kann man in [11] finden.

Kapitel 5

Datenvorverarbeitung

Daten bilden die Grundlage für alle Analysen. Es kommt aber selten vor, dass man die Rohdaten sofort für die Analyse verwenden kann. Die Daten enthalten fehlende Werte, sind fehlerhaft oder weisen die Inkonsistenzen auf. Unbereinigte Daten können die Analyseergebnisse verfälschen, deswegen ist die Datenvorverarbeitung (*Preprocessing*) so wichtig.
Die transformierte Datensätze, Detektoren Tabelle, sowie für die *Preprocessing* verwendeten Java Quellcode sind auf dem DVD im Anhang A enthalten. Der originalen Datensatz ist öffentlich zugänglich [1].

5.1 Aufbereitung der Daten

Grundlage dieser Arbeit sind Zählschleifendaten der Dubliner Verkehrsamtes, die mit den Sydney Coordinated Adaptive Traffic System(SCATS) Systemen zwischen Januar und April 2013 in Dublin, Irland erfasst wurden. Die original Daten liegen in Form von 4 Tabellen vor und sind ca. 9GB groß. Jede der Tabellen enthält die Sensormessungen der ausgewählten Stadtteilen Nord,Süd,West,Ost. Der Verkehrsfluss wurde in der Daten jede Minute dokumentiert. Die Messungen wurden zusammen mit der Datei detectors.csv (Anhang A), die die Informationen über die Standorte der Sensoren enthält, ausgewertet.
Alle Daten wurden in zwei Tabellen mit SQLite Version 3.8.2 2013-12-06[2] zusammengefasst und mittels der Projektion, wurden nur die benötigten Spalten ausgewählt.
Insgesamt enthält der Datensatz die Messwerte von 2367 Sensoren an 135 Tagen.

5.1.1 Behandlung von Inkonsistenzen

Bei der Inkonsistenz handelt es sich um die Widersprüchen in den Daten. In dem originalen Datensatz hatten manche Sensoren zwei Messwerte pro Minute. Man könnte es nicht eindeutig identifizieren welcher davon richtig war. Aus diesem Grund wurde bei der Aggregation der Daten keine

[1] `https://data.dublinked.ie/dataset/scats-dcc-jan2013`
[2] `https://www.sqlite.org/index.html`

Durchschnittswerte genommen. Alle Tabellen mit den Messungen wurden zusammengeführt und die Messungen zu 15-Minuten Blöcken zusammengefasst, so dass am Tag 96 $(24 \cdot 60/15)$ Zeitabschnitte am Ende entstanden sind. Als Repräsentant des 15-Minuten-Blocks wurde ein maximaler Wert genommen, somit wurde das Wost-Case Szenario betrachtet.

5.1.2 Behandlung fehlender Werte

Der originale Datensatz wies einige fehlende Werte auf. Es gibt mehrere Möglichkeiten, wie man in der Statistik mit fehlenden Werten umgeht. Man ersetzt die fehlende Werte durch einen anderen plausiblen Wert, zum Beispiel Erwartungswert, man lässt die fehlende Werte weg oder man wählt ein Verfahren, der mit den fehlenden Werten zurecht kommt. Da in dieser Arbeit der Maximum-Likelihood-Schätzer eingesetzt wird, wird ein lückenloser Datensatz vorausgesetzt.
Während der Vorbereitung wurde festgestellt, dass in den Daten komplette Minutenblöcke fehlen, so dass kein einziger Sensor da war, bei dem alle Werte über alle Tagen vorhanden waren. Die 2 Tage wurden aus der Datentabelle rausgefiltert. Im weiteren Verlauf wurden auch 154 Sensoren entfernt, bei denen nach der Entfernung der beiden Tagen noch Messwerte fehlten.
Leider fehlten auch bei den 20 *Intersections* (98 Sensoren) die Informationen über Ihren Standort, so dass diese *Intersections* zwar bei der Analyse berücksichtigt wurden, aber auf der Karte nicht abgebildet werden konnten.

5.1.3 Behandlung konstanter Werte

Bei der Vorbereitung der Tabellen für die PDN Analyse ist aufgefallen, dass es auch Sensoren gibt, deren Werte über alle Tagen und alle Zeitabschnitte konstant 0 sind. Diese 107 Sensoren wurden auch entfernt.

Von 2367 Sensoren und 135 Tagen, sind nach der Vorverarbeitung der Daten die Messwerte von 2106 Sensoren und 133 Tagen geblieben. Alle Daten wurde mit Java Code aus einer csv-Datei ausgelesen, vorbereitet und wieder in eine csv-Datei geschrieben.

5.2 Transformation der Daten

Nachdem alle fehlende Werte und Inkonsistenzen behandelt wurden, müssen die Daten in eine bestimmte Form, die die Analyseverfahren erwartet, gebracht werden. Alle Transformationen wurden mit Java durchgeführt. Die Darstellungsformen unterscheiden sich in STRF und PDN. Für die Analyse mit STRF wurden die Daten wie in der Tabelle 5.1 umgewandelt.
Wobei hier s_i der i-te Sensor ist und t_j der Zeitabschnitt, zudem eine Messung durchgeführt wurde. Jede Zeile in der STRF-Tabelle enthält die Messungen aller Sensoren an einem Tag, sortiert nach den Namen der Sensoren und den Zeitabschnitten. Die Reihenfolge der Sensoren ist fest, aber beliebig. In dieser Arbeit wurden die Sensoren numerisch sortiert.

	$s_1 - t_1$	$s_2 - t_1$	$\cdots$	$s_{2106} - t_1$	$s_1 - t_2$	$s_2 - t_2$	$\cdots$	$s_{2106} - t_2$	$\cdots$	$s_{2106} - t_{96}$
Tag 1	$96 - inf$	$46 - 68$		$0 - 11$	$46 - 68$	$0 - 11$		$0 - 11$		$0 - 11$
$\vdots$										
Tag 133	$11 - 27$	$0 - 11$		$27 - 46$	$11 - 27$	$0 - 11$		$0 - 11$		$11 - 27$

Tabelle 5.1: STRF Daten. Diskretisierung mit 6 Klassen, nach der Häufigkeit.

	$s_1 - t_{43}$	$s_2 - t_{43}$	$\cdots$	$s_n - t_{43}$
Tag 1	31	27		30
$\vdots$				
Tag 133	85	99		0

Tabelle 5.2: PDN Daten. Zeitabschnitt 43.

Für ein PDN Modell wurden die Daten auf 96 Tabellen aufgeteilt. Jede Tabelle entspricht einem bestimmten Zeitabschnitt t_j und sieht wie die Tabelle 5.2 aus.

5.2.1 Diskretisierung

Der Wertebereich der Zufallsvariablen in STRF ist nominal. Aus diesem Grund müssen die originale numerische Werte in nominale umgewandelt werden. Dieser Schritt kann bereits vor der Transformation der Daten oder direkt danach durchgeführt werden. Für die Diskretisierung der Daten wurde Rapid Miner[3] benutzt. Dabei wurde ein Sample als Zufallsstichprobe von 25% der Daten gezogen und die Werte in 4,6 und 8 Gruppen unterteilt (vgl. 5.3). Die Wertebereiche wurden nach der ähnlichen Anzahl der Werten (*Frequency*) und nach der gleichen Intervallbreite (*Binning*) unterteilt. Nach der Berechnung der Intervalle, wurden die nummerische Werte in den vorbereiteten Datensatz durch zugehörige Intervalle mit Java ersetzt.
Weniger Klassen ermöglichen bessere Vorhersagegüte, aber haben kleinere Aussagekraft.

Binning

Die Werte werden in k-gleich großen Intervallen unterteilt. Die Breite des Intervall wird als $w = (\max - \min)/k$ berechnet. Wobei k Anzahl der Intervalle ist. Und die Intervallgrenzen setzen sich dann als $\min + w, \min + 2w, \cdots, min + (k-1)w$ zusammen.

Frequency

Dieser Algorithmus teilt die Daten in k Gruppen auf, sodass die Daten ungefähr gleich verteilt über die Gruppen sind.

[3] `https://rapidminer.com/`

4 Bins	4 Freq	6 Bins	6 Freq	8 Bins	8 Freq
63	19	42	11	32	7
125	46	84	27	63	19
188	81	125	46	94	32
250	250	167	68	125	46
		209	96	157	62
		250	250	188	81
				219	104
				250	250

Tabelle 5.3: Obere Grenzen der Intervalle bei der Diskritizierungsverfahren

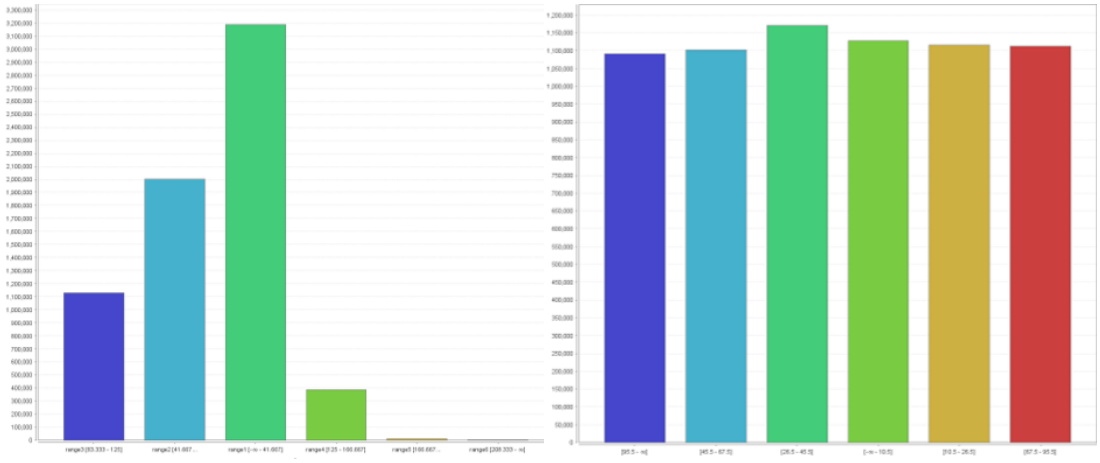

Abbildung 5.1: Diskretisierung der Daten mit 6 Intervallen nach der Intervallbreite und Häufigkeit der Daten. Erstellt mit Rapid Miner.

5.3 Graphische Darstellung der Daten

Wie schon im Kapitel 3 erwähnt wurde, um die Daten mit STRF modellieren zu können, ist eine räumliche graphische Struktur als Eingabe erforderlich. Man kann es mit der Regularisierung (2.5) schätzen lassen, anhand der Informationen aus der Datenbank modellieren oder auch das Expertenwissen benutzen.

Die Regularisierungsmethode wurde bereit, Abschnitt 2.5 vorgestellt. Die Informationen über die räumliche Struktur aus dem Datenbank (detectors.csv) waren unvollständig (s. 5.2), so dass nicht alle Zusammenhänge abgebildet werden konnten.

Da es sich in den Daten um die SCTATS-Sensoren handelt, können wir in dem Fall das Straßennetz Topologie ausnutzen, um die einzelne Sensoren miteinander zu einem Graph zu verbinden. Für die graphische Darstellung der Daten wurden einfachheitshalber die Sensoren zu den Kreuzungspunkten (*Intersection*) zusammengefasst. In jeder *Intersection* sind alle Sensoren untereinander verbunden. Wenn zwei Intersections benachbart sind, dann sind alle Sensoren von

Die Standorte der SCATS Sensoren werden mit den Punkten markiert. Dublin, Ireland. Google Maps.

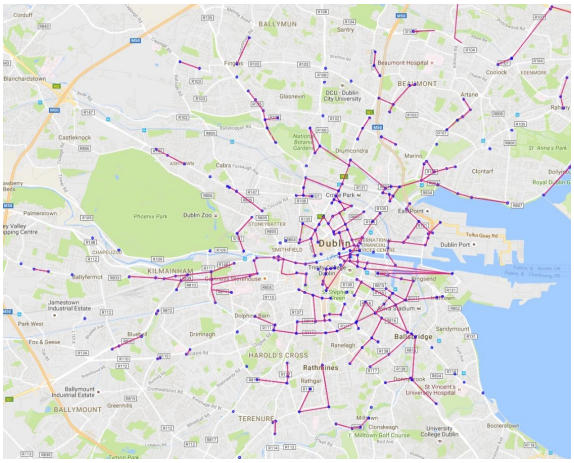

Abbildung 5.2: Räumliche Struktur abgeleitet von den Daten aus dem Datensatz.

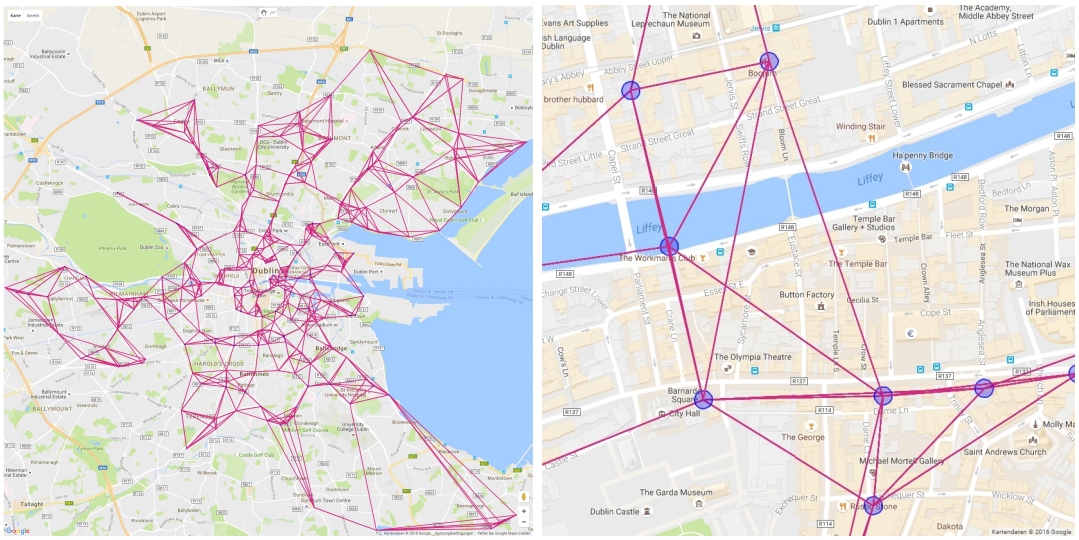

Abbildung 5.3: Ein Graph zur Veranschaulichung der Räumlichen Abhängigkeiten in STRF. 5-nächsten Nachbarn berechnet mit Haversine Distanz.

Intersection A mit allen Sensoren von *Intersection* B verbunden. Es ist wichtig zu betonen, dass diese Methode nur bei der Erstellung der Graphen für STRF mit Haversine Distanz und Google API benutzt wurde und dass sowohl bei der graphischen Struktur mittels Regularisierung, als auch bei den Analysen alle Sensoren einzeln betrachtet wurden. Am Ende wurden die Nachbarschaften in einer Erreichbarkeitsmatrix zusammengefasst. Die Matrix ist 2106*2106 groß , wobei jede Zeile die Nachbarn von einem bestimmten Sensor darstellt. Bei der räumlichen Struktur, die für STRF erforderlich ist, wurden die nächsten 5 Nachbarn betrachtet, die anhand unterschiedlichen Algorithmen ermittelt wurden. Zuerst wurde die Entfernung zwischen einzelnen Sensoren mit

der Haversine Distanz berechnet. Bei der Haversine Distanz ermittelt die Luftlinie zwischen zwei Punkten anhand ihres GPS-Koordinaten nach folgende Formel

$$\begin{aligned} \text{latDist} &= (\text{lat1} - \text{lat2})\frac{\pi}{180} \\ \text{lonDist} &= (\text{lon1} - \text{lon2})\frac{\pi}{180} \\ \text{a} &= \sin^2\left(\frac{\text{latDist}}{2}\right) + \cos\left(\frac{\text{lat2}\,\pi}{180}\right)\cos\left(\frac{\text{lat1}\,\pi}{180}\right)\sin^2\left(\frac{\text{lonDist}}{2}\right) \\ \text{distance} &= 2r\arcsin(\sqrt{a}). \end{aligned} \tag{5.1}$$

Ein Nachteil von Haversine Distanz ist es, dass das Straßennetz hier nicht im Betracht gezogen wird, so dass die Verbindungen entstehen können, die im normalen Straßennetz viel länger sind, als mit Haversine Distanz berechnet. Ein Beispiel zwei gegenüber liegender Punkte zwischen denen eine Fluss verläuft (vgl. Abb. 5.3 rechts). So wird ein gegenüber liegende Sensor als Nachbarn angesehen, obwohl die nächste Brücke vielleicht erst in 5 km ist. Da wir nur die ersten 5 Nachbarn betrachten, können die tatsächlichen Nachbarn in der Situation unbeachtet bleiben.
Als nächster Algorithmus wurde das Routen Algorithmus von Google Maps API's[4] eingesetzt. Bei Google gibt es die Möglichkeit, die Entfernung unter Berücksichtigung des Straßennetzes zu berechnen. Man kann die Entfernung oder die Reisedauer ohne Verkehr zwischen mehreren Punkten anhand der GPS ausrechnen lassen. Eine Google Anfrage [5] kann man über den Browser direkt eingeben oder eine Anfrage mit dem Java-Client zum Google-Server verschicken. Da man die Nachbarn von 307 Intersections bestimmen soll, bittet sich eine automatische Lösung an. Auf Grund der begrenzten Kontingente bei Google Distancematrix API, wurden zuerst die 10 nächsten Nachbarn mit Haversine Distanz vorberechnet und dann an Google-Server verschickt, um dann die 5 nächsten Nachbarn anhand der Entfernung und der Dauer (Abb. 5.4) unter Berücksichtigung des Straßennetzes zu bestimmen.

Der Nachteil von den räumlichen Strukturen, erstellt mit k-NN Algorithmus, dass hier auch die Abhängigkeiten abgebildet wurden, die in den Straßennetzen nicht vorhanden sind.
Auch eine neue API von Google, die Google Maps Roads API[6], wurde getestet. Es ist ein neuer Webdienst, der ein Pfad zurückgibt, der die Straßen Netzwerk abbildet. Die eingegebene Koordinaten werden an den Straßen angeheftet, auf der ein Fahrzeug am wahrscheinlichsten gefahren wäre. Meine Idee war einen Graph mit der API aufzubauen, der das aktuelle Straßennetz vom Dublin abbildet. Allerdings merkt man, dass der Dienst neu ist und funktioniert in der Realität noch nicht ganz.

[4] `https://developers.google.com/maps/?hl=de`

[5] `https://maps.googleapis.com/maps/api/distancematrix/json?key='YOURAPIKEY'&origins=53.38241958618164,-6.24354887008667&destinations=53.38094711303711,-6.238108158111572|53.37934112548828,-6.239467144012451`

[6] `https://developers.google.com/maps/documentation/roads/intro?hl=de`

```
1: GeoApiContext context = new GeoApiContext();
2: context.setQueryRateLimit(5);
3: context.setApiKey('YOUR API KEY');
4: DistanceMatrixApiRequest request = DistanceMatrixApi.getDistanceMatrix(context, origins,
   destination);
5: DistanceMatrix matrix = request.await();
6: DistanceMatrixRow[] rows = matrix.rows;
7: for DistanceMatrixRow row : rows do
8:     for DistanceMatrixElement elem : row.elements do
9:         dist.add(elem.distance.inMeters);
10:        dur.add(elem.duration.inSeconds);
11: distance.put(n.get(0), dist);
12: duration.put(n.get(0), dur);
```

Algorithmus 5.1: Distance Matrix Api

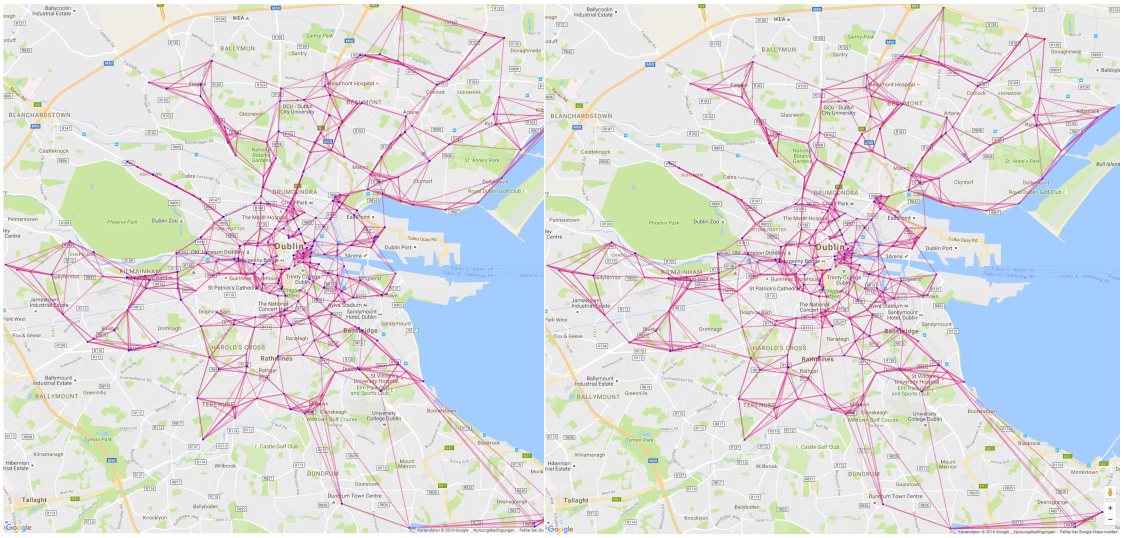

Abbildung 5.4: Ein Graph zur Veranschaulichung der Räumlichen Abhängigkeiten in STRF. 5 nächsten Nachbarn berechnet mit Google Maps API's. Das linke Bild zeigt die 5 nächsten Nachbarn mit der kürzesten Reisezeit, ohne Verkehr, das rechte Bild mit der kürzesten Entfernung.

Kapitel 6

Experimente

In diesem Abschnitt werden die durchgeführten Studien, sowie deren Ergebnisse beschrieben. Nach der Vorverarbeitung der Daten(vgl. Kapitel 5), werden die Modelle von diesen Daten gelernt und die Zusammenhänge ermittelt. Für die Durchführung der Lernvorgänge wurde auf den vorbereiteten Datensatz eine 5-fache Kreuzvalidierung durchgeführt. Das Ergebnis wird für die Vorhersagen auf anderen Daten getestet, bei denen einige Merkmale fehlen.
Die verwendete Software können Sie unter folgende Adresse finden
`http://www-ai.cs.uni-dortmund.de/PERSONAL/piatkowski.html`.

6.1 Aufbau der Experimente

Die beiden Modelle STRF (3) und PDN (4) wurden zum Modellieren der Verkehrsflusses eingesetzt. Die überwachten Lernverfahren mit dem STRF Modell wurden mit unterschiedlichen Trainingskonfigurationen durchgeführt. Das PDN Modell wurde stets mit den gleichen Parametern ausgeführt.

6.1.1 STRF

Das STRF Modell wurde an 9 verschiedenen räumlichen Strukturen trainiert. Bei der ersten drei Strukturen wurden die 5 nächsten Nachbarn (5-NN) mit Hilfe unterschiedlichen Algorithmen ermittelt (vgl. Abschnitt 5.3) und daraus die Erreichbarkeitsmatrizen abgeleitet, die dann als Vorlage für den räumliche Abhängigkeitsstruktur G_0 dienten. Die anderen räumlichen Strukturen wurden mit der ℓ_1-Regularisierung geschätzt.
Die räumliche Abhängigkeiten wurden mit folgenden Parameter gelernt:

```
./gen_stats --time-slices 96 --collapse 1 DIR/TRAIN MOD_DIR/TRAIN.mod
./cuestimate --rst 1 --eps 0.0001 --lam 0.1 --itr 50 --dev 2 --eta 0.1 --opt FISTAL1 --mod STRF
MOD_DIR/TRAIN.modG
```

	Haversine	Reisezeit	Entfernung	Reg. Bin (∅)	Reg. Freq (∅)
Anzahl Kanten	40814	40585	40406	15665	1243

Tabelle 6.1: Anzahl der Kanten in Graphen, erstellt mit verschiedenen Algorithmen

Anders als beim Modellieren der zeitlichen Abhängigkeiten, wo der ganzen Tag als ein globales Modell angesehen wird, wurden hier die 96 Zeitabschnitten als separate Modelle betrachtet (collapse=1). Die Abhängigkeiten zwischen einzelnen Schichten werden hier nicht berücksichtigt, weil die räumliche Struktur bei STRF sich nicht über die Zeit verändert.
Das Erstellen von anderen Graphen wurde grundlegend im letzten Kapitel 5.3 beschrieben.
Für die Modellierung der Daten mit dem STRF, wurden die Daten auf 6 unterschiedlichen Arten diskretisiert (vgl. Abschnitt 5.2.1), so dass insgesamt 24 (4 räumlichen Strukturen $\times$ 6 Diskretisierungen) Konfigurationen bei der STRF trainiert und getestet werden sollte.
Ausgehend von der räumlichen Struktur wurde pro Tag ein globales Modell mit 96 Zeitschichten aufgebaut. Die die 5-Fache Kreuzvalidierung musste bei STRF-Experimenten mit selbst erstellten Graphen (k-NN) auf Grund der großen Anzahl von Parametern ausgelassen werden, der Lernvorgang hat zu lange gedauert. Die Graphen haben wesentlich mehr Kanten, als die Graphen, bei denen die räumliche Struktur mit ℓ_1-Regularisierung gelernt wurde(vgl. 6.1). Auch die Trainingskonfiguration mit der 8 Klassen und k-NN Graphen wurde nicht mehr durchgeführt. Die Modelle für die drei k-NN Graphen wurden nur auf dem ersten die erste Menge trainiert und getestet. Außerdem wurden alle STRF Modelle nur für den Zeitabschnitt zwischen 12 : 00 Uhr und 18 : 30 trainiert.
Das zeitliche STRF Modell wurde mit folgende Parametern trainiert:

```
./gen_stats --time-slices 96 --collapse 0 --graph MOD_DIR/TRAIN.G DATA_DIR/TRAIN MOD_DIR/TRAIN.modG
./cuestimate --rst 1 --eps 0.0001 --lam 0.1 --itr 50 --dev 2 --eta 0.1 --opt FISTAL1 --mod STRF MOD_DIR/TRAIN.modG
```

```
./custc --dev 2 MOD_DIR/TRAIN.modG DATA_DIR/TEST MOD_DIR/TEST.confusion
```

Wobei bei den k-NN Graphen mit der 4-fachen Diskretisierung nach der Häufigkeiten, 6 und 8 Intervallen für beiden Diskretisierungsarten wurde die Annäherung bei der Optimierungsverfahren FISTA nur 10 Mal ausgeführt. Bei allen anderen Modellen war es 50.

6.1.2 PDN

Bei dem PDN Modell war keine räumliche Struktur erforderlich. Für jeden Zeitabschnitt wurde ein separates Modell gelernt. Für das Modellieren der zeitlichen Abhängigkeiten wurden zwei aufeinander folgenden Zeitabschnitten zusammengesetzt. Die Modelle für eine Zeitschicht $t + 1$ wurden

anhand der Daten von Zeitschicht t trainiert. Hier setze sich der Datensatz aus 96 Beobachtungen $\mathscr{D} = \mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^9 6$. Wobei jede Beobachtung $\mathbf{X}^i$ die Messungen zu einem Bestimmten Zeitpunkt t über alle Tage für alle Sensoren enthielt. Die Eingabe für den Lernvorgang sah dann wie folgt aus:

$$\begin{aligned} \mathscr{D} &= \{(\mathbf{x}^1, \mathbf{y}_1^2), (\mathbf{x}^1, y_2^2), \cdots, (\mathbf{x}^1, y_{2106}^2)\} \\ \mathbf{x}^t &= \{x_1^{(t)}, x_2^{(t)}, \cdots, x_{2106}^{(t)}\}, y_i = x_i^{(t+1)}. \end{aligned} \tag{6.1}$$

Insgesamt wurden 95 Modellen mit PDN erstellt. Die Modelle sind für 150 FISTA Iterationen mit $\lambda = 100$ und Schrittweite $\gamma = 10^{-9}$ gelaufen.

./cuglm **--lam 100 --itr 150** --dev 2 DIR/trX DIR/trY DIR/teX DIR/teY

6.2 Ergebnisse

Während des Testphase wurden die gelernte Modelle mit vorher ausgelassenen Testmengen T_test überprüft. Die Tests helfen uns zu erkennen, ob die gefundenen Regeln auch für die Elemente der Grundgesamtheit gültig sind. Die Ergebnisse der Testphase kann man in den untenstehenden Tabellen sehen. Zum Vergleich der beiden Ansätzen werden die im Abschnitt 2.8 vorgestellten Gütemaßen verwendet.

6.2.1 Räumliche Struktur

In allen graphischen Darstellungen wurden die Sensoren der besseren Übersicht halber zu *Intersections* zusammen gefasst.
Eigentlich ist PDN ein gerichtetes Modell, die Kantenrichtungen wurden aber der besseren Übersicht halber bei dem Graph ausgelassen. Das Modell enthält insgesamt 816029 Kanten, so dass es sehr schwierig ist, die Richtung dieser Kanten darzustellen (s. 6.1). Trotz der Reduzierung der Komplexität mit ℓ_1-Regularisierung und FISTA Optimierung, ist PDN Modell, vor allem im Vergleich zu den anderen STRF-Modellen (s. 6.1) sehr komplex. Wie man anhand der folgenden Graphiken erkennen kann, ist das STRF Modell sehr spärlich. Die meisten Abhängigkeiten, im Vergleich zu den anderen k-NN Modellen, hat das Modell, dessen Daten in vier gleich breiten Abschnitten aufgeteilt wurden (*4 Binning*), erkannt. Dieses Modell hat insgesamt 17646 Kanten.

6.2.2 Gütemaße

Das Ziel diesen Studien die zukünftige Auslastung jeden Straßenabschnittes anhand der beobachteten Daten möglichst präzise vorhersagen zu können. Trotz der spärliche Struktur, sind die

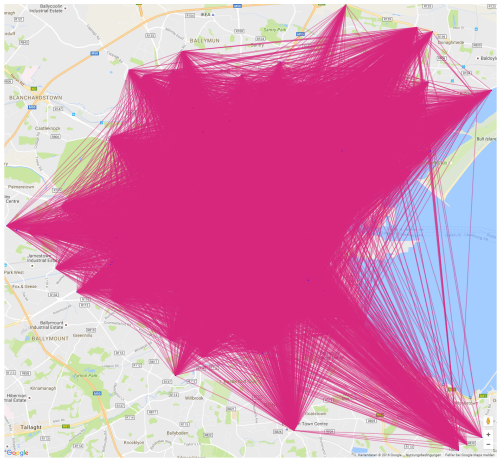

Abbildung 6.1: Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, abgeleitet von dem PDN Modell.

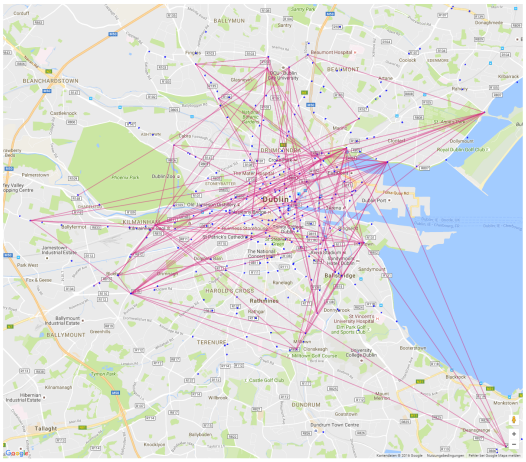

Abbildung 6.2: Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. Diskretizierte Daten nach Intervallbreite in 4 Intervallen.

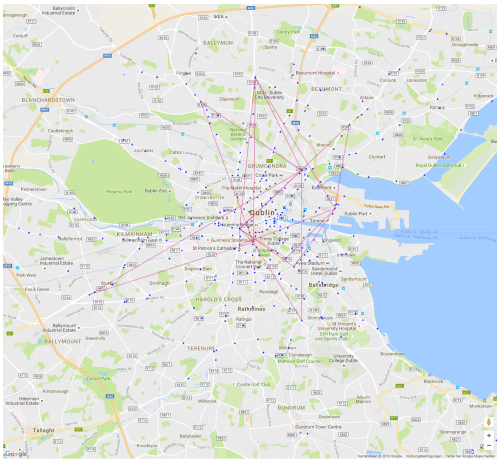

Abbildung 6.3: Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. Diskretizierte Daten nach Häufigkeit in 4 Intervallen.

F-Measures bei der mit der Regularisierung geschätzten Strukturen besser. Vor allem bei der 6-fachen Diskritisierung mit Binnig, liegt der Unterschied über 20%. Das könnte darauf zurückzuführen sein, dass die k-NN Modelle nur mit 10, anstatt 50 FISTA Iterationen trainiert wurden, obwohl die Modelle viel mehr Parameter haben. Die Ergebnisse der unterschiedlichen k-NN Modellen sehen sehr ähnlich aus. Die Größte Fehlerrate liegt meistens bei den Intervallen deren untere Grenze über 125 liegt. Die Werte größer 125 kommen in der vorliegenden Beobachtungen vergleichsweise nur selten vor, so dass die letzten Abschnitten bei Binning nur dünn besiedelt sind (vgl. 5.1). Da bei *Frequency* die Daten aus dem Datensatz über die Intervalle gleich verteilt werden, haben diese Modelle auch bessere Vorhersagequote für die Bereiche.

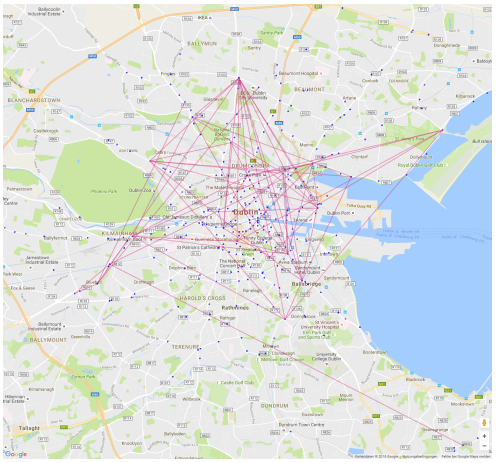

Abbildung 6.4: Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. Diskretizierte Daten nach Intervallbreite in 6 Intervallen.

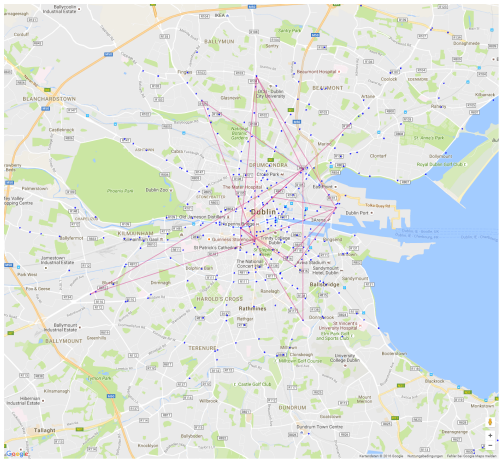

Abbildung 6.5: Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. Diskretizierte Daten nach Häufigkeit in 6 Intervallen.

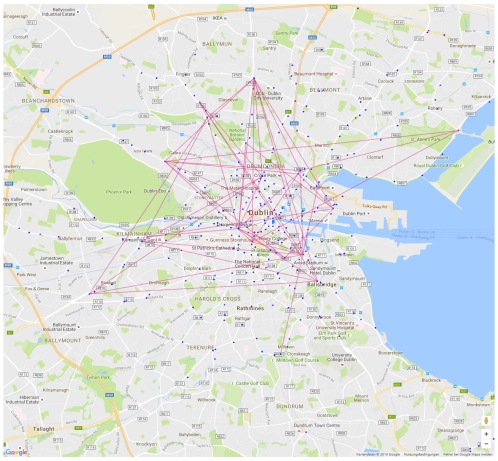

Abbildung 6.6: Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. Diskretizierte Daten nach der Intervallbreite in 8 Intervallen.

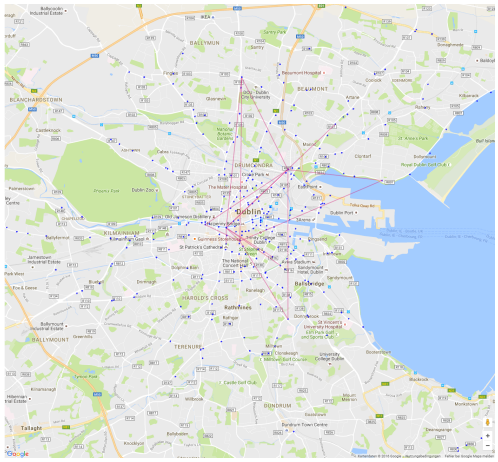

Abbildung 6.7: Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. Diskretizierte Daten nach Häufigkeit in 8 Intervallen.

STRF

	vorhersagte Klasse				
	0_63	63_125	125_188	188_*inf*	recall
0_63	808118	147878	20065	17	0.8279236
63_125	103155	275872	40566	49	0.65739846
125_188	11657	39345	31503	52	0.3815909
188_*inf*	4	59	54	18	0.13333334
precision	0.87559676	0.59563774	0.34172562	0.13235295	
fMeasure	0.8511	0.6250	0.3606	0.1328	

Tabelle 6.2: ℓ_1 mit Binning

	vorhersagte Klasse				
	0_63	63_125	125_188	188_*inf*	recall
0_63	517846	404078	53970	184	0.53053755
63_125	143816	190206	85546	74	0.45325777
125_188	16163	34501	31853	40	0.3858304
188_*inf*	4	66	55	10	0.074074075
precision	0.76397735	0.30246592	0.18581413	0.032467533	
fMeasure	0.6262094	0.36281782	0.2508298	0.04514673	

Tabelle 6.3: Duration 5-NN mit Binning

	vorhersagte Klasse				
	0_63	63_125	125_188	188_*inf*	recall
0_63	523034	396095	56735	214	0.5358527
63_125	146231	182197	91107	107	0.43417245
125_188	16395	34244	31801	117	0.38520053
188_*inf*	4	63	54	14	0.1037037
precision	0.76281387	0.29741642	0.17697012	0.030973451	
fMeasure	0.62950087	0.35301256	0.24252062	0.047700167	

Tabelle 6.4: Distance 5-NN mit Binning

	vorhersagte Klasse				
	0_63	63_125	125_188	188_*inf*	recall
0_63	521985	395538	58241	314	0.53477794
63_125	142000	188455	89131	56	0.44908518
125_188	15591	34595	32307	64	0.39132962
188_*inf*	4	67	43	21	0.15555556
precision	0.7680994	0.3046205	0.17976096	0.046153847	
fMeasure	0.63054687	0.3630079	0.24635598	0.071186446	

Tabelle 6.5: Haversine Distance 5-NN mit Binning

	vorhersagte Klasse				
	0_19	19_46	46_81	81_*inf*	recall
0_19	294034	72385	15512	10008	0.7502035
19_46	71957	209705	82301	27315	0.5359489
46_81	13408	67249	181055	96195	0.50587165
81_*inf*	6964	14837	72784	242703	0.719572
precision	0.76103044	0.5758342	0.5148698	0.6451075	
fMeasure	0.75557816	0.55517614	0.5103311	0.6803081	

Tabelle 6.6: ℓ_1 mit Frequency

	vorhersagte Klasse				
	0_19	19_46	46_81	81_*inf*	recall
0_19	231821	126068	22858	11192	0.59147215
19_46	82408	184910	90353	33607	0.4725796
46_81	16498	80013	162719	98677	0.45464045
81_*inf*	7962	24703	109813	194810	0.5775776
precision	0.6844657	0.4448224	0.42183268	0.5758737	
fMeasure	0.63458014	0.45828107	0.43762255	0.5767244	

Tabelle 6.7: Haversine Distance 5-NN mit Frequency

	vorhersagte Klasse				
	0_19	19_46	46_81	81_*inf*	recall
0_19	229176	128530	22984	11249	0.58472365
19_46	81064	184081	91731	34402	0.4704609
46_81	15948	78819	162232	100908	0.45327976
81_*inf*	7788	22961	111482	195057	0.57830995
precision	0.68620497	0.44422054	0.4176619	0.57098323	
fMeasure	0.6314128	0.45696434	0.43474254	0.5746232	

Tabelle 6.8: Duration 5-NN mit Frequency

	vorhersagte Klasse				
	0_19	19_46	46_81	81_*inf*	recall
0_19	198370	147784	32599	13186	0.5061247
19_46	80748	174953	96240	39337	0.44713223
46_81	17666	91144	148039	101058	0.4136242
81_*inf*	11387	36871	122688	166342	0.49317497
precision	0.6437011	0.38813582	0.3704995	0.51994383	
fMeasure	0.56668234	0.4155505	0.39087597	0.5062058	

Tabelle 6.9: Distance 5-NN mit Frequency

	vorhersagte Klasse						
	0_42	42_84	84_125	125_167	167_209	209_*inf*	recall
0_42	599556	107822	20175	7266	402	4	0.81547284
42_84	88120	253484	74453	16558	495	1	0.5852634
84_125	10977	69083	125191	21861	352	4	0.55036753
125_167	4414	16777	27536	30816	592	7	0.3845175
167_209	479	482	429	681	357	8	0.14655173
209_*inf*	0	0	12	6	9	3	0.1
precision	0.8521916	0.5662574	0.505218	0.39923304	0.16175804	0.11111111	
fMeasure	0.83348	0.5756	0.5268	0.3917	0.15378	0.1053	

Tabelle 6.10: ℓ_1 mit Binning

	vorhersagte Klasse						
	0_42	42_84	84_125	125_167	167_209	209_*inf*	recall
0_42	367141	302483	48892	15159	1505	45	0.4993587
42_84	92173	205065	105283	29782	800	8	0.47346985
84_125	15302	76619	99393	35509	645	0	0.43695378
125_167	4911	18230	27560	28341	1100	0	0.3536348
167_209	413	449	488	641	445	0	0.18267652
209_*inf*	0	0	12	4	14	0	0.0
precision	0.7649727	0.3401615	0.352923	0.25897327	0.09869151	0.0	
fMeasure	0.6042653	0.39589483	0.3904686	0.2989904	0.12814976	*NaN*	

Tabelle 6.11: Haversine Distance k-NN mit Binning

	vorhersagte Klasse						
	0_42	42_84	84_125	125_167	167_209	209_*inf*	recall
0_42	369124	303375	46611	14487	1582	46	0.5020558
42_84	92778	208169	101715	29503	942	4	0.4806366
84_125	15761	77347	98195	35429	736	0	0.4316871
125_167	4798	18639	27678	27966	1061	0	0.3489556
167_209	417	423	485	667	444	0	0.18226601
209_*inf*	0	0	12	4	14	0	0.0
precision	0.764425	0.3424097	0.3574679	0.25881025	0.09290647	0.0	
fMeasure	0.6060637	0.39991584	0.39108732	0.29719764	0.12307692	*NaN*	

Tabelle 6.12: Duration 5-NN mit Binning

	vorhersagte Klasse						
	0_42	42_84	84_125	125_167	167_209	209_*inf*	recall
0_42	370733	298140	49312	15411	1583	46	0.50424427
42_84	94413	202540	103406	31894	851	7	0.46763992
84_125	16694	77655	94231	38259	629	0	0.41426048
125_167	4867	19420	27313	27611	931	0	0.34452596
167_209	403	395	527	664	447	0	0.18349753
209_*inf*	0	0	12	4	14	0	0.0
precision	0.7610868	0.3386107	0.34290633	0.24253577	0.1003367	0.0	
fMeasure	0.606598	0.39280066	0.37522122	0.2846715	0.12973444	0	

Tabelle 6.13: Distance 5-NN mit Binning

	vorhersagte Klasse						
	0_11	11_27	27_46	46_68	68_96	96_*inf*	recall
0_11	186278	43069	16867	7270	4725	4164	0.709974
11_27	43703	129318	52426	15425	7860	6631	0.5064085
27_46	20657	49622	102802	55264	22260	14444	0.38786036
46_68	7542	13640	47860	84266	59393	27018	0.3515199
68_96	4110	4551	16747	46131	98417	62756	0.4229133
96_*inf*	4171	2373	7713	18427	52055	138457	0.6203382
precision	0.69908166	0.5331096	0.4206043	0.37157106	0.40217808	0.5462461	
fMeasure	0.704486	0.5194	0.40357	0.36127	0.4123	0.5809	

Tabelle 6.14: ℓ_1 mit Frequency

	vorhersagte Klasse						
	0_11	11_27	27_46	46_68	68_96	96_*inf*	recall
0_11	174494	51220	20910	7712	3873	4164	0.6651
11_27	50689	118228	53482	17678	8291	6995	0.46298
27_46	20491	54849	94669	54915	24840	15285	0.35717
46_68	6720	15318	51570	79931	57566	28614	0.33344
68_96	2732	4794	19242	51689	92369	61886	0.39692
96_*inf*	2892	2415	8175	22523	62268	124923	0.5597
precision	0.676286	0.479	0.3817	0.34093	0.3706	0.5165	
fMeasure	0.6706	0.4708	0.369	0.3371	0.3833	0.5372	

Tabelle 6.15: Haversine Distance 5-NN mit Frequency

	vorhersagte Klasse						
	0_11	11_27	27_46	46_68	68_96	96_*inf*	recall
0_11	173440	51437	21653	7841	3875	4127	0.66104364
11_27	50291	118598	53526	17778	8176	6994	0.46442908
27_46	20306	54943	94818	55170	24681	15131	0.35773763
46_68	6701	15355	51503	80262	57187	28711	0.33481702
68_96	2737	4841	19209	52196	91334	62395	0.39247653
96_*inf*	2979	2461	8187	22743	62463	124363	0.5571919
precision	0.67630064	0.4789226	0.3809543	0.34010762	0.3687045	0.5144898	
fMeasure	0.6685851	0.4715645	0.36898112	0.3374416	0.3802193	0.53499013	

Tabelle 6.16: Duration 5-NN mit Frequency

	vorhersagte Klasse						
	0_11	11_27	27_46	46_68	68_96	96_*inf*	recall
0_11	152447	65021	27310	8728	4443	4424	0.58103156
11_27	49458	112850	58491	18928	8567	7069	0.44191992
27_46	19869	53934	94699	56095	25100	15352	0.35728866
46_68	6692	15298	52576	79607	56081	29465	0.33208466
68_96	2948	4862	20357	56868	87089	60588	0.3742351
96_*inf*	3645	2530	9286	29887	70042	107806	0.48301044
precision	0.6485478	0.44342718	0.36045736	0.31828412	0.34652358	0.47976893	
fMeasure	0.612936	0.44267225	0.358866	0.32503796	0.35984662	0.4813842	

Tabelle 6.17: Distance 5-NN mit Frequency

	vorhersagte Klasse								
	0_32	32_63	63_94	94_125	125_157	157_188	188_219	219_*inf*	recall
0_32	465137	77238	13126	4489	2209	955	2	1	0.8259
32_63	119429	183602	53420	11349	5187	1127	7	4	0.49075
63_94	36174	88832	114875	32305	10669	1508	27	0	0.4039
94_125	15788	27817	53160	58552	12197	1143	32	0	0.3471
125_157	8878	13169	19223	14703	19311	1788	29	0	0.25046
157_188	1929	1970	1917	1063	1985	1924	28	0	0.17789
188_219	5	28	33	28	12	21	7	0	0.052239
219_*inf*	0	0	0	0	0	0	0	0	0
precision	0.7185	0.4676	0.4492	0.47802	0.37446	0.2273	0.05303	0	
fMeasure	0.7685	0.47889	0.4254	0.4022	0.3002	0.1996	0.05263	0	

Tabelle 6.18: ℓ_1 mit Binning

	vorhersagte Klasse								
	0_7	7_19	19_32	32_46	46_62	62_81	81_104	104_*inf*	recall
0_7	142147	24049	14147	8672	4905	3440	2480	2749	0.7017
7_19	22785	105498	34631	11309	5521	3512	2511	2566	0.5602
19_32	17881	38334	70549	37173	17253	9550	5916	5572	0.3489
32_46	11223	12654	37949	52169	36014	19532	10455	9056	0.276
46_62	5574	5143	16149	32856	47635	38958	20202	13917	0.2640
62_81	3137	2290	7112	15846	31006	54149	40594	23893	0.3042
81_104	2424	1247	3362	7417	15320	33737	61807	43343	0.36647
104_*inf*	2925	1139	2430	4901	9137	18074	39304	91182	0.5393
precision	0.6831	0.5542	0.3786	0.30626	0.2856	00.2992	0.3372	00.4742	
fMeasure	0.6922	0.55718	0.3631	0.2903	0.2744	0.3017	0.35125	0.5046	

Tabelle 6.19: ℓ_1 mit Frequency

PDN

Da PDN eine Regressionsmodell ist, kann man das Modell mittels quantitativen Fehlermaß beurteilen. Hier wird das Modell anhand der mittleren quadratischen Abweichung beurteilt. Auch die Streuungsmaße kann man auf dem Graph gut erkennen. Die 150 FISTA Iterationen waren nicht genügend für das PDN Modell. Dies kann man sowohl an dem komplexen graphischen Modell, als auch an der hohen Fehlerrate erkennen.

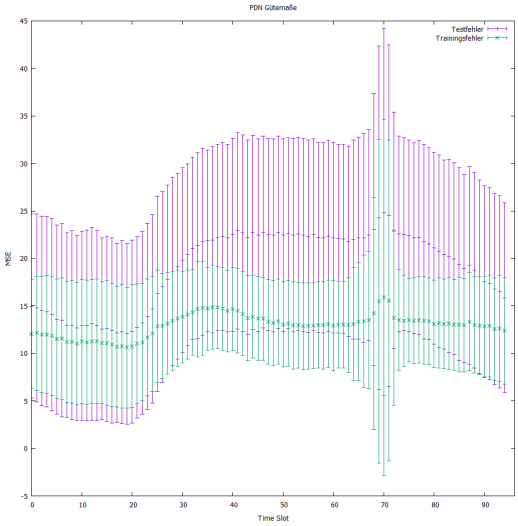

Abbildung 6.8: Trainings- und Testfehler mit Standardabweichung, geordnet nach der Zeitabschnitt, von PDN

Kapitel 7

Zusammenfassung

Im Rahmen dieser Bachelorarbeit wurden zwei Ansätze vorgestellt, die es möglich machen das Verkehrsaufkommen anhand von Sensordaten zu modellieren und vorherzusagen. Die Arbeit begann mit einer Einführung in die Grundbegriffe des Maschinellen Lernens (2) und wurde mit theoretischen Ansätzen der beiden Modellen STRF (3) und PDN(4) fortgesetzt. Obwohl es hier um zwei unterschiedlichen Ansätzen des überwachten Lernens geht, das STRF ist ein Klassifikationsverfahren und PDN ein Regressionsverfahren, versuchte man die beiden Modellen zu vergleichen. Um eine bessere Vergleichbarkeit zu ermöglichen wurden bei beiden Modellen gleiche Parameterschätzer und Optimierungsalgorithmen verwendet. Zumindest die graphische Modelle konnte man gut miteinander vergleichen und sah deutlichen Unterschied dort. Wichtig wäre vielleicht für diese Arbeit alle Lernvorgänge mit den gleichen Parametern durchzuführen, damit der Vergleich noch besser gelingt. Allerdings war es am Ende wegen der Grösse (2106*133*96) des Datensatzes leider nicht möglich. Insgesamt hätte man vielleicht den Datensatz noch mehr kürzen sollen. Kreuzvalidierung bei allen Lernvorgängen einsetzten und mehrere Iterationen bei der Regularisierung und Optimierung wären von Vorteil.

Anhang A

DVD

Abbildungsverzeichnis

2.1 Graphisches Modell 4
2.2 Graphisches Modell 7

3.1 Ein Graph zur Veranschaulichung von räumlich-zeitlichen Modell - STRF 18

5.1 Diskretisierung der Daten nach der Intervallbreite und Häufigkeit der Daten . . . 30
5.2 Räumliche Struktur abgeleitet von den Daten aus dem Datensatz 31
5.3 Ein Graph zur Veranschaulichung der Räumlichen Abhängigkeiten für STRF. Erstellt mit Haversine Distanz 31
5.4 Ein Graph zur Veranschaulichung der Räumlichen Abhängigkeiten für STRF. Erstellt mit Google Maps API's 33

6.1 Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, abgeleitet von dem PDN Modell. 38
6.2 Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. 38
6.3 Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. 38
6.4 Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. 39
6.5 Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. 39
6.6 Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. 39
6.7 Visualisierung der statistischen Abhängigkeiten zwischen den Sensoren, erstellt mit ℓ_1-Regularisierung. 39
6.8 Trainings- und Testfehlern mit Standardabweichung von PDN 46

Tabellenverzeichnis

2.1 Konfusionsmatrix 14

5.1 STRF Daten. Diskretisierung mit 6 Klassen, nach der Häufigkeit. 29
5.2 PDN Daten. Zeitabschnitt 43. 29
5.3 Obere Grenzen der Intervalle bei der Diskritizierungsverfahren 30

6.1 Anzahl der Kanten in Graphen, erstellt mit verschiedenen Algorithmen 36
6.2 ℓ_1 mit Binning 39
6.3 Duration 5-NN mit Binning 40
6.4 Distance 5-NN mit Binning 40
6.5 Haversine Distance 5-NN mit Binning 40
6.6 ℓ_1 mit Frequency 41
6.7 Haversine Distance 5-NN mit Frequency 41
6.8 Duration 5-NN mit Frequency 41
6.9 Distance 5-NN mit Frequency 42
6.10 ℓ_1 mit Binning 42
6.11 Haversine Distance k-NN mit Binning 42
6.12 Duration 5-NN mit Binning 43
6.13 Distance 5-NN mit Binning 43
6.14 ℓ_1 mit Frequency 43
6.15 Haversine Distance 5-NN mit Frequency 44
6.16 Duration 5-NN mit Frequency 44
6.17 Distance 5-NN mit Frequency 44
6.18 ℓ_1 mit Binning 45
6.19 ℓ_1 mit Frequency 45

Algorithmenverzeichnis

5.1 Distance Matrix Api 33

Literaturverzeichnis

[1] BECK, AMIR und MARC TEBOULLE: *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*. SIAM journal on imaging sciences, 2(1):183–202, 2009.

[2] BISHOP, CHRISTOPHER: *Pattern Recognition and Machine Learning*. Springer, January 2006.

[3] BOLLOBAS, BELA: *Graph theory: an introductory course*, Band 63. Springer Science & Business Media, 2012.

[4] FAHRMEIR, LUDWIG, ALFRED HAMERLE und GERHARD TUTZ: *Multivariate statistische verfahren*. Walter de Gruyter GmbH & Co KG, 1996.

[5] FAHRMEIR, LUDWIG, CHRISTIAN HEUMANN, RITA KÜNSTLER, IRIS PIGEOT und GERHARD TUTZ: *Statistik: Der Weg zur Datenanalyse*. Springer-Verlag, 2016.

[6] FRIEDMAN, JEROME, TREVOR HASTIE und ROBERT TIBSHIRANI: *The elements of statistical learning*, Band 1. Springer series in statistics Springer, Berlin, 2001.

[7] GREEN, PETER J: *Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives*. Journal of the Royal Statistical Society. Series B (Methodological), Seiten 149–192, 1984.

[8] HADIJI, FABIAN, ALEJANDRO MOLINA, SRIRAAM NATARAJAN und KRISTIAN KERSTING: *Poisson Dependency Networks: Gradient Boosted Models for Multivariate Count Data*. Machine Learning, 100(2-3):477–507, 2015.

[9] HARRACH, BASTIAN VON: *Einführung in die Optimierung*.

[10] JORDAN, MICHAEL IRWIN: *Learning in graphical models*, Band 89. Springer Science & Business Media, 1998.

[11] KOLLER, DAPHNE und NIR FRIEDMAN: *Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series)*. 2009.

[12] MOON, T. K.: *The expectation-maximization algorithm*. IEEE Signal Processing Magazine, 13(6):47–60, Nov 1996.

[13] NELDER, JOHN A und R JACOB BAKER: *Generalized linear models*. Encyclopedia of statistical sciences, 1972.

[14] PARIKH, NEAL, STEPHEN P BOYD et al.: *Proximal Algorithms.* Foundations and Trends in optimization, 1(3):127–239, 2014.

[15] PEARL, JUDEA: *Fusion, propagation, and structuring in belief networks*. Artificial intelligence, 29(3):241–288, 1986.

[16] PIATKOWSKI, NICO, SANGKYUN LEE und KATHARINA MORIK: *Spatio-temporal random fields: compressible representation and distributed estimation*. Machine learning, 93(1):115–139, 2013.

[17] SCHLITTGEN, RAINER: *Einführung in die Statistik: Analyse und Modellierung von Daten*. Walter de Gruyter, 2008.

[18] SCHMIDT, MARK: *Graphical model structure learning with l1-regularization*. Doktorarbeit, Citeseer, 2010.

[19] WAINWRIGHT, MARTIN J und MICHAEL I JORDAN: *Graphical models, exponential families, and variational inference*. Foundations and Trends® in Machine Learning, 1(1-2):1–305, 2008.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 7. Oktober 2016

Liudmila Vishnyakova