UNIVERSITY OF CALIFORNIA,
IRVINE

Probabilistic Learning for Analysis of Sensor-Based Human Activity Data

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Jonathan Hutchins

2010

The dissertation of Jonathan Hutchins
is approved and is acceptable in quality and form for
publication on microfilm and in digital formats:

Committee Co-Chair

Committee Co-Chair

University of California, Irvine
2010

TABLE OF CONTENTS

Page

LIST OF FIGURES **vi**

LIST OF TABLES **ix**

ACKNOWLEDGMENTS **x**

ABSTRACT OF THE DISSERTATION **xii**

1 Introduction **1**
- 1.1 People-Counting Sensors and Problems Addressed 1
- 1.2 Related Work 7
- 1.3 Thesis Organization and Novel Contributions 10

2 Background **14**
- 2.1 Nonhomogenous Poisson Process 14
- 2.2 Bayesian Parameter Estimation 18
  - 2.2.1 Gibbs Sampling 20
- 2.3 Directed Graphical Models 23
  - 2.3.1 Hidden Markov Models 25

3 Modeling People-Counting Sensors **29**
- 3.1 Simple Event Models and Their Limitations 29
  - 3.1.1 Event Detection Challenges 32
  - 3.1.2 Event Quantification Challenges 35
  - 3.1.3 Sensor Failure Challenge 37
- 3.2 MMPP: A Probabilistic Event Model 38
  - 3.2.1 The Normal Count: Modeling Periodic Count Data 39
  - 3.2.2 The Event Count: Modeling Rare, Persistent Events 42
  - 3.2.3 MMPP 45
- 3.3 Learning and Inference with the MMPP 48
  - 3.3.1 Sampling Hidden Data Variables Given Parameters 49
  - 3.3.2 Sampling Parameters Given the Complete Data 53
  - 3.3.3 Parameter Settings 54
- 3.4 Case Study 1: One Building Sensor and One Traffic Sensor 55
  - 3.4.1 Data Set Characteristics 55

3.4.2 Results 58
3.5 Assessing the Poisson Assumption 65
3.6 Summary of Contributions 69

4 Negative Event and Fault Tolerant Event Model Extensions 71
4.1 Negative Event Model 72
4.1.1 Negative Event Examples and Motivation 73
4.1.2 Adding a Negative Event State 75
4.1.3 Results 77
4.2 Sensor Failure Extension 78
4.2.1 Examples of Sensor Failure 79
4.2.2 Modeling Challenges 81
4.2.3 Modeling Failures Explicitly 86
4.3 Case Study 3: Scale-up Study 89
4.3.1 Failure Detection and Removal 90
4.3.2 Large-Scale Urban Network Analysis 93
4.4 Summary of Contributions 100

5 Transportation Event Model 102
5.1 Mainline Traffic Characteristics 103
5.2 Transportation Event Model 106
5.2.1 Occupancy Process 107
5.2.2 Event Process Changes 110
5.3 Case Study 3: Accident Characterization and Event Ranking 112
5.3.1 Data 112
5.3.2 Event Severity Ranking 113
5.3.3 Spatiotemporal Accident Characterization 126
5.4 Summary of Contributions 130

6 Linking Building Sensors: A Dynamic Building Occupancy Model 131
6.1 Building Sensors 133
6.2 A Baseline Method for Inferring Occupancy 135
6.3 Building Occupancy Model 138
6.3.1 Extending the Single-Sensor Model to Account for Noise . . . 139
6.3.2 Linking Multiple Sensors 141
6.4 Inference 145
6.4.1 Sampling the Individual-Sensor Parameters and Variables . . . 146
6.4.2 Sampling the Building Occupancy Variables 146
6.5 Implementation Issues 149
6.5.1 Countably Infinitely Valued Variables 149
6.5.2 Successive Convolution 150
6.5.3 Parameter Settings 152
6.6 Case Study 4: Building Population Study 153
6.6.1 Sensor Noise 153
6.6.2 Missing Data 155

6.6.3 Corrupted Data 156
6.7 Summary of Contributions 159

7 Clustering Sensors with Correlated Event Signals 160
7.1 Creating the Event Data Matrix 162
7.2 Problem Formulation and Notation 165
7.3 EM Algorithm for a Mixture of Multi-Dimensional Conditionally Independent Bernoulli Distributions 167
7.3.1 Data Likelihood and the Need for EM 167
7.3.2 Complete Data Likelihood 169
7.3.3 Q function 170
7.3.4 E-Step 172
7.3.5 M-Step 172
7.4 Simulation 176
7.5 Case Study 5: Clustering Building Streams 177
7.5.1 Data Description 179
7.5.2 Brute Force Clustering: Creating a Validation Set 179
7.5.3 EM Experiment with Real Data 181
7.6 Future Work: Temporal Event Pattern Analysis 186
7.7 Alternate Event Paradigms 191
7.7.1 Allowing Events to Span Time 191
7.7.2 Defining Event Clusters Using Multinomials 193
7.8 Summary of Contributions 195

8 Directions for Future Research on Event Modeling Problems 197
8.1 Nonstationary Behavior 198
8.1.1 Gradual Increase or Decrease 199
8.1.2 Bimodal Behavior 200
8.2 Multiple Sensor Models 200
8.3 Dynamic Occupancy Model 206
8.3.1 Ramp Sensor Flow Profile Characterization 209
8.4 Cell Phone Data, “Humans as Sensors” in Continuous Space 219
8.5 Conclusion 220

Bibliography 221

Appendices 227
A Implementation Details 227
A.1 Two-State Positive Event Model 227
A.2 Negative Event Model 229
A.3 Fault Tolerant Model 229
A.4 Transportation Model 230
A.5 Building Occupancy Model 232

LIST OF FIGURES

Page

1.1 Examples of People-counting Sensors 2
1.2 One Week of Vehicle Count Measurements 3
1.3 Profile Shape of Freeway On-Ramps Reflect Land Use 4
1.4 Building Optical Sensor Time Series Example 5

2.1 Nonhomogenous Poisson Process Illustration. 16
2.2 Poisson Probability Mass Function Example 17
2.3 Illustration of a Weak Gamma Prior 19
2.4 Illustration of a Gibbs Sampler 22
2.5 Example of a Directed Graphical Model 23
2.6 Graphical Model of an HMM 25

3.1 Example of an event successfully detected by a simple threshold method. 30
3.2 The Chicken-Egg Problem. 32
3.3 The Persistence Problem 34
3.4 The Duration Problem 35
3.5 The Popularity Problem 36
3.6 The Problem of Sensor Failure 37
3.7 The Normal Count Process; Independent Rate Parameters 39
3.8 The Normal Count Process; Consecutive Time-Segments 40
3.9 A Day with No Event Activity 41
3.10 The Event Count Process 44
3.11 MMPP Graphical Model 46
3.12 MCMC Convergence Assessment 49
3.13 Sampling Data Variables Given the Parameters 50
3.14 Sampling the Parameters Given the Data Variables 52
3.15 Three Weeks of Building People Counts 55
3.16 One Week of Traffic Measurements 58
3.17 Model Output for Building Data 59
3.18 Building Atrium Event Example 60
3.19 Model Results for Chicken-Egg Problem Day 61
3.20 Model Results for Persistence Problem Day 62
3.21 Baseball Game Attendance Prediction 64
3.22 Poisson Assumption: Simulated Data 67
3.23 Observed Data is Over-dispersed 68

3.24 Observed Data with Event Data Removed is Near Poisson 68

4.1 Illustration of a Negative Event 72
4.2 Illustration of a Negative Event due to a Holiday 74
4.3 Event Histograms for High Flow and High Occupancy Events 78
4.4 Two Examples of Sensor Failure 79
4.5 Example of a Less Obvious Sensor Failure 81
4.6 Event Model Fails for Sensor with Stuck at Zero Failure 84
4.7 Graphical Model of Fault Tolerant Model 87
4.8 Fault Tolerant Model Results for the Stuck at Zero Example 90
4.9 Fault Tolerant Model Results for the Corrupted Signal Example . . . 91
4.10 Examples of Sensors with No Regular Periodic Component 92
4.11 Map of Study Area and Sensors Locations 94
4.12 Histograms of the Duration of Events and Sensor Failures 95
4.13 Event and Fault Frequency Analysis 96
4.14 Event Frequency Mirrors Normal Vehicle Flow Profile 97
4.15 Mean Poisson Weekday Rate Analysis 97
4.16 Weekday Afternoon Oscillations in Normal Profiles and Observed Data 98
4.17 Spatial Event Analysis Example 99

5.1 Occupancy Mimics Flow Under Free Flow Condition Not Under Congested Conditions 103
5.2 Illustration of Traffic Flow Dynamics During Congestion 104
5.3 Occupancy Measurement Needed to Detect Congestion Events 105
5.4 Graphical Model of the Transportation Event Model 107
5.5 Event Histograms for High Flow and High Occupancy Events 108
5.6 Spatiotemporal Event Illustration for an Event Caused by a Single Accident 117
5.7 Spatiotemporal Event Illustration for an Event with Two Point Causes 119
5.8 Illustration of a High Flow Event Caused by Holiday Traffic 121
5.9 Illustration of a High Flow Event Caused by Construction 123

6.1 Example of a Current Method for Occupancy Analysis 132
6.2 Picture of Calit2 Building and Sensors 133
6.3 In-Flow and Out-Flow at All Six Doors 134
6.4 Building Occupancy Using Raw Measurements Results in Drift 135
6.5 Over-Counting and Under-Counting Examples 136
6.6 The Inaccuracies of the Baseline Method 137
6.7 Single Sensor Graphical Model, Noise Added 139
6.8 Building Occupancy Graphical Model 142
6.9 Grouping Sensors by Direction of Flow: Plate Diagram 142
6.10 Constraint Node 144
6.11 Gibbs Sampling Steps for the Building Occupancy Model 147
6.12 Illustration of Convolution 150
6.13 Successive Convolution Hierarchy 151

6.14 Model vs. Baseline: Sensor Noise 154
6.15 Model vs. Baseline: Missing Data 156
6.16 Model vs. Baseline: Corrupt Data 157

7.1 Convergence testing for the simulation experiment using log likelihood 178
7.2 Convergence was rapid for $K = 2$. 183
7.3 Convergence was slightly slower when K was increased to 24. 185
7.4 Kernel Density Estimation Produces a More Readable Plot 187
7.5 Temporal Analysis: Front Door Entrance 187
7.6 Temporal Analysis: Front Door Exit 188
7.7 Temporal Analysis: Front Door Exit and Entrance 188
7.8 Temporal Analysis: Loading Dock Door Exit 188
7.9 Temporal Analysis: Glass Side Door Exit and Entrance 189
7.10 Temporal Analysis: Courtyard Door Entrance 189

8.1 Examples of Gradual Rise Non-Stationarity 198
8.2 Examples of Bimodal Non-Stationarity 199
8.3 Finding Links between Road Sensors Using Inverse Covariance 202
8.4 Finding Links between Road Sensors: Jittered Plot 203
8.5 Segmenting an Urban Area around Ramp Sensors; Voronoi Diagrams 207
8.6 Map Showing a Stretch of Freeway without Working Sensors 208
8.7 Profile Shape of Freeway On-Ramps Reflect Land Use 210
8.8 Examples of Bimodal Non-Stationarity 211
8.9 First Four Principal Components 212
8.10 PCA Reconstruction Example 214
8.11 Census Trial Results 215
8.12 Landmark Information 216
8.13 Freeway Approach Direction Influences Flow Profile 217

LIST OF TABLES

		Page
3.1	Event-Detection Accuracy: Building and Traffic Data Experiments	63
4.1	Event Model Results; Using Event Fraction to Test Fitness	83
4.2	Fault Tolerant Event Model Results	89
5.1	High Occupancy Events Ranked by Delay	116
5.2	High Flow Event Rankings, 57 North	120
5.3	High Flow Event Rankings, 57 South	120
5.4	High Flow, Low Occupancy Event Rankings, 57 North	125
5.5	High Flow, Low Occupancy Event Rankings, 57 South	125
7.1	Example of **X** and **Z** for a Building with 2 Doors	164
7.2	Example of Cluster Parameter Values	165
7.3	Simulation Parameters Compared with Initialization Parameters	176
7.4	Inferred Door Event Data	178
7.5	Cluster Results for Brute Force Method	180
7.6	Model Parameters Learned by EM for the Real Door Data	182
7.7	Model Parameters for $k = 9$	183
7.8	Model Parameters for $k = 24$	184
7.9	Model parameters for Alternate Event Period Method, $k = 15$	192
7.10	Multinomial Model Parameters $k = 9$	194

ACKNOWLEDGMENTS

During my PhD program, I have had the great privilege of working with two exceptional advisors, Professor Alexander Ihler and Professor Padhraic Smyth. I want to first thank them for their excellence as teachers, researchers and communicators. I enjoy joking about the first paper I co-authored with them. They encouraged me to write the first draft of the paper as a learning experience; then after they each took an editing pass, it was difficult to find two sentences of mine that remained in their entirety. I have learned a great deal about research and writing from both of them. Secondly, I want to thank them for their care and availability. They each set aside weekly times to meet with me despite the demands of their schedules, and they gave me support through a period of illness when my productivity was low without adding pressure. I am well aware that graduate students rarely receive this level of encouragement and care. Lastly I want to thank them for their integrity. They set a high bar for our work, ensuring that our assertions were sound and that the contributions described in our submitted work were substantive. I had confidence in every paper we submitted, and I have a thesis of work I am proud of due in large part to their guidance.

I would also like to thank my collaborators at the Institute of Transportation Studies, Professor Will Recker, James Marka, and Craig Rindt, as well as Professor Sharad Mehrotra from the Rescue Project, for their advice, feedback, financial support, and especially for the interesting problems we discovered and investigated together.

My lab members also provided much support. In particular I want to thank Nick Navaroli for all his work and ideas, and Drew Frank and Arthur Asuncion for letting me bounce numerous ideas off of them. I want to thank Joshua O'Madadhain for involving me in his thesis work and including me as a co-author for my first publication. And I want to thank my other lab mates Seyoung Kim, Chaitanya Chemudugunta, America Chambers, Chris DuBois, Jimmy Foulds, David Keator, Scott Triglia, Corey Schaninger, and Scott Crawford for their input.

Dear to my heart, I want to thank my wife and family. First my devoted wife Amanda who has been my number one supporter and encourager throughout my PhD program, even as she was running a household including two, then three, and finally four children while living in a tiny apartment. She saw a broad spectrum of "for better or worse, richer or poorer, and in sickness and in health" and handled it with grace and love. I also want to mention her impressive achievement of reading and editing every single page of this thesis. I want to thank my dad for being a seemingly never ending source of aid and help, including helping care for our children and washing a countless number of dishes; and also for providing love and advice. And I want to thank my children, Tommy, Evelyn, Anna, and little Benny for being a regular source of joy and refreshment.

Finally I want to give thanks to Jesus, whose goodness and grace to me both humbles and amazes me.

> And God is able to make all grace abound to you, that always having all sufficiency in everything, you may have an abundance for every good deed
> 2 Corinthians 9:8

The material in this thesis is based upon work supported in part by the National Science Foundation under Award Numbers NSF ITR-0331707, IIS-0083489, and IIS-0431085 and by the Office of Naval Research under grant N00014-08-1-1015. Support also came from “A Measurement-based System for TMC Performance Evaluation” California Department of Transportation, Division of Research and Innovation September 1, 2007 - March 31, 2009.

ABSTRACT OF THE DISSERTATION

Probabilistic Learning
for Analysis of Sensor-Based Human Activity Data

By

Jonathan Hutchins

Doctor of Philosophy in Computer Science

University of California, Irvine, 2010

Professor Padhraic Smyth, Co-Chair
Professor Alexander Ihler, Co-Chair

As sensors that measure daily human activity become increasingly affordable and ubiquitous, there is a corresponding need for algorithms that unearth useful information from the resulting sensor observations. Many of these sensors record a time series of counts reflecting two behaviors: 1) the underlying hourly, daily, and weekly rhythms of natural human activity, and 2) bursty periods of unusual behavior. This dissertation explores a probabilistic framework for human-generated count data that (a) models the underlying recurrent patterns and (b) simultaneously separates and characterizes unusual activity via a Poisson-Markov model. The problems of event detection and characterization using real world, noisy sensor data with significant portions of data missing and corrupted measurements due to sensor failure are investigated. The framework is extended in order to perform higher level inferences, such as linking event models in a multi-sensor building occupancy model, and incorporating the occupancy measurement from loop detectors (in addition to the count measurement) to apply the model to problems in transportation research.

Chapter 1

Introduction

1.1 People-Counting Sensors and Problems Addressed

Sensors that monitor human activity are a common part of everyday life. While many sensors do not store a record of their measurements, such as sensors that open doors and turn on lights, others record a history of people or object-counts. Examples include loop sensors for monitoring freeway traffic [20] (such as shown in Figure 1.1(b)), optical sensors for counting the number of people passing a particular point [33](such as shown in Figure 1.1(a)), and pre-processed video or optical motion detectors for monitoring a specific area [79]. Rather than storing a unique time stamp for each individual count, these sensors often divide the day into equal-sized time segments and report an aggregate count for each time segment.

People-counting sensors are often installed for a simple purpose, such as tracking daily attendance or usage; however the history of counts recorded by these sensors

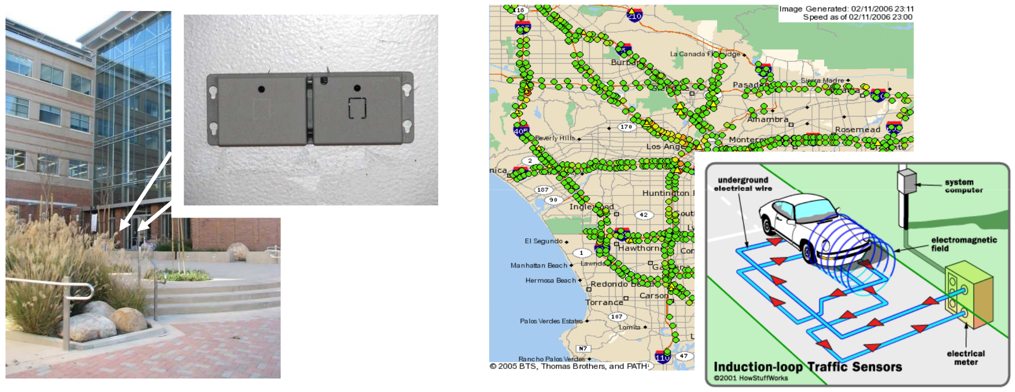


Image Generated: 02/11/2006 23:11
Speed as of 02/11/2006 23:00
underground electrical wire
system computer
electromagnetic field
electrical meter
Induction-loop Traffic Sensors
©2001 HowStuffWorks
© 2005 BTS, Thomas Brothers, and PATH

(a) (b)

Figure 1.1: Two examples of people-counting sensors. a) An optical sensor [45] that increments a counter when the optical beams at a building entrance are crossed. b) Loop detectors [30, 69] buried in the pavement of freeways count the number of vehicles traveling over them.

contain a wealth of information about human behavior and human activity that could be valuable for other applications. One important characteristic of people-counting sensor data is that they typically exhibit a periodicity that reflects the underlying rhythms of human behavior. Figure 1.2 shows a plot of one week of measurements from a loop detector. Loop detectors, illustrated in Figure 1.1(b), are wire loops buried in the pavement of freeways that count vehicles by measuring a change in inductance due to the metal in the vehicles that pass over them. The data acquired from one such sensor is shown in Figure 1.2; one can see the underlying traffic patterns; low traffic at night and early morning, rush hour traffic peaks in the morning and afternoon on weekdays, and slower rise of traffic on weekends with no rush hour activity. In aggregate, people are predictable, and the models presented in this dissertation will make use of the type of diurnal and calender periodicity which distinguishes human activity count data from other types of time-series count data.

The shape of the underlying normal pattern can provide other information about the data. For example, Figure 1.3, which shows the weekday vehicle count profile for

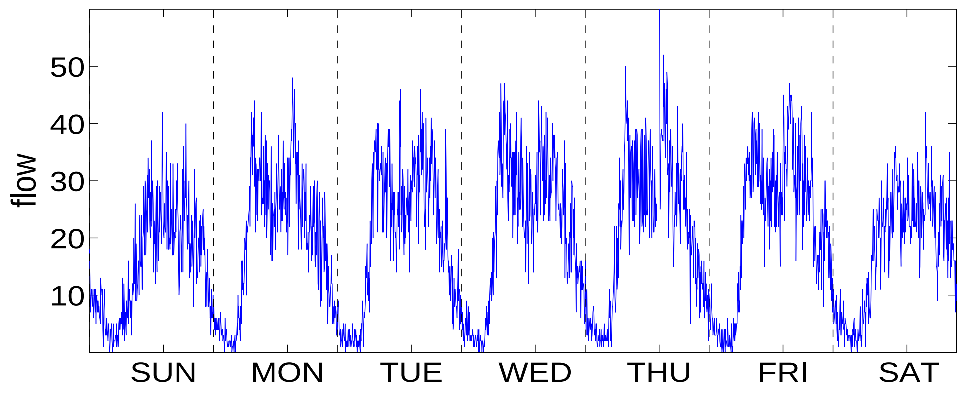
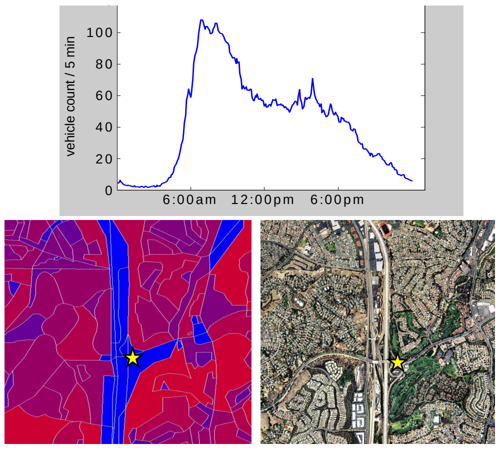
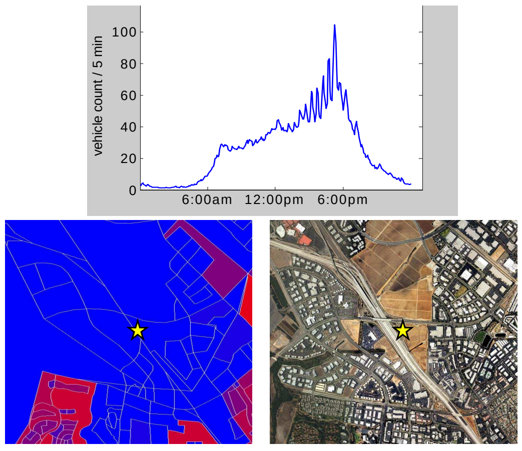


Figure 1.2: One week of vehicle count measurements (aggregated in 5-minute bins) from a loop detector. The diurnal patterns of rush hour traffic on weekdays are apparent.

loop detectors at two different on-ramps, illustrates how the shapes of the profiles can indicate land use of the area surrounding the sensors. Each ramp profile is displayed along with a satellite image of the area surrounding the sensor and a census map of the same area, reflecting the density of households in the area. The on-ramp on the left is in a residential area, as evidenced by the housing seen in the satellite image and the red areas in the census image that indicate a high density of households. The weekday ramp profile for this ramp has a large peak during morning rush hour, when people are leaving their homes to go to work, but a much smaller peak during the afternoon rush hour. The on-ramp on the right, on the other hand, is located in an industrial area, reflected by the larger buildings seen in the satellite and large blue sections in the census map indicating a low density of households. This ramp profile shows little to no morning rush hour peak, but a large afternoon rush hour peak as people leave their workplaces. The high oscillation in the profile on the right in the afternoon rush hour period is also commonly seen in on-ramps in industrial areas. A plausible explanation of the oscillation is that the peaks (which occur at the half hour and hour) correspond to common quitting times at workplaces in the area surrounding the sensor [63].



(a)



(b)

Figure 1.3: Profile shape of freeway on-ramps reflect land use. A plot of the underlying normal vehicle count pattern for a weekday for a loop detector at an on-ramp in (a) a residential area and (b) an industrial area. The ramp profiles are plotted along with a satellite image [1] of the area surrounding the sensor and a census map [27] of the surrounding area (red areas have high residential household density, blue area have low household density). The star marker indicates the position of the sensor on the maps.

While the normal periodic profile shape can provide valuable information, perhaps more interesting are when deviations from that pattern are observed. For example, if land use changes (for example, because of more residences being built) in an area that was previously industrial, we might expect to see the shape of the profile to change from one like the ramp in Figure 1.3(b) to one with more of the shape characteristics in Figure 1.3(a). In that case, the profile for normal activity changes gradually over time.

A more common type of deviation from the normal profile is a short period of deviation that does not change the underlying normal pattern, but is observed when some temporary incident alters the normal flow of people for a time. This type of deviation can be caused by a popular seminar in a building (reflected by abnormally high people-counts at a building sensor) or a large sporting event (reflected by abnormally high vehicle counts measured by a loop detector that is near to the stadium); or by a traffic accident (reflected by abnormally low vehicle counts at detectors on the freeway

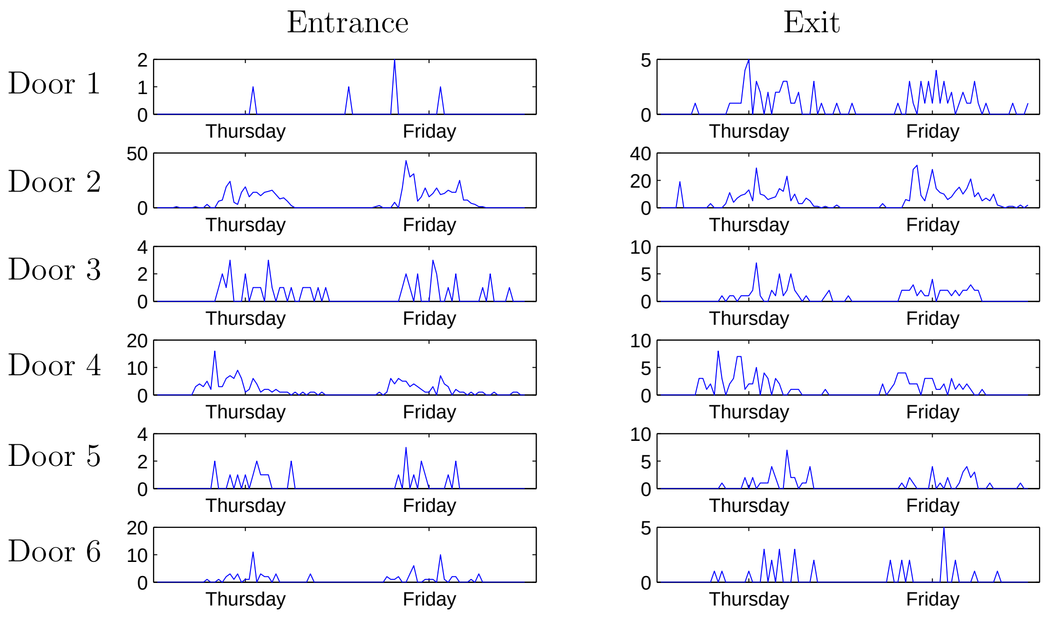


Figure 1.4: Flow of people entering the building (left) and exiting the building (right) from optical sensors at six doors of the same building. The y-axis for each plot is the count volume.

near the accident). In this dissertation, this second type of deviation is defined as an "event" and is the primary type of deviation studied.

Loop detectors are one of the sensor types used in the experiments in this dissertation. A second sensor type studied in this thesis is the optical people-counting sensor illustrated in Figure 1.1(a). This sensor emits a pair of optical beams spaced horizontally that count when the beams are broken, and also note the direction (exiting or entering) in which the beams are broken. Therefore, each sensor captures two time-series of data, one for the entrance stream and one for the exit stream. Figure 1.4 shows the data from six people-counters located at the six main entrances to a building on the campus of UC, Irvine. Each row is unique to one sensor; the first column shows two days of measurements in the entrance direction for each sensor, and the second column shows measurements in the exit direction for the same two days.

Figure 1.4 illustrates another characteristic of people-counting sensors, that they often have natural relationships with other sensors. Loop detectors on a highway have local relationships with sensors on the same freeway going the same direction, and the optical sensors in Figure 1.4 share the relationship that they measure traffic going into the same building. These relationships can be exploited for higher level inferences such as predicting the occupancy of a building.

As people-counting sensors become more dense, especially in urban areas, they form a network of relationships based on their spatial location that can lead to even more interesting higher level inferences. For example, in their recent paper "Instrumenting the City" [54] O'Neil et al. describe a project where sensors, such as bluetooth sniffers that count the number of people using bluetooth devices in a small radius, are used to create an even more dense network of people-counting sensors. With people-counters at buildings, on roadways and even on sidewalks, one can imagine the wealth of information about the presence and movement of people that can be gathered in real-time. In addition, sensors that record counts of observations of a human activity have the advantage of being relatively low-cost and privacy preserving as compared to video. As we look to the future, it is reasonable to expect these sensors to become even more common. As the networks of sensors become more dense, there is a corresponding need for algorithms that make sense of the data and extract information of value.

The primary problem studied in this dissertation is separating and characterizing "event" activity from the underlying normal patterns of human activity in people-counting sensors. We develop a probabilistic framework that uses Poisson-Markov models to simultaneously separate event and normal activity in the presence of sensor noise, sensor failure and missing data. We show how this framework can be built upon to make higher level inferences, namely creating a dynamic occupancy model for a building. We also show how the framework can be extended to make use of additional information beyond the people-count.

The research presented in this dissertation was motivated by a NSF funded project Rescue [51] aimed at providing first responders to large scale disasters, such as earthquakes or terrorist attacks, with technological tools that could enable them to make more informed decisions. Knowledge of the location and movement of people is critical for situational assessment and emergency planning; and early in the project we were impressed with the potential value of the real-time information that people-counting sensors provide. In some applications, census information was used to predict the location and density of people in a disaster area, without any real-time information used as input to the model [32]. We see the work presented in this dissertation to be an important step towards approaching some of the more ambitious goals in occupancy modeling and transportation planning, such as creating a real-time dynamic occupancy model of an urban area.

1.2 Related Work

Our definition of unusual events differs from the classical definition of outlier detection, such as defined by Hawkins [28] or used by Basu et. al. [3] for outlier detection in time series sensor data; in that we are not looking for single observations that are unusual, but rather a sequence of observations in a time series which alone might not be unusual, but together appear to be generated from a process different than the underlying normal process.

The theory for time series analysis has been widely studied and discussed in texts such as Chatfield [13]. For the specific problem of detecting unusual activity in time series there has also been much prior work.

For example, Keogh et al. [40] described a technique that represents a real-valued time-series by quantizing into it a finite set of symbols. Given a history of these symbols, normal patterns of behavior are learned. A “surprising pattern” is then found when a new pattern is observed with a frequency different than what was expected from the history.

Kawahara and Sugiyama [39] introduce a method for detecting change-points in time-series using a ratio of probability densites. Also, Guralnik and Srivastava [26] propose an iterative method for detecting change-points in sensor-based time series; where large time-segments are analyzed and divided based on whether they contain a change-point, and the time series is then segmented into piecewise homogeneous regions bordered by change-points.

Snowsill et. al. [73] describe a method of finding surprising changes in the frequency of n-grams in text streams using a generalized suffix tree. Also, Kleinberg [41] investigated the problem of detecting bursty events for a topic in a text stream using an infinite state automaton.

The goal of all of these methods is similar to that of this dissertation, namely the detection of unusual activity in time-series. However, none of this earlier work focuses on the specific problem of detecting bursty events embedded in a time series of counts that reflects the normal diurnal and calendar patterns of human activity.

Another research area that overlaps the work presented here is transportation research. Loop detector data is well studied by transportation researchers, but the prevalent missing and invalid data, sensor noise, and multiple modes of malfunction cause this data set to be considered resistant to systematic and principled approaches (such as is presented in this dissertation) as discussed by Bickel et al. [9] in a recent survey paper.

Rather than attempting to find unusual activity in general, algorithms for use with loop detector data commonly are developed to detect specific types of abnormalities. For example Jacobsen et al. [36] used volume-to-occupancy ratios to detect some common types of sensor failure, such as when occupancy (the amount of time the sensor is covered by a vehicle) is high but flow (the number of cars counted) is low (an unlikely combination). Chen et al. [14] describe an algorithm that uses a time series of samples, rather than making a decision based on one sample (common in earlier methods), to detect failures such as constant occupancy and flow, and measurements that are consistently near zero or consistently high. Kwon et al. [42] developed a method for detecting a type of configuration error when directions are switched, for example when a sensor in a southbound lane is being reported as being in a northbound lane.

Statistical models of the loop detector measurements have also been developed. Many of these create a model based on network relationships such as Singliar et al. who use a mixture of Gaussian trees that links spatially local sensors to model flow [71] and to impute missing measurements [70].

Other methods capture only a few aspects of normal periodic behavior. For example, Belomestny et al. [8] represent the loop detector measurements using a stochastic process with two components: one for free flow traffic and one for congested traffic using Gaussian distributions.

There is also prior work using HMMs in time-series to detect anomalies. There have been several methods proposed for use in detecting intrusions for computer security applications [58, 19]. However, these methods require a training set of normal activity, and are do not make use of the day-of-week and time-of-day information that is present in the data and employed by the models in this thesis.

Markov-Modulated Poisson Processes (MMPP) have been used in diverse applications such as modeling multimedia streams [68], volcanic activity [6], and schizophrenic behavior [7]. Our work is most closely related to the two-state Markov–modulated Poisson process used by Scott and Smyth [67] for analysis of Web surfing behavior and Scott [66] for telephone network fraud detection. The framework described in this thesis is derived from Scott's model. We extend this work in several ways. First, we create a more flexible model of event activity and make allowances for missing data. Additional event states were also added to the model to address common characteristics of sensor data from human activity, such as negative events (lower than normal activity levels) and sensor failure. We also show how to link event models together to form a multi-sensor building occupancy model, and we show how to incorporate other evidence (the occupancy measurement from loop detectors), in addition to count information, into the model.

1.3 Thesis Organization and Novel Contributions

The outline of the thesis including specific novel contributions is as follows.

Chapter 2 provides background information for the models described in this dissertation. We describe and illustrate nonhomogeneous Poisson processes, Bayesian parameter estimation, and Gibbs sampling; and we provide an overview of graphical models and hidden-Markov models.

Chapter 3 describes a two-state event model that uses a Markov modulated Poisson process. Inference and learning given a set of people-count observation are explained and case studies demonstrating the use of the model with two types of sensors are shown. The probabilistic framework described in this chapter is foundational to all of the remaining chapters which extend and build upon it.

Contributions: We develop an extension of the approach of Scott [65] that is useful for separating event activity and normal behavior from time-series data that is characteristic of measurements from the people counting sensors examined in this thesis. We show how the model can also be used to make other inferences such as the duration and popularity of unusual events found in two different human counting sensor measurement sets; and we show how the output of the model can be used to make higher level inferences such as predicting the attendance at nearby sporting events. We also show the advantages of the event model over a baseline method, and show how it is more accurate at predicting lists of known events in both data sets.

Chapter 4 describes two extensions of the two-state event model to handle negative events and sensor failure. A large urban case study is presented where the model is tested with over 100 million measurements from over 1700 loop detectors.

Contributions: The extensions presented in this chapter make the event model more robust to the wide variety of corruption due to noise, missing data, and sensor failure which is commonly found in real world sensors. We demonstrate how these extension provide a richer representation that can cull periods of sensor failure and that is useful in the analysis of human behavior. We present an analysis of a large urban network of traffic sensors. The model is able to provide useful insights about an urban traffic data set that is widely considered to be problematic; and to perform an analysis of a type that has not to our knowledge been accomplished before.

Chapter 5 presents an extension of the model that is crafted specifically for loop detectors and transportation applications. The model makes use of the occupancy measurement in addition to the flow (vehicle count) measurement provided by the loop detectors. A case study ranking spatio-temporal events is presented.

Contributions: We show how the general model introduced earlier in the thesis can be extended for use for a particular application, in this case transportation research applications that make use of data from loop detectors. We show how additional evidence (the occupancy measurement) can be incorporated into the event model. We present a case study showing how this model can be used in combination with other information (the location of the sensors) in order to perform application specific inferences such as detecting spatiotemporal characteristics of various event types (such as accidents) and ranking the impact of these events using various metrics. We discuss the benefits of the event model compared with current methods used in transportation research.

Chapter 6 presents a probabilistic building occupancy model that provides a dynamic estimation of the number of people in a building based on measurements from people counters at all major entrances to the building. A case study shows how the multi-sensor building occupancy model outperforms simple baseline methods, particularly in the cases where there are missing data and sensor failure.

Contributions: Large scale dynamic occupancy estimation is one of our long range objectives for the event model, and we see the dynamic building occupancy model described in this chapter as a step towards that goal. This chapter demonstrates how individual sensor models can be linked to form a multi-sensor model, which in turn can be used for higher level inferences. We introduce a model of sensor noise, and show how linking sensors through the constrained occupancy variable can be used to make inferences about the "true" count at each sensor. The case study demonstrates the robustness of the model to periods of missing data and sensor failure as compared to simpler methods.

Chapter 7 describes a method for clustering sensors that have event signals that coincide. An Expectation-Maximization algorithm for a mixture of multi-dimensional Bernoulli distributions is described in a tutorial style; and the results of the building occupancy model described in Chapter 6 are used to demonstrate the algorithm.

Contributions: People counting sensors often form logical networks with other sensors, such as the building in Chapter 6 and the freeway in Chapter 5. We see the work in this chapter as a possible step towards more closely linking sensors in order to make inferences about multi-sensor events. This chapter discusses how such linking can make the model more sensitive to group events, and presents a method for finding clusters of sensors with correlated event signals. We envision using the output of such a method to decide which sensors to link in a multi-sensor event model. A detailed EM tutorial in the context of learning the parameters of a mixture of multi-dimensional Bernoulli distributions is also presented.

Chapter 8 discusses a variety of possible directions for future work. We envision how the models developed in this thesis have potential to be useful in important applications such as dynamic population density estimation; and we discuss the role these models might play as people counting sensors become more dense in the future.

Chapter 2

Background

This chapter gives a brief introduction to some of the theory that is fundamental to the work described in this thesis. We discuss nonhomogeneous Poisson processes, Bayesian parameter estimation and Gibbs sampling, and directed graphical models and hidden Markov models.

2.1 Nonhomogenous Poisson Process

The Poisson process is commonly used to model stochastic processes that are characterized by a sequence of observations in time. Examples include the arrival of earthquakes [15], the formation of queues [78], and photon emissions due to radioactive decay [25]. The Poisson process is a continuous time process characterized by a rate parameter λ where the expected number of observations in a time interval of length m is λm.

A count process can be represented using the notation $\{N(t),\, t \geq 0\}$ where $N(t)$ is the number of observations prior to time t and where $N(0) = 0$. If the count process is a Poisson process, then the probability of observing d counts over a time period of length m is

$$\begin{aligned} p\Big(N(t+m) - N(t) = d\Big) &= \text{Poisson}(d; \lambda m) \\ &= e^{-\lambda m} \frac{(\lambda m)^d}{d!} \qquad d = 0, 1, \ldots \end{aligned} \tag{2.1}$$

The Poisson process can also be used to predict inter-arrival times. For example, the probability of waiting longer than a time interval of length m from the current time t for the next observation is

$$p\Big(N(t+m) - N(t) = 0\Big) = e^{-\lambda m} \tag{2.2}$$

Equations 2.1 and 2.2 assume that the arrival rate at a given time is the same as at any other time. If this assumption is accurate for a count process, it is referred to as a homogenous Poisson process. For example, the rate of computer processor failure in a large data center might not fluctuate much during the course of a day; therefore a homogenous Poisson process with a single rate parameter λ might be adequate to model the time series of cpu failure observations.

The processes studied in this thesis, however, are not homogenous. The rate of vehicles on a freeway, for example, is predictably lower in the middle of the night than at rush hour. When the rate parameter of the process changes as a function of time, $\lambda(t)$, the count process is referred to as a nonhomogenous Poisson process (NHPP).

In Equation 2.1 we saw that the probability of observing d counts over a time period of length m for a homogenous Poisson process was a function of the homogenous rate parameter multiplied by the length, Poisson$(d; \lambda m)$. To find the find the probability of observing d counts between two points in time in an NHPP requires integrating over

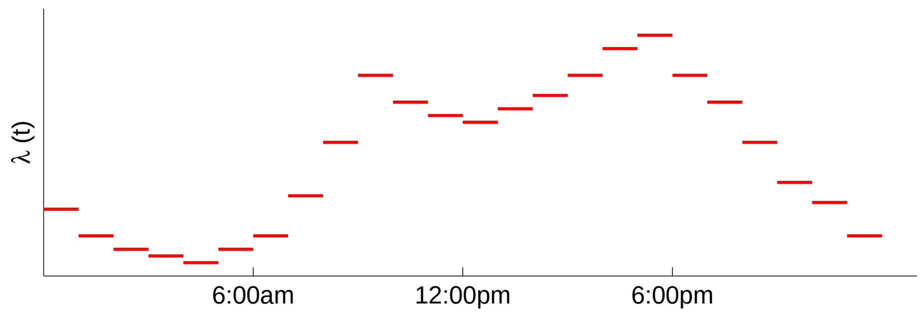


Figure 2.1: Nonhomogenous Poisson Process Illustration. Segmenting the day into equal-sized bins and learning a Poisson rate parameter for each time segment creates a piecewise constant function in time.

the rate parameter function $\lambda(t)$. So the probability of observing d counts between time a and time b in an NHPP is

$$p\Big(N(b) - N(a) = d\Big) = \text{Poisson}\left(d; \int_a^b \lambda(t)dt\right) \tag{2.3}$$

While the processes we study in this thesis are not homogenous, we can take advantage of their periodicity. For example, although vehicle traffic flow may fluctuate throughout the day, traffic at the same time of day and same day of the week (eg. Mondays at 3PM) is typically similar. A simple method for implementing an NHPP that is characterized by periodic behavior is to segment the week into equal-sized time intervals, and model each interval with a unique Poisson rate parameter. The rate parameter function $\lambda(t)$ then becomes a piecewise constant function in time as illustrated in Figure 2.1, where at each discrete time interval there is a homogenous Poisson process with constant rate $\lambda(t)$. This method is appropriate for our data because sensors that measure human activity commonly report an aggregate count measurement across a fixed time segment, and do not store time-stamps for each individual count[1]. The degree of heterogeneity of $\lambda(t)$ can then be naturally set to match the frequency that the sensor reports its measurements.

[1]If a time-stamped sequence of counts is available, a more complicated computation for determining more optimal segment divisions for the Poisson rate function $\lambda(t)$ can be made [64].

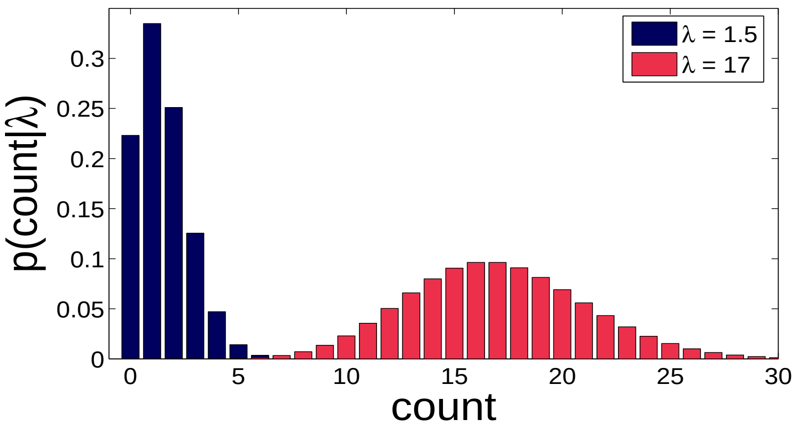


Figure 2.2: Poisson probability mass function for $\lambda = 1.5$ (blue) and $\lambda = 17$ (red).

Using this piecewise constant model, the aggregate count measurement reported at time t, follows a Poisson distribution. The Poisson distribution, $\mathrm{P}(d; \lambda(t))$ is a discrete probability distribution, where the probability of observing d counts during a fixed window of time is

$$\mathrm{P}(d; \lambda(t)) = e^{-\lambda(t)} \lambda(t)^d / d! \qquad d = 0, 1, \ldots \tag{2.4}$$

where the rate parameter $\lambda(t)$ is the expected count during the fixed time window at time t.

Figure 2.2 shows the probability mass function (pmf) for a Poisson distribution with rate parameter $\lambda = 1.5$ (blue) and the pmf for a Poisson distribution with rate parameter $\lambda = 17$ (red). A history of count measurements at the same time and day shares some common characteristics with the Poisson distribution. The observed count can not be negative, and the count process is discrete. Another characteristic of the Poisson distribution is that the variance of the distribution is equal to the mean of the distribution ($\sigma_o^2 = \bar{d} = \lambda$). We assess the accuracy of using the Poisson distribution to model the sensor measurements studied in this thesis in Chapter 3.5.

2.2 Bayesian Parameter Estimation

This section briefly describes Bayesian methods for parameter estimation. For a more in depth description see for example Gelman et. al. [22].

Given observed data $\mathbf{D}$ and a model of the data that is parameterized by the vector $\mathbf{\Theta}$, the joint probability is

$$p(\mathbf{D}, \mathbf{\Theta}) = p(\mathbf{D}|\mathbf{\Theta})p(\mathbf{\Theta}) \tag{2.5}$$

where $p(\mathbf{D}|\mathbf{\Theta})$ is the likelihood of the observed data given the model parameters, and $p(\mathbf{\Theta})$ is the prior probability of the model parameters.

For example, we explore the case where we model a series of count data ($\mathbf{D}$) using a single Poisson distribution, In this case $\mathbf{\Theta}$ is a single Poisson rate parameter $\mathbf{\Theta} = \lambda$. For reasons that will become clear, we might choose to model our prior distribution over λ as a gamma distribution, $p(\lambda) = \Gamma(\lambda; a, b)$, where a and b are parameters of the gamma distribution, with mean $= \dfrac{a}{b}$ and variance $\dfrac{a}{b^2}$.

At the heart of Bayesian inference and learning is the calculation of the posterior distribution $p(\lambda|\mathbf{D})$, which tells us our updated belief about the values of λ given the observations. The posterior can be found using Bayes' rule

$$p(\lambda|\mathbf{D}) = \frac{p(\mathbf{D}|\lambda)p(\lambda)}{p(\mathbf{D})} = \frac{p(\mathbf{D}|\lambda)p(\lambda)}{\int p(\mathbf{D}|\lambda)p(\lambda)d\lambda} \propto p(\mathbf{D}|\lambda)p(\lambda) \tag{2.6}$$

The posterior distribution is proportional to the product of the likelihood and prior because the marginal probability of the data $p(\mathbf{D})$ does not depend on Θ. $p(\mathbf{D})$ acts as a normalization constant.

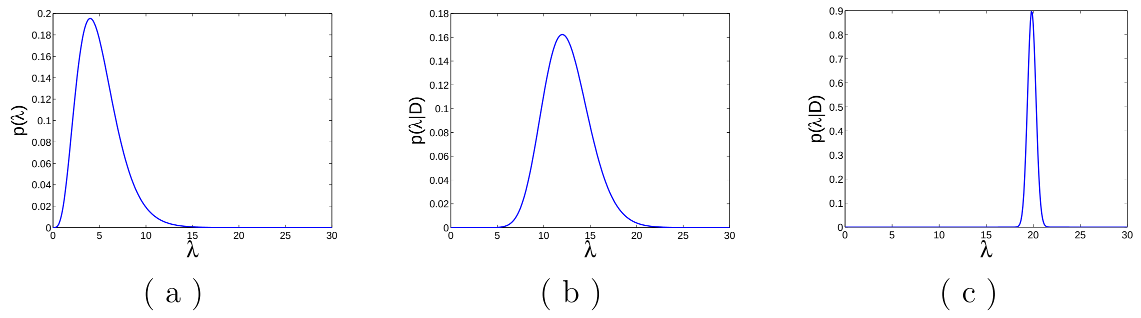


Figure 2.3: (a) the pdf for a weak gamma prior. (b) the pdf of the posterior distribution after observing one data point $d_1 = 20$. (c) the pdf of the posterior distribution after observing 100 data points averaging 20.

If the prior distribution $p(\lambda)$ is conjugate to the data likelihood $p(\mathbf{D}|\lambda)$ then the posterior distribution $p(\lambda|\mathbf{D})$ has the same functional form as $p(\lambda)$. For example, the gamma distribution is conjugate to the Poisson distribution; so given $\mathbf{D} = \{d_1, d_2, \ldots d_N\}$

$$p(\lambda|\mathbf{D}) \propto p(\mathbf{D}|\lambda)p(\lambda) = \mathrm{P}(\mathbf{D};\lambda)\Gamma(\lambda;a,b) = \Gamma(\lambda|(a + \sum_{i=1}^{N} d_i), (b+N)) \qquad (2.7)$$

The parameters of the prior distribution can be set by using the opinion of an expert or by using historical data from the same process or a similar process. Another common practice (which is used for the most part in the models in this thesis) is to use a weak prior whose parameter settings have a negligible impact on the posterior distribution. A prior is considered to be "weak" if the data have the dominant role in determining the posterior distribution, even for a small number of data samples. Figure 2.3 shows an example of a weak gamma prior. Figure 2.3(a) shows a gamma prior with $a = 5$ and $b = 1$. This plot shows our prior belief about our Poisson rate parameter λ. The prior distribution is centered near $\lambda = 5$; this indicates that we expect to observe counts near 5 in our process. After observing one data point, a count of $d_1 = 20$, Figure 2.3(b) shows the posterior distribution (a gamma distribution with parameters $a = 25$ and $b = 2$). The distribution has shifted to being centered around 12. So after observing only one data point, our belief about the rate parameter has shifted significantly from our prior belief. Figure 2.3(c) shows the posterior distribution after observing 100 counts with average value 20 (a gamma distribution with parameters $a = 2005$ and $b = 101$). Here the data has "overwhelmed" the prior. The distribution centered near 20 indicates that we no longer expect to observe counts near 5, but expect counts near 20. The relative narrowness of the distribution indicates increased confidence in our belief about λ.

2.2.1 Gibbs Sampling

This section gives a brief description of Gibbs sampling. For a brief tutorial see MacKay [46] or see Gilks et. al. [24] for a more in-depth description.

In many cases we cannot compute $p(\mathbf{\Theta}|\mathbf{D})$ in closed form, but there exist sampling methods that let us approximate it. We can then use the set of samples $\mathbf{S} = \{s_1, s_2, \ldots s_N\}$ from the target distribution to calculate point estimates of the parameters we are learning, for example

$$\hat{s} = \frac{1}{N}\sum_{i=1}^{N} s_i \tag{2.8}$$

where s_i is a sample of one of the variables in Θ and $\hat{s}$ is the point estimate for that variable given the samples.

Markov chain Monte Carlo (MCMC) methods are a family of algorithms that can be used to sample a probability distribution. Rather than generating a set of independent and identically distributed (i.i.d.) samples from the target distribution, MCMC methods generate a sequence of samples where each sample is dependent on the value of the previous sample. At each iteration, a sample is taken from a proposal distribution $Q(s_i; s_{i-1})$ given the value of the previous sample. The proposed sample must then pass an acceptance test or else a new sample is taken from the proposal distribution. Even though each sample is conditionally dependent on the previous sample, the process converges to the true distribution. So although the samples are correlated, their sample average still converges to the correct mean so Equation 2.8 is valid.

Gibbs sampling is a special case of MCMC. In Gibbs sampling, the proposal distribution is formed using conditional distributions of the joint distribution. An advantage of forming the proposal distribution this way is that proposed samples are always accepted. Figure 2.4 illustrates the steps of a Gibbs sampling algorithm for a two dimensional target distribution $p(X, Y)$ [2]. The green oval in each plot represents the 2-sigma boundary of the joint distribution. In the illustration, we assume that the joint distribution $p(X, Y)$ is computationally complex, but the conditional distributions $p(X|Y)$ and $p(Y|X)$ are more tractable. The steps of the Gibbs sampler are as follows:

1. Initial values for X and Y $(s_0 = (x_o, y_o))$ are needed to start the sampler. A possible initial setting, s_0, is shown in Figure 2.4(a).

2. A value y_1 is sampled from the conditional distribution $p(Y|X = x_0)$, as seen in Figure 2.4(b).

3. A value x_1 is sampled from the conditional distribution $p(X|Y = y_1)$, as seen in Figure 2.4(c). The first iteration of the Gibbs sampler is now complete and sample 1 has been drawn $s_1 = (x_1, y_1)$.

4. Steps 2 and 3 are repeated (conditioned on the most recent sample of x and y) for each iteration of the Gibbs sampler.

[2] Gibbs sampling is normally used for sampling higher dimensional distributions. For low dimensional distributions, simpler methods such as rejection sampling are often sufficient [46].

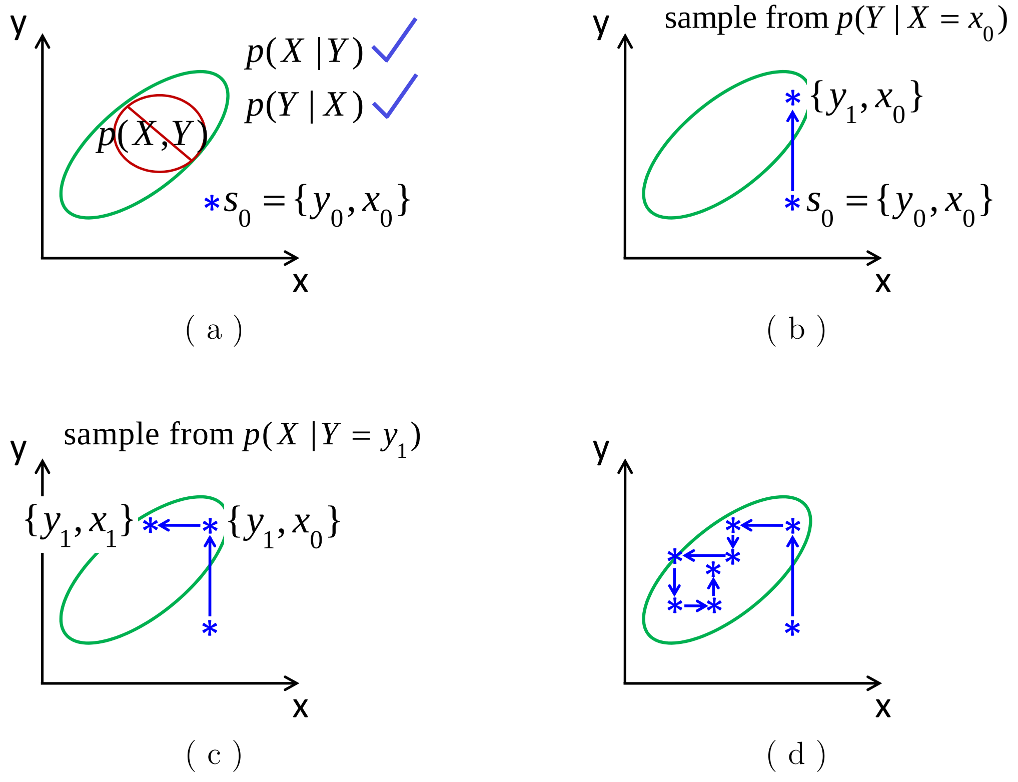


Figure 2.4: Illustration of a Gibbs Sampler. (a) Initial conditions: assume the joint distribution does not have a closed form but the conditional distributions are tractable. (b) Sample y_1 from $p(Y|X = x_0)$. (c) Sample x_1 from $p(X|Y = y_1)$. (d) The chain of samples after several iterations of the Gibbs sampler. Figure adapted from MacKay [46].

Figure 2.4(d) shows the chain created after several samples of the Gibbs sampler.

Depending on the quality of the initial guess, the first several samples of the Gibbs sampler may be far from the target distribution. Because of this, the first samples returned by the sampler are often not included in post-analysis (such as the calculation in Equation 2.8). The period of time during which these initial ignored samples are drawn is referred to as the "burn in" period.

A practical difficulty of MCMC methods is assessing convergence. Detecting convergence is a difficult problem [46], so a common way to evaluate convergence is to visually inspect a plot one or more of the variables being sampled. Another method

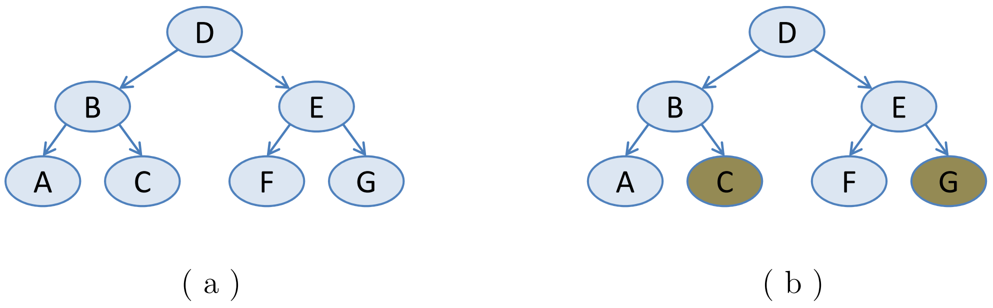


Figure 2.5: (a) Example of a directed graphical model. (b) the same model as in (a) after observing evidence (observing and fixing values for variables C and G) indicated by the shaded nodes.

is to run several Gibbs samplers with different initial values and test to see if they appear to be converging to similar ranges. Experimentally we find that the sampler converges rapidly on our data sets and that 10 burn in samples followed by 50 retained samples were sufficient. An example of an empirical convergence plot can be seen in Figure 3.12 on page 49.

2.3 Directed Graphical Models

A directed graphical model (also known as a Bayesian Network) provides a useful representation of the dependence structure among a set of random variables. Nodes in the graphical model represent random variables and edges encode dependencies. A graphical model is consistent with how a distribution factors. For example the graphical model shown in Figure 2.5 [72] factors as follows:

$$\begin{aligned} &p(A, B, C, D, E, F, G) \\ &\qquad = \prod p(\texttt{variable}|\texttt{parents}) \\ &\qquad = p(A|B)p(C|B)p(B|D)p(F|E)p(G|E)p(E|D)p(D) \end{aligned} \tag{2.9}$$

The graphical model can also be used to determine conditional independence relationships. Conditional independence is determined by a set of rules called d-separation. The Markov blanket of a node is the minimal set of nodes that make it conditionally independent of (or d-separated from) all remaining nodes in a graphical model. The Markov blanket of a node contains its parents, its children, and the parents of its children.

In addition, graphical models can make computation simpler and more intuitive. Computational properties of inference and learning can be determined by viewing the structure of the graph. If the graph is a chain or a tree, computation can be done exactly in time linear in the number of variables [57]. If the graph has undirected cycles, a chain or tree can be created by clustering variables [31] or by using cycle cutsets [16]; alternatively, approximate methods can be used for inference [48, 38].

To illustrate how the graphical model structure can be used to simplify inference, consider the graph in Figure 2.5(b). Here the value of some of the variables are known (the evidence), and the values of other variables are not known. Suppose we wish to compute the probability distribution of A given the evidence (i.e. $p(A|C,G)$). To calculate this directly from the joint distribution, we could use the law of total probability by summing over all of the values of the remaining unobserved variables

$$p(A|C,G) = \sum_{BDEF} p(A,B,D,E,F|C,G) \tag{2.10}$$

The time complexity of this computation is $O(k^5)$ where k is the number of values in the range of the variables.

However the sum in Equation 2.10 can be reordered using the following factorization of the graphical model

$$p(A|C,G) = \sum_{B} p(A|B) \left(\sum_{D} p(B|D,C) \left(\sum_{E} p(D|E) \left(\sum_{F} p(E,F|G) \right) \right) \right) \quad (2.11)$$

which has a time complexity of $O(k^2)$ since each sum is $O(k^2)$ and there are only 4 sums required.

The computational structure becomes more important with large numbers of variables. Some of the experiments in this thesis use models with over 100,000 unknown variables.

2.3.1 Hidden Markov Models

A special case of a graphical model is a hidden Markov model (HMM). Figure 2.6 shows a graphical model illustrating three time steps for a simple HMM. An HMM is used to represent a system that can be defined in terms of a sequence of states.

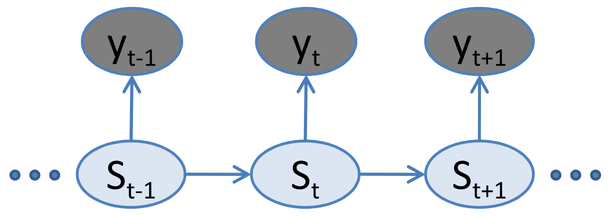


Figure 2.6: Three slices of the graphical model for a hidden Markov model. The state nodes “s” form a Markov chain and the “y” nodes are the observations.

For example, the daily activities of a person can be seen as a sequence of the states sleeping, eating, exercising, showering, and so forth.

HMMs often are used to model stochastic processes and the slices of the HMM are often defined at fixed time intervals; and the states are then indexed by discrete t values, for example the interval between values might be five minutes. Therefore, s_t could be the state of the system at the current time, s_{t-1} the state of the system five minutes ago, and s_{t+1} the state of the system five minutes from now. From one time slice to the next, the system either stays in the same state, or shifts to a new state. The tendency of the system to change states is captured in the state transition matrix, A

$$A = \begin{pmatrix} a_{00} & a_{01} & \cdots \\ a_{10} & a_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{2.12}$$

where a_{ij} is the probability of transitioning from state i to state j, and where each row of A sums to 1.

So in our example, the transition probability between the exercising state and the showering state might be larger than the transition from eating to showering. Since self-transitions indicate remaining in the same state, the self transition probability determines the expected length of an activity. For example, if state 0 is the sleeping state, $\frac{1}{1-a_{00}}$ is the expected number uninterrupted time slices in the sleeping state.

HMMs are used in applications where the state is hidden or unobserved, but a signal or measurement that reflects the state of the system is observed (represented by the "y" nodes in Figure 2.6). In order to make inferences about the state of the system given the observations, we define a model of the observations given each of the states (i.e. $p(y|s=0)$, $p(y|s=1)$, etc.). We will refer to the parameterization of this model as Φ for the remainder of this section.

A common use of the HMM is to infer the state sequence given the measurements. For example, given a time series of measurements of heart rate or brain activity [52], we might want to infer the most likely activity sequence of the individual (for example, eating began at 9:05, went to sleep at 11, etc.). Using an HMM to find the most likely sequence of states can be accomplished using the Viterbi algorithm [77, 59].

Two other common uses for HMMs are to find the likelihood of a sequence of observations given an HMM [59], and determining the model parameters that best explain the observations (commonly accomplished using the Baum-Welch algorithm [5, 59]).

In this thesis, we compute samples of a distribution modeled using an HMM. We can use the structure of the HMM to sample efficiently. In an HMM the Markov blanket (defined in Section2.3) of the state variable at time t, s_t, is s_{t-1} and s_{t+1}. From this, we take advantage of the fact that s_t is conditionally independent of all measurements and state variables from time $t=0$ to time $t=t-1$ given s_{t-1} to make a recursive algorithm that is linear in the length of the chain for each sampling step.

Inference in the HMM is accomplished efficiently using the forward-backward algorithm [4, 59]. The forward-backward algorithm involves computing an alpha function α_t from the beginning of the chain ($t=1$) sequentially at each slice to the end of the chain ($t=T$), and a beta function β_t from the end to the beginning.

The alpha function is defined as

$$\alpha_t = p(y_1, y_2, \ldots y_t, s_t|\Theta) \tag{2.13}$$

where Θ are the parameters of the HMM, $\Theta = [A, \Phi]$. This alpha function can be computed recursively

$$\begin{aligned}
\alpha_t &= \sum_{s_{t-1}} p(y_1, y_2, \ldots y_t, s_t, s_{t-1}|\Theta) \\
&= \sum_{s_{t-1}} p(y_t|s_t, y_1, y_2, \ldots y_{t-1}, s_{t-1}, \Theta) p(s_t|s_{t-1}, y_1, y_2, \ldots y_{t-1}, \Theta) p(s_{t-1}, y_1, y_2, \ldots y_{t-1}|\Theta) \\
&= p(y_t|s_t, \Theta) \sum_{s_{t-1}} p(s_t|s_{t-1}, \Theta) p(s_{t-1}, y_1, y_2, \ldots y_{t-1}, \Theta) \\
&= p(y_t|s_t, \Theta) \sum_{s_{t-1}} p(s_t|s_{t-1}, \Theta) \alpha_{t-1} \qquad (2.14)
\end{aligned}$$

since observation y_t is conditionally independent of all other variables given s_t, and s_t is conditionally independent of all variables from times 1 to $t-1$ given s_{t-1}.

The beta function is defined as

$$\beta_t = p(y_{t+1}, y_{t+2}, \ldots y_T|s_t, \Theta) \qquad (2.15)$$

which can similarly be written recursively

$$\begin{aligned}
\beta_t &= \sum_{s_{t+1}} p(y_{t+1}, y_{t+2}, \ldots y_T, s_{t+1}|s_t, \Theta) \\
&= \sum_{s_{t+1}} p(y_{t+1}|s_t, s_{t+1}, y_{t+2}, \ldots y_T, \Theta) p(y_{t+2}, \ldots y_T|s_t, s_{t+1}, \Theta) p(s_{t+1}|s_t, \Theta) \\
&= \sum_{s_{t+1}} p(y_{t+1}|s_{t+1}, \Theta) p(y_{t+2}, y_{t+3}, \ldots y_T|s_{t+1}, \Theta) p(s_{t+1}|s_t, \Theta) \\
&= \sum_{s_{t+1}} p(y_{t+1}|s_{t+1}, \Theta) p(s_{t+1}|s_t, \Theta) \beta_{t+1} \qquad (2.16)
\end{aligned}$$

For further details about HMMs see Rabiner [59].

Chapter 3

Modeling People-Counting Sensors

Chapter 1.1 described some of the characteristics of human activity that is measured by people-counting sensors. These time-series often reflect two behaviors: 1) the underlying hourly, daily, and weekly rhythms of natural human activity, and 2) bursty periods of unusual behavior corresponding to events.

In this chapter, a probabilistic framework is introduced that addresses the problem of identifying such events, by learning the patterns of both normal behavior and events from historical data.

3.1 Simple Event Models and Their Limitations

While at first glance the problem of separating event activity from normal behavior might seem relatively straightforward, the problem is quite difficult in an unsupervised context when events are not labeled or tagged in the data. This section examines the limitations of a simple approach that we will use as a baseline.

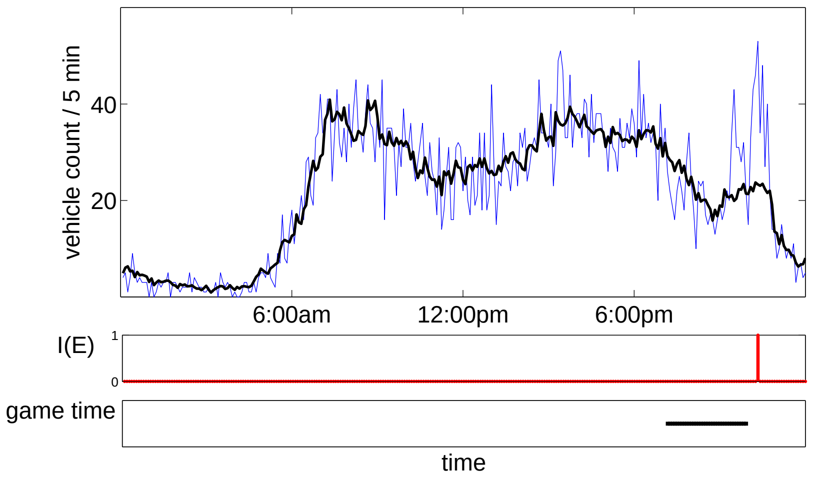


Figure 3.1: Example of an event successfully detected by a simple threshold method. In the top panel, the dark line is the historical average of flow data for weekdays used by the simple method to model "normal" flow. The light line is the observed vehicle count for one particular weekday, May 17, 2007. When the observed flow deviates sufficiently far from the historical average, the simple model predicts that an unusual event is taking place ($I(E) = 1$) as shown by the indicator function in the middle panel. The bottom panel shows the time period in which a baseball game took place at a stadium relatively close to the sensor (note: for this on-ramp sensor, event activity is expected around the end of the actual event time).

If all of the count measurements were generated by the underlying normal pattern; then the average count for each time period (given a history of measurements at the same time and day) would be an appropriate estimate for the normal pattern. Assuming that the count data follows the Poisson distribution, these time-segment specific count averages become the rate parameters in a non-homogenous Poisson process (see Chapter 2.1 for a discussion of NHPPs), we test the Poisson distribution assumption later in Section 3.5.

Detecting unusual events in the count data can then be performed using a simple threshold test based on a Poisson model for each time period. Specifically, the Poisson rate λ of a particular time and day can be estimated by averaging the observed counts on similar days (e.g., Mondays) at the same time of day, i.e., the maximum likelihood estimate,

$$\hat{\lambda}(t) = \frac{1}{N} \sum_{i=1}^{N} o_{it} \tag{3.1}$$

where t is the time period, o is an observed count, i is an index over weeks, and N is the total number of weeks.

Then, we detect an event of increased activity when the observed count, o_t, is sufficiently different than the average, as defined by a low (below-threshold) probability under the estimated Poisson distribution. So the event indicator function $I(E)$ is defined as

$$I(E) = \begin{cases} 1 & \text{if } \mathrm{P}(o_t; \hat{\lambda}(t)) < \epsilon \\ 0 & \text{otherwise;} \end{cases} \tag{3.2}$$

Figure 3.1 illustrates how this model could be used to find events. Here the threshold ϵ is set to 1×10^{-6} (the smallest threshold value that would detect the event). So if the count measurements are generated by a Poisson distribution, we would expect only 1 out of every 1 million measurements to *naturally* fall outside of the threshold. For the day shown in Figure 3.1, the event generated a large volume of "extra" traffic, making detection of the event elementary.

This baseline method ignores the presence of event activity (or periods of deviation from the underlying normal pattern) that corrupt the historical training data. For some data sets, this approach can be quite adequate—in particular, if the events interspersed in the data are sufficiently few compared to the amount of non–event observations, and if they are sufficiently noticeable in the sense that they cause a

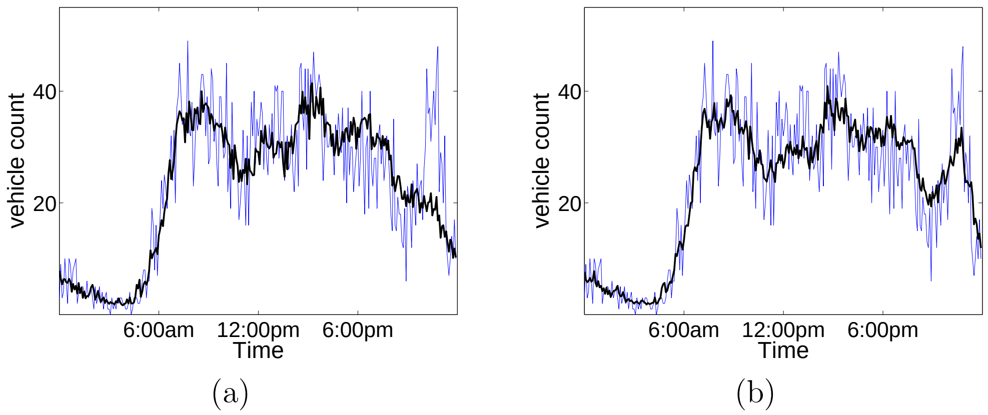


Figure 3.2: Example of freeway traffic data for Fridays for a particular on-ramp. (a) Ideal Model: Average time profile for normal, non game-day Fridays (dark curve) and observed measurements (light curve) for a particular Friday (6/10/05) with a baseball game that night. (b) Baseline Model: Average time profile over all Fridays (dark curve) superimposed on the same Friday data (light curve) as in the left panel.

dramatic change in activity. However, these assumptions do not always hold and we can observe several modes of failure in this baseline model.

3.1.1 Event Detection Challenges

One way the baseline model can fail is due to a "chicken-and-egg" problem. The historical data used to train the model are typically not labeled (i.e. each measurement marked according to whether or not an event was happening at the time of the measurement), so an unsupervised approach is necessary. If the historical data were labeled, the events could simply be removed and the normal pattern would emerge from the remaining data. Learning a model of normal behavior without labeled data would therefore ideally be accomplished by first detecting and then removing the abnormal events from the historical record; however, detecting the abnormal events can be accomplished reliably only by knowing the baseline of normal behavior — hence the "chicken-and-egg" problem.

An example of this chicken-and-egg problem is shown in Figure 3.2. Both panels plot vehicle counts every five minutes for an on-ramp on the 101 Freeway in Los Angeles (LA) located near Dodger Stadium, where the LA Dodgers baseball team plays their home games. The darker line in panel A shows the "ideal" or true normal profile, found by averaging the count for the set of "normal" Fridays when there were no baseball games. Labeled data are not commonly available, but in this case we have a schedule of home games for the Dodgers, and by filtering the game times out, we can find the ideal model of normal activity. The daily rhythm of normal Friday vehicle flow is clear from the data: little traffic in the early hours of the morning, followed by a sharp consistent increase during the morning rush hour, relatively high volume and variability of traffic during the day, another increase for the evening rush hour, and a slow decay into the night returning to light traffic.

The light line in both panels shows the counts for a particular Friday when there was a baseball game: the "event" can be seen in the form of significantly increased traffic around 10:00 P.M., corresponding to a surge of vehicles leaving from the baseball stadium. It is clear that relative to the average profile (the darker line) the baseball traffic is anomalous and should be relatively easy to detect.

Now consider what would happen if we did not know when the baseball games were being held. Since we normally do not have labeled data, the baseline model makes the assumption that the training set has no event activity. The dark line in the right panel is the baseline model of normal activity which is the average time-profile over *all* Fridays. Friday night is a popular time to schedule a game, and the average time-profile found when including both game-day and non game-day Fridays has an artificial bump on Friday nights at around 10:00 P.M.. The lighter line in the right panel shows the vehicle counts for the same Friday as the left panel. The attendance

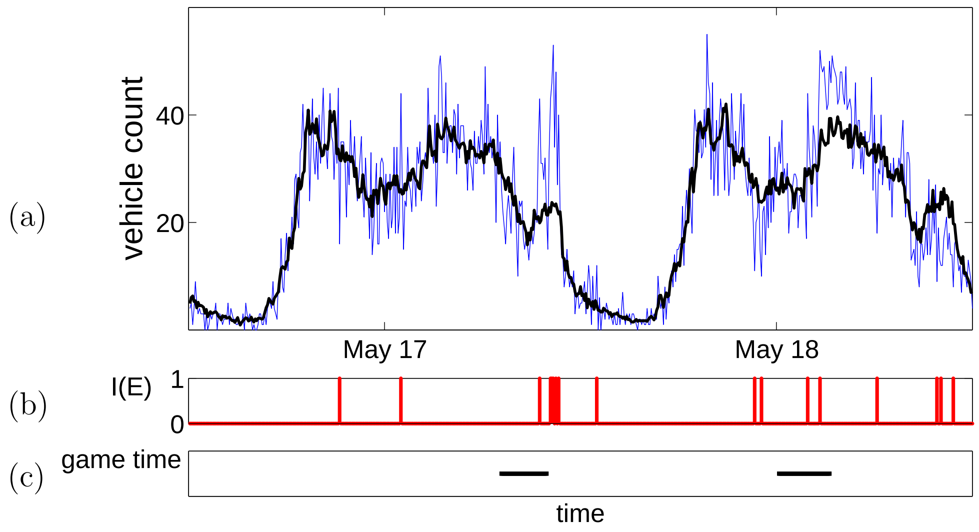


Figure 3.3: Illustration of the baseline threshold model set to detect the event on the second day, with (a) original freeway traffic time series (light curve) for May 17-18, and mean profile as used by the baseline model (dark curve), (b) events detected by the baseline method, and (c) ground truth (known events) in the bottom panel. Note the false alarms.

for the game on this particular night was relatively low compared to other game nights, and detecting the event signal is not as obvious.

A second type of failure occurs when there is a slight increase in traffic level which is not of sufficient magnitude to be noticed; however, the increase is *sustained* over a period of several observations signaling the presence of a persistent event. In Figure 3.3, the event indicated for the first day can be easily found by the threshold model by setting the threshold sufficiently high to detect the event but low enough so that there are no false alarms (as in Figure 3.1). In order for the threshold model to detect the event on the second day, however, the threshold must be increased, which also causes the detection of several false alarms over the two-day period. Anomalies detected by the threshold model are shown in the second panel of the figure, while known events (baseball games) are displayed in the third panel.

(a)

(b)

(c)

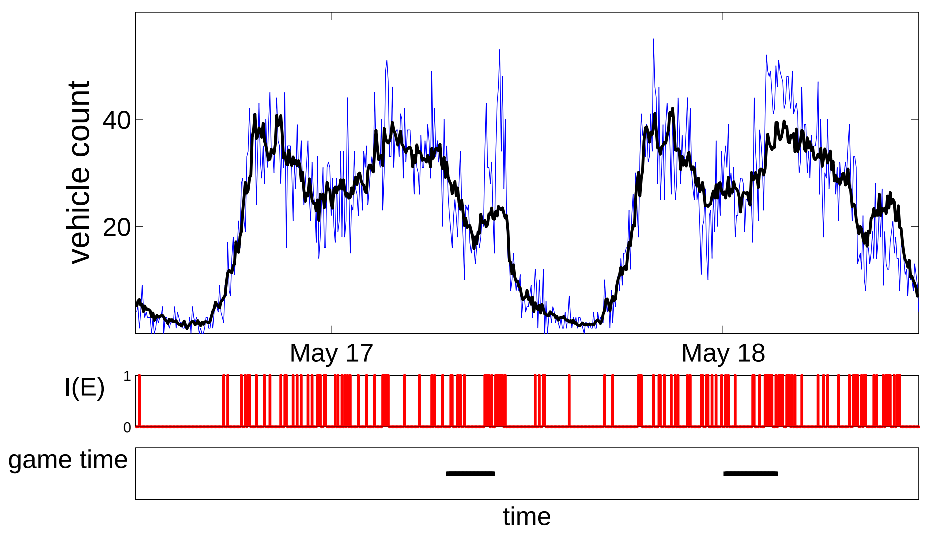


Figure 3.4: Panel (a) shows the traffic for the same day as in Figure 3.3. A higher threshold is used to detect the full duration of the large event on the second day causing many more false alarms (Panel (b)).

3.1.2 Event Quantification Challenges

The previous section examined challenges in the task of detecting the presence of an event; but there are other tasks of interest, such as determining event duration and event popularity, that add an additional layer of difficulty for the baseline model.

Consider the problem of capturing the duration of an event. In order to detect not only the presence of the event on the second day (Figure 3.3) but also its duration, the threshold must be raised to the point that the number of false alarms becomes prohibitive, as illustrated in Figure 3.4. Note that the traffic event, corresponding to people departing the game, begins at or near the end of the actual game time.

Another important task is quantifying event popularity. Suppose we wish to know not only that an unusual event occurred, but also a measure of the severity or popularity of the event. This type of query could be important for event planners or emergency response teams for example. During an event, the observed traffic count contains both event traffic and normal traffic (e.g. traffic count near a baseball stadium on a game

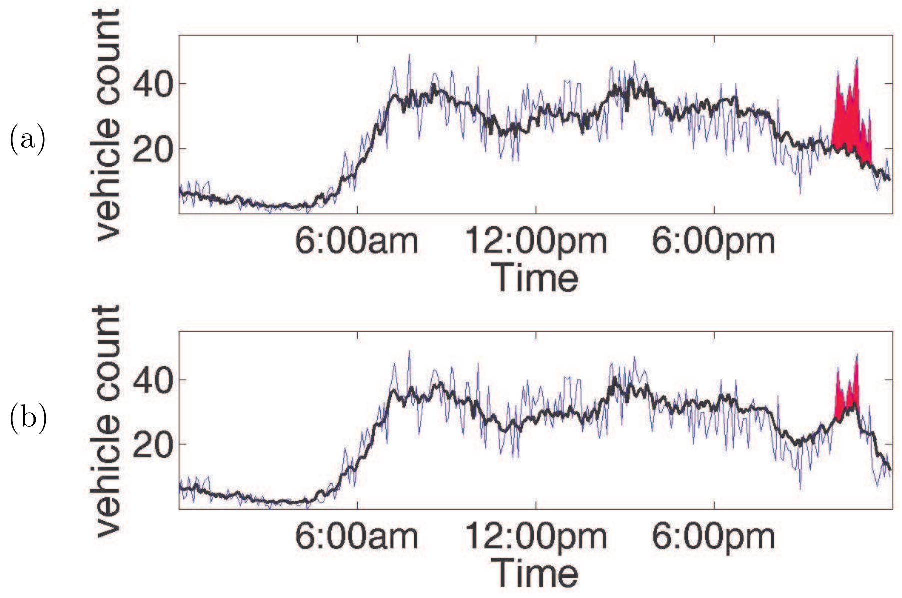
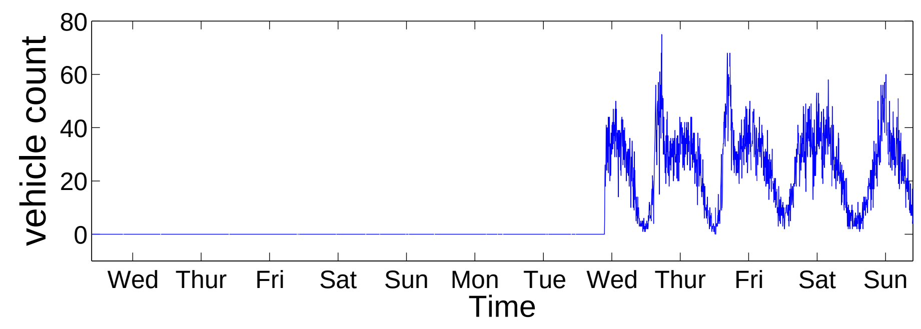


Figure 3.5: Illustration of the popularity estimation problem for the same day as in Figure 3.2. The red shaded region between the observed count (light blue line) and the model's estimation of normal traffic (dark thick line) reflects the popularity or severity of the event for the ideal model (a) and the baseline model (b). Note the under-counting of the baseline model.

night contains both people attending the game and people going home from work who are not attending the game). A natural way to measure event popularity, then, is to look at the difference between the observed count and the expected normal count over the duration of the event. This is a difficult task for the baseline method as illustrated in Figure 3.5. Figure 3.5(a) shows how the "ideal" model would judge the popularity of the event. Figure 3.5(b) shows that the baseline model would significantly under-count the event traffic. As discussed earlier, the presence of large events distorts the estimated rate of "normal"behavior, increasing to create the artificial bump seen in Figure 3.5(b); this causes the baseline model to miss the presence of other events or underestimate their popularity.

So even if the baseline is able to detect the event (limitation illustrated in Figures 3.2 and 3.3), and also is able to detect the duration of the event (limitation illustrated

(a)


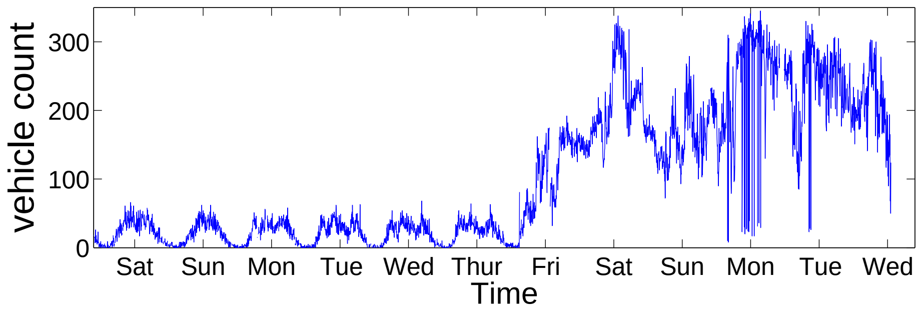
(b)


Figure 3.6: Examples of sensor failure. (a) shows a sensor with a stuck-at-zero failure while (b) shows a large corrupted signal.

in Figure 3.4), the corruption of the event activity that is included in the baseline model's prediction of normal traffic can cause large errors in its estimation of the event traffic (Figure 3.5).

3.1.3 Sensor Failure Challenge

The previous sections illustrated limitations of the baseline model in the context of sensors with relatively clean data. In reality, sensors commonly experience periods of failure where corrupted measurements are reported. Figure 3.6 shows two examples of sensor failure. Figure 3.6(a) shows several days of a stuck-at-zero failure followed by normal activity. Figure 3.6(b) shows several normal days followed by a rather large corrupted signal.

Figure 3.2 showed how the presence of event activity in the training data could artificially alter the prediction of the underlying normal activity in the baseline model. Sensor failure of the type illustrated in Figure 3.6 would have a much more drastic adverse effect on the baseline's estimation of normal activity.

We will see later in the thesis that by extending our probabilistic model, we can also deal with corrupted data.

3.2 MMPP: A Probabilistic Event Model

Our task is both to learn the normal behavior (intuitively corresponding to the periodic portion of the data) and to detect the presence and duration of unusual events (relatively rare deviations from the norm). The baseline method discussed in Section 3.1 approaches these tasks in two steps. First the normal pattern is learned (ignoring the corruption of the signal in the historical train set caused by events), and then the events are detected. Chapter 1, and in particular the issues raised in Section 3.1, however, motivate the use of a probabilistic model capable of reasoning simultaneously about the rate of normal behavior and the presence of events.

We assume that the two processes are additive, so that

$$o_t = n_t + e_t, \qquad o_t \geq 0 \tag{3.3}$$

where o_t is the observed (measured) count at time-segment t, n_t is the number of occurrences attributed to the underlying normal process, and e_t represents the change in the number of occurrences which is attributed to an event at time t; the non-negativity condition indicates that we cannot observe fewer than zero counts.

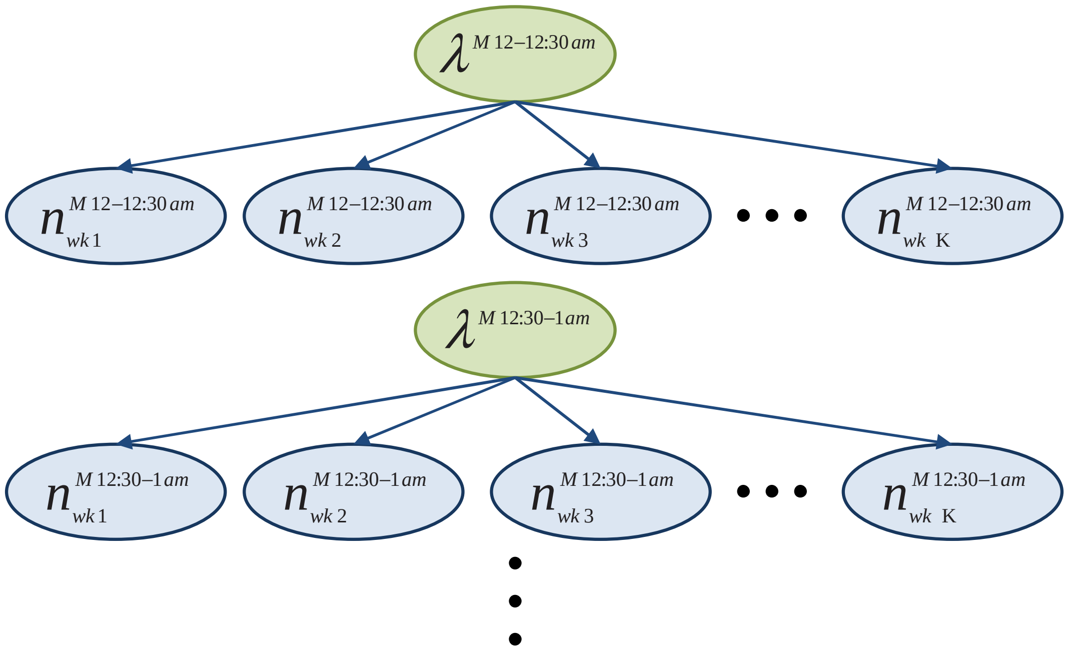

$\lambda^{M\ 12\text{–}12{:}30\,am}$
$n^{M\ 12\text{–}12{:}30\,am}_{wk\,1}$
$n^{M\ 12\text{–}12{:}30\,am}_{wk\,2}$
$n^{M\ 12\text{–}12{:}30\,am}_{wk\,3}$
$n^{M\ 12\text{–}12{:}30\,am}_{wk\ K}$
$\lambda^{M\ 12{:}30\text{–}1\,am}$
$n^{M\ 12{:}30\text{–}1\,am}_{wk\,1}$
$n^{M\ 12{:}30\text{–}1\,am}_{wk\,2}$
$n^{M\ 12{:}30\text{–}1\,am}_{wk\,3}$
$n^{M\ 12{:}30\text{–}1\,am}_{wk\ K}$

Figure 3.7: Graphical model showing the temporal connection of the rate parameters. The rate parameters (λ) are connected to observations that fall in the same time-of-day and day-of-week, but are not connected to any other rate parameter or measurement, forming independent models.

The following sections discuss modeling each of the variables n_t and e_t in turn. Note that, although the models described here are defined for discrete time periods, it may also be possible to extend them to continuous time measurements [65, 66].

3.2.1 The Normal Count: Modeling Periodic Count Data

To model the periodic, predictable portion of behavior corresponding to normal activity (n_t), we use a nonhomogeneous Poisson process as discussed in Chapter 2.1.

For any given time-segment, t, the count generated by the normal process is assumed to come from a Poisson distribution

$$n_t \sim \mathrm{P}(n_t; \lambda(t)) \tag{3.4}$$

where the parameter $\lambda(t)$ represents the rate, or average number of occurrences in a fixed time interval.

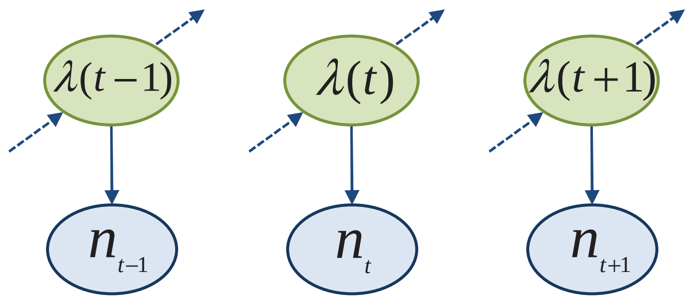


Figure 3.8: Graphical model showing three consecutive time-segments of the normal count process. Note that there are no dependencies between neighboring counts.

A Poisson rate parameter can be learned for each unique time-segment during the week, or similar days can share rate parameters. If the normal pattern for each day is different, a unique rate parameter should be learned for each time-of-day and day-of-week combination. For a system with a 30-minute bin size, there would be 336 independent rate parameters (λ). The normal process in this case can be thought of as a set of 336 models. As Figure 3.7 illustrates, the rate parameter for each time-segment is learned only using historical measurements falling in the corresponding day of week and time of day.

It is often beneficial to share information between similar days. Weekday traffic is usually different from weekend traffic, but the differences between one weekday and another weekday are often small. So instead of 48 rate parameters for each of the 7 days of the week (in the 30-minute time-segment case), 48 general weekday and 48 weekend day parameters are learned. Another benefit of reducing the number of rate parameters from 336 to 96, is that fewer training data are needed.

Even with sharing information between weekdays or weekends; however, there is still no dependence relationship between neighboring time-segments as illustrated in Figure 3.8. The dotted arrow through the rate parameter is a reminder that the rate parameters do have a dependence relationship to other measurements in time (Figure 3.7), though not to neighboring time-segments. The assumption that neighboring

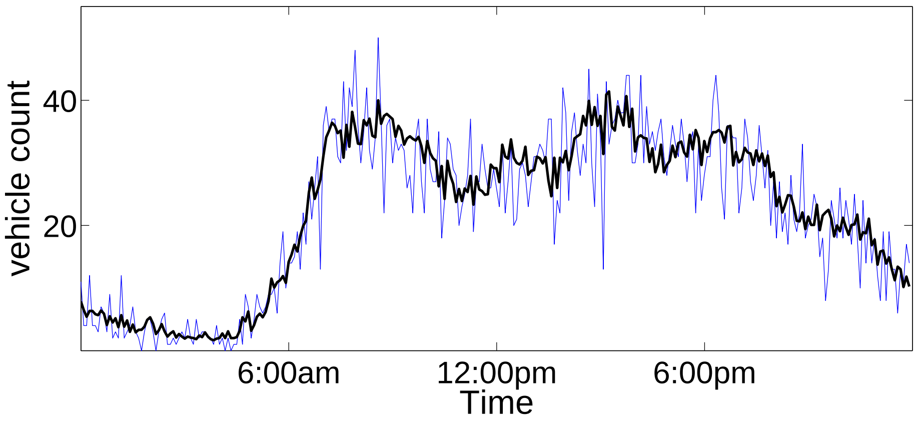


Figure 3.9: A day with no apparent event activity.

rate parameters are independent is not accurate - for example in the middle of the night, knowing that the previous three rate parameters are near zero indicates that the current rate parameter is probably also near zero. However, this assumption reduces the computational cost, as the dependency between neighboring rate parameters is not explicitly modeled. Also, as the size of the training set increases, the importance of a link between neighboring rate parameters decreases. In addition, the smoothing of the normal process profile ($\lambda(t)$) caused by linking neighboring rate parameters could cause certain phenomena to be missed, such as the natural oscillation discovered in some sensor measurements discussed in Chapter 4.3. If smoothing is important for a certain application, the links between neighboring rate parameters could be added [75].

A second assumption implied by the graphical model in Figure 3.8 is that neighboring observed counts are independent given the rate parameters. This assumption does appear to be valid. Figure 3.9 shows a day where there does not appear to be any event activity. The light blue line in the figure is the observed count (o_t) and the dark black line is the time-varying rate parameter ($\lambda(t)$). From Equation 3.3, if there is no event activity then $o_t = n_t$, i.e. the observed count is coming entirely from the normal process. So for this particular day, the light blue line can be seen as the normal count, n_t, at each of the 5-minute time-segments. The blue line appears to be a noisy sample of the thick black line. So knowing that the normal count in the previous time-segment is larger or smaller than the rate parameter for that time-segment does not appear to help in predicting the value of the current normal count.

By choosing conjugate prior distributions for the rate parameters we can ensure that the inference computations in Section 3.3 have a simple closed form:

$$\lambda \sim \Gamma(\lambda; a^N, b^N) \tag{3.5}$$

where λ implicitly represents $\lambda(t)$, and where Γ is the Gamma distribution with parameters a^N and b^N. The superscript N indicates the normal process to distinguish these hyperparameters from those of the event process. And

$$\Gamma(\lambda; a, b) \propto \lambda^{a-1} e^{-b\lambda} \tag{3.6}$$

(See Chapter 2.2 for more information on Bayesian inference and conjugate priors).

3.2.2 The Event Count: Modeling Rare, Persistent Events

In the data examined in this thesis, the anomalous measurements can be intuitively thought of as being due to relatively short, rare periods in which an additional random process changes the observed behavior, increasing or decreasing the number of observed counts. In this section we will only consider events that increase activity (assuming no "negative" events where fewer counts than expected are observed) which is applicable in some cases and is also simpler to introduce. Models with additional event states will be described in Chapter 4.

When an event occurs, it contributes to the observations (e.g., people arriving for an event) for a relatively short, continuous period of time. To model this behavior, we use a binary process z_t to indicate the presence of an event:

$$z_t = \begin{cases} 1 & \text{if there is an event at time } t \\ 0 & \text{otherwise;} \end{cases} \tag{3.7}$$

and define the probability distribution over z_t to be Markov in time, with transition probability matrix:

$$M = \begin{pmatrix} z^{00} & z^{01} \\ z^{10} & z^{11} \end{pmatrix} \tag{3.8}$$

where:

z^{00} is the probability of remaining in the non-event state given the previous state is also non-event.

z^{01} is the probability of transitioning from the non-event to the event state, and $z^{01} = 1 - z^{00}$.

z^{11} is the event state self-transition probability.

z^{10} is the probability of going from an event to non-event state, and $z^{10} = 1 - z^{11}$.

The number of time-segments between events is geometric in distribution with expected value $1/z^{01}$ (in units of time-segments), and the length of each event is geometric with expected value $1/z^{10}$. We give z^{00} and z^{11} priors

$$z^{00} \sim \beta(z^{00}; a_{00}, b_{00}) \qquad\qquad z^{11} \sim \beta(z^{11}; a_{11}, b_{11}) \tag{3.9}$$

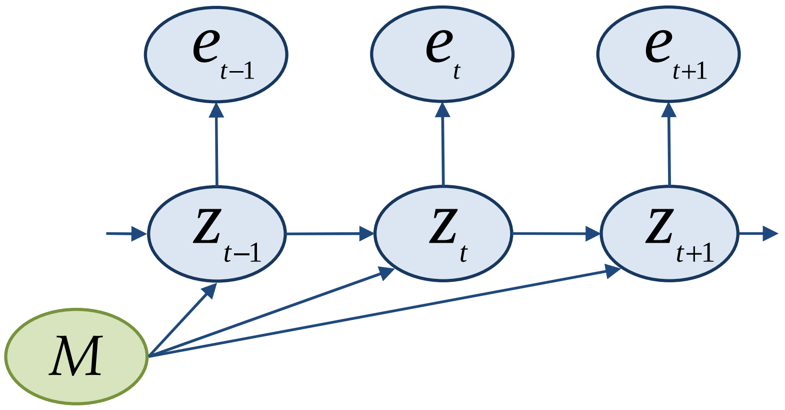


Figure 3.10: Three consecutive time-segments of the event count process. z_t is the event state, e_t is the event count, and M is the matrix of event transition parameters.

where $\beta(\cdot)$ is the Beta distribution, and a_{00}, b_{00}, a_{11}, and b_{11} are parameters for the two Beta priors.

The event process is modeled as an HMM (see Chapter 2.3.1 for a description of HMMs), and is illustrated in Figure 3.10. Given z_t, we can model the increase in observation counts due to the event, e_t, as Poisson with rate γ

$$e_t \sim \begin{cases} 0 & z_t = 0 \\ \mathrm{P}(e;\gamma) & z_t = 1 \end{cases} \tag{3.10}$$

and γ is independent at each time t

$$\gamma \sim \Gamma(\gamma; a^E, b^E). \tag{3.11}$$

where a and b are parameters of the Gamma distribution and the superscript E indicates the event process.

Just as there are different Poisson rate parameters for different times-of-day in the normal process, one can imagine different Poisson rate parameters for the event process based on time. However, although the observed time and day indicate the appropriate normal process rate parameter to use (e.g. traffic Monday mornings at 11am at a specific sensor location is generally predictable), the same cannot be said for the event process (event traffic on Monday morning at 11am might be due to a small event with relatively little extra traffic or due to a large event with more traffic).

One method of approaching this problem of different sized events would be to define several event types and let the model learn the appropriate rate parameters for each type (perhaps extra traffic due to large sporting events has one distribution and extra traffic due to concerts has a different distribution); however, not only does this increase computational complexity, it could make the model overly flexible and too sensitive to naturally occurring noise in the count measurements. A more robust solution for data sets with a variety of event sizes is to marginalize over γ analytically

$$\int \mathrm{P}(e;\gamma)\Gamma(\gamma;a^E,b^E) = \mathrm{NBin}(e;a^E,b^E/(1+b^E)) \tag{3.12}$$

where NBin is the negative binomial distribution with parameters a^E and $b^E/(1+b^E)$.

The negative binomial distribution has a larger variance than the Poisson distribution, and is thus able to model a "catch-all" event type that gives some probability mass to event counts from various-sized event types.

3.2.3 MMPP

Figure 3.11 shows the graphical model that represents the distribution of the combined event and normal processes. The normal process illustrated in Figure 3.8 and the event process illustrated in Figure 3.10 are combined using the deterministic relationship , $o_t = n_t + e_t$. Note that the only variables that are known are the observed counts. The remaining parameters and data variables are unknown and must be learned and inferred by the model.

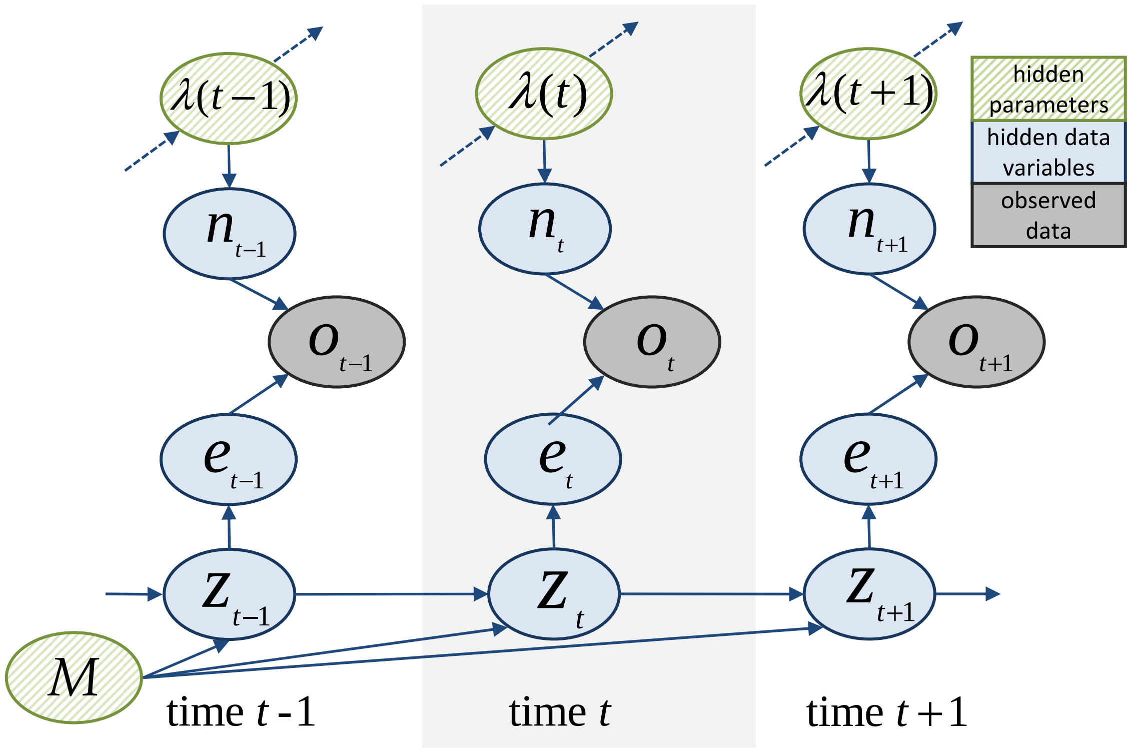


Figure 3.11: The event model, a Markov-modulated Poisson process. $\lambda(t)$ is the time-varying rate parameter for the normal process, n_t is the normal count, o_t is the observed count, z_t is the event state, e_t is the event count, and M is the transition matrix for the event process.

At every time-segment t, the normal process contributes a normal count n_t according to a Poisson distribution with rate parameter $\lambda(t)$. On rare occasions when an unusual event is taking place, an additional Poisson count e_t is added due to the event traffic. This type of gated Poisson contribution, called a Markov-modulated Poisson model, is a common component of network traffic models [29, 66].

The following is a walk-through of this generative model, i.e. the process by which the data variables and observed counts could be sampled if, instead of knowing the observed counts and learning the model parameters, we had the opposite situation – knowing the parameters for the normal and event process and wanting to generate observed counts. For time t:

- Sample the normal count, n_t, from a Poisson distribution with rate parameter $\lambda(t)$ ($n_t \sim \mathrm{P}(n_t; \lambda(t))$. For example, a normal count of 17 might be sampled given a rate parameter of 15.7.

- Sample the current event state, z_t, given the previous event state, z_{t-1}, and the event transition matrix parameters (Equation 3.8). For example if the previous state was no event, $z_{t-1} = 0$, the non-event state will be sampled with probability z^{00} and the event state will be sampled with probability $1 - z^{00}$.

- If the non-event state is sampled, $z_t = 0$, then $n_t = 0$ and then $o_t = n_t = 17$ in this case.

- If the event state is sampled, $z_t = 1$, then an event rate would be sampled [1] $\gamma_t \sim \Gamma(\gamma; a^E, b^E)$ (for example, $\gamma_t = 7.3$), which would be used to sample the event count, $e_t \sim \mathrm{P}(e_t; \gamma_t)$ (for example, $e_t = 9$). Then the observed count would be the sum of the normal and event counts, $o_t = n_t + e_t = 17 + 9 = 26$).

In our applications we are specifically interested in detecting the periods of time in which our event process z_t is active, using the associated count e_t to provide information about the event's popularity. At other times we are interested in using the rate parameter of the normal process $\lambda(t)$ for imputing missing data or as a feature for classifying the type of traffic used by this sensor; for example in Chapter 8 we discuss a problem connecting land use to the normal profile of traffic at the location of the sensor. For these tasks we must use the observed data to reason about the hidden parameters and data variables.

[1] A more complex model would make the γ_t, γ_{t+1}, γ_{t+2}, $\ldots$ be dependent throughout the event, but we did not find it necessary to model this dependence in this work.

3.3 Learning and Inference with the MMPP

Since the model has temporal structure both from $\lambda(t)$ and z_t, loops are created in the graphical model illustrated in Figure 3.11, making inference more difficult. Although exact inference is computationally prohibitive, approximate algorithms using Markov chain Monte Carlo (MCMC) methods [23, 21], such as Gibbs sampling (see Chapter 2.2.1 for an overview of Gibbs Sampling), work well in practice.

Given the complete data $\{n_t, e_t, z_t\}$, it is straightforward to compute maximum a posteriori (MAP) estimates or draw posterior samples of the parameters $\lambda(t)$ and $\{z^{00}, z^{11}\}$, since they are conditionally independent.

We can thus infer posterior distributions over each of the variables of interest using MCMC. Specifically, we iterate between drawing samples of the hidden variables $\{z_t, n_t, e_t\}$ (described in Section 3.3.1) and the parameters given the complete data (described in Section 3.3.2). The complexity of each iteration of MCMC is $\mathcal{O}(T)$, linear in the length of the time series. Experimentally we have found that the sampler converges quite rapidly on the data sets used in this thesis, where convergence is informally assessed by monitoring the parameter values. For both data sets used in this chapter, 10 burn-in iterations followed by 50 more sampling iterations appeared sufficient. An example of an empirical convergence plot is seen in Figure 3.12.

In practice, on a 3GHz desktop machine in Matlab, the building data (15 weeks of 30-minute intervals) took about 3 minutes while the traffic data (25 weeks of 5-minute intervals) took about one hour for all 60 iterations. The samples obtained from MCMC can be used not only to provide a point estimate of the value of each parameter (for example, its posterior mean) but also to gauge the amount of uncertainty about that value. If this degree of uncertainty is not of interest (for example, if the data

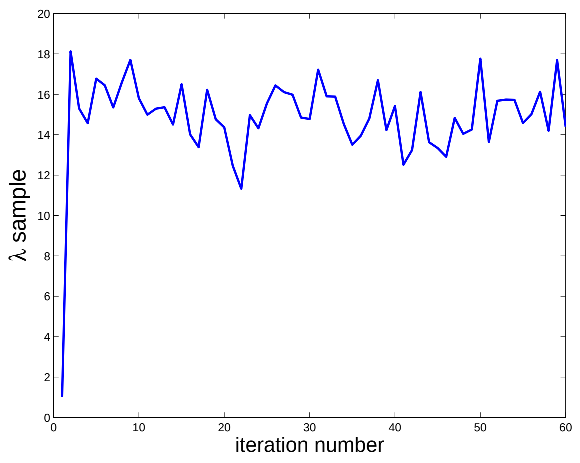


Figure 3.12: MCMC convergence is assessed empirically by monitoring the parameter values at each iteration of the sampler. In this case, the sampler appears to reach the stationary distribution after the first iteration. The parameter monitored here is the rate parameter for the Wednesday, 5 to 5 : 30PM time slot for the building data set described in Section 3.4.

are sufficiently many that the uncertainty is very small) we could use an alternative method such as expectation–maximization (EM) to learn the parameter values [11].

3.3.1 Sampling Hidden Data Variables Given Parameters

The graphical model in Figure 3.13 represents the distribution that is used to sample the hidden data variables when the parameters, $\lambda(t)$ and M, are held constant. Holding the parameters constant decouples the model in time at the rate parameters, removing the cycles present in Figure 3.11, and making inference easier.

In the following description of the sampling procedure, variables with a "hat" indicate that the variable is being held constant at a sampled value.

Conditioned on specific values of $\lambda(t)$ and the transition probability matrix M, it is relatively straightforward to draw a sample sequence z_t using a variant of the forward–backward algorithm [5] as described in Chapter 2.3.1.

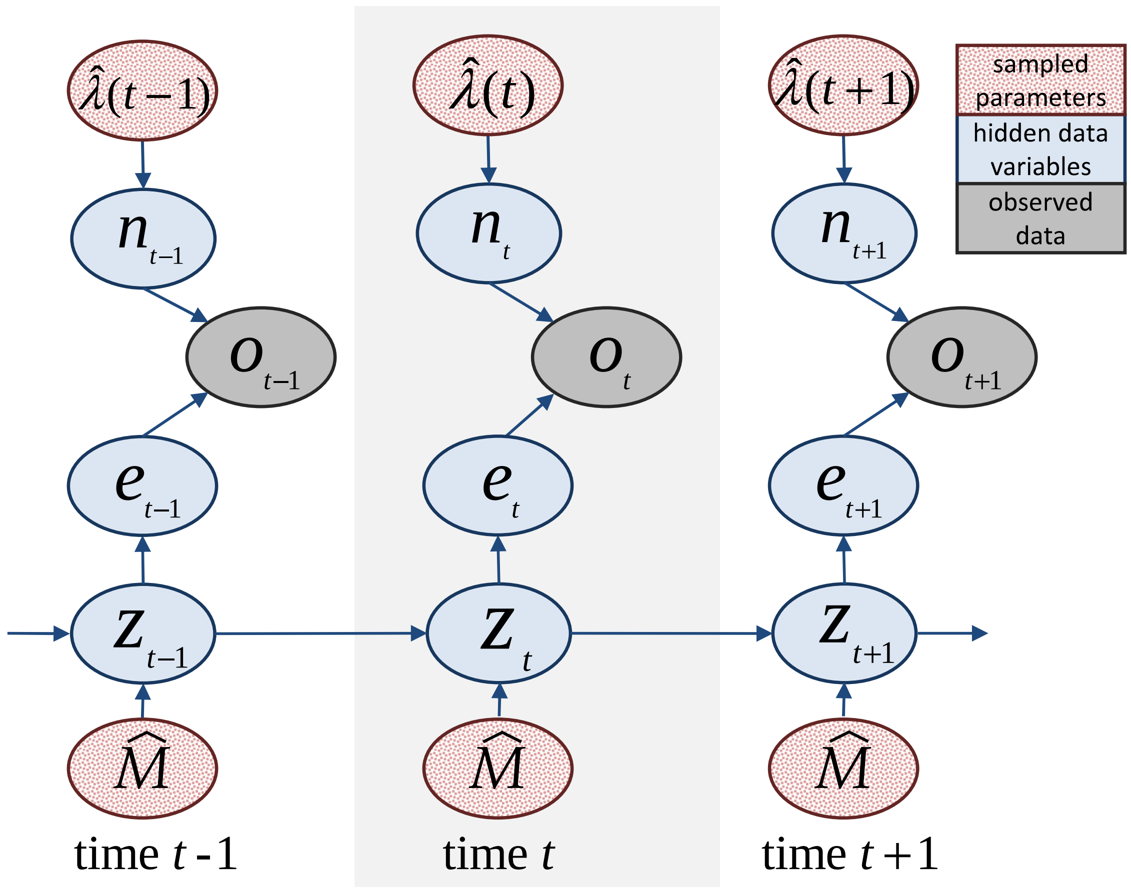


Figure 3.13: The parameters, shaded in red and marked with a hat, are held constant at their sampled values of the previous iteration (Figure 3.14). The data variables (light blue) can now be sampled given the parameters and the observed data. Note that holding the parameters constant eliminates the cycles seen in Figure 3.11 making inference easier.

Specifically, in the forward pass we compute, for each $t \in \{1, \ldots, T\}$, the conditional distribution $p(z_t|\{o(t'), t' \le t\})$ using the likelihood functions

$$p(o_t|z_t) = \begin{cases} \mathrm{P}(o_t; \hat{\lambda}(t)) & z_t = 0 \\ \sum_i \mathrm{P}(o_t - i; \hat{\lambda}(t)) \, \mathrm{NBin}(i) & z_t = 1 \end{cases} \tag{3.13}$$

where $\hat{\lambda}(t)$ are the normal process rate parameters sampled in the previous iteration (Section 3.3.2) and the parameters of $\mathrm{NBin}(\cdot)$ are as in Equation 3.12.

Then, for $t \in \{T, \ldots, 1\}$, we draw samples

$$\hat{z}_t \sim p(z_t|\hat{z}_{t+1}, \{o(t'), t' \le t\}). \tag{3.14}$$

Given the sample $\hat{z}_t$, we can then sample n_t and e_t. If $\hat{z}_t = 0$, we simply take $\hat{n}_t = o_t$; if $\hat{z}_t = 1$ we draw $\hat{n}_t$ from the discrete distribution

$$\hat{n}_t \sim f(n_t, o_t; \lambda(t), a^E, b^E) \propto \mathrm{P}(n_t; \lambda(t)) \, \mathrm{NBin}(o_t - n_t; a^E, b^E/(1 + b^E)) \tag{3.15}$$

then set $\hat{e}_t = o_t - \hat{n}_t$.

When o_t is unobserved (missing), n_t and e_t are coupled only through z_t and the positivity condition on o_t. Thus, $\hat{n}_t$ and $\hat{e}_t$ can be drawn independently. Missing data are relatively rare for the data sets tested in this chapter, with essentially no observations missing in the building data and about 7% of observations missing in the traffic data (due to loop sensor errors or down-time).

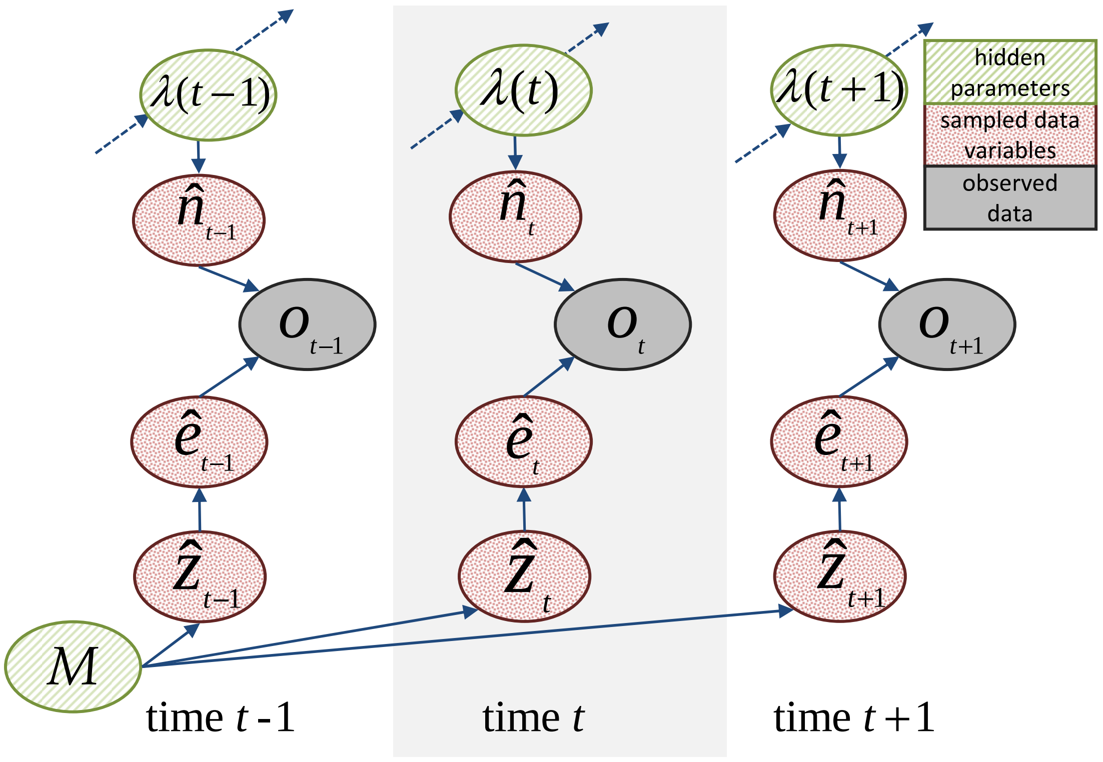


Figure 3.14: The data variables, shaded in red an marked with a hat, are held constant at their sampled values of the previous iteration (Figure 3.13). The parameters (light green) can now be sampled given the data variables and the observed data. Note there is no longer time dependence through the state variables (z_t).

3.3.2 Sampling Parameters Given the Complete Data

The graphical model in Figure 3.13 represents the distribution that is used to sample the model parameters when the data variables are held constant. The fixed data variables make the normal process rate parameters $\lambda(t)$ conditionally independent of the event process parameters M, thus they can be sampled independently and in any order.

The complete data likelihood is

$$\prod_t \mathrm{P}(n_t; \lambda(t)) \prod_t p(z_t|z_{t-1}) \prod_{z_t=1} \mathrm{NBin}(e_t; a^E, b^E/(1+b^E)) \tag{3.16}$$

Due to the conditional independence of the model parameters, each term can be sampled from independently. Consider the first term, which only involves $\lambda(t)$:

$$\prod_t \frac{e^{-\lambda(t)}\lambda(t)^{n_t}}{n_t!} \propto e^{-T\lambda(t)}\lambda(t)^{\sum n_t} \tag{3.17}$$

By virtue of choosing the conjugate prior distribution (Equation 3.5), the posterior is a distribution of the same form, but with parameters given by the sufficient statistics of the data (see Chapter 2.2 for a discussion about the benefits of using conjugate distributions). The rate parameters of the normal process, then, can be sampled from the posterior distribution:

$$\hat{\lambda}(t) \sim \Gamma(\lambda(t); a^L + \sum n_t, b^L + T) \tag{3.18}$$

Sampling the transition matrix parameters $\{z^{00}, z^{11}\}$ is similarly straightforward—we compute

$$Z^{ij} = \sum_{t=1}^{T-1} \delta(z_t = i \ \&\& \ z_{t+1} = j) \qquad \text{for } i = 0,1, \ j = 0,1 \tag{3.19}$$

where $\delta(z_t, z_{t+1}) = 1$ if $z_t = i$ and $z_{t+1} = j$ and $\delta(z_t, z_{t+1}) = 0$ otherwise; to obtain posterior distributions

$$z^{00} \sim \beta(z; a^Z_{00} + Z^{00}, b^Z_{00} + Z^{01}) \qquad z^{11} \sim \beta(z; a^Z_{11} + Z^{11}, b^Z_{11} + Z^{10}) \tag{3.20}$$

3.3.3 Parameter Settings

As noted by Scott [66], Markov–modulated Poisson processes appear to be relatively sensitive to the selection of prior distributions over the z^{ij} and γ, perhaps because there are no direct observations of the processes they describe. This appears to be particularly true for our model, which has considerably more freedom in the anomaly process (i.e., in γ) than the telephony application of Scott [66]. However, for an event detection application such as those under consideration, we have fairly strong ideas of what constitutes a "rare" event, e.g., approximately how often we expect to see events occur (say, 1–2 per day) and how long we expect them to last (perhaps an hour or two). We can leverage this information to form relatively strong priors on the transition parameters of z_t, which in turn influences the marginal probability of z_t. This avoids over-explanation of the data, such as using the event process to compensate for the fact that the "normal" data exhibits slightly larger than expected variance for Poisson data. By adjusting these priors one can also increase or decrease the model's sensitivity to deviations and thus the number of events detected.

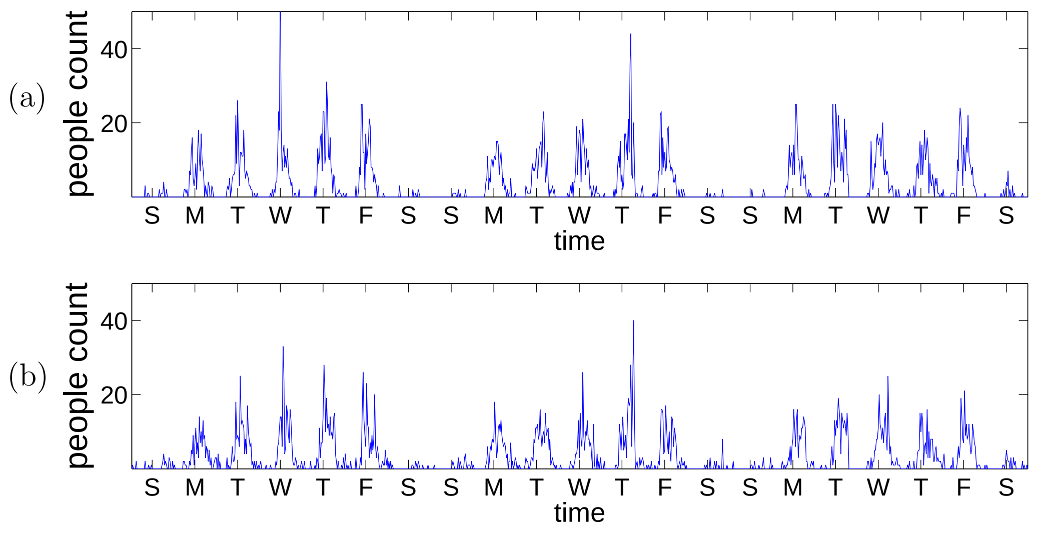


Figure 3.15: (a) Entry data for the main entrance of the Calit2 building for three weeks, beginning 7/23/05 (Sunday) and ending 8/13/05 (Saturday). (b) Exit data for the same door over the same time period.

3.4 Case Study 1: One Building Sensor and One Traffic Sensor

This first case study is a proof of concept for the probabilistic framework developed in this thesis. In this study, the effectiveness of the most basic model described in this thesis (the two-event state MMPP described in Section 3.2) is demonstrated by comparing it against the baseline method described in Section 3.1. A people count sensor and a vehicle count sensor with relatively clean data (i.e., no large periods of missing measurements or sensor failure) are used as test data, and results are validated using known event lists for each data set.

3.4.1 Data Set Characteristics

We use two different data sets in this case study to test our approach. In this section we describe these data sets in more detail.

The first data set will be referred to as the *building* data, consisting of fifteen weeks of count data automatically recorded every 30 minutes at the front door of the Calit2 building on the University of California, Irvine campus. The data are generated by a pair of battery–powered optical detectors that measure the presence and direction of objects as they pass through the building's main set of doors. The number of counts in each direction are then communicated via a wireless link to a base station with internet access, where they are stored.

We conducted experiments to compare the measured sensor counts with manually-generated hand counts, and the results indicated that the sensor counts are generally very accurate, with a small amount of systematic under-counting when individuals or objects enter or exit at the same time. In addition to the uncertainty due to imperfections in the data collection process, we also have the issue that the sensor covers only the building's main doors; there are several other entrances which were not monitored at the time these data were collected [2]. Nonetheless, the sensor provides a useful measurement of the current level of pedestrian traffic going in and out of the building via the main door.

The observation sequences ("door counts") acquired at the front door form a noisy time series with obvious structure but many outliers (see Figure 3.15). The data are corrupted by the presence of events, namely, non-periodic activities which take place in the building and (typically) cause an increase in foot traffic entering the building before the event, and leaving the building after the event, possibly with some "churn" (people going in and out) during the event. Some of these events can be seen easily in the time–series, for example the two large spikes in both entry and exit data on days four and twelve (first Wednesday and second Thursday) in Figure 3.15. However,

[2] Later in Chapter 6 we discuss data from different doors that were not yet monitored at the time of this study.

many of these events may be less obvious and only become visible when compared to the behavior over a long period of time.

The second data set will be referred to as the *freeway traffic data* and consists of estimated vehicle counts every 5 minutes over 6 months from an inductive loop–sensor located on the Glendale on-ramp to the 101-North Freeway in Los Angeles [20]. Figure 3.16 shows the temporal pattern for a particular week starting with Sunday morning and ending Saturday night. The daily rhythms of traffic flow are clearly visible as is the distinction between weekdays and weekends. Also visible are "local excursions" corresponding to significantly different counts compared to relatively smooth normal pattern, such as the baseball games on Sunday afternoon and every evening except Thursday. The lower panel of Figure 3.16 shows a set of known (ground truth) events for this data (which will be unknown to the model and only used for validation) corresponding to the dates and times of baseball games. Note that since we monitor the freeway on-ramp, events in the data correspond to traffic leaving at the end of a baseball game when attendees leave the stadium and get on the freeway—thus, the baseball game is reflected by a signature in the data that will tend to lag in time from that of the game itself.

Most of the events present in the data are characterized by increased activity, but some (particularly holidays) exhibit a noticeable decrease in activity. For this case study we confine our discussion to a "positive" model capturing only increases in activity, and to avoid significant biases in this analysis we remove the observations on known holidays (about 1-3 days in each data set). An extension to the model discussed in Chapter 4.1 will allow for "negative" events and will remove the need for the filtering step of removing holiday measurements.

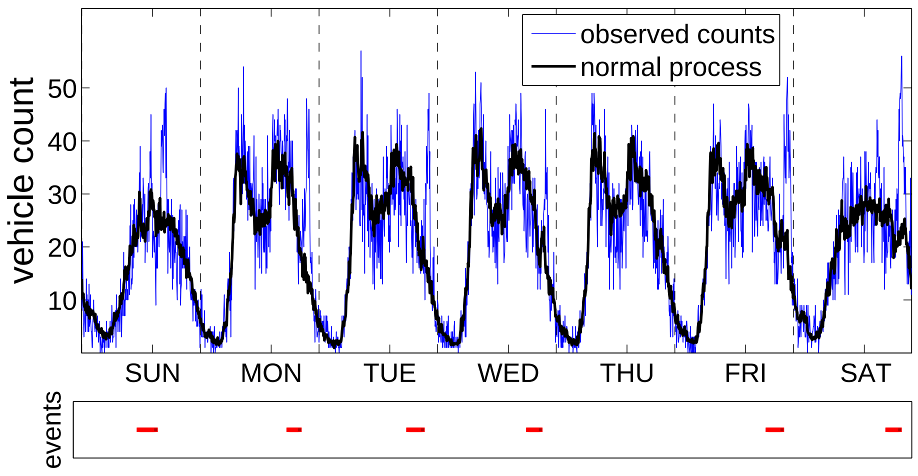
(a)
(b)


Figure 3.16: (a) One week of traffic data (light curve) from Sunday to Saturday (June 5-11), with the estimated normal traffic profile (estimated by the MMPP model described in the Section 3.2) superposed as a dark curve. (b) Ground truth list of events (baseball games).

3.4.2 Results

One of the primary goals in our application is to automatically detect the presence of unusual events in the observation sequence. The presence or absence of these events is captured by the process z_t, and thus we may use the posterior probability $p(z_t = 1|o_{1:T})$ as an indicator of when such events occur.

Given a sequence of data, we can use the samples drawn in the MCMC procedure (Section 3.3) to estimate the posterior marginal distribution over events

$$p(z_t) = \frac{1}{S}\sum_{s=1}^{S} z_{ts} \tag{3.21}$$

where S is the total number of samples and s is an index over the samples.

For comparison to a ground truth of the events in the building data set, we obtained a list of the events which had been scheduled over the entire time period from the building's event coordinator. For the freeway traffic data set, the game times for 78

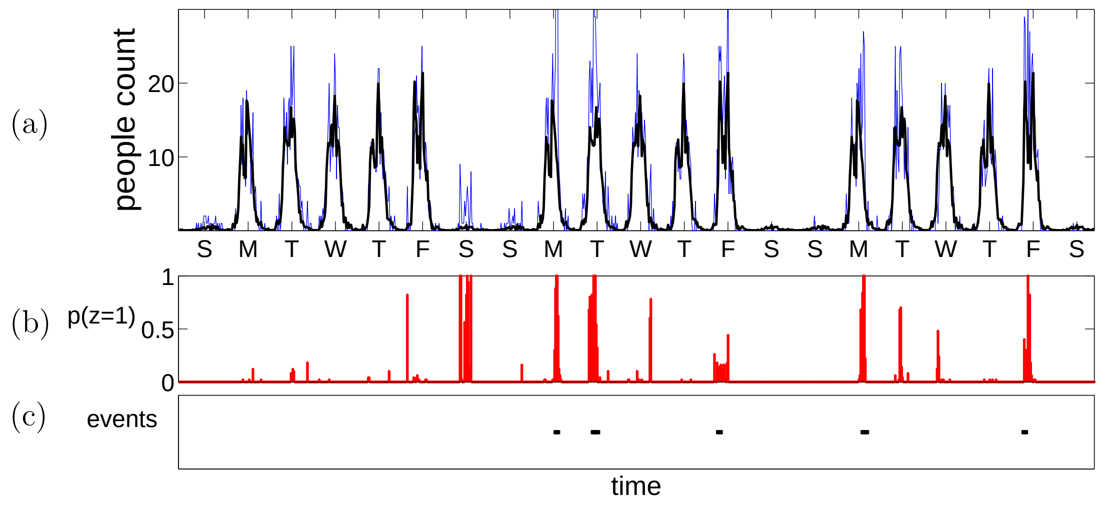


Figure 3.17: (a) Entry data, along with $\lambda(t)$, over a period of three weeks (Sept. 25–Oct. 15 2005). Also shown are (b) the posterior probability of an event being present, $p(z_t = 1|o_{1:T})$, and (c) the periods of time in which an event was scheduled for the building. All but one of the scheduled events are detected, along with a few other time periods (such as a period of greatly heightened activity on the first Saturday).

home games in the LA Dodgers 2005 regular season were used as the validation set. Three additional regular season games were not included in this set because they occurred during extended periods of missing loop sensor count information. Note that both sets of "ground truth" may represent an underestimate of the true number of events that occurred (e.g., due to unscheduled meetings and gatherings, concerts held at the baseball stadium, etc.). Nonetheless this ground truth is very useful in terms of measuring how well a model can detect a known set of events.

The results obtained by performing MCMC for the building data are shown in Figure 3.17. We plot the observations o_t together with the posterior mean of the rate parameters $\lambda(t)$ over a three week period (Sept. 25–Oct. 15); Figure 3.17 shows incoming (entry) data for the building. Displayed below the time series is the posterior probability of $z_t = 1$ (i.e. the probability that an event is taking place) at each time t, drawn as a sequence of bars, below which dashes indicate the times at which scheduled events in the building took place. In this sequence, all of the known events

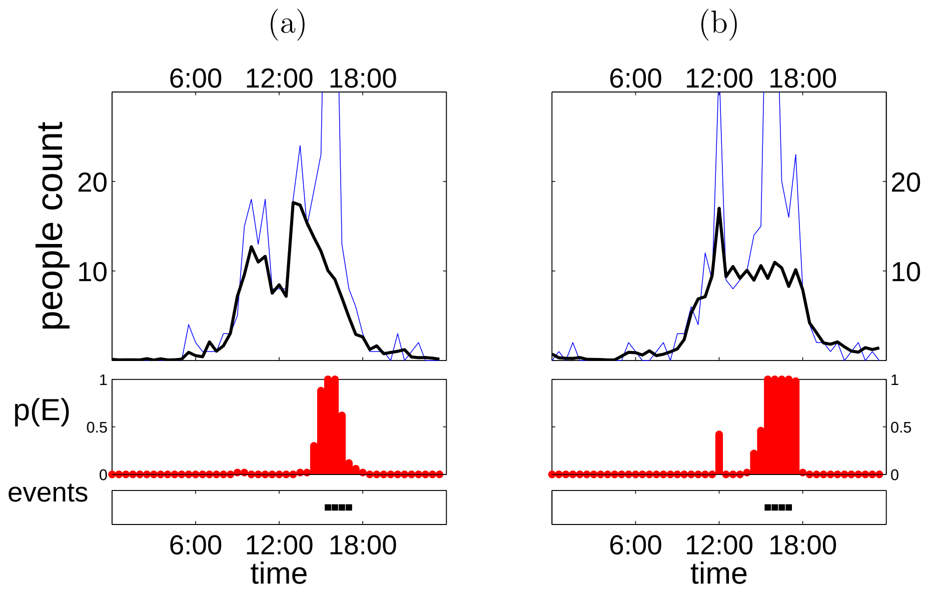


Figure 3.18: Data for October 3, 2005 for counting people (a) entering the building, and (b) exiting the building; along with rate $\lambda(t)$ and probability of event $p(z) = 1$. At 3:30 P.M. an event was held in the building atrium, causing anomalies in both the incoming and outgoing data over most of the time period.

are successfully detected, along with a few additional detections that were not listed in the building schedule. These additional detections are not necessarily false positives, but could be real events that were not on the schedule. Such unscheduled activities often occur over weekends where the baseline level of activity is particularly low. Since an event is defined as a deviation from the norm, even a group of 4 or 5 graduate students who have worked into the night on a group project and have left the building at the same time (at a time when traffic is normally near zero) would qualify as an event even though it is not on a schedule.

Figure 3.18 shows a detailed view of one particular day, during which there was an event scheduled in the building atrium. Events in the atrium often have a significant amount of "churn" (people entering and exiting the building multiple times) and thus are easier to detect. Plots of the probability of an unusual event for both the entering and exiting data show a high probability over the entire period allocated to the event,

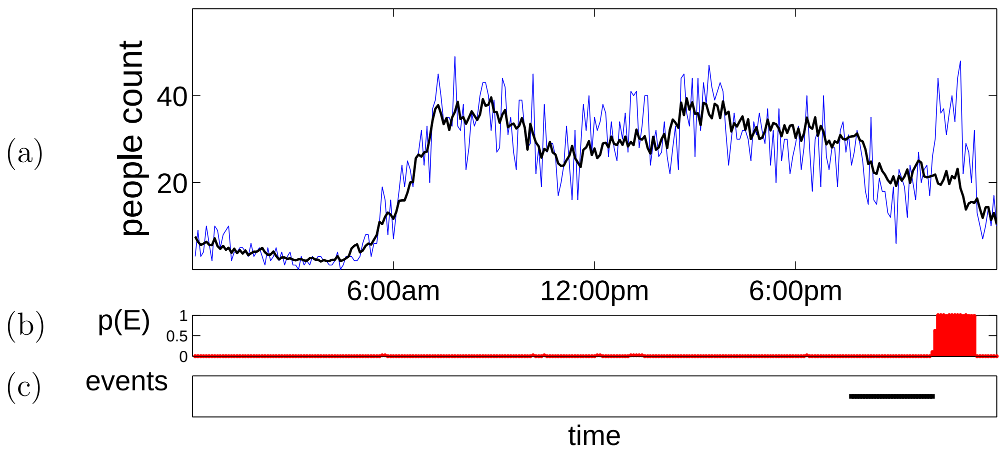


Figure 3.19: A Friday evening game, Apr. 29, 2005 (the same day used in the plots in Figures 3.2 and 3.5 where the threshold model failed due to the chicken-egg problem). Shown are (a) the prediction of normal activity, $\lambda(t)$ (dark line) along with the observed count, o_t (light blue line); (b) the estimated probability of an event, $p(z) = 1$; and (c) the actual game time. Note, the event activity at this sensor would detect vehicles leaving the event and thus the event activity would be detected post-game.

while slight increases earlier in the day were deemed much less significant due to their relatively short duration.

The results obtained by performing MCMC for the freeway traffic data for three game-days are shown in Figures 3.19 and 3.20. Figure 3.19 shows the same Friday game used in the plots in Figures 3.2 and 3.5 that was used to demonstrate the limitations of the baseline model. The baseline model was unable to detect this event (when the threshold was set to find the same number of events as our model) due to the chicken-egg problem. However, this day is an example of where our model successfully separates the normal Friday evening activity from game-day evening activity. The threshold model was able to detect the Friday games with heavy attendance, but more sparsely attended games such as this one were missed.

Figure 3.20 displays the same two-day period where the threshold model was shown to detect false alarms when the threshold level was set appropriately to detect the event on day two (Figure 3.3). Our model detects the two events with no false alarms,

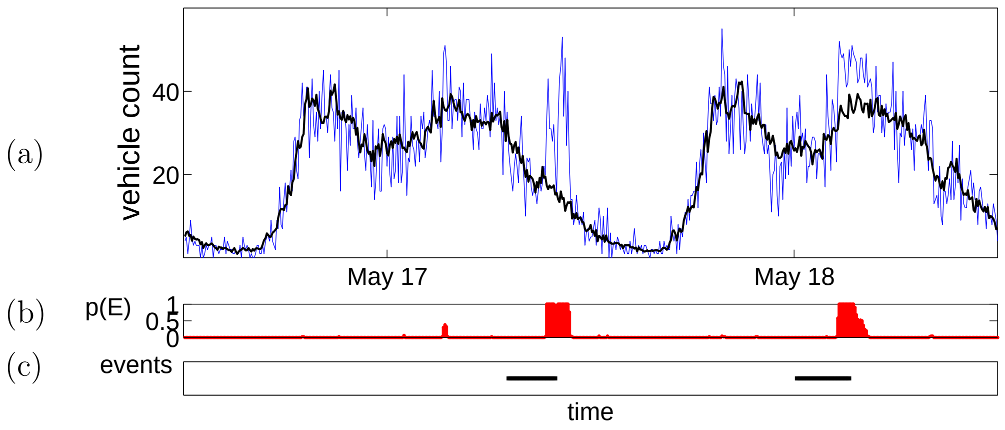


Figure 3.20: (a) Freeway data for May 17-18,2005 - light blue line (the same two-day period used in the plots in Figures 3.3 and 3.4 where the threshold model failed due to the persistence problem). Shown are (a) the prediction of normal activity, $\lambda(t)$ (dark line) along with the observed count, o_t (light blue line); (b) the estimated probability of an event, $p(z) = 1$; and (c) the actual event times.

and nicely shows the duration of the predicted events (something that cannot easily be done by the baseline method as illustrated in Figure 3.4).

Table 3.1 compares the accuracies of the Markov-modulated Poisson process (MMPP) model described in Section 3.2 and the baseline threshold model of Section 3.1 on validation data not used in training the models for both the building and freeway traffic data, respectively. For each row in the table, the MMPP model parameters were adjusted so that a specific number of events were detected, by adjusting the priors on the transition probability matrix. The threshold model was then modified to find the same number of events as the MMPP model by adjusting its threshold ϵ. For example, the number 129 in the fourth row of the Freeway Data table intuitively corresponds to the 129 top ranked events found by each model. Then the accuracy reported refers to the fraction of events in the validation set that were also found in the top 129 ranked events by each method.

In both data sets, for a fixed number of predicted events (each row), the number of true events detected by the MMPP model is significantly higher than that of the baseline

Building Data:

Total # of Predicted Events	MMPP Model	Threshold Model
149	100.0%	89.7%
98	86.2%	82.8%
68	79.3%	65.5%

Freeway Data:

Total # of Predicted Events	MMPP Model	Threshold Model
355	100.0%	85.9%
264	100.0%	82.1%
154	100.0%	66.7%
129	97.4%	55.1%

Table 3.1: Accuracies of predictions for the two data sets, in terms of the percentages of known events found by each model, for different total numbers of events predicted. There were 29 known events in the building data, and 78 in the freeway data.

model. This validates the intuitive discussion of Section 3.1 in which we outlined some of the possible limitations of the baseline approach, namely its inability to solve the "chicken and egg" problem and the fact that it does not explicitly represent event persistence. As mentioned earlier, the events detected by the MMPP model that are not in the ground truth list may plausibly correspond to real events rather than false alarms, such as unscheduled activities for the building data and accidents and non-sporting events for the freeway traffic data.

The probabilistic model outperforms the baseline for both data sets, but the difference in performance is much greater for the traffic data set. One plausible explanation for this is the difference in the size of the time-segments. The building data are segmented into 30-minute bins, while the traffic data are segmented into 5-minute bins. It is not unreasonable that the event traffic could fall into only one or two bins, thus not showing a persistent signal for the probabilistic model to take advantage of. Whereas the 5-minute bin size of the traffic data would see event traffic over a much larger number of segments, allowing the model to take advantage of the persistence observed.

Given that our model is capable of detecting and separating the influence of unusual periods of activity, we may also wish to use the model to estimate other quantities of interest. For example, we might wish to use our estimates of the *amounts* of abnormal

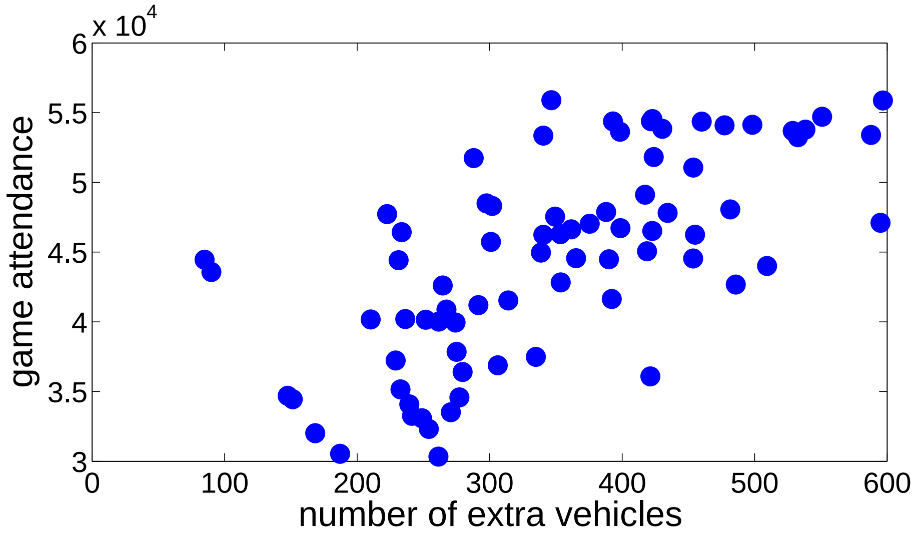


Figure 3.21: The attendance of each baseball game (y-axis) shows correlation with the number of additional (event–related) vehicles detected by the model (x-axis).

behavior to infer other, indirectly related aspects of the event, such as its popularity or importance.

Along with estimating the probability that an unusual event is taking place, as part of the inference procedure our model also estimates the number of counts which appear to be associated with that event (i.e. how many additional people seem to be entering or leaving the building or the number of extra vehicles entering the freeway during a particular time period). This type of information is potentially very useful to traffic engineers, and current traffic monitoring systems do not provide such estimates. Chapters 4 and 5 give examples of how the event count can be used in traffic applications.

One intriguing use for this information is to provide a score, or a measure of popularity, of each event. As an example, taking our collection of LA Dodgers baseball games, we compute and sum the posterior mean of extra (event-related) vehicles observed,

e_t, during the duration of the event detection.

$$\text{event popularity} = \sum_{t=t_{beg}}^{t_{end}} \hat{e}_t \tag{3.22}$$

where t_{beg} is time segment at the beginning of the event (the first time-segment where $p(z_t|o_{1:T}) > 0.5$) and t_{end} is the final of consecutive time-segments where $p(z_t|o_{1:T}) > 0.5$, and where $\hat{e}_t$ is the sample average $1/S \sum_{s=1}^{S} e_{ts}$.

Figure 3.21 shows that our estimate of the number of additional cars is positively correlated with the actual overall attendance recorded for the games (correlation coefficient 0.67). Similar attendance scores can be computed for the building data, or other quantities such as the duration estimated, though for these examples no ground truth exists for comparison.

3.5 Assessing the Poisson Assumption

Let o_t, for $t \in \{1, \dots, T\}$, generically refer to the observed count at time t for any of the time-dependent counting processes, such as the freeway traffic 5-minute aggregate count process or either of the two (entering or exiting) building 30-minute aggregate door count processes.

Perhaps the most common probabilistic model for count data is the Poisson distribution, whose probability mass function is given by

$$\mathrm{P}(o_t; \lambda) = \frac{e^{-\lambda} \lambda^{o_t}}{o_t!} \qquad o_t = 0, 1, \dots \tag{3.23}$$

where the parameter λ represents the rate, or average number of occurrences in a fixed time interval. When λ is a function of time, i.e. $\lambda(t)$, (3.23) becomes a nonhomogeneous Poisson distribution (see Chapter 2.1), in which the degree of heterogeneity depends on the function $\lambda(t)$.

The assumption of a Poisson distribution may not always hold in practice, and it is reasonable to ask whether it fits the data at hand. For a simple Poisson distribution, there are a number of classical goodness-of-fit tests which can be applied, such as the chi-squared test [56]. However, in this case we typically have a large number of potentially different rates, each of which has only a few observations associated with it. For example in the freeway traffic data, there are 2016 five-minute time-segments in a week, each of which may be described by a different rate, and for each time-segment there are between 22 and 25 non-missing observations over the 25 weeks of our study.

Although classical extensions for hypothesis testing exist in such situations [74], here we use a more qualitative approach since our goal is simply to assess whether the Poisson assumption is reasonable for the data. Specifically, we know that, if the data are Poisson, the true mean and variance of the observations at each time-segment should be equal. Given finite data, the empirical mean and variance at each time provide noisy estimates of the true mean and variance, and we can visually assess whether these estimates are close to equal by comparing them to the distribution expected for Poisson-distributed data.

Figure 3.22 gives an illustration of what this type of mean/variance visualization would look like if the data being tested were in fact generated by a Poisson distribution. In a Poisson distribution. $mean = variance = \lambda_t$. In our non-homogenous Poisson process for the building data, for example, $\lambda(t)$ ranges from near 0 (in the middle of the night when there is little to no traffic) to around 20 (around mid-day when a lot of people are leaving the building for lunch). In our simulation for Figure 3.22, we also allowed $\lambda(t)$ to range from 0 to 20. If an infinite number of samples

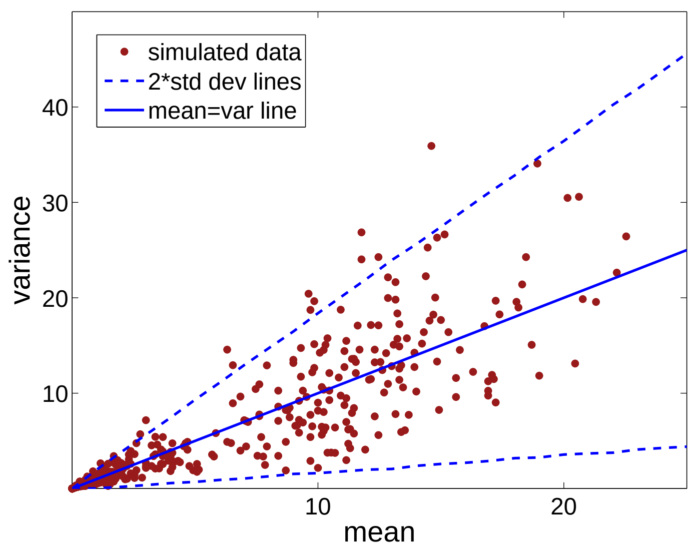
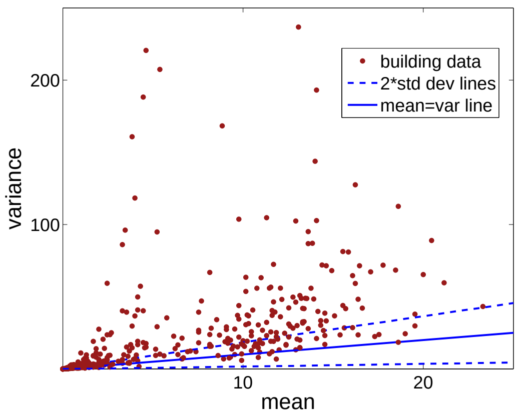


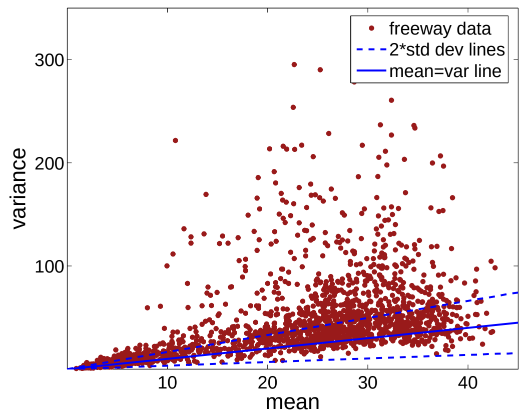
Figure 3.22: Simulated (mean,variance) pairs generated by Poisson distributions with λ's ranging from 0 to 20 (similar to the rate parameters found in the building data). The solid blue line is the $mean = variance$ line and the dashed lines show $\pm$2-sigma.

were taken for each λ, the (mean, variance) data pair plotted points for the sampled values would fall directly on the solid blue $mean = variance$ line. However, in our data sets, we have a limited number of observations for each λ, one data point for each week of history in the training set. In our simulation, we sampled 13 points for each λ value. With fewer data points, we expect the (mean, variance) points to lie near, but not usually on the $mean = variance$ line. The dashed lines in Figure 3.22 show $\pm$2-sigma lines. If the data are truly generated by a Poisson process, then we would expect approximately 95% of the data points to lie in between these two lines, as is the case in our simulation and illustrated in Figure 3.22.

This (mean, variance) comparison, performed using all observations of the building data set and the traffic data set, is shown in Figure 3.23(a) and (b) respectively. The empirical estimates of observed (mean,variance) pairs at each time-segment for the traffic data are shown as a scatter-plot, while the two-sigma confidence region for the null hypothesis (that the data are Poisson) is demarcated by solid lines. Visually, the two distributions seem quite different, indicating that the raw data are not Poisson.

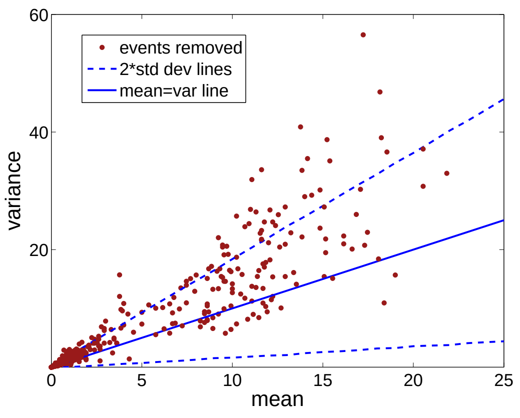


(a)

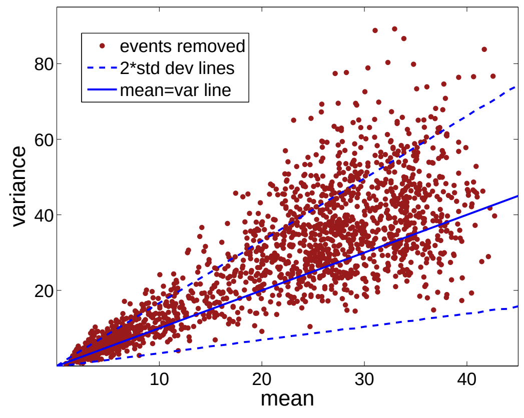


(b)

Figure 3.23: Scatter-plot of empirical (mean,variance) pairs observed in the building data set (a) and the traffic data set (b). The (mean,variance) pairs are compared with the theoretical distribution expected for Poisson-distributed random variables (dashed lines show ±2-sigma interval). The presence of events makes these distributions quite dissimilar. The observed data are very over-dispersed compared to what is expected for a Poisson process.



(a)



(b)

Figure 3.24: The same scatter-plot of empirical (mean,variance) pairs observed in the building data set (a) and the traffic data set (b) as in Figure 3.23 along with corresponding confidence intervals, except event observations are removed. After removing about 5% of data thought to be influenced by events, the data show much closer correspondence to the Poisson assumption.

However, this should not be surprising, since the raw data are corrupted by the presence of occasional bursts of events.

A better evaluation of the model's appropriateness is given by first running the Markov-modulated Poisson model described in the sequel, then using the results to remove the fraction of the data thought to be anomalous. After removing only about 5% of the data, the scatter-plot looks considerably more consistent with the Poisson hypothesis, as shown in Figure 3.24(b) for the traffic data set. Visually, one can see that the data may be slightly over-dispersed (higher variance than mean), a common issue in count data, but that the approximation looks fairly reasonable. Figure 3.24(a) shows the same plot for the 15 weeks of building data (including points for both entry and exit data). Here too, the distribution of (mean, variance) pairs appears to be reasonable in the context of our primary goal of event detection, although again there is evidence that the data are slightly over-dispersed compared to that of a Poisson distribution.

3.6 Summary of Contributions

- Developed a probabilistic framework for separating event and normal behavior in a time-series of count measurements that is an extension of Steve Scott's work [65]. This MMPP provides an unsupervised method for simultaneously learning the rate parameters for a non-homogenous Poisson process (the normal process) and inferring the periods of time when an unusual event is taking place (the event process). The model can also be used to make other inferences such as the duration and popularity of unusual events.

- Defined an iterative MCMC procedure for learning and inference for the MMPP, and presented an intuition for setting some of the parameters (event transition priors and event count negative binomial distribution parameters).

- Presented a baseline threshold method for separating the event and normal processes. Demonstrated that while a simple model may be useful for some data sets, real data challenges such as the “chicken-egg” and “persistence” problems limit the baseline model. Also showed examples of sensors with periods of failure that would cause simple models to break down.

- Demonstrated the effectiveness of the model with a case study on a building data set and a traffic data set. The model’s ability to address the “chicken-egg” and “persistence” problems was demonstrated; and the model’s prediction of event activity was found to be superior to the baseline model. Gave an example of using the model output to estimate other quantities of interest with a baseball attendance prediction experiment.

- Tested the Poisson assumption using time series of count data from two different human counting sensor types. Found that while the measured count from the normal process is a bit over-dispersed, the Poisson assumption is reasonable for this type of count measurements of human activity.

Chapter 4

Negative Event and Fault Tolerant Event Model Extensions

While the two state model outlined in the previous chapter is sufficient for some applications where event activity only causes an increase in the observed signal and where the data are relatively clean, real world sensor data are often significantly noisier and more difficult to work with. Different types of unusual event behavior and periods of corrupt measurements due to sensor failure provide challenges to the event model. This chapter describes two extensions to the event model that make it more robust to the variation found in real world sensor measurements: an extension to model negative event behavior, and an extension to model sensor failure. The chapter closes with two case studies: a study showing use of the model to detect and remove sensor failure, and a case study illustrating the utility of the model when applied to over 1700 sensors in an urban environment.

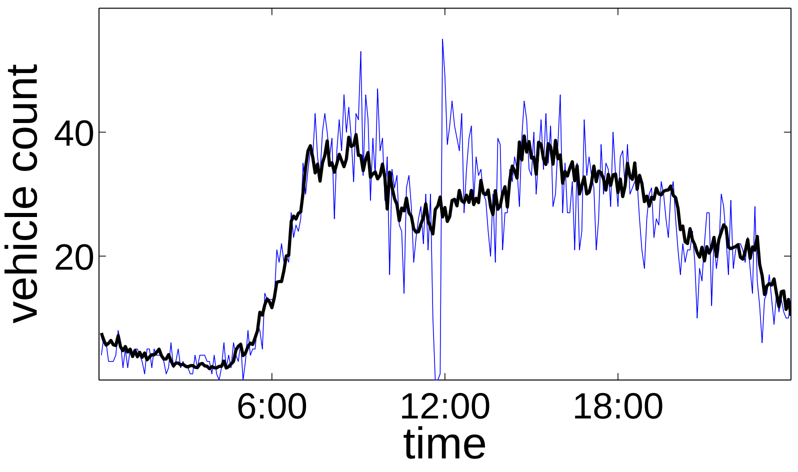


Figure 4.1: Negative event illustration. Observed freeway vehicle count data (light blue line) and estimated normal profile (thick black line) for May 6. The number of observed cars drops sharply around 12:30 showing evidence of a negative event. The decrease is followed by a sustained abnormally large increase in traffic, evidencing a positive event.

4.1 Negative Event Model

The event process in the model described in Chapter 3 is a binary process with a non-event state describing normal activity and a positive event state describing short periods of where higher than normal activity is observed. This two state model is appropriate for applications such as the telephony application studied by Scott [66] where the unusual activity is caused by a process that can only add to the normal activity. However, the data captured by the sensors studied in this thesis commonly reflect periods of lower than normal activity. In this section, we describe an extension of the event model to add an additional state for negative events. This negative event model provides a richer representation that is a more accurate and robust model of the data, and it provides additional valuable information about the human behavior captured by the sensors.

4.1.1 Negative Event Examples and Motivation

Figure 4.1 shows an example of a negative event. The data come from a loop sensor embedded in an on-ramp to a freeway. At approximately 12:30PM, the vehicle count measurement (light blue line) falls to zero for about 15 minutes. Vehicle flow normally measures around 30 vehicles every 5 minutes (the normal flow profile is represented by the thick black line). After this period where no vehicle flow is observed, the measurement immediately jumps to an abnormally high level for the next 45 minutes. One plausible explanation for this period of observations is that the initial negative event could be due to an accident or construction, which shut down the ramp for a short period, followed by a positive event during which the resulting build-up of cars were finally allowed onto the highway.

Negative events (corresponding to lower than expected activity) tend to be more rare than positive events (higher than expected activity), but can play an important role in the model. The two state model would be forced to explain the abnormally low measurements for the 15 minute period shown in Figure 4.1 using the no event state (i.e. $z_t = 0$), which further adds noise to the inference of the rate parameters of the normal process ($\lambda(t)$). If negative events are relatively rare and small, the event model can absorb them into the normal process without much ill effect. However, larger and more frequent negative event activity can cause the two state event model to fail.

Figure 4.2 shows a typical example of holiday behavior. This plot shows the observed vehicle flow (light blue line) for a traffic sensor during Memorial Day. The observed traffic is significantly lower than the normal Monday traffic (thick black line) for most of the day. Since Monday is a popular day for a holiday, the impact of this abnormally low daytime holiday behavior is amplified; and in the experiments in Case Study 1

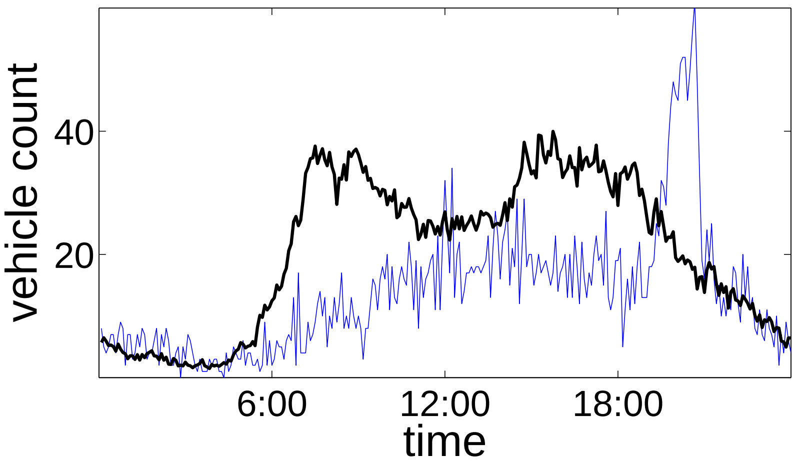


Figure 4.2: Holiday traffic illustration. Observed freeway vehicle count data (light blue line) and estimated normal profile (thick black line) for Memorial Day, 2007. The number of observed cars is significantly lower than the normal Monday traffic for most of the day, followed by an abnormally high surge of traffic in the evening.

in Chapter 3.4, the holiday measurements needed to be removed before running the model. If negative event traffic is known ahead of time, as in the case of holidays, it can be removed; but in many cases, negative event activity is not known a priori (for example flow data from traffic sensors embedded in freeway through lanes that are regularly impacted by accidents and construction). By explicitly treating such abnormally low periods as negative events, the model can be made robust to such periods.

In addition to making the model more robust, characterizing negative events can provide valuable information about human behavior. Analysis of negative event activity has potential value in areas such as transportation and security.

For example, using our model with data from security cameras could provide useful alerts and at the same time preserve privacy. Video camera data can be processed to count people [47] which could then be used as input to our event model. The processed video data are relatively anonymous and can still be useful in security applications. With slight modifications, our model could provide real time alerts when unusual activity occurs[1]. If this is applied to security cameras at a bank, for example, the bank could create a policy where police or an outside security firm would be normally restricted from viewing the camera feeds (thus preserving privacy of bank workers and customers); but could be permitted (and prompted) to view the video feeds when unusual activity is taking place (for example, no one leaves the bank for a period of time when there is normally higher traffic).

A similar use could be in the case where a large building or mall is outfitted with hundreds of security cameras. One can imagine that it would be difficult to security personnel to effectively monitor activity in all of the video feeds. The output of the event model could be used to alert, and prioritize video feeds that have potential security concerns.

Another application where negative events are important is in transportation research. Congestion causes decreased traffic flow on freeways and thus lost productivity, and negative event detection could be useful for the problem of characterizing the effect of an accident [62], for example. This particular problem is examined in more depth in Chapter 5.

4.1.2 Adding a Negative Event State

In order to model negative events, several modifications must be made to the model.

[1]We created a demonstration of a real time event detection system using a traffic loop detector with the help of an undergraduate student, Sean Li, as part of the Surf-IT undergraduate research program at UC, Irvine [43].

Instead of a binary process, the event process z_t becomes a ternary process:

$$z_t = \begin{cases} 0 & \text{if there is no event at time } t \\ 1 & \text{if there is a positive event} \\ -1 & \text{if there is a negative event} \end{cases} \tag{4.1}$$

which is reflected in the transition matrix

$$M_z = \begin{pmatrix} z_{00} & z_{0+} & z_{0-} \\ z_{+0} & z_{++} & z_{+-} \\ z_{-0} & z_{-+} & z_{--} \end{pmatrix} \tag{4.2}$$

with each row summing to one, e.g., $z_{00} + z_{0+} + z_{0-} = 1$.

A beta prior distribution was used for the transition parameters for the two state process, since the beta distribution is the conjugate prior of the binomial distribution (see Chapter 2.2 for more information about the benefits of using conjugate priors). The conjugate prior for a multinomial distribution is the Dirichlet distribution, so the priors on the transition probability variables z_{ij} are specified as

$$[z_{00},\, z_{0+},\, z_{0-}] \sim \mathrm{Dir}(z\,;\, [a^Z_{00},\, a^Z_{0+},\, a^Z_{0-}]) \tag{4.3}$$

and similarly for the other matrix rows, where $\mathrm{Dir}(\cdot)$ is the Dirichlet distribution.

While the procedure for drawing samples of n_t and e_t given the sample $\hat{z}_t$ is relatively straightforward for positive events, the negative event case requires a few extra steps. In each case, n_t is from the discrete distribution

$$\hat{n}_t \sim \mathrm{P}(n_t; \lambda(t))\, \mathrm{NBin}(o_t - n_t; a^E, b^E/(1 + b^E)) \tag{4.4}$$

and then e_t is simply found using the deterministic relationship $\hat{e}_t = o_t - \hat{n}_t$.

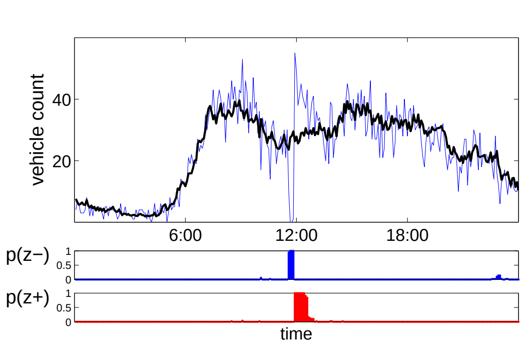
In the case of a positive event, n_t has fixed boundaries and can only take values in $\{0, 1, \ldots o_t\}$; therefore Equation 4.4 is evaluated for $o_t + 1$ combinations of n_t and e_t. However, in the negative event case, $n_t \geq o_t$, and thus n_t has no fixed upper limit. Equation 4.4 could be evaluated for an infinite number of values of n_t. In practice, for computational efficiency we truncate the distribution (imposing an upper limit) at the point given by $\mathrm{P}(o_t + i; \lambda(t)) < 10^{-4}$.

When o_t is unobserved (missing), n_t and e_t are no longer coupled through the deterministic relationship $o_t = n_t + e_t$. In the case where a positive event is drawn $\hat{z}_t = +1$, n_t and e_t can be sampled independently. In the negative event case $\hat{z}_t = -1$, however, we must ensure that $n_t - e_t > 0$. This can be accomplished fairly easily through rejection sampling, where n_t and e_t are repeatedly drawn independently until they satisfy the positivity condition.

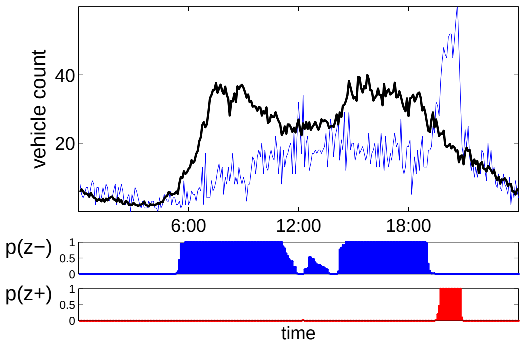
The remainder of inference and learning proceeds as in Chapter 3.3.

4.1.3 Results

The experiments in Case Study 1 (Chapter 3.4) were repeated using the negative event model. Figure 4.3 shows the output of the negative event model for the same days shown in Figures 4.1 and 4.2. The first row shows the observed flow (light blue line) and normal profile (thick black line) for the accident event (Figure 4.3 (a)) and the holiday event (Figure 4.3 (b)). The second row shows the probability of a negative event, and the third row shows the probability of a positive event as inferred by the model.



(a)



(b)

Figure 4.3: Histograms of the number of extra observations (counts in the flow case and seconds in the occupancy case) due to (a) high flow event activity and (b) high occupancy event activity.

Since the negative event model is robust to periods of unusually low activity, the holiday measurements did not have to be removed before running the model. A preprocessing step was eliminated and analysis of negative event behavior is made possible.

Parameter settings for the experiment conducted for this section appear in Appendix A.2. The models and case studies in the remainder of this thesis all include the negative event extension described in this section.

4.2 Sensor Failure Extension

This section examines some of the characteristics of sensor failure and the modeling challenges that failures pose, and it presents an extension of the event model to detect and remove failure.

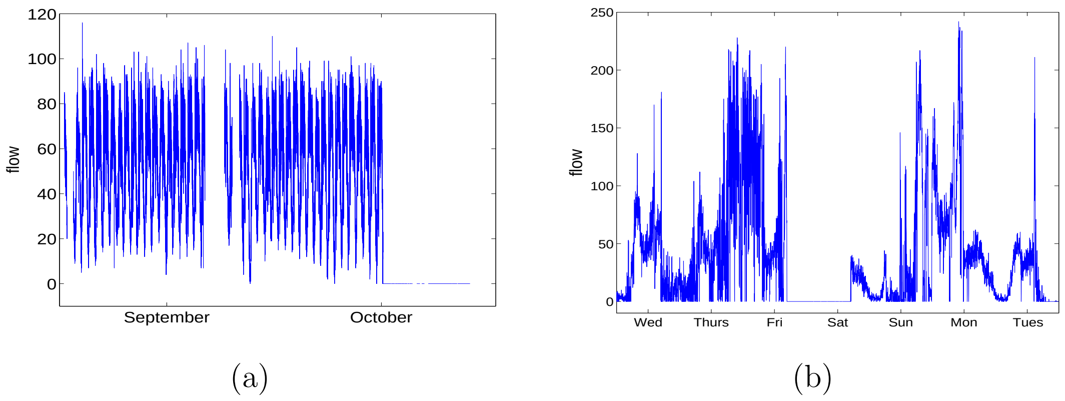


Figure 4.4: Two examples of sensor failure. (a) shows a “stuck at zero” failure at the end of October and several days of where the sensor did not report measurements at the end of September. (b) shows a sensor that sporadically fails and reports high flow measurements during failure.

4.2.1 Examples of Sensor Failure

Large-scale sensor instrumentation is now common in a variety of applications including environmental monitoring, transportation, industrial automation, surveillance and security. These sensors report a large variety of observed behavior and also experience a large variation in types of sensor failure. It is common to create rules based on heuristics to filter out the measurements that occur during periods of sensor failure, but it is difficult to filter out corrupted measurements from a wide variety of failure modes without losing good data, particularly periods of unusual but valid measurements.

One example of large scale sensor instrumentation is the freeway loop detectors introduced in Chapter 1.1 that are used in transportation research. The California Department of Transportation (Caltrans) maintains an extensive network of over 20,000 inductive loop sensors on California freeways [9, 20]. Every 30 seconds each of these traffic sensors reports a count of the number of vehicles that passed over the sensor and the percentage of time the sensor was covered by a vehicle, measurements known as the flow and occupancy respectively. The data are continuously archived, providing a potentially rich source from which to extract information about urban transportation patterns, traffic flow, accidents, and human behavior in general. In the remainder of this chapter, we will use a seven month history of measurements from a set of over 1700 loop detectors embedded in on- and off-ramps to the freeways in Orange County and Los Angeles, California.

The seven months of time-series data from the 1700 loop sensors contain a wide variety of anomalous behavior including "stuck at zero" failures, missing data, suspiciously high readings, and more. Some sensor failures, such as the "stuck at zero" failure seen in Figure 4.4(a), are fairly easy to detect and remove in a preprocessing step. But other failure types, such as illustrated in Figures 4.4(b) and 4.5, are not easy to eliminate automatically. Figure 4.4(b) shows one week of sensor measurements where the sensor appears to be reporting correct measurements at times (most of Tuesday) and is likely reporting corrupted measurements (particularly Thursday night through Friday morning) at other times. Unlike the "stuck at zero" failure, however, a simple rule for detecting the type of failure seen in Figure 4.4(b) is not easy to construct.

The failure in Figure 4.5 is not as obvious. The top panel shows a zoomed-out view of the flow measurements over a seven month period with large sections of missing data. There are two periods of time where the sensor recorded measurements, separated by a gap of missing data. The middle panel zooms in on one week of flow measurements in the first period and shows strong weekday and weekend patterns. The bottom shows a zoomed-in view of a week that is characteristic of the behavior seen in the second period of measurements. There are a lot of missing measurements, but many of the remaining measurements look reasonable and might plausibly pass for daily traffic variations if we did not know the normal conditions. Other measurements, such as on Saturday, Sunday and Wednesday nights, are unusually high; but it would be hard to say whether these measurements are due to normal activity, event activity, or

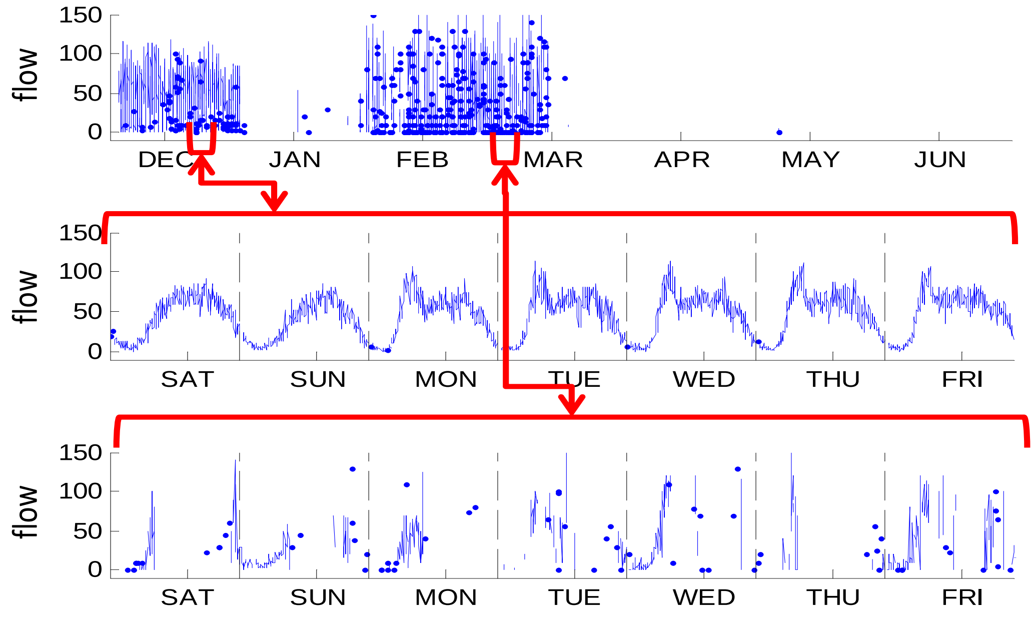


Figure 4.5: Example of a less obvious sensor failure. The blue line indicates observed measurements for one sensor for (top) the entire study period and (middle, bottom) two specific weeks. The top panel is a shows two periods of measurements separated by a gap of missing measurements. The middle panel zooms in on one week of accurate measurements from the first period, while the bottom panel shows one week of corrupted measurements due to sensor failure from the second period.

whether the sensor is failing. Upon closer examination, however, the measurements in the second period of measurements do not have a predictable pattern that we expect from human-generated traffic. These measurements are also consistently significantly different than the measurements in the first set. It is likely that the sensor reported accurate measurements during the first period and was in failure during the second period. This type of failure is particularly difficult to automatically filter.

4.2.2 Modeling Challenges

Large scale loop sensor data of this form are well known to transportation researchers, but have resisted systematic analysis due to the significant challenges (particularly due to sensor failure) of dealing with noisy real-world sensor data at this scale. Bickel et al. [9] outline some of the difficulties in a recent survey paper:

> Single-loop detectors are the most abundant source of traffic data in California, but loop data are often missing or invalid. Missing values occur when there is communication error or hardware breakdown. A loop detector can fail in various ways even when it reports values. Payne et al. (1976) identified various types of detector errors including stuck sensors, hanging on or hanging off, chattering, cross-talk, pulse breakup and intermittent malfunction. Even under normal conditions, the measurements from loop detectors are noisy; they can be confused by multi-axle trucks, for example. Bad and missing samples present problems for any algorithm that uses the data for analysis, many of which require a complete grid of good data.
>
> ...
>
> A systematic and principled algorithm [for detecting faulty sensors] is hard to develop mainly due to the size and complexity of the problem. An ideal model needs to work well with thousands of detectors, all with potentially unknown types of malfunction.
>
> ...
>
> Even constructing a training set is not trivial since there is so much data to examine and it is not always possible to be absolutely sure if the data are correct even after careful visual inspection.

We observed some of these challenges when applying the event model to the diverse variety in the data from the loop detectors in our study and discovered a limitation of the three state event model, particularly when applied to sensors that experienced lengthy periods of sensor failure. We applied the original model to the data from our seven month study involving 1716 sensors and over 100 million hidden variables.

Table 4.1: Event Model Results. The 1716 sensors in this study are categorized using a measure of the ability of the model to find a predictable periodic component in the sensor measurements (if present). The *event fraction* is defined as the fraction of time a sensor's measurements are classified as a positive or negative event. For sensors with lower event fractions, the model has found a strong periodic component with fewer periods of unusual event activity.

Event Fraction	Number of Sensors
0 to 10%	912
10 to 20%	386
20 to 50%	265
50 to 100%	153

Table 4.1 shows one method for judging how well the model fit the data. The table shows the fraction of time that the model inferred unusual event activity for each of the sensors during our seven month study (i.e. the fraction of time slices where $z_t \neq 0$). The lower this event fraction value is, the better the model fit the data; the higher the value is, the greater fraction of measurements are not explained by the periodic component of the data.

There is reason to be suspicious when the model infers unusual event activity for a large fraction of the time, especially in cases where unusual event activity is more common than normal activity (as in the last row of the table). A review of the sensors where event activity was inferred over 50% of the time revealed some weaknesses of the model. Some sensors in this category were apparently faulty throughout the study; and thus the model would not be expected to find a normal pattern. Another group of sensors recorded non-missing measurements for only a very small fraction of the study, which were not enough to form a good model. However, there were many sensors which appeared to have an underlying periodic behavior pattern that was missed by the original model.

The sensor with the "stuck at zero" failure (Figure 4.4(a)) is an example of a sensor with a clear periodic pattern that the original model missed. Figure 4.6 shows the

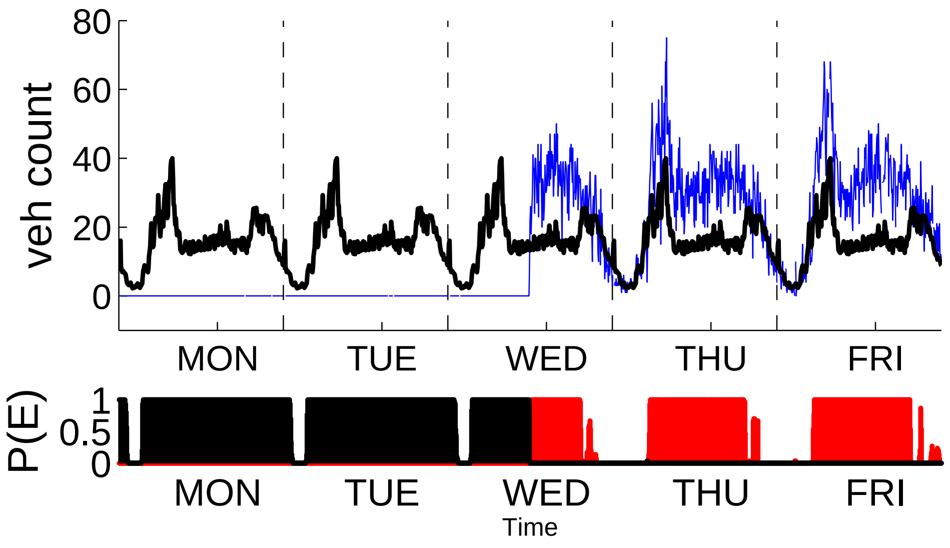


Figure 4.6: Event model output for the sensor in Figure 4.4(a). Shown are the observed measurements (blue) for the week that the "stuck at zero" failure ended and normal measurements returned; superimposed with the model's inferred Poisson rate (black). With a long period stuck at zero, a poor model is inferred for normal behavior in the middle of the day. This is reflected in the event probabilities (bottom), where unusual event activity is predicted for most of each day.

week that the "stuck at zero" failure ended and normal measurements returned. The measurements after the failure have a strong periodic pattern of the kind that the event model is designed to find. The dark thick line in the top panel shows that the model's attempt to fit the data. The model is able to learn early morning and late night behavior, but an inaccurate profile is inferred for normal behavior in the middle of the day. Examples such as this were observed across many other sensors, and in many cases where a poor model was inferred for normal behavior there also appeared to be long periods where the sensor was faulty.

We experimented with a number of modifications to the model to help the model find the normal pattern and to classify the periods of sensor failure as event activity including:

- fixing the state transition probabilities so that the model could not adjust them to define events as frequent occurrences. In some cases, the strong priors put on the state transition variables to encourage events to be relatively rare were overcome in the model's attempt to fit the data.

- avoiding poor initialization of the normal process parameters $\lambda(t)$ Gibbs sampler. When extensive periods of failure were present, the original guess used for the normal process $\lambda(t)$ was sometimes far from the actual periodic pattern. We tried an empirical Bayes approach [12] by making a preprocessing pass of the data to filter out apparent failures and outliers, then using the mean (separated by day and time) of the remaining measurements as the initial guess for $\lambda(t)$. The model was then run on the original data (not the filtered data).

- changing the inference procedure for missing data. In the original event model, when missing data was encountered, values for n_t and e_t were still sampled given the z_t sample. These samples were then included in the learning procedure for the model parameters. When there were large periods of missing data, however, these samples sometimes dominated the set of samples used to learn the model parameters. This complicated the learning process, especially in cases where the initial guesses for the model parameters were far from the true values. The inference and learning procedures were modified to ignore missing data samples.

These adjustments improved the performance of the model in some cases. But in many cases (particularly in sensors with extensive periods of sensor failure) inaccurate profiles were still inferred for normal behavior.

4.2.3 Modeling Failures Explicitly

It is clear from the experiments in the previous section that in order to make the original model more general and robust, sensor failures should be addressed directly instead of bundling them together with unusual event activity. We note that heuristic approaches to sensor fault detection in traffic data have been developed in prior work [14, 36], but these techniques are specific to loop detectors and to certain types of sensor failures. Our focus in this chapter is on developing an approach that can handle more general types of faults, not only in loop sensor data but also in other sensors that measure count data.

One possible approach to solve these problems would be to modify the model to broaden the definition of "events" to include sensor failures. However, sensor failures and events (as we define them) tend to have quite different characteristic signatures. Events tend to persist for a few hours while failures often have a much broader range of temporal duration. Events also tend to be associated with a relatively steady change (positive or negative) in count rates over the duration of the event, while failures can have significant variability in count rates during the duration of the fault. Ultimately, while there is not a crisp boundary between these two types of non-normal measurements, we will show in later sections that both types are sufficiently different and prominent in the data to merit separate treatment.

When we detect sensor failures visually, we are in essence recognizing extensive periods of time where the periodic structure that we have come to expect is not present. This reasoning is built into a new model, defined by the graphical model in Figure 4.7. A second Markov chain has been added to model periods of sensor failure.

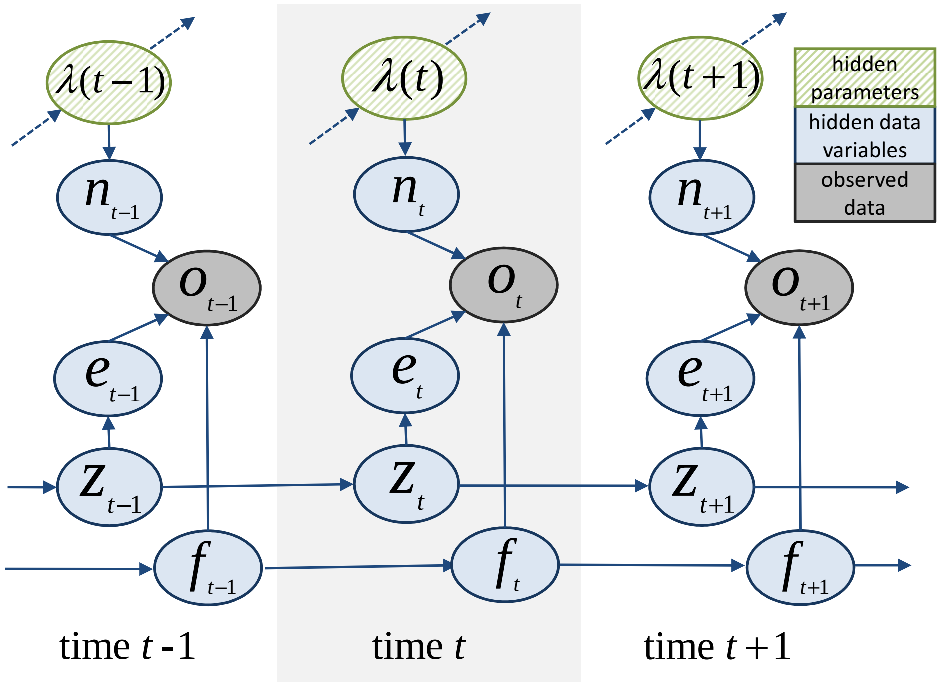


Figure 4.7: Graphical model of the fault tolerant event model. A second Markov chain has been added to model sensor failures.

This failure state is a binary variable indicating the presence or absence of a sensor failure.

$$f_t = \begin{cases} 1 & \text{if there is a sensor failure at time } t \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

with transition matrix

$$M_f = \begin{pmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{pmatrix} \tag{4.6}$$

with each row summing to one, e.g., $f_{00} + f_{01} = 1$. The parameters of the transition matrix are set to encourage extremely rare (parameter settings for the fault tolerant event model used in the case studies at the end of the chapter can be seen in Appendix A.3).

The implementation of the inference procedure when the fail state is sampled, $\hat{f}_t = 1$, is much different than when the fail state is not sampled, $\hat{f}_t = 0$. When there is no sensor failure, if the state variable in the event chain is in an event state, the observed measurement is accounted for by both the normal and event count components. If the state variable in the fault chain is in the fault state, however, the observed measurement is treated as if it were missing. This allows our model to ignore the faulty part of the data when inferring the time-varying Poisson (normal) component of the data.

At a high level, two main factors affect inference of a sensor failure: the distance of the measured count from the rate parameter, and the probability of a sensor failure in neighboring time slices. In general, smaller deviations of the measured count from the rate parameter of the normal process are more likely to be caused by events than failures, while larger deviations have higher likelihood for a failure. Also, an unusual activity of short duration favors being explained by an event, while longer periods of unusual activity favor a fault.

In a Bayesian framework, our belief about the relative duration of events and failures can be encoded into the priors that are put on the transition parameters for the two Markov chains. We expect events to be short, ranging from a few minutes to a couple of hours; in contrast, we expect failures to be relatively lengthy, ranging from days to months.

The modular nature of graphical models makes it possible to extend the original model without starting over from scratch. Some modifications were made to the inference calculations when the fault sensor process was added to the original model, but learning and inference proceeds in the same general fashion as before. From a practical viewpoint, relatively few changes to the software code were needed to

Table 4.2: Comparison of the event model and the fault tolerant event model using the fraction of time in the event state for the 1716 sensors in the study. Missing data and measurements detected as faulty were removed from both models before calculating the event fraction.

	Event Model	Fault Tolerant Model
Event Fraction	Number of Sensors	Number of Sensors
0 to 10%	960	1285
10 to 20%	375	242
20 to 50%	244	117
50 to 100%	137	72

extend the model to include a failure state. Details of these changes are given in Appendix A.3.

4.3 Case Study 3: Scale-up Study

In this section we report on a large case-study involving statistical data mining of over 100 million measurements from 1700 freeway traffic sensors over a period of seven months in Southern California. Additional information about the sensors is also utilized during analysis, such as their geographic location and the sensor type (i.e. whether the sensor was on an exit or entrance ramp).

We first test the effectiveness of the fault tolerant event models described in the previous section, using the same experiment that exposed limitations of the previous event model illustrated in Table 4.1. We then show a large scale analysis of an urban network of loop detectors using the results of the model.

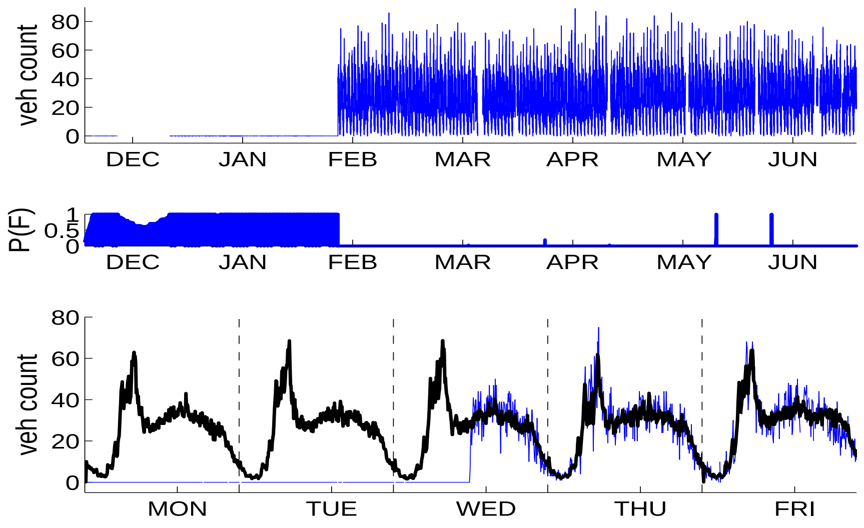


Figure 4.8: Fault tolerant model results for the “stuck at zero” example shown before in Figures 4.4(a) and 4.6. Shown are the raw data (top), the model’s estimate of the probability of a faulty sensor (center), and the inferred time-varying Poisson rate of the normal component (bottom dark line) along with one week of observations (thin blue line). This is the same as in Figure 4.6. The model detects the “stuck at zero” failure, and the model’s rate fits the periodic signal in the data that was missed by the original model (Figure 4.6).

4.3.1 Failure Detection and Removal

Table 4.2 gives a sense of the gains made by the fault tolerant event model compared to the original model. We compare the percentage of time spent in an event state for sensors under each model. In order to provide a fair comparison, missing measurements as well as measurements detected as faulty by the fault tolerant event model were removed for *both* models before calculating the event fraction of the remaining measurements predicted by each model. The fault tolerant event model is able to infer a better model of the periodic structure hidden within the count data. This is apparent in the shift of sensors to the upper rows of the table where unusual activity is detected less frequently and thus more of the data are explained by the time-varying Poisson process. Of the 381 sensors in the $> 20\%$ range for event fraction with the original model, only 189 remain under the fault tolerant event model, a 50% reduction.

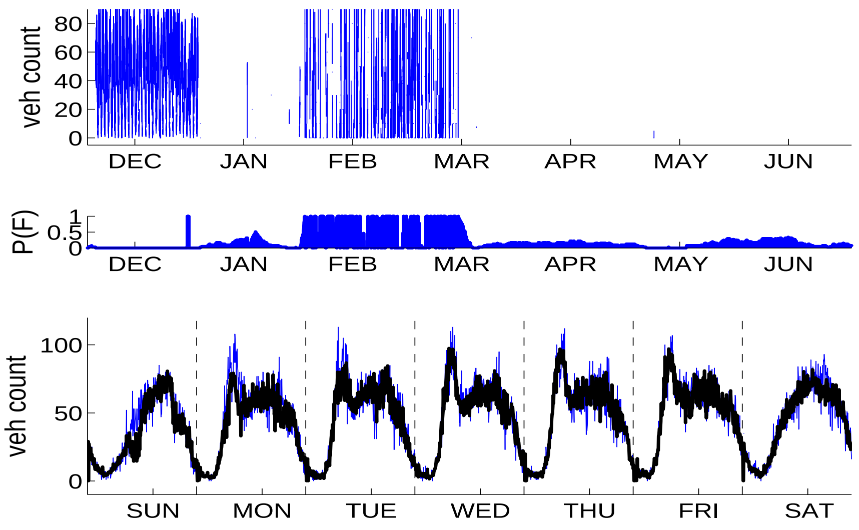
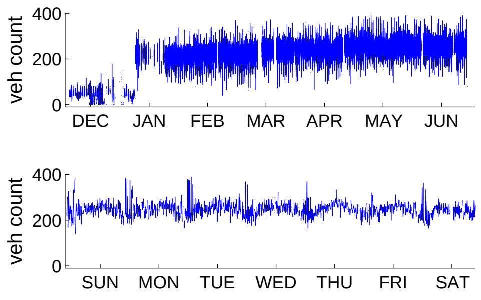


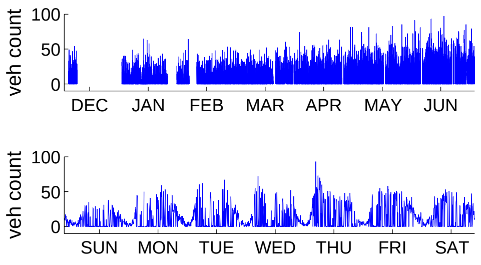
Figure 4.9: Fault tolerant model results for the corrupted signal example (refer to Figure 4.5). The corrupted portion of the signal is detected (center), and the model's inferred time-varying Poisson rate (bottom) fits the periodic signal present in the first months of the study.

Figure 4.8 is a plot of the same sensor as in Figure 4.4(a) and 4.6, where the original model was not able to find the normal traffic behavior during the middle of the day. The fault tolerant event model was able to detect the "stuck at zero" failure at the beginning of the study and find a much more accurate model of normal behavior.

Figure 4.9 shows the performance of the fault tolerant event model for a different type of failure. This is the same sensor shown earlier in Figure 4.5, where the measurements display periodic behavior followed by a signal that appears to be corrupted. During this questionable period, the measurements are missing more often than not, and unusually large spikes (many 50% higher than the highest vehicle count recorded during the first two months of the study) at unusual times of the day are often observed when the signal returns. The fault tolerant event model can now detect the corrupted signal and also in effect removes the faulty measurements when inferring the time-varying Poisson rate.



(a)



(b)

Figure 4.10: Examples of sensors with no regular periodic component. (a) Sensor with a corrupt signal that appears to be faulty for the entire duration of our study. There is no consistent periodic pattern to the signal, and large spikes often occur in the middle of the night when little traffic is expected. (b) Sensor signal with no consistent periodic component. There may be some periodic structure within a particular week (bottom panel), but there appears to be no consistent week-to-week pattern.

In the 50% to 100% row of the table, there are still a number of sensors where the fault tolerant event model is not able to discover a strong periodic pattern. However, it is arguable that the sensors in this group that remain would be harmful if used in further analysis; moreover the event fraction metric used in the results table provides a useful way to cull these questionable sensors.

About half of the sensors that remained in the over 50% category had large portions of missing data with too few non-missing measurements to form a good model. Others, such as in Figure 4.10(a) and (b), had no consistent periodic structure. Figure 4.10(a) is an example of a sensor that appears to be faulty for the duration of our study. The measurements for the sensor in Figure 4.10(b), on the other hand, appear to have some structure; morning rush hour with high flow, and low flow in the late evening and early morning as expected. However, the magnitude of the signal seems to alter significantly enough from week to week so that there is no consistent "normal" pattern. Even though the non-zero measurements during the day could perhaps be accurate measurements of flow, the unusual number of measurements of zero flow during the day along with the weekly shifts make the sensor output suspicious.

Before performing our large-scale analysis, we pruned the highly suspicious sensors. With most sensors, the fault tolerant event model makes a decent fit, and can be used to parse the corresponding time-series count data into normal, event, fault, and missing categories, and the results can be used in various analyses. When the model gives a poor fit (Figs. 4.10(a) and (b) for example), the parsed data are probably invalid, and may cause significant errors in later analysis if included. Therefore it is beneficial to exclude the outputs of such models (and the corresponding sensors).

The information found in Table 4.2 was used to prune the sensor list. We limited our evaluation to the 89% of the sensors that predicted less than 20% unusual event activity. The retained sensors sufficiently cover the study area of Los Angeles and Orange County, as seen in Figure 4.11. Removing sensors with questionable signals visually, without the use of a model, is not practical. Our model allows us to prune away sensors for which the model can not make any sense in an automated way.

4.3.2 Large-Scale Urban Network Analysis

After pruning the sensor list, 1508 of the original 1716 sensors remain, and the fault tolerant event model has learned normal, predictable traffic flow behavior of approximately 9 million vehicles daily in Los Angeles and Orange County. During the seven month study, these models detected over 270,000 events and almost 13,000 periods of sensor failure. Sensors experienced unusual event activity approximately once every 30 hours on average and sensor failure once every 26 days on average.

Figure 4.12 compares the typical durations of events and sensor failures. Sensor failures persisted longer than events by a factor of ten; events inferred by the model persisted 59 minutes on average, whereas sensor failures averaged 11 hours. Some presumably lengthy sensor failures were broken up into multiple consecutive day-long

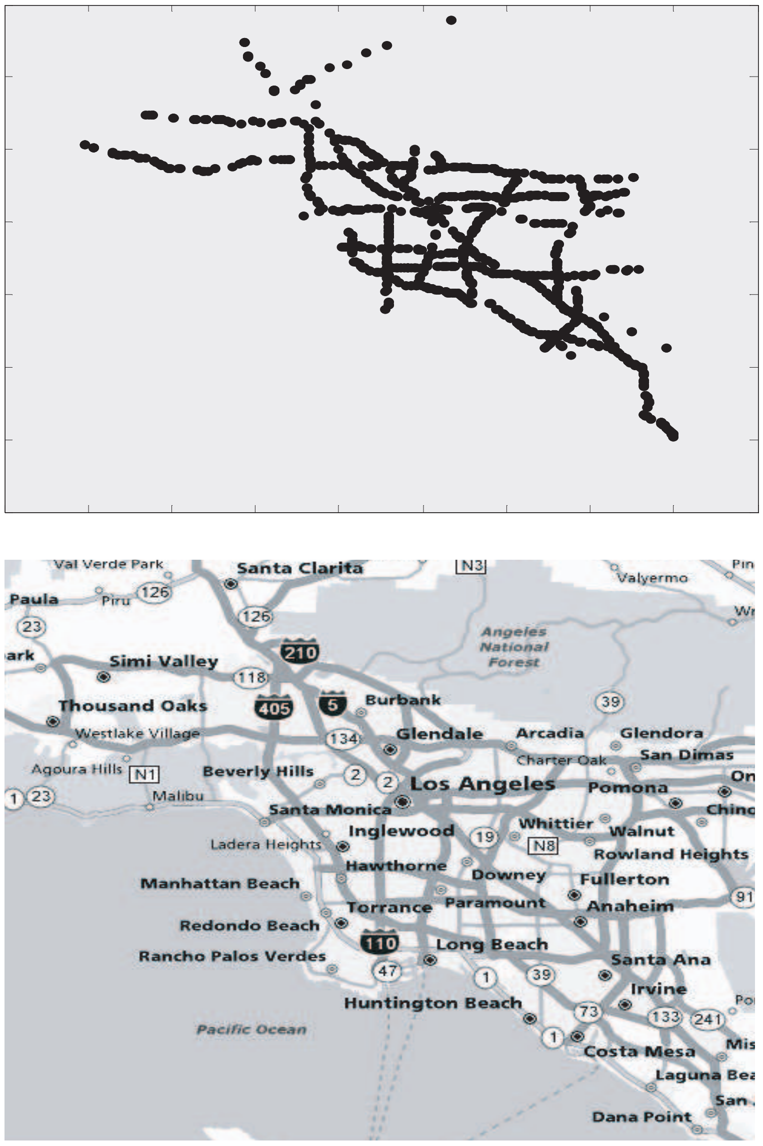


Figure 4.11: (top) The locations of the sensors used in our large scale analysis which remain after pruning sensors that had no periodic component that could be discovered by the fault tolerant event model, and (bottom) a road map of our study area, Los Angeles and Orange County.

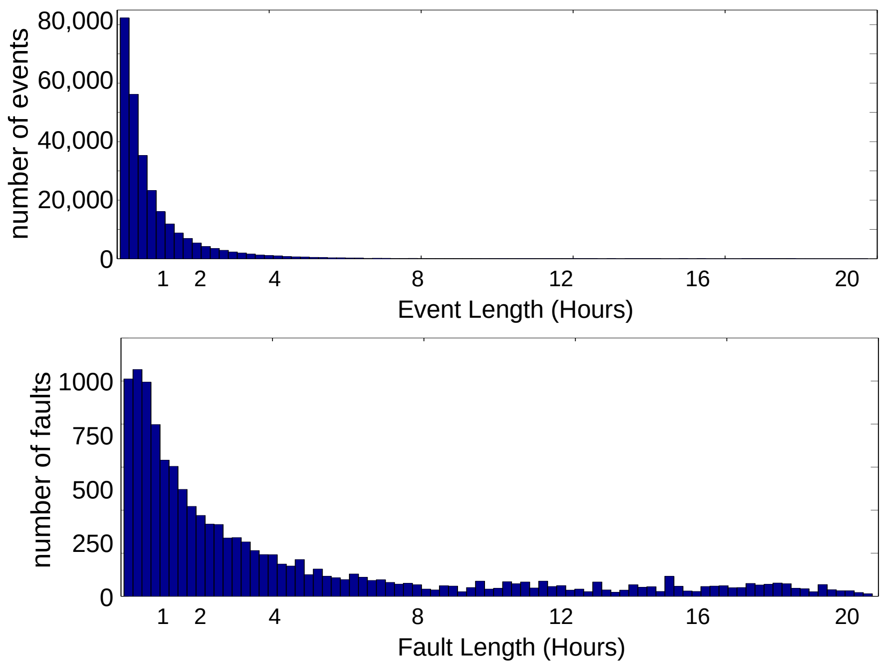


Figure 4.12: Histograms of the duration of events and sensor failures. The frequency of event lengths (top, 59 minute average) and fault lengths (bottom, 11 hour average) are displayed. Events and faults with lengths greater than 24 hours are not shown - the longest event lasted 2 days while the longest failure lasted 2 months.

failures since it is often difficult to detect sensor failures at night when there is little or no activity. The common stuck-at-zero failure, for example, often closely resembles the normal night time pattern. Chen et al. [14] also found it difficult to reliably detect failure events in loop sensor data at night and as a consequence limited fault detection to the time period between 5am and 10pm.

In our first analysis of the almost 300,000 periods of unusual and faulty activity, we examine the times of day and days of week that abnormal activity most commonly detected. Figure 4.13 shows a plot of the frequencies of unusual events and of sensor failures as a function of time of day and day of week. Sensor failures do not appear to have much of a pattern during the day. The troughs seen at night in the sensor failure line are due to the reduced night time activity discussed in the previous paragraph.

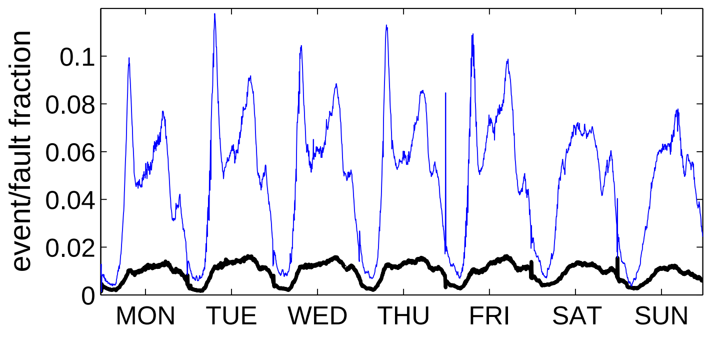


Figure 4.13: Unusual event frequency and fault frequency. The thin blue line with the greater magnitude shows the fraction of time that events were detected as a function of time of day, while the thick black line shows the fraction of time that faults were detected.

In contrast, the unusual event frequency plotted in Figure 4.13 does have a strong pattern. This pattern appears proportional to normal traffic patterns; that is, weekdays have spikes in unusual activity that appear to correspond to morning and afternoon rush hour traffic. The event pattern and the normal traffic flow pattern are compared in Figure 4.14. There is a strong relationship between the two (correlation coefficient 0.94), although there are significant bumps in the event activity each evening, particularly on weekend evenings, that depart from the normal flow pattern.

To explain the shape of the event fraction curve in Figure 4.14, it is reasonable to consider two types of event activity: events correlated with flow and events independent of flow. Traffic accidents might fall into the correlated event category because one would expect an accident on the freeway or on an artery close to a ramp to affect traffic patterns more when there is already heavy traffic. Much less of a disruption is expected if the accident occurs in the middle of the night. Traffic from large sporting events, which often occur in the evening, might fit the second type of event that is not correlated with traffic flow. In this case, the extra traffic is caused by a scheduled event whose timing does not necessarily correspond to the time of day that normally has the most congestion.

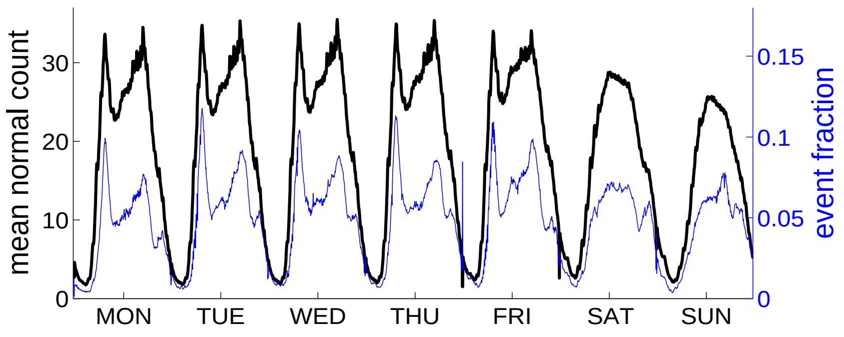


Figure 4.14: The event frequency (thin blue line, smaller magnitude, right y-axis) is plotted alongside the mean normal vehicle flow profile (the inferred Poisson rate averaged across the sensors) shown as the thick black line and using the left y-axis. The profiles are similar, with a correlation coefficient of 0.94.

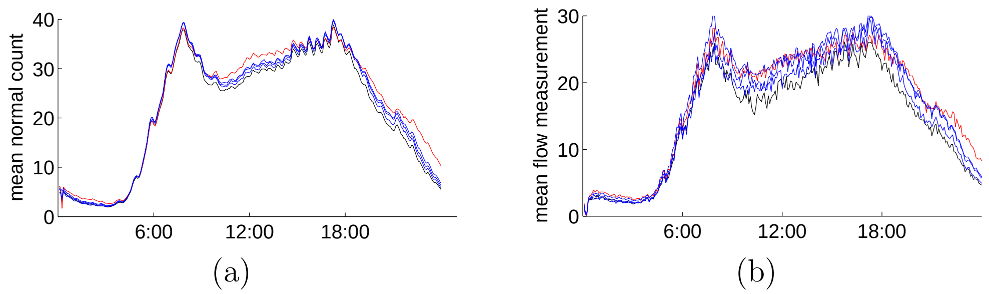


Figure 4.15: (a) The mean Poisson rate (across all sensors) for each weekday, superimposed. Although nothing links different weekdays, their profiles are quite similar, and the oscillation during morning and afternoon rush hour is clearly visible. (b) The mean vehicle flow rates for each weekday (average of raw measurements over all sensors), superimposed. The similarity in patterns is far less clear than in Panel (a).

Also of note in Figure 4.14 is that the flow patterns for weekdays look very similar. In Figure 4.15(a), the inferred time-varying Poisson rate profile for normal activity, averaged across all 1508 sensors, for each week day are plotted on top of each other. This figure shows that the average normal traffic pattern does not vary much between Monday and Friday. Note that in the fault tolerant event model used for the scale up experiments, there is no information sharing between weekdays, so there is nothing in the model that would influence one weekday to look similar to another. The similarity is much less clear in the raw data (Figure 4.15(b)).

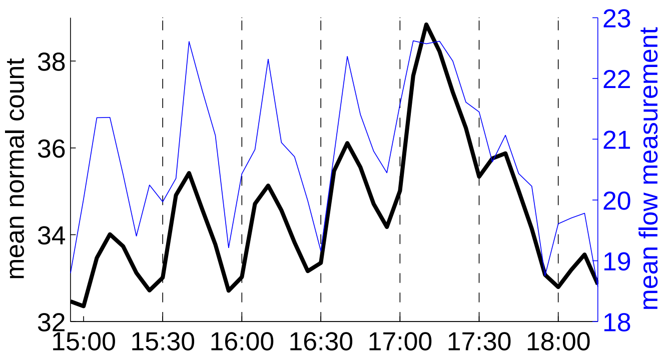


Figure 4.16: The mean normal vehicle profile (as in Figure 4.15) shown by the thick black line (using the left y-axis), is plotted against the actual mean flow (light blue line, right y-axis) for Mondays between 3pm and 5pm. The bumps that occur regularly at 30-minute intervals in the model's inferred time-varying Poisson rate are also present in the raw data, and are plausibly caused by people leaving work in bursts at the half hour intervals.

In Figure 4.15(a) there is also evidence of bumps occurring at regular intervals, especially in the morning and late afternoon. To investigate if the model was accurately reflecting a true behavior, we plotted the raw flow measurements for each weekday to compare with the model prediction. Figure 4.16 shows the plot of raw data and model profile for Monday, zoomed in on the afternoon period where the phenomenon is more pronounced. The raw data generally follows the same pattern as the model, confirming that these oscillations are not an artifact of the model. Interestingly, weekend days do not experience this behavior; and when individual ramps were examined, some showed the behavior and some did not. The peaks of the bumps appear regularly at 30 minute intervals. One plausible explanation [63] is that many businesses are located close to the highway, and people generally report to work and leave work on the half hour and on the hour. Thus, the bumps are caused by people getting to work on time and leaving work.

Note that this type of discovery is not likely to be made with the raw data alone. In Figure 4.15(b), the mean flow profiles for the weekdays appear to be potentially different because events and failures corrupt the observed data and mask true patterns

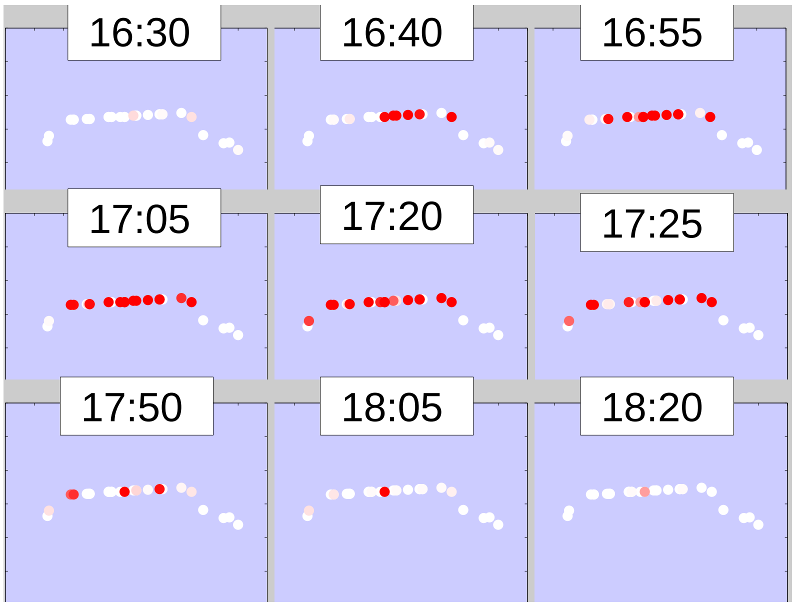


Figure 4.17: Example of a spatial event that occurs along a stretch of Interstate 10 in Los Angeles. Each circle is a sensor on an exit or entrance ramp. A sensor is colored white when no unusual event activity was inferred by the sensor's model during the previous 5 minute interval, and is darker red as the estimated probability of an unusual event (inferred by the model) increases. The 9 snapshots span a nearly two hour period where unusual activity spreads out spatially then recedes.

of normal behavior. It is not easy to see how similar these daily patterns are, and the half hour bumps in common between the days (Figure 4.16) are less likely to be spotted. An important point here is that the model (in Fig 4.15(a)) has automatically extracted a clear signal of normal behavior, a signal that is buried in the raw data (Fig 4.15(b)).

Lastly, we present an example of spatial analysis of the model output. Figure 4.17 shows an example of a “spatial event.” The series of plots span a two hour period beginning with a plot where one ramp saw unusual activity, followed by plots showing a spread of unusual activity detection. At its height, the unusual event activity spans a seven mile stretch of Interstate 10 in Los Angeles, which is followed by a gradual reduction of unusual event activity. One can imagine using information such as this to find the extent of disruption caused by an accident. Note: there is no spatial component to the current model; adding spatial links between the sensors could improve spatial event detection.

4.4 Summary of Contributions

- Introduced a negative event extension to the event model, resulting in a model that is more robust and that finds and characterizes periods of human behavior that are reflected in an abnormally low signal. This extension eliminates a preprocessing step needed in the previous event model.
- Examined periods of measurements that were corrupted by sensor failure; and demonstrated the limitations of a model that does not explicitly find and remove these measurements.

- Introduced a fault tolerant event model that explicitly models periods of sensor failure; and demonstrated how this model is robust to many of the limitations of previous sensor models.

- Presented a case study of a large-scale analysis of an urban traffic sensor data set in Southern California. The model was able to provide useful insights about an urban traffic data set that is widely considered to be problematic; and to perform an analysis of a type that has not to our knowledge been accomplished before. 100 million flow measurements from 1700 loop detectors over a period of seven months were parsed using the fault tolerant event model into normal activity, unusual event activity, and sensor failure components. Various analyses of event activity for this urban network were illustrated, including event duration and frequency analyses and spatial event analysis.

Chapter 5

Transportation Event Model

The event models described in previous chapters were general in the sense that they could be applied to any type of sensor that recorded count observations. Loop detectors fall into this general category, capturing a time-series of vehicle counts. However, loop detectors capture a second measurement of traffic as well, the fraction of time the sensor is covered by a vehicle. This second measurement is called occupancy. When traffic is in free flow (i.e. no congestion), the occupancy measurement mimics the flow measurement for the most part and provides little additional information (as seen in Figure 5.1(a)). However, when congested conditions occur, the occupancy measurement ceases to mimic the flow measurement (as observed in Figure 5.1(b)). Identifying congested conditions is important in many transportation applications, and the flow measurement alone is not sufficient for this task.

This chapter discusses a model that is specifically designed for use in transportation applications on data from loop sensors. This transportation event model was created as a result of a collaboration with the Institute of Transportation Studies (ITS) [35] at

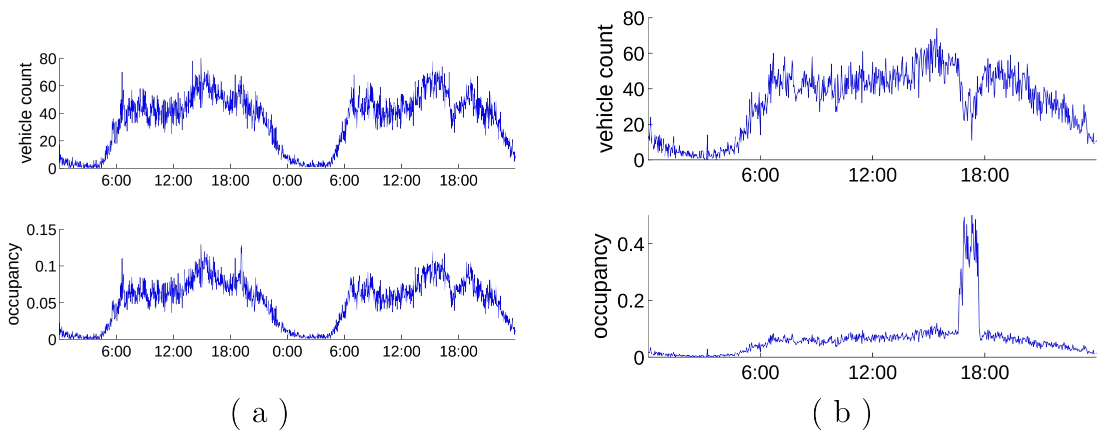


Figure 5.1: (a) Two days of observed flow measurements (top) and occupancy measurements (bottom). Traffic is in free flow conditions for the entire two days. (b) One day of observed flow and occupancy measurements for the same sensor as in (a) but with a period of congested traffic in the evening.

the University of California, Irvine. The transportation event model is more sensitive to traffic events involving congestion, and results in a richer analysis of traffic events.

5.1 Mainline Traffic Characteristics

This thesis uses loop detectors extensively to test the various event models. Up to this point, however, only loop detectors embedded in freeway ramps were used in the experiments. The traffic measured by these ramp sensors is less affected by congestion than the traffic measured by mainline sensors (i.e. sensors on the freeway through lanes excluding high occupancy vehicle (HOV) lanes). While the flow-only event model can provide valuable information about mainline traffic, the occupancy measurement is needed to provide information about the role of congestion in events and traffic patterns.

The effects of traffic congestion distinguishes mainline vehicle count processes from other count processes. A freeway has a maximum capacity level, and when the density of traffic approaches capacity, traffic flow becomes less efficient. When additional

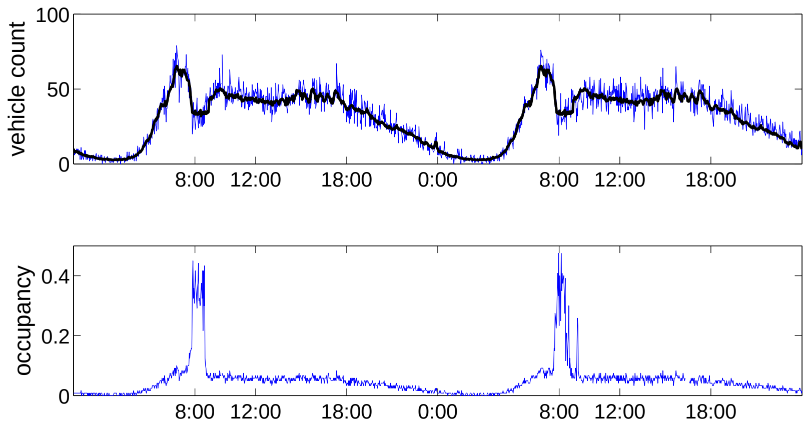


Figure 5.2: Two weekdays of flow measurements (light blue line, top panel) superimposed with the model's inferred normal flow profile (thick black line), along with occupancy measurements (bottom panel) for a sensor at a location with heavy morning rush hour congestion.

traffic is introduced to a lane that is near capacity, drivers decrease the speed of their vehicles; this causes a decrease in flow volume despite the increase in demand. This phenomenon is illustrated in Figure 5.2. For many sensors in our study area, morning flow volume peaks around 8 am. The traffic at the sensor illustrated in Figure 5.2, however, is normally highly congested during morning rush hour. The vehicle flow (top panel) at this sensor peaks at around 7 am, then as traffic becomes more congested, vehicle speeds decrease causing an increase in the occupancy measurement (bottom panel) and a decrease in the vehicle flow.

Applying the flow-only models described in previous chapters to mainline sensor data could result in misleading analyses of traffic events, specifically those events that are influenced by congestion. The morning traffic at the sensor used in the previous example (Figure 5.2) is highly congested on a normal day. If a morning event occurs at this sensor that decreases demand enough so that the normally congested conditions are relieved and the traffic is in free flow, the observed flow could counter-intuitively increase (as compared to the normally congested flow) resulting in a positive (high flow) event detected by the event model. High flow events are most often caused by

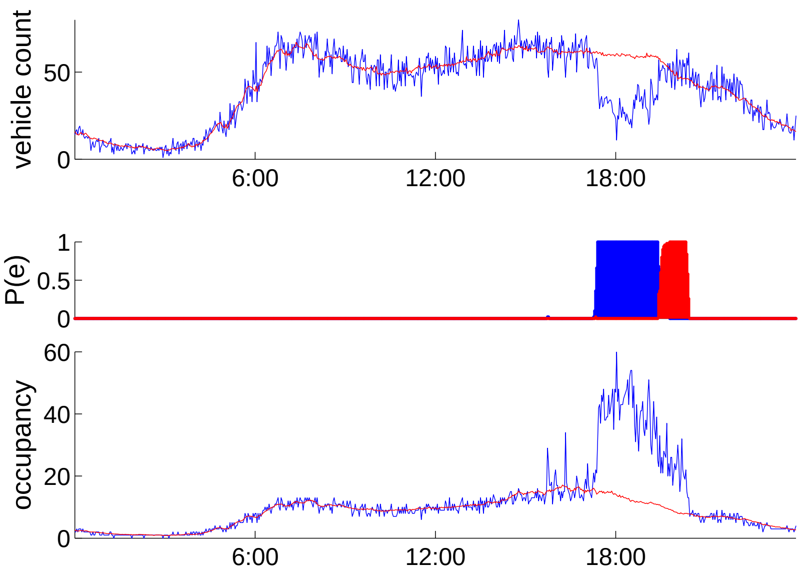


Figure 5.3: A day with a high occupancy low flow event in the evening. The blue line represents the observed measurements for both plots and the red line indicates the normal expected behavior. At the tail end of the event, the observed flow returns to normal, but there continues to be abnormal delay as indicated by the occupancy signal remaining abnormally high.

an increase in demand, but without considering the occupancy measurement it would be difficult to correctly conclude that this event was instead caused by a decrease in demand.

Negative (low flow) events can also be misinterpreted without the occupancy measurement. For example, the top panel of Figure 5.3 shows a sensor with a low flow event in the evening. Without the occupancy measurement, one can not determine whether the low flow is due to lower demand (such as what often occurs on weekday mornings during a holiday) or due to abnormally high congestion (as is the case for the event in Figure 5.3) which can occur after an accident or which can be caused by extra traffic attributed to a large sporting event, for example.

Figure 5.2 illustrates another advantage for including occupancy in the model. When traffic is mildly congested, the traffic flow (the number of vehicles counted in each time interval) may not change, but the traffic speed can decrease and delay might increase significantly. At the end of the low flow event in the evening in Figure 5.2 (the time of the event indicated in blue in the second panel) the flow measurement (light blue line top panel) returns to normal (dark thick line top panel), however the occupancy measurement (light line bottom panel) remains elevated above normal (dark line bottom panel) for another 45 minutes. Although flow has returned to normal levels, the effects of the event have not disappeared. There remains considerable delay in traffic that is normally free flowing during this 45 minute period (the time span of this high occupancy, normal flow event is indicated in red in the second panel).

In order to make the event model more sensitive to detecting congestion-related events, and to analyze the effects of congestion-related activity in mainline traffic, the occupancy measurement must be incorporated into the model.

5.2 Transportation Event Model

This section explains the modifications made to the event model when the occupancy measurement o_t at each time t is included as evidence given to the model in addition to the flow measurement f_t.

Figure 5.4 shows three time segments of the transportation event model. The occupancy process is modeled as a NHPP similar to the flow process (described in Chapter 3.2.1) and the two processes are connected by a single event process.

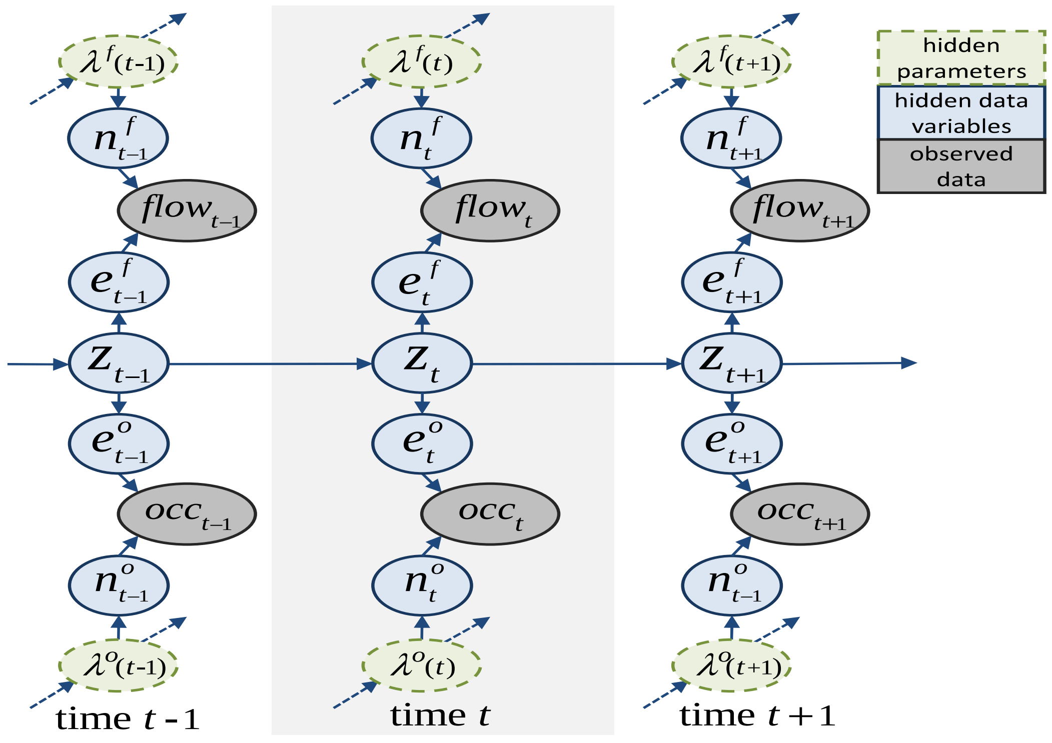


Figure 5.4: Three time segments of the graphical model for the transportation event model.

5.2.1 Occupancy Process

Although flow and occupancy are modeled similarly, there are some notable differences in the processing of the occupancy measurement. The Poisson distribution is a discrete distribution, however the occupancy measurement is reported as a continuous value between 0 and 1 that indicates the fraction of time a vehicle covers the sensor. The occupancy measurement can be converted to a discrete value by returning it to the original occupancy time value that was used to calculate the occupancy fraction; for example an occupancy fraction of 0.5 for 2 minute time interval, indicates that the sensor was covered for 60 seconds out of the 120 second time interval. So a preprocessing step is needed to convert the occupancy measurement to a discrete time value. In the experiments in this chapter the occupancy fraction measurement is converted to seconds (rounded to the nearest second).

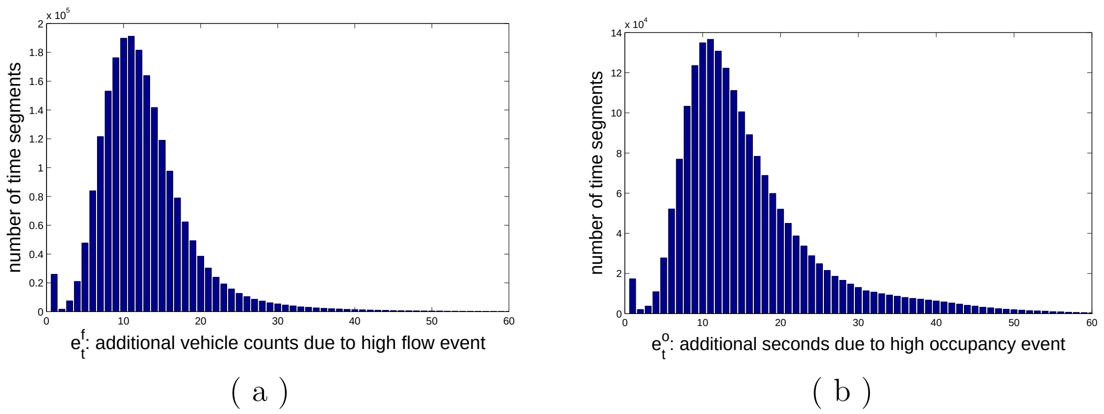


Figure 5.5: Histograms of the number of extra observations (counts in the flow case and seconds in the occupancy case) due to (a) high flow event activity and (b) high occupancy event activity.

The transformation of occupancy to a discrete value was made for ease of implementation. The transformation, however, results in a small amount of information loss. A possible alternative could be to use a Beta distribution to model the normal occupancy process, $n^o_t \sim Beta(n^o_t, a, b)$, and then learn parameters for the event process for both positive and negative events (eg. $e^o_t \sim Beta(n^o_t, a + a^+, b + b^+)$ where $a^+ > 0$ and $b + b^+ > 0$ for positive events). However, we leave such a technique as possible future work.

A second difference in the occupancy process is in the nature of event activity. A diverse range in the magnitude of event activity is reflected in the flow and occupancy measurements. Figure 5.5 shows histograms of the the number of extra observations (counts in the flow case and seconds in the occupancy case) due to (a) high flow event activity and (b) high occupancy event activity, from over 2 million event measurements from approximately 1000 mainline traffic sensors. The number of extra vehicles attributed to high flow event (Figure 5.5(a)) appears to form a smooth distribution that could be effectively modeled using a single negative binomial distribution.

$$e^f_t \sim \mathrm{NBin}(e^f_t; a^f, b^f/(1 + b^f)) \tag{5.1}$$

where NBin is the negative binomial distribution with parameters a^f and $b^f/(1+b^f)$ where the superscript f indicates the flow process. The choice of modeling the event count with a negative binomial distribution is discussed in Chapter 3.2.2.

The distribution of the extra number of seconds attributed to high occupancy event (Figure 5.5(b)), however, has a much thicker tail at the high end of the distribution. This extra probability mass at the higher end of the scale is often more pronounced when looking at the occupancy event distribution for an individual sensor, often creating a bi-modal distribution. This second mode is due to the attributes of congested traffic. When traffic nears capacity, small increases in traffic demand can cause large jumps in the occupancy measurement with relatively little impact on the flow measurement.

We find, then, two types of occupancy events in practice:

1. Events that do not involve a transition between congested traffic and free flow (un-congested) traffic. Examples include when (a) normal traffic is not congested (free flow) and the additional activity due to an event is not enough to cause congestion, and when (b) normal traffic is congested or near congested and event activity increases congestion. An illustration of this type of event can be seen in Figure 5.1(b). On the second day there is a negative flow, negative occupancy event just before 6PM. The magnitude of the drop in the occupancy measurement is similar to the magnitude of the drop in the flow measurement.

2. Events that do involve a transition between congested traffic and free flow (un-congested) traffic. The jump (or drop) in the occupancy measurement during these events is substantially greater than the change in magnitude of the flow measurement during the event.

To more accurately model the distribution of occupancy event activity, we use a mixture of two negative binomials

$$e_t^o \sim w_1 * \text{NBin}(e_t^o; a_1^o, b_1^o/(1+b_1^o)) + w_2 * \text{NBin}(e_t^o; a_2^o, b_2^o/(1+b_2^o)) \tag{5.2}$$

where a_1^o and b_1^o are parameters of the negative binomial distribution of the first component and a_2^o and b_2^o are parameters of the second component and where the superscript o indicates the occupancy process, and where the mixture weights sum to one, $w_1 + w_2 = 1$. The event parameter settings used for the experiments in this chapter are defined in Appendix A.4.

The distribution of event activity (Figure 5.5) is partly influenced by the parameters of the event distributions (Equations 5.1 and 5.2), but the distribution shape of the inferred event activity is not very sensitive to the parameters of the event distribution. For example, when the event model was run using identical parameterizations for both the occupancy event and flow event activity (both using Equation 5.1 and the flow settings listed in Appendix A.4), the shapes of the distributions seen in Figure 5.5 appeared much the same.

5.2.2 Event Process Changes

The event process for transportation event model also changes from earlier models. Rather than a general positive or negative event classification, an event is now characterized in terms of a combination of high, low, or normal flow and high, low, or normal occupancy. We define a system with 8 states, one state for normal activity, six event states and a sensor failure state.

$$z_t = \begin{cases} 0 & \text{if there is no event at time } t \\ 1 & \text{if there is a high flow, high occupancy event at time } t \\ 2 & \text{if there is a high flow, low occupancy event at time } t \\ 3 & \text{if there is a low flow, low occupancy event at time } t \\ 4 & \text{if there is a low flow, high occupancy event at time } t \\ 5 & \text{if there is a high flow, normal occupancy event at time } t \\ 6 & \text{if there is a normal flow, high occupancy event at time } t \\ 7 & \text{if there is a sensor failure at time } t \\ \textit{unused} & \text{if there is a low flow, normal occupancy event at time } t \\ \textit{unused} & \text{if there is a normal flow, low occupancy event at time } t \end{cases} \tag{5.3}$$

The combinations low flow, normal occupancy and normal flow, low occupancy were not observed in practice and were removed from the event state space. The state transition settings are similar to previous models, however event state to event state transitions are more common in this model (for example the transition from a low flow, high occupancy event directly to a normal flow, high occupancy event in Figure 5.3). The transition parameter settings used for the experiments in this chapter are defined in Appendix A.4.

The inference and learning proceeds as described in Chapter 3.3 with relatively minor modifications in the implementation.

5.3 Case Study 3: Accident Characterization and Event Ranking

[1] This case study shows the application of the combined event model for two transportation specific problems, spatiotemporal accident characterization and traffic event severity ranking.

5.3.1 Data

Nine months of measurements from 1150 loop detectors on the 57 and 405 freeways in Orange County, California were used for the experiments in this section. The raw (30-second aggregate) measurements were obtained using the PeMS [20] (Performance Measurement System) data clearinghouse maintained by the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley, in cooperation with the California Department of Transportation. The 30-second data were further aggregated into two-minute length bins before running the event model. While traffic data is normally analyzed using 30-second or 5-minute intervals, we found the 2-minute intervals to be better suited for our purposes. For example, when viewing changes in the spatial area representing the effects of an accident from one time segment to the next, the changes in the boundaries were too disjointed using the 5-minute intervals (i.e. large changes were observed in the boundary). Using 2-minute sized intervals, however, resulted in relatively smooth transitions in the spatial boundary between time segments. Reducing the interval length to 30-seconds did not appear to provide benefits to justify the additional computational costs of modeling the smaller interval lengths.

[1]The plots and tables used in the experimental analysis in this section were created as part of joint work with Nicholas Navaroli from the UCI Datalab [53].

5.3.2 Event Severity Ranking

In this section, we describe a simple method for finding the spatiotemporal impact of an event, given the output of the event model. Here, an event is defined in terms of multiple sensors that are spatially connected, rather than a single sensor as in previous chapters. Since events span space and time, events found in a history of freeway data can be ranked based on the duration of the event, or the distance span of the event, or the magnitude of the deviation from the normal traffic pattern. Alternatively, a metric that combines these three measures can be used to rank events.

Using the six event states defined in the transportation event model (listed in Equation 5.3), we examine several different types of events, each of which has a different impact on the normal traffic pattern, and rank them using a metric appropriate for the event type.

Finding the Spatiotemporal Extent of an Event

Given a history of measurements and the output of the event model for all of the sensors along a stretch of a freeway, and given the spatial location of the sensors (lane number, post mile, and flow direction), we used a simple method to find all maximal contiguous regions of event activity in time and space.

A 3-dimensional grid is formed in space and time where each sensor is connected to the sensors in neighboring lanes at the same post mile, to the sensors in the same lane north and south of itself, and to itself in the previous time segment and the immediate future time segment. Starting with the location and time of a sensor indicating event activity ($p(event) > .5$), the spatiotemporal boundary of the event is found by recursively extending the event to all neighbors in time and space, for neighbors that also are indicating event activity. Our method used a preprocessing step to interpolate event probability values for missing or bad sensor measurements with at least one non-missing, good neighboring sensor.

The model used in the experiments in this section for each sensor is independent of neighboring sensors. A future potential extension of this work (discussed in more detail later in Chapter 8.2) would be to use a more principled approach to finding the spatiotemporal outline of events by creating a multi-sensor model with links between neighboring sensors.

The events found by our method differ from the spatiotemporal characterizations of accidents in previous work [62], where events are caused by accidents that originate at a single point in space and time, propagate from that single point in a specific manner, and also clear in a specific manner, and the event shapes are constrained to match this description. The events found by our method are more general; they can be caused by a single incident with a specific time and location (such as an accident), or they can be caused by multiple incidents (such as multiple accidents at points near enough in space and time to overlap and create a more severe event), or they can have a cause that can not be linked to specific points in time and space (such as events caused by holiday traffic that can appear at multiple locations at the same time).

Ranking Events by Delay; Accident Event Analysis

Events that cause additional delay (i.e. more delay than normally experienced for that time of day and day of week) are of particular importance in transportation research. For example, the goal of the accident characterization tool [62] discussed later in Section5.3.3 is to find the impact of accidents in terms of additional delay, where

$$\text{Additional Delay} = \sum_{s,t} L_s \left(\frac{1}{\hat{v}_{s,t}} - \frac{1}{\bar{v}_s(t)} \right) f_{s,t} \tag{5.4}$$

This equation estimates the extra (above normal) delay caused by the accident, where

- L_s is the length of the freeway segment associated with sensor s
- $\hat{v}_{s,t}$ is the estimate of the velocity of the vehicles going over sensor s at time t calculated using the occupancy and flow measurements for sensor s at time t
- $\bar{v}_s(t)$ is the mean estimated velocity of vehicles going over sensor s given a one year history (52 measurements) of occupancy and flow measurements at the same time of day and day of week as t, and
- $f_{s,t}$ is the flow measurement for sensor s at time t.

This metric gives an estimate of the extra number of $vehicle \cdot hours$ caused by the accident.

The velocity calculation can be quite complex [37]. Individual velocity models can be constructed for each sensor given factors such as the density of truck traffic (which can vary according to lane number), freeway, time of day, and day of week. For simplicity, the velocity measurements used in our experiments were all based on a mean vehicle length of 20 feet,

$$\begin{aligned} \hat{v}_{s,t} &= \frac{o_{s,t}(\text{seconds})}{f_{s,t}(\text{cars})} \cdot \frac{20(\text{feet})}{1(\text{car})} \cdot \frac{1(\text{mile})}{5280(\text{feet})} \cdot \frac{3600(\text{seconds})}{1(\text{hour})} \\ &= 13.64 \cdot \frac{o_{s,t}}{f_{s,t}} \end{aligned} \tag{5.5}$$

Table 5.1: North 57 high occupancy events ranked by extra delay in $vehicle \cdot hours$. PM indicates post mile.

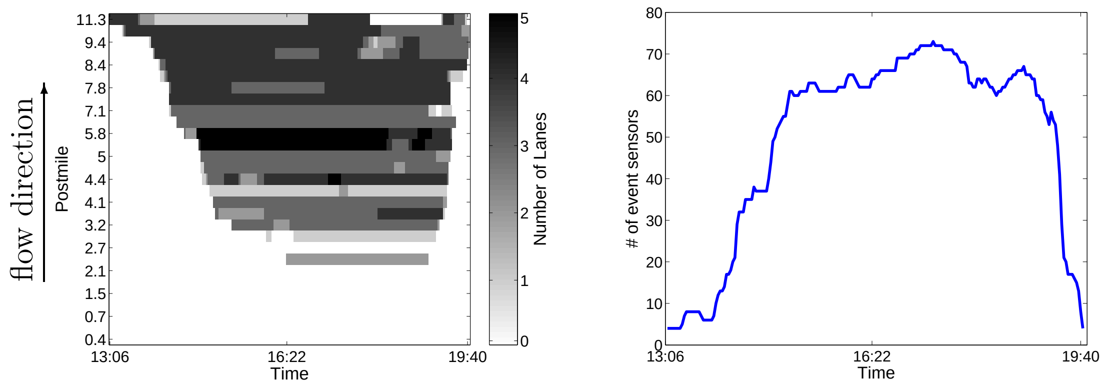
Event Date	Begin Time	End Time	Time Span (HRS)	Begin (PM)	End (PM)	Delay ($vehicle \cdot hours$)	Ave Speed (MPH)
30-Jul-07	13:06	19:40	6.6	2.2	11.3	8445	15.2
13-Dec-07	12:54	19:56	7.0	1.0	11.3	6022	18.8
24-Apr-07	15:08	20:30	5.4	0.7	11.3	5928	13.5
20-Nov-07	15:08	20:32	5.4	1.0	11.3	5421	17.6
18-Jul-07	13:56	19:56	6.0	0.4	11.3	5268	15.6
29-Aug-07	14:00	19:56	5.9	0.7	10.5	4504	19.9
14-Sep-07	13:28	19:50	6.4	2.9	11.3	3999	21.1
19-Sep-07	14:46	19:52	5.1	0.7	11.3	3941	20.7
25-Jul-07	15:34	20:02	4.5	0.5	10.5	3768	18.3
5-Sep-07	14:00	20:00	6.0	0.4	11.3	3592	21.0

producing a velocity estimate in units of miles per hour (MPH), where $o_{s,t}$ is the occupancy measurement for sensor s at time t, and where $o_{s,t}$ has been converted to seconds for use in our model.

Instead of the mean velocity estimate that was used in the previous delay calculation (Equation 5.4), the delay metric used in the experiments in this section incorporate the models of normal flow $\lambda^f(t)$ and occupancy $\lambda^o(t)$ learned by the event model.

$$\text{Additional Delay} = \sum_{s,t} p_{s,t}(event) L_s \frac{1}{13.64} \left(\frac{f_{s,t}}{o_{s,t}} - \frac{\lambda_s^f(t)}{\lambda_s^o(t)} \right) f_{s,t} \tag{5.6}$$

where $p_{s,t}(event)$ is the probability of the event for sensor s at time t. The advantage of this approach over using mean velocity is that it can use the models described in this thesis to automatically remove abnormalities in the historic training data due to sensor failure, construction, and other events, and hence provide a more robust estimate.



(a) (b)

Figure 5.6: An event with one point cause: plots illustrating the spatiotemporal extent of the top ranked high delay event as seen in Table 5.1. (a) A plot of the number of lanes (darker blocks indicating more lanes) showing event activity for sensors grouped by post mile (y-axis) for the time span of the event (x-axis). (b) A plot showing the total number of sensors showing event activity at each time segment of the event.

The delay metric of Equation 5.6 was applied to events found along the 12 mile stretch of the 57 Freeway in Orange County, California for the last nine months of 2007. The top ten events that caused the most additional delay are listed in Table 5.1. The 57 North freeway has a higher demand in the afternoon than in the morning, so incidents that occur in the afternoon have a greater potential to disrupt traffic. Accordingly, all of the ten most severe events occurred in the afternoon.

Figure 5.6 shows an example of an event shape that is typical of an event caused by a single accident. Figure 5.6(a) shows the spatiotemporal extent of the top ranked event in Table 5.1. Each block in the plot shows the number of lanes that had sensors which showed event activity for a given time and post mile; the darker the block, the more lanes at that post mile with event activity predicted. Only “good” sensors (sensors with non-missing, non-failure measurements) were used in the plots. Interpolated event values for “bad” sensors used to help find the event borders were not included, so the number of possible sensors at each post mile could be as few as zero and as many as five. The gradual slope, down and to the right from the top left corner of the event space in Figure 5.6(a), is characteristic of an “accident shock-wave” [62] that shows the beginning of the event in time. The flow of traffic is in the northward direction, so an accident to the north of this section of highway first affects the sensors most north (top left corner of the event area). Then as time progresses (movement to the right along the x axis) traffic backs up, and sensors to the south (lower post mile, downward direction along the y axis) begin to show event activity. Traffic remains at an unusually high congestion for a time with no gaps or holes in the event area, and then the clearing wave (the boundary on the right side of the plot showing the ending of the accident effect in time), shows a relatively smooth transition back to normal. Figure 5.6(b) illustrates another way to view the spatiotemporal extent of an event, a line plot of the number of sensors involved in the event at each time segment. This shape is also consistent with an event caused by a single accident: a steady increase in the number of sensors showing event activity as time increases, a single peak at the height of the accident's effect, followed by a steady decline.

Our method for finding the spatiotemporal extent of events for the experiments in this section does not constrain the allowable shape of the events as did the method used in our collaborators' previous work [62] (used to find the impact of a single accident as discussed later in Section 5.3.3). Because the shape is not restricted, spatiotemporal event shapes can be found that are not consistent with what we would expect from a single accident. The next set of figures illustrate an event caused by a single accident and an event that has more than one cause.

Figure 5.7(a) shows an example of a delay event (the third ranked delay event in Table 5.1) that is not consistent with that caused by a single accident, but appears to have two point causes. An Anaheim Angels home baseball game ended around 15:30 introducing post-game traffic to the freeway. The additional flow caused an unusual level of congestion for a two mile span north of the stadium for about 90 minutes. At that point an incident occurred about 10 miles north of the stadium. While there

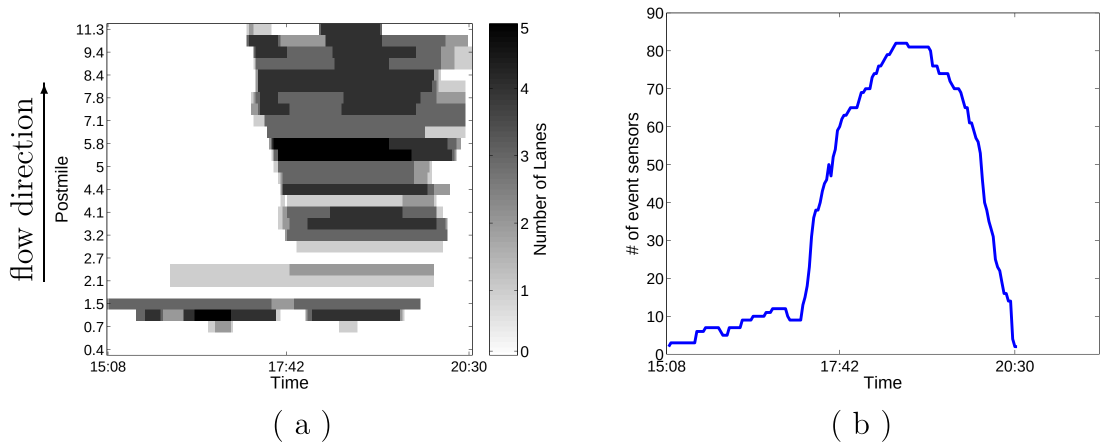


Figure 5.7: An event with two point causes: plots illustrating the spatiotemporal extent of the third ranked high delay event as seen in Table 5.1. (a) A plot of the number of lanes (darker blocks indicating more lanes) showing event activity for sensors grouped by post mile (y-axis) for the time span of the event (x-axis). (b) A plot showing the total number of sensors showing event activity at each time segment of the event.

are clearly two separate causes for this event, it would be difficult to separate the impacts of the two, particularly in the case of the second event. The extra post game traffic undoubtedly increased the congestion in the second incident. The two event causes are also reflected in the total event sensor versus time plot in Figure 5.7(b). The sharp increase in the number of event sensors corresponds to the beginning of the second event.

Ranking Events by Extra Vehicle Miles - Event Popularity

While accidents are often the cause of one kind of event found by the model (high occupancy, low flow or high occupancy, normal flow event states), other event states can reflect additional types of unusual traffic behavior. This section analyzes unusual traffic behavior that is reflected by high flow event states. In these events, an unusually large number of vehicles are on the road given the time of day and day of week. These events give an indication of event popularity; although as discussed in

Table 5.2: North 57, high flow events ranked according to extra $vehicle \cdot miles$.

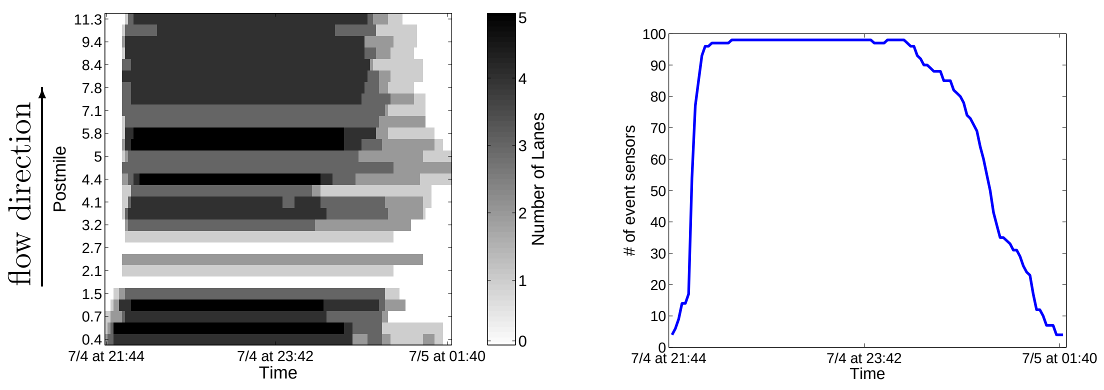
Begin Time	End Time	Extra $Veh \cdot Miles$	Event Cause
7/4 at 21:44	7/5 at 01:40	67,545	Independence Day
9/8 at 19:26	9/9 at 03:34	48,805	Angels game
7/29 at 18:54	7/30 at 00:18	35,531	Angels game
4/28 at 04:40	4/29 at 01:20	32,914	construction?
8/4 at 19:50	8/5 at 00:08	30,849	Angel Stadium - Harvest crusades
8/25 at 20:02	8/26 at 00:06	30,431	Angels game
6/19 at 18:48	6/20 at 01:12	30,271	Angels game
8/23 at 18:46	8/24 at 00:20	30,171	Angels game
6/21 at 21:06	6/22 at 04:24	26,749	construction
8/22 at 19:38	8/23 at 00:22	26,539	Angels game

Table 5.3: South 57, high flow events ranked according to extra $vehicle \cdot miles$.

Begin Time	End Time	Extra $Veh \cdot Miles$	Event Cause
12/24 at 20:56	12/25 at 02:36	43396.12795	Christmas Eve
11/22 at 19:14	11/23 at 00:46	42003.87904	Thanksgiving
12/25 at 19:12	12/25 at 23:52	33481.57826	Christmas Day
4/8 at 17:28	4/8 at 22:36	27107.00147	Angels game?
5/20 at 09:12	5/20 at 14:10	22926.00652	construction + Doheny blues festival - Angel Stadium
9/7 at 20:52	9/8 at 01:58	21817.23895	night before Columbus Day?
5/13 at 18:00	5/13 at 22:58	18646.21976	?
7/4 at 21:50	7/5 at 00:32	15970.67546	Independence Day
6/17 at 18:38	6/17 at 23:18	15943.82359	?

Section 5.1, an event generating additional traffic during an already congested period could actually cause a decrease in flow rather than an increase in flow. Analyses of this type of event may be of interest in applications such as large event planning and realistic traffic simulation (as opposed to simulating normal traffic patterns only).

Tables 5.2 and 5.3 show the top ten events for the North 57 and South 57 Freeways ranked according to extra vehicle miles. The probable cause of each event is listed in the last column. Holiday traffic and extra traffic due to events at Angel Stadium



(a) (b)

Figure 5.8: High flow event caused by July 4 holiday traffic: plots illustrating the spatiotemporal extent of the top ranked high flow event as seen in Table 5.2. (a) A plot of the number of lanes (darker blocks indicating more lanes) showing event activity for sensors grouped by post mile (y-axis) for the time span of the event (x-axis). (b) A plot showing the total number of sensors showing event activity at each time segment of the event.

(particularly Angels baseball home games) appear to be the primary causes of the higher ranked high flow events for this section of the freeway.

Figure 5.8 shows the spatiotemporal extent of the top ranked flow event for 57 North listed in Table 5.2, which was presumably caused by extra traffic following Independence Day firework celebrations. Unlike events caused by accidents that have a single point of origin in time and space, an event caused by holiday traffic has no one point of origin. This is reflected in the event shape seen in Figure 5.8(a). The event shape caused by an accident has a distinct boundary early in time (for example, the left boundary of the single accident event seen in Figure 5.6(a)); the further upstream of the event (south in the example), the later in time that the start of the event signature appears as it takes time for the traffic to back up. The left hand boundary that shows the beginning of the event in time for the holiday traffic event (Figure 5.8(a)); however, shows that the effect of the event occurs nearly simultaneously regardless of location. So the constraints considered for accident characterization to restrict allowable event shapes and to separate events that are caused by more than one accident, may not be applicable for high flow events.

The plot of the number of sensors involved in the event at each time segment for the July 4 event is shown in Figure 5.8(b). This event is unusual in that the effect of the event is observed in 100 percent of the sensors for the majority of the event duration. This is reflected in the flat horizontal line in the plot; normally the number of sensors affected by the event changes from one time to the next. Missing sensors and sensors experiencing failure are not included in these plots. The shading in the post mile plots such as Figure 5.8(a) reflect the number of "good" (non-missing, non-failure) sensors that are predicting event behavior. The white horizontal lines in Figure 5.8(a) around post miles 2 and 3, for example, appear in this case because there are no "good" sensors at these post miles reporting measurements for the duration of this event.

Construction also appeared to cause some highly ranked high flow events. Construction causes a different type of high flow event, however. Rather than an abnormal increase in *total* vehicle flow on the freeway, construction that shuts down a lane (or more) of traffic diverts an unusual level of flow to the other lanes, triggering a high flow event (even though the total flow for section of freeway may be unchanged from normal conditions). Figure 5.9 is an example of what occurs during a high flow event caused by construction. The figure shows flow plots for 9 sensors in the event area during the time of the event. The light blue line in each plot is the observed measurement during the time of the event, and the thick black line is the normal flow $\lambda^f(t)$. The three columns show sensors in lanes 1, 2 and 4. Sensors in each row are from the same post mile; rows 1, 2 and 3 have sensors from post miles 1, 5, and 6 respectively). The sensors in this figure show that there was slightly higher than normal flow between 21:00 and 23:00. The California Department of Transportation

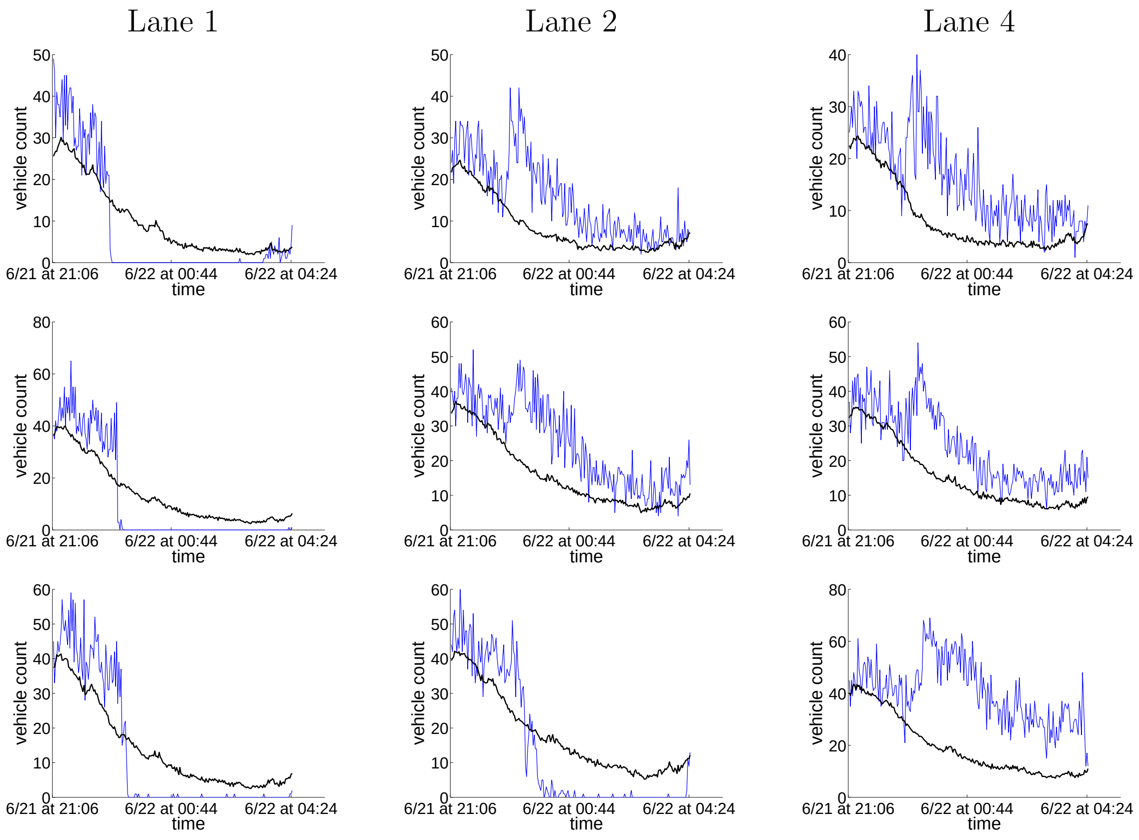


Figure 5.9: Flow plots for 9 sensors during a high flow event caused by construction. The light blue line is the observed measurement during the time of the event, and the thick black line is the normal flow $\lambda^f(t)$. The three columns show sensors in lanes 1, 2 and 4. Sensors in each row are from the same post mile (at post mile 1, 5, and 6). At around 23:00 the flow in lane 1 (and lane 2 in one case) drops to zero as the lanes are closed off for construction. Extra flow is diverted to the other lanes.

event log states that construction closure on the 57 North Freeway started at 23:18. At that time, the flow measurements for the sensors in lane 1 in Figure 5.9 (and lane 2 for the sensor at post mile 6) dropped to zero as the lanes are closed off for construction. Extra flow was diverted to the other lanes resulting in a significantly higher than normal flow for the next 5 hours at which time the closure ended and traffic returned to normal.

Anaheim Angel home baseball games were the dominant cause of extra flow events for the 57 North, accounting for 19 of the top 25 ranked high flow events. Surprisingly, Angel games were not as prominent of a cause of high ranked events for the 57 South.

Angel Stadium is located at the south end of this section of the freeway (around post mile 1.5), and one would expect extra traffic (reflected by a high flow event) on the 57 South before the game and on the 57 North after the game. The post-game event signal is clearly apparent in the northbound lanes, but the pre-game southbound signal is not as prevalent.

High Flow Low Occupancy Events

The six event states of the model (listed in Equation 5.3) reflect different types of abnormal traffic. When searching for the high flow events for the experiments in the previous section, all three high flow event states were included (states 1, 2, and 5). In this section, we investigate a subset of these high flow events, namely high flow, low occupancy events (state 2).

Abnormally high flow measurements are most commonly accompanied by abnormally high occupancy measurements. High flow, low occupancy events are relatively rare and occur only under special circumstances. In Section 5.1, we showed that as traffic approaches capacity the magnitude of flow reaches a ceiling, and the congestion that occurs with increased demand causes the flow to decrease instead. A high flow, low occupancy event occurs when traffic is normally congested (causing less efficient flow). The event, then, occurs when demand is just slightly less than normal, relieving the congestion enough to let more cars pass than normal. So even though fewer vehicles than normal are attempting to access the freeway, more vehicles than normal are gaining access. Possible event causes include an accident on an arterial road that feeds the freeway causing more vehicles to take surface streets and thus reducing demand. Another possible cause is a holiday that is observed by a relatively small percentage of travelers. The (difficult) goal of ramp metering [55] is to manage access

Event Date	Begin Time	End Time	Time Span (hours)	Begin (PM)	End (PM)	Vehicle HRS Saved	Ave Speed (MPH)
6-Apr-07	13:46	18:48	5.0	2.9	10.5	2378	59.3
7-Dec-07	13:42	18:46	5.1	2.9	6.5	2236	62.5
9-Apr-07	14:36	18:54	4.3	1.5	5.8	2189	60.4
24-Dec-07	14:24	18:54	4.5	1.5	6.5	2146	64.8
26-Dec-07	14:38	18:54	4.3	1.5	5.8	2050	60.8
21-May-07	14:30	18:54	4.4	1.5	5.8	2027	59.0
28-Dec-07	14:18	18:48	4.5	2.9	6.5	2016	59.9
12-Nov-07	14:30	18:40	4.2	1.5	6.5	1909	62.2
11-Jun-07	14:30	18:54	4.4	1.5	5.8	1801	55.9
5-Jul-07	14:32	18:54	4.4	1.5	5.8	1781	54.4

Table 5.4: North 57, high flow, low occupancy events ranked according to vehicle hours saved.

Event Date	Begin Time	End Time	Time Span (hours)	Begin (PM)	End (PM)	Vehicle HRS Saved	Ave Speed (MPH)
5-Jul-07	6:04	8:52	2.8	6.6	9.2	421	63.6
12-Nov-07	6:16	8:30	2.2	6.6	9.2	330	65.1
5-Jul-07	6:56	8:58	2.0	1.0	4.3	277	61.6
6-Nov-07	6:36	8:42	2.1	6.6	9.2	266	62.0
12-Apr-07	6:56	9:00	2.1	6.9	9.2	253	58.2
18-Dec-07	6:48	8:58	2.2	1.0	4.3	246	61.7
27-Dec-07	6:28	8:32	2.1	7.6	9.2	237	65.2
3-Aug-07	6:26	8:04	1.6	6.9	9.2	208	66.9
19-Dec-07	7:28	8:58	1.5	0.3	4.3	208	60.2

Table 5.5: South 57, high flow, low occupancy events ranked according to vehicle hours saved.

to the freeway during periods of high demand to maintain efficient flow conditions such as occurs during these high flow, low occupancy events.

Tables 5.4 and 5.5 show the top 10 high flow, low occupancy events ranked according to $vehicle \cdot hours$ saved or reduced delay. This metric is the opposite of the delay calculation defined in Equation 5.6 that was used in the first ranking experiment.

The extra vehicle hours saved is calculated as

$$\text{Reduced Delay} = \sum_{s,t} p_{s,t}(event) L_s \frac{1}{13.64} \left(\frac{\lambda_s^f(t)}{\lambda_s^o(t)} - \frac{f_{s,t}}{o_{s,t}} \right) f_{s,t} \tag{5.7}$$

where $p_{s,t}(event)$ is the probability of the event for sensor s at time t.

Traffic is most congested during weekday mornings in the south lanes and weekday evenings in the north lanes. Since the greatest potential for reduced delay happens when congestion is the greatest, it is not surprising that the top ranked events occur during these same times. The north lanes, which are normally more greatly affected by congestion than the south lanes, also have events with greater reduction in total delay. One can see from the average velocities for vehicles during the event (last column) that traffic is in free flow for the most part, where free flow traffic is considered to travel at 75 MPH [44].

5.3.3 Spatiotemporal Accident Characterization

The previous section showed how the output of the transportation event model can be used to characterize a variety of event types. An important subset of these events are events caused by accidents.

Our collaborators at the Institute of Transportation Studies (ITS) [35] at UC Irvine developed a tool for determining the spatiotemporal outline of an accident and then calculating the extent of the non-recurrent delay caused by the accident [62]. Using the vocabulary common to this thesis, recurrent delay is defined as the delay experienced under normal conditions (i.e. normal patterns of flow and occupancy, represented by $\lambda^f(t)$ and $\lambda^o(t)$ in the event model); and non-recurrent delay is defined as additional delay (in $vehicle \cdot hours$) due to an event (reflected in part by the additional occupancy attributed to the event activity e_t^o in the event model).

In our collaborators previous work [62], the occupancy and flow were combined using a calculation for velocity, then a Gaussian distribution was fit to a one year history of velocity measurements to determine the recurrent or normal activity. A velocity measurement that was much smaller than the empirical mean, then, had a higher probability of being classified as non-recurrent (or event) activity. The corruption of the measurements in the training history due to event activity and sensor failure as discussed in Chapters 3 and 4 caused some sensors in the accident area to provide misinformation to the model. This misinformation was partly responsible for creating (1) "holes" where sensors expected to be affected by the accident reported normal levels of activity and resulted in impermissible configurations of the accident boundary, and (2) false positives where sensors not expected to be affected by the accident are reported as abnormal. The previous work overcame these challenges by using an algorithm to find the "best" border that minimized an objective function, while at the same time meeting constraints defining the allowable shape of the accident border. The objective function penalized the inclusion of holes (measurements with high probability of normal) inside the accident boundary, and also penalized the exclusion of sensors reporting abnormal measurements that were inside the allowable spatiotemporal range for the accident.

Our event models are designed to minimize many of the challenges that the spatiotemporal accident characterization tool had to overcome, and our collaboration led to the joint flow/occupancy model of this chapter, which incorporates both the MMPP's robustness with realistic traffic assumptions.

Incorporating the event model into our collaborators' accident characterization tool was beyond the scope of this thesis. Nonetheless, it is worth discussing the potential advantages of doing so in future work. The following are several potential benefits of incorporating the event model into the accident characterization tool:

1. The event model filters out questionable measurements and sensor failure. Sensor failure is common in the loop detectors in our study (as discussed in Chapter 4), and additional factors such as the effects of construction can corrupt the history of measurements making the estimation of normal activity challenging. This can account for many of the "holes" and false positives found in the earlier work. The event model automatically removes periods of sensor failure and the output of the model can be used to further cull sensors that have no apparent normal pattern and which would otherwise provide misinformation to the accident border detection algorithm. So rather than providing misinformation, these measurements can be removed before running the accident border detection algorithm, making the border easier to detect.

2. A more principled approach for inferring the normal (recurrent) pattern and event activity eliminates some of the pitfalls encountered by methods that use the historical mean to estimate normal traffic patterns. Chapter 3.1.1 illustrated how event activity in the history of measurements can cause gross errors when trying to estimate the effect of an event. Chapter 3.4.2 shows how the event model avoids many of these pitfalls. Use of the event model could result in potentially large gains in accuracy when estimating the non-recurrent delay due to an accident (see Equation 5.4 for the calculation of non-recurrent delay used by the previous method and see Equation 5.6 for the calculation of non-recurrent delay using the output of the transportation event model).

3. The event model operates on the raw, unadulterated data. The earlier work relied on preprocessed data [14] that detects four common types of sensor failure and imputes values for missing measurements and sensor failures. However,

abnormal measurements due to construction, some types of sensor failure, and other event activity remain in the historical data. Depending on the situation, the imputation algorithm uses information from neighboring sensors, historical data from the sensor, or information from a cluster of similar sensors to replace the missing or bad measurement. While having imputed values is helpful for many applications that require a complete grid of clean data, they can muddy some of the analysis relevant to the accident boundary detection problem. For example, if historical data is used to impute a value for a missing measurement in the middle of an accident area, the imputed value would probably appear to be normal, and would provide misinformation (a "hole") to the border detection algorithm. Also, if many imputed values appear in the history of measurements for a sensor (which is not uncommon), the estimated value for normal or recurrent activity could become skewed. Using the raw data also allows us to work with finer time increments, since the imputed data is only available at 5-minute intervals.

4. With many of the holes and false positives eliminated by using the event model, a combination of (1) the simple method for event boundary detection used in the previous section, along with (2) the constraints used in the current tool to define the event shape, could be sufficient for the accident event detection task. Also, the earlier work used only one lane of data creating a 2-dimensional spatiotemporal space to reason over. The use of information from neighboring lanes reduces the impact of missing data and erroneous inferences.

5.4 Summary of Contributions

- Developed an event model for specific use in transportation applications. This model incorporates both the observed flow and occupancy measurements recorded by loop detectors as evidence presented to the model.

- Introduced a method for characterizing the spatiotemporal signature of events using the output of the transportation event model and the locations of mainline sensors embedded in a section of a freeway.

- Demonstrated the utility of the transportation event model by discovering and ranking events of various types. This was accomplished using a nine month history of loop detector measurements from all mainline sensors in a 12 mile section of the 57 Freeway in Orange County. Several metrics that measured the severity of the impact were introduced and used to rank various types of events.

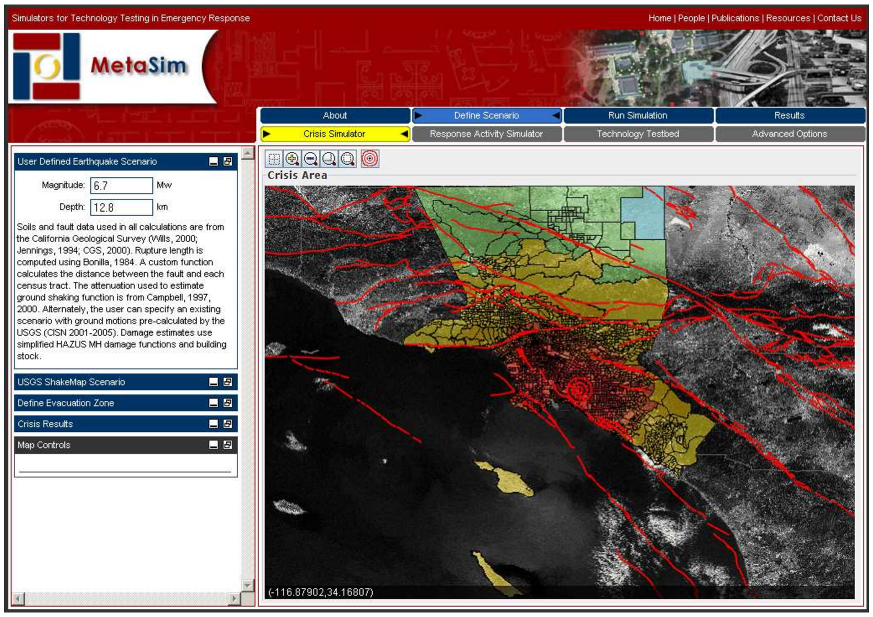
- Discussed the use of the model for the specific application of estimating the non-recurrent delay caused by accidents. The benefits of incorporating the transportation event model in existing tools were illustrated and discussed.

Chapter 6

Linking Building Sensors: A Dynamic Building Occupancy Model

Occupancy is defined in this chapter as the number of people in a building at a given time. In other chapters occupancy refers to the fraction of time a traffic sensor is covered by a vehicle

Knowledge of the number of people in a building at a given time is crucial for applications such as emergency response. Current methods for predicting occupancy [34, 32], however, mainly rely on census information. Figure 6.1 shows a screen-shot of a current system that performs loss estimation calculations in the event of a natural disaster such as an earthquake. The software has sophisticated algorithms for determining structural damage given an earthquake of a certain magnitude, but its estimate of human loss is based on census data. There is an adjustment in the occupancy esti-



Magnitude: 6.7 Mw

Depth: 12.8 km

Soils and fault data used in all calculations are from the California Geological Survey (Wills, 2000; Jennings, 1994; CGS, 2000). Rupture length is computed using Bonilla, 1984. A custom function calculates the distance between the fault and each census tract. The attenuation used to estimate ground shaking function is from Campbell, 1997, 2000. Alternately, the user can specify an existing scenario with ground motions pre-calculated by the USGS (CISN 2001-2005). Damage estimates use simplified HAZUS MH damage functions and building stock.

Figure 6.1: Screen-shot of a current loss-estimation system. Human loss is estimated using census information.

mation if the incident occurs during the day, but it is not affected by time of day (for example, the estimate at 10:00 a.m. is the same as the estimate at 4 p.m.).

Systems that to do not make use of dynamic information can make large errors. For example, non-dynamic systems would most likely predict the occupancy at a sports stadium (which is normally empty) to be zero, and would grossly under-count occupancy if a large sporting event was taking place at the time of an incident. We are not aware of any techniques that use dynamic information, such as is available from the sensors examined in this thesis, to update an occupancy prediction based on current evidence (i.e. potentially in real-time).

This chapter introduces a probabilistic model for predicting the occupancy of a building using people-counting sensors. Even with complete sensor coverage of all doors to a building, occupancy prediction is non-trivial. The single-sensor model described in Chapter 3 is extended to a multi-sensor environment where several such models are coupled together to infer an estimate of how many people are in a building given noisy

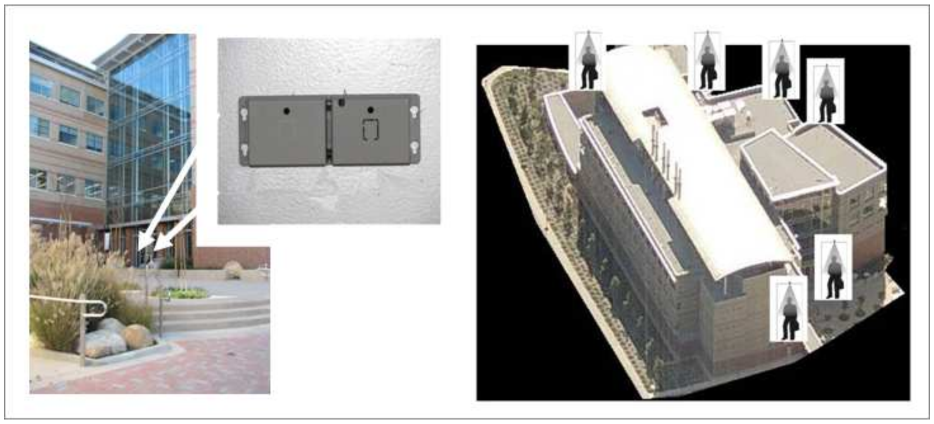


Figure 6.2: The picture on the left shows the front door of the Calit2 building and a picture of an optical people-counting sensor. The picture on the right shows a top view of the Calit2 building and the location of the six doors outfitted with sensors.

count data from its entrances and exits. The probabilistic nature of the model makes it relatively robust to both typical sensor noise and to missing and corrupted data from malfunctioning sensors. The model is experimentally validated by comparing it to a baseline method using real data, the network of optical people-counting sensors that was introduced in Chapter 1. This chapter concludes by showing how the model can be extended to link the sensors further by clustering sensors with correlated event signals.

6.1 Building Sensors

The people-counting sensors that we will use to illustrate the limitations of simple baseline methods in Section 6.2, and later in the case study (Section 6.6), are optical people-counting sensors [45] on the Calit2 building on the UC Irvine campus. The left picture in Figure 6.2 shows one of the sensors that is installed on the front door of the building. These sensors emit a pair of optical beams that, when broken, record a count and a direction depending on the order in which the beams were broken. Each sensor, therefore, records two time-series of counts, one for entrance flow into

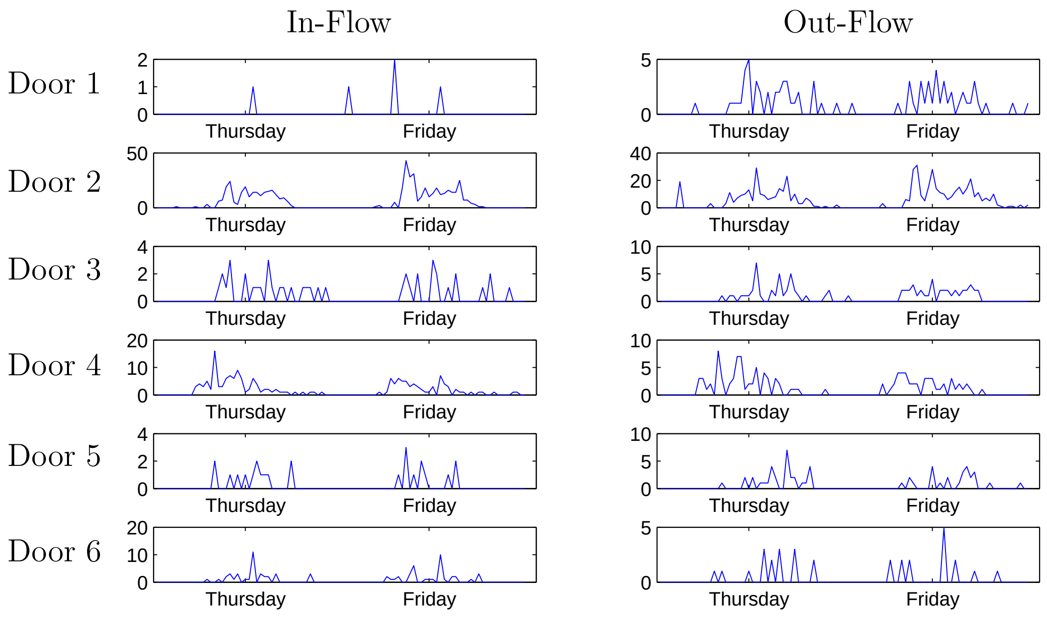


Figure 6.3: Flow of people entering the building (left) and exiting the building (right) at all six doors. The y-axis for each plot is the people-count. Door 2 (the main front door of the building) has the highest flow; and other doors, such as door 1, see very little traffic.

the building and one for exit flow. The counts are aggregated and recorded by a central server every 30-minutes. Counts are sent via wireless connection back to the server, and thus there are multiple possibilities for data to be lost (the battery dies on a sensor, communication link problem, etc.). Figure 3.15 on page 55 shows a plot of the time-series for one door over three weeks.

The picture on the right side of Figure 6.2 shows the location of the six doors in the Calit2 building that were instrumented with sensors. Three additional doors that were not expected to be used were not instrumented. The volume of traffic varies across the six doors as shown in Figure 6.3, with the main front door (Door 2) accounting for approximately half of the entrances and exits. The plots on the left side of Figure 6.3 show the entrance counts at each door, and the plots on the right display the exit counts. Notice that some doors, such as Door 1, have more traffic in one direction than the other.

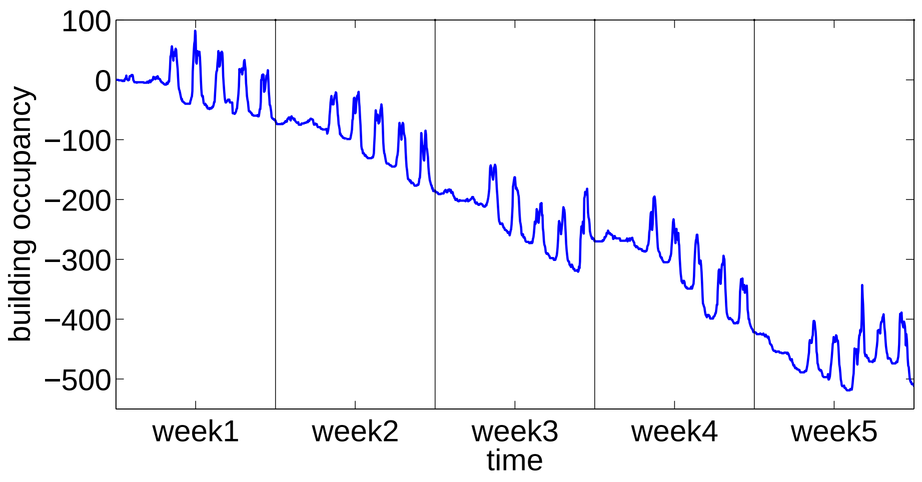


Figure 6.4: An estimate of building occupancy assuming the measured values have no errors, so that occupancy at time t equals that at time $t-1$, plus all incoming counts and minus all outgoing counts between t and $t-1$. Biases and miscounts can cause large systematic errors over a period of time.

Although our study uses optical people counters to measure entrances and exits to the building, the model described in this chapter can be used with other types of people counting sensors with only minor modification. Turnstiles, pressure-sensitive sensors, and video-based people counting systems are examples of other common sensors that can provide the time-series of counts used as input to our model.

While only a fraction of the buildings in a city such as Los Angeles are equipped with people-counting sensors, as the ubiquity of sensors for measuring human activity increases, the type of analysis described in this chapter may become practical not only for a single building but also on a larger scale such as a city.

6.2 A Baseline Method for Inferring Occupancy

Consider a trivial approach to occupancy estimation based on the assumption that we have perfect information from a set of sensors about the number of people entering and exiting at each door in a building, i.e., no noise in the counts and complete

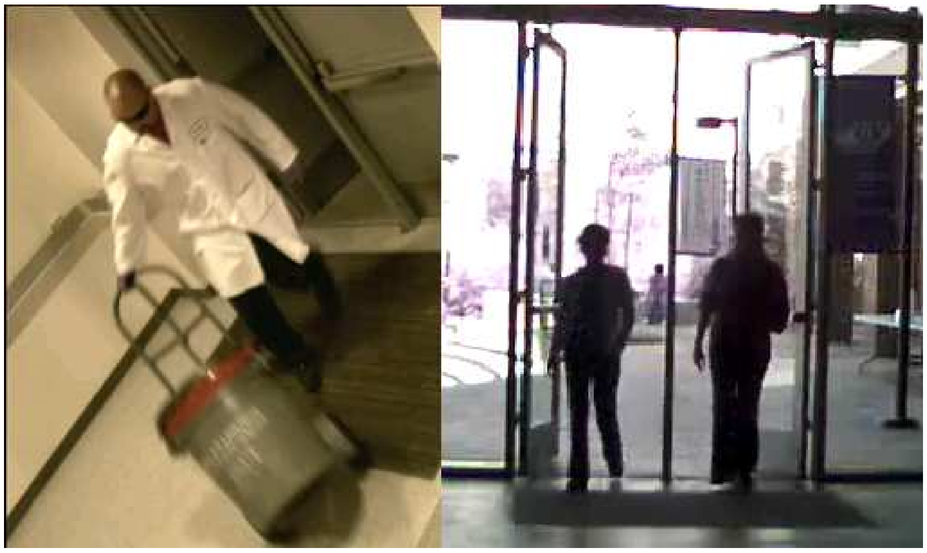
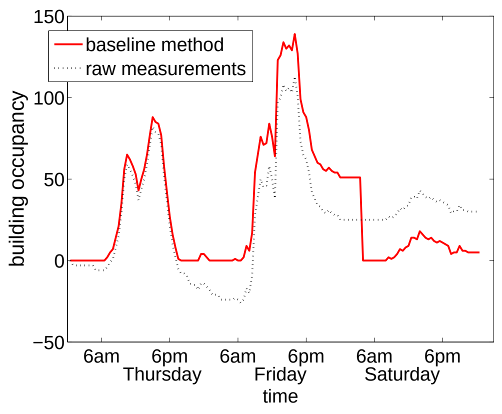


Figure 6.5: The left panel shows an example of a double count at the loading dock entrance of the building; the right panel shows an example of a missed count at the front door.

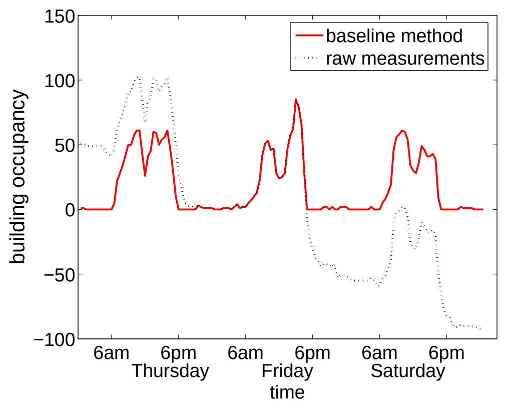
coverage of all doors. The building occupancy at time t is then simply the occupancy at time $t - 1$, plus the sum (across sensors) of the counts of people who have entered since time $t - 1$, minus the sum of counts of people who have exited.

Figure 6.4 shows the result of estimating the Calit2 building occupancy over a period of 5 weeks using this trivial method. We immediately see from Figure 6.4 that the simple approach produces very poor results, with a systematic negative trend (or "drift") in the estimate of the number of people in the building.

This problem of drift arises because the sensors are imperfect, with noise corresponding to both under- and over-counting. The sensors used in Figure 6.4 are pairs of optical sensors that register a count when an optical beam is interrupted. They are spaced in such a way as to determine whether a person is exiting or entering the building. "Non-human objects" can cause over-counts such as the one captured in the left panel of Figure 6.5. More commonly, people entering in groups at the same time can cause under-counting such as is captured in the right panel of Figure 6.5.



(a)



(b)

Figure 6.6: Inaccuracies of the baseline method observed for (a) a day where more entrances are observed than exits and (b) a day where more exits are observed than entrances. The days immediately before and after are also displayed for context.

In addition, a sensor can fail outright. One of the largest discrepancies in Figure 6.4 occurs at the beginning of week 5 and is partly due to a malfunction that occurred in a sensor at a door that is used more frequently for incoming traffic than outgoing traffic. Like many systems, malfunctions for this particular type of sensor result in erroneous values, often zero, rather than any kind of explicit error signal.

One approach to counter the effects of the measurement noise is to simply enforce two constraints on the trivial estimation method used in Figure 6.4:

1. the occupancy can never be negative, and

2. early each morning the building population returns to zero.

Hereafter, this method will be referred to as the “baseline” method.

While incorporation of these types of constraints can improve the estimate quality, the results of this approach (illustrated in Figure 6.6) are still quite inaccurate. The second day (Friday) in Figure 6.6(a) shows the effect of the baseline method’s constraints for a day where more people were counted entering the building than exiting.

The missed exit counts (or extra entrance counts) cause the estimate of the building's occupancy to be artificially inflated during the day. Just before 3 a.m. when the re-zeroing constraint is enforced, the baseline estimate of the occupancy is improbably high, and just after 3 a.m. approximately 50 people suddenly disappear. The second day (Friday) in Figure 6.6(b) shows the baseline method's estimate of occupancy on a day where more exits are measured than entrances, which is the opposite of the situation seen in (a). After 5:30 p.m. the building is estimated to be empty despite the observation of approximately 50 more exits from the building than entrances between 5:30 and the following morning.

In the next section we outline a probabilistic model for the problem of estimating occupancy. This approach allows us to model sensor noise in a systematic manner, combine uncertain information from multiple sensors, leverage our prior beliefs about occupancy at particular times of the day, use statistical learning techniques to learn the parameters of our model from historical data, and systematically infer a probability distribution for occupancy over time conditioned on observed sensor data.

6.3 Building Occupancy Model

In Chapter 3.4.2, we showed that a single traffic sensor that was relatively far from Dodgers stadium in Los Angeles recorded information that could be used to identify when a game was occurring and could be trained to roughly predict the attendance of the Dodger's home baseball games. The prediction task, however, required a training set containing known attendance levels. This type of training data are normally not available, and this section will introduce an unsupervised method for dynamic occupancy estimation.

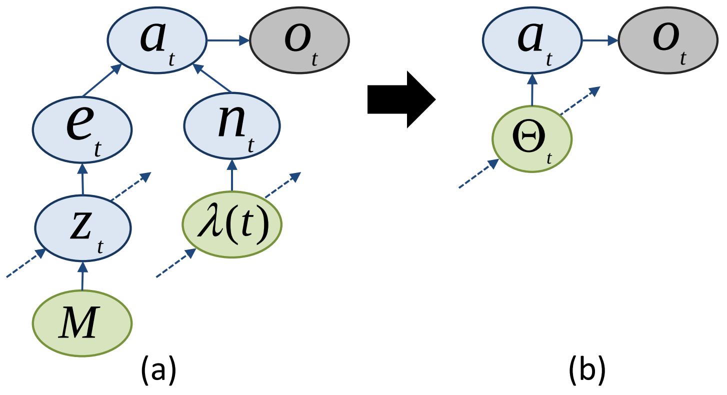


Figure 6.7: Panel (a) shows an extension of the single-sensor model introduced in Chapter 3, to incorporate the notion that the observed count o_t is a noisy observation of the true or "actual" (unobserved) count a_t. The graphical model here shows one time-slice, and the dotted arrows indicate nodes that are connected to other nodes in earlier and later time-segments. For clarity in later figures, the hidden data variables and the parameters of this single-stream model are grouped into one node, Θ, as in panel (b).

The preceding chapters treated individual sensors independently. Information from multiple data streams, however, is required for analysis of the total occupancy of a building. In addition, the information from multiple streams can be used to detect faulty sensor readings, and to provide a more informative way of predicting missing measurements or countering the effects of faulty measurements compared to considering each sensor stream independently.

6.3.1 Extending the Single-Sensor Model to Account for Noise

We first revisit the model for a single sensor (whose graph is shown in Figure 6.7); before extending the model to multi-sensor data. The first departure from the model described in Chapter 3 is made in order to incorporate the concept of noise. Figure 6.5 shows that these sensors are prone to over-counting and under-counting. Our earlier single-stream models did not explicitly address the problem of noise[1], but later in

[1]Note that the Poisson model could in theory implicitly "absorb" some sensor measurement noise in terms of Poisson variation; i.e. it could mix natural "true" human variation in counts, with variation due to noise.

this chapter we will demonstrate that multiple streams can provide information about whether over- or under-counting is taking place. This additional information will be used by the model to estimate the hidden true (or "actual") count at each door, which may deviate from observed count.

Figure 6.7(a) shows the graphical model for one time-slice of the single-sensor model. In the model described in Chapter 3, the event count and the normal count summed to the observed count. Here, the normal and event counts sum to the *unobserved* actual count a_t, and the observed count o_t is a noisy version of the actual count. The ability of the actual count a_t to deviate from the observed count gives the model flexibility, so if the information in the other streams and the model constraints provide evidence that the observed count is too high or too low, a_t can be modified appropriately.

The relationship between the observed count o_t and the actual count a_t at time t for a particular sensor, is defined using binomial distributions to model the noise due to over- and under-counting:

$$o_t = a_t + \Upsilon_t^O - \Upsilon_t^U \tag{6.1}$$

where $o_t \geq 0$ and $a_t \geq 0$. The number of under-counts Υ_t^U and over-counts Υ_t^O are modeled using separate binomial distributions:

$$\Upsilon_t^O \sim \text{Bin}(a_t, v_O) \qquad \text{and} \qquad \Upsilon_t^U \sim \text{Bin}(o_t, v_U) \tag{6.2}$$

subject to the constraints in Equation (6.1) and where v_O and v_U are parameters of the binomial distribution. This allows the the expected number of under-counts and over-counts to increase with the number of people using a door[2]. In our experiments,

[2]We have also experimented with using a double-geometric distribution centered at the observed count o_t to define the distribution of a_t. However, the number of miscounts increases with the number of people being measured, making the binomial distribution a more appropriate choice.

we set $v_O = 1/70$ and $v_U = 1/20$ based on empirical observations of over- and under-counting.

Note that the number of under-counts is modeled as a function of the observed count, where the number of over-counts is a function of the actual count. In the language commonly used when describing the binomial distribution, for $\text{Bin}(n, p)$, p is the probability of success for each of n independent yes/no experiments. In our case, a "success" corresponds to a miscount. In the case of over-counting, each observed count can be seen as an experiment (i.e. $n = o_t$) with probability v_O of being an incorrect count. It may appear more intuitive to use the actual count ($n = a_t$), but consider the unlikely (but possible) situation where the observed count is 5, but all are false counts. In this case the actual count a_t is zero; and if the actual count were used for the number of experiments ($n = a_t = 0$), there would be a zero probability of seeing any number of over-counts.

In the case of under-counting, however, we consider each actual count as an experiment ($n = a_t$) with a probability v_U of being a missed count. To help understand why a_t is used in this case, consider the unusual (but possible) situation where 5 people enter but all are mis-counted (i.e. $a_t = 5$ and $o_t = 0$). If the observed count were used for the number of experiments ($n = o_t = 0$), there would be a zero probability of seeing any number of mis-counts.

6.3.2 Linking Multiple Sensors

At each time segment t, the individual-sensor models for the entrance and exit streams at all doors are connected via a building occupancy node b_t that represents the estimated number of the people who are in the building at time t. Three time-segments of the building occupancy graphical model are shown in Figure 6.8.

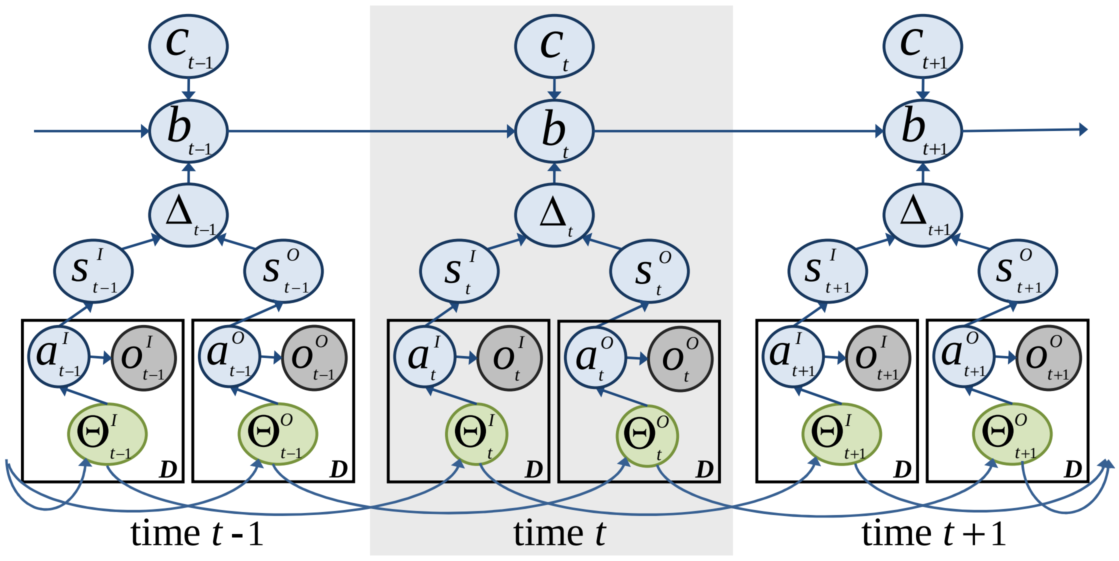


Figure 6.8: Three time-segments of the building occupancy graphical model. The sensors are grouped according to direction and the constraint node c_t encourages the building occupancy (b_t) towards zero once every 24 hours.

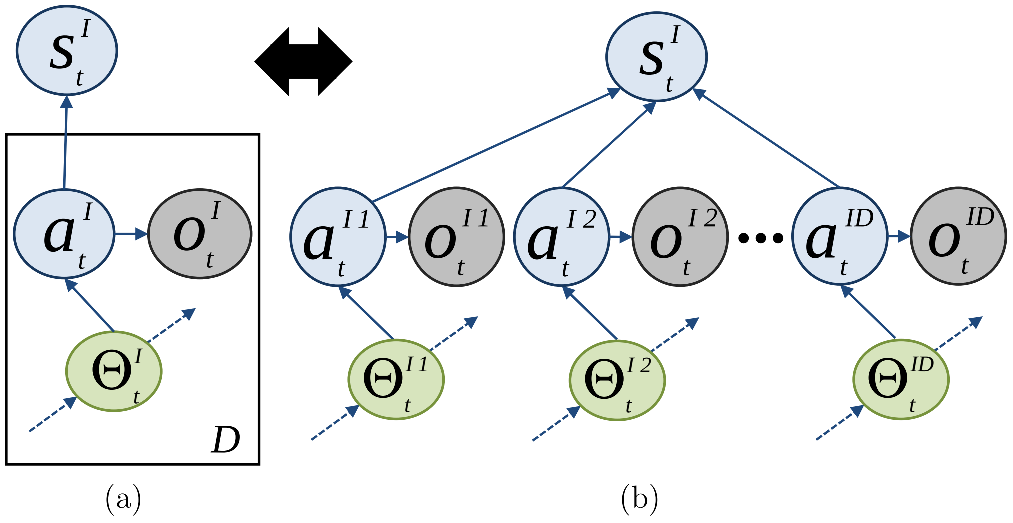


Figure 6.9: Panel (b) shows the graphical model for the "Sum In" node that groups the building entrance streams. Panel (a) shows the same model using plate notation.

Each door to a building has separate data streams for the entrance ("in") counts and the exit ("out") counts. We group the individual count streams as illustrated in Figure 6.9, by aggregating streams of the same direction into a total building entrance (or "sum-in") node and a total building exit (or "sum-out") node.

Figure 6.9(a) shows the graphical model of the sum node using plate notation. Plate notation allows us to represent repeated elements of a graph concisely so that large graphs with many repeated elements are more clear. The plate in the figure (the rectangular box) contains the sub-graph that is repeated, and the number in the bottom right hand corner of the plate indicates the number of times the sub-graph is replicated. In this case, the individual-sensor models for the entrance flow of each of the D doors ($D = 6$ for our data set) are represented by the plate as illustrated in Figure 6.9(b).

The true count for all of the "in" sensors at time t is represented by S_t^I, and similarly S_t^O for the "out" sensors. S_t^I and S_t^O are deterministic sums of the actual counts a_t for in and out sensors, respectively:

$$S_t^I = \sum_{d=1}^{D} a_t^{Id} \qquad \text{and} \qquad S_t^O = \sum_{d=1}^{D} a_t^{Od} \tag{6.3}$$

where a_t^{Id} and a_t^{Od} are the true number of people across time-segment t who entered and exited door d respectively.

The number of people in the building (the occupancy) at time t is denoted b_t, and is given by the sum

$$b_t = b_{t-1} + \Delta_t \tag{6.4}$$

where $\Delta_t = S_t^I - S_t^O$, which is the true (unobserved) change in occupancy over time-period t.

The constraint node c_t acts like a prior over the building occupancy b_t, encouraging the model to leave few or no people in the building overnight according to a geometric distribution (see Figure 6.10). This helps to offset any systematic bias in the mea-

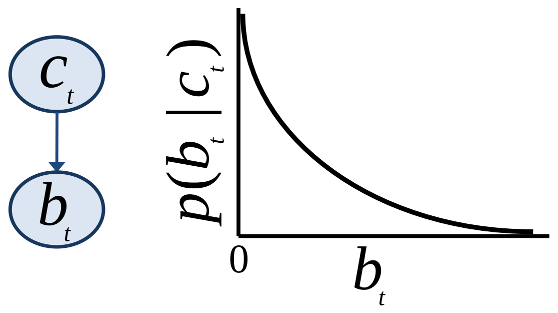


Figure 6.10: A geometric prior (with parameter set to .9 in the results in this Chapter) on b_t , encourages the model to leave few or no people in the building overnight. The constraint is applied once every 24 hours at $t = 3$ a.m.

surement noise which if unaccounted for could lead to ever-increasing or decreasing estimates of the number of people in the building (see Figure 6.4). The constraint is applied once every 24 hours at $t = 3$ a.m..

The sum nodes, delta node and occupancy node are all formed using rigid deterministic relationships. The flexibility in the model is found in the relationship between the observed count nodes and the actual count nodes as defined in Equation 6.1 and illustrated in Figure 6.7. So when the constraints of the model are in conflict with the observed data (for example, when more people are counted entering the building during the day than exiting it), this conflict must be explained by the noise at the individual stream level.

The model pictured in Figure 6.8 assumes that miscounts are only caused by noisy observations. In reality, however, the reported count o_t can also deviate from the actual count due to sensor failure (as demonstrated in Chapter 4). The building occupancy model described in this chapter can be made more robust with only minor modifications to the implementation, using the single-sensor model extension described in Chapter 4 which models sensor failure. It is worth noting, however, that even without an explicit model for sensor failure, the building occupancy model is robust to some forms of sensor failure. As will be demonstrated in Section 6.6, if failure is limited to one stream, the model can often correct for it due to two factors:

1. There is information in the remaining streams that indicates the volume of traffic expected.
2. The corrupt signal will be generally much different than the historical pattern learned for that sensor $\lambda(t)$.

A corrupt signal will usually result in an inconsistency between the total entrance and exit counts to the building for the day. If only one stream is showing a major deviation from its historical pattern, the model will encourage the count inconsistency to be explained by correcting the corrupt stream. This phenomenon will be illustrated later in the results section.

6.4 Inference

Given the probabilistic model described in the previous section, we now turn our attention to the inference problem, i.e., computing the conditional probability of quantities of interest (such as the building occupancy b_t as a function of time) given both the observed measurements o_t (at all doors across all times of interest) and the priors. The variables and parameters of the model are learned by inferring their posterior distributions using Gibbs sampling, in a manner similar to the sampling process used for the individual-sensor model in Section 3.3.

In Gibbs sampling, we iteratively sample each set of variables given the current sampled values of the other variables in the model. After a sufficient number of iterations, these samples converge to the true posterior distribution. Due to the deterministic relationships among several of the variables, some variables *must* be sampled in blocks in order to explore the full posterior distribution. For example, suppose in one sampling iteration, $b_1 = 50$, $b_2 = 55$ and $\Delta_2 = 5$. In the following iteration, if we sample each of these quantities in sequence, we will be stuck repeating the same samples. If we sample Δ_2 given $b_1 = 50$ and $b_2 = 55$ we can sample no value other than 5, and so forth. When variables are sampled together in a block, they are sampled from the joint posterior distribution of the variables in the block.

6.4.1 Sampling the Individual-Sensor Parameters and Variables

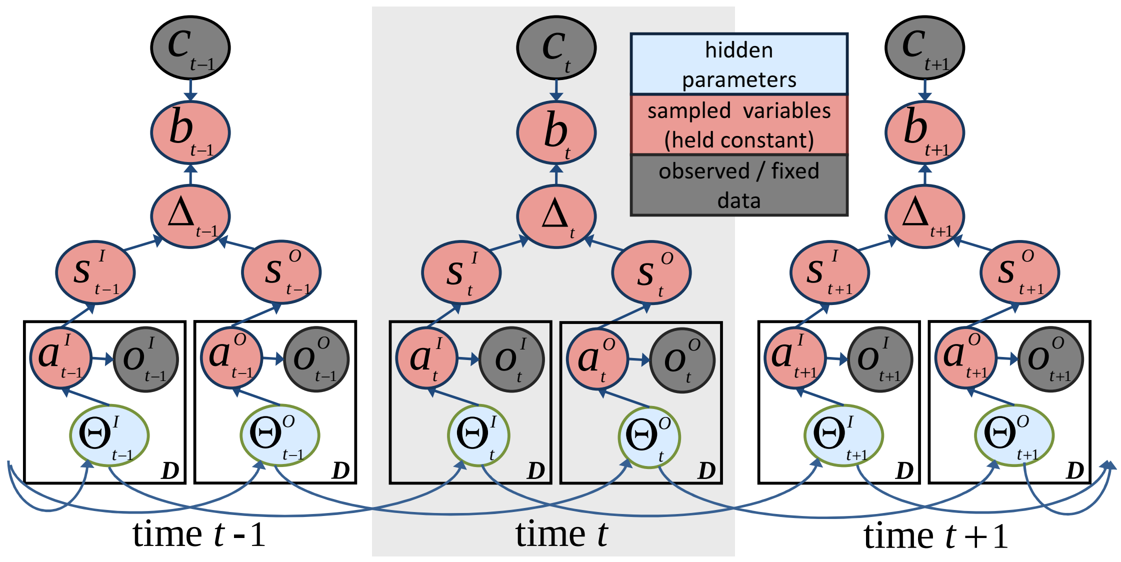
Figure 6.11 shows the Gibbs sampling steps used for inference and learning in the building occupancy model. The sampling process for the step illustrated by Figure 6.11(a) proceeds as described in Chapter 3.3, except that the sampled actual count a_t takes the place of the observed value o_t in Figures 3.13 and 3.14. Unlike the process in Chapter 3.3, however, here the actual count value a_t can change between iterations as the constraints of the occupancy model are enforced and as the belief about the actual counts of the other sensor streams change.

The hidden variables and parameters of the individual-sensor models (jointly represented by Θ in Figure 6.11) are conditionally independent of the parameters and variables specific to the building occupancy model given the actual count a. The elements of Θ for each sensor can therefore be sampled independently. Specifically, for each stream we sample $\lambda(t)$ given the previous samples of n_t, and then sample z_t, n_t, and e_t using the forward–backward algorithm as described in Chapter 3.3.1.

6.4.2 Sampling the Building Occupancy Variables

In the next sampling step (illustrated by Figure 6.11(b)), $\lambda(t)$ and z_t (represented by Θ_t) for all individual-streams are fixed at their previous sampled values, and we

(a)



(b)

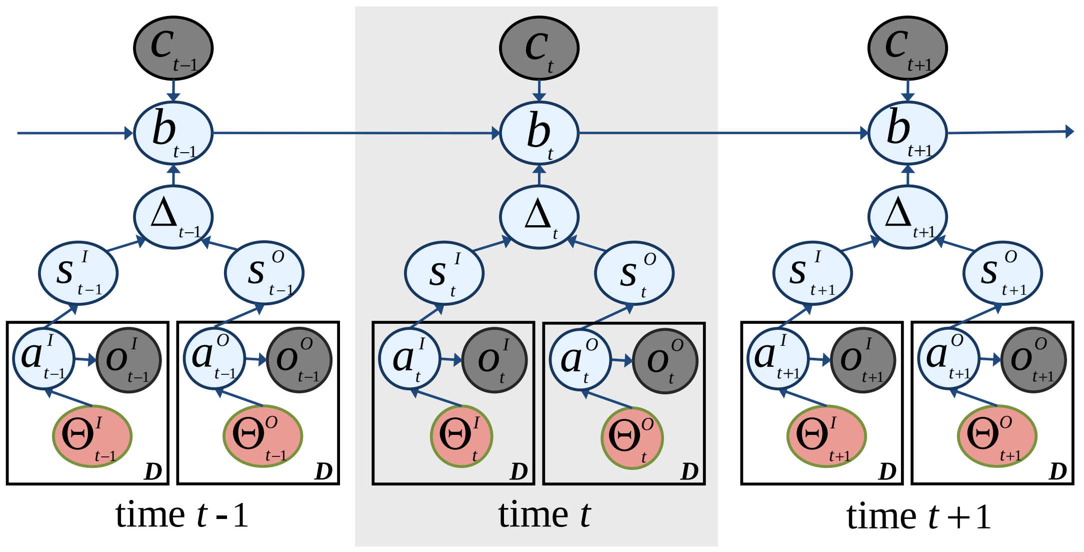


Figure 6.11: Gibbs sampling steps for the building occupancy model. In (a), the individual-stream parameters and hidden variables (Θ) are drawn given samples of the actual counts a_t and the building variables. In (b), the building parameters and actual counts a_t are drawn given fixed values for Θ. In each case, the variables fixed at their previously sampled values decouples the model in time which allows for efficient inference. The dark grey nodes are observed or constant values, the red (medium grey) nodes are variables or parameters that are held fixed to their previously sampled values, and the light blue (light grey) nodes represent variables or parameters that are being sampled in the current step.

perform a forward–backward sampling procedure similar to that used for z_t, to draw the total building occupancy b_t and the true counts a_t for each sensor.

In the forward inference pass, information flows from the individual streams up to the occupancy node and is combined with the belief about the occupancy found for the previous time slice. The backward pass then samples values for each of these variables. Since the graphical model is singly–connected given Θ of the individual streams, this procedure can be performed efficiently (in time linear in the number of measurements).

To compute the posterior distribution of the building occupancy b_t, given the individual-stream parameters Θ and the observed counts o_t, we first note that

$$p(a_t|\Theta_t, o_t) \propto p(o_t|a_t)p(a_t|\Theta_t) \tag{6.5}$$

by applying Bayes' rule and noting that o_t is conditionally independent of Θ_t given a_t. We can then compute the distribution of the variables S_t^I, S_t^O and Δ_t via successive convolution as described in Section 6.5.2. If we define the evidence E_t to be the set of all observations o_t at any of the sensors, this convolution process gives us the distribution $p(\Delta_t|E_t, \Theta_t)$.

The updated posterior of the building occupancy at time t is then

$$p(b_t|E_{1:t}, \Theta_{1:t}) \propto \sum_{b_{t-1},\Delta_t} \delta(b_t - b_{t-1} - \Delta_t)\pi_t(b_t)p(b_{t-1}|E_{1:t-1}, \Theta_{1:t-1})p(\Delta_t|E_t, \Theta_t) \tag{6.6}$$

where $\delta(k) = 1$ for $k = 0$ and 0 otherwise (reflecting the deterministic relationship between b_t, b_{t-1}, and Δ_t), and where $\pi_t(b_t) = \text{Geom}(b_t; .9)$ when $t = 3$ a.m..

We proceed forward in time to the maximum (or current) time $t = T$, then sample $b_T, b_{T-1}, \ldots$ backward to time $t = 1$.

Given b_t and b_{t-1}, Δ_t is deterministic. The sum nodes S_t^I and S_t^O are sampled conditioned on their difference Δ_t, and the true counts a_t are finally sampled conditioned on their total S_t^I or S_t^O as discussed in Section 6.5.2.

6.5 Implementation Issues

This section discusses implementation issues such as parameter settings, working with variables which take on countably infinitely many possible values, and complexity reduction.

6.5.1 Countably Infinitely Valued Variables

The variables n_t, s_t^I, s_t^O, Δ_t, and a_t can take on a countably infinite number of possible values (the set of nonnegative integers). Since no closed form exists for the conditional distributions of interest, we use heuristics to reduce the range of values under consideration.

In our experiments, the range of values were chosen based on empirical information. For example, the maximum count of people entering or exiting the building over one time-segment in our history of measurements was around 100. So s_t^I and s_t^O were allowed to range between 0 and 200, where the 200 value, $2 *$ (max observed count) was chosen arbitrarily.

Buildings with much larger capacities would require a more efficient approach. For these cases the range of values for the count variables can be truncated based on their posterior distributions (Equations 6.5 and 6.6 for example). If we define a general

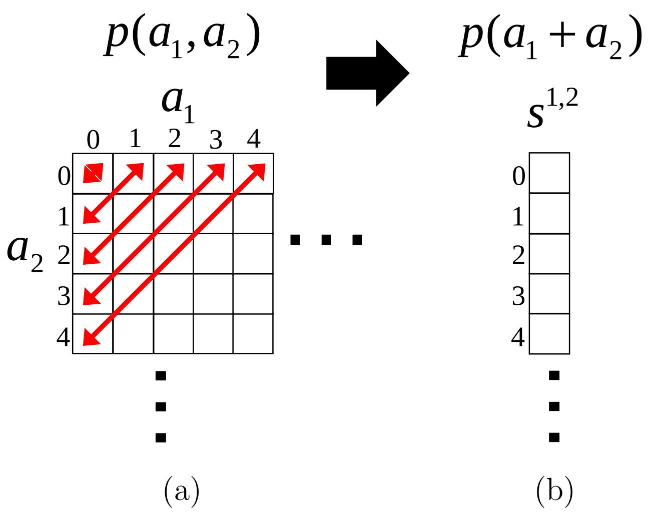


Figure 6.12: Illustration of the convolution operation. In (a), the diagonals marked with red arrows in the joint probability table indicate combinations of a_t^1 and a_t^2 that sum to the same value. The result of the convolution operation (summing along the diagonals), (b), is a one dimensional vector that represents the posterior distribution of the sum of a_t^1 and a_t^2.

count variable C_t (which can represent n_t, s_t^I, s_t^O, Δ_t, or a_t) and set a threshold, ϵ, then we can set the limits of the range when $p(C_t|E_{1:t}, \Theta_{1:t}) < \epsilon$.

6.5.2 Successive Convolution

Computing the posterior distributions of s_t^I and s_t^O requires an efficient algorithm. To calculate $p(s_t^I = n|E_t, \Theta_t)$ requires summing over all possible combinations of actual counts a_t^d for the entrance streams at all d doors which sum to n.

$$p(s_t^I = n|E_t, \Theta_t) = \sum_{a_t^1=0}^{N} \sum_{a_t^2=0}^{N} \cdots \sum_{a_t^D=0}^{N} \prod_{d=1}^{D} p(a_t^d|E_t, \Theta_t)\delta(\sum_{d=1}^{D} a_t^d) \tag{6.7}$$

where $\delta(k = n) = 1$ and $\delta(k \neq n) = 0$, a_t^d is the actual count for the entrance stream at door d, and the values of a_t^d range from 0 to N. Note that the posterior for the sum of actual counts is a simple product of the posterior probabilities of the actual counts at each door, due to the model's assumption that the actual counts are independent.

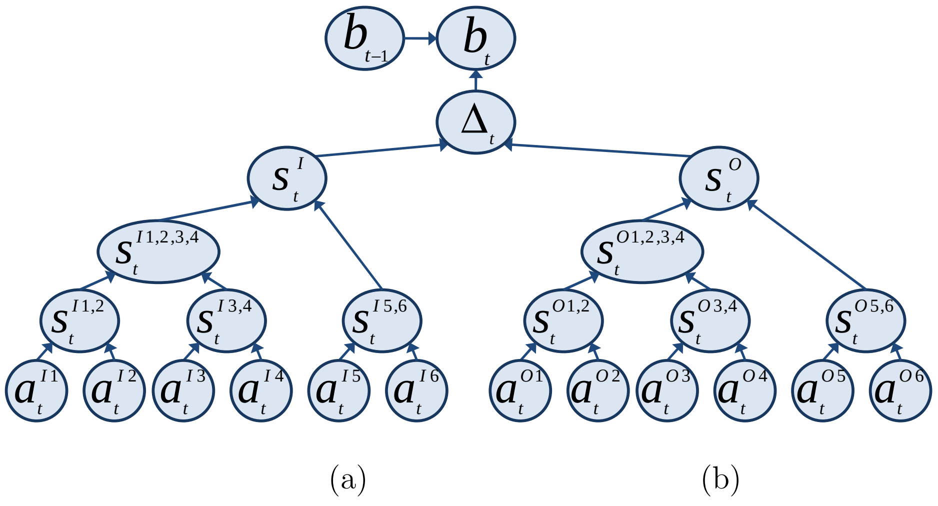


Figure 6.13: Illustration of the successive convolution steps for a building with 6 doors. The convolution operation shown in Figure 6.12 is performed on each pair of sibling nodes in this figure. In the convolution operation of the two sum nodes to find the posterior distribution of the Δ_t variable; however, the diagonals go the opposite direction since the difference is taken rather than the sum.

The computation in Equation 6.5.2 has complexity $O(N^D)$; however, we can use a successive convolution technique to reduce the complexity to $O(DN^2)$. Figure 6.12 illustrates the convolution operation. Figure 6.12(a) shows the joint posterior distribution table for the actual counts at doors 1 and 2, $p(a_t^1, a_t^2|E_t, \Theta_t)$. The diagonals marked in the table indicate combinations of a_t^1 and a_t^2 that sum to the same value. For example, the third diagonal from the corner indicates the sum $a_t^1 + a_t^2 = 2$ for the (a_t^1, a_t^2) pairs (0,2), (1,1) and (2,0). Summing each marked diagonal produces a 1-dimensional vector representing the posterior distribution of the sum of a_t^1 and a_t^2. The posterior distribution of this super-stream of actual counts from doors 1 and 2, $s_t^{I1,2}$, can then be combined with other super-streams using the same operation. The convolution operation can be repeated for any number of streams and super-streams, requiring only a 2-dimensional table to be created each time.

Figure 6.13 shows the hierarchy created when performing successive convolution operations for a building with 6 doors. The backward sampling step discussed in Section 6.4.2 can be performed using the tables created in the convolution operations in the following manner:

1. Sample b_T from the vector created by the convolution of b_{T-1} and Δ_T.
2. The $\hat{b}_T$ sample corresponds to a diagonal in the joint distribution table of b_{T-1} and Δ_T. Sampling along this diagonal produces a ($\hat{b}_{T-1}$, $\hat{\Delta}_T$) sampled pair.
3. The $\hat{\Delta}_T$ sample corresponds to a diagonal in the joint distribution table of s_t^I and s_t^O. Sampling along this diagonal produces a ($\hat{s}_t^I$, $\hat{s}_t^O$) sampled pair.
4. The super-stream and individual stream counts are then sampled in the same way, each time using the 2-dimensional joint probability tables created in the original convolution operations.

6.5.3 Parameter Settings

The constraint that prevents the type of drift seen in Figure 6.4 uses a parameter setting of 0.9 for the geometric prior and was enforced once every 24 hours at 3 a.m.. The building in our study emptied at night, so a prior that encouraged the occupancy towards zero in the middle of the night was appropriate. Other buildings that do not empty or are less predictable at night would require a different method. One possible approach would be to enforce a self-similarity constraint on the daily profile shape of the building occupancy. For example, if there is a large regularly scheduled event in a building that has a consistent attendance which dominates the building occupancy at that time-segment; then there must be a penalty for an occupancy estimation that deviates from that predictable attendance level. This self-similarity constraint could also improve imputation of missing data and could potentially help offset the effects of corrupted signals as well.

The parameters for the binomial distributions that model the observation noise (Equation 6.3.1) were set based on empirical observation. The results of inference did not seem to be sensitive to these settings based on limited testing.

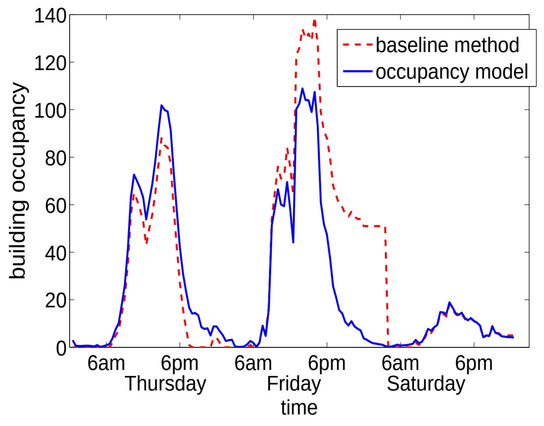
6.6 Case Study 4: Building Population Study

The data used in this case study come from a campus building with six doors with optical people counters measuring the flow of people in both the entrance and exit directions as described in Section 6.1. Nine weeks of measurements (6/11 to 8/12/2006) for each of the twelve streams were used for learning the model. All of the inference experiments in this section were run off-line, although on-line inference is a relatively straightforward extension of the techniques described in this chapter.

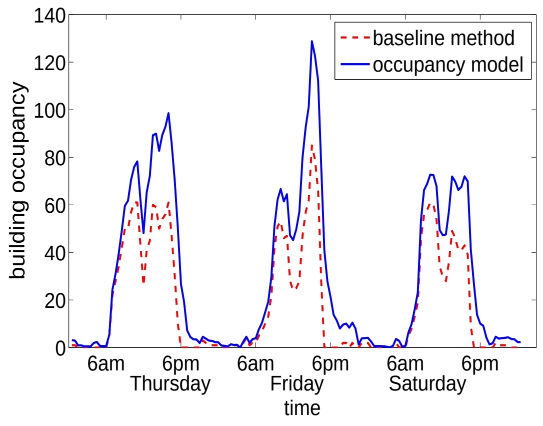
In each of the experiments, the building occupancy model is compared to the simple baseline method described in Section 6.2, where two occupancy constraints are enforced: (a) occupancy can not be negative, so if $(O_{t-1} + \Delta_t) < 0$ then $O_t = 0$; and (b) occupancy is reset every 24 hours, so that at $t = 3$ a.m. we have $O_t = 0$.

6.6.1 Sensor Noise

The examples in this section contrast the occupancy model and the baseline method for days where the measured flow of people entering or exiting the building is disproportionately larger than the flow in the opposite direction. This count difference is



(a)



(b)

Figure 6.14: (a) A day with more building entrance measurements than building exit measurements; the preceding and subsequent days are also shown for context. (b) Two days with more building exit measurements than building entrance measurements.

caused by the normal day-to-day noise of the sensor measurements as illustrated in Figure 6.5.

The data plotted in Figure 6.14 is the same time-period shown in Figure 6.6 (used to illustrate a weakness of the baseline method). Figure 6.14 (a) shows a day where using the baseline method would lead to the belief that approximately 50 people are in the building at 2:59 am. These 50 people are promptly forced to disappear due to the 24 hour constraint. By smoothing, the occupancy model provides a more believable prediction for the day (solid line in Figure 6.14(a)). Although we do not have ground truth, it is especially unlikely that the building held many people this particular Friday night since the two following days are weekend days with low activity and no large egresses (refer to the last day of Week 3 and the first 2 weekend days of Week 4 in Figure 6.4).

Figure 6.14(b) shows days with the opposite situation where more people are measured leaving the building than entering. The baseline model ignores all the extra exit counts at the end of the day. On the second day, the building is predicted to be empty after 5:30PM. Approximately 50 extra exits without corresponding entrance counts after

5:30 are ignored by the baseline model, resulting in occupancy predictions that are likely too low. The probabilistic model, however, uses this information to adjust the occupancy at previous times upward, resulting in a more believable prediction.

Although we do not have a ground truth for comparison, these examples illustrate that the probabilistic model appears to provide more reasonable outputs than the baseline for typical amounts of sensor noise.

6.6.2 Missing Data

In this section, we investigate robustness to missing measurements. A missing measurement is defined by a time-segment where no measurement is reported. This can occur due to a power failure, a communication error, or a faulty measurement that is flagged as a result of an internal diagnostic of a sensor.

Since we do not have the true building occupancy values, we address the issue of validating the model by removing some information and seeing how well the model recovers. For the example in this section, we remove information by replacing observed measurements with missing labels, such as happens when a sensor stops communicating. We present the modified data as input to the model, and will test the model's ability to recover the original information.

In the experiment, shown in Figure 6.15, one day of measurements for the entrance stream of the main door to the building was replaced by missing labels. This particular door is used by approximately 50% of people who enter the building. Figure 6.15(a) shows the raw building occupancy calculation (occupancy at time t equals that at time $t-1$, plus all entering count observations and minus all exiting count observations between t and $t-1$) before and after the replacement. The red dashed line in

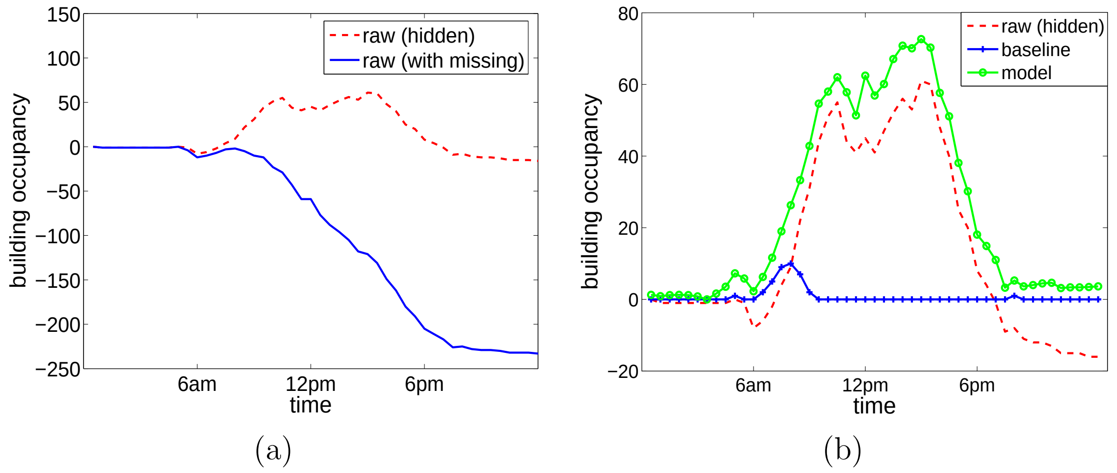


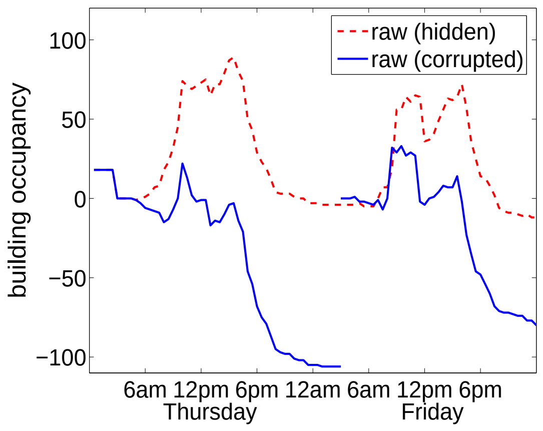
Figure 6.15: (a) The raw occupancy calculation before (indicated by the red dashed line which will be hidden to the competing methods) and after (the solid blue line) the measurements for one building entrance stream accounting for approximately 50% of the total entrance counts were replaced by missing data labels. (b) The baseline method has no means for replacing missing data, but the building occupancy model makes a reasonable imputation.

Figure 6.15(a) shows the original data that the model will try to recover; these data were hidden from the baseline and building occupancy models. The solid blue line in Figure 6.15(a) shows the data given as input to each model where some of the original data are replaced by missing measurements.

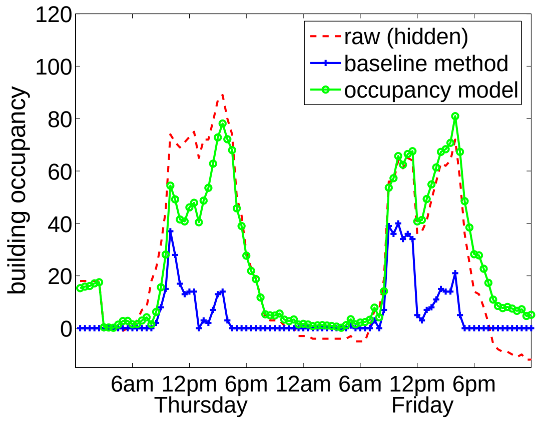
The baseline method (shown by the solid blue line with "+" symbols in Figure 6.15(b)) has no way to deal with missing data and degrades quickly. The occupancy model (shown by the solid green line with "o" symbols in Figure 6.15(b)) is able to recover much of the missing information, using the model of typical behavior (Poisson rate) and the information from the other streams such as extra exit counts.

6.6.3 Corrupted Data

In the case of missing data such as examined in Section 6.6.2, the model is alerted of the need to fill the missing data with something reasonable, i.e. the model knows that



(a)



(b)

Figure 6.16: (a) The raw building occupancy before (which will be hidden to the competing methods) and after the measurements from one entrance stream accounting for approximately 25% of the total entrance counts were replaced by zeros. (b) Occupancy estimation given the corrupted data using the baseline method (blue line with "+" symbols) and the building occupancy model (green line with "o" symbols).

measurements are missing. Corrupted measurements create a more difficult problem because they are presented to the model as valid data.

A corrupted measurement is defined by a time-segment where a "faulty" measurement is reported. These errors occur when a sensor malfunctions but continues to send false information; this happens when a sensor is blocked or damaged, or when a software error occurs.

To test the model's robustness to corrupted data, we replace the measurements of one of the entrance streams with zeros, at a door used by roughly 25% of the building occupants. Figure 6.16(a) shows the raw building occupancy calculation for the pre- and post-modified data (just as in Figure 6.15(a)).

The observed measurements (pre-corruption) for all twelve streams is seen in Figure 6.3, for the two days tested in the corrupted data experiment. In our test, corruption was introduced by replacing all of the observations for the stream displayed in the fourth row of the first column of Figure 6.3 (Door 4, entrance stream) with zeros for the entire two day period. This “stuck at zero” failure that we are imitating is a common failure type such as can be experienced by a blocked sensor.

The results for corrupted data are shown in Figure 6.16. As with the missing data experiment, the baseline method (blue line with “+” symbols) fails completely. The occupancy model (green line with “o” symbols) performs much better, although it does not recover all of the missing information. Two things help the multi-sensor model model recover the corrupted information. First, the corrupted data appears unusual at the individual stream level, as the model expects data similar to the rate parameter. Second, if the corrupted data are only in one direction, the “excess” counts from the other direction will try to balance it out.

The corruption introduced for the second day of the experiment was minor enough for the model to produce a plausible estimation of the building occupancy. However, the corruption introduced in the first day was more severe (approximately twice the magnitude of the second day), and the model did not recover all of the hidden information. The noise model resists deviation from the observed values, but the shared information is able to offset at least some of the effects of the corrupted data. This property of the occupancy model could also be used to detect a faulty sensor and provide an early alert prediction of a malfunction.

If a sensor is believed to be faulty due to factors such as extremely unlikely reported measurements while other streams show normal activity, the measurements predicted to be false could be replaced by missing data labels. A model with an explicit notion of sensor faults, such as described in Chapter 4 could improve performance still further.

6.7 Summary of Contributions

- Introduced a probabilistic building occupancy model that provides a dynamic estimation of the number of people in a building based on measurements from people counters at all major entrances to the building.
- Demonstrated how individual sensor models could be linked to form a multi-sensor model, which in turn can be used for higher level inferences.
- Introduced a model of sensor noise, and showed how linking sensors through the constrained occupancy variable can be used to make inferences about the "true" count at each sensor which differs from the observed count.
- Presented a case study that demonstrated the advantages of the building occupancy model over baseline methods, in particular with periods of missing data and sensor failure.
- Large scale dynamic occupancy estimation is one of our long range objectives for the event model, and we see the dynamic building occupancy model described in this chapter as a step towards that goal.

Chapter 7

Clustering Sensors with Correlated Event Signals

One weakness of the multi-sensor building occupancy model described in Chapter 6 is that it is not sensitive to event predictions that have evidence from multiple streams. Consider the following situation, *Situation A*:

- There were 50 more entrance counts to the building than exit counts during the day.
- Between 2 and 3 p.m., there was an event predicted for the entrance stream to door 1.
- The estimated additional entrances due to the event at door 1 was 20.
- No other events were inferred for the day.

Because of the 50 extra entrance counts, the model will infer over-counts from entrance streams and under-counts from exit streams over the course of the day in order to return the building occupancy towards zero at 3 a.m.. A cost is incurred for every miscount according to Equation 6.3.1. Also, in order to encourage the prediction of events to be relatively rare, a penalty is also paid in the individual model in order to infer an event (enforced by the priors on event-transitions in the individual-stream models as discussed in Chapter 3.3.3). If the event signal is relatively small for the event predicted in Situation A, the building model will be encouraged to take enough miscounts from the stream with the event in order to get a reward for removing the event. For Situation A, this course of action (removing the event) may be reasonable because it may be unusual to observe an abnormal amount of traffic in one stream that is not supported by event activity at other streams.

However, consider a different situation, *Situation B*:

- There were 50 more entrance counts to the building than exit counts during the day, as in Situation A.
- Between 2 and 3 p.m., there was an event predicted for the entrance stream to door 1, with an estimated 20 additional entrances due to the event, as in Situation A.
- Between 2 and 3 p.m., there was also an event predicted for a different entrance stream, door 3, with an estimated 10 additional entrances due to the event.
- At 3:30 p.m., there was an event predicted for the exit stream at door 1 with an estimated 30 additional exits due to the event.

The presence of the additional entrance event and the exit event in Situation B gives evidence to support the prediction of the entrance event for door 1. However, in the current model, there is no difference between how Situation A and B would be treated, and the model would still be encouraged to remove one of the entrance events to help make up for the extra entrance counts during the day.

In the building occupancy model described in Chapter 6, the individual sensor streams are linked only through the occupancy node. Another potentially helpful way to link the individual-stream models is a connection between the event nodes, z_t (refer to Figure 6.7 on page 139). In the current model, the presence or absence of an event at one door has little influence on the prediction of events at other doors at the same time. Linking the individual models at the event node would allow us to encode information such as "when the entrance stream at door 3 has an event, the entrance stream at door 5 often has an entrance event as well." This additional information could result in more sensitive event detection when the event traffic is spread out over several doors, and could also aid in detecting and correcting for missing and corrupt data such as examined in Sections 6.6.2 and 6.6.3.

In this chapter, we will explore a method that can be used to determine which streams to link via their event nodes. The method uses an EM algorithm to group streams that have correlated event activity.

7.1 Creating the Event Data Matrix

The twelve data streams captured at the Calit2 building (described in Section 6.1) contain intermittent periods of event activity. For example, in one time-segment, the entrance stream at door 1 and the entrance stream at door 3 might show event activity as event attendees enter through both doors; whereas in the next time-segment perhaps only the exit stream at door 1 shows event activity as event attendees primarily leave the building through that door.

For each event stream at each time slice, there can either be no event, a positive event, or a negative event. For simplicity, in this section we are going to consider relationships among the 12 streams for the positive event case separately from the negative case. That is, we assume that positive events happen together and negative events happen together, but that there are no prominent patterns where one stream shows negative event activity while another stream shows positive event activity. Later in Section 7.7.2, we will show how to extend the algorithm to also include correlations between negative and positive events. In this section we will also only consider correlations that exist in the same time-segment. This is reasonable for our data set because of the lengthy 30-minute time segments, but the method will be extended to consider correlations between streams at different time segments in Section 7.7.1.

The training data that are used by our clustering algorithm were created in the following manner:

1. The event model described in Chapter 4.1 was run on each of the 12 streams independently (i.e. not using the current occupancy model which links the streams).

2. A binary value, x, indicating the presence or absence of event activity is created for each time-segment, and is set according to

$$x = \begin{cases} 1 \text{ if } p(z_t|o_{1:T}) > 0.1 \\ 0 \text{ otherwise} \end{cases} \tag{7.1}$$

   In previous sections, we defined event activity as $p(z_t|o_{1:T}) > 0.5$, but here we are interested in grouping streams that appear to show any event signal, in

Table 7.1: Example of event indicator data **X** and the hidden cluster membership indicators **Z**, for a building with two doors (four sensor streams) with event data generated by two clusters. The values for **Z** are italicized to indicate that they are not observed.

time-segment	Event Data **X**				**Z**	
1/1/07 12 to 12:30pm	0	0	1	1	*0*	*1*
1/1/07 12:30 to 1pm	0	0	0	1	*0*	*1*
1/1/07 1 to 1:30pm	1	0	0	0	*1*	*0*
1/1/07 1:30 to 2pm	0	0	1	1	*0*	*1*
1/1/07 2 to 2:30pm	0	0	1	1	*0*	*1*
1/1/07 2:30 to 3pm	1	1	0	0	*1*	*0*
⋮	⋮	⋮	⋮	⋮	⋮	⋮

order to create a data matrix that is less sparse, with a greater prevalence of event co-occurrences.

3. The **x** event indicator vectors for all streams are combined to form the event data matrix, **X**, where each column represents a sensor stream and each row represents a time slot, and so we define:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_T} \end{bmatrix} \tag{7.2}$$

where a data instance at one time slice, $\mathbf{x_t}$, is an S dimensional binary vector:

$$\mathbf{x_t} = [x_{t1}, x_{t2}, ...x_{tS}], \tag{7.3}$$

where x_{ts} is the binary event indicator for stream s at time t, and where there are S total sensor streams.

An example of an event data matrix for a building with two doors (four sensor streams) is in Table 7.1. In our data set, **X** is a 4368 by 12 matrix.

7.2 Problem Formulation and Notation

In the event data matrix $\mathbf{X}$, we expect to find some common patterns. Perhaps when data stream 1 shows event activity, more often than not stream 2 also shows event activity and neither streams 3 nor 4 show event activity. Our goal in this section is to define clusters which explain these recurring patterns.

More formally, we define a cluster, k, as an S-dimensional Bernoulli distribution with parameters

$$\mathbf{p_k} = [p_{k1}, p_{k2} \dots p_{kS}] \tag{7.4}$$

where p_{ks} is the Bernoulli parameter for stream s of cluster k, and is defined as the probability that a data instance $\mathbf{x_t}$, which is generated by cluster k, has an event indicated for stream s (i.e. $x_{ts} = 1$). Table 7.2 shows possible parameter settings that could have been used to generate the example data in Table 7.1. The parameter settings in Table 7.2 are arbitrary and are for illustrative purposes only.

If we assume that each data instance $\mathbf{x_t}$ is generated by one, and only one, of the K clusters; then we can define a hidden data matrix $\mathbf{Z}$, made up of binary variables

Cluster	p_{k1}	p_{k2}	p_{k3}	p_{k4}	α_k
$k = 1$	.95	.6	.02	.03	.2
$k = 2$	.01	.02	.8	.98	.8

Table 7.2: Example of cluster parameter values that could have been used to generate the data seen in Table 7.1. These values are arbitrary and are for illustrative purposes only.

indicating which cluster is responsible for generating each data instance. Specifically,

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z_1} \\ \mathbf{z_2} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{z_T} \end{bmatrix} \tag{7.5}$$

where the hidden cluster indicator vector at one time slice, $\mathbf{z_t}$, is a K dimensional binary vector

$$\mathbf{z_t} = [z_{t1}, z_{t2}, ...z_{tk}], \tag{7.6}$$

where $z_{tk} = 1$ if the data instance $\mathbf{x_t}$ was generated by cluster k and $z_{tk} = 0$ if $\mathbf{x_t}$ was generated by one of the other $K - 1$ clusters. Note that the variable z is being used differently in this chapter than in previous where it referred to the event state variable.

By this definition, $\mathbf{Z}$ has T rows and K columns. The number of clusters K is unknown. For the purpose of illustration, however, assume that two clusters (i.e. two mixture components) are present in our observations $\mathbf{X}$. Then the hidden data matrix $\mathbf{Z}$ would appear as in Table 7.1. Possible values of $\mathbf{z_t}$ (given the parameter settings of Table 7.2) are listed in Table 7.1 for illustration purposes, in practice these data labels are unknown and will be inferred.

The likelihood of one data instance $\mathbf{x_t}$ being generated by a particular cluster k can then be written as

$$p(\mathbf{x_t}|z_{tk} = 1) = \prod_{s=1}^{S} {p_{ks}}^{x_{ts}} (1 - p_{ks})^{1-x_{ts}} \tag{7.7}$$

Our objective is to find the hidden cluster indicator data $\mathbf{Z}$, and the Bernoulli parameters that define each cluster $\mathbf{p_k}$ that maximize the probability of generating the data, $\mathbf{X}$.

Before discussing data likelihood further, however, we need to define parameters for cluster (mixture) weights and simplify the notation. For each of the K clusters we define a mixture weight, α_k, that is the marginal probability of cluster k

$$\alpha_k = p(z_k = 1) \tag{7.8}$$

which can be thought of as the popularity of cluster k, and where $\sum_{k=1}^{K} \alpha_k = 1$.

Table 7.2 shows possible cluster weight values for generating the data in Table 7.1.

To simplify notation we define a parameter $\mathbf{\Theta}$ that encompasses the cluster variables so that

$$\mathbf{\Theta} = [\alpha_1, \alpha_2, \ldots \alpha_K,\ \mathbf{p_1}, \mathbf{p_2} \ldots \mathbf{p_K}] \tag{7.9}$$

and

$$\theta_{\mathbf{k}} = [\alpha_k, \mathbf{p_k}] \tag{7.10}$$

7.3 EM Algorithm for a Mixture of Multi-Dimensional Conditionally Independent Bernoulli Distributions

7.3.1 Data Likelihood and the Need for EM

Using the notation introduced in Section 7.2, our objective is to find the parameters $\mathbf{\Theta}$ that maximize the data likelihood, $p(\mathbf{X}|\mathbf{\Theta})$. $p(\mathbf{X}|\mathbf{\Theta})$ is not easy to model since it is a mixture of functions; however, the joint distribution, $p(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta})$, factors into distributions that are more tractable,

$$p(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta}) = p(\mathbf{X}|\mathbf{Z}, \mathbf{\Theta})p(\mathbf{Z}|\mathbf{\Theta}) \tag{7.11}$$

$\mathbf{Z}$ is unknown, but we can marginalize the joint distribution to find $p(\mathbf{X}|\mathbf{\Theta})$; so the incomplete data likelihood (incomplete because $\mathbf{Z}$ is not known) can then be written as

$$p(\mathbf{X}|\mathbf{\Theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z}, \mathbf{\Theta})p(\mathbf{Z}|\mathbf{\Theta}) \tag{7.12}$$

Since each of the T data instances are independent, Equation 7.12 becomes

$$\prod_{t=1}^{T} \sum_{k=1}^{K} p(\mathbf{x_t}|z_k, \mathbf{\theta_k})p(z_k) \tag{7.13}$$

where the $\sum_K$ term appears because each data instance $\mathbf{x_t}$, $\mathbf{z_t}$ has K dimensions, one for each of the mixture components (Note: the terms "mixture component," "component," and "cluster" are used interchangeably).

The likelihood equation in 7.13 is easy to work with because we have a model for $p(\mathbf{x}|z_k, \mathbf{\theta_k})$ (refer to Equation 7.7), and $p(z_k)$ is the mixture weight for component k ($p(z_k) = \alpha_k$) as discussed in Equations 7.8.

For computational purposes, the log of the data likelihood equation (7.13) is used in order to prevent underflow due to large values of T:

$$\ln p(\mathbf{X}|\mathbf{\Theta}) = \sum_{t=1}^{T} \ln \sum_{k=1}^{K} p(\mathbf{x_t}|z_k, \mathbf{\theta_k})p(z_k) \tag{7.14}$$

Equation 7.14, however, is difficult to optimize because it contains the log of a sum. The benefit of the EM algorithm is that it allows us to get rid of the sum term by performing calculations on the complete data likelihood.

7.3.2 Complete Data Likelihood

If instead of just observing the binary sequences, $\mathbf{X}$, as in Table 7.1, we also knew the hidden component membership values $\mathbf{Z}$; then the calculation in Equation 7.14 becomes more tractable.

First observe the data likelihood of the complete data set (meaning the observed data $\mathbf{X}$, plus the "true" values of the hidden variables $\mathbf{Z}$),

$$p(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta}) = \prod_{t=1}^{T} \prod_{k=1}^{K} \Big(p(\mathbf{x_t}|z_k, \theta_\mathbf{k}) p(z_k) \Big)^{z_{tk}} \tag{7.15}$$

where $z_{tk} = 0$ for all but one value of k in each time segment t. Therefore Equation 7.15 reduces to a product of only T terms.

So comparing the incomplete data likelihood (Equation 7.13) with the complete data likelihood in Equation 7.15, we see that a sum over mixture components has been replaced with a product term. The log likelihood of the complete data is then

$$\ln p(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta}) = \sum_{t=1}^{N} \sum_{k=1}^{K} z_{tk} \Big(\ln p(\mathbf{x_t}|z_k, \theta_\mathbf{k}) + \ln p(z_k) \Big) \tag{7.16}$$

Note that in the log likelihood of the complete data, Equation 7.16, the log term is *inside* the sum over T and K; whereas in log likelihood of the incomplete data, Equation 7.14, the log is *outside* the sum over K term. So if we can fix $\mathbf{Z}$, we can work with the more tractable calculation in Equation 7.16.

7.3.3 Q function

The complete log likelihood (Equation 7.16) is easier to use, but we can not compute it directly because in reality the missing variables, $\mathbf{Z}$ are not known. However, the EM algorithm allows us to indirectly optimize the complete data likelihood.

The EM algorithm [17, 10] is an iterative method consisting of two repeated steps (the E-step and M-step). In the E-step we compute

$$p(z_{tk} = 1|\mathbf{x_t}, \theta_\mathbf{k}) \tag{7.17}$$

which is the posterior probability that data instance $\mathbf{x_t}$ came from component k given the model parameters, $\mathbf{\Theta}$ (which are fixed from the previous M-step). $p(z_{tk})$ is known as the membership probability of component k for data instance $\mathbf{x_t}$, or the component "responsibility." So there are $T \times K$ membership responsibilities to compute in each E-step.

In the M-step we find the model parameters, $\mathbf{\Theta}$, that maximize the data likelihood given the membership probabilities found in the previous E-step.

The values for z_{tk} that we get from the E-Step (Equation 7.17) are probabilities that range between 0 and 1; however, the complete data likelihood equations (7.15 and 7.16) require binary values of the hidden parameters z_{tk}. We can still combine the membership probabilities z_{tk} found in the E-step with Equation 7.16 by using the expectation of the complete data log likelihood. This expectation is referred to as the Q function.

$$E[\ln p(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta})] = Q(\mathbf{\Theta^{old}}, \mathbf{\Theta}) \tag{7.18}$$

where $\mathbf{\Theta^{old}}$ are the model parameters calculated in the previous M-Step (and are fixed), and $\mathbf{\Theta}$ are the model parameters being optimized in the current step (and are variable). The Q function is then:

$$Q(\mathbf{\Theta^{old}}, \mathbf{\Theta}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{\Theta^{old}}) \sum_{t=1}^{T} \sum_{k=1}^{K} z_{tk} \Big(\ln p(\mathbf{x_t}|z_k, \theta_\mathbf{k}) + \ln p(z_k) \Big) \tag{7.19}$$

where we consider every possible combination of cluster labels ($\mathbf{Z}$), and weight each according to the posterior probabilities of each combination.

$p(\mathbf{Z}|\mathbf{X}, \mathbf{\Theta^{old}})$ can be written as a product of the membership probabilities found in the E-Step (Equation 7.17), and the majority of terms in Equation 7.19 drop out leaving

$$Q(\Theta^{old}, \Theta) = \sum_{t=1}^{T} \sum_{k=1}^{K} p(z_{tk} = 1|\mathbf{x_t}, \theta_\mathbf{k}^\mathbf{old}) \Big(\ln p(\mathbf{x_t}|z_k, \theta_\mathbf{k}) + \ln p(z_k) \Big) \tag{7.20}$$

where $p(z_k)$ is the mixture weight α_k.

This is the equation we will be maximizing in the M-step. The membership probabilities, $p(z_{tk} = 1|\mathbf{x_t}, \theta_\mathbf{k}^\mathbf{old})$ which are found in the E-step are held constant during the M-Step. To avoid confusion between the variable valued $\mathbf{\Theta}$ and the constant valued $\mathbf{\Theta^{old}}$, in the remainder of the discussion the membership probabilities will be denoted

$$m_{tk} = p(z_{tk} = 1|\mathbf{x_t}, \theta_\mathbf{k}^\mathbf{old}) \tag{7.21}$$

where m_{tk} is calculated in the E-step and is a constant scalar value in the M-step.

The Q function of Equation 7.20 is then

$$Q(\Theta^{old}, \Theta) = \sum_{t=1}^{T} \sum_{k=1}^{K} m_{tk} \Big(\ln p(\mathbf{x_t}|z_k, \theta_\mathbf{k}) + \ln \alpha_k \Big) \tag{7.22}$$

7.3.4 E-Step

In this section we examine the E-step in more detail. As mentioned in Section 7.3.3, the E-step calculates the membership probabilities (the posterior probability of the missing data). The parameters found in the previous M-step, $\mathbf{\Theta}$, are held constant, and Bayes Rule is used to find the membership probabilities, m_{tk}.

$$m_{tk} = p(z_{tk} = 1|\mathbf{x_t}, \theta_\mathbf{k}) = \frac{p(\mathbf{x_t}|z_{tk} = 1, \theta_\mathbf{k})p(z_{tk} = 1|\theta_\mathbf{k})}{\sum_{k=1}^{k} p(\mathbf{x_t}|z_{tk} = 1, \theta_\mathbf{k})p(z_{tk} = 1|\theta_\mathbf{k})} \tag{7.23}$$

where $p(z_{tk} = 1|\theta_\mathbf{k})$ is the mixture weight for cluster k and is independent of Θ $[p(z_{tk} = 1|\theta_\mathbf{k}) = p(z_k = 1) = \alpha_k]$. So Equation 7.23 can be rewritten as

$$m_{tk} = \frac{p(\mathbf{x_t}|z_{tk} = 1, \theta_\mathbf{k})\alpha_k}{\sum_{k=1}^{K} p(\mathbf{x_t}|z_{tk} = 1, \theta_\mathbf{k})\alpha_k} \tag{7.24}$$

and $p(\mathbf{x_t}|z_{tk} = 1, \theta_\mathbf{k})$ is a mixture of Bernoulli distributions as defined in Equation 7.7, thus the E-step equation finally becomes

$$m_{tk} = \frac{\alpha_k \prod_{s=1}^{S} {p_{ks}}^{x_{ts}}(1 - p_{ks})^{1-x_{ts}}}{\sum_{k=1}^{K} \alpha_k \prod_{s=1}^{S} {p_{ks}}^{x_{ts}}(1 - p_{ks})^{1-x_{ts}}} \tag{7.25}$$

where the α_k's and the p_{ks}'s are constant values found in the previous M-step and the $\mathbf{x_t}$'s are observed.

7.3.5 M-Step

In the E-step, $\mathbf{\Theta}$, the Bernoulli distribution parameters for each dimension for each mixture component $\{p_{k1}, p_{k2}, ...p_{ks}\}$ and the mixing weights α_k, are fixed, and the membership probabilities m_{kt} are calculated. In the M-step, the parameters $\mathbf{\Theta}$ are now variables and are optimized given the fixed membership probabilities calculated in the previous E-step.

The optimal values for the α_k's and the p_{ks}'s are found by maximizing the Q function. For a multi-dimensional Bernoulli distribution, Equation 7.22 can be written as

$$Q(\Theta^{old}, \Theta) = \sum_{t=1}^{T} \sum_{k=1}^{K} m_{tk} \left[\ln \prod_{s=1}^{S} {p_{ks}}^{x_{ts}} (1 - p_{ks})^{1 - x_{ts}} + \ln \alpha_k \right] \tag{7.26}$$

where $p(\mathbf{x_t} | z_k, \Theta_k)$ is substituted using the definition in Equation 7.7.

Taking the log of the product, Equation 7.26 becomes

$$\sum_{t=1}^{T} \sum_{k=1}^{K} m_{tk} \left[\sum_{s=1}^{S} \Big(x_{ts} \ln p_{ks} + (1 - x_{ts}) \ln (1 - p_{ks}) \Big) + \ln \alpha_k \right] \tag{7.27}$$

First, we optimize p_{ks} by taking the partial derivative of Equation 7.27 with respect to p_{ks}:

$$\begin{aligned} \frac{\partial Q}{\partial p_{ks}} &= \sum_{t=1}^{T} m_{tk} \left(\frac{x_{ts}}{p_{ks}} - \frac{(1 - x_{ts})}{(1 - p_{ks})} \right) \\ &= \sum_{t=1}^{T} m_{tk} \left(\frac{x_{ts}(1 - p_{ks}) - (1 - x_{ts}) p_{ks}}{p_{ks}(1 - p_{ks})} \right) \\ &= \frac{1}{p_{ks}(1 - p_{ks})} \sum_{t=1}^{T} m_{tk}(x_{ts} - p_{ks}) \end{aligned} \tag{7.28}$$

Setting Equation 7.28 to zero to maximize p_{ks} gives

$$\sum_{t=1}^{T} m_{tk} x_{ts} = p_{ks} \sum_{t=1}^{T} m_{tk}$$

and finally

$$p_{ks} = \frac{\sum_{t=1}^{T} m_{tk} x_{ts}}{\sum_{t=1}^{T} m_{tk}} \quad = \quad \bar{x}_{ks} \tag{7.29}$$

This result is intuitive; $\bar{x}_{ks}$ is the empirical average of the x values in dimension s weighted by the probability (m_{ks}) that each x_{ts} was generated by cluster k.

The denominator of Equation 7.29 is the expected number of the T data examples (x_t) that are presumed to come from cluster k, and will be denoted as N_k in the remainder of the discussion,

$$N_k = \sum_{t=1}^{T} m_{tk} \tag{7.30}$$

where $\sum_{k=1}^{K} N_k = T$.

The second set of parameters optimized in the M-step are the α_k parameters. When maximizing the Q function (Equation 7.27) with respect to α_k, we need to enforce the constraint: $\sum_{k=1}^{K} \alpha_k = 1$. We do this using a Lagrange multiplier, λ. The function then becomes

$$\sum_{t=1}^{T}\sum_{k=1}^{K} m_{tk} \left(\sum_{s=1}^{S} [x_{ts} \ln p_{ks} + (1 - x_{ts}) \ln (1 - p_{ks})] + \ln \alpha_k \right) + \lambda \sum_{k=1}^{K} (\alpha_k - 1) \tag{7.31}$$

When taking the partial derivative of this Lagrangian with respect to α_k most of the terms in Equation 7.31 drop out. The only terms that include α_k are $\sum_{t=1}^{T} \sum_{k=1}^{K} m_{tk} \ln \alpha_k$ and $\lambda \sum_{k=1}^{K} \alpha_k$. Also, the sum over K disappears because when finding the derivative with respect to α_2, for example, terms involving $\alpha_1, \alpha_3, \alpha_4$... will all go to zero. The derivative is then

$$\left(\sum_{t=1}^{T}\frac{m_{tk}}{\alpha_k}\right)+\lambda \tag{7.32}$$

Setting Equation 7.32 equal to zero to find the value of α_k that maximizes the data likelihood, and substituting N_k for $\sum_{t=1}^{T} m_{tk}$ gives

$$N_k = -\lambda\alpha_k \tag{7.33}$$

To find a usable solution for α_k, we need to find λ. If both sides of Equation 7.33 are summed with respect to K,

$$\begin{aligned}\sum_{k=1}^{K}\alpha_k &= -\sum_{k=1}^{K}\frac{N_k}{\lambda} \quad \text{The mixture weights must sum to one, so}\\ 1 &= -\sum_{k=1}^{K}\frac{N_k}{\lambda} \quad \text{and the sum of the } N_k\text{'s over all K is T, so}\\ \lambda &= -T \end{aligned} \tag{7.34}$$

We plug our solution for λ back into Equation 7.33 giving:

$$N_k = \alpha_k \times T \quad \text{and} \quad \alpha_k = \frac{N_k}{T} \tag{7.35}$$

which also is an intuitive result – the optimal mixture weight is the empirical average found by dividing the expected number of instances explained by cluster k by the total number of instances.

Now, the optimized values for the Bernoulli distribution parameters $\{p_{k1}, p_{k2}, ...p_{ks}\}$ found in Equation 7.29, and the mixing weights α_k found in Equation 7.35 will be

Mixing Weights

	α1	α2
true	0.6	0.4
Initial EM	0.07	0.93
learned	0.607	0.393

Bernoulli Parameters

	p1	p2	p3	p4
True component 1	.9	.8	.2	.3
Initial EM Setting 1	.339	.912	.788	.767
Learned Component 1	.881	.798	.190	.295
True Component 2	.3	.2	.7	.7
Initial EM Setting 2	.510	.455	.249	.760
Learned Component 2	.299	.174	.719	.721

Table 7.3: Simulation parameters used to sample the simulated data $\mathbf{X}$, compared with the parameter values used to initialize the EM algorithm, and the parameters learned by the EM algorithm.

held constant for the following E-step. The iteration continues until convergence is achieved. In the results in this chapter, convergence was determined by monitoring the data log likelihood until it stopped changing as in Figure 7.1. Both the data log likelihood and the expected complete data log likelihood will only increase during this process.

7.4 Simulation

Before running the EM algorithm on the door data set, a simulation was run to determine the feasibility of the approach. A data set was created where each instance, $\mathbf{x_t}$, is a 4-dimensional binary variable as in Table 7.1. The simulated data $\mathbf{X}$ were generated using two mixing components (i.e. two clusters) with mixing weights and

Bernoulli parameters as in Table 7.3. Three thousand data samples were generated using ancestral sampling – for each simulated $\mathbf{x_t}$, a component k was first sampled given the true mixing weights, then the data instance $\mathbf{x_t}$ was created by sampling binary x_{ts} variables using the true Bernoulli parameter settings for component k (p_{ks}).

The true model parameters were hidden from the EM algorithm, and random values from a uniform distribution were used to initialize the model parameters before the start of the EM algorithm. Table 7.3 shows a comparison of the true model parameters, the initial settings, and the parameters learned by the EM algorithm after 50 iterations.

The EM algorithm gave satisfactory results and converged quickly. Figure 7.1 shows a plot of the log likelihood calculated after each iteration of the EM algorithm. One nice property of EM is that the data likelihood will only increase after every iteration [49]; however, it may converge to a local maximum rather than a global maximum. After ten iterations, the log likelihood does not improve significantly. This type of plot is commonly used to test convergence of an EM algorithm.

7.5 Case Study 5: Clustering Building Streams

With the feasibility of our approach validated by the simulation, we apply the EM algorithm to real data.

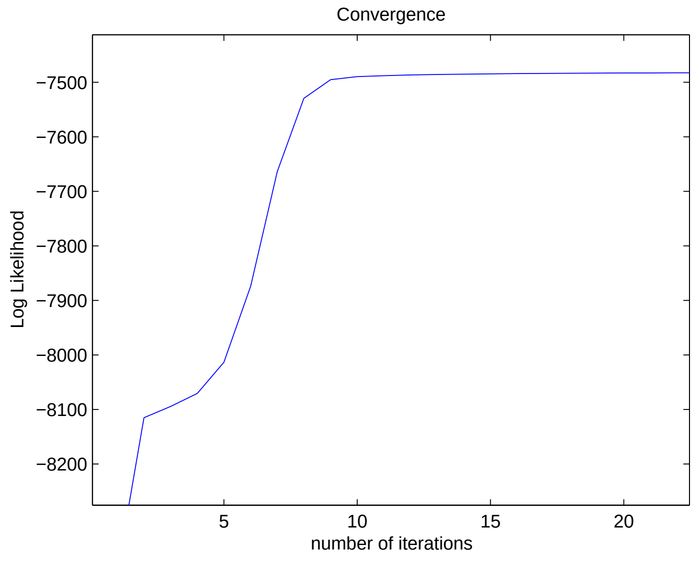


Figure 7.1: Convergence testing for the simulation experiment using log likelihood

Observed Data (X)												Cluster (Z)
0	1	0	0	0	0	0	1	0	1	0	0	?
0	1	0	0	0	0	0	1	0	1	0	0	?
0	1	1	0	0	0	0	1	0	0	0	0	?
0	1	0	0	0	0	0	1	0	0	0	0	?
0	0	0	0	0	0	0	0	0	0	0	1	?
0	0	1	0	0	0	0	0	0	0	0	0	?
0	0	0	0	0	0	0	1	0	0	0	0	?
0	1	0	0	0	0	0	0	0	0	0	0	?
0	0	0	0	0	0	0	0	0	1	0	0	?
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.

Table 7.4: Inferred door event data. A “1” means event activity was seen in that stream at that time-segment. **Z**, the hidden component information, also has 12 columns (not shown in the figure); one of which in each row contains a “1” indicating the component that generated that data instance, and the rest are zeros.

7.5.1 Data Description

The door data are similar to the data in the simulation, except that there are 12 streams so $\mathbf{X}$ has 12 columns. We do not know the number of mixture components, so the observed data appears as in Table 7.4. The data for each column s come from a run of the event model involving only one data stream with T time-slices.

To run the EM algorithm, the number of clusters, K, must be specified. We do not know how many clusters of sensors there are in reality, so there is not a clear choice for setting K. Still, we can get information about which sensors tend to have events at the same time by varying K, and examining the results of all the runs. The results are often not particularly sensitive to the value of K (meaning the same patterns are generally found regardless of the value of K), as will be demonstrated in the results section.

7.5.2 Brute Force Clustering: Creating a Validation Set

With only 12 data streams, a brute force method can be used to cluster the data streams to get a rough idea of how many clusters to expect, and of which streams appear to have correlated event activity before running EM. In this method all 4095 possible combinations of events among the 12 streams are compared with the 581 events to find the most common combinations. This brute force approach is only possible for data sets with a small number of streams. The number of combinations that need to be checked is a function of the number of streams, S:

$$\sum_{i=1}^{S} \frac{S!}{i!(S-i)!} \tag{7.36}$$

# time-slices (out of 581 possible)	Data stream ID's (column numbers) with event activity
116	2 & 8 (Front Door In and Out)
109	8 (Front Out)
59	2 (Front In)
44	10 (Loading Dock Out)
39	6 & 12 (Side door In and Out)
28	3
23	4 & 10 (Loading Dock In and Out)
21	12
21	6
21	4
19	6 & 8 (Side in and Front Out)
15	9
15	7
15	8 & 10 (Front Out and Loading Dock Out)
15	2 & 6 (Front and side in)
14	8 & 12
12	2 & 12
11	6, 8, & 12
10	2 & 10
9	1
9	2 & 4
8	7 & 8
8	2, 6 & 12
7	2, 8 & 10

Table 7.5: Cluster results for the brute force method. 116 out of 581 possible event time-segments showed event activity for both the entrance and exit streams for the front door (columns 2 and 8 of **X**), probably due to atrium events with a lot of churn (the same people entering and exiting multiple times throughout the event).

So the number of combinations to checked is only 4095 when $S = 12$, but this number increases to 1.210^{30} when S goes to 100 and the brute force method is infeasible for cases such as our traffic data set where $S \sim 10,000$.

When comparing the 581 time slices where events were predicted with the 4095 possible event combinations, only 83 different event stream combinations were found. Before ranking the matches, combinations of streams (i.e. clusters) that are subsets of other clusters are examined to increase the number of matches for the subset. For example if one cluster of streams $\{S2, S4\}$ has 20 matches, but another cluster of streams $\{S2, S4, S5\}$ has 15 matches, then the $\{S2, S4\}$ cluster is modified to have 35 matches.

The top 24 clusters by the brute force method described above are listed in Table 7.5 along with the number of occurrences for each cluster. For context, there are thirteen weeks of count data binned into 30-minute intervals, giving 4368 time-slices where an event could occur. 581 of those 4368 time slices showed positive event activity in at least one of the 12 data streams. We will return to this section for comparison in our analysis of the EM results.

7.5.3 EM Experiment with Real Data

In this section we will analyze the results of applying the EM algorithm to the real door data (the same data set shown in Table 7.4 and used in the brute force clustering). The experimental procedure is similar to that of the simulation (Section 7.4). The initial guesses for the parameters were chosen from a uniform random distribution. The number of clusters, however, is not known, and we do not have the true parameters to compare with the parameters learned by EM after the algorithm converges. We do have some validation data in the form of the brute force clustering performed in

Table 7.6: Model parameters learned by EM for the real door data, number of clusters set to 2.

α_k	p_{k1}	p_{k2}	p_{k3}	p_{k4}	p_{k5}	p_{k6}	p_{k7}	p_{k8}	p_{k9}	p_{k10}	p_{k11}	p_{k12}
.434	.001	.582	.020	.007	.000	.005	.010	.808	.022	.033	.000	.003
.566	.060	.144	.113	.165	.040	.230	.096	.156	.075	.239	.036	.213

Section 7.5.2 and displayed in Figure 7.5. The number of clusters K to be learned by the EM algorithm will be varied in the experiments.

In the first run, the number of mixture components K was set to 2 for illustration purposes and for the sake of curiosity. The parameters learned by EM for the two components are seen in Table 7.6. The first cluster is dominated by the entrance and exit streams at the front door (streams 2 and 8). This is not surprising since the top three results of the brute force experiment were: 1) streams 2 and 8 together, 2) stream 8 alone and 3) stream 2 alone. The second cluster was a combination of all streams that explains all the remaining less popular patterns.

The algorithm converged within 10 iterations of the EM algorithm as seen in Figure 7.2. Multiple runs at $K = 2$ with different initial settings for the model parameters led to very similar results.

The next experiment tested 9 mixture components, $K = 9$. Table 7.7 shows the model parameters learned for the 9 components. The stream probabilities that are in **bold** typeface in the table indicate streams that are significant for each cluster. The decision of which streams to mark as significant was based simply on visual inspection. A better approach would choose significant streams not only based on the magnitude of p_{ks}, but also based on the relative value of p_{ks} for the same stream s across clusters. For example, imagine stream 1 has a seemingly low probability for cluster A but has near zero probability in all other clusters. Stream 1, then, is significant for cluster A

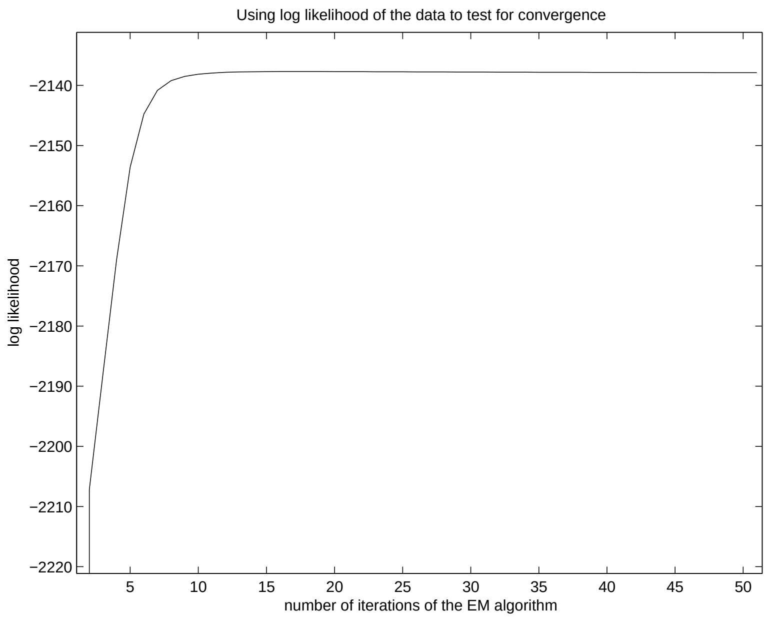


Figure 7.2: Convergence was rapid for $K = 2$.

		FDI		LDI		SDI		FDO		LDO		SDO
α_k	p_{k1}	p_{k2}	p_{k3}	p_{k4}	p_{k5}	p_{k6}	p_{k7}	p_{k8}	p_{k9}	p_{k10}	p_{k11}	p_{k12}
.486	.006	**.575**	.020	.010	.004	.001	.019	**.784**	.022	.034	.000	.000
.185	.016	.193	.022	.042	.024	**.661**	.049	.226	.044	.015	.000	**.610**
.088	.000	.000	.000	.000	.037	.000	.032	.070	.000	**1.00**	.020	.035
.078	.000	.095	.066	**1.00**	.021	.000	.000	.080	.000	**.469**	.000	.000
.077	.000	.000	**.697**	.000	.000	.037	.000	.000	**.381**	.000	.000	.000
.061	**.445**	.122	.000	.000	.119	.000	**.558**	.000	.000	.000	.000	.097
.017	.100	.100	.000	.100	**.201**	.000	.100	.100	.100	.000	**.995**	.000
.006	.000	.000	.000	**.352**	.000	**.854**	.000	.000	.000	**1.00**	.000	.000
.002	.000	**1.00**	.000	**1.00**	.000	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	.000

Table 7.7: Model parameters learned by EM for the real door data. The number of mixture components is set to 9. The stream probabilities that are in **bold** typeface in the table indicate streams that are significant for each cluster.

because if any event activity is seen in this stream, we can be confident that event activity in other streams is occurring with probabilities according to cluster A.

The main multi-stream clusters found by the brute force method, Table 7.5, appear in the $K = 9$ EM results as well. Both sets of results rank the front door entrance and exit stream combination as the most common cluster $\{S2, S8\}$. Additionally, both

		FDI		LDI		SDI		FDO		LDO		SDO
α_k	p_{k1}	p_{k2}	p_{k3}	p_{k4}	p_{k5}	p_{k6}	p_{k7}	p_{k8}	p_{k9}	p_{k10}	p_{k11}	p_{k12}
.215	.000	**.458**	.019	.000	.000	.000	.011	**1.00**	.000	.007	.000	.000
.117	.000	**1.00**	.000	.000	.000	.008	.000	**.304**	.000	.000	.000	.044
.104	.000	.000	.000	**.206**	.000	.050	.011	.000	.000	**1.00**	.017	.033
.065	.000	.000	**.807**	.000	.044	.000	.000	.005	.108	.000	.158	.000
.058	.000	.000	.000	.003	.000	**.289**	.036	.093	.000	.000	.000	**1.00**
.056	.030	.072	.001	.000	.004	.000	.018	**1.00**	.000	.002	.025	.000
.052	.000	**1.00**	.000	**.132**	.000	**.168**	.000	**.362**	.000	.000	.000	.000
.047	.000	.016	.000	.000	.000	**1.00**	.000	.041	.024	.000	.000	.175
.039	.000	.000	.002	**1.00**	.044	.000	.000	.000	.000	.004	.044	.039
.029	.000	.119	.000	.000	.000	.000	**1.00**	.000	.000	.000	.059	.000
.028	.002	.173	.025	.000	.000	.000	.017	**1.00**	.000	.000	.012	.000
.026	.000	.000	.037	.000	.000	.001	.000	.000	**1.00**	.000	.000	.000
.026	**1.00**	.200	.000	.000	.000	.000	.200	.000	.000	.000	.000	.067
.025	.000	**.591**	.000	**.383**	.000	.000	.000	**.768**	.000	**1.00**	.000	.000
.023	.000	.000	.000	.152	.076	**.995**	.210	**.424**	.000	.000	.000	**1.00**
.017	.000	.000	.000	.000	**.537**	.000	.172	**.496**	.000	**.545**	.000	.000
.016	.000	**1.00**	**.221**	.000	.000	.000	.000	**1.00**	**.444**	.000	.000	.000
.014	.000	**.993**	.000	.000	.125	**1.00**	.000	**.503**	.000	.125	.000	1.000
.012	.000	.000	.000	.000	.000	**.461**	.000	**1.00**	**.290**	.000	.000	.000
.012	.000	.000	**.412**	**1.00**	.000	.000	.000	.000	.000	**.689**	.000	.000
.009	.000	.000	**.596**	.000	.000	**1.00**	.000	.000	**.667**	.000	.000	**.664**
.005	**.374**	**.497**	.000	**.374**	.000	**.374**	**1.00**	**1.00**	.000	.000	.000	.000
.005	**1.00**	.000	.000	.000	1.000	.000	**.667**	.000	.000	.000	**.333**	**.333**
.002	.000	**1.00**	.000	**1.00**	.000	**1.00**	**1.00**	**1.00**	**1.00**	.000	**1.00**	.000

Table 7.8: Model parameters learned by EM for the real door data, when the number of mixture components is set to 24. The stream probabilities that are in **bold** typeface in the table indicate streams that are significant for each cluster.

sets of results cluster the loading dock entrance and exit streams together $\{S4, S10\}$ and the side door entrance and exit streams together $\{S6, S12\}$. It is interesting that the the EM algorithm does not find single-stream clusters when K=9.

One more experiment was run, setting K to 24. The results of this test are displayed in Table 7.8.

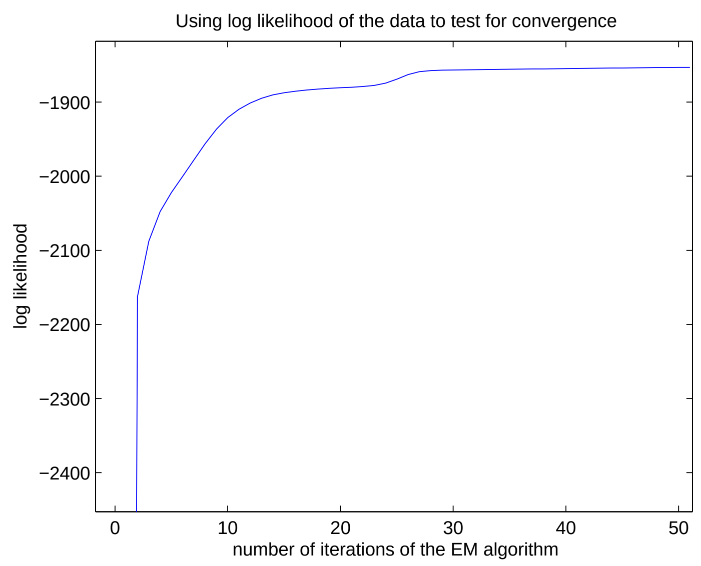


Figure 7.3: Convergence was slightly slower when K was increased to 24.

When $K = 9$, no individual-stream clusters were found. However, increasing K to 24 did result in finding some individual-stream clusters, such as the sixth highest ranked cluster and perhaps the fourth ranked cluster. Interestingly, the individual-stream clusters $\{S2\}$ and $\{S8\}$ (front door exit stream and front door entrance stream) were not as prominent in the EM results as in the brute force results. However, they appeared together three times in the top seven ranked clusters by EM, favoring one of the two streams in each case. Many of the lone door events seen in the brute force method probably were explained by one of these clusters in the EM run.

Figure 7.3 shows the convergence when K is set to 24. Convergence occurs quickly again, but slightly slower than when $K = 2$ as might be expected for a more complicated model.

Even though the number of clusters K is unknown for our problem, the similarity in the results of the $K = 9$ and $K = 24$ experiments show that the main relationships between the streams in our data set can be found with a reasonable setting of K. In our problem, we are interested in finding which individual-sensor streams to link in the combined model. One way this could be accomplished is by creating a cluster list ranked by the mixture weight α_k as in Table 7.8, then thresholding the list and choosing to link the streams that are part of the significant stream combinations in each cluster.

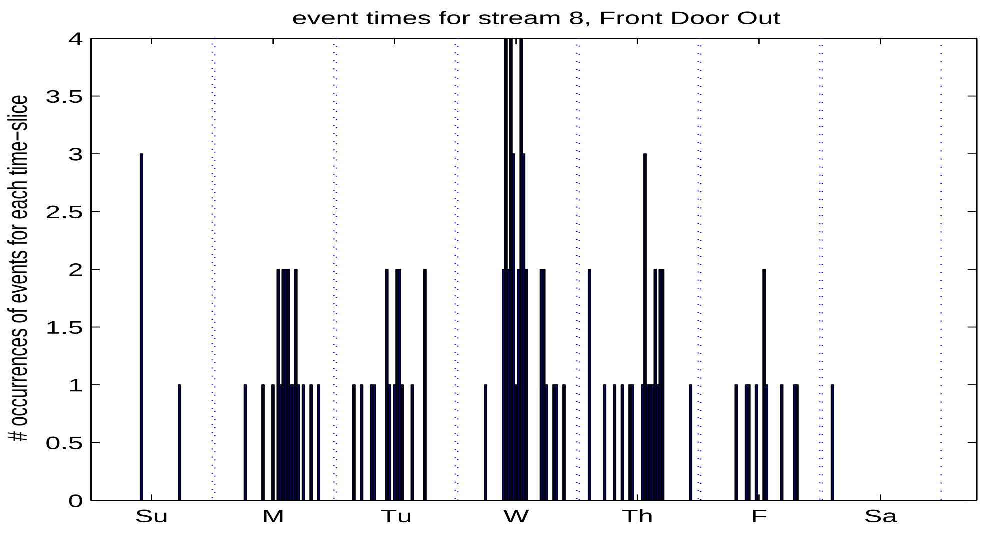
7.6 Future Work: Temporal Event Pattern Analysis

In this section, we discuss another potential extension to the building occupancy model, adding a temporal component to the event state prediction.

First using the results of the case study, the clusters with the highest mixture weights are examined for temporal patterns. Figure 7.4(a) is a bar plot that shows the frequency of event activity inferences for cluster $\{S2\}$ as a function of time of day and day of week. In Figure 7.4(b), the same data as in (a) are plotted, but kernel density estimation (using a Gaussian kernel) was used to smooth the plot. The smoothing makes the distribution of events with respect to time visually easier to interpret.

Figures 7.5 to 7.10 show the temporal patterns for the six most popular clusters (using the results of the brute force method). Since there are only 13 weeks of data, and only 581 total time segments with event activity observed, there are not enough data to make conclusive statements about the temporal patterns. However, we can make some anecdotal observations, and discuss how this type of analysis could be used to extend the model further.

Friday evenings was the most popular event time for the building in our study (Calit2), and the front door entrance and exit cluster $\{S2, S8\}$ most often was used to explain

(a) 

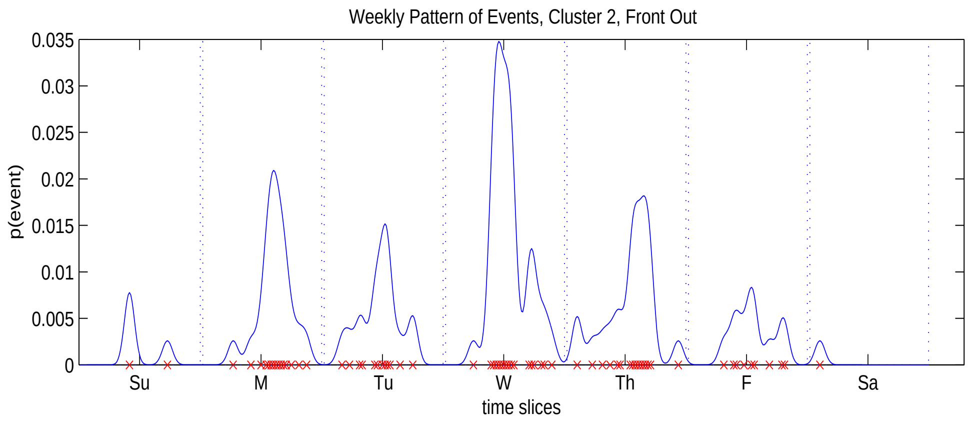
(b) 

Figure 7.4: Kernel density estimation produces a more readable plot of the distribution of events with respect to time. (a) a histogram of event occurrences at each time slot. (b) smoothed plot using a Gaussian kernel.

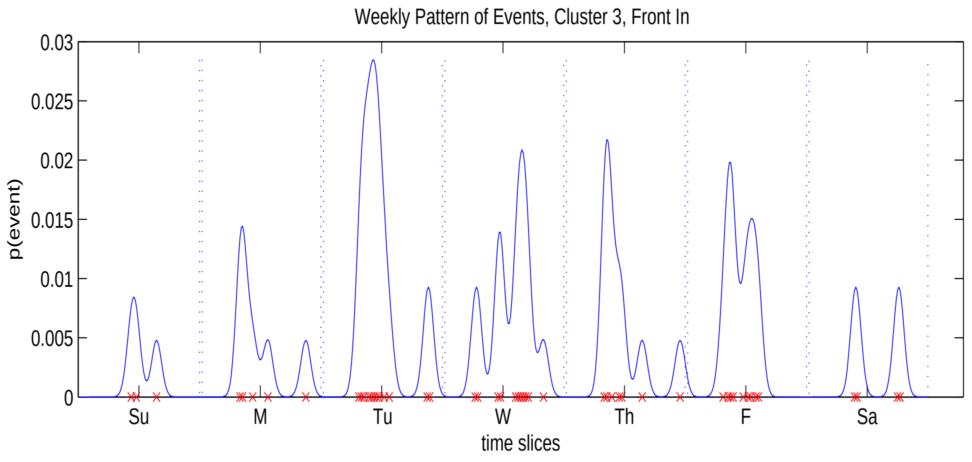


Figure 7.5: Temporal Analysis: Front Door Entrance $\{S2\}$

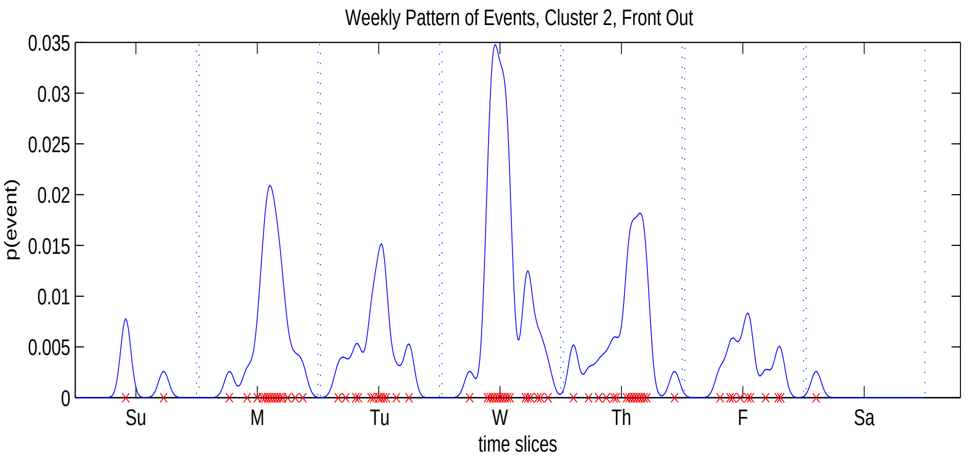


Figure 7.6: Temporal Analysis: Front Door Exit $\{S2\}$

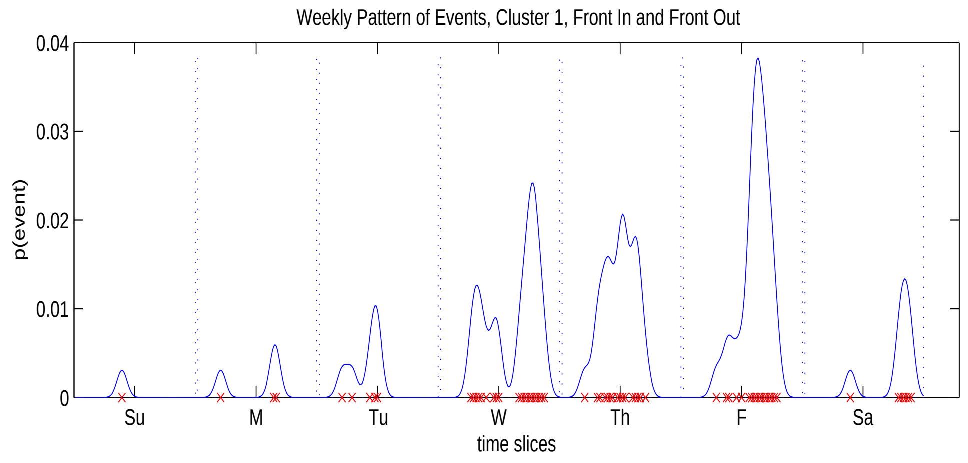


Figure 7.7: Temporal Analysis: Front Door Exit and Entrance $\{S2, S8\}$

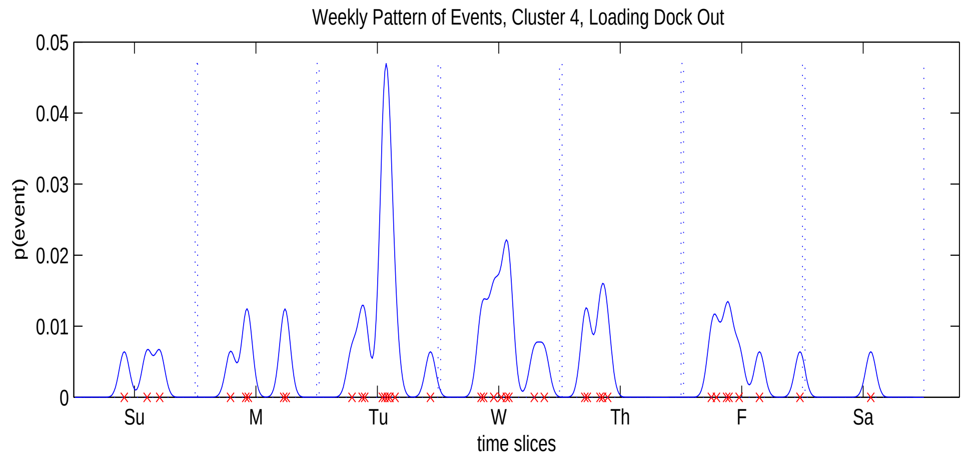


Figure 7.8: Temporal Analysis: Loading Dock Door Exit $\{S10\}$

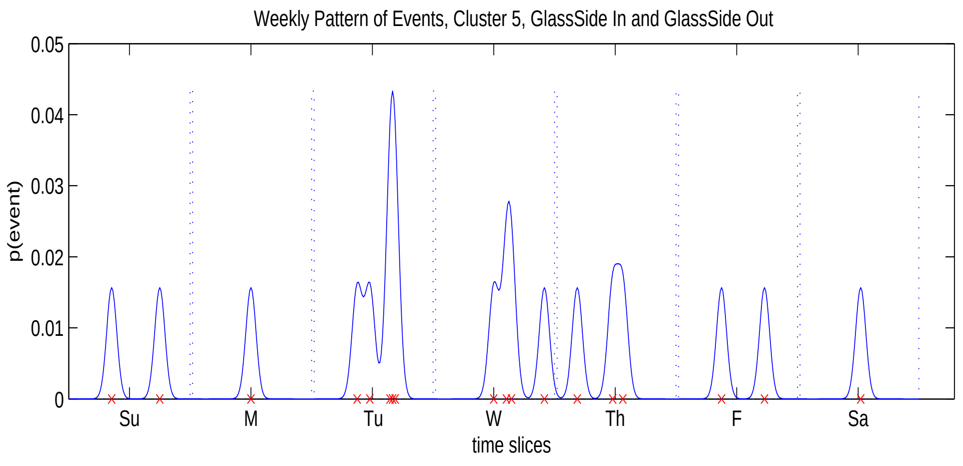


Figure 7.9: Temporal Analysis: Glass Side Door Exit and Entrance $\{S6, S12\}$

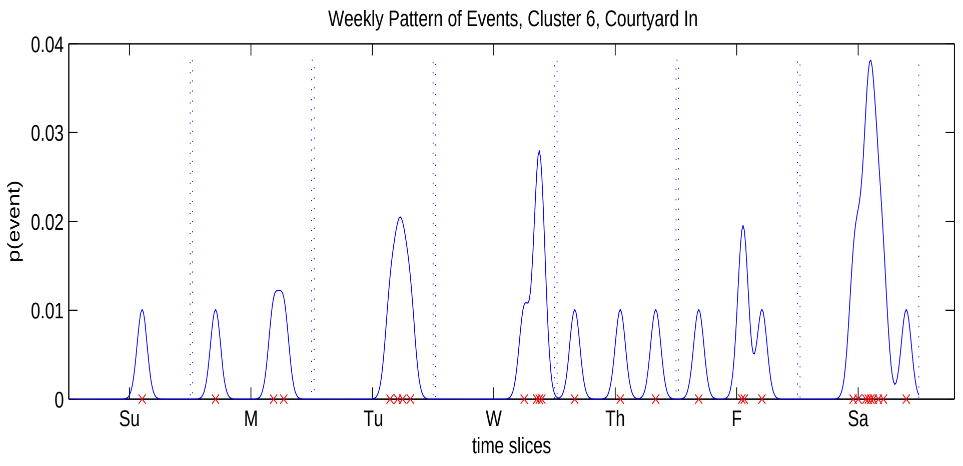


Figure 7.10: Temporal Analysis: Courtyard Door Entrance $\{S3\}$

Friday night events. The front door of the Calit2 building (pictured in the left side of Figure 6.2) has a large atrium that is popularly used for special events on Friday evenings. These atrium event often experience "churn" (people going in and out the front entrance multiple times within the same time segment) due to simultaneous attractions in the courtyard outside and the atrium inside, which explains why the $\{S2, S8\}$ cluster is often used to explain Friday night events.

One possible use for this temporal event information is to add a temporal component to the model of the event process, z (note that this z is the state variable used in previous chapters, we are no longer using z to indicate cluster membership as earlier in the chapter). In the current model, the event state z_t is influenced by the current observation o_t, the rate parameter $\lambda(t)$, and the neighboring event states z_{t-1} and z_{t+1}. If we were to extend the event model to include temporal event patterns (for example, the common $\{S2, S8\}$ Friday evening pattern), the model might become more robust in its prediction of the normal pattern $\lambda(t)$ and more sensitive in its detection of events occurring at historically popular times.

If temporal information of this type were included, the model would need to address the dynamic nature of the temporal patterns (eg. patterns can change abruptly due to construction, event scheduling changes, etc). Consider the following observation: Most events in our study occurred on weekdays. The courtyard entrance cluster (Figure 7.10) is a notable exception because it is the only major cluster to experience significant weekend event activity. The doors to the Calit2 building are locked on weekends, and the key-card reader was broken at the front door for the duration of our study. As a result, the courtyard door, which did have a working key-card reader during our study period, received most of the weekend entrance traffic (containing both event and normal activity). Presumably, this pattern changed after the card reader at the front door was repaired.

7.7 Alternate Event Paradigms

In this section we discuss two potentially helpful extensions to the clustering algorithm. In the original algorithm, streams are linked based only on event correlations within the same time segment, and only binary events are considered. Here we will remove each of these restrictions and show experimental clustering results from the generalized algorithms.

7.7.1 Allowing Events to Span Time

In the original clustering method, two streams can appear together in the same data point $\mathbf{x_t}$ only if event activity occurred for both within the same time segment. Perhaps a better way of clustering the streams is by observing whether event activity co-occurs within the same "event period", where an event period could span several time segments. In *Situation B* (of the motivating example at the start of this chapter), one of the reasons we had more confidence in the entrance event prediction was the presence of the exit event in the following time segment. The original clustering algorithm may not learn this type of time-delayed relationship. To take advantage of this time-delayed correlation, event nodes would need to be linked across time, and the clustering algorithm would need to be modified to find these correlations.

The modification to the clustering algorithm is simple as it only involves modifying the way the data matrix $\mathbf{X}$ is created – namely, by defining an event data point x_t over an event period rather than a single time-segment. There are multiple ways that one could define an event period. If the average length of an event was known, for example, the event period could start at the end of an entrance stream event and extend past the average length of an event in order to find any corresponding exit

		FDI		LDI		SDI		FDO		LDO		SDO
α_k	p_{k1}	p_{k2}	p_{k3}	p_{k4}	p_{k5}	p_{k6}	p_{k7}	p_{k8}	p_{k9}	p_{k10}	p_{k11}	p_{k12}
.237	.000	**.277**	.066	.000	.015	.000	.000	**1.00**	.038	.027	.015	.006
.165	.000	**.608**	.000	**.358**	.000	.072	.000	.115	.000	.106	.000	.000
.130	.000	.000	**.128**	.000	.000	**.742**	.000	.087	**.344**	.063	.000	**.270**
.120	.000	.000	.030	**.375**	.030	.000	.039	.002	.000	**1.00**	.000	.058
.085	.000	.000	**1.00**	.000	.000	.000	.000	.020	.084	.000	.000	.000
.082	.000	.000	.011	.131	.044	**.367**	.087	.000	.000	.000	.000	**1.00**
.063	.113	.000	.000	.000	.056	.113	**1.00**	.113	.000	.153	.000	.000
.027	.000	**.586**	.131	.000	.000	**.678**	.012	**1.00**	.000	.239	.000	**1.00**
.022	**1.00**	.000	.000	.000	.000	.000	.000	**.167**	.000	.000	.000	.000
.021	**.339**	**1.00**	.000	**.339**	.169	**1.00**	**1.00**	**1.00**	**.339**	**.504**	**.339**	**1.00**
.018	.000	**.200**	.000	**.200**	.000	.000	**.200**	.000	.000	.000	**1.000**	.000
.012	.000	**1.00**	.000	.000	.000	**1.00**	.000	**1.00**	.000	**.629**	.000	**.457**
.007	.000	**1.00**	.000	**1.00**	**1.00**	**1.00**	.000	**1.00**	.000	**1.00**	.000	.500
.007	.000	.000	.000	.000	**1.00**	.000	.000	.000	**.500**	.500	**1.00**	.000
.004	.000	**1.00**	**1.00**	**1.00**	.000	.000	.000	**1.00**	.000	**1.00**	.000	**1.00**

Table 7.9: Model parameters learned by EM for the real door data using "event-periods" to create the data matrix **X**, and the number of mixture components set to 15. The row highlighted in green displays the event period results for the loading dock entrance stream, front door entrance stream relationship, $\{S2, S4\}$, which is much more prominent in the event period results than in the time segment results. The stream probabilities that are in **bold** typeface in the table indicate streams that are significant for each cluster.

stream events. For the experiment in this section, an event period is defined as a set of contiguous time-segments where there is event activity in at least one sensor stream. So an event period continues as long as at least one stream sees event activity.

With this alternate definition of event periods, the number of rows in our event data matrix **X** decreases from 581 time segments to 279 event periods. Running EM on this new data set with the number of clusters set to 15 gives the results displayed in Table 7.9.

In data sets with shorter length time segments, one would expect this alternate definition of event periods to be necessary in order to capture the entrance stream, exit stream relationships corresponding to traffic before and after events. In our data set, the relatively large 30-minute time-segment size is large enough that this definition for a time-period captures many of the {entrance event stream, exit event stream} relationships. The lengthy time segments combined with the Markov event process that encourages an event to have persistence, contributed to enough overlapping event activity for the relationship to be found. The only significant difference between the results using time segments and event periods is that the loading dock entrance stream, front door entrance stream relationship, $\{S2, S4\}$ (highlighted in green in Table 7.9), is much more prominent in the event period results. The event activity for these two streams (presumably from the people setting up the event, and then the people attending the event) does not overlap as often in the same time-segment, as compared to entrance and exit event activity for the same door. Perhaps the trickle of people arriving late for an event and leaving early is more prominent than one might expect.

7.7.2 Defining Event Clusters Using Multinomials

In the original algorithm, the event data **X** are made up of binary event variables x_{ts} representing two possible event states, a "positive event" state and a "no event" state. In this section, we generalize our algorithm to accommodate more than two event states. In Chapter 3, a three state model was described, having a "positive" event state, a "negative event" state, and a "no event" state. In the experiment in this section, the event data **X** is created using the three-state model. So now

$$x_{ts} = \begin{cases} 1 \text{ if there is positive event activity for stream s at time t} \\ 0 \text{ if there is no event activity for stream s at time t} \\ -1 \text{ if there is negative event activity for stream s at time t} \end{cases} \tag{7.37}$$

α_k	p_{k1}	p_{k2}	p_{k3}	p_{k4}	p_{k5}	p_{k6}	p_{k7}	p_{k8}	p_{k9}	p_{k10}	p_{k11}	p_{k12}
.255	.000	**1.00**	.018	.015	.000	.000	.010	**.636**	.024	.036	.000	.002
	.000	.000	.000	.000	.000	.000	.000	.006	.000	.000	.000	.000
.178	.009	.000	.017	.000	.010	.000	.000	**1.00**	.008	.022	.008	.000
	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
.159	.000	.021	.032	**.437**	.037	.033	.000	.054	.000	**.713**	.034	.027
	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
.156	.000	.177	.024	.031	.023	**.676**	.043	.211	.045	.021	.000	**.641**
	.000	.000	.000	.000	.000	.000	.000	.023	.000	.000	.000	.000
.101	.000	.000	.023	.000	.000	.016	.000	.000	.000	.000	.000	.017
	.000	**.843**	.000	.000	.000	.000	.000	**.751**	.000	.000	.000	.000
.078	.000	.000	**.596**	.017	.036	.037	.000	.000	**.391**	.000	.072	.000
	.000	.000	.000	.000	.000	.000	.000	.016	.000	.000	.000	.000
.034	**.770**	**.222**	.000	.000	**.175**	.000	**.274**	.000	.000	.000	**.135**	**.124**
	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
.031	.111	.000	.000	.000	.000	.050	**1.00**	.197	.000	.099	.000	.000
	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
.008	.000	**1.00**	.000	**1.00**	.000	**.401**	**.404**	**1.00**	**.205**	**.333**	**.205**	.000
	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 7.10: Multinomial model parameters learned by EM for the real door data, where the number of clusters is set to 9. Positive event probability parameters p_{ks}^{+1} are in the top row and negative event probabilities p_{ks}^{-1} are in the bottom row for each cluster. The highlighted cluster explains time segments with negative event activity, and the stream probabilities that are in **bold** typeface in the table indicate streams that are significant for each cluster.

Negative events are less frequent than positive events. While 581 of the 4368 time-segments in the thirteen weeks of our study contained positive events, only 69 time-segments contained negative events. Also, positive events were inferred in all 12 data streams, whereas negative events were only predicted for the Front door entrance ($S2$) and Front door exit ($S8$) streams.

When modeling binary data, an S-dimensional Bernoulli distribution was used to define a cluster. To accommodate ternary data, we use an S-dimensional multinomial distribution. So now instead of one parameter per cluster k and stream s, there are two:

$$\mathbf{p_{ks}} = \{p_{ks}^{+1}, p_{ks}^{-1}\} \tag{7.38}$$

where p_{ks}^{+1} and p_{ks}^{-1} are the probabilities of observing a positive and negative event respectively in stream s of cluster k, and where $(p_{ks}^{+1} + p_{ks}^{-1}) \leq 1$, and the probability of not seeing an event in stream s is $(1 - p_{ks}^{+1} - p_{ks}^{-1})$.

The EM algorithm proceeds similarly as in the original algorithm. The clustering results when setting the number of clusters to nine ($K = 9$) are displayed in Table 7.10. The top row for each cluster shows the positive event probability parameters p_{ks}^{+1}, and the negative event probabilities p_{ks}^{-1} are in the bottom row. The highlighted cluster explains time segments with negative event activity.

7.8 Summary of Contributions

- People counting sensors often form logical networks with other sensors, such as the building in Chapter 6 and the freeway in Chapter 5. We see the work in this chapter as a possible step towards more closely linking sensors in order to make inferences about multi-sensor events. We envision using the output of such a method to decide which sensors to link in a multi-sensor event model.

- Identified a bias in the current building occupancy model against event inference (i.e. the model will often adjust for a net positive or negative occupancy count at the end by removing counts from inferred events). In some cases, removing event inferences is reasonable, but the current model does not protect events that are supported by evidence in the form of corresponding event activity at other doors.

- Presented an EM algorithm for clustering sensor streams that have correlated event activity whose results could be used to counter the bias when event inferences are supported by additional event evidence from other streams.
- Demonstrated the use of the clustering algorithm with a case study that used real building sensor data.
- Analyzed the temporal patterns of the event clusters and suggested a potential future model extension that would add a temporal component to the model of the event process.
- Presented and demonstrated two generalizations of the clustering algorithm: discovering time-delayed event relationships between streams, and allowing event state variables to have more than two states by defining clusters as an S-dimensional mixture of multinomials.
- Created a detailed EM tutorial in the context of learning the parameters of a mixture of multi-dimensional Bernoulli distributions.

Chapter 8

Directions for Future Research on Event Modeling Problems

This chapter explores some of the potential future directions for extending the event model. Future work could include creating a multi-sensor event model that is more sensitive to events observed at multiple locations; this model could potentially be useful for sensor failure detection and missing measurement imputation. Other future work could include incorporating GIS data, such as the location of residential and industrial areas, into the prediction of flow patterns and potentially the estimation of dynamic population density. We also discuss potential future "humans as sensors" applications where instead of one sensor that measures and reports consistently, counting many people; there is one sensor for each person that sporadically reports a measurement.

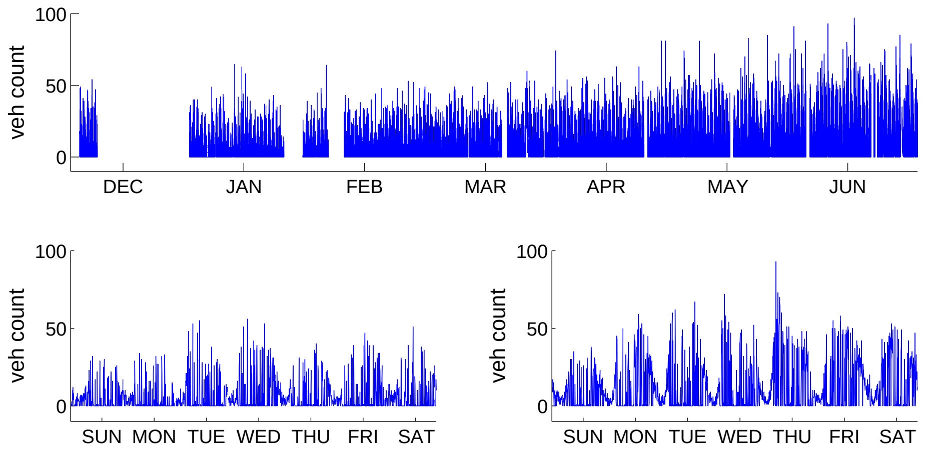


Figure 8.1: (top) Seven months of measurements for a sensor that appears to have a gradual increase of activity throughout the study period. (bottom) A week near the beginning and end - many zero measurements are present, but there may be some signal in the remaining measurements.

8.1 Nonstationary Behavior

This section discusses extensions to the event model to make it robust to nonstationary data sets. Here we define nonstationarity as a week-to-week change in the normal patterns of behavior, rather than the intra-week nonstationarity that is already handled by the NHPP in the current model. In the majority of data sets we examined we did not see significant nonstationarity, though we did see some examples. Nonstationarity may be a bigger factor in other people counting data sets, so it is worth considering how to extend the current model for this type of behavior.

Figures 8.1 and 8.2 show two examples of sensors that showed nonstationarity. Figure 8.1 shows a sensor that appears to have a gradual increase throughout the study and Figure 8.2 shows a sensor that appears to capture a bi-modal behavior.

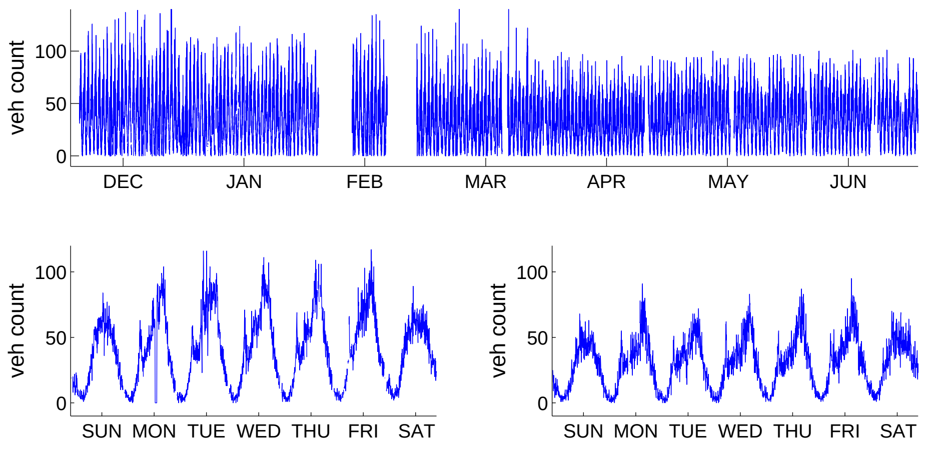


Figure 8.2: (top) Seven months of measurements for a sensor that has a bimodal normal pattern throughout the study period. (bottom) Two weeks with different normal patterns.

8.1.1 Gradual Increase or Decrease

If an increase or decrease in normal behavior is gradual enough, the current model may be sufficient. One could regularly retrain the model based on the past three to four months of measurements, for example. If the change is not gradual, as is the case in Figure 8.1, a model of normal behavior that does not adjust with time will not fit the data well. This less flexible model will frequently infer event activity, and the event fraction will probably fall above the threshold used to determine bad sensors as described in Chapter 4.2.2. We did not see many examples like Figure 8.1 in our studies, and in this case the rise may be due to sensor failure since it is accompanied by frequent fail-to-zero measurements (as illustrated in the two weeks in the bottom panel). Still, it is worth considering how to address this problem if it becomes more relevant in other sets. If the rate of increase or decrease is constant, a possible extension could be to include a multiplicative adjustment to the model that increases or decreases the normal rate with time, so instead of learning the rate parameters $\lambda(t)$, there would be an additional parameter to learn $\rho t \lambda(t)$.

8.1.2 Bimodal Behavior

A second example of nonstationary behavior is seen in Figure 8.2. This type of bimodal behavior was more common in the data sets we studied. The two weeks in the bottom panel show the two patterns that were common to the seven month period. Both patterns exhibit strong periodic behavior, but at significantly different magnitudes. The event model would be forced to pick one of these patterns as normal, and the other pattern would probably be inferred as event activity. A sensor such as this would also probably have a high event fraction and would be culled from the list of good sensors, even though some (if not all) of the measurements are probably accurate. This type of behavior could occur during periods of construction, when traffic in one lane or at one exit is shut down temporarily and traffic is then increased at other lanes or exits; or it could occur when a sensor measurement is an aggregate of two or more sensors, with one or more of these sensors sporadically failing for periods of time, for example. A possible model extension to address the problem of bimodal behavior would be to learn two normal patterns (or more if there are more than two such patterns) and have a switch variable that determines which pattern is in use at any given time. To avoid making the model too flexible, so that event activity could be explained by switching between the two normal patterns, it might be beneficial to make the switch variable apply to an entire day or week, forcing the entire day or week to maintain the same normal profile. Another possibility could be to look for a change point in the Poisson process similar to Raftery [60].

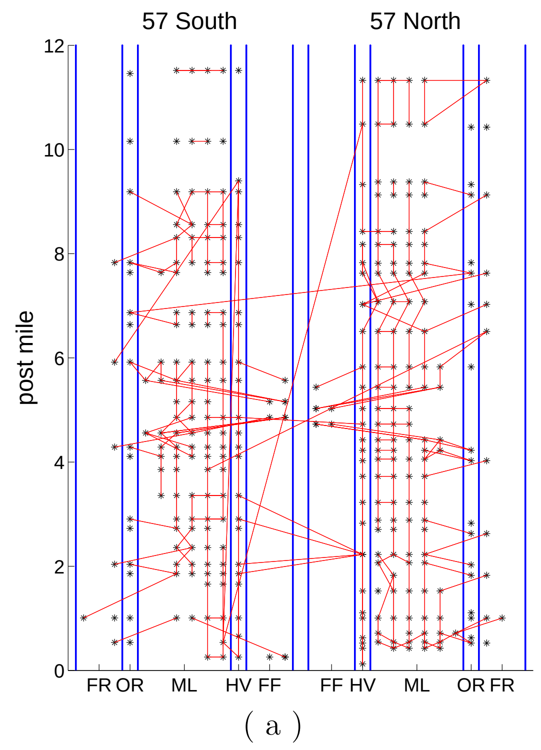
8.2 Multiple Sensor Models

Chapter 6 showed a multi-sensor building occupancy model, where individual sensor event models were linked via an occupancy node. In this section, a different kind of multi-sensor model, where the individual event models are linked via the event node, is discussed. At the beginning of Chapter 7, an illustration showed that while the building occupancy model linked sensors, the model did not make use of information about event activity between the sensors. If pairings or clusters of sensors that experience event activity at the same time can be learned, including this information into the model can make it more sensitive to events that are measured across several sensors. If, for example, there is a low amount of event activity measured at the same time at three different sensors, and if the sensors are not linked, the event activity might be seen as noise that could be attributed to the normal process. However, if we know these sensors tend to measure event activity at the same time, we have more evidence that an event is taking place rather than just noise.

The sensors we examined in this thesis tend to have highly correlated signals. For example the average traffic sensor in Los Angeles has a similar shape to the average sensor in San Diego. However, to impute missing values for a sensor in LA, measurements from neighboring sensors are usually more useful than distant sensors. In some cases, however, neighboring sensors may give misleading information due to sensor errors (particularly configuration errors).

While the normal pattern of behavior is highly correlated between people counting sensors, event activity is less correlated. For example, all of the doors to a building may have similar normal patterns: traffic beginning around 7AM, a burst of activity around lunchtime, and decreasing activity through the evening; however event activity for a seminar in an auditorium on the East side of the building may only be observed by the doors on the East side. So clustering sensors according to event activity such as in Chapter 7 is an appealing way to determine which sensors to link.

A simpler method than that discussed in Chapter 7 which may work well in practice is to find pairs of sensors with conditional dependence by thresholding the inverse



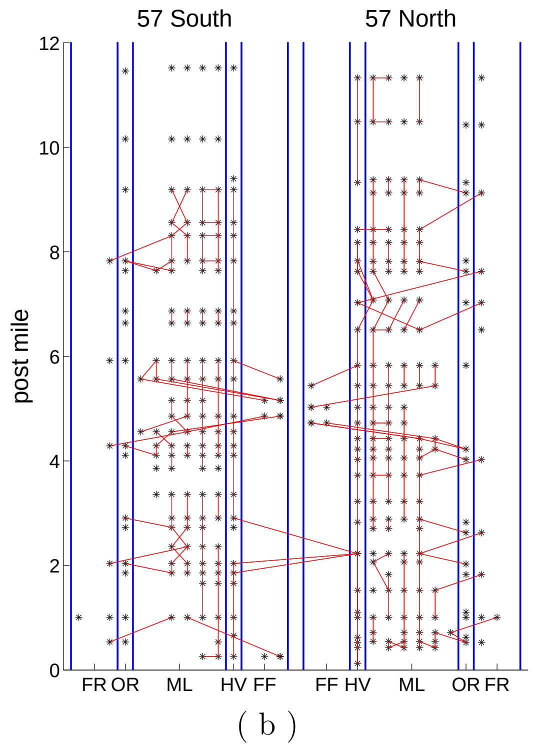
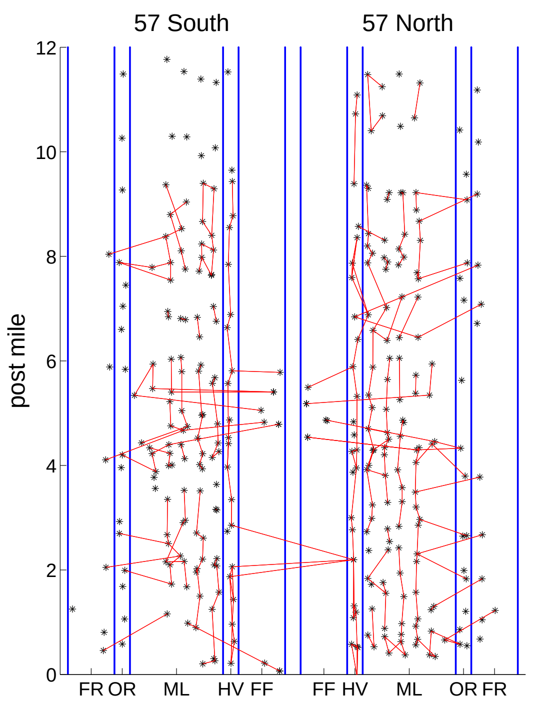


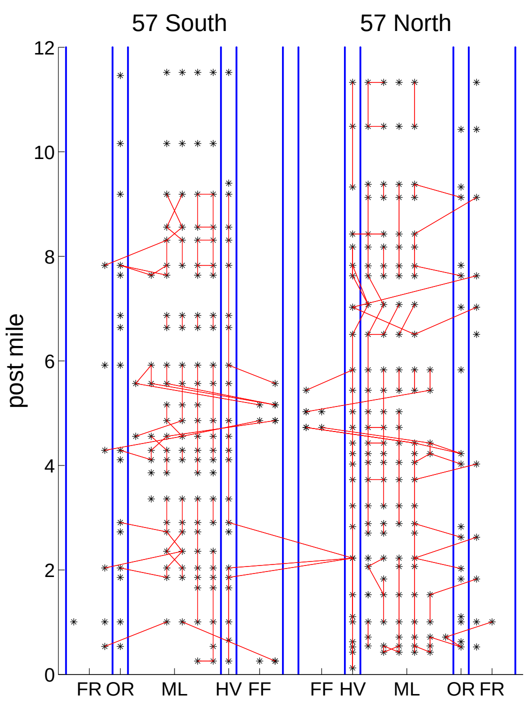
Figure 8.3: Two different threshold settings, to yield either (a) 2.1 links per node, or (b) 1.5 links per node.

covariance matrix given a history of observations. This method can be used to find a sparse set of links between the variables. Figure 8.3 shows the output of this method for approximately 300 sensors on the 57 Freeway in Orange County (measurements at 2-minute time intervals). The following is a legend for Figures 8.3 and 8.4:

- *: indicates a sensor location
- blue vertical lines: separates lane types
- red lines: links between sensors
- FR: Off Ramp; sensors between these blue lines are on off-ramps
- OR: On Ramp; sensors between these blue lines are on on-ramps
- ML: Main Line; between these blue lines are main line sensors
- HV: HOV; sensors between these blue lines are on HOV lanes



(a)



(b)

Figure 8.4: (a) Jittered plot shows that the links in (b) are mainly to nearest neighbors.

- FF: Freeway to Freeway; sensors between these blue lines are on freeway connectors

Questionable sensors with high event fractions or too few measurements were culled according to the method described in Chapter 4.3.1 before running the algorithm. Figure 8.3(a) shows the links found with a threshold that is probably a little too low (the further the normalized inverse covariance value is from zero, the stronger the conditional dependence), since there are too many links that are not intuitive (sensors linked that are far from each other and to sensors in different directions). Figure 8.3(b) shows the links found with an appropriate threshhold.

Figure 8.4(a) jitters the positions of the sensors in order to show that the links in (a) are between neighbors for the most part. Note that the method uses no location information and so there is nothing in the method to give preference for nearest neighbors. The off-ramps and on-ramps also show some strong dependencies with sensors upstream (for off-ramps) and downstream (for on-ramps) of the ramp sensors.

A model that links sensors spatially can be used in several applications such as finding failed sensors, imputation of missing measurements and finding the spatiotemporal signature of events.

In the freeway example, sensors that are spatially close to one another, but do not have a strong conditional dependence relationship are suspicious. Also, if all nearby sensors show event activity, but a sensor in the midst of this activity does not, there is reason for suspicion. The dependence plot in Figure 8.4 shows an example of some suspicious links. The sensor in the HOV lane for 57 northbound near postmile 2 has some unusual links to the HOV sensors in the southbound direction at neighboring post miles. Kwon [42] describes a configuration failure where a southbound sensor is mistakenly reporting as a northbound sensor, for example. One might expect such configuration errors to have links similar to the ones in our example.

Another possible application is the imputation of missing measurements. Main line sensors on freeways use imputation techniques to filter the sensor data. The current practice is described by the PeMS Data Extraction Methodology and Execution Technical Memorandum for the Southern California Association of Governments [76]:

> This algorithm imputes the missing values at a detector based on the surrounding neighbors. A neighbor is a loop in other freeway lanes at the same location as well as all of the loops at the locations immediately upstream and downstream of the current loop.
>
> - Linear Regression from Neighbors based on Global Coefficients: This algorithm is similar to the one above but in situations where loops never report data it is not possible to compute local regression co-

efficients. Therefore global regression coefficients that represent the general relationship seen throughout the district are used.

- Temporal Medians: If adjacent detectors do not report data, linear regression does not work. In this case if you have a hole at 3pm on Wednesday at a particular loop then you can look back over the past N Wednesdays at 3pm for this loop and take the median of those measurements.
- Cluster Medians: This is the last method of imputation. A cluster is a group of detector stations that have the same macroscopic flow and occupancy patterns over the course of a typical week. A station is assigned to a cluster and any holes are assigned to the cluster ID. If all of the other imputation methods fail then the quantity profiles for the cluster are pulled out of the database and the correct value for this day of the week and time of day is assigned.

Potentially valuable alterations to this method include:

- Using a more selective linking methodology such as found in Chapter 7 or the inverse covariance method described earlier in this section. Spatial neighbors do not always seem to provide helpful information and a more restrictive neighbor list might prove helpful.

- Linking sensors with a time delay. Some of the sensors in the northern part of Figure 8.4 do not show any dependence links. This could be due to the short time segment length. Sensors that are half of a mile apart and in the same lane have a greater chance of seeing the same vehicles pass over them in a two minute time segment than sensors such as these that are a mile and a half apart. The inverse covariance method described earlier could be modified to consider pairwise dependencies between sensors at different time shift (for example measurements at one sensor at time t are compared with measurements from another sensor at time $t-1$). The method could be modified to only consider sensors within a fixed radius of a sensor and in the same direction to increase computational efficiency.

- Using the output of the event model instead of the historical mean, when nearby neighbors can not provide helpful information.
- Using a hybrid model that uses a combination of the normal pattern found by the model and information from the nearest sensors.
- Considering a separate model for ramp sensors and freeway-freeway connectors. The current method does not impute values for these ramps, but we found some surprisingly strong dependencies between main line sensors and ramp sensors, and perhaps combining this information with the event model output could provide a reasonable imputation.

Another possible application of a multi-sensor event model would be to provide a more principled approach to the spatiotemporal event characterization problem described in Chapter 5.3.2.

8.3 Dynamic Occupancy Model

One of the motivating examples we gave for studying people counting sensors was their potential use in the dynamic occupancy estimation problem. The current method for determining the number of people in an area affected by a natural disaster [34], for example, is based on census data and does not make use of dynamic information such as is provided by these sensors. Using information from loop detectors on freeway

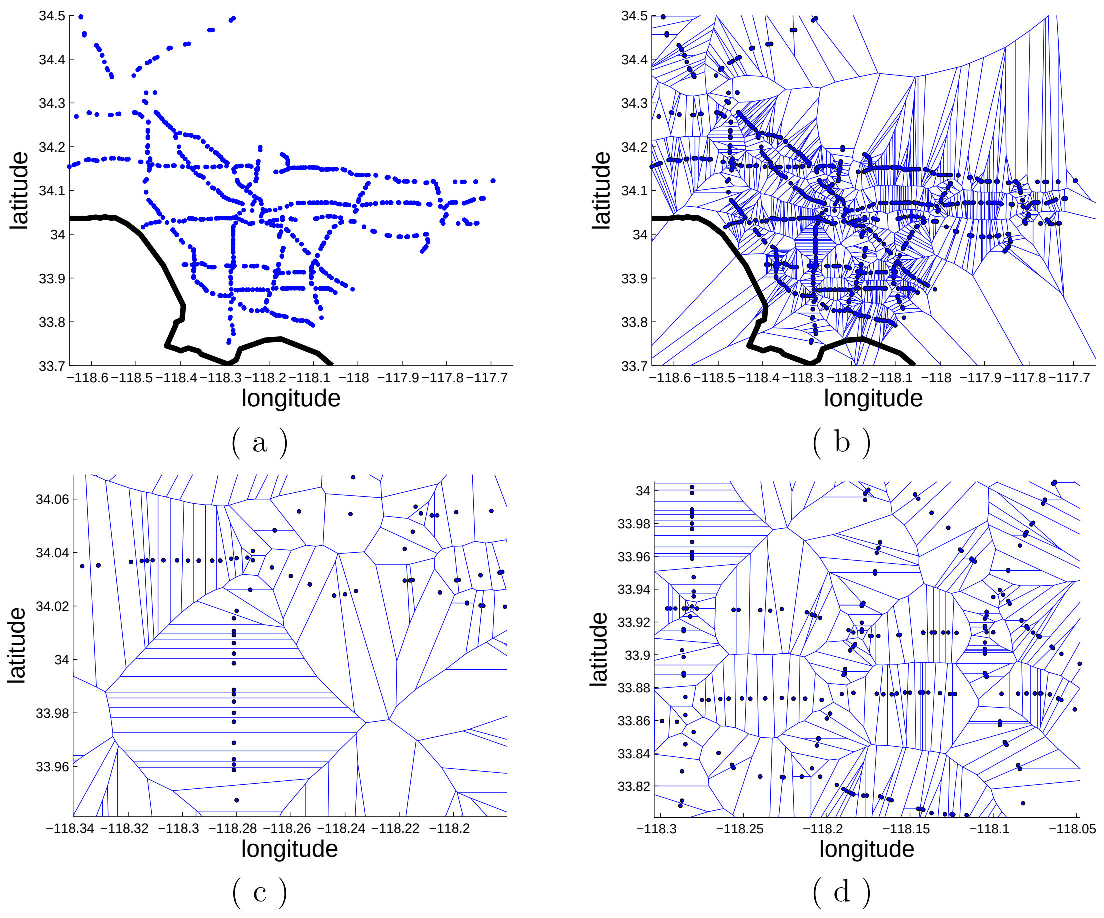


Figure 8.5: (a) freeway ramp locations used in our study, including locations with "bad" sensors. Each location has one or more on- and off-ramps. The dark thick line is the Pacific Ocean coastline near Los Angeles and Orange County. (b) The Voronoi diagram segmenting the study area in (a) around the ramp locations. (c) and (d) show two zoomed-in areas of plot (b).

ramps alone could give a rough dynamic estimation of commuter population. A more accurate estimation is a very challenging task, but this thesis has taken several steps towards addressing these challenges. The remainder of this section discusses some of the challenges, the progress we have made, and what we have learned along the way.

The area surrounding the ramp sensors was segmented using a Voronoi diagram [2] shown in Figure 8.5, and the population density of each area was increased as vehicles were measured using an off ramp and decreased as vehicles were measured using an on ramp. In our studies, missing measurements occurred approximately 30% of the

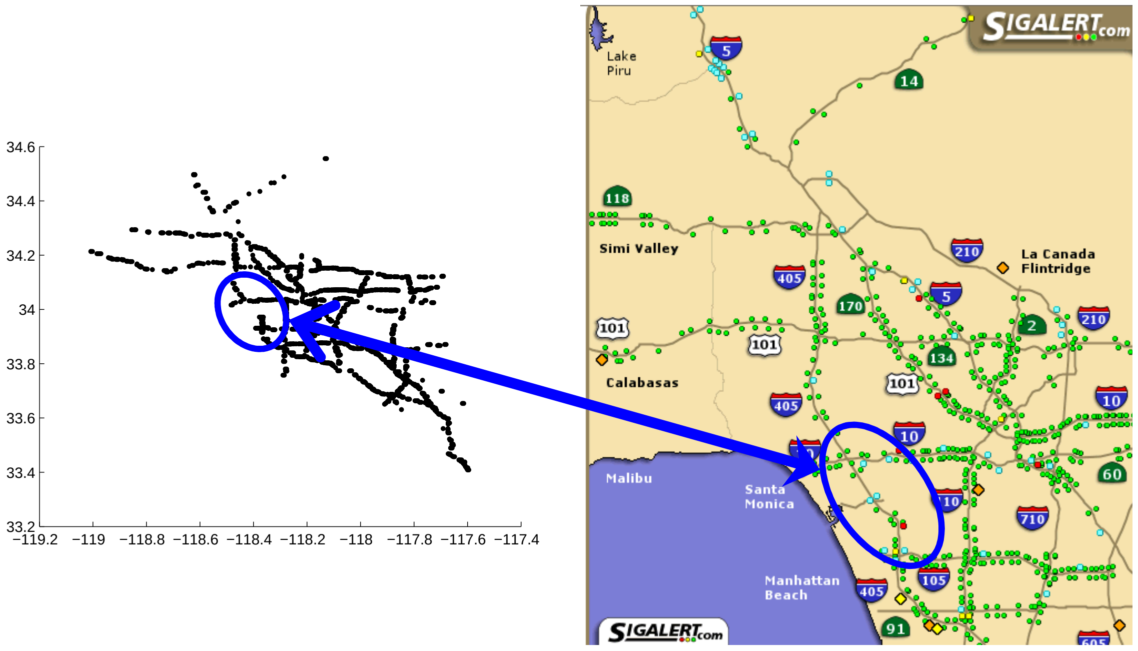


Figure 8.6: (left) A map showing the locations of “good” ramp sensors used in our study. (right) The Sigalert [69] map for the same area shows that the gap where no good sensors were present in our study also had an abnormally low number of working mainline sensors two years after the study period.

time. To address this, we imputed values for missing measurements using the normal flow process parameters $\lambda(t)$ learned by the event model.

A second challenge was imputing values for “bad” sensors. Approximately 30% of the ramps in our study area of Los Angeles and Orange County either did not report enough measurements to model (less than 2 weeks of measurements during the nine month study), did not have a periodic pattern that could be found by the event model in the history, or did not have a sensor present. These failures were generally spread out evenly across the study area, except for a ten mile stretch of the 405 freeway in Los Angeles as shown in Figure 8.6. Figure 8.6 shows the locations of the “good” sensors used in our study on the left side of the figure, and the right side of the figure shows a SigAlert map [69] that displays real-time traffic conditions. The SigAlert map shows that most of the main line sensors were not reporting measurements in the same ten mile gap where the ramps were not reporting. This is a stretch of highway near LAX Airport and is relatively dense with on- and off-ramps which made this section particularly problematic for our city level dynamic population estimation problem.

To address the problem of “bad” sensors, we attempted to impute measurements based on the mean ramp profile.

After running a proof of concept experiment, it became evident that the mean ramp profile replacement of bad sensors was not sufficient.

8.3.1 Ramp Sensor Flow Profile Characterization

Further examination of ramp profiles revealed a large variety of shapes. Figure 8.7 shows the weekday profile shapes for two ramps that are quite different in shape. The ramp shapes are shown along with a satellite image and a heat map of 2000 census information showing concentrations of households in the area surrounding the sensor. For these two example sensors, one can see that the land use has an intuitive influence on the profile shape of the corresponding on-ramp; Figure 8.7(a) is an on-ramp in a residential area, and has a large morning peak as people get on the freeway to go to work, but Figure 8.7(a) shows an on-ramp sensor in an industrial area, where relatively few vehicles are seen getting on the freeway in the morning, but a large number of vehicles get on the freeway in the evening as people leave work.

So instead of using a mean ramp profile to predict ramp use in cases where no model of normal use is available, we attempted to use information about land use in the area surrounding the sensor to predict a more accurate flow profile.

Using Census Data to Predict Ramp Profile Shape

Our goal is to create a dynamic population density model using ramp information, but the many ramps without a history of measurements create gaps that our model needs to address. Figure 8.7 illustrates that land use information might be useful for the task

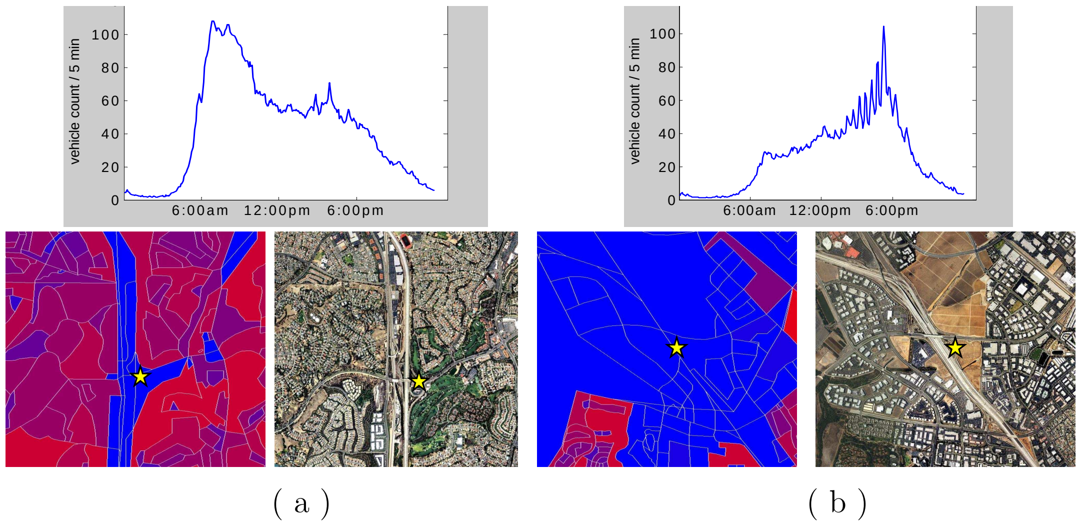


Figure 8.7: Profile Shape of Freeway On-Ramps Reflect Land Use. A plot of the underlying normal vehicle count pattern for a weekday for a loop detector at an on-ramp in (a) a residential area and (b) an industrial area. The ramp profiles are plotted along with a satellite image [1] of the area surrounding the sensor and a census map [27] of the surrounding area (red areas have high residential household density, blue area have low household density). The star marker indicates the position of the sensor on the maps.

of predicting ramp profile shapes, which could then be used in our population density model. In this section we discuss our first effort for predicting ramp profile shapes using land use information (information about the number of households surrounding the ramp provided by the 2000 census). On-ramps and off-ramps are separated in this study, and weekday profiles also need to be predicted separately from weekend profiles. The census map for the area surrounding a sensor, such as seen in Figure 8.7, divides the area into blocks colored with one of five different colors. The color of the block gives an indication of the number of households in that block, red indicates the category with the highest number of households and blue indicates the category with the lowest number of households.

Figure 8.8 illustrates the problem set up. The census maps provide the features that will be used to predict the on-ramp profile shapes. Specifically, the fraction of the map covered by each color (as seen on the left side of Figure 8.8) will provide the input to the algorithm which will be used to predict the weekday shapes of individual

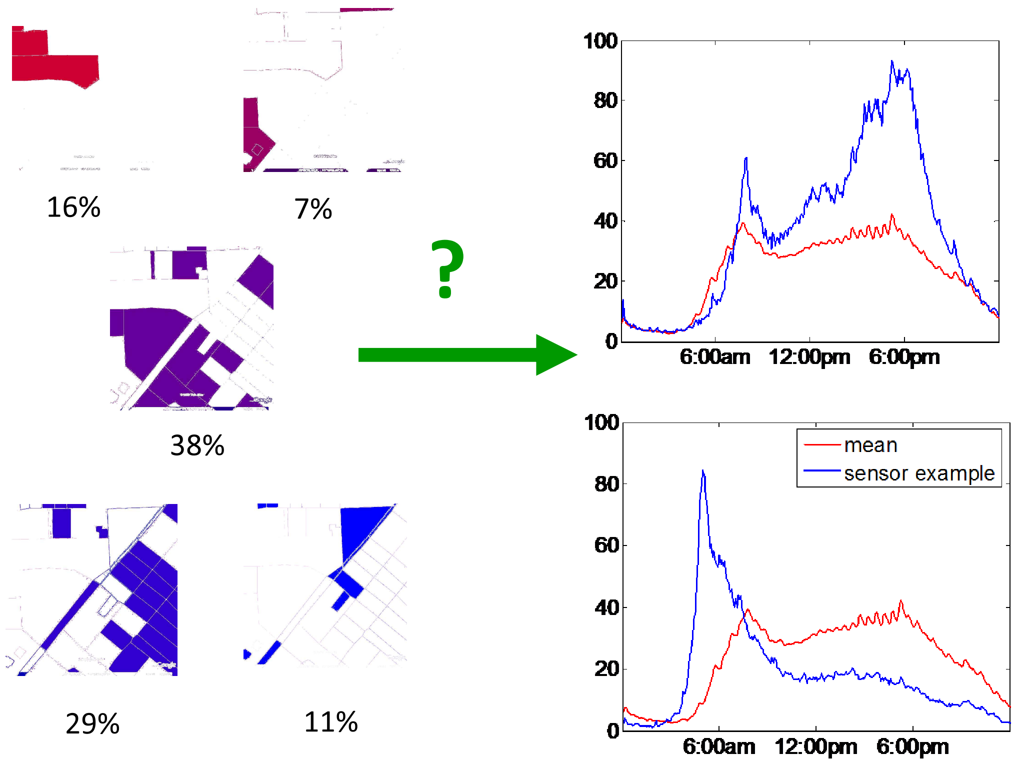


Figure 8.8: (left) The features used to predict the weekday on-ramp profiles are the fraction of the census map of the area surrounding the sensor that is covered by the 5 different census categories indicating the number of households (blue indicating the category with the fewest number of households, red indicating the category with the highest number of households). (right) Weekday profiles for two on-ramps are plotted in blue, and the mean on-ramp weekday profile is plotted in red.

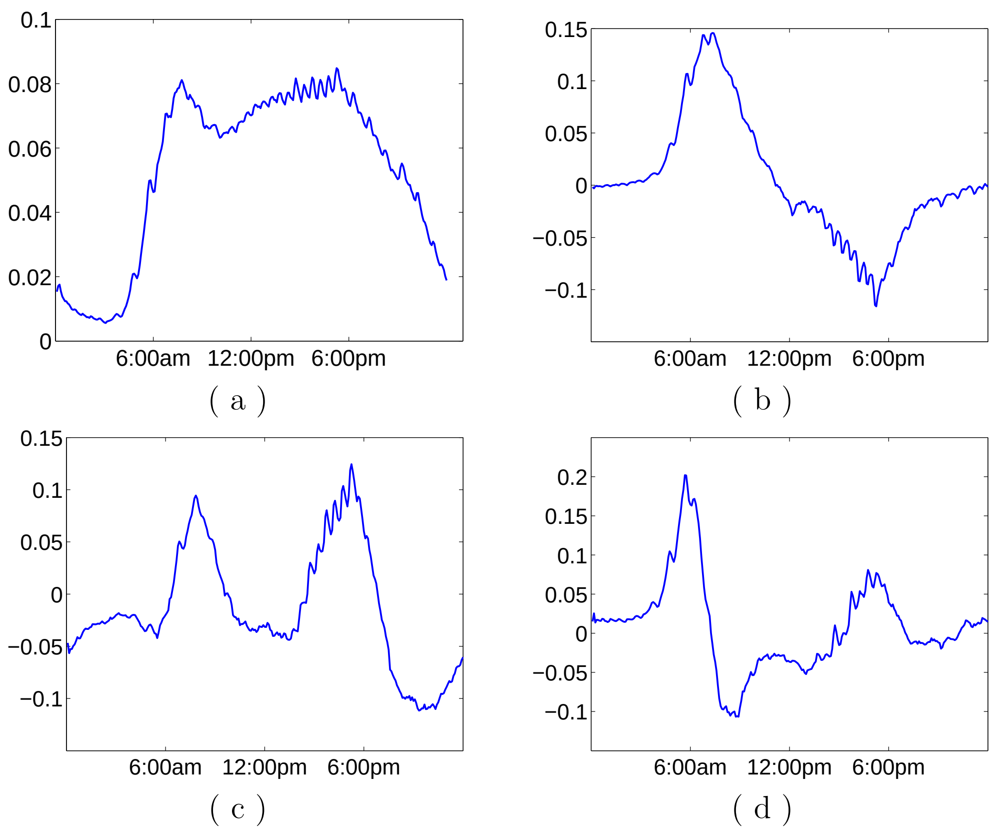


Figure 8.9: The first four principal components for weekday on-ramp profiles.

on-ramps (two examples are shown in blue on the right hand side of Figure 8.8). The output of the algorithm will be compared with the mean weekday profile shape (shown in red in the plots on the right hand side of Figure 8.8).

The ramp profile consists of 288 points, one for each 5-minute time segment. We used PCA (principal component analysis) to reduce the number of dimensions, simplifying the problem and to avoid overfitting. 855 on-ramps in LA and Orange County were used in our study. We found that the majority ($> 95\%$) of the variation from the mean profile was captured by the first five principal components. Rather than using 288 points, then, we learn five coefficients that can be used to represent the shape. An approximate reconstruction of the original ramp shape can be found using the following equation

$$x_i \approx \bar{x} + \alpha_{1i}V_1 + \alpha_{2i}V_2 + \alpha_{3i}V_3 + \alpha_{4i}V_4 + \alpha_{5i}V_5 \tag{8.1}$$

where x_i is the 288 dimensional vector for the original ramp profile, $\bar{x}$ is the mean ramp profile, V_k is the k'th principal component, and α_{ki} is the k'th coefficient for ramp i.

Figure 8.9 shows the first four principal components for weekday on-ramp profiles. The first component, Figure 8.9(a), is a similar shape to the mean profile. This component provides a scaling effect; ramps with much larger than average flow have a high coefficient value multiplied to component, and ramps with a much smaller than average flow have a negative coefficient value for this component. The second component, Figure 8.9(b), provides an asymmetry effect; the ramp in the industrial area, Figure 8.7(b) has a negative value for the coefficient multiplied to this component. The third component, Figure 8.9(b), describes variation in the peakedness of the rush hour activity; and the fourth component, Figure 8.9(d), describes variations in how early the morning rush hour begins.

Figure 8.10 shows how a ramp profile can be reconstructed given the coefficients and the principal components. The figure shows six pairs of plots. The first plot in the pair is always the original, 288 dimensional ramp profile. The second plot in each pair is the approximate shape using a different number of principal components. As more principal components are used, the ramp looks more like the original shape.

Our task is now to use the census features to predict coefficients for the first five principal components. So instead of predicting 288 points, we are predicting values for the five α coefficients.

We used linear regression for predicting the ramp profile using the census features. For each of the 855 ramps in the study, we learned the model using the other 854

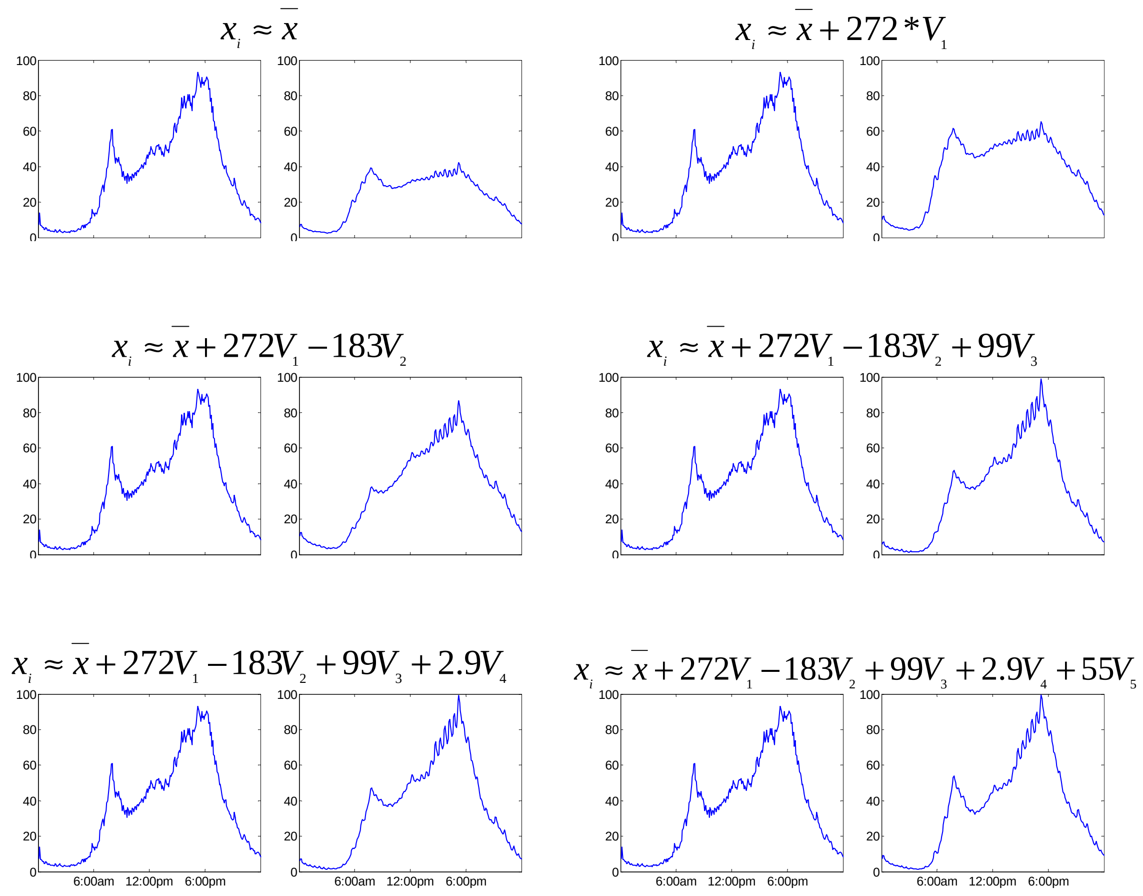


Figure 8.10: Diagram showing the reconstruction of a ramp profile using the principal components.

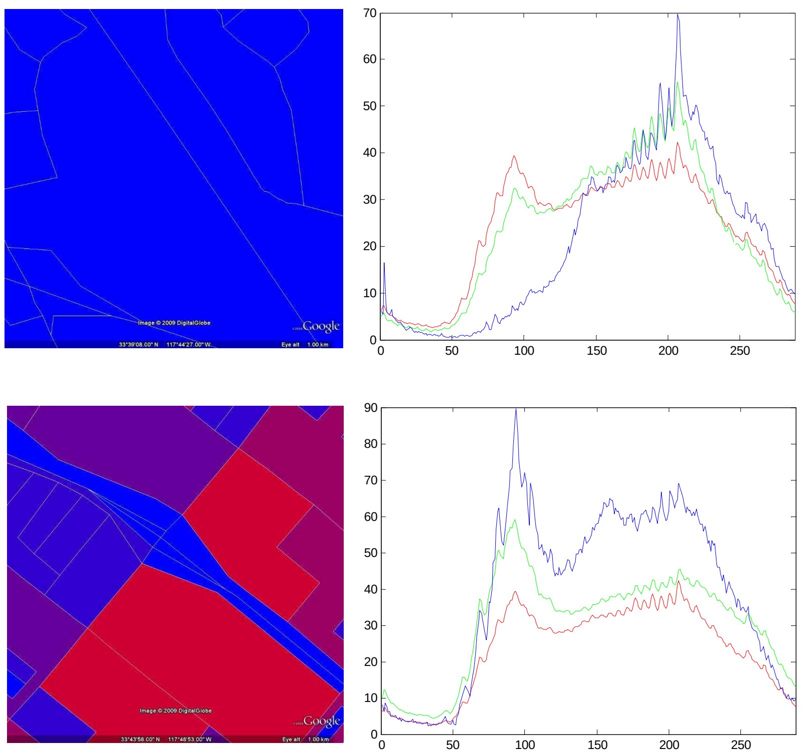


Figure 8.11: Two examples of weekday on-ramp profiles learned using census features. The blue line is the true ramp profile, the green line is the predicted ramp profile, and the red line is the mean ramp profile (our baseline).

ramps to predict the ramp left out (i.e. the ramp left out is treated as if it were a bad sensor with no reliable information contained in its history of measurements). Figure 8.11 shows the results for two of the ramps. The blue line is the ramp profile we are trying to predict, the red line is the baseline given by the mean ramp profile, and the green line is the predicted ramp profile. In these two cases, the predicted profile is significantly closer in shape to the original profile than the baseline.

Although these two examples show improvement over the baseline, the improvement overall was negligible. Still, the results show some promise. The weekday profile learned for on-ramps in an industrial area (top panel of Figure 8.11) has the characteristic shape we would expect (low morning peak, higher afternoon peak), and the profile learned for on-ramps in residential areas (bottom panel of Figure 8.11) also have an intuitive shape.

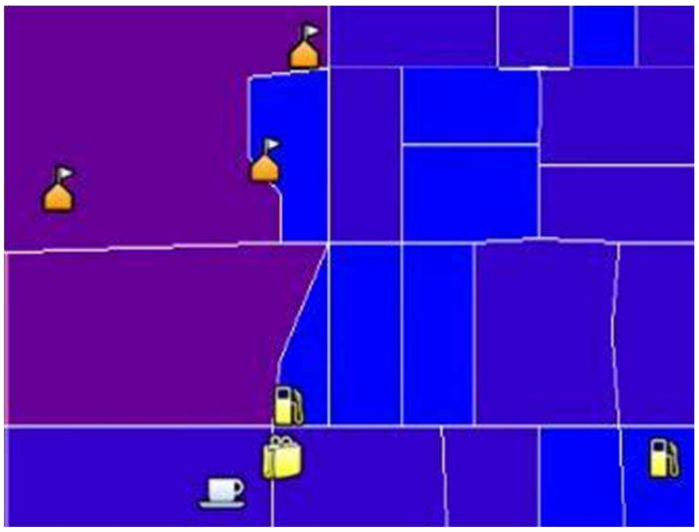
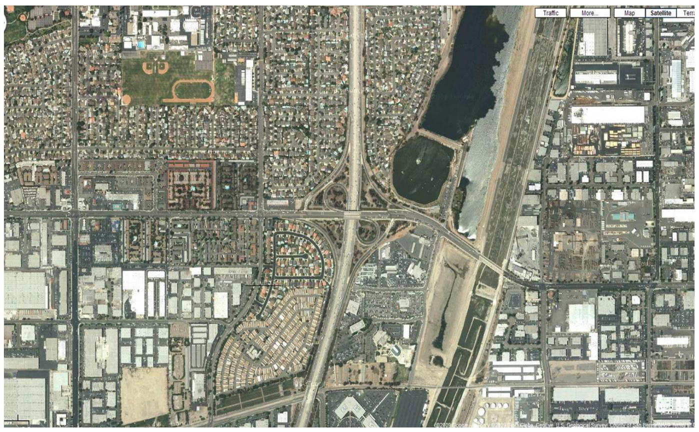


Figure 8.12: Landmark information. The locations of coffee shops, schools, gas stations, and retail stores pictured here are among the landmark features that can be obtained from Google Earth [1].

Additional Features

While census data alone may not provide enough information to predict ramp profiles where there is not a reliable history of measurements available, incorporating other features can improve the prediction. After examining individual ramps where the prediction based on census data was not an improvement over the baseline, we discovered that there are a large number of variables that influence the shape the flow profiles.

Landmark information provided one set of features that was useful for predicting ramp profiles. Figure 8.12 shows an example of some of the landmark information that is available from web mapping services such as Google Earth [1]. The locations of coffee shops, schools, gas stations, and retail stores are pictured in the figure for a small area. Other landmark features include airports, bars, dining, grocery, hospitals, lodging and malls. These features can be added to the ramp prediction model by finding the number of each type of landmark in the area surrounding the ramp.



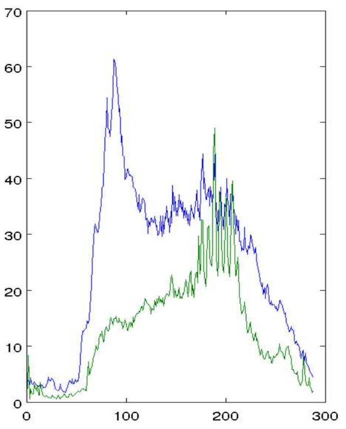


Figure 8.13: The interchange in the middle of the satellite image on the left has two on-ramps for entering the North Freeway. The flow profile for the on-ramp that is fed by traffic approaching from the largely industrial region to the east of the ramp is in green in the plot on the right; and the flow profile for the on-ramp that is fed by traffic approaching from the largely residential region to the west of the ramp is in blue.

One problem with using the landmark information with the census information is that the census data is from 2000, and the landmark information is from 2010. To address this, we hand labeled land use information (% industrial, % residential, % undeveloped) for the areas around each of the 855 ramps using satellite images from 2007. Combining landmark information with this land use information resulted in a significant improvement in the ramp prediction problem. Using a leave-one-out cross validation experiment, and evaluating the prediction using sum-of-squared error (SSE), the prediction reduced the SSE by 50% compared to the baseline error.

Another important feature for many locations is the direction of flow on the freeway. At the same interchange for a north-south freeway for example, there is often one or two freeway on-ramps for the north direction and one or two more on-ramps for the south direction. One direction is often more popular than the other given the time of day (into or out of the city for example), which can cause significantly different profile shapes for ramps in the same location with the same surrounding census information.

While this is an important feature, it is hard to predict. We had some success using information from neighboring sensors, however.

In addition to freeway direction, the direction that vehicles approach the freeway can also be influential on the profile shape at some locations. For example, Figure 8.13 shows a satellite image of an interchange for a north/south freeway in Orange County. To the east of the freeway is a largely industrial area, but to the west of the freeway is a largely residential area. Vehicles approaching from the east will generally use a different set of on-ramps than those approaching from the west. The green line shows the flow profile for the north on-ramp for vehicles approaching from the east. This shape is characteristic of what we have come to expect from on-ramp traffic in an industrial area. The blue line shows the flow profile for the north on-ramp for traffic approaching from the west, and the shape is characteristic of on-ramp traffic in a residential area.

Other influential features we have identified include the number of lanes of the arterial road that feeds the ramps and the spatial location. We clustered ramp profile shapes using a k-means algorithm and found the ramps in each cluster were often in proximity to each other.

We did not pursue the dynamic population density problem further, but it is an interesting problem that warrants further attention, particularly as the 2010 census information becomes available, and as new data sets such as described in the next section become available.

8.4 Cell Phone Data, "Humans as Sensors" in Continuous Space

The "humans as sensors" concept [50] involves making use of measurements from wearable sensors. Cell phones are probably the most common device that could fit into this category. Within the next year, AT&T will make cell call location information (to the nearest cell tower) publicly available [61]. And starting in 2012, the FCC will require all US wire carriers to provide emergency responders with the latitude and longitude, accurate to within 300 meters (984 feet), of a 911 caller. As stated in a wireless 911 services consumer report by the FCC [18]:

> within six minutes of a valid request by a PSAP, provide more precise location information to PSAPs; specifically, the latitude and longitude of the caller. This information must be accurate to within 50 to 300 meters depending on the type of technology used.

While it is not clear that all of this information will be available for research, it is worth considering the relevance to the problems presented in this thesis, especially since some of the applications are in the context of emergency response.

A first issue to address is privacy. One option for adding anonymity to the data is jittering the location data for each call location. A second method would be to create a grid of the urban area and aggregate the calls that fall into each cell. There are a several benefits to this second method for use with our models. The first option would result in a continuous space, but it is hard to see how to incorporate continuous positions into the current model. In the second method each cell could be thought of as a supersensor, and the calls within a fixed time length could be aggregated to provide input to our models without needing modification. Also, the grid structure provides a natural way to link these supersensors for use in a multi-sensor event model.

As with the ramp profile detection problem, however, demographics will play an important role if this data were to be used in the dynamic population density application. For example, the number of text messages that originate from a location where a high school is present might lead to an overestimation of the number of people at that location.

8.5 Conclusion

In the future, human activity sensors such as those studied in this thesis will likely have a significantly greater presence in our daily life, and there will be a greater need for algorithms such as those presented in this thesis to make sense of the raw data and draw out information of value. We have shown that models such as ours that simultaneously separate event and normal data are important for many applications such as estimating the severity of abnormal activity. In addition, we have demonstrated the utility of the model in single sensor and multiple sensor applications, and have shown how to extend the model for use for specific applications such as the loop detector model. As we look to the future we see the framework discussed in this thesis as useful for more challenging large scale problems such as estimating the dynamic population density of an urban area.

Bibliography

[1] Google earth. http://earth.google.com/.

[2] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[3] S. Basu and M. Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2):137–154, 2007.

[4] L. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[5] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[6] M. Bebbington. Identifying volcanic regimes using hidden Markov models. *Geophysical Journal International*, 171(2):921–942, 2007.

[7] T. Belin and D. Rubin. The analysis of repeated-measures data on schizophrenic reaction times using mixture models. *Statistics in Medicine*, 14(8):747–768, 1995.

[8] D. Belomestny, V. Jentsch, and M. Schreckenberg. Completion and continuation of nonlinear traffic time series: a probabilistic approach. *Journal of Physics A: Mathematical and General*, 36:11369, 2003.

[9] P. Bickel, C. Chen, J. Kwon, J. Rice, E. van Zwet, and P. Varaiya. Measuring traffic. *Statistical Science*, 22(4):581–597, 2007.

[10] C. Bishop et al. *Pattern Recognition and Machine Learning*. Springer New York:, 2006.

[11] W. Buntine. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[12] B. Carlin and T. Louis. Bayes and empirical Bayes methods for data analysis. *Statistics and Computing*, 7(2):153–154, 1997.

[13] C. Chatfield. *The analysis of time series: an introduction*. CRC press, 2004.

[14] C. Chen, J. Kwon, J. Rice, A. Skabardonis, and P. Varaiya. Detecting errors and imputing missing data for single-loop surveillance systems. *Transportation Research Record*, 1855:160–167, 2003.

[15] C. Cornell. Engineering seismic risk analysis. *Bulletin of the Seismological Society of America*, 58(5):1583, 1968.

[16] R. Dechter. Constraint networks. *Encyclopedia of Artificial Intelligence*, 1:276–285, 1992.

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

[18] Federal Communications Commission (FCC). Wireless 911 Services. http://www.fcc.gov/cgb/consumerfacts/wireless911srvc.html.

[19] G. Florez-Larrahondo, S. Bridges, and R. Vaughn. Efficient modeling of discrete events for anomaly detection using hidden Markov models. *Information Security*, pages 506–514, 2005.

[20] Freeway Performance Measurement System (PeMS). http://pems.eecs.berkeley.edu/.

[21] A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990.

[22] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. CRC Press, 2004.

[23] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. 6(6):721–741, Nov. 1984.

[24] W. Gilks, W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC, 1996.

[25] P. Green. Bayesian reconstructions from emission tomography data using a modified EM algorithm. *IEEE transactions on medical imaging*, 9(1):84–93, 1990.

[26] V. Guralnik and J. Srivastava. Event detection from time series data. In *KDD '99: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 33–42, New York, NY, USA, 1999. ACM Press.

[27] I. Haque. gcensus. http://gecensus.stanford.edu/gcensus/index.html.

[28] D. Hawkins. *Identification of outliers*. Chapman and Hall London, 1980.

[29] H. Heffes and D. Lucantoni. A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *Selected Areas in Communications, IEEE Journal on*, 4(6):856–868, 2002.

[30] HowStuffWorks. Induction-loop traffic sensors. http://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/red

[31] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–264, 1996.

[32] C. Huyck, H. Chung, S. Cho, M. Mio, S. Ghosh, R. Eguchi, and S. Mehrotra. Loss estimation online using INLET (Inter-based Loss Estimation Tool). In *Proceedings of the 8th US National Conference on Earthquake Engineering, San Francisco*, volume 1535, 2006.

[33] A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying Poisson processes. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 207–216, 2006.

[34] ImageCat. Risk Management Technologies. http://www.imagecatinc.com/.

[35] Institute of Transportation Studies (ITS). UC, Irvine. http://www.its.uci.edu/.

[36] L. N. Jacobson, N. L. Nihan, and J. D. Bender. Detecting erroneous loop detector data in a freeway traffic management system. *Transportation Research Record*, 1287:151–166, 1990.

[37] Z. Jia, C. Chen, B. Coifman, and P. Varaiya. The PeMS algorithms for accurate, real-time estimates of g-factors and speeds from single-loop detectors. In *Intelligent Transportation Systems*, pages 536–541. IEEE, 2002.

[38] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[39] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of 2009 SIAM International Conference on Data Mining (SDM2009)*, pages 389–400, 2009.

[40] E. Keogh, S. Lonardi, and B. Y. chi' Chiu. Finding surprising patterns in a time series database in linear time and space. In *KDD '02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 550–556, New York, NY, USA, 2002. ACM Press.

[41] J. Kleinberg. Bursty and hierarchical structure in streams. In *KDD '02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 91–101, New York, NY, USA, 2002. ACM Press.

[42] J. Kwon, C. Chen, and P. Varaiya. Statistical methods for detecting spatial configuration errors in traffic surveillance sensors. *Transportation Research Record: Journal of the Transportation Research Board*, 1870(-1):124–132, 2004.

[43] Li, S. Automatic inference of anomalous events from (California) traffic patterns. *Summer Undergraduate Research Fellowship in Information Technology (SURF-IT)*, 2006.

[44] H. Liu, W. Recker, and A. Chen. Uncovering the contribution of travel time reliability to dynamic route choice using real-time loop data. *Transportation Research Part A: Policy and Practice*, 38(6):435–453, 2004.

[45] W. W. LLC. Directional people counting sensors. http://peoplecounter.walkerwirelessco.com/.

[46] D. MacKay. Introduction to Monte Carlo Methods. *Learning in Graphical Models*, pages 175–204, 1998.

[47] O. Masoud and N. Papanikolopoulos. A novel method for tracking and counting pedestrians in real-time using a single camera. *IEEE Transactions on Vehicular Technology*, 50(5):1267–1278, 2001.

[48] R. Mateescu, R. Dechter, and K. Kask. Tree approximation for belief updating. In *Proceedings of the National Conference on Artificial Intelligence*, pages 553–559. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[49] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley New York, 1997.

[50] S. Mehrotra, C. Butts, D. Kalashnikov, N. Venkatasubramanian, K. Altintas, R. Hariharan, H. Lee, Y. Ma, A. Myers, J. Wickramasuriya, et al. CAMAS: a citizen awareness system for crisis mitigation. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, page 956. ACM, 2004.

[51] S. Mehrotra, C. Butts, D. Kalashnikov, N. Venkatasubramanian, R. Rao, G. Chockalingam, R. Eguchi, B. Adams, and C. Huyck. Project RESCUE: challenges in responding to the unexpected. *SPIE Journal of Electronic Imaging, Displays, and Medical Imaging*, 5304:179–192, 2004.

[52] A. Moore. Hidden Markov model tutorial. http://www.autonlab.org/tutorials/hmm.html.

[53] N. Navaroli. UCI Datalab.

[54] E. ONeill, V. Kostakos, T. Kindberg, A. Schiek, A. Penn, D. Fraser, and T. Jones. Instrumenting the city: Developing methods for observing and understanding the digital cityscape. *UbiComp 2006: Ubiquitous Computing*, pages 315–332, 2006.

[55] M. Papageorgiou and A. Kotsialos. Freeway ramp metering: An overview. *IEEE Transactions on Intelligent Transportation Systems*, 3(4):271–281, 2002.

[56] A. Papoulis, S. Pillai, and S. Unnikrishna. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill New York, 2002.

[57] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[58] Y. Qiao, X. Xin, Y. Bin, and S. Ge. Anomaly intrusion detection method based on HMM. *Electronics Letters*, 38(13):663–664, 2002.

[59] L. Rabiner. A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[60] A. Raftery and V. Akman. Bayesian analysis of a Poisson process with a change-point. *Biometrika*, pages 85–89, 1986.

[61] Ramaswamy Hariharan. AT&T, personal communication, 2010.

[62] W. Recker, Y. Chung, T. Golob, C. D. of Transportation, B. I. o. T. S. University of California, P. for Advanced Transit, and H. (Calif.). *A Tool for the Incorporation of Non-Recurrent Congestion Costs of Freeway Accidents in Performance Management*. California PATH Program, Institute of Transportation Studies, University of California at Berkeley, 2005.

[63] W. Recker, J. Marca, and C. Rindt. Institute of Transportation Studies, UC Irvine, personal communication.

[64] M. Salmenkivi and H. Mannila. Using Markov chain Monte Carlo and dynamic programming for event sequence data. *Knowledge and Information Systems*, 7(3):267–288, 2005.

[65] S. Scott. *Bayesian Methods and Extensions for the Two State Markov Modulated Poisson Process*. PhD thesis, Harvard University, 1998.

[66] S. Scott. A Bayesian paradigm for designing intrusion detection systems. *Computational statistics & data analysis*, 45(1):69–83, 2004.

[67] S. Scott and P. Smyth. The Markov modulated Poisson process and Markov Poisson cascade with applications to web traffic data. *Bayesian Statistics*, 7:671–680, 2003.

[68] S. Shah-Heydari and T. Le-Ngoc. MMPP models for multimedia traffic. *Telecommunication Systems*, 15(3):273–293, 2000.

[69] Sigalert. Southern California traffic information. http://www.sigalert.com/Map.asp.

[70] T. Singliar. *Machine Learning Tools for Transportation Networks*. PhD thesis, Department of Computer Science, University of Pittsburgh, 2008.

[71] T. Šingliar and M. Hauskrecht. Modeling Highway Traffic Volumes. In *Proceedings of the 18th European conference on Machine Learning*, pages 732–739. Springer-Verlag, 2007.

[72] P. Smyth. Probabilistic learning tutorial. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2005.

[73] T. Snowsill, F. Nicart, M. Stefani, T. De Bie, and N. Cristianini. Finding surprising patterns in textual data streams. In *Proceedings of the 2010 IAPR Workshop on Cognitive Information Processing (CIP2010), Elba Island, Italy*. IEEE Press, 2010.

[74] Å. Svensson. On a goodness-of-fit test for multiplicative Poisson models. *The Annals of Statistics*, 9(4):697–704, 1981.

[75] Thibaux, R. Personal communication, 2009.

[76] Urban Crossroads, Inc. PeMS Data extraction methodology and execution technical memorandum for the Southern California Association of Governments. 2006. http://www.scag.ca.gov/modeling/pdf/Pems_Technical_Memorandum_Final.pdf.

[77] A. Viterbi et al. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[78] R. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice hall, 1989.

[79] C. Wren, D. Minnen, and S. Rao. Similarity-based analysis for large networks of ultra-low resolution sensors. *Pattern Recognition*, 39(10):1918–1931, 2006.

Appendices

A Implementation Details

This appendix lists the parameter settings and other information needed to implement each of the models described in the thesis.

A.1 Two-State Positive Event Model

The model for the building senor stream had a different set of parameters than the loop detector sensor stream, due to the differences in the time segment length (30 minutes versus 5 minutes) and the magnitudes of the normal signal.

Building Sensor Stream Parameters

We used Beta prior distributions for the rows of the Markov transition matrix for the event process Z:

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} = \begin{pmatrix} .99 & .01 \\ .5 & .5 \end{pmatrix} \times 10^4$$

These values are consistent with the duration and frequency we expect for event activity. The strengths of the priors are relative to the data size. Strong priors are used for the event transition matrix parameters to avoid over-explanation of the data, such as using the event process to compensate for the fact that the "normal" data exhibits slightly larger than expected variance for Poisson data.

The event count was modeled with a negative binomial distribution:

$$e_t \sim \mathrm{NBin}(e_t; a^E = 30, b^E = 0.33))$$

Loop Detector Sensor Stream Parameters

We used Beta prior distributions for the rows of the Markov transition matrix for the event process Z:

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} = \begin{pmatrix} .999 & .001 \\ .2 & .8 \end{pmatrix} \times 10^4$$

The event count was modeled with a negative binomial distribution;

$$e_t \sim \mathrm{NBin}(e_t; a^E = 5, b^E = 0.33)$$

A.2 Negative Event Model

We used Dirichlet priors for the rows of the Markov transition matrix for the event process Z:

$$\begin{pmatrix} a_{00}^Z & a_{01}^Z & a_{02}^Z \\ a_{10}^Z & a_{11}^Z & a_{12}^Z \\ a_{20}^Z & a_{21}^Z & a_{22}^Z \end{pmatrix} = \begin{pmatrix} .99 & .005 & .005 \\ .195 & .8 & .005 \\ .195 & .0005 & .8 \end{pmatrix} \times 10^4$$

The event count was modeled with a negative binomial distribution

$$e_t \sim \mathrm{NBin}(e_t; a^E = 5, b^E = 0.33)$$

A.3 Fault Tolerant Model

The Poisson rate parameters have prior distributions

$$\lambda_{i,j} \sim \Gamma(\lambda; a_{i,j}^L = 0.05, b_{i,j}^L = 0.01)$$

where i takes on values $\{1, \ldots, 7\}$ indicating the day of the week and j indicates the time-of-day interval $\{1, \ldots, 288\}$.

We used Dirichlet priors for the rows of the Markov transition matrix for the event process Z:

$$\begin{pmatrix} a_{00}^Z & a_{01}^Z & a_{02}^Z \\ a_{10}^Z & a_{11}^Z & a_{12}^Z \\ a_{20}^Z & a_{21}^Z & a_{22}^Z \end{pmatrix} = \begin{pmatrix} .999 & .0005 & .0005 \\ .14 & .85 & .01 \\ .14 & .01 & .85 \end{pmatrix} \times 10^6$$

The Beta parameters for the transition matrix of the fault process (F) were:

$$\begin{pmatrix} a_0^F & b_0^F \\ a_1^F & b_0^F \end{pmatrix} = \begin{pmatrix} .00005 & .99995 \\ .0005 & .9995 \end{pmatrix} \times 10^6$$

The event count was modeled with a negative binomial distribution

$$e_t \sim \text{NBin}(e_t; a^E = 5, b^E = 0.33)$$

A.4 Transportation Model

The Poisson rate parameters had priors set using an empirical Bayes approach

$$\lambda_{i,j} \sim \Gamma(\lambda; a_{i,j}^L = \bar{o}_{i,j}/3, b_{i,j}^L = 0.33)$$

where i takes on values $\{1, \ldots, 7\}$ indicating the day of the week and j indicates the time-of-day interval $\{1, \ldots, 288\}$, and where $\bar{o}_{i,j}$ is the mean of the non-missing observed values at that time of day and day of week. This procedure was used for both the flow $\lambda^f(t)$ and occupancy $\lambda^o(t)$ processes.

We used Dirichlet priors for the rows of the Markov transition matrix for the event process Z:

$$\begin{pmatrix}
0.9979 & 0.0004 & 0.0004 & 0.0004 & 0.0004 & 0.0004 & 0.0004 & 0.00005 \\
0.0200 & 0.9778 & 0.0004 & 0.0004 & 0.0004 & 0.0004 & 0.0004 & 0.0004 \\
0.0200 & 0.0004 & 0.9778 & 0.0004 & 0.0004 & 0.0004 & 0.0004 & 0.0004 \\
0.0200 & 0.0004 & 0.0004 & 0.9778 & 0.0004 & 0.0004 & 0.0004 & 0.0004 \\
0.0200 & 0.0004 & 0.0004 & 0.0004 & 0.9778 & 0.0004 & 0.0004 & 0.0004 \\
0.0200 & 0.0004 & 0.0004 & 0.0004 & 0.0004 & 0.9778 & 0.0004 & 0.0004 \\
0.0200 & 0.0004 & 0.0004 & 0.0004 & 0.0004 & 0.0004 & 0.9778 & 0.0004 \\
0.00005 & 0.000001 & 0.000001 & 0.000001 & 0.000001 & 0.000001 & 0.0000 & .99995
\end{pmatrix} \times 10^6$$

where the states are defined as follows:

$$z_t = \begin{cases}
0 & \text{if there is no event at time } t \\
1 & \text{if there is a high flow, high occupancy event at time } t \\
2 & \text{if there is a high flow, low occupancy event at time } t \\
3 & \text{if there is a low flow, low occupancy event at time } t \\
4 & \text{if there is a low flow, high occupancy event at time } t \\
5 & \text{if there is a high flow, normal occupancy event at time } t \\
6 & \text{if there is a normal flow, high occupancy event at time } t \\
7 & \text{if there is a sensor failure at time } t \\
\textit{unused} & \text{if there is a low flow, normal occupancy event at time } t \\
\textit{unused} & \text{if there is a normal flow, low occupancy event at time } t
\end{cases}$$

The event count for the flow process was modeled with a negative binomial distribution

$$e_t^f \sim \text{NBin}(e_t^f; a^E = 14/4, b^E = 1/4)$$

The event count for the occupancy process was modeled with a mixture of two negative binomial distributions

$$\begin{aligned} e_t^o \sim & (w_1 = 0.8) * \text{NBin}(e_t^o; a_1^o = 15/4, b_1^o = 1/4) \\ & + (w_2 = .2) * \text{NBin}(e_t^o; a_2^o = 45, b_2^o = 1) \end{aligned}$$

A.5 Building Occupancy Model

We used Dirichlet priors for the rows of the Markov transition matrix for the event process Z:

$$\begin{pmatrix} a_{00}^Z & a_{01}^Z & a_{02}^Z \\ a_{10}^Z & a_{11}^Z & a_{12}^Z \\ a_{20}^Z & a_{21}^Z & a_{22}^Z \end{pmatrix} = \begin{pmatrix} .98 & .01 & .01 \\ .395 & .6 & .005 \\ .395 & .005 & .6 \end{pmatrix} \times 10^4$$

The event count was modeled with a negative binomial distribution

$$e_t \sim \text{NBin}(e_t; a^E = 5, b^E = 0.33)$$