Nonparametric Bayesian Behavior Modeling

by

Joshua Mason Joseph

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author .....
Department of Aeronautics and Astronautics
May 30, 2008

Certified by .....
Nicholas Roy
Assistant Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Adam Rzepniewski
Senior Member, Technical Staff, C. S. Draper Laboratory
Thesis Supervisor

Accepted by .....
David Darmofal
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Nonparametric Bayesian Behavior Modeling

by

Joshua Mason Joseph

Submitted to the Department of Aeronautics and Astronautics
on June 2, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

As autonomous robots are increasingly used in complex, dynamic environments, it is crucial that the dynamic elements are modeled accurately. However, it is often difficult to generate good models due to either a lack of domain understanding or the domain being intractably large. In many domains, even defining the size of the model can be a challenge. While methods exist to cluster data of dynamic agents into common motion patterns, or "behaviors," assumptions of the number of expected behaviors must be made. This assumption can cause clustering processes to under-fit or over-fit the training data. In a poorly understood domain, knowing the number of expected behaviors *a priori* is unrealistic and in an extremely large domain, correctly fitting the training data is difficult. To overcome these obstacles, this thesis takes a Bayesian approach and applies a Dirichlet process (DP) prior over behaviors, which uses experience to reduce the likelihood of over-fitting or under-fitting the model complexity. Additionally, the DP maintains a probability mass associated with a novel behavior and can address countably infinite behaviors. This learning technique is applied to modeling agents driving in an urban setting. The learned DP-based driver behavior model is first demonstrated on a simulated city. Building on successful simulation results, the methodology is applied to GPS data of taxis driving around Boston. Accurate prediction of future vehicle behavior from the model is shown in both domains.

Thesis Supervisor: Nicholas Roy
Title: Assistant Professor of Aeronautics and Astronautics

Thesis Supervisor: Adam Rzepniewski
Title: Senior Member, Technical Staff, C. S. Draper Laboratory

Acknowledgments

Thanks to...

Nick for his guidance, whose depth of knowledge and understanding is truly amazing.

Adam for continually pushing me to show him results and stay on my timeline.

The Charles Stark Draper Laboratory for its support and the seemingly limitless knowledge amongst its staff.

Paul for the discussions about what will surely be a bestseller, if we ever get around to writing it.

Ben for the all the arguments about the most important topic of them all: philosophy.

Kristian whose help and ideas, all the way from Germany, were fantastic.

My brother, uncles, aunts, cousins, and friends for their understanding that the recent lack of communication is by no means a reflection of the value I place on our relationship.

My mom for her patience and dedication to what is best for me (as difficult as I make it).

My dad, who is a real life example of living up to what you think.

Sara Jane, whose infectious love of life ensures that I will never have a normal day.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

On November 3^{rd}, 2007, the DARPA Urban Challenge was held. This was a competition of fully autonomous vehicles racing in an urban environment while obeying all traffic laws (i.e., not only stopping at stop signs but maintaining intersection precedence). While this was arguably one of the most impressive feats of autonomous robots to date, it also demonstrated how difficult it is to replace or even mimic humans. This is illustrated by an event that occurred during the race between MIT's and Cornell's vehicles. Cornell's vehicle came to a stop in the middle of a road and MIT's vehicle, approaching from behind, decided to pass. As MIT's vehicle completed the pass and pulled back into the lane, Cornell's vehicle began to move forward, causing a minor collision. If either vehicle had understood the behavior of the other, this, potentially, could have been avoided. This incident is a shining example of how autonomous robots need to understand the behaviors of the objects with which they interact, in order to truly succeed in complex domains.

As autonomous robots are increasingly used in a variety of complex domains, behavior modeling stands out as being extraordinarily useful. It should be no surprise that humans have become particularly adept at creating models of objects with which they must interact. The ability to predict an object's response to different actions allows humans to achieve desired outcomes.

1.2 Problem Statement

The attention of this thesis is on behavior modeling in domains that are poorly understood and extremely large. Motivated by the Urban Challenge, the problem of modeling the behaviors of cars driving in a city environment is the chosen application of this work. A successful driver behavior model will need to solve two problems: predicting the destination of vehicles and the routes chosen to these destinations.

The first step of a solution is to model vehicle destinations. Building on the way humans typically understand destinations, this thesis defines a destination as a location that assigns a probability to nearby road segments. This probability represents how likely drivers are to park there while heading to the destination. An example of a destination, and something that appears in the results section, is an airport. This is not only a logical destination to model, but knowing a driver is heading to the airport does not exactly specify the road on which the driver parks. What it does mean is the driver will tend to park close to the airport, with this "closeness" measured by a probability distribution (see Section 2.2 for a proper definition of a probability distribution). This destination representation attempts to model how varied the parking locations are of drivers heading to a destination. Compared to a destination that is a residential area, it is expected that an airport would have a much "tighter" distribution. The problems this presents are finding the number of destinations, their central locations, and the probabilities of each road segment near a central location being a stopping point for that destination.

Once the destination model has been created, the next step is to understand how drivers travel to these destinations. When routes are discussed it is sufficient to think of them as a sequence of road segments that lead to an ending location near the vehicle's desired destination. A simple approach is to learn the probabilities of cars turning or going straight at every intersection by counting how frequently they happen in training data and making predictions based on that. While there are some disadvantages to this, which are discussed later, the central problem is that making predictions this way ignores the past. The intuition behind this is seen by attempting

to predict a destination given the following two premises:

There is a vehicle on the highway.

There is a vehicle on the highway coming from the airport.

The second statement would make prediction easier and contains additional information that the simple approach, described previously, cannot capture. Building on this intuition, a successful driver behavior model needs a more general approach: one that can learn the probability of a driver taking every possible path. Although it may seem necessary to have nearly infinite training data to learn the nearly infinite possible paths through a city, a solution requiring that much training data is unreasonable. To overcome this, the solution must "understand" the domain is extremely large and be able to reason about paths not seen in the training data. When a path is not seen in the training data, this does not mean there is no information about it. What is known about unseen paths is how frequently new paths appeared in the training data. Based on the size of the domain, the amount of training data, and the number of unique paths, the model should be able to predict about how likely an unseen path is to occur.

A central assumption of this thesis is that the behavioral domain is too large and poorly understood to have a designer provide a model for either destinations or paths to each destination. While it is obvious to designate the airport as a destination, it is unrealistic to expect a designer to know all the destinations of all drivers. The paths drivers choose may be subject to factors that are difficult to model (e.g., ignorance of a "better" path, driver's mood, etc). To overcome these challenges, the model will need to be learned from training data.

1.3 Approach

1.3.1 Finding Destinations

To capture the selection of parking locations around a destination, each destination is represented as a Gaussian distribution (Section A.1). Building on this, the entire

city is represented as a mixture of Gaussians, where each destination Gaussian also has a weight associated with it. This weight corresponds to how likely a destination is for a car, so the entire city is a combination of these weighted Gaussians.

Learning models of destinations is made difficult by an unknown number of destinations. To overcome the challenge of unknown model size, the destinations are assumed to be distributed according to a Dirichlet process (DP, Section 2.4.2). By assuming the destinations were generated from this process, it is possible to infer the most likely number of destinations based on how frequently new destinations are seen in the training data. This is in opposition to typical methods that either require the number of model components to be explicitly specified beforehand or a penalty placed on the number used. To be able to specify the number of destinations or a penalty, a fair amount of designer understanding is required, an assumption that this thesis wishes to avoid.

1.3.2 Learning the Probability of Every Path

Section 1.2 briefly discussed a simple approach to learning the probabilities of paths by finding the frequency that different actions at an intersection happen and the intuition this provides for needing to learn the probability of every path. To do this naively places a parameter on each path (representing its probability), in total, requiring nearly an infinite number of them.

To better understand how many parameters are needed, consider the following illustration. Figure 1-1 shows four copies of a 5x5 block city, each with a driver's possible route drawn in green. Even if loops are not allowed, there are an extremely large number of paths traversing even a small city. By knowing the probability of each possible path, predictions about a vehicle's next action or end location can be made. As an example, assume the bottom right path occurred far more frequently in the training data than the others. If this were the case, it is then expected for the route to have a significantly higher likelihood of occurring. At the same time, to naively learn the probabilities associated with infinitely many paths, infinite data is needed. Figure 1-2 shows three possible infinite probability distributions over paths

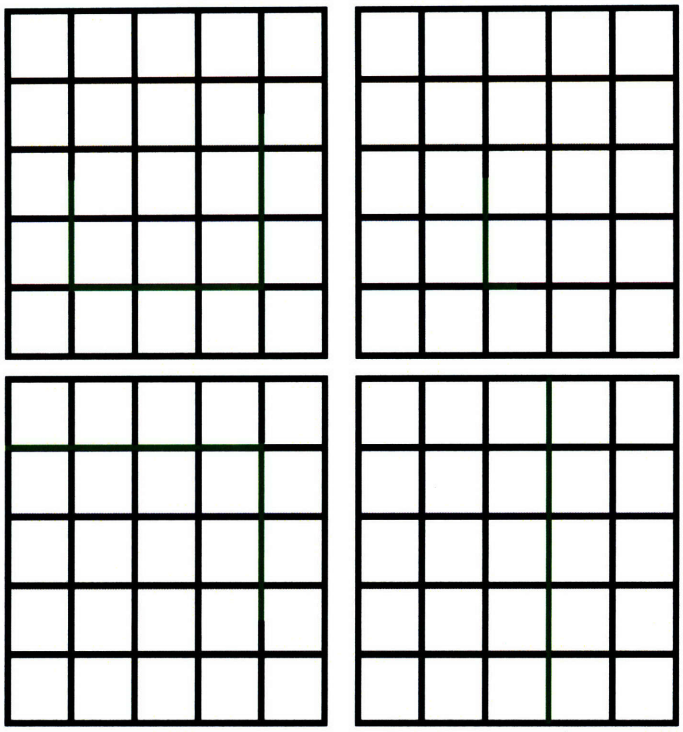

Figure 1-1: Four possible routes traversing a 5x5 block city.

where each path is indexed and each index correspond to a bar with probability proportional to its height. To overcome requiring infinite data, it is assumed that these paths are generated from a DP. While DPs are also used in Section 1.3.1, its justification here is quite different. A DP ranks the kind of probability distributions shown in Figure 1-2 by how well they fit the training data. Using this approach, it is not necessary to have infinite data to capture the probability of each path. This DP ranking is based on the number of paths that have been seen and how frequently they occur.

Consider another explain regarding Figure 1-1. Of 100 observed paths of data the paths shown in the figure were the only ones seen (i.e., all 100 cars took one of the four paths). The DP based on these 100 observed paths would say some distributions (a and b, Figure 1-2) are more likely (rank them higher) than other distributions (c, Figure 1-2). By having this ranking on path distributions, it is possible to sum over all distributions, enabling predictions about paths that both appear and do not appear in the training data. A DP has a parameter α (that can be learned from data) which

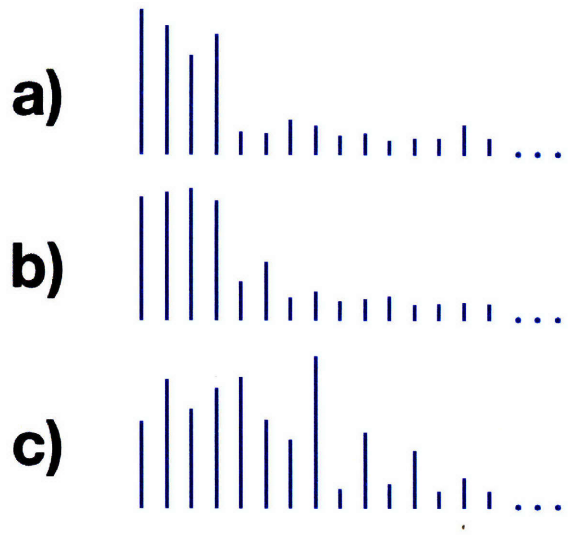

Figure 1-2: Three possible distributions over all paths.

is approximately proportional to the probability that the next path will be novel. For example, if ten paths of data were collected resulting in the training data {1, 2, 2, 2, 1, 1, 2, 1, 2, 2} (where each unique path is given a number), α would be small since the training data leads one to believe new occurrences happen relatively infrequently. Conversely, if the data had been {1, 2, 3, 4, 5, 3, 6, 7, 8, 9}, α would be large.

With the understanding that different types of destinations have differing degrees of path variety, it is undesirable for a destination with a high variety of paths to influence the distribution over paths to a destination with low variety. By modeling driver behavior as having a destination with each having its own distribution over paths, the model can learn how varied the paths are to each destination without being influenced by others.

The high degree of flexibility provided by DPs make one wonder if perhaps the model sounds too good and must require an unattainable amount of data. This question is actually part of a far bigger issue: how well a model "fits" the training data. This is a fundamental issue that is addressed in this thesis and the argument is made in Chapter 4 that one of the strongest elements of a DP is its ability to appropriately fit the training data through the use of a probability associated with a novel path appearing as the next path.

1.4 Contribution

The main contribution of this thesis is to present a powerful tool for behavior modeling. This thesis asserts that DPs are ideally suited to driver modeling. A DP's ability to learn the number of behaviors from the training data, handle countably infinite behaviors, and maintain a probability of a novel behavior occurring makes a strong case for its use here. While DPs have gained increased attention more recently, they have not yet been discussed in the broad terms of behavior modeling. It is the hope of this thesis to show that the sole reason DPs have not yet been used to address behavior modeling is due to their relatively recent appearance in common literature.

Since the applications of DPs are fairly new, most focus almost exclusively on the benefits and do not address the drawbacks. This thesis also attempts to address the two main disadvantages of DPs: the amount of training data needed and the slower learning time than conventional methods. Even with these drawbacks, DPs have the potential to solve many behavior modeling problems that typical approaches cannot address due to large, poorly understood domains. In Chapter 4, the DP-based driver behavior model is demonstrated in both simulation and on real data.

1.5 Thesis Overview

The main purpose of Chapter 2 is to clarify DPs to the reader and facilitate understanding of how they differ from typical approaches. Chapter 3 introduces the model of a vehicle in an urban environment. The methodology developed is then tested in Chapter 4, both in simulation and on real GPS data of taxis driving around Boston. Chapter 5 offers some concluding remarks and future work.

Chapter 2

Background

2.1 Chapter Overview

This chapter begins with a brief introduction to probability theory in Section 2.2. Building on the foundation set in the Section 2.2, Section 2.3 discusses Bayesian inference. Section 2.4 introduces stochastic processes, and the process that is the basis of this thesis, the Dirichlet process in Section 2.4.2. Section 2.5 describes past approaches to modeling the behaviors of drivers.

2.2 Probability Theory

It is sometimes difficult to say with certainty that an event will always or never occur, and in these cases, discussing the event in terms of a degree of belief is necessary. For example, it is rare to be able to say a driver will always turn left at an intersection, but to say the driver will turn left 90% of the time is typically much more reasonable. In probability theory, this is denoted P(Left Turn) = 0.9. Probability theory allows us to capture this degree of belief as a probability distribution. A probability distribution is a function that, for the entire space Θ (the set consisting of all possible events), obeys

1. $\mathrm{P}(A) \geq 0$, for all events $A \subset \Theta$,

2. $\mathrm{P}(A_1 \cup A_2 \cup \ldots) = \mathrm{P}(A_1) \cup \mathrm{P}(A_2) \cup \cdots$, for disjoint[1], events $A_i \subset \Theta$

3. $\mathrm{P}(\Theta) = 1$,

which are referred to as the axioms of probability [7]. A probability distribution is a function from subsets of Θ to the real numbers.

Rather than describing distributions of an event A, it becomes more useful to think of this event as a variable, X. This is not an ordinary variable, it is a variable that has a probability associated with it taking each value in Θ. Since this distribution of $X \in \Theta$ is specified by a distribution, X is thought of as "randomly" taking one of these values. Due to this randomness, it is called a random variable.

There are two kinds of random variables, discrete and continuous. A discrete random variable is defined by a probability mass function (PMF), $p_X : X \to \mathbb{R}$, that gives the probability of X taking an exact value in Θ. The notation $p_X(x)$ represents the probability mass function for the random variable X at x. For example, if X is a random variable describing the probabilities of a vehicle going left, straight, or right at an intersection, its PMF may be written as

$$p_X(x) = \begin{cases} 1/6 & \text{if } x = \text{Left} \\ 1/2 & \text{if } x = \text{Straight} \\ 1/3 & \text{if } x = \text{Right} \\ 0 & \text{otherwise} \end{cases}.$$

So, to compute the probability of the vehicle turning, the probability "masses" can be added such that $\mathrm{P}(\text{Left} \cup \text{Right}) = 1/2$.

The second kind of random variable, a continuous random variable, is defined by a probability density function (PDF), $f_X : X \to \mathbb{R}$. Unlike in PMFs, where probabilities are given for the random value taking an exact value, the probability of a continuous random variable taking on an exact value is zero because the probability

[1] sets that do not share any elements

of choosing any one value out of infinitely many values is zero. The PDF gives the *relative* probability of different values. Probabilities for continuous random variables are only meaningful over a range of values of X. Consider a simple random variable representing the speed of a car where the speed is uniformly distributed between zero and 70. The PDF of this random variable, X, is

$$f_X(x) = \begin{cases} 1/70 & \text{if } 0 \leq x \leq 70 \\ 0 & \text{otherwise} \end{cases}.$$

Note that for continuous random variable X, f_X is used to represent its PDF, in contrast to p_X representing the PMF of a discrete random variable X. While the probability density at $X = 45$ is $1/70$, $\mathrm{P}(X = 45) = 0$. Similarly to how it is possible to sum over the probability masses with PMFs to calculate the probability the vehicle is traveling over 45, $f_X(x)$ is integrated over all values of x greater than 45. So,

$$\begin{aligned} \mathrm{P}(X > 45) &= \int_{45}^{\infty} f_X(x)dx \\ &= 5/14. \end{aligned}$$

There are three terms that appear throughout this thesis: distribution, density, and measure. A distribution can either refer to the function describing a discrete or continuous random variable but the term density uniquely refers to the function for a continuous random variable. Anything described as a distribution or a density must obey the axioms of probability. The term measure is an abstraction of the ideas such as "length" and "size." A measure maps a set onto the line $[0, \infty)$. It allows for comparison of different sets based on the "size" of the set defined by the measure. Therefore, all distributions are measures but not all measures are distributions. This abstraction is important as distributions are introduced over sets that are more complex than the real numbers or the natural numbers. When a distribution over distributions is discussed, it is typically described as a measure over distributions since the word measure allows for more abstract thinking about how some distributions are "bigger" or "more likely" than others. Additionally, in this chapter, random variables

are always represented by a capital letter and instances of random variables, a lower case letter. This is not strictly adhered to in other chapters.

2.3 Bayesian Inference

When a random variable of a model has not been observed but evidence of the variable is known, the techniques of probabilistic inference can be used to infer the value of the random variable. If the probability of X, the variable of interest, given E, the evidence, is known, then $\mathrm{P}(X)$ can be determined immediately. More often, however, the probability of the evidence is known as a function of the unobserved X. Based on the evidence (e.g., a measurement), inference can often be performed using

$$\mathrm{P}(X|E) = \frac{\mathrm{P}(E|X)\mathrm{P}(X)}{\mathrm{P}(E)} \tag{2.1}$$

(read as the probability of X conditioned on E), known as Bayes' rule. Bayes' rule is derived from the conditional relationships

$$\begin{aligned} \mathrm{P}(Y,Z) &= \mathrm{P}(Y|Z)\mathrm{P}(Z) \\ &= \mathrm{P}(Z|Y)\mathrm{P}(Y) \end{aligned} \tag{2.2}$$

where $\mathrm{P}(Y,Z)$ is a distribution of two random variables, also called a joint distribution. In Equation (2.1) $\mathrm{P}(X)$ is referred to as the prior distribution, the belief about the distribution of X before any evidence has been observed. $\mathrm{P}(E|X)$ is called the likelihood, representing how likely the evidence is given X. $\mathrm{P}(X|E)$, the posterior, is the distribution after the evidence has been incorporated. Often, for mathematical simplicity, a conjugate prior of $\mathrm{P}(E|X)$ is chosen for $\mathrm{P}(X)$. It is called a conjugate prior when $\mathrm{P}(X|E)$ has the same type of distribution as $\mathrm{P}(X)$. To illustrate these concepts, imagine a coin with unknown bias, $0 \leq Q \leq 1$, that is then flipped ten

times. The PMF of X (the outcome of the coin flip) is defined as

$$p_X(x) = \begin{cases} Q & \text{if } x = \text{H} \\ 1 - Q & \text{if } x = \text{T} \\ 0 & \text{otherwise} \end{cases}. \tag{2.3}$$

The outcome of these flips is used as evidence to update the belief of the coin bias. Representing the evidence as the number of heads, n_H, and the number of tails, n_T, from the flips, makes the distribution on n_H and n_T multinomial (Section A.5), written as

$$\text{P}(n_H, n_T|Q) = \binom{n_H + n_T}{n_H} Q^{n_H}(1-Q)^{n_T}.$$

Its conjugate prior, the Dirichlet distribution (Section A.6)

$$\text{P}(Q) = \frac{Q^{\alpha_1 - 1}(1-Q)^{\alpha_2 - 1}}{\int_0^1 u^{\alpha_1 - 1}(1-u)^{\alpha_2 - 1} du},$$

is used as the prior. The posterior, as shown by

$$\begin{aligned} \mathbf{P}(Q|n_H, n_T) &= \frac{\mathbf{P}(n_H, n_T|Q)\mathbf{P}(Q)}{\int_0^1 \mathbf{P}(n_H, n_T|Q)\mathbf{P}(Q) dQ} \\ &= \frac{\binom{n_H+n_T}{n_H} Q^{n_H}(1-Q)^{n_T} \frac{Q^{\alpha_1-1}(1-Q)^{\alpha_2-1}}{\int_0^1 u^{\alpha_1-1}(1-u)^{\alpha_2-1} du}}{\int_0^1 \binom{n_H+n_T}{n_H} Q^{n_H}(1-Q)^{n_T} \frac{Q^{\alpha_1-1}(1-Q)^{\alpha_2-1}}{\int_0^1 u^{\alpha_1-1}(1-u)^{\alpha_2-1} du} dQ} \\ &= \frac{Q^{n_H}(1-Q)^{n_T} Q^{\alpha_1-1}(1-Q)^{\alpha_2-1}}{\int_0^1 Q^{n_H}(1-Q)^{n_T} Q^{\alpha_1-1}(1-Q)^{\alpha_2-1} dQ} \\ &= \frac{Q^{\alpha_1+n_H-1}(1-Q)^{\alpha_2+n_T-1}}{\int_0^1 Q^{\alpha_1+n_H-1}(1-Q)^{\alpha_2+n_T-1} dQ} \end{aligned} \tag{2.4}$$

has the same form (i.e., Dirichlet distributed) as the prior with new parameters. If the prior is chosen such that all values of Q are equally likely ($\alpha_1 = \alpha_2 = 1$), the initial distribution of Q is shown in blue on Figure 2-1 and the posterior in green after seven heads and three tails were observed ($n_H = 7$, $n_T = 3$).

Something that was not properly explained yet was the denominator of Equation (2.4), which, according to Equation (2.1), should be $\text{P}(n_H, n_T)$. Before this can be

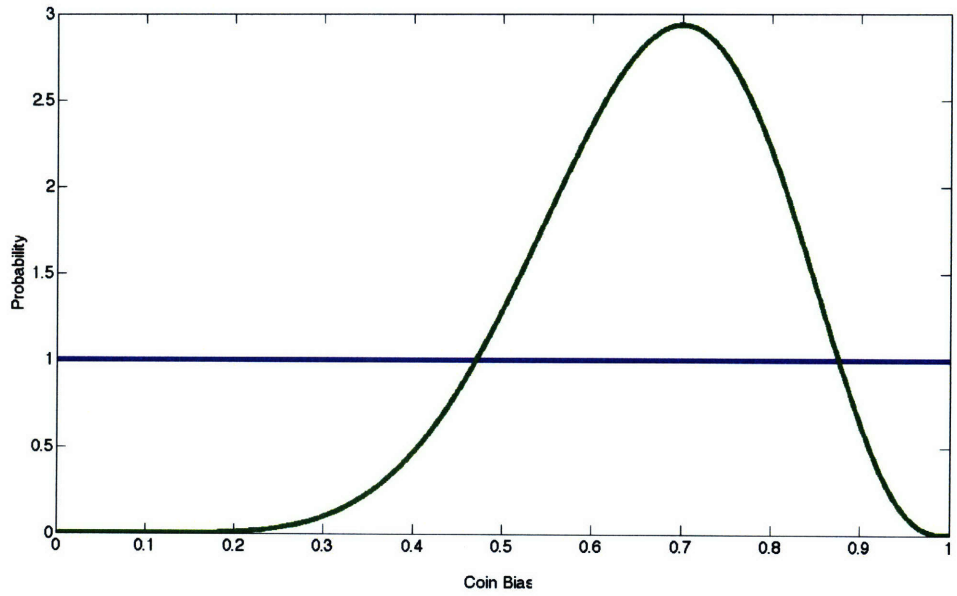

Figure 2-1: The prior coin bias (blue) and posterior (green) after observing seven heads and three tails.

explained, the concept of a marginal distribution must be introduced. Given a distribution of two random variables $\mathrm{P}(Y, Z)$, one of them, say Z, can be "marginalized" out by summing over all possible values it can take. Explicitly, $\sum_Z \mathrm{P}(Y, Z) = \mathrm{P}(Y)$ and can be extended for densities by replacing the summation with integration. Combining the idea of marginalization with Equation (2.2) explains the denominator of Equation (2.4).

The last part of this example that requires explaining is the fundamental methodology choice of this thesis by using a prior distribution. The discussion of choosing to use a prior distribution quickly falls into a much heated philosophical debate between Bayesians and Frequentists. A Frequentist would criticize the arbitrary choice of the prior shape and that the coin bias is an actual value and is misrepresented as a random variable. The Bayesian reply would argue that this is managing uncertainty and the prior allows for not over-fitting the training data. This back and forth argument is apparent in Figure 2-1 by the posterior distribution. It revolves around its shape and raises questions such as:

- Was it correct to assume the coin bias is Dirichlet distributed?
- Perhaps 7/10 is a better estimation of the coin bias?
- What other distributions could have been used?
- Is the mathematical simplicity afforded by using a conjugate prior worth the potential loss of accuracy?

If one believes a coin is truly fair, the model described by Equation (2.3) with $Q = 0.5$ is expected to perform well, a model that performs equally on the first and thousandth trial. If one has reason to believe it is a biased coin, domain experience leads, once again, to Equation (2.3). This time Q is unknown, but it can be learned from the data. While the model is not fully specified, domain experience allows for the application of a parameterized model and, in general, with each flip of the coin the estimation of the bias becomes more accurate.

The argument behind the use of DPs in this thesis is that perhaps the designer has reason to believe the coin had severe manufacturing defects, such that it potentially had more than two sides. Clearly Equation (2.3) would no longer be able to capture the behavior of this coin. If the extension is then made that this coin may have countably infinitely many sides, a parametric model would require infinitely many parameters to capture this behavior. In this case it is perhaps less useful to focus on designing a parametric model, and instead discuss the generative process that creates the sides.

A main assertion of this thesis is that the domains are extremely large and poorly understood. Therefore, in these types of domains, it is unreasonable to expect a designer to create an accurate model. While this thesis makes no claims about putting The Bayesian versus Frequentist argument to rest, the Bayesian approach, using a DP prior, is justified by its ability to fulfill the problem statement.

A concept that is essential to finding the "best" model is data likelihood. Data likelihood refers to how likely the data is to have been generated from a model. More formally, the likelihood of the data, $\mathrm{P}(E|\theta)$, is the probability that the evidence E

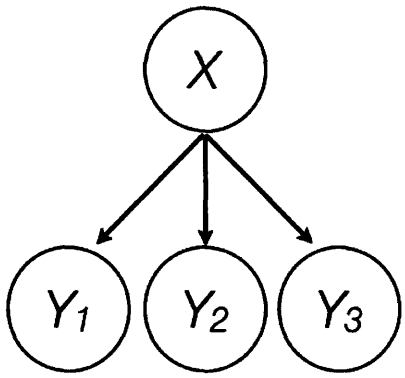

Figure 2-2: A simple Bayesian network.

came from a model with parameters θ. There are two approaches: maximum likelihood estimation and maximum *a posteriori* estimation. The word estimation is used since the goal is to approximate (estimate) the model as accurately as possible with the limited view provided by the training data. With θ^* as the desired parameterization, they can be explicitly written as:

$$\text{Maximum likelihood: } \theta^* = \text{argmax}_\theta \, \text{P}(E|\theta)$$
$$\text{Maximum } a \textit{ posteriori}\text{: } \theta^* = \text{argmax}_\theta \, \text{P}(E|\theta)\text{P}(\theta)$$

2.3.1 Bayesian Networks

A Bayesian network [20] is a directed graph representing the dependencies between random variables. The network is composed of a set of random variables with an arrow from a node X to a node Y standing for the conditional relationship $\text{P}(Y|X)$ [35].

Figure 2-2 shows a sample Bayesian network with four random variables, X, Y_1, Y_2, and Y_3, illustrating conditional distributions $\text{P}(Y_1|X)$, $\text{P}(Y_2|X)$, and $\text{P}(Y_3|X)$. Figure 2-3 shows an equivalent Bayesian network to Figure 2-2 using a common "plates" notation [21].

Bayesian networks are useful for understanding which variables have an influence on other variables. For example, imagine having weighed a person two times and based on these two measurements, a predicted distribution over a third measurement

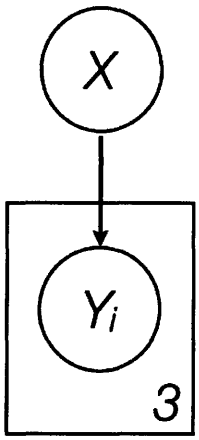

Figure 2-3: An equivalent Bayesian network to Figure 2-2 redrawn using "plates."

is desired. This could be represented as two nodes, Y_1 and Y_2 (the first two measurements), and an arrow connecting them to node Y_3 (the third measurement). This is a perfectly valid Bayesian network since the values of Y_1 and Y_2 are closely related to Y_3, but just because it can be written that way does not mean specifying those conditional distributions will be easy. Instead, if Figure 2-2 were used, where X is the true weight of the person, the conditional distributions would, typically, be easier to describe. This is mainly due to quantifying either how three noisy measurements relate to one another (first representation) or how much the scale's noise corrupts the person's true weight (second representation). Usually, it is far easier to understand how much noise is added by a sensor. These graphical models appear again in Chapter 3 to assist in visualizing the relationships between random variables of the DP-based behavior model.

2.3.2 Markov Models

Markov models [32] are useful in inferring the state of a system that evolves over time. The Markov assumption [35] asserts that the current state of a system only depends on a finite history of past states, typically just on the previous state. Sometimes called Markov processes or Markov chains, Markov models are stochastic processes (Section 2.4) whose structure exploits the Markov assumption. The model is defined

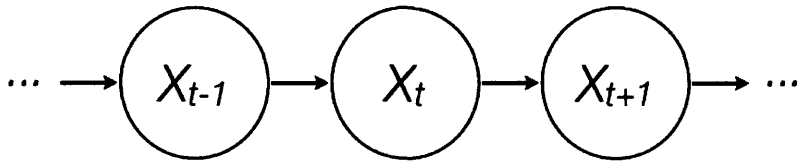

Figure 2-4: A Bayesian network representation of a Markov model

as a set of states X_t at a time t and a transition model $\mathrm{P}(X_{t+1}|X_t)$. The transition model, stating that the next state only depends on the current state, is a direct result of the Markov assumption. Figure 2-4 is a Bayesian network drawing of a Markov model.

A simple vehicle model where the vehicle's state is its current road segment and its next state is its next road segment can be formulated as a Markov model. In this model, the behavior of vehicles passing through a specific intersection is captured by the transition model. Say road A connects to roads B, C, and D and a vehicle transitions from A to one of these three segments with the probabilities described in Table 2.1. These values can be easily learned from data by counting the number

X_{t+1}	$\mathrm{P}(X_{t+1}\|X_t = A)$
B	0.4
C	0.35
D	0.25

Table 2.1: A Markov model describing transition probabilities for road segment A.

of times each transition is taken and dividing that number by the total times a car departed road A. Given training data of 100 cars traversing road A, learning a simple transition model, with 40, 35, and 25 of these paths transitioning to roads B, C, and D, respectively, would result in the learned transition model shown in Table 2.1. It can be shown that this is, in fact, the maximum likelihood estimate (Section 2.3) of the parameters. The subtle thing about this model, and a property that is strong motivation to question the use of a Markov model to model drivers, is that there is no concept of how well the number of observations fit the size of the domain. If 4 of 10 or 40,000 of 100,000 transitioned to segment B, the learned probability would

be the same. In extremely large domains, understanding how well the learned model has captured the underlying behavior is crucial, something that is difficult to reason about taking this approach.

While the Markov assumption is sometimes incorrect, even in those instances where it is questionable, its ease of use makes it an attractive model choice. Additionally, the state can be augmented to increase accuracy (e.g., making the state the past two road segments) at the cost of increasing the size of the state-space. In this thesis, the Markov model is used as a baseline of comparison. In a simple domain it is expected that the proposed DP-based model will agree with the Markov model predictions, providing a form of a sanity check. When the domain is extended to include underlying dynamics, the past history of a vehicle becomes increasingly important, something the approach described in this section cannot model well.

2.3.3 Clustering

To solve the problem of finding destinations, observations need to partitioned into groups, a methodology known as clustering. These clustering techniques aim to place similar observations together. The algorithms have no understanding of what the groups represent. They could be personalities of people or destinations of cars driving around a city; it is simply grouping them based on some group specific distribution.

Algorithm 1 k-means

```
1: Randomly place c_1, ..., c_k in the space
2: while c_1, ..., c_k changed do
3:    for i = 1 to n do
4:       Assign d_i to the closest cluster
5:    end for
6:    for j = 1 to k do
7:       Recalculate the position of c_j to best fit the data assigned to cluster j
8:    end for
9: end while
```

As one may suspect, this problem has been addressed multiple times. The most common algorithm for clustering is the k-means algorithm [25]. As the name implies, the number of clusters, k, must be specified in advance. The algorithm then attempts

to find the k Gaussian distributions that minimize the variance inside of each cluster. The k-means algorithm is shown in Algorithm 1 for centroids $c_1, ..., c_k$ and data $d_1, ..., d_n$.

There is a family of algorithms, called expectation maximization (EM) [8, 36], that behave similarly to the iterative process of k-means. The goal of EM is to find the maximum likelihood (Section 2.3) estimate of the parameters of a model with latent variables. These algorithms are composed of two steps, an expectation (E) step and a maximization (M) step.

- E Step: Estimate the values of the hidden variables, given the parameters and observed data.
- M Step: Find the parameters that best explain the estimated hidden variables and the observed data.

While k-means and EM have been used in numerous applications, in all of them a designer had sufficient domain understanding to be able to specify k or build a parametric model for EM.

The necessity of inferring the number of clusters has been overcome in a variety of ways. This has been done by starting with a small number of clusters and then splitting clusters based on a statistical test [19]. In [6], model complexity is directly penalized by subtracting the number of clusters, scaled by a constant, from the likelihood. An iterative process of increasing k and decreasing k is used to find the correct k using this augmentation. Both of these approaches utilize domain knowledge to set the statistical test or penalty scaling constant. While less restrictive than k-means and EM, this assumption is still too strong for the domains of interest to this thesis.

A play on the name k-means, X-means [30] attempts to infer X using the Bayesian information criterion (BIC) [22]. The BIC of the model M is

$$\text{BIC}(M) = \ln(L) - \frac{m}{2}\ln(n) \tag{2.5}$$

where L is the likelihood of the data, m is the number of parameters, and n is the

number of observations. X-means decides the number of parameters by maximizing Equation (2.5). While this approach is less restrictive than all of the others previously mentioned, choosing this as a way of penalizing complexity is as arbitrary as assuming the locations were generated from a DP. Through the use of the probability of a novel path appearing, the DP approach has the added benefit of being able to judge how well the behavior is seen in the data.

By assuming the clusters are generated from a DP, the approach taken here is similar to what was first presented as the infinite Gaussian mixture model [33]. The details of applying the method described in the paper are discussed in Section 3.2.2.

2.3.4 Monte Carlo Methods

Sometimes exact inference is impossible and in these cases Monte Carlo methods can be used. Monte Carlo methods rely on repeated use of random numbers to approximate probability distributions. To use these Monte Carlo methods, samples must be drawn from distributions. The straightforward way to draw values from a distribution is to input a value drawn uniformly on the unit interval into the inverse of the distribution's CDF. An issue arrises in some distributions when the inverse of the CDF is difficult to calculate.

Rejection Sampling

To sample from a distribution, such as Equation (3.15),

$$\mathrm{P}(\beta|s_1,...,s_k,\omega) \propto \Gamma(\beta/2)^{-k}(\beta\omega/2)^{(k\beta-3)/2}e^{-1/(2\beta)}\prod_{j=1}^{k} s_j^{\beta/2}e^{-\beta\omega s_j/2},$$

calculating the inverse of the CDF would be extremely difficult. Instead, a method called rejection sampling [16] can be used where samples are generated from a "proposal" distribution, g, that is both easier to sample and is nonzero where the distribution of interest, f, is nonzero. A sample is either accepted or rejected based on the region of probability between the $f(x)$ and $g(x)$ at the sample x. The procedure is

shown in Algorithm 2 for $cg(x) \geq f(x)$ for all $x \in \mathcal{R}^N$ and $c \geq 1$. In the algorithm, $\mathcal{U}[a, b]$ is the uniform distribution between a and b.

Algorithm 2 Rejection Sampling

1: $u = \infty$
2: **while** $u \geq f(y)$ **do**
3: Draw $y \sim g$
4: Draw $u \sim \mathcal{U}[0, cg(y)]$
5: **end while**

Markov Chain Monte Carlo

For a given Bayesian network and some observed variables, the objective is usually to infer the posterior distributions of the hidden (not observed) variables. Often, it is impossible to perform exact analytical inference, in which case Markov chain Monte Carlo (MCMC) methods can be used [28]. In MCMC, the Bayesian network is said to be in a current state where each variable in the network has a specific value. The state then evolves by updating the values of the hidden variables conditioned on the variable settings from the previous state. As the evolution continues, it approaches a "dynamic equilibrium" [35] where the number of occurrences of a network state is proportional to the posterior probability of that state. This methodology is at the heart of the inference procedure described in Chapter 3.

Gibbs Sampling

Gibbs sampling [12], a MCMC method, often appears in conjunction with more complex MCMC inference procedures. Its purpose is to generate samples from a difficult to sample multivariate distribution. Typically for Gibbs sampling, conditional distributions between the variables are known and are used to draw samples of the multivariate distribution. This algorithm is typically performed for a set number of iterations, or "sweeps." Algorithm 3, for variables $\theta_1, ..., \theta_n$ whose conditional distributions are known, is as follows:

Algorithm 3 Gibbs Sampling

1: **for** sweep = 1 to # of sweeps **do**
2: **for** i randomly chosen from $1, ..., n$ **do**
3: Draw $\theta_i \sim \theta_i|\theta_1, ..., \theta_{i-1}, \theta_{i+1}, ..., \theta_n$
4: **end for**
5: **end for**

2.4 Stochastic Processes

A stochastic process [11] is a sequence of random variables with an evolution that has some element of chance, resulting in no method for exactly predicting the future values. One of the simplest examples of a stochastic process is the Bernoulli process [7].

The Bernoulli process is a sequence composed of N independent Bernoulli trials (i.e., coin flips). In a Bernoulli trial, a success (heads) happens with probability $\mathrm{P}(X = 1) = p$ and a failure with probability $\mathrm{P}(X = 0) = 1 - p$. To use this intuition to eventually understand Dirichlet processes, it is not enough to think of stochastic processes as a sequence of outcomes that were generated over time. It is important to understand a stochastic process as defining a probability measure on sequences. The Bernoulli process defines a probability distribution on sequences of ones and zeros. Consider the two sequences HTHTH and HHHHH known to have been generated from coin flipping with a biased coin where $\mathrm{P}(X_i = H) = 0.8$ for the ith flip. Therefore the Bernoulli process defines a measure over heads and tails sequences such that $\mathrm{P}(\mathrm{HTHTH}) = 0.02048$ and $\mathrm{P}(\mathrm{HHHHH}) = 0.32768$.

Gaussian processes (GP) [34] are the next logical step to building the intuition for DPs. Similarly to how the Bernoulli process defines a measure over sequences, GPs define a measure over functions. It does this by creating an infinite multivariate Gaussian distribution (Section A.1). The idea is that this infinite vector is approximately a function, and a Gaussian process defines a measure over these infinite vectors. By marginalizing (integrating) over all but a finite set of these infinite variables, the resulting distribution over them is Gaussian. Since GPs are useful here solely to build intuition, with the goal of not getting lost in the details of GPs, Figure 2-5 helps to

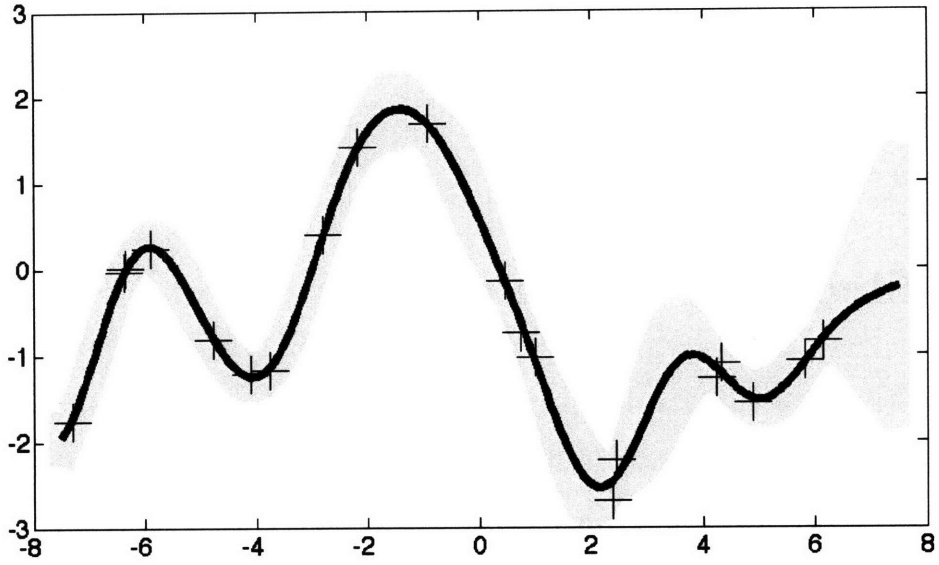

Figure 2-5: A posterior Gaussian process conditioned on 20 observations.

understand them on a basic level[2]. The graph shows a GP as a mean (the solid line) and a 95% confidence interval (the gray region) conditioned on the 20 observations shown as crosses. The observations define a distribution over functions, where functions that more closely resemble the black line are more likely. There is far more to GPs than that which was mentioned here, but any further explanation would have little purpose for this thesis.

Instead of considering all functions, the next step is to limit the discussion to special kinds of functions: random measures. These random measures are random functions that obey the axioms of probability (Section 2.2). As an example of a process whose draws are random measures, consider a one dimensional Gaussian distribution with an unknown standard deviation. For the sake of the example, and being Bayesian, let's say domain knowledge leads us to place a prior over the variance

[2]This figure was generated using the Gaussian process for machine learning MATLAB toolbox (www.gaussianprocess.org/gpml/code).

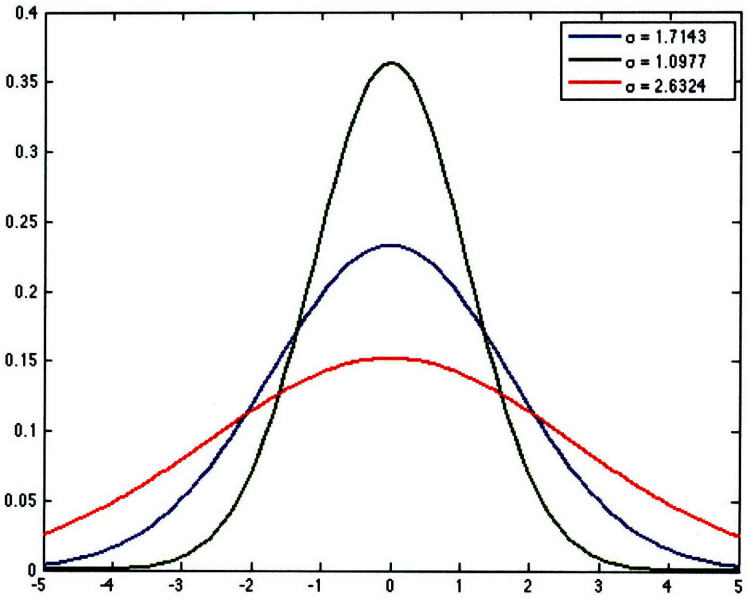

Figure 2-6: Plots of three random measures.

with distribution

$$p_\Sigma(\sigma) = \begin{cases} 3/4, & \text{if } 1 < \sigma < 2 \\ 1/4, & \text{if } 2 \leq \sigma < 3 \\ 0, & \text{otherwise} \end{cases}.$$

Thinking of this as a random process, Figure 2-6 shows three random measures drawn from this process. This implies that the process is a measure over measures. Distributions shaped like the blue or green distribution on Figure 2-6 are three times more likely to occur as those shaped as the red distribution. This notion of a measure of measures is essential to understanding Dirichlet processes.

2.4.1 Model Types

Before Dirichlet processes are discussed in the next section, it is important to understand the difference between the two types of models that have, thus far, been presented to represent probability distributions. These two types are parametric and nonparametric. A parametric model is defined by a finite set of parameters and a relationship between them. A simple example of this is a Bernoulli trial, parameterized

by a single value, p. Nonparametric models, although the name may not imply it, do have parameters, though there are infinitely many of them. Another way to view the two types of models is that the number of parameters in a nonparametric model tend to grow with the amount of training data.

A Gaussian process (as briefly discussed in Section 2.4) is an example of a nonparametric probability distribution since it would require an infinite number of mean and covariance values to fully specify. A second nonparametric distribution, and the main building block of this thesis, is described in the following section.

2.4.2 Dirichlet Processes

Dirichlet processes (DP) [42, 3, 15, 5, 41] define not just a measure over any measure but over measures that have Dirichlet distributed (Section A.6) finite marginals. This is identical to how a GP has Gaussian distributed finite marginals. Formally, for a random measure G over space Θ, it is said that $G \sim DP(\alpha, H)$ is Dirichlet process distributed with base distribution H and concentration parameter α if

$$(G(A_1), ..., G(A_r)) \sim \text{Dir}(\alpha H(A_1), ..., \alpha H(A_r))$$

for every finite partition $A_1, ..., A_r$ of Θ. A partition of Θ is a collection of disjoint sets that contain all elements of Θ. $A_1 = [0, 0.4)$ and $A_2 = [0.4, 1]$ is an example of a partition of Θ when Θ is the unit interval. To further clarify the notation, if H is a uniform distribution over the unit interval, $H(A_1) = \int_0^{0.4} dx = 0.4$.

The base distribution, H, can be understood as the "mean" of the DP. The concentration parameter, α, controls how closely a draw from this DP resembles its "mean." This is similar to the precision describing how closely a draw from a Gaussian distribution resembles its mean. In Figure 2-7 the top plot is the base distribution, $H = f_G(2, 1/2)$ (Section A.2), and the three plots below are for $\alpha = \{1, 10, 10000\}$. Immediately, what may strike the reader is that the draws of this process are discrete probability distributions even though the base measure is continuous. Something less obvious from the picture is that, while it is true they are discrete, there is a prob-

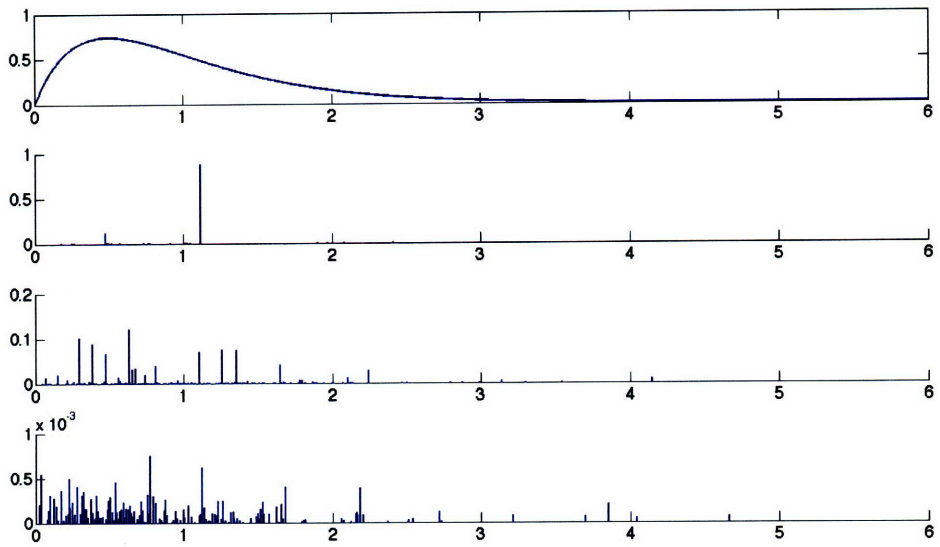

Figure 2-7: Three sample draws (bottom three plots) from the base measure (top) with $\alpha = 1$ (second from the top), $\alpha = 10$ (second from the bottom), $\alpha = 10000$ (bottom).

ability mass associated with every point of Θ (in this example $\Theta = [0, \infty)$). These plots show how the base measure can be thought of as the mean of a DP and α can be thought of as precision. The higher α is, the more closely the draw resembles H.

While there are many related processes, there are two that are particularly useful to this thesis: the Chinese restaurant process and the GEM process. Note that the Pólya urn scheme, the original process used to prove the existence of the DP, is not covered here but an interested reader can refer to [9].

The Posterior Dirichlet Process

As Section 2.2 discussed, the choice of a conjugate prior can often greatly simplify the calculation of the posterior. As illustrated by the conjugate prior example in Section 2.2, the Dirichlet distribution is the conjugate prior of the multinomial distribution. That relation allows for the posterior distribution (the DP conditioned on previously observed $\theta_1, ..., \theta_n$) to itself be DP distributed when a DP is used as the prior. Formally, conditioned on $\theta_1, ..., \theta_n$ drawn from $G \sim DP(\alpha, H)$, for every finite

partition,

$$(G(A_1), ..., G(A_r))|\theta_1, ..., \theta_n \sim \text{Dir}(\alpha H(A_1) + n_1, ..., \alpha H(A_r) + n_r)$$

where n_i is the number of $\theta_1, ..., \theta_n$ that fall in the partition A_i. Therefore the posterior DP can be written as

$$G|\theta_1, ..., \theta_n \sim DP\left(\alpha + n, \frac{\alpha}{\alpha + n}H + \frac{1}{\alpha + n}\sum_{i=1}^{n}\delta_{\theta_i}\right)$$

where δ_{θ_i} is a point mass centered at θ_i (i.e., the contribution of $\frac{1}{\alpha+n}$ is only added to $\frac{\alpha}{\alpha+n}H$ at θ_i for $i = 1, ..., n$).

Chinese Restaurant Process

The Chinese restaurant process (CRP), initially named in [2], comes from that author's observation that Chinese restaurants appear to have infinitely many tables. The generative model for the CRP is as follows. Consider an empty restaurant with countably infinite tables. The first customer walks in and sits at the first table deterministically. The second customer sits at the first table with probability $\frac{1}{1+\alpha}$ and the second table with probability $\frac{\alpha}{1+\alpha}$. It then continues such that the nth customer sits at a previously occupied table proportional to the number of customers already seated at it and a new table proportional to α (the same α as in Section 2.4.2).

As described thus far, this process is a measure on partitions of the integers where each customer is an integer and each table is a partition. This process can be extended to produce a draw from a DP with base distribution H (the same H as in Section 2.4.2). When a customer sits at a new table, a parameter for that table is drawn from the base measure H. A customer sitting at an already occupied table inherits the parameter of that table. Looking back at Figure 2-7, the bottom three plots show the next customer sitting at a table with parameter θ proportional to the height of the bar at θ. Therefore, the distribution of $\theta_1 \in \Theta$ as the first customer walks in is

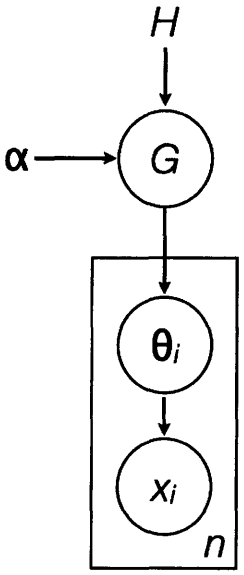

Figure 2-8: The Bayesian network representation of the CRP.

$\theta_1|\alpha, H \sim H$, and after n people have been seated is

$$\theta_{n+1}|\theta_1, ..., \theta_n, \alpha, H \sim \sum_{i=1}^{n} \frac{1}{n+\alpha}\delta(\theta_{n+1}, \theta_i) + \frac{\alpha}{n+\alpha}H. \tag{2.6}$$

where δ is the Kronecker-delta function ($\delta(a, b) = 1$ if and only if $a = b$ and zero otherwise). The purpose of explaining this process is that n samples from the CRP are equivalent to sampling from a draw from a DP n times. Figure 2-8 shows the Bayesian network representation of n data points, x_i for $i = 1, ..., n$, generated from this process. It is important to note that it is very likely that many of the θ_i values are the same (all people at the same table have the same θ_i) and while it is not obvious yet, this formulation makes inference about these θ_is difficult.

The purpose of explaining processes that are related to DPs is to enable inference procedures of the latent variables θ_i. These variables are referred to as latent since they are not directly observable from the data. Instead, the variables must be inferred from their relationship with variables that are observable. The inference algorithm, Algorithm 4, is an implementation of Gibbs sampling (Section 2.3.4). Equations (2.6) and (3.18) describe how to implement Steps 6 and 7, respectively. This often leaves

Algorithm 4 CRP Inference

for $i = 1$ to n **do**
  Draw $\theta_i \sim H$
end for
for sweep = 1 to # of sweeps **do**
  for i randomly chosen from $1, ..., n$ **do**
    Draw $\theta_i \sim \theta_i | \theta_1, ..., \theta_{i-1}, \theta_{i+1}, ..., \theta_n, \alpha, H$
  end for
  Draw $\alpha \sim \alpha | n, k$
end for

one with less than ideal parameters due to this procedure's inability to "move" chunks of data at once. If 10 people were sitting at a table that happened to have been drawn a mediocre parameter, the only way for the parameter to change is for one of them to land on a new table, draw a better parameter, and then have all of them switch. This is difficult because it is not only important how well the data fits the parameter of the table but also how many people are already sitting at the table (see Equation (2.6)). In some sense, the table parameters and assignments would need to "travel" through an area of lower likelihood to find better parameters, something this procedure is unlikely to accomplish.

The next section describes another process related to DPs that make inference significantly easier. As a final remark on CRPs, if H is a continuous distribution it is impossible for two tables to have the same parameter, but if H is a discrete distribution there is a nonzero probability that two tables have the same parameter (this follows from the basic probability theory in Section 2.2).

GEM Process

One of the simplest constructions of the DP was shown in [37] by the use of "stick breaking." This construction is similar to that of the Chinese restaurant if there are infinite customers. The result of stick breaking is a discrete probability distribution over all natural numbers. These stick breaking weights, $\pi = (\pi_n)_{n=1}^{\infty}$ such that $\sum_{n=1}^{\infty} \pi_i = 1$, are said to be drawn from a GEM process [31], standing for Griffiths,

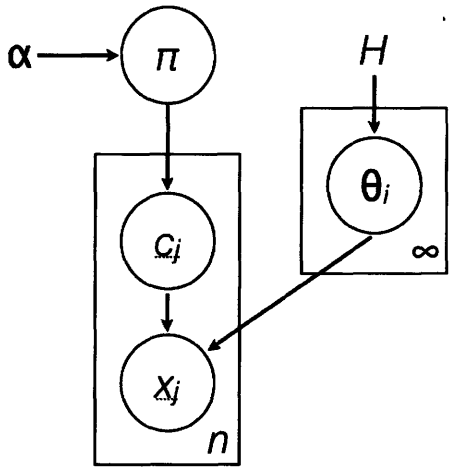

Figure 2-9: The Bayesian network representation of the GEM process.

Engen, and McCloskey. More concretely,

$$\pi'_n|\alpha \sim \text{Beta}(1, \alpha)$$

$$\pi_n = \pi'_n \prod_{i=1}^{n-1}(1 - \pi'_i).$$

For Dirichlet process distributed $G \sim DP(\alpha, H)$, $G = \sum_{n=1}^{\infty} \pi_n \delta_{\theta_n}$ where $\delta_{\theta_n} = 1$ at $\theta_n \sim H$ and zero otherwise. Figure 2-9 shows the Bayesian network representation of n data points drawn from this process where c_j indexes the θ_i from which x_j was generated. Due to its simplicity, this is the exact method that was used to generate Figure 2-7.

It was mentioned in Section 2.4.2 that this formulation, while slightly more difficult to understand than the CRP, is far easier on which to perform inference. To make this algorithm clear, it is going to be described using the same terms as the CRP formulation (customers, tables, etc) but it is important to understand that this procedure was enabled by understanding a DP as Figure 2-9 and not as Figure 2-8. The reason the CRP terminology is still used is that it is extremely intuitive. Once again, Gibbs sampling is the foundation of the inference algorithm. In Algorithm 5 $c_1, ..., c_n$ are the table assignments, $x_1, ..., x_n$ are the data points, θ_j is the parameter for table j, and H is the base measure. The distribution $\text{P}(c_i|c_1, ..., c_{i-1}, c_{i+1}, ..., c_n, \theta_1, ..., \theta_k, \alpha)$

Algorithm 5 GEM Inference

for $i = 1$ to n **do**
  $c_i = 1$
end for
$k = 1$
Draw $\theta_1 \sim H$
for sweep = 1 to # of sweeps **do**
  for i randomly chosen from $1, ..., n$ **do**
    $c_{old} = c_i$
    Draw $c_i \sim c_i | c_1, ..., c_{i-1}, c_{i+1}, ..., c_n, \theta_1, ..., \theta_k, \alpha$
    if $c_i > k$ **then**
      Draw $\theta_{k+1} \sim \theta_{k+1} | x_i, H$
      $k = k + 1$
    end if
    if table c_{old} is empty **then**
      Remove $\theta_{c_{old}}$
      $k = k - 1$
    end if
  end for
  for $j = 1$ to k **do**
    Draw $\theta_j \sim \theta_j | c_1, ..., c_n, \theta_1, ..., \theta_k, \alpha$
  end for
  Draw $\alpha \sim \alpha | n, k$
end for

is a discrete distribution with $k+1$ elements. Each element $i = 1, ..., k+1$ is proportional to the likelihood that c_i belongs to table i, where table $k+1$ is a new table. This distribution is defined as

$$\mathrm{P}(c_i = j | c_1, ..., c_{i-1}, c_{i+1}, ..., c_n, \theta_1, ..., \theta_k, \alpha) \propto \frac{n_j}{n-1+\alpha} \mathcal{F}(x_i, \theta_j) \tag{2.7}$$

for $c_i \leq k$ and

$$\mathrm{P}(c_i = j | c_1, ..., c_{i-1}, c_{i+1}, ..., c_n, \theta_1, ..., \theta_k, \alpha) \propto \frac{\alpha}{n-1+\alpha} \int \mathcal{F}(x_i, \theta) dG_0(\theta) \tag{2.8}$$

for $c_i > k$ where $x_i | \theta_i \sim \mathcal{F}(\theta_i)$, and G_0 is the conjugate prior of $\mathcal{F}$ [28].

Equations (2.6) and (3.18) describe how to implement Steps 19 and 22, respectively. As the algorithm shows, it is now not only possible to move chunks of data, but this happens for each θ_j every sweep during Step 19.

2.4.3 Past Applications of Dirichlet Processes

This section briefly discusses some publications using DPs to give a flavor of the kind of problems DPs have been used to solve. To learn the hierarchy of topics in groups of documents, [10] used a nested Chinese restaurant process. [43] used DPs to identify motif structures in biopolymer sequences (such as proteins and DNA). To track maneuvering targets, [17] used a hierarchical Dirichlet process Markov model to represent potentially unknown underlying system modes. [39] used hierarchical DPs to tackle the revisiting problem in robot map building by learning "typical" room structures offline and the novel probability mass corresponds to observing a room whose structure was not in the training set. [38] used hierarchical DP Markov models to model genetic recombination for the potentially infinite "founders" and events which generate genealogies. All of these applications clearly highlight a DP's ability to either learn the number of states, handle infinite states or maintain a probability mass of a novel state.

2.5 Driver Modeling

There have been numerous approaches to modeling the route preferences that drivers choose. They can be loosely grouped into designer-provided models and common models. Based on a driving behavior survey, [14] developed a behavior model that used a behavior survey to classify drivers into different profiles to enable prediction. This is not the type of data that this thesis is concerned with and, to use this methodology in other cities, it may be necessary to perform additional surveys. [23] focused on driver preferences by accounting for both the time it takes to traverse a road segment based on the time of day and the amount of time loss a driver is willing to accept. The focus of the paper is on finding better paths to suggest to drivers, not to predict the actual routes drivers choose.

Section 2.3.2 outlined Markov models, and while the Markov assumption is rather strong, their ease of use makes them an attractive choice for driver modeling. [29] took an approach similar to that described in Section 2.3.2 except the true driver state is "hidden" by GPS sensor noise and the mode of the driver is unobservable. This formulation is difficult to make end location predictions due to an intractable sum over all possible actions leading to a destination and is very susceptible to over-fitting the training data. To enable end location predictions, [4] models these transition probabilities explicitly, but in doing so prevents any method for updating the probabilities based on a partially observed path. Using a hierarchy of Markov models, [24] is able to make both local and destination predictions but is still susceptible to over-fitting the training data in regions of sparse data. Taking a machine learning [27, 40] approach, [44] uses inverse reinforcement learning. While this paper presents impressive results, it does not discuss a method for finding destinations and is vulnerable to over-fitting the training data.

As a side note, a fascinating fact was mentioned in the results section of [23]. According to the data collected, only 34.5% of drivers choose the fastest routes (based on their aggregated data) and only 30% of drivers follow the route that would be provided by a traditional fastest-route planner. This further drives home the point

that attempting to create a model for driver behavior is extremely difficult. One of the first aspects of any model that attempts to model drivers, one would assume, is that a driver tries to minimize travel time. Whether this is due to either driver ignorance or another factor (e.g., a stressful route) is highly debatable and most likely difficult to model.

Chapter 3

The Driver Behavior Model

3.1 Chapter Overview

This chapter introduces a driver behavior model based on Dirichlet processes. Section 3.2.1 explains the overall model structure. The methods for clustering destinations and learning the distribution over paths are described in Sections 3.2.2 and 3.2.3, respectively.

3.2 Behaviors as Paths to Destinations

3.2.1 Model Description

The behavior model of a driver, as described in Section 1.3, consists of two parts: destinations and paths to the destinations. The easiest way of understanding the model is to think about how it would generate a path. First, an ending road segment is chosen from the mixture of Gaussian distributions representing destinations. This is done by selecting a destination proportional to the "weight" of each Gaussian, and then choosing the ending location based on the distribution of that destination. Once the destination and ending location have been selected, the distribution on all paths for the destination (Section 1.3 described how each destination indexes a distribution on all paths) is used to choose a path.

In this thesis, DPs (Section 2.4.2) are used both for clustering destinations and for finding a distribution over paths based on the justifications provided in Section 1.3. Figure 3-1 shows the Bayesian network (Section 2.3.1) of the driver behavior model[1].

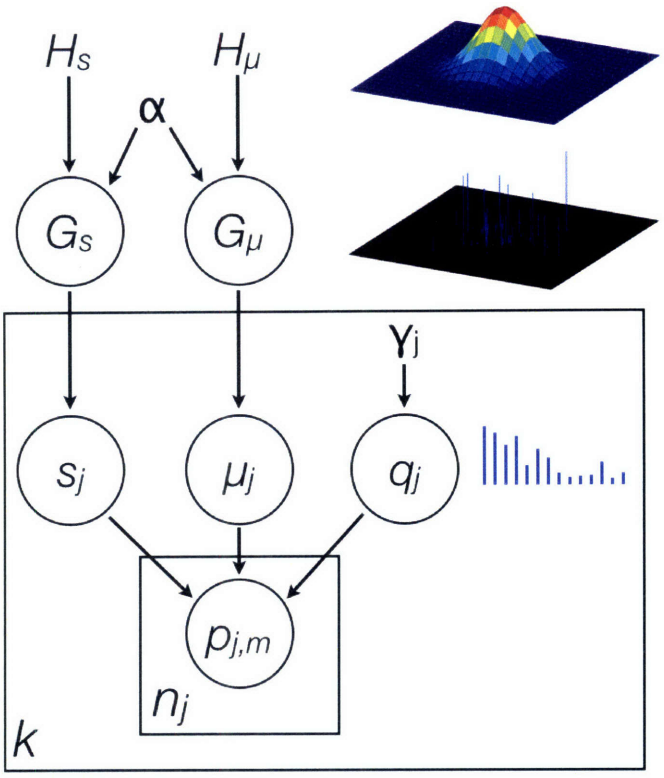

Figure 3-1: A Bayesian network representation of the behavior model.

The variables in the figure are held consistent throughout this chapter and are defined as follows:

- k: the number of total destinations
- n_j: the number of paths ending at destination j
- $p_{j,m}$: the m^{th} path generated to destination j
- q_j: the distribution over all paths to destination j
- γ_j: the parameter of the GEM measure on q_j

[1]This diagram was motivated by a similar diagram in [21].

- μ_j: the mean of destination j
- s_j: the precision of destination j
- G_μ: the distribution over μ_j
- G_s: the distribution over s_j
- H_μ: the base measure of the DP over G_μ
- H_s: the base measure of the DP over G_s
- α: the concentration parameter of the DPs of over both G_μ and G_s

In the figure, example shapes of H_μ and $G_\mu \sim \mathcal{DP}(\alpha, H_\mu)$ are shown. Each group of data (e.g., the cluster the data belongs to) has a mean $\mu_j \sim G_\mu$, where the height of the blue lines represent the probability of μ_j choosing the parameter value at that location[2]. There is a similar relationship for the precision s_j, governed by H_s, α, and G_s. The distribution over the paths leading to destination j is defined by the distribution $q_j \sim \text{GEM}(\gamma_j)$ (Section 2.4.2). Shown next to q_j, in blue, is a plot similar to G_μ's except each probability mass on q_j's diagram is indexed by a natural number[3] where each index corresponds to a path. Based on the destination parameters μ_j and s_j and distribution q_j a path $p_{j,m}$ is drawn. This is repeated for $m = 1, 2, ..., n_j$ data points (paths) for $j = 1, 2, ..., k$ groups (destinations).

To learn the model, destinations must first be found by clustering the last road segment of the training data. After this is accomplished, the training data is grouped based on its most likely destination. Given these paths for each destination, the DP measure over the distribution on paths is learned.

The purpose of Figure 3-1 is to understand the relationship between the observable and latent variables. Except for $p_{j,m}$, all other variables in the figure are hidden. Given a set of training data, the learned model will consist of inferred values for $\alpha, \gamma_j, \mu_j, s_j, n_j$, and k for $j = 1, ..., k$. The distributions H_μ and H_s are assumed and

[2]Despite showing only one hundred visible blue lines, there is a probability mass at all locations.
[3]Only fifteen values are shown but there is a probability mass associated with all natural numbers.

G_μ, G_s, and q_j are not necessary to make predictions (this can be seen from Equation (2.6)).

Figure 3-1 can be redrawn slightly to show that learning the behavior model can be accomplished in two distinct steps. Figure 3-2 represents the same model except a variable $x_{j,m}$, the parking location, has been introduced. This new variable is

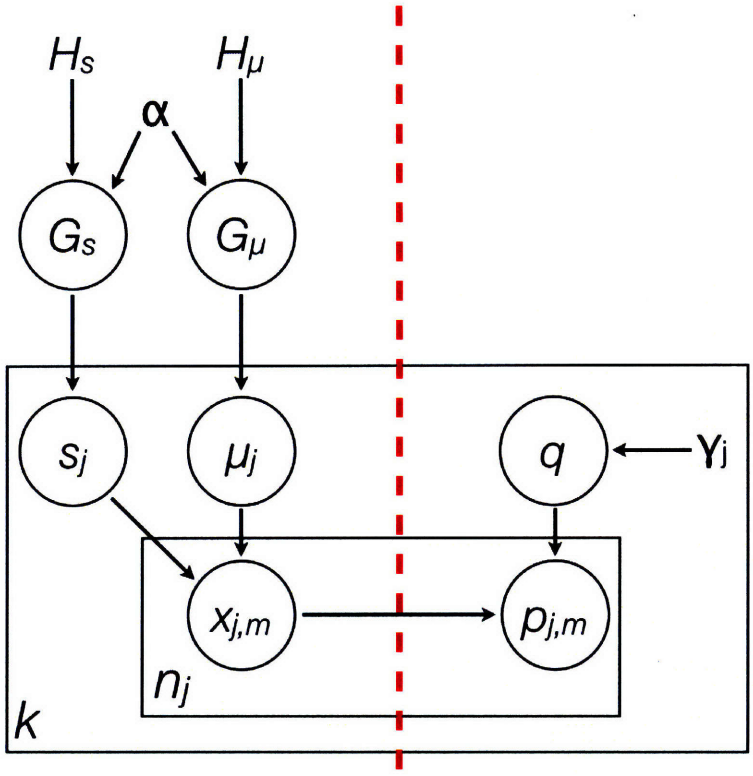

Figure 3-2: A second Bayesian network representation of the behavior model.

observable from the data and the dotted red line on the figure shows the separation between the clustering problem (left) and the path distribution learning (right). The only technicality is the clustering part must be solved first, since the path distribution learning problem requires knowing to which destination each path belongs. Section 3.2.2 is concerned with solving the left side of the figure and Section 3.2.3 focuses on the right side of the figure. Section 3.2.4 puts the full model back together to make destination and route predictions.

3.2.2 Infinite Gaussian Mixture Model

The infinite Gaussian mixture model (IGMM) was introduced in [33] but uses a somewhat uncommon parameterization of the gamma distribution and lacks a detailed description of multivariate clustering. [26] applies [33]'s model with the same parameterization described in Section A.2 but they use prior knowledge to set the DP parameter α and do not learn it from the data. The IGMM is defined as a mixture of k Gaussian distributions with mixing proportion π_i (i.e., the "weight" of each destination), mean μ_i and precision s_i for $i = 1, ..., k$. The probability of paths with end road segments $x_1, ..., x_n$ generated from these clusters can be calculated as

$$\mathrm{P}(x_1, ..., x_n | \mu_1, ..., \mu_k, s_1, ..., s_k, \pi_1, ..., \pi_k) = \prod_{i=1}^{n} \sum_{j=1}^{k} \pi_j \mathcal{N}(x_i; \mu_j, s_j^{-1}). \tag{3.1}$$

Indicator variables, $c_1, ..., c_n$, are now introduced to assist in model learning with $c_i = j$ implying data point i belongs to group j. Conditioning Equation (3.1) on these indicators yield

$$\begin{aligned} \mathrm{P}(x_1, ..., x_n |, \mu_1, ..., \mu_k, s_1, ..., s_k, c_1, ..., c_n) &= \mathrm{P}(x_1 |, \mu_{c_1}, s_{c_1}), ..., \mathrm{P}(x_n |, \mu_{c_n}, s_{c_n}) \\ &= \prod_{i=1}^{n} f_{\mathcal{N}}(x_i; \mu_{c_i}, s_{c_i}^{-1}), \end{aligned}$$

due to each data point's conditional independence caused by the indicators. More specifically, the distribution of the data points belonging to group j is

$$\begin{aligned} \mathrm{P}(\tilde{x}_1, ..., \tilde{x}_{n_j} | \mu_j, s_j) &= \prod_{m=1}^{n_j} f_{\mathcal{N}}(\tilde{x}_m; \mu_j, s_j^{-1}) \\ &= \prod_{i=1}^{n_j} \frac{s_j^{1/2}}{\sqrt{2\pi}} e^{-s_j(\tilde{x}_m - \mu_j)^2/2} \\ &\propto s_j^{n_j} e^{-s_j \sum_{i=1}^{n_j} (\tilde{x}_m - \mu_j)^2/2} \end{aligned} \tag{3.2}$$

where $\tilde{x}_1, ..., \tilde{x}_{n_j}$ are the data points belonging to group j and n_j is their count. The symbol "$\propto$" stands for "proportional to," and is commonly used in probability theory

since the axioms of probability demand that distributions have a total mass or density that equals one. By incorporating this notion of proportionality, the algebra is made simpler since all constants can be removed, as was done in Equation (3.2).

To find $\{\mu_1, ..., \mu_k, s_1, ..., s_k, \pi_1, ..., \pi_k\}$ that best explains the data, the notion of DPs must be introduced, since this is the approach taken to overcoming not knowing k. A modified version of the left half of Figure 3-2 that includes the indicator variables $c_1, ..., c_n$ is shown in Figure 3-3. In the figure, $\{\pi_1, \pi_2, ...\} = \pi \sim \text{GEM}(\gamma)$ as described

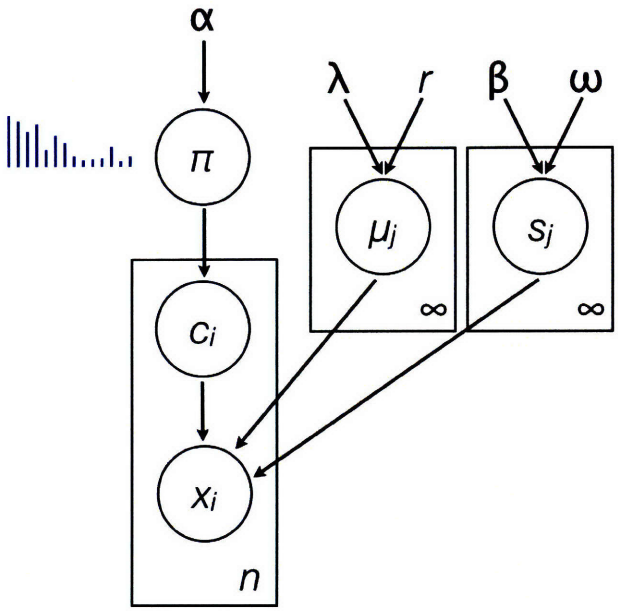

Figure 3-3: A Bayesian network representation of the mixtures of Gaussians.

in Section 2.4.2. Additionally, λ, r, β, and ω have been added with the relations $H_\mu \triangleq \mathcal{N}(\lambda, 1/r)$ and $H_s \triangleq \mathcal{G}(\beta, \omega)$. It is important to note that there is a minor overloading of notation here. In Figure 3-1 $\mu_i \sim G_\mu$ and $s_i \sim G_s$ but in Figure 3-3

$$\mu_j \sim \mathcal{N}(\lambda, 1/r) \tag{3.3}$$

$$s_j \sim \mathcal{G}(\beta, (\beta\omega)^{-1}). \tag{3.4}$$

Even though these two sets of relations are similar, notice how many μ_is and μ_js are generated (the same applies to s_i and s_j). In Figure 3-1 there are exactly n μ_i, with any two μ_i having a positive probability of having the same value (G_μ is a multivariate

distribution). In Figure 3-3 there are infinitely many μ_j, with any two having zero probability of taking the same value ($\mathcal{N}(\lambda, 1/r)$ is continuous) with the indices of the used parameters stored in the indicators $c_1, ..., c_n$. From this point on, assume any mention of μ_j or s_j refers to them as they are in Equations (3.3) and (3.4).

To begin, the prior distributions are introduced. While some may seem somewhat arbitrary, they were chosen with the purpose of simplifying much of the algebra while restricting the flexibility of the model as little as possible. By taking this approach, the posterior distributions are either normal or gamma distributed for all but one of the parameters. All distributions used are discussed in more detail in the appendix.

The prior distributions are defined as:

$$\begin{aligned}
\mathrm{P}(\lambda) &= f_{\mathcal{N}}(\lambda; \mu_x, \sigma_x^2) \\
&= \frac{1}{\sigma_x\sqrt{2\pi}} e^{\frac{-(\lambda-\mu_x)^2}{2\sigma_x^2}} \propto e^{\frac{-(\lambda-\mu_x)^2}{2\sigma_x^2}} && (3.5) \\
\mathrm{P}(r) &= f_{\mathcal{G}}(r; 1/2, \sigma_x^{-2}/2) \\
&= \frac{1}{\Gamma(1/2)(\sigma_x^{-2}/2)^{1/2}} r^{1/2-1} e^{-r\sigma_x^2/2} \propto r^{-1/2} e^{-r\sigma_x^2/2} && (3.6) \\
\mathrm{P}(\mu_j|\lambda, r) &= f_{\mathcal{N}}(\mu_j; \lambda, 1/r) \\
&= \frac{r^{1/2}}{\sqrt{2\pi}} e^{-r(\mu_j-\lambda)^2/2} \propto r^{1/2} e^{-r(\mu_j-\lambda)^2/2} && (3.7) \\
\mathrm{P}(\beta^{-1}) &= f_{\mathcal{G}}(\beta^{-1}; 1/2, 2) \\
\Rightarrow \mathrm{P}(\beta) &= f_{\mathcal{G}^{-1}}(\beta; 1/2, 1/2) \\
&= \frac{(1/2)^{1/2}}{\Gamma(1/2)} \beta^{-1/2-1} e^{-1/(2\beta)} \propto \beta^{-3/2} e^{-1/(2\beta)} && (3.8) \\
\mathrm{P}(\omega) &= f_{\mathcal{G}}(\omega; 1/2, 2\sigma_x^2) \\
&= \frac{1}{\Gamma(1/2)(2\sigma_x^2)^{1/2}} \omega^{1/2-1} e^{-\omega/(2\sigma_x^2)} \propto \omega^{-1/2} e^{-\omega/(2\sigma_x^2)} && (3.9) \\
\mathrm{P}(s_j|\beta, \omega) &= f_{\mathcal{G}}(s_j; \beta/2, 2/(\beta\omega)) \\
&= \frac{1}{\Gamma(\beta/2)(2/(\beta\omega))^{\beta/2}} s_j^{\beta/2-1} e^{-\beta\omega s_j/2} \\
&= \Gamma(\beta/2)^{-1} (\beta\omega/2)^{\beta/2} s_j^{\beta/2-1} e^{-\beta\omega s_j/2} && (3.10)
\end{aligned}$$

As noted in [33], it is questionable to define a prior in terms of the observations, as done in Equations (3.5), (3.6), and (3.9), but this could be circumvented by normal-

izing the observations. It is left in the form shown for mathematical convenience.

From these priors, posterior distributions can be derived using Equation (2.1), beginning with the parameters of the model. Treating Equation (3.2) as the likelihood and Equation (3.7) as the prior yields

$$\begin{aligned}
\mathrm{P}(\mu_j|\tilde{x}_1,...,\tilde{x}_{n_j},s_j,\lambda,r) &\propto \mathrm{P}(\tilde{x}_1,...,\tilde{x}_{n_j}|\mu_j,s_j)\mathrm{P}(\mu_j|s_j,\lambda,r) \\
&= \mathrm{P}(\tilde{x}_1|\mu_j,s_j)\cdots\mathrm{P}(\tilde{x}_{n_j}|\mu_j,s_j)\mathrm{P}(\mu_j|s_j,\lambda,r) \\
&\propto \left(s_j^{n_j/2}e^{-s_j\sum_{i=1}^{n_j}(\tilde{x}_m-\mu_j)^2/2}\right)\left(r^{1/2}e^{-r(\mu_j-\lambda)^2/2}\right) \\
\mu_j|\tilde{x}_1,...,\tilde{x}_{n_j},s_j,\lambda,r &\sim \mathcal{N}\left(\left[s_j\sum_{m=1}^{n_j}\tilde{x}_m+\lambda r\right]\left[\frac{1}{s_jn_j+r}\right],\frac{1}{s_jn_j+r}\right).
\end{aligned} \tag{3.11}$$

Next, using Equation (3.10) as the prior for Equation (3.2) results in

$$\begin{aligned}
\mathrm{P}(s_j|\tilde{x}_1,...,\tilde{x}_{n_j},\mu_j,\lambda,r) &\propto \mathrm{P}(\tilde{x}_1,...,\tilde{x}_{n_j}|\mu_j,s_j)\mathrm{P}(s_j|\mu_j,\lambda,r) \\
&= \mathrm{P}(\tilde{x}_1|\mu_j,s_j)\cdots\mathrm{P}(\tilde{x}_{n_j}|\mu_j,sj)\mathrm{P}(s_j|\mu_j,\lambda,r) \\
&\propto \left(s_j^{n_j/2}e^{-s_j\sum_{m=1}^{n_j}(\tilde{x}_m-\mu_j)^2/2}\right)\left(s_j^{\beta/2-1}e^{-\beta\omega s_j/2}\right) \\
&= s_j^{\beta/2+n_j/2-1}e^{-s_j/\left[\sum_{m=1}^{n_j}(\tilde{x}_m-\mu_j)^2/2+\beta\omega/2\right]^{-1}} \\
s_j|\tilde{x}_1,...,\tilde{x}_{n_j},\mu_j,\lambda,r &\sim \mathcal{G}\left(\frac{\beta+n_j}{2},2\left[\sum_{m=1}^{n_j}(\tilde{x}_m-\mu_j)^2+\beta\omega\right]^{-1}\right).
\end{aligned} \tag{3.12}$$

To derive the posterior distributions for the parameters, the parameter values are treated as observations. Starting with the hyperparamteters λ and r, Equation (3.7) is treated as the likelihood and Equation (3.5) and (3.6) are used as the prior distributions for λ and r, respectively. These two posterior distributions are:

$$\begin{aligned}
\mathrm{P}(\lambda|\mu_1,...,\mu_k,r) &\propto \mathrm{P}(\mu_1,...,\mu_k|\lambda,r)\mathrm{P}(\lambda) \\
&= \mathrm{P}(\mu_1|\lambda,r)\cdots\mathrm{P}(\mu_k|\lambda,r)\mathrm{P}(\lambda) \\
&\propto \left(r^{1/2}e^{-r(\mu_j-\lambda)^2/2}\right)\cdots\left(r^{1/2}e^{-r(\mu_j-\lambda)^2/2}\right)\left(e^{\frac{-(\lambda-\mu_x)^2}{2\sigma_x^2}}\right)
\end{aligned}$$

$$
\begin{aligned}
\lambda|\mu_1,...,\mu_k,r &\sim \mathcal{N}\left(\left[r\sum_{j=1}^{k}\mu_j+\mu_x/\sigma_x^2\right]\left[\frac{1}{rk+1/\sigma_x^2}\right],\frac{1}{rk+1/\sigma_x^2}\right) \quad (3.13)\\
\mathrm{P}(r|\mu_1,...,\mu_k,\lambda) &\propto \mathrm{P}(\mu_1|\lambda,r)\cdots\mathrm{P}(\mu_k|\lambda,r)\mathrm{P}(r)\\
&\propto \left(r^{1/2}e^{-r(\mu_j-\lambda)^2/2}\right)\cdots\left(r^{1/2}e^{-r(\mu_j-\lambda)^2/2}\right)\left(r^{-1/2}e^{-r\sigma_x^2/2}\right)\\
&= r^{k/2+1/2-1}e^{-r/\left[\sum_{j=1}^{k}(\mu_j-\lambda)^2/2+\sigma_x^2/2\right]^{-1}}\\
r|\mu_1,...,\mu_k,\lambda &\sim \mathcal{G}\left(\frac{k+1}{2},2\left[\sum_{j=1}^{k}(\mu_j-\lambda)^2+\sigma_x^2\right]^{-1}\right) \quad (3.14)
\end{aligned}
$$

Finally, the last two posterior distributions are:

$$
\begin{aligned}
\mathrm{P}(\beta|s_1,...,s_k,\omega) &\propto \mathrm{P}(s_1|\beta,\omega)\cdots\mathrm{P}(s_1|\beta,\omega)\mathrm{P}(\beta)\\
&\propto \left(\prod_{j=1}^{k}\Gamma(\beta/2)^{-1}(\beta\omega/2)^{\beta/2}s_j^{\beta/2-1}e^{-\beta\omega s_j/2}\right)\left(\beta^{-3/2}e^{-1/(2\beta)}\right)\\
&\propto \Gamma(\beta/2)^{-k}(\beta\omega/2)^{(k\beta-3)/2}e^{-1/(2\beta)}\prod_{j=1}^{k}s_j^{\beta/2}e^{-\beta\omega s_j/2} \quad (3.15)\\
\mathrm{P}(\omega|s_1,...,s_k,\beta) &\propto \mathrm{P}(s_1|\beta,\omega)\cdots\mathrm{P}(s_1|\beta,\omega)\mathrm{P}(\omega)\\
&\propto \left(\prod_{j=1}^{k}\Gamma(\beta/2)^{-1}(\beta\omega/2)^{\beta/2}s_j^{\beta/2-1}e^{-\beta\omega s_j/2}\right)\left(\omega^{-1/2}e^{-\omega/(2\sigma_x^2)}\right)\\
&\propto \omega^{k\beta/2+1/2-1}e^{-\omega/\left[\beta/2\sum_{j=1}^{k}s_j+1/(2\sigma_x^2)\right]^{-1}}\\
\omega|s_1,...,s_k,\beta &\sim \mathcal{G}\left(\frac{k\beta+1}{2},2\left[\beta\sum_{j=1}^{k}s_j+1/\sigma_x^2\right]^{-1}\right) \quad (3.16)
\end{aligned}
$$

Unfortunately, Equation (3.15) does not simplify to a common distribution as the others did. Methods to sample from this distribution are discussed in Section 3.2.2.

The final step is to derive the posterior distribution for α given the indicators $c_1,...,c_n$. The symmetric Dirichlet prior, placed on mixing proportions, is written as

$$
f_{Dir}(\pi_1,...,\pi_k;\alpha/k,...,\alpha/k)=\frac{\Gamma(\alpha)}{\Gamma(\alpha/k)^k}\prod_{j=1}^{k}\pi_j^{\alpha/k-1}.
$$

It is then possible to integrate out the multinomially distributed indicators:

$$\begin{aligned} \mathrm{P}(c_1, ..., c_n|\alpha) &= \int p_m(c_1, ..., c_n; \pi_1, ..., \pi_k) f_{\mathrm{Dir}}(\pi_1, ..., \pi_k; \alpha) d(\pi_1, ..., \pi_k) \\ &= \frac{\Gamma(\alpha)}{\Gamma(\alpha/k)} \int \prod_{j=1}^{k} \pi_j^{n_j+\alpha/k-1} d\pi_j \\ &= \frac{\Gamma(\alpha)}{\Gamma(n+\alpha)} \prod_{j=1}^{k} \frac{\Gamma(n_j+\alpha/k)}{\Gamma(\alpha/k)} \end{aligned} \tag{3.17}$$

Placing a vague inverse gamma prior on $\alpha \sim \mathcal{G}^{-1}(1/2, 1/2)$ and combing with Equation (3.17) yields

$$\mathrm{P}(n_1, ..., n_k|\alpha) = \frac{\alpha^k \Gamma(\alpha)}{\Gamma(n+\alpha)}$$

and the posterior distribution for α is

$$\mathrm{P}(\alpha|k, n) = \frac{\Gamma(\alpha)\alpha^{k-2}e^{-1/\alpha}}{\Gamma(n+\alpha)}. \tag{3.18}$$

To extend the methodology to a second dimension, the simplest approach is to maintain a second set of all parameters and hyperparameters. By taking this approach it is assumed that the first and second dimensions are uncorrelated. While there are no significant issues with this assumption here, in an application where such an assumption would be unreasonable, multivariate Gaussians (Section A.1) and Wishart distributions (Section A.4) can be used to replace univariate Gaussians and gamma distributions, respectively.

A Simpler Infinite Gaussian Mixture Model

It is fair to ask if the formulation of Section 3.2.2 is overly complex. The previous approach had distributions over parameters of prior distributions, leading one to wonder if this is truly necessary and if having vague priors, especially over μ_j, is sufficient. Formally, consider the following priors:

$$\lambda = \mu_x \tag{3.19}$$

$$
\begin{aligned}
r &= \sigma_x^{-2} && (3.20)\\
\mathrm{P}(\mu_j|\lambda, r) &= f_{\mathcal{N}}(\mu_j; \lambda, 1/r) \\
&= \frac{r^{1/2}}{\sqrt{2\pi}} e^{-r(\mu_j-\lambda)^2/2} \propto r^{1/2} e^{-r(\mu_j-\lambda)^2/2} && (3.21)\\
\mathrm{P}(\beta^{-1}) &= f_{\mathcal{G}}(\beta^{-1}; 1/2, 2) \\
\Rightarrow \mathrm{P}(\beta) &= f_{\mathcal{G}^{-1}}(\beta; 1/2, 1/2) \\
&= \frac{(1/2)^{1/2}}{\Gamma(1/2)} \beta^{-1/2-1} e^{-1/(2\beta)} \propto \beta^{-3/2} e^{-1/(2\beta)} && (3.22)\\
\omega &= \frac{1}{\beta} && (3.23)\\
\mathrm{P}(s_j|\beta, \omega) &= f_{\mathcal{G}}(s_j; \beta/2, 2/(\beta\omega)) \\
&= \frac{1}{\Gamma(\beta/2)(2)^{\beta/2}} s_j^{\beta/2-1} e^{-s_j/2} \\
&= \Gamma(\beta/2)^{-1} (1/2)^{\beta/2} s_j^{\beta/2-1} e^{-s_j/2} && (3.24)
\end{aligned}
$$

This simpler model proposes that having a vague prior over μ_j and a distribution over only one parameter of s_j's prior[4]. This simpler model is compared to the complete IGMM in Section 4.2.1.

Sampling Procedure

Due to the analytical intractability of exactly inferring $\mu_1, ..., \mu_k, s_1, ..., s_k, c_1, ..., c_n$ and α from the distributions described in Sections 3.2.2, a MCMC (Section 2.3.4) method is used. The sampling procedure used to generated the results in Chapter 4 is shown in Algorithm 6. Equations (3.15) and (3.18) both require a bit more effort to sample from since they are not in a standard form. Rejection sampling, as discussed in Section 2.3.4, is used due to the ease of implementation but adaptive rejection sampling [18] may also be used to improve performance.

[4]Equation (3.24) is actually the chi-squared distribution, a common distribution used as a prior for an unknown variance.

Algorithm 6 IGMM Inference

Draw $\lambda \sim \mathcal{N}(\mu_x, \sigma_x^2)$
Draw $r \sim \mathcal{G}(1/2, \sigma_x^{-2}/2)$
Draw $\beta \sim \mathcal{G}^{-1}(1/2, 1/2)$
Draw $\omega \sim \mathcal{G}(1/2, 2\sigma_x^2)$
Draw $\mu_1 \sim \mathcal{N}(\lambda, 1/r)$
Draw $s_1 \sim \mathcal{G}(\beta, (\beta\omega)^{-1})$
Draw $\alpha \sim \mathcal{G}^{-1}(1/2, 1/2)$
k = 1
for sweep = 1 to # of sweeps **do**
  for $j = 1$ to k **do**
    Draw $\mu_j \sim \mu_j | \tilde{x}_1, ..., \tilde{x}_{n_j}, s_j, \lambda, r$
    Draw $s_j \sim s_j | \tilde{x}_1, ..., \tilde{x}_{n_j}, \mu_j, \lambda, r$
  end for
  Draw $r \sim r | \mu_1, ..., \mu_k, \lambda$
  Draw $\lambda \sim \lambda | \mu_1, ..., \mu_k, r$
  for i randomly chosen from $1, ..., n$ **do**
    $c_{old} = c_i$
    Draw $c_i \sim c_i | c_1, ..., c_{i-1}, c_{i+1}, ..., c_n, \theta_1, ..., \theta_k, \alpha$
    if $c_i > k$ **then**
      Draw $\theta_{k+1} \sim \theta_{k+1} | x_i, H$
      $k = k + 1$
    end if
    if table c_{old} is empty **then**
      Remove $\theta_{c_{old}}$
      $k = k - 1$
    end if
  end for
  for $j = 1$ to k **do**
    Draw $\theta_j \sim \theta_j | c_1, ..., c_n, \theta_1, ..., \theta_k, \alpha$
  end for
  Draw $\alpha \sim \alpha | n, k$
end for

3.2.3 The Dirichlet Process Prior Over Paths

Each destination found using the methodology in Section 3.2.2 indexes a distribution over paths leading to it. Two paths, each defined as a sequence of road segments, are said to be equivalent if they are exactly the same. Consider a parametric approach to modeling this distribution where a probability mass is maintained over every path. For all paths $p_{j,i}$ indexed by destination j,

$$p_{j,i} \sim q, \quad i = 1, ..., l \tag{3.25}$$

where $q = \{q_1, ..., q_l\}$ and l is the total number of paths. These values can be approximated from the data by the expression

$$q_i = \frac{n_{j,i}}{n_j} \tag{3.26}$$

where $n_{j,i}$ is the number of times path i was taken to destination j. Figure 3-4 shows a small sampling of paths that use the blue circled road segment. Imagining the sheer

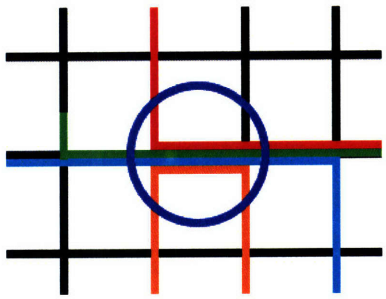

Figure 3-4: A small sample of the paths traveling through the circled road segment.

number of paths that travel on this one road segment, then realizing that it is just one road segment in the whole city, should begin to give a sense for the number of parameters needed to maintain a probability of each path (e.g., l is extremely large). This means Equation (3.26) will need an immense amount of data. To circumvent maintaining these nearly infinite number of probability masses, this thesis assumes that q is Dirichlet process distributed.

The best way of understanding this part is through the CRP. Each unique path

is the parameter for a table with the number of customers sitting at the table corresponding to the number of times the path was taken. The base distribution, H, is a uniform distribution over all paths. So the probability distribution[5] of path $p_{j,n_j+1} = (r^1_{n_j+1}, ..., r^{m_{n_j+1}}_{n_j+1})$ where $r^i_{n_j+1}$ is the ith road segment of p_{j,n_j+1} is

$$p_{j,n_j+1}|p_{j,1}, ..., p_{j,n_j}, \gamma_j, H \sim \sum_{i=1}^{l'} \frac{n_{j,i}}{n_j + \gamma_j} + \frac{\gamma_j}{n_j + \gamma_j} H \tag{3.27}$$

where l' is the number of unique paths in the data ($l' << l$). Equation (3.27) shows how the CRP maintains a distribution similar to Equation (3.25) with the understanding that not all the paths will show up in the data. It does this by not explicitly specifying the q_j values for each path and instead asserting that $q_j \sim \mathcal{DP}(\gamma_j, H)$. This implies new paths will be drawn from H with probability $\frac{\gamma_j}{n_j+\gamma_j}$. In Section 2.4.2 it was noted that there is a nonzero probability that two tables may take the same parameter if H is discrete. Therefore it may be possible, in the sense of generating a series of paths from the CRP, of having a customer sit at a new table but draw a path that is also at another table.

To learn the DP distribution over paths, the path (customer) is assigned to a nonempty table if the path matches the path (parameter) of the table. If it differs from all of the parameters at occupied tables, the path of data is "seated" at a new table. While it is possible that two identical paths were drawn for two different tables, it is assumed this does not happen in the model learning. This is a valid assumption since the DP with the highest likelihood is what this thesis is interested in. The domain is so large that it is far more likely a customer sat at a previously occupied table with that parameter, than sat at a new table and drew the same parameter. Once all the training data has been assigned to tables, γ_j that best explains the seating arrangement is calculated. Unlike the method for finding α in Section 3.2.2, where α is sampled from its distribution, for the DP over path distributions, the maximum *a posterior* estimate of γ_j is used. This is calculated by finding the γ_j which maximizes Equation (3.18) (setting $\alpha = \gamma_j$). This is done because there are no latent variables

[5]A series of road segments is written this way because the order matters (i.e., $(r_1, r_2) \neq (r_2, r_1)$).

in this part of the model. Using the methodology described in this paragraph, there is no ambiguity about which customers sit at which table, unlike Section 3.2.2.

3.2.4 Path Inference and Prediction

Reverting back to the model where a probability mass for each path is maintained leads to the methodology for solving path inference and prediction from the DP-based model. The probability of path $p_{j,i}$, given an observation of the road segment series $(r_1, ..., r_m)$, is formally written as

$$\mathrm{P}(p_{j,i}|(r_1, ..., r_m), n_{j,1}, ..., n_{j,l}) = \begin{cases} \frac{n_{j,i}}{n_j} & \text{if } (r_1, ..., r_m) \in p_{j,i} \\ 0 & \text{otherwise} \end{cases} \tag{3.28}$$

where $n_{j,i}$ is the number of times $p_{j,i}$ was observed in the data $p_{j,1}, ..., p_{j,n_j}$, indexed by destination j.

After observing a vehicle on a series of road segments, the probability of the vehicle's next action is given by its probability of taking the road segment to which the action leads. Explicitly,

$$\begin{aligned} &\mathrm{P}(r|(r_1, ..., r_m), p_{j,1}, ..., p_{j,l}, n_{j,1}, ..., n_{j:l}) \\ &\qquad \propto \mathrm{P}(p_{j,1}, ..., p_{j,l}|r, (r_1, ..., r_m), n_{j,1}, ..., n_{j:l})\mathrm{P}(r|(r_1, ..., r_m), n_{j,1}, ..., n_{j:l}) \\ &\qquad \propto \sum_{i=1}^{l} \mathrm{P}(p_{j,i}|(r_1, ..., r_m, r), n_{j,1}, ..., n_{j:l}) \\ &\qquad \propto \sum_{i:(r_1,...,r_m,r)\in p_{j,i}} n_{j,i} \end{aligned} \tag{3.29}$$

since it is assumed all actions (straight, left, and right) are equally likely. Equation (3.29) shows that if the data doesn't show a complete picture of how often paths are taken, the prediction accuracy will severely suffer. Specifically, $n_{j,i}$ should rarely be zero, as most paths are eventually taken (no matter how infrequently) but enough data must be seen to capture that.

This problem of over-fitting the data can be overcome by using a DP prior over

paths. The purpose of the remainder of this section is to rewrite Equation (3.28) and (3.29) by using Equation (3.27) as the model. To begin,

$$\mathrm{P}(p_{j,i}|n_{j,1},...,n_{j,l'},\gamma_j,H) \propto \begin{cases} n_{j,i} & \text{if } i \leq l' \\ \gamma_j & \text{if } i > l' \end{cases} \tag{3.30}$$

is the prior over road segments, a slight approximation of Equation (3.27). It is a slight approximation since $p_{j,i}$ should, technically, be allowed to be drawn again from H. Since the chance of this happening is so small, $1/l$, it is ignored. This is equivalent to saying that once a path is assigned to a table it is not allowed to be drawn for another. For $i \leq l'$, the distribution over paths

$$\mathrm{P}(p_{j,i}|(r_1,...,r_m),n_{j,1},...,n_{j,l'},\gamma_j,H) \propto \begin{cases} n_{j,i} & \text{if } (r_1,...,r_m) \in p_{j,i} \\ 0 & \text{otherwise} \end{cases}$$

stays the same, and for $i > l'$

$$\mathrm{P}(p_{j,i}|(r_1,...,r_m),n_{j,1},...,n_{j,l'},\gamma_j,H) \propto \begin{cases} \gamma_j & \text{if } (r_1,...,r_m) \in p_{j,i} \\ 0 & \text{otherwise} \end{cases}.$$

The distribution over the next road segment is then

$$\begin{aligned} &\mathrm{P}(r|(r_1,...,r_m),p_{j,1},...,p_{j,l'},n_{j,1},...,n_{j,l'},\gamma_j,H) \\ &\quad \propto \mathrm{P}(p_{j,1},...,p_{j,l'}|r,(r_1,...,r_m),n_{j,1},...,n_{j,l'},\gamma_j,H)\mathrm{P}(r|(r_1,...,r_m),n_{j,1},...,n_{j,l'},\gamma_j,H) \\ &\quad \propto \sum_{i=1}^{l'} \mathrm{P}(p_{j,i}|(r_1,...,r_m,r),n_{j,1},...,n_{j,l'},\gamma_j,H) \\ &\qquad + \sum_{i=l'+1}^{l} \mathrm{P}(p_{j,i}|(r_1,...,r_m,r),n,\gamma_j,H) \\ &\quad \propto \sum_{i:(r_1,...,r_m,r)\in p_{j,i}} n_{j,i} + \frac{\gamma_j}{3}. \end{aligned} \tag{3.31}$$

The last term in Equation (3.31) represents the probability of a path not seen in the data traversing $(r_1,...,r_m,r)$. To understand this term, think back to the CRP.

A new path is drawn with probability proportional to γ_j and, assuming there are three possible actions (straight, left, and right), the novel probability mass is split evenly amongst them (having no reason to believe one action is more likely than the others). It should not surprise the reader that H does not appear in the final expression, since, thinking back to Figure 3-2, the distribution over paths is drawn from a GEM distribution with parameter γ_j. This is equivalent to a draw from a DP with parameter γ_j and a base measure that is uniform over all natural numbers (as was used here). To incorporate the paths assigned to all destinations, a summation is performed over Equation (3.31) for $j = 1, ..., k$.

While this section thus far has discussed predicting the next road segment, Section 1.2 discussed how a driver model should also be able to predict destinations. The probability of partial path p going to destination j is

$$\begin{aligned} \mathrm{P}(j|p,a) &\propto \mathrm{P}(p|j,a)\mathrm{P}(j|a). \\ &= \left(\frac{\gamma_j}{n_j+\gamma_j} + \sum_{i:p\in p_{j,i}} \frac{n_{j,i}}{n_j+\gamma_j}\right)\left(\frac{n_j}{n+\alpha}\right) \end{aligned} \tag{3.32}$$

where $a = \{p_{1,1}, ..., p_{1,l'}, ...p_{k,l'}, n_{1,1}, ..., n_{1,l'}, ...n_{k,l'}, \gamma_1, ..., \gamma_k\}$. To understand where this expression comes from, consider the two terms on the right hand side independently. The first term comes from Equation (3.30) and can be understood using the following intuition: p may either be part of a novel path heading to j with probability $\frac{\gamma_j}{n_j+\gamma_j}$ or may be part of known path $p_{j,i}$ (if it is consistent with $p_{j,i}$) with probability $\frac{n_{j,i}}{n_j+\gamma_j}$. The second term is the "weight" of the destination. This weight can be calculated by thinking back to the CRP for the infinite Gaussian mixture model. In this CRP, a known parameter (destination j) is drawn again with probability equal to $\frac{n_j}{n+\alpha}$.

Chapter 4

Experimental Results

4.1 Chapter Overview

The two problems of finding destinations and learning the DP over path distributions to the destinations are first tested in separate simulations in Sections 4.2.1 and 4.2.2, respectively. This is followed by Section 4.2.3, where simulation results from the full model are shown. Section 4.3 extends the results from simulation to real GPS data of taxis driving around Boston.

4.2 Simulation

4.2.1 Finding Destinations

To validate the IGMM, it was compared against the following three models:

1. the randomly chosen generative model from which the data was drawn,

2. expectation maximization (Section 2.3.3) given the number of clusters that were used in the generative model[1],

3. the simplified IGMM.

[1]Patrick Tsui's expectation maximization code was used which can be found at: www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=8636&objectType=file

For this test Gaussian distributions were generated with mean and variance chosen uniformly on the intervals [-40,40] and [0,25], respectively. Before the results from the tests are shown, a sample run with $k = 3$ clusters is plotted in Figure 4-1. This

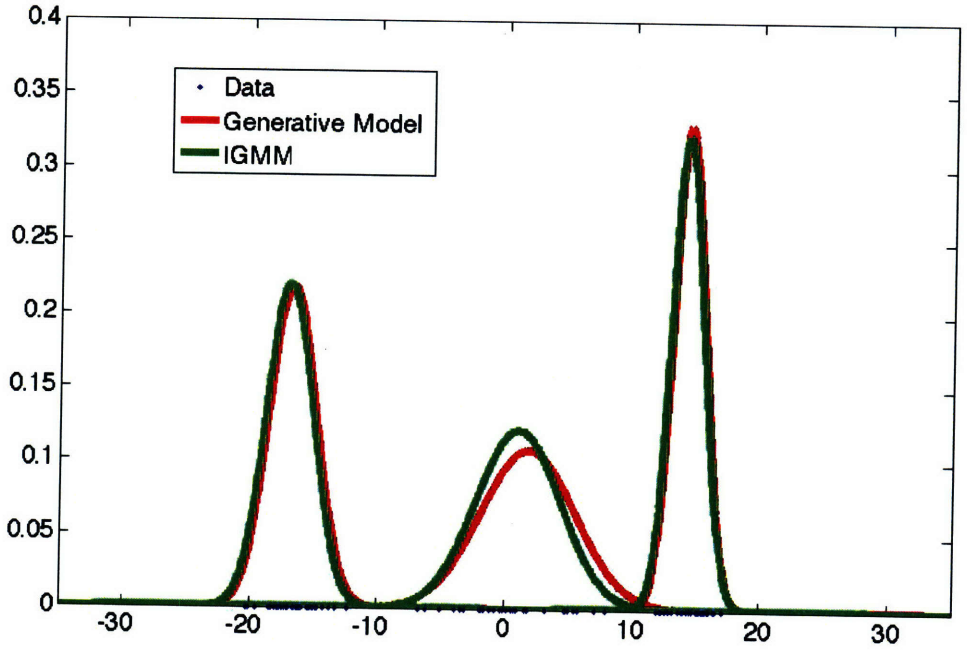

Figure 4-1: A sample generative and learned model with $k = 3$.

figure clearly shows the IGMM (green) resembling the generative model (red). It is important to note that while it is promising that the IGMM learned a model close to the generative one, a model with more or less clusters may often explain the data set better, especially as the number of clusters increase. For example, Figure 4-2 demonstrates how the IGMM learns a model with a higher likelihood. In this figure the IGMM decided on three clusters as opposed to the six cluster generative model.

The lines and error bars in Figure 4-3 are the results from 45 tests. Each test was run using data generated along a single dimension with five different trials at each $k = 2, ..., 10$. In Figure 4-3, the log-likelihood, $\ln(\mathrm{P}(E|\theta))$ (Section 2.3), is used to simplify calculations. Since the log function is monotonic, there is no difference in drawing conclusions from either two models' likelihoods or log-likelihoods to see which model is best. The plot shows the IGMM having a higher log-likelihood than the other

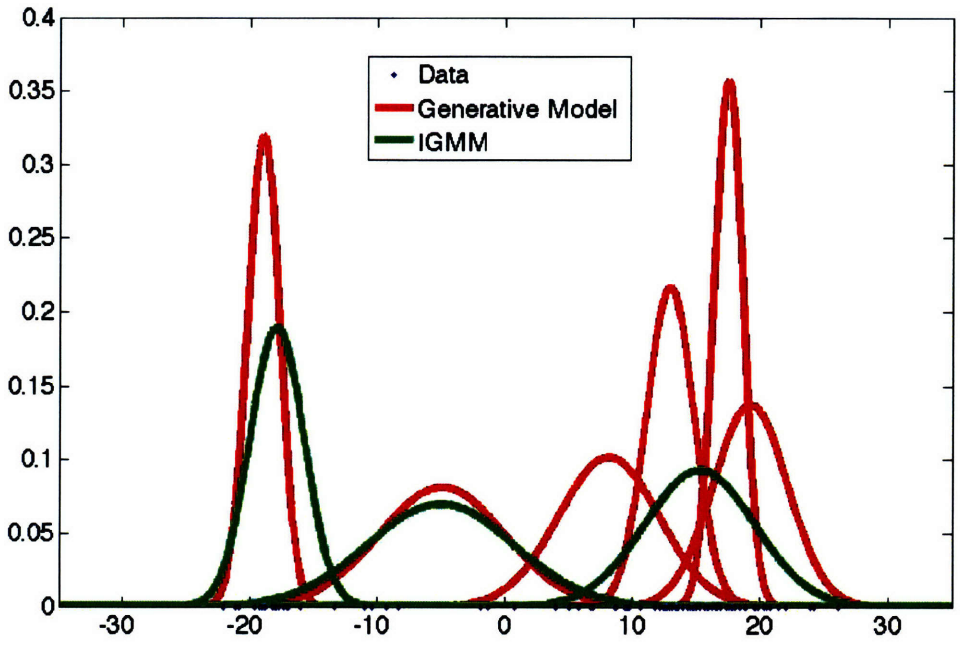

Figure 4-2: A sample generative and learned model with $k = 6$.

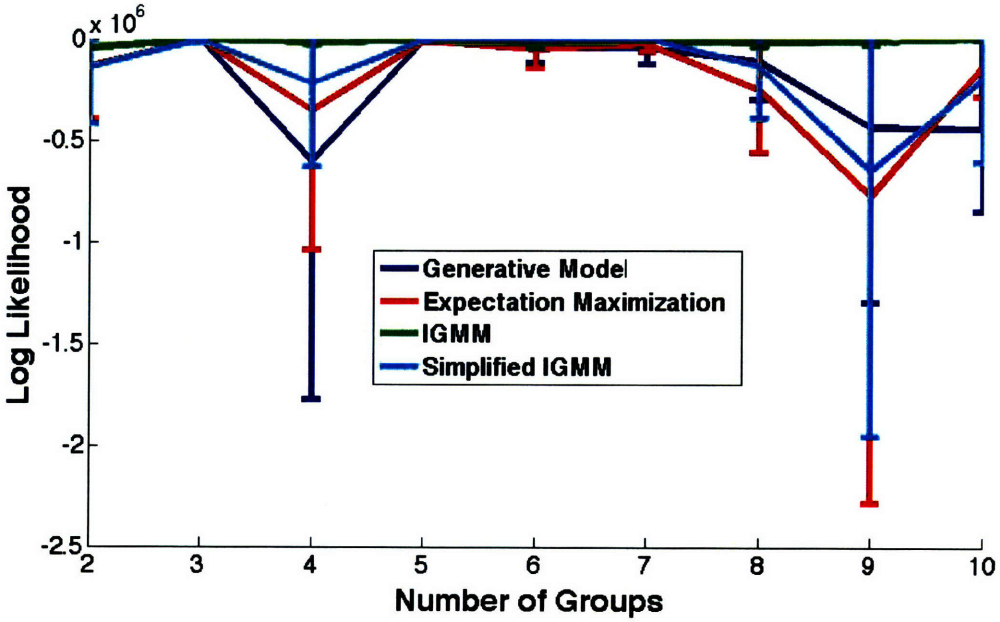

Figure 4-3: A comparison of different clustering techniques.

three models. To understand why this is, consider Figures 4-1 and 4-2. With data resembling Figure 4-1, all four approaches tended to have similar likelihoods. The real differences arose when generated clusters fell close to one another, as in Figure 4-2. In these tests, both the IGMM and simplified IGMM models would, in general, group clusters that fell close together as a single cluster. EM was explicitly told the number of clusters to find, so if it were to group two clusters as one, it would need to find a spot where another cluster could fit best. The results also show that the added complexity of the full IGMM seems to be worth it, as compared to the simplified IGMM.

Finally, since this will actually be run on two dimensional data, Figure 4-4 demonstrates the kind of clustering that is done to find destinations.

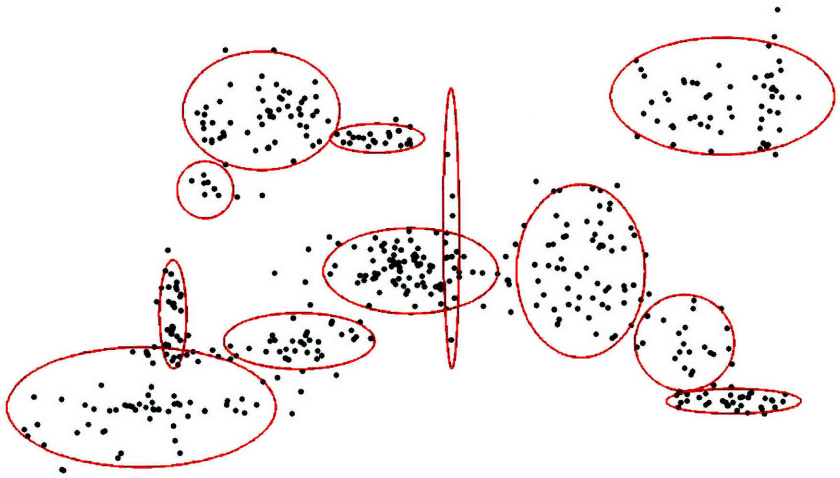

Figure 4-4: A demonstration of two dimensional synthetic data clustering.

4.2.2 Learning a Distribution Over Paths to a Destination

The second part of the model is composed of the distribution over paths that lead to a specific destination. This test is designed to assist in understanding how the learned distribution is affected by more data. Figure 4-5 shows the 10x10 city that

was used to generate the paths of data. Four sample paths (yellow), and the three road segments that are the focus of Figure 4-6 (blue, green, and red) are shown in the figure. A path of data is generated by picking a starting segment uniformly over the

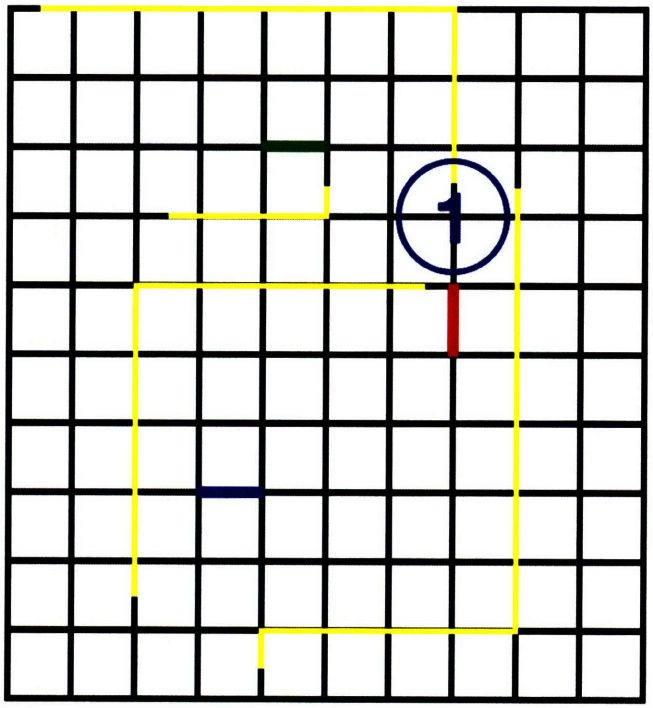

Figure 4-5: The 10x10 simulated city used as a sanity check for modeling the distribution over path distributions as a DP.

city, a goal segment from the distribution on location 1, and then the shortest path is found between the start and goal. When two paths are encountered with the same length, one of them is randomly chosen. The DP-based behavior model is compared to a Markov model (Section 2.3.2) with a state as a road segment and transitions that cause the state to change to any of the neighboring road segments. From 10,000 paths of training data, Figure 4-6 shows the learned probabilities for three different road segments (blue, green, and red) taking the actions straight, left, and right from the Markov model (dashed) and the DP-based behavior model (solid).

These results show that the Markov model and the DP-based model both tend to approach the same values. The DP-based model lagging behind the Markov model is a result of γ_j in Equation (3.31). As more data is seen, γ_j shrinks compared to $n_{j,i}$ and it approaches the Markov model probabilities.

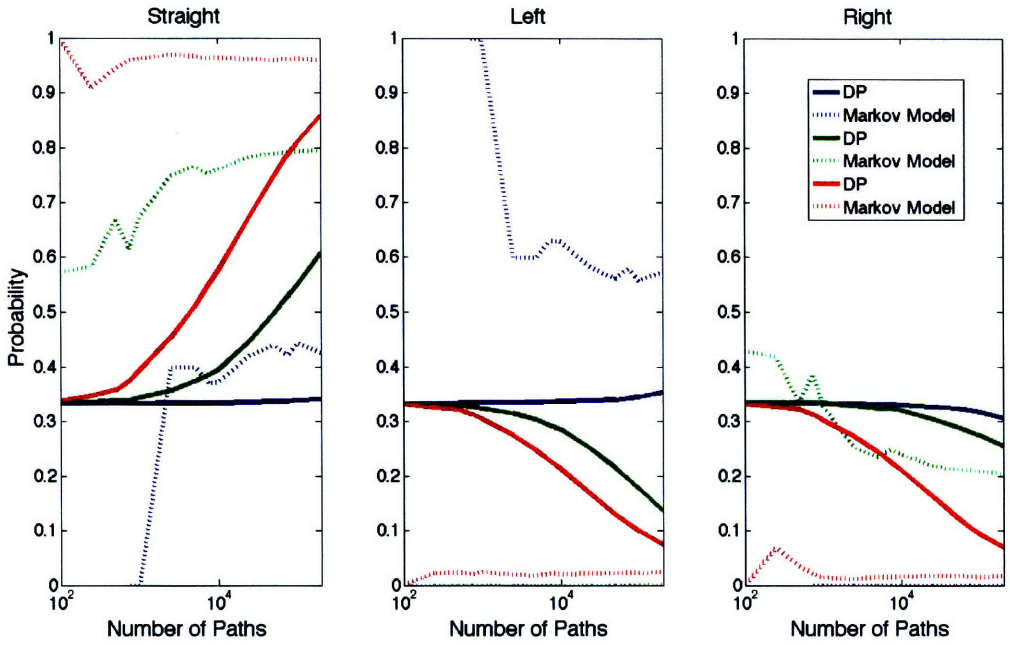

Figure 4-6: Markov model and DP-based model predictions for straight, left, and right turns for the similarly colored road segments in Figure 4-5.

At first glance these results may give the impression that this thesis is using powerful machinery to accomplish the same thing as a simple Markov model. This simulation is extremely simple, so a Markov model should perform well. The fact that the DP-based model approaches the Markov model probabilities should be viewed as a sanity check. Looking at the blue road segment, an argument can be made that the DP-based model does not over-fit the data as the Markov model does. The DP-based model essentially "understands" how limited a picture the data gives. The blue road segment is an example of a road segment used infrequently and how the model is more "hesitant" to make any strong predictions. Seeing the large fluctuations in the blue Markov model probabilities lends validity to the DP-based model's relatively small change in distribution. Comparing the red and green lines shows the effect of the DP-based model's understanding of how well it has learned the underlying structure. Both of these road segments have similar probabilities, but the red segment showed up far more frequently, meaning the car behavior associated with it is better understood.

One of the main assertions of this thesis is that the paths drivers choose are complex due to their individual understanding of the domain. This understanding

translates into preferences toward or away from road segments that very likely differ from the shortest time path. Due to there being no such underlying behavior of the vehicles in this simulation and only one destination, it is unsurprising that the two models would make similar predictions.

To adequately compare the two models, it is also necessary to look at how long each of them took for learning. Looking at Figure 4-7, the Markov model, as expected, is linear in the number of paths. This is due to it having only to look at each path once and store the counts. In a naive implementation, the DP-based model has polynomial

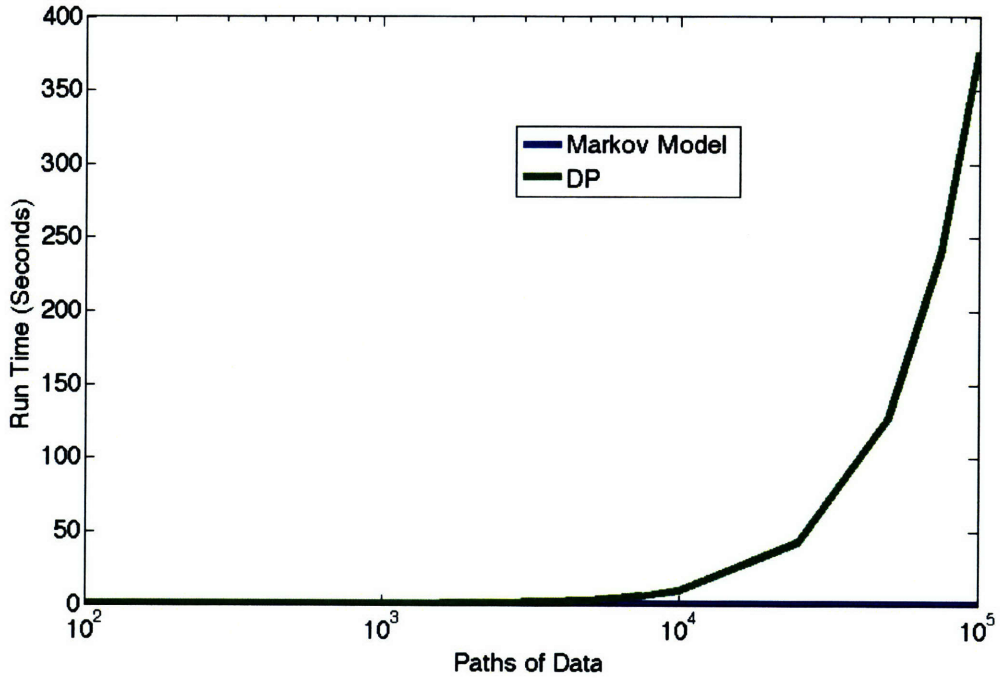

Figure 4-7: A comparison of learning times for the Markov model and the DP-based model.

learning time, since for every path of data, it needs to check all previously seen paths.

4.2.3 The Full Driver Behavior Model

This section tests the full DP-based behavior model. First destinations are clustered and then a DP over path distributions are learned for each cluster. The city shown in Figure 4-8 is used to test how well this model can learn underlying structure (as opposed to the city used in the previous section which had very little underlying

structure). In this diagram the drivers understand that the green road is faster and

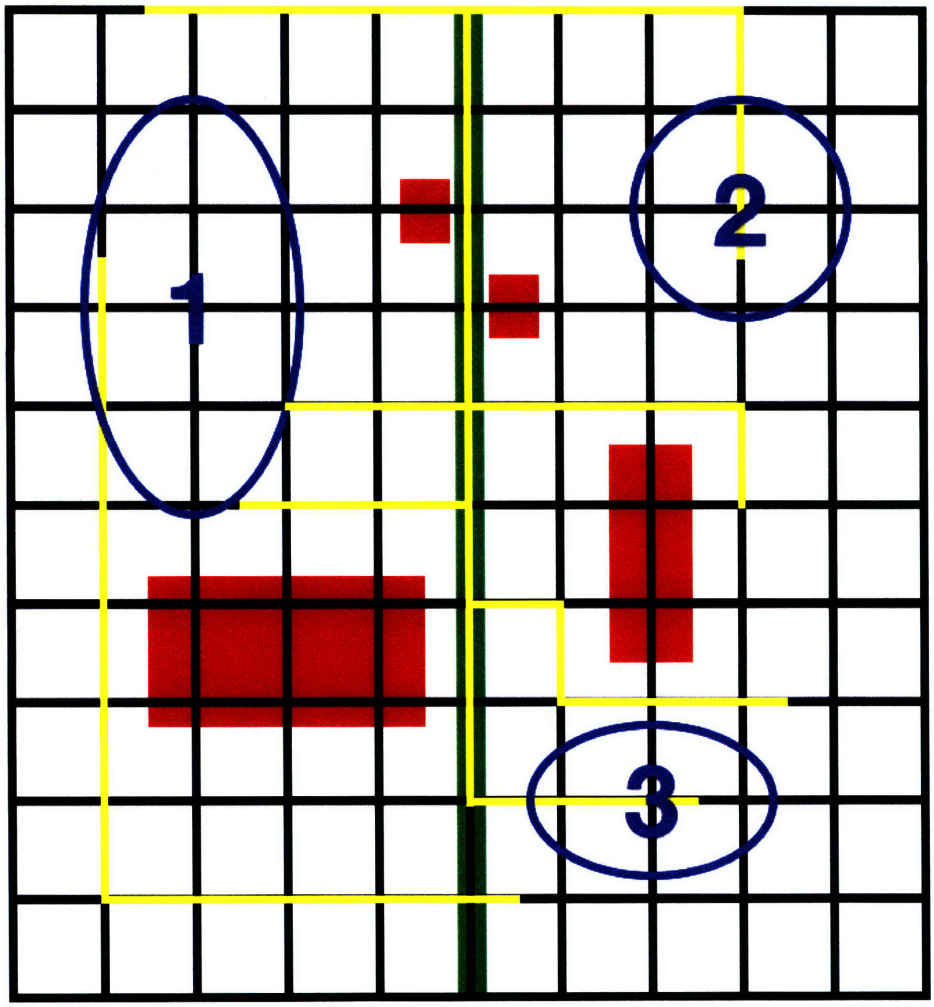

Figure 4-8: The 10x10 simulated city used to the test the full model.

that the red areas are slower. As such, the data generated from this city will take that into account.

The DP-based model is compared to the same Markov model as described in Section 4.2.2. The models were compared using next road segment prediction accuracies and goal location predictions, given a partial path. Figures 4-9, 4-10, and 4-11 show the prediction accuracy for a variety of city sizes and training data. These tests demonstrate that the DP-based model does significantly better than the Markov model. In some of the plots, it appears the Markov model has an increased accuracy as more of the path is seen. While this may seem counterintuitive since the Markov model disregards the past, the increase in accuracy is not a result of the model but

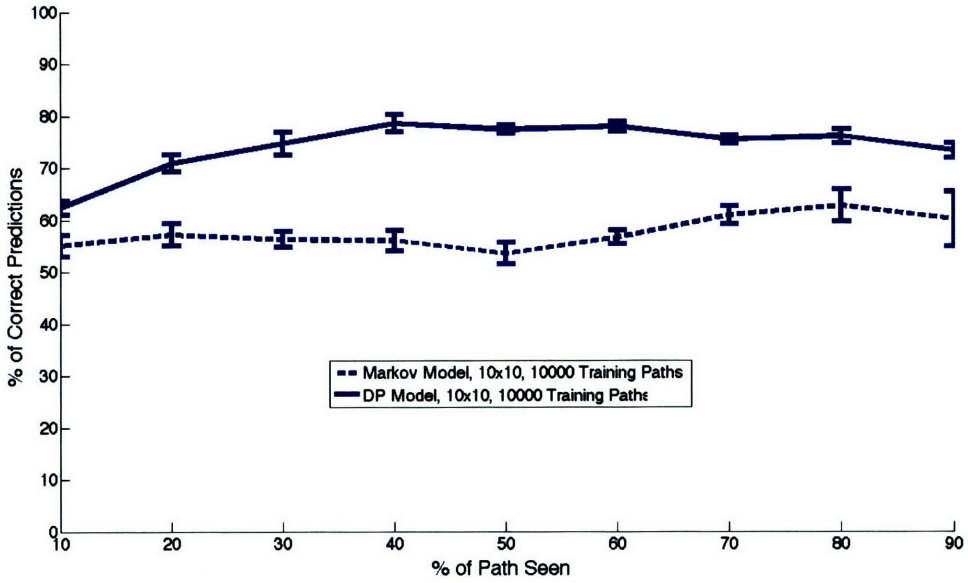

Figure 4-9: The results comparing the next road segment prediction accuracy of the Markov model and the DP-based model in a 10x10 city with 10,000 paths of training data.

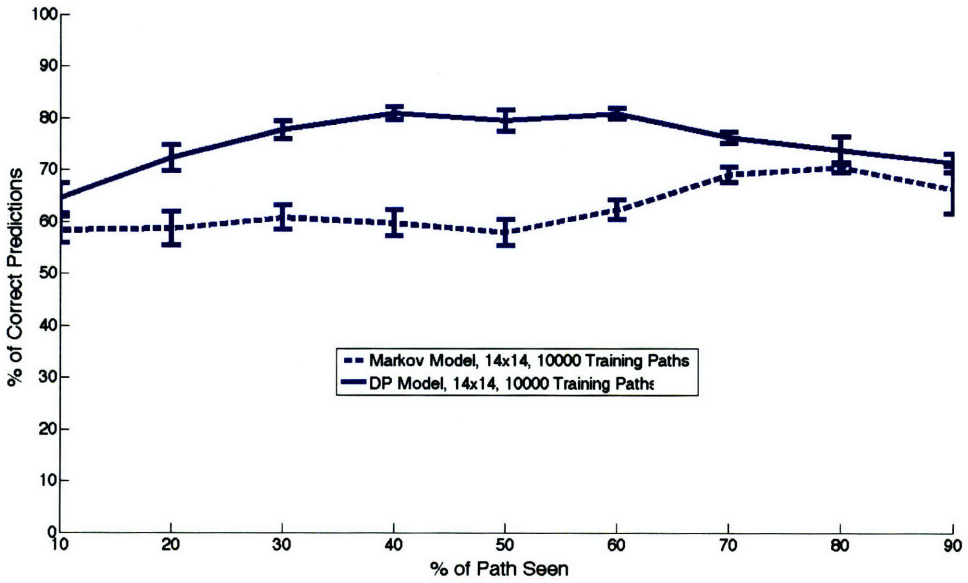

Figure 4-10: The results comparing the next road segment prediction accuracy of the Markov model and the DP-based model in a 14x14 city with 10,000 paths of training data.

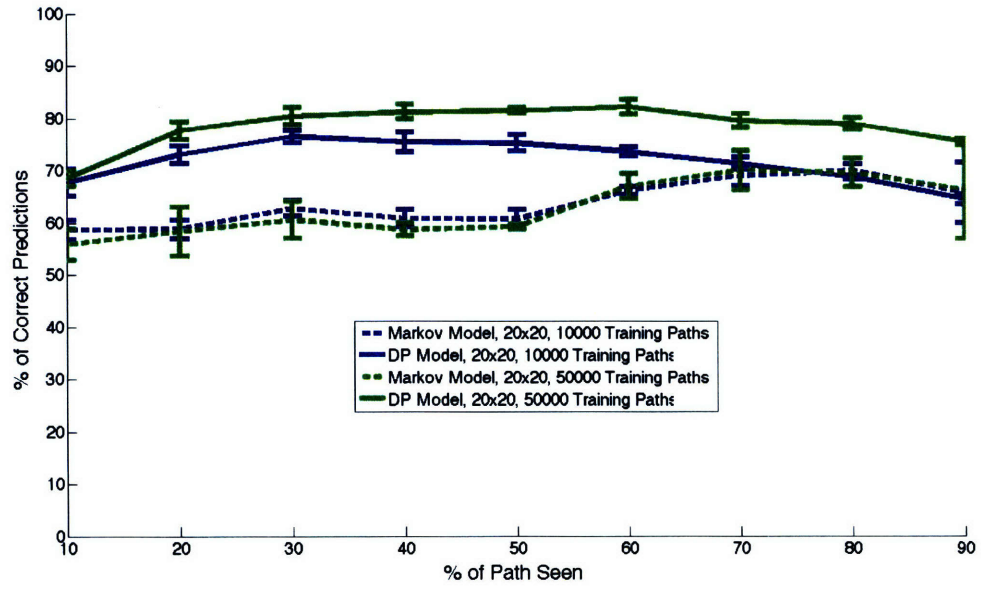

Figure 4-11: The results comparing the next road segment prediction accuracy of the Markov model and the DP-based model in a 20x20 city with 10,000 paths of training data (blue) and 50,000 paths of training data (green).

the test becoming easier. Referring back to the yellow paths drawn on Figure 4-8, as a path gets closer to an end location, the prediction gets easier. This accounts for the increase in the Markov model prediction accuracy.

The argument behind using a DP distribution over paths is that any latent structure can be captured. Again, looking at the yellow paths, it seems most of the latent structure would occur around the middle of the paths, when the DP-based model performs the best. From these plots one can imagine that the more underlying structure a domain has, the better the DP-based model will perform. In practice, it seems unreasonable to expect to see more than 30% of a driver's path.

Lastly, one of the more interesting observations is gained from Figure 4-11. The differences between the two sets of colored lines show that for five times more data, the Markov model sees a slight, if any, improvement. On the other hand it has a noticeable effect on the DP-based model. This implies that when implemented on a real system, the longer it is in use the better the predictions become.

Figures 4-12 and 4-13 are demonstrations of how the DP-based model not only can make predictions about future actions, but also make predictions about destinations.

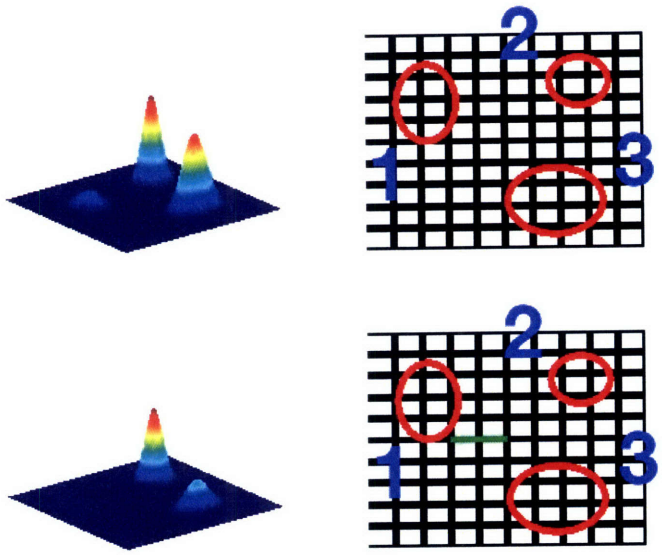

Figure 4-12: The prior destination distribution with no observed road segments (top) and two observed road segments heading right (bottom).

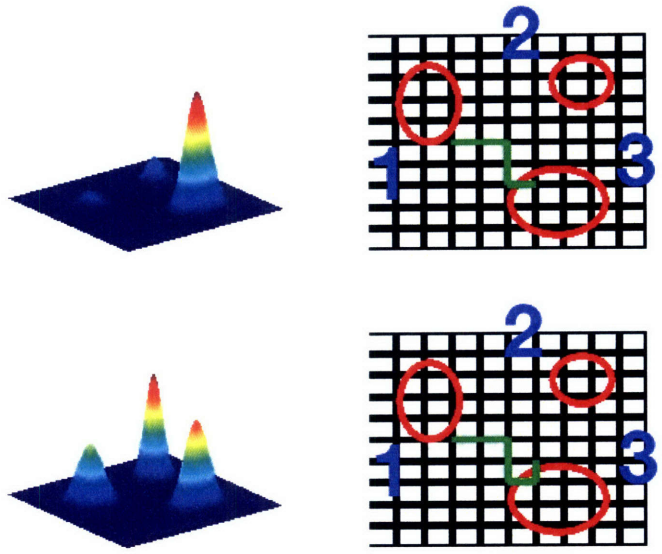

Figure 4-13: The posterior destination distribution after taking a very revealing turn (top) and if the driver were to take an unexpected turn (bottom).

The mixing proportions used to generated the data were $\pi_1 = 15\%$, $\pi_2 = 35\%$, and $\pi_3 = 50\%$ and the learned model inferred $\gamma_1 = 2752$, $\gamma_2 = 797$, and $\gamma_3 = 1205$, where γ_j is the DP parameter associated with the DP indexed by location j (the location numbers are the same as assigned in Figure 4-8). In the figures, the right columns of the plots show a sequence of observations about a vehicle and the left column (generated using Equation (3.32)) shows the model's predicted ending location distribution for the vehicle.

The top set of plots in Figure 4-12 shows the prior distribution on stopping locations (nothing about the vehicle has been observed). The vehicle is then observed on two road segments (green) heading to the right. Looking at the bottom right plot, it is fair to assume the vehicle could either be heading to destination 2 or 3. The destination distributions look reasonable, considering there are some paths leading to location 3 that have already been passed where as nearly all of the paths that head to 2 are still valid. Once it turns right, as in the top right plot Figure 4-13, the model strongly predicts it will head to location 3. Despite this strong prediction in the top set of plots, something interesting happens once the vehicle unexpectedly turns left, as shown in the bottom right plot (it now becomes a novel path). Once the path stops matching any known path, Equation (3.32) becomes a function of only $\gamma_1, \gamma_2, \gamma_3, \pi_1, \pi_2$, and π_3, and the bottom left plot reflects this.

Another interesting behavior of the model is seen from comparing the bottom distribution of Figure 4-12 and the top distribution of Figure 4-13. While the distributions over destinations 2 and 3 behave as expected, the probability that the vehicle is heading to destination 1 actually increases as more of the green path is seen. As the observed path becomes longer it fits fewer paths from the training data, so it gradually becomes more likely to be a novel path to destination 1 (due to the large value of γ_1). The model realizes this path was actually novel in the bottom of Figure 4-13 and it becomes clear how mixing proportions and concentration parameters effect end destination prediction.

4.3 Boston GPS Data

The CarTel project[2] at MIT has collected an extremely large amount of data from GPS receivers placed inside taxis. Fortunately, they have allowed their data to be used in this thesis to test the methodology on real world data.

The data was recorded as time-stamped GPS locations, from which sequences of road segments must be extracted. The data is divided into a path any time a taxi has moved less than 100 meters in one and half minutes or GPS measurements jumped over 500 meters. While there are certainly taxi stops that last less than a minute and a half, any shorter time window will make it impossible to distinguish stop lights or traffic from the end of a path. In fact, in looking at the data, it seems the one and a half minute stopping window did pick up some traffic. This, combined with attempting to match noisy GPS to road segments, adds a fair amount of uncertainty to the problem for which the model has not been designed. The model, in its current form, relies on being able to match sequences of road segments exactly, so any error in extracting the sequence of road segments (e.g., briefly snapping the data to an incorrect road segment) will cause the model to think that two identical paths are different. In an attempt to clean up the data, rather than dealing with sequences of road segments, paths will be represented as sequences of waypoints along the roads. While this was by no means perfect, it reduced the error a significant amount, and as far as learning a distribution of paths is concerned, both methods are just sequences of integers. Even with this noise, the DP-based behavior model was applied to the data to see if perhaps any of the latent structure can be captured.

From 18,000,000 GPS data points taken over a month and a half period, 22,724 paths of waypoints were extracted. Figure 4-14 shows the waypoints (blue), clusters that were learned (one standard deviation shown in red) from the stopping locations of each path (black). Anyone familiar with the greater Boston area will agree that the clusters found by the model are unsurprising. It is especially reassuring that the airport (location 6) was found, as this is probably one of the most common taxi

[2]cartel.csail.mit.edu

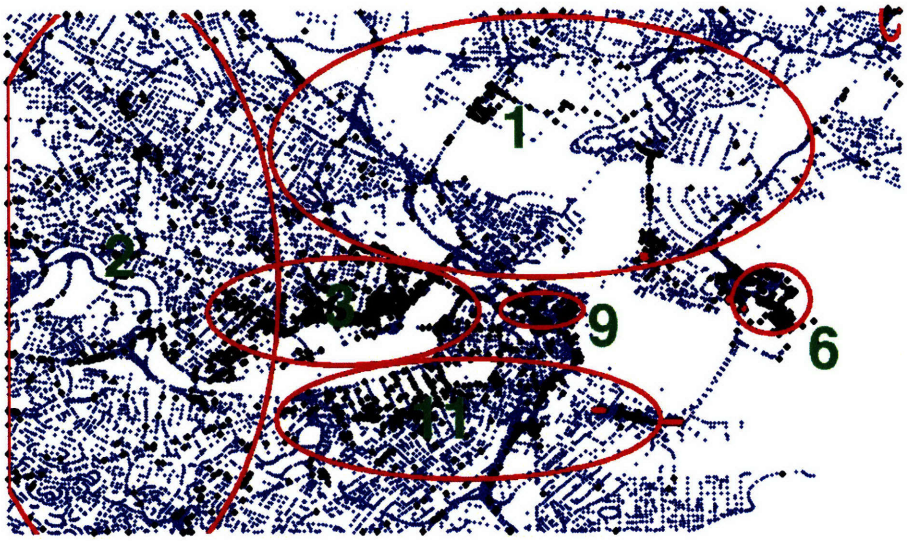

Figure 4-14: The destinations (red) found from clustering the stopping locations of each path of data (black).

destinations. As a side note, it should become clear looking at Figure 4-14, why the probability mass over novel behaviors is so important. Expecting training data to capture all of the possible paths in a domain this large is certainly unreasonable.

Figure 4-15 shows some of the paths extracted from the data. The noisy paths caused the learned DPs for each location to have very large γ_j values. Even if two taxis took the same path, just a single noisy data point snapping to a different waypoint would convince the model they were two different behaviors. Referring back to Equation (3.32), if γ_j is large compared to n_j for all j, the model will just predict all vehicles are heading to the most popular destination. In a naive attempt to overcome this, two paths were said to be equivalent if 90% of the waypoint sequence matched. Figure 4-16 shows two paths that matched (the green waypoints were laid on top of the red) and Figure 4-17 shows two that did not.

While requiring only 90% of the waypoints to match helped the model some, to be useful for end location prediction, more training data is needed (to increase n_j relative to γ_j). This is not surprising if one considers the amount of data used in the simulated city (a far smaller domain) with the amount used for the greater Boston

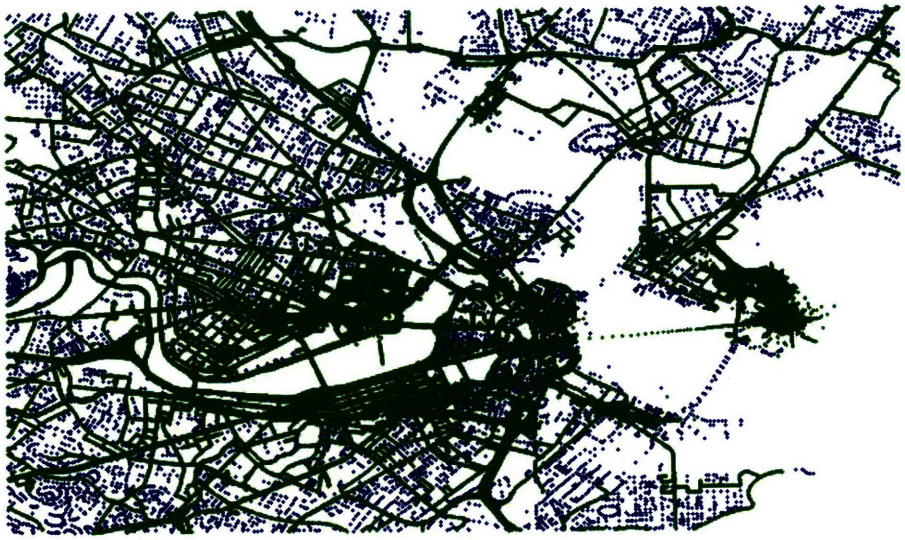

Figure 4-15: Some of the paths of data of taxis driving around the greater Boston area.

Figure 4-16: Two paths (green overlaid on red) with more than 90% of the waypoints in agreement.

Figure 4-17: Two paths (green overlaid on red) with fewer than 90% of the waypoints in agreement.

area. The take-away message from this conclusion is that 22,724 noisy paths of data are too few to make reliable predictions, according to the model, something that other models have no concept of.

Despite not having enough training data to draw any useful conclusions about end destinations, the resulting model can be used to demonstrate something not previously discussed: anomaly detection. Table 4.1 shows the number of paths leading to a destination (as numbered in Figure 4-14), the γ_j value of that destination, and the resulting probability the next path to that destination will be novel. The bottom row is the probability of a novel destination appearing from the next path of data. Since 22,724 paths of data mapped to 12 destinations, it is expected that the probability of a new location appearing is extremely low. The last column corresponds to testing performed on 1,370 withheld paths to determine accuracy of the probability of a novel path. As the table shows, and considering the previous discussion on the relatively small training set for the domain, the model seems to be able to understand how likely new paths are to occur. To be useful for anomaly detection, more training data would need to be provided. For a specific application, if anomalies were expected 1%

Location	# of Paths	γ_j	Prob of Novel	% Novel in Test Paths
1	2021	2207	0.52	73%
2	2138	2925	0.58	79%
3	8167	3910	0.32	48%
4	884	264.8	0.23	35%
5	788	197.7	0.20	16%
6	2057	2427	0.54	70%
7	311	103.5	0.25	44%
8	437	199.9	0.31	42%
9	3290	2464	0.42	66%
10	42	259.7	0.86	75%
11	2535	3606	0.59	73%
12	54	124.38	0.70	67%
Novel	-	1.316	.00005089	0%

Table 4.1: Results comparing predicted probability that a novel path would occur from the training data and the actual frequency of novel paths in the test data.

of the time, the DP-based model gives a structured method for knowing when enough training data had been provided.

Previously, it had been stated that the model needed more training data to predict a vehicle will go anywhere but the most popular destination. While this is true, the model may be forced to make predictions by removing all novel paths from the test data and assuming the remaining paths had been previously observed. Predictions are based on the first third of the 788 remaining paths (from the 1,370 test paths). Given these partial paths, the model predicted the correct destination in 610 of the paths, an accuracy of 77.4%. It is encouraging that the model, once it is given what it thinks is an appropriate amount of training data, will perform well.

Chapter 5

Future Work and Conclusions

5.1 Future Work

5.1.1 A Better Observation Model

While the simulation showed good results and the real data was promising, the DP-based behavior model needs improvements to operate effectively in the presence of noise. The reason for this is that the initial formulation had no mechanism for handling noise. To overcome this, a naive observation model was added such that 90% of the waypoints must match to say two paths are equivalent. This approach was by no means meant as a real solution to handling noise. The work presented here has shown no reason why this model will not work in the presence of noise, if the noise were dealt with properly. The solution will need to redefine a path as a distribution over "similar" paths or a distribution over points close to a path. This could be accomplished by something such as modeling a path as a series of "close" road segments, a series of Gaussian distributions as in [13], or even using DPs.

5.1.2 Path Extraction

The naive method of converting road segments to waypoints and snapping GPS points to the nearest waypoint leaves much to be desired. Not only does this method make predictions about the next road segment somewhat cumbersome, it ignores much of

the road connection information stored in the original representation. In the future, a simple car model may be used to properly filter road sequences from the GPS data.

5.1.3 Recognizing Drawings of Battlefield Strategies

A potential application for hierarchical Dirichlet processes (HDP) is a handheld commander advisor unit in development at the Charles Stark Draper Laboratory. In the field, a commander draws the current battlefield situation and his future plans directly on the device and it computes the probability of possible outcomes. While there are set symbols of inputting a combat unit and its plan, the overall battlefield strategy needs to be recognized. This has proven to be a difficult problem because of the potentially infinite ways a battlefield may be laid out, especially when taking into consideration different types of environments. To begin to understand where the difficulty lies, consider a commander attempting to input a coordinated attack on both a flat plain and on a hilly landscape. The two drawn pictures may look quite different, but the device must recognize them both as coordinated attacks to compute the probabilities of potential outcomes accordingly.

On the bottom level of the hierarchy is a DP for each strategy. Each DP will learn the types of ways a strategy is drawn and learn a distribution over them. The top level of the HDP shares information about movements over strategies. For example, if data were acquired for coordinated attack and retreat in a hilly landscape, it may be desirable to share qualities of these two strategies. If this hierarchy were not in place, the model will not share information about retreat strategies to enhance its understanding of coordinated attack. A better approach is to allow the model to decide during learning how useful transferring knowledge is, rather than making those types of assumptions a priori.

There are three main reasons why this is a potentially good application for HDPs. First, DPs are very flexible in finding what they believe is the correct complexity of the model (e.g. what is the "right" number of ways to represent coordinated attack?). Approaches where the model complexity must be specified in advance are prone to over-fitting or under-fitting of the training data. Second, DPs can handle countably

infinite strategies and countably infinite ways of drawing each strategy. While countably infinite strategies will not be seen in practice, a framework that naturally allows for this may prove extremely useful when dealing with a large number of strategies that need to be inferred. Third, DPs maintain a probability mass associated with a novel strategy. A determination of a strategy as "novel" may be used to conclude poor strategy specification. Typically an ad-hoc method is used for this, where a threshold needs to be exceeded for it to have "decided" which strategy is most likely. Using HDPs for this will allow for a probabilistically justified approach for prompting the commander to redraw the strategy. The probability mass associated with a novel strategy may be used to draw conclusions about the amount of data necessary to learn a sound model.

5.2 Conclusions

This thesis has presented a nonparametric Bayesian approach to behavior modeling, focusing on large, poorly understood domains in which accurate models are difficult to create and typical parametric models break down from over-fitting training data. The basis of the nonparametric approach was the DP. Rather than using a parametric model, a DP defines a process that governs the parameters of the model. Using DPs for behavior modeling for large, poorly understood domains is motivated by three main benefits. DPs do not require any *a priori* knowledge of the number of expected behaviors; they "understand" the training data is a limited view of the domain, and there are no issues with having infinitely many behaviors. With this justification, and inspired by the DARPA Urban Challenge, a behavior model for drivers in a city environment was constructed.

It was discussed that a successful driver model would need to be able to predict both the next action of a driver (e.g., a left turn) and its desired destination (e.g., the airport). From training data the model first clustered path ending locations into destinations. Based on the assignments from the clustering, each destination learned a process that governed the distribution over all paths in the city that drivers took

to the destination.

The DP-based driver model was then tested by predicting paths and destinations for a held-out data set. In simulation, the model performed extremely well compared to a Markov model approach. For the 20x20 simulated city (840 road segments) it was able to predict a vehicle's next action correctly over 80% of the time with only 30% of the path observed. Building on the simulation results, the model was then applied to taxi data from the greater Boston area. The results on the real data were promising but inconclusive due to a relatively small data set and the model not being explicitly designed to handle noisy paths. Based on the encouraging GPS results presented here, if a realistic observation model is incorporated and advanced data processing techniques are utilized, it is reasonable to expect a model that can make sound predictions of a vehicle's next action and destination.

Appendix A

Probability Distributions

All probability distributions that appear in this thesis are defined in Sections A.1-A.6. This is especially important due to the prevalence of multiple common parameterizations for some of the following distributions. For a general introduction to probability theory the reader should refer to [7].

A.1 Gaussian

A Gaussian distribution, also known as a normal distribution, is a continuous distribution defined by two parameters μ and Σ which represent the mean and covariance, respectively. A random variable $x \sim \mathcal{N}(\mu, \Sigma)$ implies

$$\mathrm{P}(x) = f_{\mathcal{N}}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}. \tag{A.1}$$

A less common parameterization, but one used in this thesis for algebraic simplicity is $\mathcal{N}(\mu, s^{-1})$ where the precision $s = \Sigma^{-1}$.

In Section 3.2.2 a posterior distribution is derived from the product of n Gaussians with a Gaussian prior. Fortunately, as discussed in Section 2.3, the Gaussian distribution is its own conjugate prior, which results in a posterior distribution of the

form [1]:

$$\begin{aligned} f_{\mathcal{N}}(x;\tilde{\mu},\tilde{\Sigma}) &\propto f_{\mathcal{N}}(x;\mu_1,\Sigma_1)\cdots f_{\mathcal{N}}(x;\mu_{n+1},\Sigma_{n+1}) \\ &= f_{\mathcal{N}}\left(x;\left[\sum_{i=1}^{n+1}\Sigma_i^{-1}\right]^{-1}\left[\sum_{i=1}^{n+1}\Sigma_i^{-1}\mu_i\right],\left[\sum_{i=1}^{n+1}\Sigma_i^{-1}\right]^{-1}\right) \end{aligned} \tag{A.2}$$

A.2 Gamma

A gamma distribution is a continuous distribution defined by shape parameter, a, and scale parameter, b. A random variable $x \sim \mathcal{G}(a,b)$ implies

$$\mathrm{P}(x) = f_{\mathcal{G}}(x;a,b) = \frac{1}{\Gamma(a)b^a}x^{a-1}e^{-x/b}. \tag{A.3}$$

In (A.3), $\Gamma(a) = \int_0^\infty z^{a-1}e^{-z}dz$ is the gamma function, the extension of the factorial function to complex numbers. There are other common parameterizations of the gamma distribution that the reader may be familiar with but this is the only one used in this thesis.

A.3 Inverse-Gamma

The inverse-gamma distribution is the distribution of a random variable $x \sim \mathcal{G}^{-1}(a,b)$ when $x^{-1} \sim \mathcal{G}(a,1/b)$. $x \sim \mathcal{G}^{-1}(a,b)$ implies

$$\mathrm{P}(x) = f_{\mathcal{G}^{-1}}(a,b) = \frac{b^a}{\Gamma(a)}x^{-(a+1)}e^{-b/x}. \tag{A.4}$$

Equation (A.4) can be derived [7] from the relation

$$f_{\mathcal{G}^{-1}}(y;a,b) = f_{\mathcal{G}}(g^{-1}(y);a,b)\left|\frac{d}{dy}g^{-1}(y)\right|$$

with $y = g(x) = 1/x$ and an inverted second parameter.

A.4 Wishart

The Wishart distribution is a distribution over $p \times p$ positive-definite matrices, W. Its probability density function is written as

$$\mathrm{P}(W) = f_{\mathcal{W}}(W; n, V) = \frac{1}{\Gamma_p(n/2)|V|^{n/2}2^{np/2}}|W|^{\frac{n-p-1}{2}}e^{-\mathrm{Tr}(V^{-1}W)/2} \tag{A.5}$$

where n is the degrees of freedom, V is the scale matrix, and $\Gamma_p(z) = \pi^{p(p-1)/4}\prod_{i=1}^{p}\Gamma(z+(1-i)/2)$ is the multivariate gamma function. It is important to note that the Wishart distribution is the multivariate generalization of the gamma distribution. By setting $p = 1$ and restricting W and V to be 1×1 matrices yields

$$\begin{aligned} f_{\mathcal{W}}(W; n, V) &= \frac{1}{\Gamma(n/2)(2V)^{n/2}}W^{n/2-1}e^{-W/(2V)} \\ &= f_{\mathcal{G}}(x; n/2, 2V). \end{aligned}$$

This may be useful in Section 3.2.2 for readers who wish to extend the methodology to multiple dimensions by replacing gamma distributions with Wishart distributions.

A.5 Multinomial

A multinomial distribution is a probability distribution over a set of draws from a discrete probability distribution. For a discrete distribution

$$\mathrm{P}(x) = \begin{cases} \pi_1 & \text{if } x = 1 \\ \pi_2 & \text{if } x = 2 \\ \vdots & \\ \pi_k & \text{if } x = k \\ 0 & \text{otherwise} \end{cases}$$

where $\sum_i \pi_i = 1$, the probability of outcome counts $n_1, ..., n_k$ is

$$\mathrm{P}(n_1, ..., n_k | \pi_1, ..., \pi_k) = p_m(n_1, ..., n_k; \pi_1, ..., \pi_k) = \frac{(\sum_i n_i)!}{\prod_i (n_i!)} \prod_i \pi_i^{n_i}.$$

A.6 Dirichlet

A Dirichlet distribution is a continuous distribution with parameters $\alpha_1, ..., \alpha_k$ that defines a distribution over $\pi_1, ..., \pi_k$ where $\sum_i \pi_i = 1$. It can be thought of as a distribution on k dimensional discrete distributions, explicitly written

$$\begin{aligned} \mathrm{P}(\pi_1, ..., \pi_k | \alpha_1, ..., \alpha_k) &= f_{Dir}(\pi_1, ..., \pi_k; \alpha_1, ..., \alpha_k) && \text{(A.6)} \\ &= \frac{\prod_i \Gamma(\alpha_i)}{\Gamma(\sum_i \alpha_i)} \prod_i \pi_i^{\alpha_i - 1}. && \text{(A.7)} \end{aligned}$$

Bibliography

[1] P. Ahrendt. The Multivariate Gaussian Probability Distribution. Technical report, IMM, Technical University of Denmark, jan 2005.

[2] D. Aldous. Exchangeability and Related Topics. In *Ecole d'Ete de Probabilities de Saint-Flour XIII 1983*, pages 1–198. Springer, 1985.

[3] C. E. Antoniak. Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems. *The Annals of Statistics*, 2:1152–1174, November 1974.

[4] D. Ashbrook and T. Starner. Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users, 2003.

[5] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The Infinite Hidden Markov Model. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[6] M. Bennewitz, W. Burgard, and S. Thrun. Using EM to Learn Motion Behaviors of Persons with Mobile Robots, 2002.

[7] Dimitri P. Bertsekas and John N. Tsitsiklis. *Introduction to Probability.* Athena Scientific, 2002.

[8] J. Bilmes. A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models, 1997.

[9] D. Blackwell and J. B. Macqueen. Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, 1:353–355, 1973.

[10] D. Blei, T. Gri, M. Jordan, and J. Tenenbaum. Hierarchical Topic Models and the Nested Chinese Restaurant Process, 2003.

[11] Robert G. Brown and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering.* New York: John Wiley & Sons, Inc., 1997.

[12] George Casella and Edward I. George. Explaining the Gibbs Sampler. *The American Statistician*, 46(3):167–174, 1992.

[13] G. Czielniak, M. Bennewitz, and W. Burgard. Where is ...? Learning and Utilizing Motion Patterns of Persons with Mobile Robots. In *Proc. of the International Conference on Artificial Intelligence (IJCAI)*, 2003.

[14] Hussein Dia. An Agent-Based Approach to Modeling Driver Route Choice Behaviour Under the Influence of Real-Time Information. *The American Statistician*, 10:331–349, 2002.

[15] T. S. Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209–230, 1973.

[16] Bernard D. Flury. Acceptance-Rejection Sampling Made Easy. *SIAM Review*, 32(3):474–476, 1990.

[17] Emily Fox, Erik Sudderth, and Alan S. Willsky. Hierarchical Dirichlet Processes for Tracking Maneuvering Targets. In *Proceedings of the International Conference on Information Fusion*, January 2007.

[18] W. R. Gilks and P. Wild. Adaptive Rejection Sampling for Gibbs Sampling. In *Applied Statistics 41*, pages 337–348, 1992.

[19] Greg Hamerly and Charles Elkan. Learning the k in k-means. In *Advances in Neural Information Processing Systems*, volume 17, 2003.

[20] D. Heckerman. A Tutorial on Learning with Bayesian Networks. Technical report, Microsoft Research, Redmond, Washington, 1995. Revised June 1996.

[21] Michael I. Jordan. Dirichlet Processes, Chinese Restaurant Processes and all that.

[22] Robert E. Kass and Larry Wasserman. A Reference Bayesian Test for Nested Hypotheses and its Relationship to the Schwarz Criterion. *Journal of the American Statistical Association*, 90(431):928–934, 1995.

[23] Julia Letchner, John Krumm, and Eric Horvitz. Trip Router with Individualized Preferences (TRIP): Incorporating Personalization into Route Planning. In *AAAI*, 2006.

[24] Lin Liao, Donald J. Patterson, Dieter Fox, and Henry Kautz. Learning and Inferring Transportation Routines. *Artif. Intell.*, 171(5-6):311–331, 2007.

[25] J. B. Macqueen. Some Methods of Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[26] Mario Medvedovic and Siva Sivaganesan. Bayesian Infinite Mixture Model Based Clustering of Gene Expression Profiles. *Bioinformatics*, 18:1194–1206, 2002.

[27] Thomas M. Mitchell. *Machine Learning.* McGraw-Hill Higher Education, 1997.

[28] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[29] D. Patterson, L. Liao, D. Fox, and H. Kautz. Inferring High-Level Behavior From Low-Level Sensors, 2003.

[30] Dan Pelleg and Andrew Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, 2000. Morgan Kaufmann.

[31] Jim Pitman. Poisson–Dirichlet and GEM Invariant Distributions for Split-and-Merge Transformations of an Interval Partition. *Comb. Probab. Comput.*, 11(5):501–514, 2002.

[32] L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[33] C. E. Rasmussen. The Infinite Gaussian Mixture Model. In S.A. et al. Solla, editor, *Advances in information processing systems 12*, pages 554–560. MIT Press, 2000.

[34] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, December 2005.

[35] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[36] Sean Borman. The Expectation Maximization Algorithm - A Short Tutorial. July 2004.

[37] J. Sethuraman. A Constructive Definition of Dirichlet Priors. *Statistica Sinica*, 4:639–650, 1994.

[38] Kyung-Ah Sohn and Eric P. Xing. Hidden Markov Dirichlet Process: Modeling Genetic Recombination in Open Ancestral Space. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1305–1312. MIT Press, Cambridge, MA, 2007.

[39] B. Stewart, J. Ko, D. Fox, and K. Konolige. The Revisiting Problem in Mobile Robot Map Building: A Hierarchical Bayesian Approach, 2003.

[40] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

[41] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical Dirichlet Processes, 2003.

[42] Y. W. Teh. Dirichlet Processes. Submitted to Encyclopedia of Machine Learning, 2007.

[43] E. Xing, M. Jordan, R. Karp, and S. Russell. A Hierarchical Bayesian Markovian Model for Motifs in Biopolymer Sequences, 2003.

[44] Brian D. Ziebart, Andrew Maas, James Bagnell, and Anind K. Dey. Maximum Entropy Inverse Reinforcement Learning. In *Proceeding of AAAI 2008*, July 2008.