A Bayesian Nonparametric Approach to Modeling Mobility Patterns

Joshua Joseph and **Finale Doshi-Velez** and **Nicholas Roy**
Massachusetts Institute of Technology
Cambridge, MA
{jmjoseph, finale, nickroy}@mit.edu

Abstract

Constructing models of mobile agents can be difficult without domain-specific knowledge. Parametric models flexible enough to capture all mobility patterns that an expert believes are possible are often large, requiring a great deal of training data. In contrast, nonparametric models are extremely flexible and can generalize well with relatively little training data.

We propose modeling the mobility patterns of moving agents as a mixture of Gaussian processes (GP) with a Dirichlet process (DP) prior over mixture weights. The GP provides a flexible representation for each individual mobility pattern, while the DP assigns observed trajectories to particular mobility patterns. Both the GPs and the DP adjust the model's complexity based on available data, implicitly avoiding issues of over-fitting or under-fitting. We apply our model to a helicopter-based tracking task, where the mobility patterns of the tracked agents—cars—are learned from real data collected from taxis in the greater Boston area.

1. Introduction

The success of autonomous agents in realworld domains hinges on the quality of the agent's world model. In complex domains, these models often contain aspects that are difficult to design; specifically, environments may contain other agents that may be difficult to model from expert-knowledge alone. For example, suppose a helicopter (the agent) must track a specific car in a large region such as a city. The model of traffic patterns may be hard to specify and even specifying the model's complexity may be a challenging task.

We apply a Bayesian nonparametric approach to model the mobility patterns of the tracked agents. Nonparametric approaches are well-suited for poorly-understood environments because they let the data determine the sophistication of the model. They also provide a principled, robust framework for working with trajectories of varying lengths. The Bayesian aspect helps the model generalize to unseen data and make inferences from noisy data. We focus on scenarios where the tracked agents' motion patterns are initially unknown. Our goal is to learn these agents' mobility patterns and apply this knowledge to a planning task.

Specifically, we model the agent's mobility patterns with a Gaussian process mixture model. Suppose there are M mobility patterns $b_1, b_2, \ldots, b_M$, with prior probabilities $p(b_1), p(b_2), \ldots, p(b_M)$. Then the probability of the i^{th} observed trajectory t^i under the mixture model is

$$p(t^i) = \sum_j^M p(b_j) p(t^i|\theta_j) \quad (1)$$

where θ_j contains the parameters for mobility pattern b_j.

Finally, a data-driven approach to mobility modeling sidesteps issues around the agent's motivations, which may appear irrational to an outside observer. For example, drivers rarely take the "optimal" route to a location. Instead, we focus on the features our own agent needs: good predictions of where other agents will be (which may depend both on those agents' current positions and their past histories). The core contribution of this work is developing a nonparametric mobility model and using it for planning with real-world data.

2. Mobility Model

We represent an agent's trajectory t^i as a set of xy-locations $\{(x_1^i, y_1^i), \ldots, (x_{T_i}^i, y_{T_i}^i)\}$, where T_i is the length of trajectory t^i. A *mobility pattern* is a mapping from trajectories to a distribution over trajectory derivatives $(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})$ indicating the agent's future motion.[1] Given a position (x_t, y_t) and a trajectory derivative $(\frac{\Delta x_t}{\Delta t}, \frac{\Delta y_t}{\Delta t})$, the predicted next position (x_{t+1}, y_{t+1}) is $(x_t + \frac{\Delta x_t}{\Delta t}\Delta t, y_t + \frac{\Delta y_t}{\Delta t}\Delta t)$. Thus, modeling trajectory derivatives is equivalent to modeling trajectories.

The *mobility model* is a mixture model of mobility patterns. We model each mixture component with a pair of Gaussian processes mapping $(x, y) \rightarrow \frac{\Delta x}{\Delta t}$ and $(x, y) \rightarrow \frac{\Delta y}{\Delta t}$, and a Dirichlet process prior (DP) over mixture weights. Our model is similar to Rasmussen and Ghahramani (2002) and Meeds and Osindero (2006); however, unlike these previous works, our goal is to cluster trajectories of varying lengths, not just partition single points.

More formally, the posterior probability of mobility pattern b_j for a trajectory t^i under our DPGP model is proportional to $\pi_j \cdot l(b_j; t_i)$. The prior probability of mobility pattern b_j is given by its DP mixture weight π_j. The likelihood

[1] The choice of Δt determines the scales we can expect to predict the car's next position well, making the trajectory derivative more useful than instantaneous velocity.

$l(b_j; t_i)$ of mobility pattern b_j under trajectory t^i is

$$l(b_j; t_i) = \prod_t^{T_i} p\left(\frac{\Delta x}{\Delta t}_t \middle| x^i_{1:t}, y^i_{1:t}, \{t_k : z_k = j\}, \theta^{GP}_{x,j} \right) \cdot \prod_t^{T_i} p\left(\frac{\Delta y}{\Delta t}_t \middle| x^i_{1:t}, y^i_{1:t}, \{t_k : z_k = j\}, \theta^{GP}_{y,j} \right) \quad (2)$$

where z_k indicates the mobility pattern to which trajectory t^k is assigned, and $\theta^{GP}_{x,j}$ and $\theta^{GP}_{y,j}$ are the hyperparameters of the Gaussian process for mobility pattern b_j. Equation 2 may be applied to trajectories with differing numbers of nodes or even trajectories that are only partially complete, which is particularly important when we wish to compare model likelihoods across a varied dataset.

2.1 Gaussian Process Mobility Patterns

Points from a trajectory represent a continuous path through space. The Gaussian process (Rasmussen and Williams 2005) places a distribution over functions, serving as non-parametric form of interpolation. While GP inference is somewhat involved, GP models are extremely robust to unaligned, noisy measurements and are thus well-suited for modeling the continuous paths underlying our non-uniformly sampled time-series motion data.

The GP for each trajectory derivative is specified by a mean function $E[\frac{\Delta x}{\Delta t}] = \mu_x(x, y)$ and $E[\frac{\Delta y}{\Delta t}] = \mu_y(x, y)$. We set both of these mean functions to initially be zero everywhere (for all x and y), encoding the prior bias that, without any additional knowledge, we expect the agent to stay in the same place. Zero-mean GP priors also simplify computations. Finally, because we are mapping (x, y) *locations* to velocities, we remove time-dependent aspects from the model: traffic conditions before the agent reached a certain location has no effect on the velocity predictions for that point. Inherent variability of the velocity at a particular location (due to varying traffic conditions at that point) are captured in the covariance function of the Gaussian process.

We denote the covariance function for the trajectory derivative in the x-direction $K_x(x, y, x', y')$, which describes the correlations between trajectory derivatives at two different points (x, y) and (x', y'). Given a set of points $(x_1, .., x_k, y_1, .., y_k)$ the corresponding trajectory derivatives $(\frac{\Delta x_1}{\Delta t}, .., \frac{\Delta x_k}{\Delta t})$ will have a multivariate Gaussian distribution with mean $\{\mu_x(x_1, y_1), .., \mu_x(x_k, y_k)\}$ and covariance Σ, where the $\Sigma_{ij} = K(x_i, y_i, x_j, y_j)$. In this work, we use the standard squared exponential covariance function

$$K_x(x, y, x', y') = \sigma_x^2 \exp\left(-\frac{(x - x')^2}{2{w_x}^2} - \frac{(y - y')^2}{2{w_y}^2} \right) + \sigma_n^2 \delta(x, y, x', y'). \quad (3)$$

The exponential term above encodes that similar trajectories should make similar predictions. The length-scale parameters w_x and w_y normalize for the scale of the data. The σ_n-term represents within-point variation (e.g., due to noisy measurements); the ratio of σ_n and σ_x weights the relative effects of noise and influences from nearby points. We use $\theta^{GP}_{x,j}$ to refer to the set of hyperparameters σ_x, σ_n, w_x, and w_y associated with mobility pattern b_j (each mobility pattern has a separate set of hyperparameters).[2]

For a GP over trajectory derivatives trained with tuples $(x_k, y_k, \frac{\Delta x_k}{\Delta t})$, the predictive distribution over the trajectory derivative $\frac{\Delta x}{\Delta t}^*$ for a new point (x^*, y^*) is given by

$$\mu_{\frac{\Delta x}{\Delta t}^*} = K(x^*,y^*,X,Y)K(X,Y,X,Y)^{-1}\frac{\Delta X}{\Delta t} \quad (4)$$
$$\sigma^2_{\frac{\Delta x}{\Delta t}^*} = K(x^*,y^*,X,Y)K(X,Y,X,Y)^{-1}K(X,Y,x^*,y^*)$$

where the expression $K_x(X, Y, X, Y)$ is shorthand for the covariance matrix Σ with terms $\Sigma_{ij} = K_x(x_i, y_i, x_j, y_j)$. The equations for $\frac{\Delta y}{\Delta t}^*$ are equivalent to those above, using the covariance K_y.

2.2 Dirichlet Process Mixture Weights

The Dirichlet process places a distribution over discrete distributions in which the number of mobility patterns is potentially unbounded, but there are a few patterns we expect to see most of the time.[3] If z_i indicates the mobility pattern to which trajectory t^i is assigned, the prior probability that t^i belongs to an existing mobility pattern b_j is

$$p(z_i = j | z_{-i}, \alpha) = \frac{n_j}{N - 1 + \alpha}, \quad (5)$$

where z_{-i} refers to the mobility pattern assignments for the remaining trajectories, α is the concentration parameter of the Dirichlet process, n_j is the number of trajectories assigned to mobility pattern b_j, and N is the total number of observed trajectories. The probability that trajectory t^i exhibits a new mobility pattern is

$$p(z_i = M + 1 | z_{-i}, \alpha) = \frac{\alpha}{N - 1 + \alpha}. \quad (6)$$

where M is the number of observed mobility patterns.

Equation 6 implies that the number of mobility patterns can grow as more data is obtained. This property is key when modeling real agents: the more trajectories we observe, the more patterns we expect to encounter. Figure 1 shows how the number of mobility patterns grows (under our model) as new trajectories are observed for an actual dataset of greater Boston taxi routes. We show in section 5 that we can efficiently plan even when the number of actively observed mobility patterns are changing; moreover, this flexibility yields significantly improved results in the performance of the planner.

3. Model Inference

Exact inference over the space of DPs and GPs is intractable; instead we draw samples from the posterior mobility model. These samples are then used by our agent for planning.

3.1 Training the Model

Our model contains two sets of parameters—the DP mixture weights π_k, the mobility pattern assignments z_i, and the the

[2] We described the kernel for two dimensions, but it can be easily generalized to more.

[3] See Teh (2007) for an overview of Dirichlet processes.

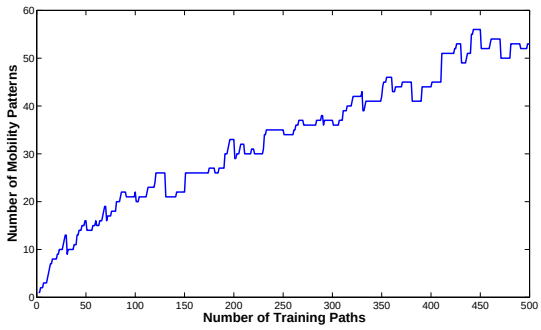


Figure 1: As expected, the number of mobility patterns in the taxi dataset increases as more trajectories are added.

DP hyperparameter α—the GP hyperparameters $\theta^{GP}_{x,j}, \theta^{GP}_{y,j}$ and the trajectories assigned to each cluster. Following the work of Rasmussen and Ghahramani (2002) and Rasmussen (2000), learning the model involves Gibbs sampling the parameters (see Algorithm 1).

We first resample each z_i in turn, using the exchangeability properties of the DP and GP to model trajectory t^i as the most recent trajectory. The probability that the trajectory t^i will be assigned to an instantiated mobility pattern is

$$p(z_i = j|T, \alpha, \theta^{GP}_{x,j}, \theta^{GP}_{y,j}) \propto l(b_j; t_i) \left(\frac{n_j}{N - 1 + \alpha} \right) \quad (7)$$

where $l(b_j; t_i)$ is the likelihood of mobility pattern b_j from equation 2 and n_j is the number of trajectories currently assigned to mobility pattern b_j. The probability that the trajectory t^i belongs to a new mobility pattern is given by

$$p(z_i = M+1|t_i, \alpha) \propto \int l(b_j; t_i) d\theta^{GP}_{x,j} d\theta^{GP}_{y,j} \left(\frac{\alpha}{N - 1 + \alpha} \right), \quad (8)$$

and we use Monte Carlo integration (Bishop 2006) to approximate the integral. The likelihood from equation 7 also must be approximated for popular mobility patterns, as the computations are cubic in the cluster size n_j. Following Rasmussen and Williams (2005), we approximate the likelihood for these larger clusters using the N_{max} trajectories that are closest to the trajectory t^i.[4]

The DP concentration hyperparameter α is resampled using standard techniques (Rasmussen 2000). The GP hyperparameters—the length-scale and the variance—are harder to resample; however, since their posteriors were extremely peaked, we instead always set them to their maximum likelihood values (using gradient ascent). In applications where the posteriors are less peaked, hybrid Monte Carlo techniques may be used (Duane et al. 1987).

3.2 Classification and Prediction with New Trajectories

The mobility model from algorithm 1 can now be used to make predictions for a trajectory that is still in progress.

[4]We tested the validity of this approximation by comparing approximations in which only the nearest points were used to the true likelihood and found no practical difference when discarding 75% of trajectories for large clusters.

Algorithm 1 Mobility Model Inference

```
for sweep = 1 to # of sweeps do
    for each mobility pattern b_j do
        Draw the GP hyperparameters θ^GP_{x,j}, θ^GP_{x,j}
    end for
    Draw the DP hyperparameter α
    for each trajectory t_i do
        Draw z_i using equations 7 and 8
    end for
end for
```

Given a partial trajectory t_i, we first apply equations 7 and 8 to compute the relative probability of it belonging to each mobility pattern b_j. Equation 2 is used to compute the likelihoods. Just as in the full trajectory case, we are interested in comparing across different mobility patterns, so the partial trajectory may contain any number of points.

For each pattern b_j, we first compute the expected trajectory derivatives $(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})_j$ conditioned on GP parameters $(\theta^{GP}_{x,j}, \theta^{GP}_{y,j})$ (equation 4). The expected trajectory derivative is a weighted average over all the conditional derivatives $\sum_j \pi_j (\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})_j$.[5]

4. Helicopter-based Tracking Task

Recall that our primary objective was to build a mobility model for use in a planning context. We now illustrate how the mobility model of section 2 can be applied in a helicopter-based tracking task. Our (simulated) agent is an unmanned helicopter tasked with tracking a particular car. The helicopter is rewarded for every time-step in which it can "observe" the car. While it knows the effects of its own actions—hovering over various parts of the city—and the car's current location, it must predict the car's future position to accomplish its task successfully. The helicopter receives periodic position reports of the car position from a sparse network of ground sensors, allowing us to model the problem as fully observable with uncertain action outcomes. While a real "chase" scenario would have many more complexities, this simplified tracking task allows us to show empirically that our model, initially trained on likelihood-based criteria, also performs well on a planning problem.[6]

We model the scenario as a Markov decision process (MDP), a common tool for autonomous decision making. An MDP is defined by a set of states S, a set of actions A, a transition function T, and a reward function R. Here the state s is the joint position of the helicopter and the car (x_h, y_h, x_c, y_c). Given an action a and helicopter position (x_h, y_h), the helicopter's next position (x'_h, y'_h) is determin-

[5]In practice, we found that the mobility pattern likelihoods were highly peaked. In this situation, it was sufficient to only consider the maximum likelihood mobility pattern when predicting the future locations in partial trajectories.

[6]Likelihood-based methods try to explain the data well, while the goal of the planning problem is to maximize rewards. A model that best-explains the data is not guaranteed to be the best model for planning.

istic and known. In contrast, the car's transitions are stochastic and initially unknown: we can only place a distribution over the car's next position (x'_c, y'_c). Given an MDP, we can find the optimal policy, using standard forward search techniques (Puterman 1994).

To model the helicopter and its rewards, we place a 20×20 grid over a city (an area of approximately 10 square miles) and represent the helicopter's state by its grid cell. This representation is reasonable since the helicopter can monitor large parts of the city from one location. At each time step, the helicopter can stay in place (cost 0), move one cell (cost -1), or move two cells (cost -2). It receives a bonus of +10 if it is in the same cell as the car it is meant to be tracking.

The mobility model determines the movement of the car. Under the DPGP model, the car's position at time $t+1$ given its position at time t is estimated using the method described in section 3.2. Given a predicted next-location for the car, the helicopter can try to move to the appropriate grid cell that covers the car's location.

We compare our DPGP mobility model to a Markov model that uses a discretized version of the city, where we overlay a second grid over the city and use our trajectory data to compute car transition probabilities between grid cells. Given a car's current position and velocity, we can first map it to a grid cell and then predict its next grid cell with the model. Using a tighter grid for the car's motion than the helicopter's grid allows us to model the car's motion at a finer level of granularity. We call this model the "Markov model" because predictions about the car's next position depend only on the car's current position and velocity, not its past history. In contrast, the DPGP model considers the car's entire trajectory so far when deciding to which mobility pattern in belongs.

5. Experiments

We tested various driver models with the helicopter-based tracking task described in section 4. We simulated the collection of observations of the car by introducing new trajectories one by one, one movement at a time. Given a partial sequence of from a trajectory, the helicopter predicted and moved to future locations of the moving car, subject to the constraints of its own dynamics. Once each tracking task was completed, that car's trajectory was incorporated as training data into the mobility model. This process was repeated for 500 trajectories taken from a dataset of time-stamped GPS coordinates of greater-Boston area taxis.[7] Thus, we could monitor the data-efficiency of our models as well as their overall performance. A few trajectories (red points) are plotted in figure 2 on a map of Boston[8]. The green points highlight a single trajectory. The trajectories lie on a road network, but we make no assumptions about the positions of the cars. Figure 3 shows an example of two trajectories that share a segment of road but branch in different directions based on their starting position.

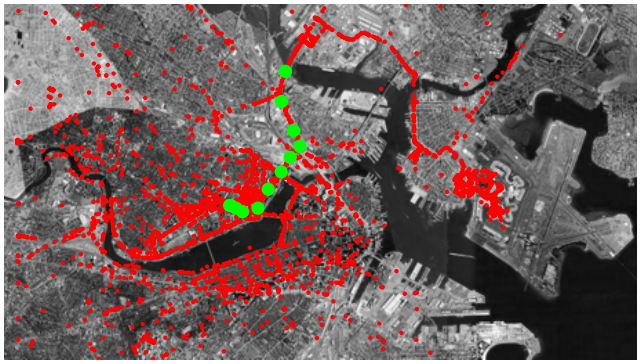


Figure 2: A small set of the raw GPS data points (red) and a single trajectory (green) used to learn our model.

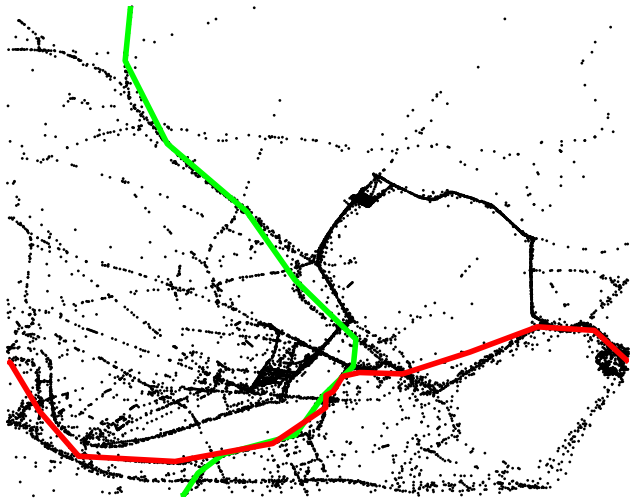


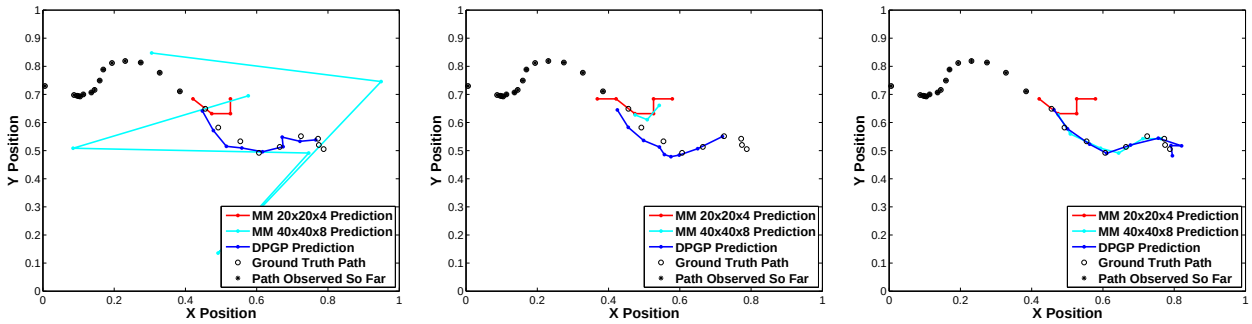
Figure 3: A example of two trajectories that share a road segment. The Markov model cannot distinguish the two trajectories once they cross, but the DP model classifies them as two different paths.

As new trajectories arrived, the Markov driver model was updated by changing the counts on how often transitions were observed among adjacent grid cells (we also added a uniform Dirichlet prior). The x and y-positions were discretized on a 20×20, 40×40, or a 60×60 grid (the helicopter's discretization never changed). Velocity was either discretized into four states (north, south, east, west) or eight (north, northeast, east, southeast, etc.). The models with finer discretizations were more expressive but contained more unknown parameters.

After each trajectory was completed, our DPGP driver model was updated using algorithm 1. Each update was initialized with the most recently sampled model. Since a full update required significant computation, new trajectories were initially binned with their most likely mobility pattern (which could have been a new pattern). Every 10 new trajectories, a complete set of 5 Gibbs sweeps were run to update the model parameters and trajectory assignments (we found that samples generally stopped changing after the first 2 sweeps). The noise parameter σ_n in equation 3 was fit

[7] CarTel project, cartel.csail.mit.edu. The data was downsampled to a rate of 1 reading per minute and pre-processed into trajectories based on if the car had stayed in the same place for a long enough time to indicate the end of a trajectory.

[8] maps.google.com



(a) 100 paths (b) 300 paths (c) 500 paths

Figure 4: Predictions given a partial path for the DPGP and two Markov models for various amounts of training data.

from the current trajectory set. While the DPGP model required more computation than the Markov model (about 10 times slower), it could still incorporate a new set of samples in minutes, an update rate fast enough for a real scenario where the model may be updated several times a day. The planning time was nearly instantaneous for both the DPGP and the Markov driver models.

We first carried out a series of experiments to evaluate the quality of our models. Example predictions of the DPGP and Markov models are seen in figure 4. The solid circles show a partial trajectory; the open circles show the true continuation of the trajectory. The paths show the continuations predicted by the DPGP model and two Markov models. With only 100 training trajectories, none of the models predict the full path, but the DPGP is close while the other models are completely lost. As more training data is added, the DPGP and the finer-grained Markov model match the true trajectory, while the simpler Markov model is not flexible enough to fit the data.

Figure 5 compares the predictive test likelihood of the DPGP model to the Markov models using 50 withheld paths. The likelihoods for the DPGP model, which has a continuous density, were evaluated by integrating over an equivalent grid as in the Markov models. The DPGP model's consistently-higher likelihoods indicate it models the trajectories better than the Markov models.

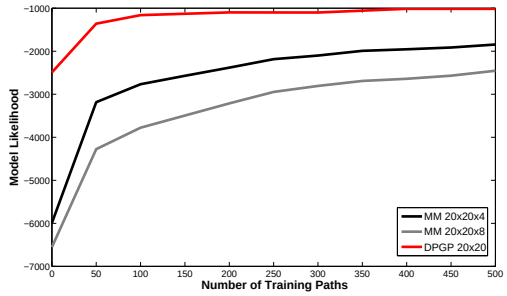


Figure 5: Model likelihood for 50 withheld test paths for the DPGP and Markov models.

As the goal of our model is to predict the motion of mobile agents within a planner, we compared the performance of planners using the DPGP and Markov dynamics models, as well as a naive pursuit approach that simply assumed the vehicle's position at time $t+1$ would be the same as its location at time t. We also compared the DPGP model to a simple k-nearest neighbor technique that, given an (x, y) point, simply searched the training set of trajectories for nearby (x, y) points and interpolated the trajectory derivatives $\frac{\Delta x}{\Delta t}$ and $\frac{\Delta y}{\Delta t}$ from the trajectory derivatives of nearby training points.[9] Finally, we compared the DPGP model against a GP model that was fit to only the current trajectory and ignored all past training data. This single GP model ensured that the previous trajectories were important for making predictions about the current trajectory, that is, the current trajectory could not be well-predicted based on its own velocities.

Figure 6 shows the cumulative difference of total reward between all the approaches and naive pursuit method. The k-nearest neighbor and simple GP rarely out-perform pursuit. The Markov models initially do worse than pursuit because they have many parameters (making them vulnerable to over-fitting) and often make incorrect predictions when the agent observes a trajectory in a sparse region of their state space. In contrast, the DPGP starts out similar to pursuit, since the zero-mean prior on trajectory derivatives naturally encodes the bias that, in the absence of other data, the car will likely stay still. Moreover, the GP quickly generalizes from a few trajectories and thus attains the highest sloped cumulative rewards of the other methods. The Markov models eventually also exhibit similar performance, as measured by the slope of the cumulative reward, but they never make up for the rewards that they lost during training.

Finally, we compared the DPGP to a set of finite mixture models that also used Gaussian processes to model mobility patterns (that is, the finite mixture model first described in equation 1). Each model was trained on a batch of 200 trajectories using 5 different initializations. The helicopter's tracking performance was then tested on a different set of 15 trajectories. None of the models were updated during the testing phase. The results in figure 7 show that while the finite GP-based models perform well overall, our DPGP model has nearly the best performance *without having to*

[9] For reasonably dense data, Gaussian process and nearest neighbor approximations are very close; thus, the k-nearest neighbor technique also served as a close approximation of a solution trained on a single GP for the entire dataset.

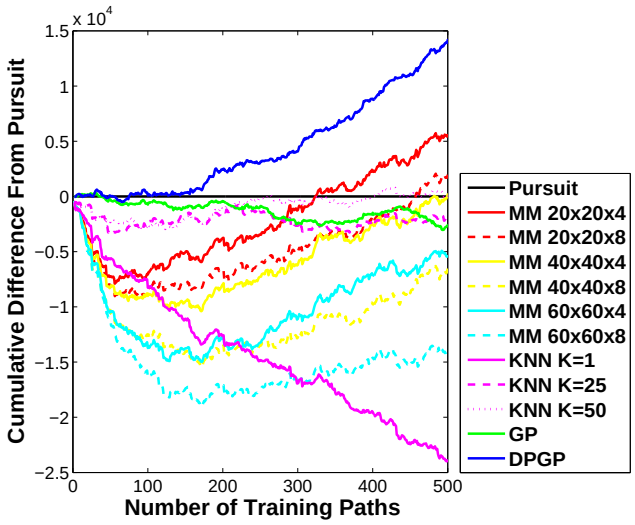


Figure 6: Cumulative difference in reward from the pursuit approach for the DPGP model, various Markov models (MM), k-nearest neighbors (KNN), and a GP fit to the current trajectory (GP) (higher values are better).

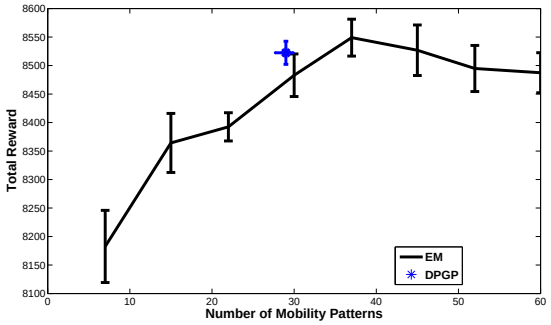


Figure 7: Performance vs. model size for a variety of finite models (green) and the DPGP (blue). The error bars represent the standard error of the mean from five runs. The DPGP point is drawn at the number of mobility patterns discovered in the training data.

perform a search over the finite model space. This last point is important, not only because a search over finite models would require more computation but also because the search requires us to choose a regularization criterion to avoid overfitting. Standard criteria, such as the Bayesian information criterion (Raftery 1986) cannot be applied in this context because the GP contains an unbounded number of parameters; thus we must choose from various cross-validation or bootstrap procedures. The DPGP provides a principled, simple-to-use regularization criterion within its model.

Searching in the space of finite models is especially computationally expensive when the data arrives online and the number of clusters are expected to grow with time. (The DP can update the number of clusters incrementally.) To gain insight into the extra computation cost of this search process we implemented EM where every 10 paths we search over models sizes that are within five clusters of the current model. Figure 8 is the plot of run time as the number of training paths increase for our DPGP model and this adaptive EM technique. The running time grows exponentially faster for EM with model search compared to the DPGP.

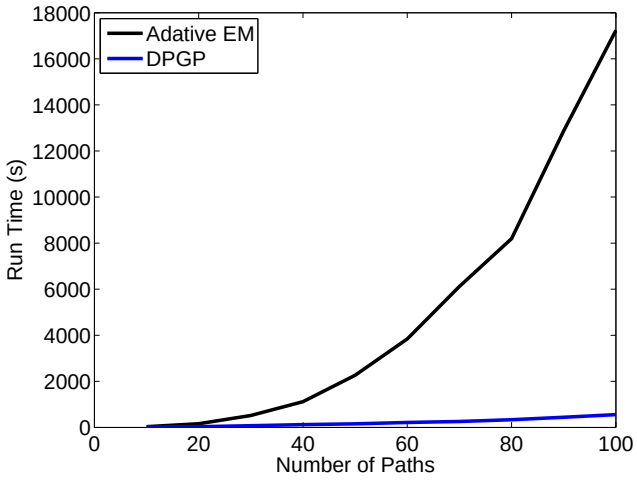


Figure 8: Run time vs. number of paths for adaptive EM and our DPGP model.

6. Related Work

Much of the past work in modeling mobile agents has focused on two problems: expert systems (which require specialized data) and modeling a single agent (requiring data generated by the single agent). Letchner, Krumm, and Horvitz (2006) built a model that predicted trajectories based on the optimal path between two locations (given factors such as the time of day) and the amount of "wasted time" a driver was willing to accept. Dia (2002) used a survey to classify drivers into different profiles to enable better prediction. Both of these works note that it is difficult to specify a model for human mobility patterns based on logical reasoning. For example, Letchner, Krumm, and Horvitz (2006) note only 34.5% of drivers choose the fastest route between two locations. Whether these statistics are a result of driver ignorance or another factor (*e.g.*, avoiding a stressful route) is highly debatable and difficult to incorporate into expert models of human mobility patterns. Without access to similar data for the greater Boston area or having similar time-stamped GPS data for their models, we were unable to compare them to our approach; however, it would be interesting to see if incorporating expert features related to human psychology into the priors of our model could improve predictions.

Another body of literature has trained Markov models (generally using data from only one person) in which each road segment is a state and transition probabilities encode the probabilities of moving from one segment to another. For example, Patterson et al. (2003) treated the true driver state as hidden by GPS sensor noise and a hidden driver mode. Ashbrook and Starner (2003) model the end position and transition probabilities explicitly, but doing so prevents the method from updating the probabilities based on a partially observed trajectory. Using a hierarchy of Markov models, Liao et al. (2007) were able to make both local and destination predictions but still had difficulty in regions of

sparse training data. In contrast, Ko and Fox (2009) demonstrated that Gaussian processes improved the model of a vehicle's dynamics, but for the control of a single vehicle, rather than a predictive model of many mobile agents. Taking a machine learning approach, Ziebart et al. (2008) used inverse reinforcement learning with good results but would be inapplicable to our data containing paths from many different drivers heading to many different destinations (meaning different reward functions).

The Markov model approaches lose critical information by only considering the most recent observation or location instead of the entire trajectory. In the taxi scenario, where it is common to have a shared road between two different pairs of popular destinations, knowing the past history of the car can significantly improve the prediction of how the car might exit the shared road. This scenario is demonstrated in figure 3 where the red path (heading east) and green path (heading north) share the same road segment for a significant amount of time but only knowing the cars were on that road segments provides far less predictive power than knowing its past history. These paths are overlaid on the remainder of the data set to provide some intuition of the road network.

Finally, a somewhat different but related approach was taken in Fox, Sudderth, and Willsky (2008). Using our terminology, the authors define a mobility pattern using a linear-Gaussian state space model. Like our approach, the number of mobility patterns was modeled using a Dirichlet process prior. Unlike our approach, agents could switch between mobility patterns using an underlying hidden Markov model. In our specific dataset and application, the agents usually know their start and end destinations from the very beginning; not allowing mobility pattern changes helped predict a car's path on roadways that were common to many mobility patterns. However, our framework could certainly be extended to allow agents to change mobility patterns. Future work could also incorporate additional information—such as inputs of the road network—to further constrain the trajectories.

7. Conclusions

Accurate agent modeling in large domains often breaks down from over-fitting or under-fitting the training data. We used a Bayesian nonparametric approach to mobility-pattern modeling to circumvent these issues. This approach allowed us to build flexible models that could generalize sensibly when given only a little data and add structure as more data was added. We demonstrated our mobility model on a helicopter-based tracking task trained and tested on a real dataset of car trajectories. Our encouraging results suggest that this approach will be useful in a variety of agent-modeling situations. Moreover, since the underlying structure of the our model is based on a Gaussian process framework, our approach could easily be applied to beyond road networks to generic metric spaces. The mobility models could also be used for tracking multiple agents.

References

Ashbrook, D., and Starner, T. 2003. Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users. *Personal Ubiquitous Computing* 7(5):275–286.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

Dia, H. 2002. An Agent-Based Approach to Modeling Driver Route Choice Behaviour Under the Influence of Real-Time Information. *The American Statistician* 10.

Duane, S.; Kennedy, A. D.; Pendleton, B. J.; and Roweth, D. 1987. Hybrid Monte Carlo. *Physics Letters B* 195(2).

Fox, E.; Sudderth, E.; and Willsky, A. S. 2007. Hierarchical Dirichlet Processes for Tracking Maneuvering Targets. In *Proc. Inter. Conf. Information Fusion*.

Ko, J., and Fox, D. 2009. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots* 27(1):75–90.

Letchner, J.; Krumm, J.; and Horvitz, E. 2006. Trip Router with Individualized Preferences (TRIP): Incorporating Personalization into Route Planning. In *AAAI*.

Liao, L.; Patterson, D. J.; Fox, D.; and Kautz, H. 2007. Learning and Inferring Transportation Routines. *Artif. Intell.* 171(5-6):311–331.

Meeds, E., and Osindero, S. 2006. An alternative infinite mixture of Gaussian process experts. In *NIPS 18*.

Patterson, D.; Liao, L.; Fox, D.; and Kautz, H. 2003. Inferring High-Level Behavior From Low-Level Sensors. In *Proc. UBICOMP*.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Raftery, A. 1986. Choosing models for cross-classifications. *Americal Sociological Review* 51.

Rasmussen, C. E., and Ghahramani, Z. 2002. Infinite mixtures of Gaussian process experts. In *NIPS 14*.

Rasmussen, C. E., and Williams, C. K. I. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*.

Rasmussen, C. E. 2000. The Infinite Gaussian Mixture Model. In *NIPS 12*.

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

Ziebart, B. D.; Maas, A.; Bagnell, J.; and Dey, A. K. 2008. Maximum Entropy Inverse Reinforcement Learning. In *AAAI*.