# GAUSSIAN PROCESSES FOR TRAVEL TRAJECTORY PREDICTION

LIAM CHU

*Math Department, University of Washington, Seattle, WA*
`liamchu@uw.edu`

Abstract. In this final project, we explore the classification performance of the ensemble decision tree, known as the random forest classifier, in comparison to a Dirichlet-based Gaussian process classifier. The context of interest is human travel trajectory prediction, and to our knowledge, Gaussian processes have not been widely explored for such a task, while random forest classifiers are well established within the field. In addition to evaluating classification accuracy, we develop a custom frequency map to determine the "closeness" of incorrect predictions in order to understand the flaws in both models' decisions. We conclude that the Gaussian process has both better accuracy, as well as closer misclassifications compared to random forest, leading us to believe that Gaussian processes have a strong potential for travel trajectory prediction tasks.

## 1. Introduction and Overview

Travel trajectory and next location prediction of urban road network users is a hot topic in modern research. By accurately predicting users' destinations and analyzing collective mobility patterns, city transportation planners and traffic managers can alleviate traffic flows by devising active management and control strategies in advance [8]. Additionally, the analysis of a population's collective mobility patterns can be applied towards urban planning, epidemic control, and other large-scale problems arising within the built environment. Mobility analysis is also helpful for understanding personalized travel behavior, thus allowing travel recommender systems to create personalized schedules and routes based on the preferences of each user.

Common methods for predicting a user's next location involve learning sequences from massive historical trajectory datasets and their conditional probabilities [3]. In past research, both traditional machine learning (decision trees, k-nearest neighbors, support vector machines) and neural networks have been popularly employed for location prediction, and have yielded strong results as classifiers. In their literature review, Sun et. al. [8] noted that Markov models, decision trees, and neural networks are historically popular and successful methods for location prediction. Xiao et. al. [10] use an ensemble decision tree classifier with bootstrap aggregation–often referred to as the bagging method or random forest–and note that single decision trees, k-nearest neighbors, and support vector machines tend to perform worse than ensemble methods in terms of both accuracy and robustness (to noise) in classification problems.

In our project, we employ a novel technique within the field of travel trajectory research to predict a user's next location. To our knowledge, Gaussian processes (GPs) have not yet been widely explored for such a task. In particular, we employ the use of a Dirichlet-based GP which has found success as a large-scale calibrated classifier [4]. In this report, we use mobility data provided by the UW THINK Lab and compare the performance of the Dirichlet-based GP against a random forest classifier. We analyze our results using a raw difference table and a custom made frequency map of sequences to location.

*Date*: December 11, 2024.

It is important to note that a portion of this research was performed before hand during the summer, which is a superficial implementation of the Dirichlet-based GP, a simple comparison to the random forest classifier, and a literature review of travel behavior. We emphasize that our work for the final project includes a more in-depth research and report on *how* the Dirichlet-based GP makes predictions, providing a mathematically backed reasoning behind the algorithm. Additionally, we consider best practices of data analysis as taught in this course, and incorporate experimentation and research procedures accordingly to create a well structured process. Our work now includes graphs and other modes of explanatory visual output to increase professionalism and authenticity of our findings.

The remainder of this paper is structured as follows. We briefly explore related works on travel trajectory analysis and random forest classification before explaining the fundamental theory of GPs and the Dirichlet-based GP model. We then cover our methodology, and finish with an analysis of the results and present our conclusions.

## 2. Theoretical Background

Much work has been done regarding the prediction of a person's next location, with great emphasis on finding new tools or advancements upon existing technology. In this section, we analyze some relevant literature and briefly review recent studies on travel trajectory, random forest classification, mean negative log likelihood, and GPs.

### 2.1. **Travel Trajectory Analysis.**

In recent research experiments on location prediction and travel trajectory analysis, we found that most articles described advancements upon existing technologies and methods for improving current models. For instance, Xia et. al. [9] builds upon graph network theory to acquire a best fitting distribution for their data, and Sun et. al. [8] uses a neural network variation to make a joint prediction of location prediction and arrival time. There have also been new technologies researched, such as the Travel Time Difference Model Liu et. al. [3] proposed, which is integrated into a Markov model to decrease time and space algorithmic complexity. Furthermore, as a point of inspiration, it is exciting to note that Kong et. al. [1] successfully utilized a GP regression model to act as a taxi recommender system using time, location, and the get-on/off relationship data.

### 2.2. **Random Forest Classifiers.**

Random forest (RF) classification is a tried and true method for successfully predicting labels. This learning method is known as an ensemble of decision trees, meaning it utilizes multiple decision trees that each produce individual results, and reports the outcome of the majority vote. Decision trees on their own lack the ability to perform complex multi-class classification. However, a collection of trees can learn complex tasks, are robust to noise due to the individual decisions being made independently, computationally very fast by virtue of parallel decisions. RF can be optimized via two hyperparameters: the number of layers (depth) and the number of trees per layer (density).

In their review of the random forest classifiers, Parmar et. al. [5] note that it has high rates of success as a multi-class classification system in numerous fields, including biology, sociology, and image recognition. Due to its independent resampling methods, random forest is able to train relatively quick, and is also robust to noise by virtue of bootstrap theory. Unfortunately, random forest suffers similarly to neural networks due to its hidden computation stages before the majority vote. Xiao et. al. [10] discuss how their "future work will investigate how to explain the ensemble model more intuitively, as the ensemble method is like a black box and is not easy to intuitively explain."

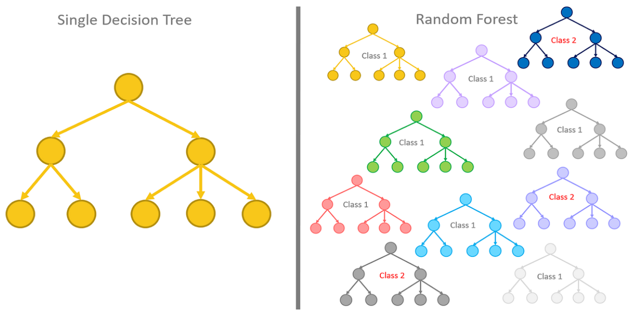


FIGURE 1. Decision Tree and Random Forest depiction. Source: Towards Data Science [6]

## 2.3. **Mean Negative Log-Likelihood.**

The mean negative log-likelihood, or MNLL, is an error rate defined as:

$$\text{MNLL} = -\frac{1}{|\mathbf{X}|}\sum \log\, p(y|\mathbf{X}, \mathbf{Y}, x),$$

where $\mathbf{X}$ is the set of input data, with $x$ as the input data for a particular label $y$, and $\mathbf{Y}$ as the set of labels. This metric is defined such that a lower value indicates better performance.

## 2.4. **Gaussian Processes.**

GPs are non-parametric statistical learning algorithms, which are useful when we cannot make assumptions regarding the distribution of the data [2]. In general, the goal of a GP is to estimate the distribution over the class probabilities. Theoretically, it is a multivariate extension of the Gaussian distribution to infinite dimensions, making it very useful for classification tasks using large amounts of features. Computationally, a GP can be thought of as a neural network with a single hidden layer of infinite width–infinite neurons within that layer. Each neuron is assigned a weight, and a unique distribution is associated with each weight. The weights can then be interpreted as the likelihood of the distribution, and are adaptively altered based on the input data. This allows GPs to be extremely flexible as they do not require an input function and are able to effectively interpolate predictions from past data.

Moving one step further to the Dirichlet-based GP, we note that it is unlike conventional GP classifiers, which are limited in their applicability to "small/moderate size of datasets, due to the high computational complexity of approximating the true posterior distribution" [4]. Instead, this new model allows the conversion of a multi-class classification problem to a regression problem by implementing a pre-processing step, where labels are transformed to Dirichlet distributed random variables. This method drastically reduces the training time while still maintaining high accuracy standards compared to conventional GP classification.

More concretely, suppose $\pi$ is a vector of class probabilities such that $\pi\ Dir(\alpha)$, where $\alpha$ is the vector of concentration parameters. Our goal is to estimate $\alpha$ such that a best-fitting degenerate Dirichlet distribution covers each class probability. In other words, we want to estimate the concentration parameters such that the Dirichlet distribution for a given class focuses on a single probability (the correct label) and sends all other probabilities close to 0. Since the performance of the classifier is dependent on the optimization $\alpha$, Milios et. al. [4] suggest choosing $\alpha$ such that the MNLL is minimized.

## 3. Algorithm Implementation and Development

In this section, we discuss the algorithms we implement. Specifically, we discuss our dataset, the frequency map, our random forest classifier, and our Dirichlet-based GP classifier. Our work is done in a jupyter notebook on Google CoLab with packages from GPyTorch, PyTorch, pandas, matplotlib, and sklearn.

### 3.1. **Dataset.**

Data is used with permission from a mobility data study user on their daily travel habits. The data is tabular, and contains information such as date-times for leaving/arriving, cluster, activity duration, etc. In this report, we use the user's past two locations, their leaving hour from the last location, and the day of week to predict their next location. It contains 462 datapoints, with each event defined as arriving at a cluster. Clusters are unnamed locations with the convention that cluster 0 is the most frequently visited area, and higher cluster values indicate decreasing visitation frequency. The max cluster value in this dataset is 20. Note that in general, we interpret cluster 0 as home. We split our dataset into train/val/test sets, with 258/111/93 samples respectively. Since our inputs are all discrete data types (last locations, hour of day, and day of week), we proceed with this data engineering process carefully. We one-hot encode these afore mentioned features such that we avoid weights being assigned to these discrete data types.

### 3.2. **Frequency Map.**

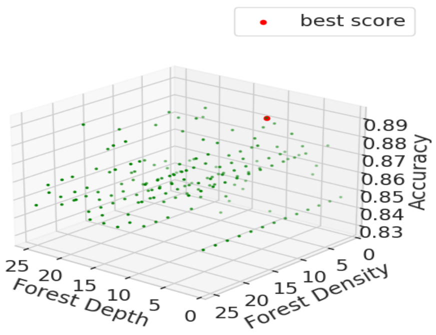
The frequency map is a custom tool made for this report to help identify the error from both the random forest and GP classifiers. Given every combination of the input features, we tally the number of times it produces each unique output, and record the conditional frequency of the target given the inputs. This is useful as an analysis tool to help us be more or less forgiving/critical to the model results, since cluster prediction is essentially boiled down to frequency.

### 3.3. **Random Forest.**

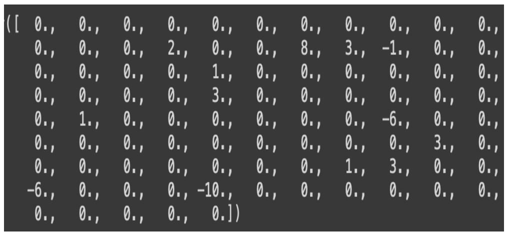
To construct our random forest (RF) classifier, we use the sklearn decision tree classifier. We first perform a grid-search optimization on depth and density, with values ranging from 1 to 26. The classifier is optimized on the validation set. We then test our model, observe our inaccuracy chart, and construct our frequency map.

### 3.4. **GP Classifier.**

We construct our model using stock code provided by GPyTorch documentation on Dirichlet-based classifiers. We use the AdaM optimizer and the marginal log-likelihood loss function. For training, we depend on the GPyTorch training function, which wraps all optimization procedures into a single call. It is a similar process to training on PyTorch, with forward and backward calls used to update the gradients. The only difference, from our understanding, is that the accuracy is not accounted for, and inference cannot be run while training the model. As such, we do not output training nor validation accuracy curves-only training loss curves. Still, this is sufficient for visualizing how our model is performing. We train the GP model over 150 epochs.



(A) RF Hyperparameter Optimization.



```
([  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
    0.,   0.,   0.,   2.,   0.,   0.,   8.,   3.,  -1.,   0.,   0.,
    0.,   0.,   0.,   0.,   1.,   0.,   0.,   0.,   0.,   0.,   0.,
    0.,   0.,   0.,   0.,   3.,   0.,   0.,   0.,   0.,   0.,   0.,
    0.,   1.,   0.,   0.,   0.,   0.,   0.,   0.,  -6.,   0.,   0.,
    0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   3.,   0.,
    0.,   0.,   0.,   0.,   0.,   0.,   0.,   1.,   3.,   0.,   0.,
   -6.,   0.,   0.,   0., -10.,   0.,   0.,   0.,   0.,   0.,   0.,
    0.,   0.,   0.,   0.,   0.])
```

(B) RF Actual - Predicted Location Results

Figure 2. Optimization Visualization and Results for RF Classifier.

## 4. Computational Results

In this section, we discuss our results from random forest classification and GP classification. Plots are graphed using the python matplotlib package.

### 4.1. **Random Forest.**

RF provided a competitive baseline of 86% testing accuracy, with optimized settings of 7 layers deep and 7 trees per layer. A plot of the results from grid-search optimization is shown 2a, as well as a visual representation of testing results calculated by the actual cluster minus predicted cluster 2b. We reconstruct the frequency table below for better viewing 1. In particular, we note that from the difference table above, we interpret our results as: 0's being correct predictions; positive values being undershot predictions (chose something more frequent); negative values being overshot predictions (chose something less frequent).

For instance, we can see that Index 17, 52, 77, and 81 make drastically incorrect classifications, off by at least a popularity of 6. However, it is understandable for sample 17, as the actual cluster 10 is much less frequently visited than cluster 2, which was predicted, so the irregularity from this misclassification is not worrisome. However samples 52, 77, and 81 are overshot, meaning a less frequently visited cluster was predicted. This is unexpected behavior, as we would assume our model would choose something more commonly seen rather than less common, leading us to believe that random forest randomly chose those values, which is sub-optimal classification behavior.

### 4.2. **GP Classifier.**

Using the given training optimizer from GPyTorch, we captured the training loss in a graph 3. We achieve an 89% testing accuracy, which is higher than the random forest baseline. Similarly as above, we construct a frequency map, and output the results in a table for better visibility 2. Interestingly, we see that there are significantly less misclassifications by the GP classifier greater than a popularity distance of 1, and further, that it never overshot its misclassification. In other words, we can see that there are a significant amount of 1 values under "Actual - Pred" 2, and the other three misclassifications are also non-negative, and relatively close to the actual cluster. At index 17, the GP model predicts 5, when the actual cluster is 10, which is not concerning as 10 is a very infrequently visited location.

We plot scatterplots of misclassifications, focusing on RF vs Actual, GP vs Actual, and then both RF and GP on the same plot 4. Here, we can visually see how wide spread and varied the RF classifier's predictions are. On the other hand, all but one of the GP misclassifications are within

| Sample Index | Actual Cluster | Pred Cluster | Actual - Pred |
|---|---|---|---|
| 14 | 2 | 0 | 2 |
| 17 | 10 | 2 | 8 |
| 18 | 3 | 0 | 3 |
| 19 | 0 | 1 | -1 |
| 26 | 1 | 0 | 1 |
| 37 | 3 | 0 | 3 |
| 45 | 1 | 0 | 1 |
| 52 | 3 | 9 | -6 |
| 64 | 3 | 0 | 3 |
| 73 | 2 | 1 | 1 |
| 74 | 3 | 0 | 3 |
| 77 | 2 | 8 | -6 |
| 81 | 0 | 10 | -10 |

Table 1. Above are the misclassification results from the RF classifier.

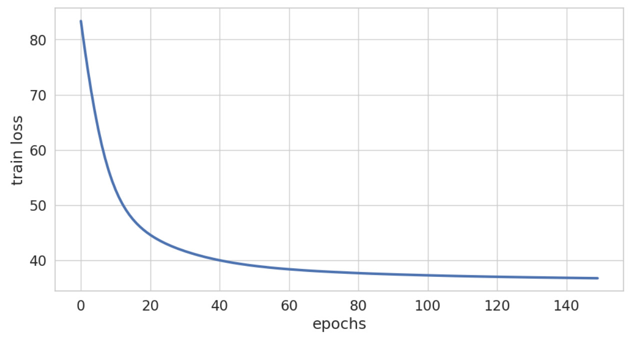
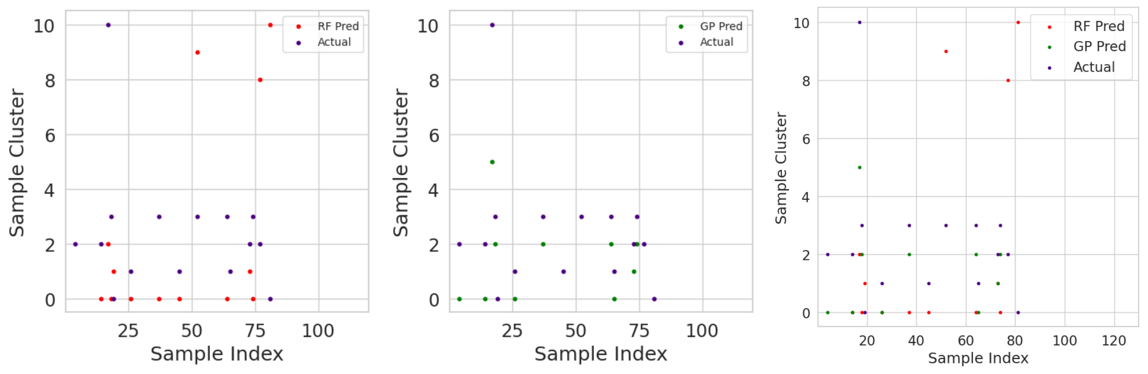


Figure 3. The training loss of the Dirichlet-based GP classifier over 100 epochs.

| Sample Index | Actual Cluster | Pred Cluster | Actual - Pred |
|---|---|---|---|
| 4 | 2 | 0 | 2 |
| 14 | 2 | 0 | 2 |
| 17 | 10 | 5 | 5 |
| 18 | 3 | 2 | 1 |
| 26 | 1 | 0 | 1 |
| 37 | 3 | 2 | 1 |
| 64 | 3 | 2 | 1 |
| 65 | 1 | 0 | 1 |
| 73 | 2 | 1 | 1 |
| 74 | 3 | 2 | 1 |

Table 2. Above are the misclassification results from the Dirichlet-based GP classifier.

2 popularity distance of the actual. This leads us to believe that the GP classifier has a better ”understanding” of the relationship between the inputs and the outputs.



(A) RF Misclassifcation. (B) GP Misclassifcation. (C) RF and GP.

Figure 4. Plots of misclassifications for RF and GP.

## 5. Summary and Conclusions

According to Song et. al., roughly 93% [7] of human travel trajectory behavior is predictable, which is similarly reflected in our results. It is exciting to see the power of the Dirichlet-based GP and its empirically tested ability to achieve close results on travel trajectory data compared to the ensemble RF classifier. Although some of this work was done before hand, as noted in the introduction, we developed a more scientific and rigorous approach to both implementing classification tools and analyzing our results. These methods were introduced to us in AMATH 482/582, and we strive to uphold a high standard of professionalism in this final report, compared to the casual nature of the previous work. Additionally, we conduct a more thorough analysis of our data, identifying the ”closeness” of the GP classifier's misclassifications, and the stark comparison to the RF classifier's varied and seemingly nonsensical guesses. We conclude that Dirichlet-based GP classifiers have potential as strong models for human travel trajectory prediction tasks.

Although much improvement has been made to our approach and analysis, we do have a few points to improve on. First, we would like to investigate further on accessing the training accuracy, as we feel that is important to analyze to prevent overfitting of the data. Second, we hoped to show the transparency of the classifier through the covariance matrix provided by the GP model, but we are unsure how to represent the results to showcase this wonderful feature. As such, we hope to conduct more research and explore that area further. Finally, we recognize in hindsight that a random forest classifier, while a strong predictor, is not necessarily the SOTA sequence based learning algorithm. This title generally belongs to deep learning processes such as recurrent neural networks (RNN), long-short term memory (LSTM) RNNS, and transformer-based deep networks. We would be interested in experimenting further with the afore mentioned models, as we know deep learning requires significant amounts of data to perform well, and the GP is not beholden to such a restriction.

## Acknowledgements

The author is thankful for the support of Professor Cynthia Chen and Ekin Ugurel from THINK Lab, as well as access to the dataset. Additionally, the author would like to thank the AMATH 482/582 course staff for their guidance and facilitation during this challenging and enlightening course.

## References

[1] X. Kong, F. Xia, J. Wang, A. Rahim, and S. K. Das. Time-location-relationship combined service recommendation based on taxi trajectory data. *IEEE Transactions on Industrial Informatics*, 2017.

[2] M. Kupilik. An introduction to gaussian processes for spatial data (predictions!). Technical report, University of Alaska Anchorage, 2016.

[3] Q. Liu, Y. Zuo, X. Yu, and M. Chen. Ttdm: A travel time difference model for next location prediction. *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, 2019.

[4] D. Milios, R. Camoriano, P. Michiardi, L. Rosasco, and M. Filippone. Dirichlet-based gaussian processes for large-scale calibrated classification. *Advances in Neural Information Processing Systems*, 2018.

[5] A. Parmar, R. Katariya, and V. Patel. A review on random forest: An ensemble classifier. *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*, 2019.

[6] R. Silipo and K. Melcher. From a single decision tree to a random forest. Technical report, Towards Data Science, 2019.

[7] C. Song, Z. Qu, N. Blumm, and A. Barabsi. Limits of predictability in human mobility. *Science 327(5968), 1018–1021*, 2010.

[8] J. Sun and J. Kim. Joint prediction of next location and travel time from urban vehicle trajectories using long short-term memory neural networks. *Transportation Research Part C: Emerging Technologies*, 2021.

[9] F. Xia, J. Wang, X. Kong, Z. Wang, J. Li, and C. Liu. Exploring human mobility patterns in urban scenarios: A trajectory data perspective. *IEEE Communications Magazine, vol. 56, no. 3*, 2018.

[10] Z. Xiao, Y. Wang, K. Fu, and F. Wu. Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. *ISPRS International Journal of Geo-Information*, 2017.