Efficient Location Sampling Algorithms for Road Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many geographic information systems applications rely on data provided by user devices in the road network, including traffic monitoring, driving navigation, and road closure detection. The underlying signal is generally collected by sampling locations from user trajectories. The sampling process, though critical for various applications, has not been studied sufficiently in the literature. While the most natural way to sample a trajectory may be to use a frequency based algorithm, e.g., sampling locations every x seconds, such a sampling strategy can be quite wasteful in resources (e.g., server-side processing, user battery) as well as stored user data. In this work, we conduct a horizontal study of various location sampling algorithms (based on frequency, road geography, reservoir sampling, etc.) on the road network of New York City and assess their trade-offs in terms of various metrics of interest, such as the size of the stored data and the induced quality of training for prediction tasks (e.g., predicting speeds).

1 Introduction

For many geographic information systems (GISs) that operate on road networks, the input received from user devices is vital for the purposes of providing services back to the users. Specifically, trajectories obtained from user devices can help in a wide range of downstream tasks. Examples of such tasks include identifying new roads and correcting the locations of nodes and edges in the graph [3], mobility prediction and next point of interest recommendation [1], and of course the central application of monitoring the road network and estimating delays on road segments [4], which in turn enables other use cases such as routing with dynamic information [2, 10] and global optimization in the road network [5]. The trajectories that enable these applications typically come in the form of timestamped location (e.g., GPS) samples. Various problems have been studied with respect to these timestamped location collection, such as completing long gaps in the trajectory [6] or efficiently processing these trajectories [9]. However designing exactly how the device samples locations and how the server decides what to use and what to discard has received little attention.

In this work we will focus precisely on this problem and we will explore different algorithms and strategies for sampling location data from devices. We begin with what is the most natural and most widely used method: periodic sampling. This method relies on a fixed frequency at which the device sends a location sample to the server. Our study is motivated by the fact that this method is inherently wasteful. First, think of a congested highway where devices stuck in the same location are periodically transmitting the same information over and over. This results in a lot of data being transferred and processed, wasting user battery and server processing resources for little information in terms of the state of the road network. Second, this method ignores the unit that forms the road network: the *road segment*. Road segments are the real-world counterpart of the edges of the graph on which GIS algorithms typically operate and direct measurements of their delays are important.

Submitted to ICML 2023 Workshop: Sampling and Optimization in Discrete Space. Do not distribute.

Periodic sampling will instead use interpolation to identify the timestamps during which a car entered and exited a segment, assuming a constant speed between location samples.

This interpolation process is precisely what makes delay measurements sensitive to the exact sampling policy used. Consider a simple example of two consecutive 50 meter road segments with and average traversal time of 10 seconds for the first segment and 20 seconds for the second segment. Consider a device that passes through these segments and gives a location sample at the start of the first segment and another one 20 seconds later. If this device travels at the average speed of each segment, the second sample will be in the middle of the second segment. Assuming a uniform speed between the samples, we would get that this uniform speed will be 3.75 m/s (since the car would have driven 75 meters in 20 seconds) and we would get that the perceived traversal time for the first segment is $50/3.75 = 13.333$ seconds. Similarly a device that gives a location sample at the start of first segment and another at the end of the second segment would give a perceived traversal time of 15 seconds for the first segment. Finally, a device that provides a location sample exactly every 10 seconds (starting from the beginning of the first segment) would correctly identify the end of the first segment and give a (correct) perceived traversal time of 10 seconds.

The above discussion suggests that it is an interesting question to (i) study the locations at which devices should provide location samples and (ii) determine which of these samples can be dropped using sub-sampling strategies. To this end, we study several classes of algorithms for this problem. In all classes of algorithms, tweaking a corresponding parameter changes the number of location samples given by the devices. Typically, more samples will improve the signal given by the algorithm, meaning the performance of downstream tasks such as speed/traversal time prediction is improved. To understand the trade-off for each class of algorithms and their relative performance, we conduct an experimental analysis. We create synthetic user trajectories in the network of New York City and use the induced congestion to define road segment delays. We then apply the different sampling algorithms on the synthetic trajectories to generate training data for speed estimation. We present results on the performance of the different algorithms and parameter choices with respect to the number of locations sampled versus the quality of the predictions.

2 Sampling Methods

In this section, we given an overview of various algorithms that we consider for sampling location data. Our algorithms are tailored toward our special use case of road networks and they are based on well-known sampling algorithms in the literature.

2.1 Uniform Sampling

The uniform sampling algorithm is perhaps the most natural one and the one most widely used in practice, due to its simplicity. In uniform sampling, we want to sample with a preset frequency. This frequency is set in advance and it is communicated to each device so as the device travels their assigned path, they will send related data at the requested frequency. It is easy to see that in cases that there is congestion or device is traveling on a high-traffic road, we collect too many unnecessary samples. This class of algorithms is parameterized by the sampling frequency.

2.2 Randomized Segment-Based Sampling

In this approach, roads are broken down into smaller segments. Each device provides a location sample as close as possible to the start of a new segment. This is typically not precise, due to GPS noise, or due to limitations on the device's ability to ping frequently. We parameterize this class of algorithms with a probability of measurement p. Each device provides a measurement for any given road segment with probability p. Note that to provide such a measurement the device will need to ping at the beginning of the segment and at the beginning of the next segment.

2.3 Congestion-Based Sampling

This is another class of segment-based sampling algorithms. The difference from randomized segment-based sampling is in how the probability of pinging is determined. In order to avoid having unnecessary samples, we propose a probability of pinging that correlates with the inverse of the congestion. Now one challenge here is that, we want to have independent (decentralized) sampling on each device but in order to get the congestion, we need to consider all devices traveling on the road. To circumvent this difficulty, we rely on additional data on the road such as expected traveling speed or speed limit, if we observe that our traveling speed is much lower then we lower our sampling probability. The parameter of this class of algorithms is a scaling factor that is applied to the congestion-based probability. We can think of this scaling factor as the expected number of samples we wish to extract per road segment.

2.4 Reservoir Sampling

This class of algorithms also relies on segment-based sampling. The idea of this sampling algorithm, is that more samples is not necessarily better. By the law of large numbers, the larger the sample size the more the sample will look like the population, so we want to make sure we get enough samples that can help us estimate the average speed correctly, however the gain beyond certain point becomes marginal. To this end, we consider Reservoir sampling which chooses k samples from a list of n objects where n is not required to be known in advance. Not requiring the number of objects in advance is a positive point, that both helps with running sampling methods in streaming settings and also not collecting too many samples for large n. More precisely, in this sampling, we put the first k items in the reservoir list and iteratively go through the remaining items by drawing a random number x between 1 and t for the t-th item, and then swapping this item with x-th item in the reservoir if $x \leq k$. The caveat about this method is that it requires each item to know where in the list they are[1] which means that we need to have a centralized algorithm. Since routes are communicated through a centralized platform, we can ask the platform to communicate an index for each road to each device and then the device can decide to send their data if their generated number falls below k for their road segment. Note that this approach works with uniform k or specialized k for each road, however that needs to be communicated to the device traveling on the road. The obvious parameter for this class of algorithms is the size of the reservoir, k.

3 Experiments

We conduct experiments on synthetic user trajectories built on the real network of New York City. The use of synthetic trajectories is due to the fact that real trajectories are not available one the public data sets due to their sensitive nature. We now describe how we obtained these trajectories. We extract the map of New York City from OpenStreetMap [8] and generate a congestion function for each road segment using the functional form of the Bureau of Public Roads [7]. Specifically, for an edge (road segment) e with traffic demand x, we set:

$$l_e(x) = \frac{0.6 t_e^f}{c_e^4} x^4 + t_e^f,$$

where t_e^f is the time needed to cross the edge when the road is empty, i.e., the free-flow travel time, and c_e is the capacity of the street, defined as the number of lanes multiplied by the free-flow speed. We then partition the map into 8 areas and pick a random travel demand for each pair or areas. For each demand, we compute a set of (at most 10) candidate routes between the origin and destination using the alternative route computation algorithm from [10]. We split the demand equally over these candidate routes. This process induces a certain congestion on each road segment, which in turn determines the average travel time of all cars passing through it by means of applying the congestion function $l_e(x)$. For each car and each segment that it passes through, we add a noise parameter, which finally gives rise to our ground truth user trajectories. This noise models how different cars traverse the same road segment with different delays and is the cause of downstream prediction errors.

We apply each one of the sampling algorithms to each one of these "ground truth" trajectories. Each one of our four algorithms types uses a parameter (the sampling frequency for uniform sampling, the sampling probability for random sampling, the expected number of samples per segment for congestion-based sampling, and the size of the reservoir for reservoir sampling) and we select 4 values of each parameter, giving rise to 16 total algorithms. Using the location samples selected by the algorithms, we recompute the perceived road segment entry and exit timestamps by interpolating

[1]Note that the order does not matter, however we need to assign a unique number from 1 to n to each of the items

Sampling Method	Parameter	MSE	MAE	MBE	Sample Size
Uniform Sampling	5	6.90	1.35	0.40	1252746
Uniform Sampling	10	11.34	2.03	0.38	630995
Uniform Sampling	15	17.63	2.68	0.36	423639
Uniform Sampling	20	24.25	3.17	0.34	320088
Congestion-Based Sampling	10	5.68	0.77	0.46	35104
Congestion-Based Sampling	20	4.90	0.73	0.40	66116
Congestion-Based Sampling	30	5.38	0.74	0.41	92717
Congestion-Based Sampling	40	4.59	0.73	0.41	115688
Reservoir Sampling	10	4.59	0.72	0.43	33954
Reservoir Sampling	20	4.42	0.72	0.42	57404
Reservoir Sampling	30	4.29	0.71	0.41	76407
Reservoir Sampling	40	4.51	0.72	0.41	93329
Randomized Segment-Based Sampling	25	4.47	0.72	0.41	129616
Randomized Segment-Based Sampling	50	4.37	0.71	0.41	223111
Randomized Segment-Based Sampling	75	4.41	0.71	0.40	279424
Randomized Segment-Based Sampling	100	4.42	0.71	0.40	299244

Table 1: Performance of various sampling algorithm on test data set.

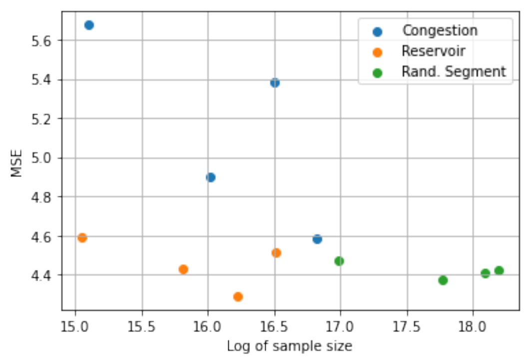

Figure 1: The mean square error of sampling algorithms on test data. X-Axis is on log scale and the Uniform sampling for more focused view.

between location samples and assuming a constant speed. This process induces a different perceived road segment travel time for each location sampling strategy. These perceived travel times are in fact our training data for road segment delay prediction. We have 16 training data sets, one per algorithm.

Since we only have travel time and not much additional information on segment properties or trajectories, for square loss minimization, the average speed values for each segment, yields the optimal prediction. We then use this calculated average per segment and evaluate each algorithm on the test data set which again is from the trajectories that were sampled uniformly and excluded from the training. All trajectories are from the same time period. For evaluate our algorithms by reporting the mean square loss (MSE), mean aboslute loss (MAE), and mean biased error (MBE) along side the number of samples in the training set (see Table 1).

We also plot the trade-off between the size of the sample set and the quality of the predictions for each algorithm in Figure 1. We exclude the Uniform sampling since it requires a lot of samples and has high error. Since we add noise to our data, we don't necessarily see decrease in the reported error as we allow more samples. Based on reported numbers, it is easy to see that the Randomized segment-based and Reservoir sampling have the best performance. The Congestion based sampling starts to get more competitive results as we allow more samples. Overall, the Reservoir sampling yields the best result, especially if we consider the Pareto curve where we prefer a point with lower error and lower sample size, we see that the curve consists of the three points on the bottom left corresponding to the Reservoir sampling.

References

[1] Q. Chen, R. Jiang, C. Yang, Z. Cai, Z. Fan, K. Tsubouchi, R. Shibasaki, and X. Song. Dualsin: Dual sequential interaction network for human intentional mobility prediction. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '20, page 283–292, New York, NY, USA, 2020. Association for Computing Machinery.

[2] D. Delling, A. Goldberg, T. Pajor, and R. Werneck. Customizable route planning. In *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*, Lecture Notes in Computer Science. Springer Verlag, May 2011.

[3] A. Hendawi, S. S. Sabbineni, J. Shen, Y. Song, P. Cao, Z. Zhang, J. Krumm, and M. Ali. Which one is correct, the map or the gps trace. SIGSPATIAL '19, page 472–475, New York, NY, USA, 2019. Association for Computing Machinery.

[4] W. Jiang and J. Luo. Graph neural network for traffic forecasting: A survey. *CoRR*, abs/2101.11174, 2021.

[5] K. Kollias, A. Chandrashekharapuram, L. Fawcett, S. Gollapudi, and A. K. Sinop. Weighted stackelberg algorithms for road traffic optimization. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '21, page 57–68, New York, NY, USA, 2021. Association for Computing Machinery.

[6] J. Krumm. Maximum entropy bridgelets for trajectory completion. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '22, New York, NY, USA, 2022. Association for Computing Machinery.

[7] B. of Public Roads. *Traffic assignment manual*. US Department of Commerce, 1964.

[8] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2017.

[9] S. A. Shaikh, H. Kitagawa, A. Matono, and K.-s. Kim. Tstream: A framework for real-time and scalable trajectory stream processing and analysis. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '22, New York, NY, USA, 2022. Association for Computing Machinery.

[10] A. K. Sinop, L. Fawcett, S. Gollapudi, and K. Kollias. Robust routing using electrical flows. In X. Meng, F. Wang, C. Lu, Y. Huang, S. Shekhar, and X. Xie, editors, *SIGSPATIAL '21: 29th International Conference on Advances in Geographic Information Systems, Virtual Event / Beijing, China, November 2-5, 2021*, pages 282–292. ACM, 2021.