CS5350/6350 Project Midterm Report

Simple Distances for Trajectories via Landmarks [Phillips and Tang, 2019a]

Hasan Pourmahmood [u1255635@utah.edu]
Aravinda Kanchana Ruwanpathirana [u1209978@utah.edu]

1 Introduction

There are many different popular metrics that can be used (e.g. Discrete Frechet distance, dynamic time wrapping distance, etc.) to measure the distance between two trajectories (i.e. piecewise linear curves), but most of them have drawbacks when it comes to using them in machine learning tasks. The paper, [Phillips and Tang, 2019a] introduces a new metric that data-based (based on special landmarks) distance, which might be more desirable in a machine learning setting. Our goal is to test the claims in the paper against different machine learning tasks and on different classes of datatsets and also see how the proposed metrics can be modified for different tasks.

2 Preliminaries

In this section we bring main concepts and definitions we need in the project.

A continuous piece-wise linear curve $\gamma : [0,1] \to \mathbb{R}^2$ is called a trajectory. We denote the set of trajectories by Γ. So any $\gamma \in \Gamma$ can be identified by a series of critical points $\langle c_0, c_1, \ldots, c_k \rangle$, where we assume that $s_i = c_{i-1} - c_i$ is the line segment between critical points c_{i-1} and c_i.

Let $Q = \{q_1, \ldots, q_n\}$ be a set of landmarks in $\mathbb{R}^2$ and let $\gamma \in \Gamma$. For each $i \in \{1, 2, \ldots, n\}$ define $v_i(\gamma) = \|p_i - q_i\|$, where $p_i = \text{argmin}_{p \in \gamma} \|p - q_i\|$, and set $v(\gamma) = (v_i(\gamma), \ldots, v_1(\gamma))$.

The following distance is defined on Γ in [Phillips and Tang, 2019a], which is a data based metric.

For a fixed set of landmarks $Q = \{q_1, \ldots, q_n\}$ the distance between two curves $\gamma, \gamma' \in \Gamma$ is the normalize Euclidean distance of $v(\gamma)$ and $v(\gamma')$, i.e.

$d_Q(\gamma, \gamma') = \left(\frac{1}{n} \sum_{i=1}^{n} |v_i(\gamma) - v_i(\gamma')|^2 \right)^{1/2}$.

Another metric defined there is

$d_Q^{\pi}(\gamma, \gamma') = \frac{1}{n} \sum_{i=1}^{n} \|p_i(\gamma) - p_i(\gamma')\|$,

where $p_i(\gamma) = \text{argmin}_{p \in \gamma} \|p - q_i\|$ and $p_i(\gamma') = \text{argmin}_{p \in \gamma} \|p - q_i\|$.

It is worth mentioning that these metrics are generalized to a larger set of curves, not just for trajectories, in [Phillips and Tang, 2019b], but they are more of interest mathematically.

3 Progress Summary

The Section 3.1 gives an overview of the data that has been acquired and the Section 3.2 gives the details of the implemented programs. The Section 3.3 gives an overview of the results we have seen so far, from the preliminary tests.

3.1 Dataset Acquisition

Dataset acquisition phase included the procurement of the following major datasets which include the location data of people/ taxis and etc.

1. Geolife Trajectories 1.3 [Zheng et al., 2008, 2009, 2010]

   Geolife Trajectories dataset is a GPS trajectory dataset that was collected as part of the Geolife project by Microsoft Research Asia and includes data of 182 users from 2007 to 2012, with the points being $\approx 5 - 10$ meters apart. .

2. T-drive Taxi Trajectories [Yuan et al., 2010]

   The T-drive Taxi Trajectory dataset contains trajectory info from 10357 taxis in Beijing for the time duration from February 2 to 8, 2008 with about 15 million

GPS location points in total. The sampling was carried out at a rate of once every $177s$ and about every $623m$.

3. GPS Trajectories Data Set [Cruz, 2015]

   This is a relatively small dataset UCI Machine Learning repository and this contains 163 data-points.

4. Manhattan Taxi Trajectory Dataset [Benson et al., 2017]

5. CVRR Trajectory Clustering Dataset [Morris and Trivedi, 2009]

6. NY City, Roma City and Porto City datasets

3.2 Implementation

The implementation is Python based. The trajectories are maintained as an ordered list of line segments and line segment class basically contains the definition of a line (start and end point) and the functions to calculate the distance to the line given a point. The distance metric class basically handles the calculation of the trajectory distances as well as the euclidean distances. At the moment the calculation algorithms implemented are naive, direct methods which we observed to have a large time complexity. The plans to mitigate this are included in the Section 4

The ML models implemented include k-Medoids algorithm which uses an iterative approach of local swaps to select a set of best k centers closer to the data. In the implementation, the landmarks are selected randomly, but this method requires further analysis. The SVM implementation relies on the SVC function provided in the Sklearn package. The distance metrics are provided as pre-computed kernels.

The data extracted from the datasets are in GPS coordinates, so we use a code snippet based on UTM package and the UTM translation equations, to convert the GPS coordinates to XY coordinates.

3.3 Experimental Results

The Section 3.3.1 gives the results from running k-Medoids with $k = 2$ and using Euclidean distance and the two distance metrics introduced. The Section 3.3.2 gives the results from running SVM with precomputed distances set as the kernel (with 2 unique labels).

3.3.1 k-Medoids Clustering

The testing was carried out on a small sample of trajectories (~ 20), each of length 100 from 2 users in the Geolife dataset. The landmark points were randomly generated within a confined area around the spread of the trajectories. Multiple trials were carried out each with maximum 10 iterations so as to see the effect of the different metrics.

	Euclid	d_Q	d_Q^π
mean	0.32	0.325	0.375
min	0.15	0.2	0.2
max	0.45	0.45	0.45
median	0.3	0.3	0.425
modes	0.45	0.45	0.45

Table 1: The mean, median and other statistical properties of the k-Medoid error with Euclidean distance, d_Q and d_Q^π

The error rates in Table 3.3.1 could be seen to be the same for medians, modes, and max in all 3 distance setups, but the Euclid distance seems to perform well in this case when we compare the mean and the minimums.

3.3.2 Kernelized SVM

The setup is similar to the case of 3.3.1 but in this case we used ≈ 40 samples.

	Euclid	d_Q	d_Q^π
mean	0.6325	0.585	0.6425
min	0.5	0.5	0.5
max	0.95	0.875	0.95
median	0.55	0.525	0.55
modes	0.5	0.5	0.5

Table 2: The mean, median and other statistical properties of the SVM error with Euclidean distance, d_Q and d_Q^π

The performance results of SVM (provided in Table 3.3.2) can be seen to be better when

the metric d_Q is used. The performance of the Euclidean distance and d_Q^π are almost the same (d_Q^π performs slightly worse) in the selected dataset.

4 Future Plans

The following tasks are planned for the next phase of the project.

1. Update and optimize the distance calculations removing dependencies on Sklearn's SVC

   The plan is to optimize the distance calculation by reducing the effective trajectory components from a given landmark point using distance cutoff radii. This will be useful in mitigating the performance issues when it comes to local algorithms such as k-Medoids. For the SVM, we plan to implement from-scratch module that would enable us to see how different kernel structures (rbf, linear and etc.) perform, when the distance measures are substituted by trajectory distance measures.

2. Exploring how the metric can be used in a decision tree setting.

   The idea in this case is to view the individual distances from the landmark points as individual features. We plan to look into how using a distance measure from the trajectories and training on a discretized version of them would be applicable in a decision tree setting. The hope is that this will be useful in measuring the quality of the effect of each landmark on the trajectories and would thus help reduce the number of landmarks used at a minimal loss.

3. Implementing Discrete Frechet Distance and Dynamic Time Warping Distance for benchmarking.

4. Exploring other machine learning tasks for which the given trajectory distance measure could be useful.

5. Exploring the effect of dimensionality reduction (using MDS, t-SNE and etc.) using the given metric

   We hope to look into how the data would perform if the trajectories were mapped to a point space by running dimensioanlity reduction techniques and the effect of mapping the trajectories to points in a n-dimensional space, where n is the number of landmarks. This task will be carried out if the time permits.

6. Obtaining the landmarks given the trajectories and the distances.

   We hope to implement the algorithm introduced in [Phillips and Tang, 2019b] and try to obtain the landmark points from the trajectories. This task will be carried out if the time permits.

References

A. R. Benson, D. F. Gleich, and L.-H. Lim. The spacey random walk: A stochastic process for higher-order data. 2017.

M. Cruz. Grouping similar trajectories for carpooling purposes. 2015.

B. Morris and M. Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. 2009.

J. M. Phillips and P. Tang. Simple distances for trajectories via landmarks, 2019a.

J. M. Phillips and P. Tang. Sketched mindist, 2019b.

J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: Driving directions based on taxi trajectories. 2010.

Y. Zheng, X. Xie, and W.-Y. Ma. Understanding mobility based on gps data. 2008.

Y. Zheng, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. 2009.

Y. Zheng, X. Xie, and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. 2010.