Predicting clustered locations using HMMs with varying information levels

AUTHORS

Sadaf MD Halim, ID: 144422
Md. Najib Murshed, ID: 144432

A Thesis submitted to the Academic Faculty in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Islamic University of Technology
Bangladesh

Predicting clustered locations using HMMs with varying information levels

AUTHORS

Sadaf MD Halim, ID: 144422
Md. Najib Murshed, ID: 144432

Department of Computer Science and Engineering
Islamic University of Technology
Organization of Islamic Cooperation
Board Bazar, Gazipur-1704
Bangladesh

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and investigations carried out by Sadaf MD Halim and Md. Najib Murshed in the Academic Year 2017-2018, under the supervision of Dr. Abu Raihan Mostofa Kamal, Professor at the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Gazipur, Bangladesh. It is also declared that neither this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors

Sadaf MD Halim	Md Najib Murshed
ID: 144422	ID: 144432

Approved By

Dr. Abu Raihan Mostofa Kamal
Professor
Department of Computer Science and Engineering
Islamic University of Technology
Board Bazar, Gazipur-1704, Bangladesh

Contents

Abstract

Due to the recent explosive growth of location-aware services, predicting the next position of a user is of increasing importance in enabling intelligent information services. To this effect, we first introduce a system where the Hidden Markov Model (HMM), a very popular tool used in fields like Natural Language Processing and Bioinformatics, is used to generate predictions over clustered location data. This method prioritizes the generality of the “area” of the user in favor of his or her exact co-ordinates. Secondly, frequent data reporting is expensive, and minimizing this whilst maintaining reasonable accuracy is preferable. On the flipside, security concerns like information leakage arise where hiding a user and preventing good prediction is preferable. Both these issues require us to analyze a relationship between the accuracy of future predictions against the frequency of current data being reported. This was our second end goal – we investigated how accuracy changed with data-reporting frequency (and other parameters) in an attempt to establish a model for accuracy variation.

Introduction

We are currently living in an era where we strive for efficiency where the ultimate goal is to save time or resources. With location prediction, we are basically unlocking a door to finding mobility patterns of people which may be utilized in a multitude of ways. Popular apps like Google Maps may consider mobility patterns of certain users as a factor for suggesting routes that will be particularly attractive to the user. A typical scenario of this could be Google Maps suggesting places that you usually visit after going to a particular location and showing you optimum routes and letting you stay updated on traffic. Another example could be Maps creating customized routes for you based on places you visited and places you may visit in the near future. Provided mobility patterns are deciphered, you may be getting certain advertisements of products or places. The advertisements could be of products of certain shops that are popular or running a sale and these shops are on the way to your destination. How convenient! Another useful application of location prediction could be regarding demand forecasting in a supermarket, where the supermarket is notified of the number of people who may be heading on their way on a certain Thursday.

With all the pros we discussed above, location prediction has certain cons as well. Breaching of privacy and security of individuals by using movement trajectory data is a growing concern, especially with cases of people who do not give consent for being tracked. However, we have successfully found trends in accuracy percentages after tweaking the reporting frequency of GPS. So, the fact that reducing reporting frequency decreases future location predictions can be utilized in maintaining their whereabouts to be unknown.

Not only that, future location prediction is quite difficult on its own and accuracy percentage fluctuates in an approximate range of 10-50% due to short term human mobility. So, working on this field not only serves as a challenge but also gives us opportunity to explore relationships between density of training data and prediction accuracy. The patterns found from predicting location provides basis for studying periodicity of human mobility as well.

Investigation on reporting frequencies leads us to finding characteristics like, provided we want a certain level of prediction to be made how much must the reporting frequency be in order to obtain that level of prediction. So, in the process this leads to saving energy because now people know how much they must report. Too much reporting may lead to major information

The Strava location breach [7]

leaks, occurring on a very large scale. An example of this is a military base location in Afghanistan being compromised due to reporting too much trajectory data via an exercise app called Strava.

Problem Statement and Motivation

Predicting locations and generalizing human mobility accurately is one of the biggest research challenges at present. This is because, at least over short term periods, human mobility can be incredibly random. Therefore, predicting the “next step” can be incredibly difficult.

Furthermore, reporting trajectory data in too small intervals is one of the major reasons for draining battery-life of mobile devices. Data reporting is expensive with respect to telecommunication costs and energy. But predicting user location is key to many of the latest mobile applications and technologies, which, by definition requires location data reporting. However, keeping the GPS indefinitely turned on for these applications is very expensive and takes a toll on the battery. So, to avoid this we find ideal reporting frequencies that would allow prediction at an acceptable accuracy while still preserving battery and data.

Also, too much reporting may lead to information leakage (Strava™ leak) and this could also be prevented provided we know how to shield users from reliable prediction by finding the optimum reporting interval in their case. In their case, the reporting interval should be such that it will not be easily possible to predict future locations with high accuracy.

To address these problems, we intend to use a Hidden Markov Model in order to:

1. Suggest a way to predict location based on meaningful clustered locations
2. Investigate the relationships between different parameters and the predicted accuracy. To this end, we
    a. Visualize the relationship between data reporting frequency and accuracy
    b. Visualize the relationship between testing sequence length (explained later) and accuracy
    c. Compare k-means and k-medoids clustering and check for differences in accuracy

Initial Literature Review

Attribute Inference Attacks in Online Social Networks

ACM Transactions on Privacy and Security, January 2018
Neil ZG, Bin L [6]

Online social networks (e.g., Facebook, Google+, and Twitter) have become increasingly important platforms for users to interact with each other, process information, and diffuse social influence. A user in an online social network essentially has a list of social friends, a digital record of behaviors, and a profile. For instance, behavioral records could be a list of pages liked or shared by the user on Facebook, or they could be a set of mobile apps liked or rated by the user in Google+ or Google Play. A profile introduces the user's self-declared attributes such as majors, employers, and cities lived. To address users' privacy concerns, online social network operators provide users with fine-grained privacy settings, e.g., a user could limit some attributes to be accessible only to his/her friends. Moreover, a user could also create an account without providing any attribute information.

As a result, an online social network is a mixture of **both public and private user information**. However, often private attributes can be inferred from users' publicly available data using **inference attacks**. Existing attribute inference attacks can be roughly classified into two categories:

1) Friend-based
2) Behavior-based

However, these inference attacks consider either social friendship structures or user behaviors, but not both, and thus they achieve limited inference accuracy. As such, this paper aims to combine social structures and user behaviors to infer user attributes. To this end, they first propose a social-behavior-attribute (**SBA**) network model to gracefully integrate social structures, user behaviors, and user attributes in a unified framework.

SBA

An SBA network consists of:

1) Social Nodes
2) Behavior Nodes
3) Attribute Nodes
4) Links between the different nodes, which can have weights

We denote an SBA network as G = (V , E,w,t), where V is the set of nodes, n = |V | is the total number of nodes, E is the set of links, m = |E| is the total number of links, w is a function that maps a link to its link weight, i.e., wuv is the weight of link (u,v), and t a function that maps a node to its node type, i.e., Tu is the node type of u.

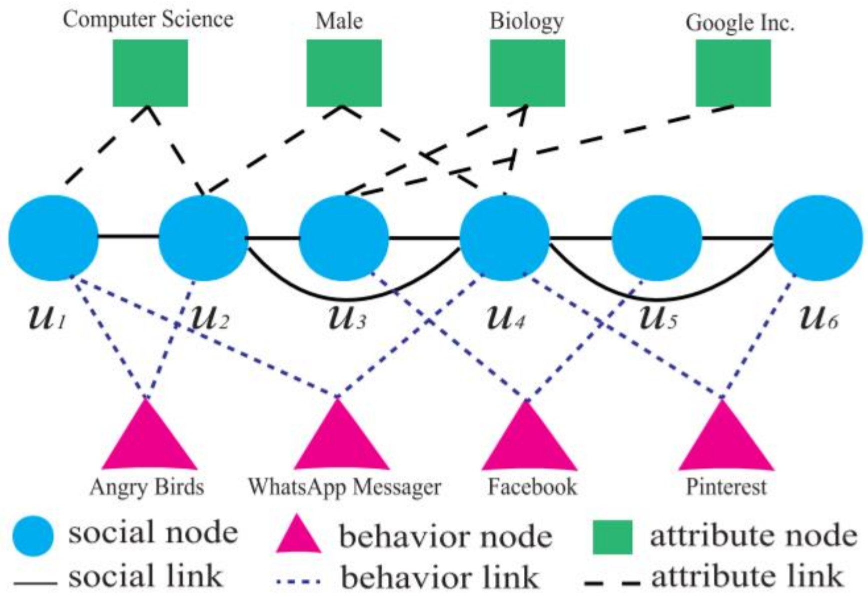


Figure: Illustration of social-behavior attribute network

VIAL

They then designed a vote distribution attack (VIAL) under the SBA network model to perform attribute inference. VIAL works in two phases.

Phase I: VIAL iteratively distributes a fixed vote capacity from the targeted user to the rest of users in Phase I.

Phase II: Each social node votes for its attribute values via dividing its vote capacity among them, and each attribute value sums the vote capacities that are divided to it by its social neighbors. We treat the summed vote capacity of an attribute value as its similarity with the target user.

Phase I

Consists of three stages:

- Dividing: The vote capacity of a social node is divided between its social, behavior-sharing and attribute sharing neighbors
- Backtracking: For each social node u, the backtracking rule takes a portion α of u's vote capacity back to the targeted user v.
- Aggregating: The aggregating rule computes a new vote capacity s for a social node u by aggregating the vote capacities that are divided to u by its neighbors.

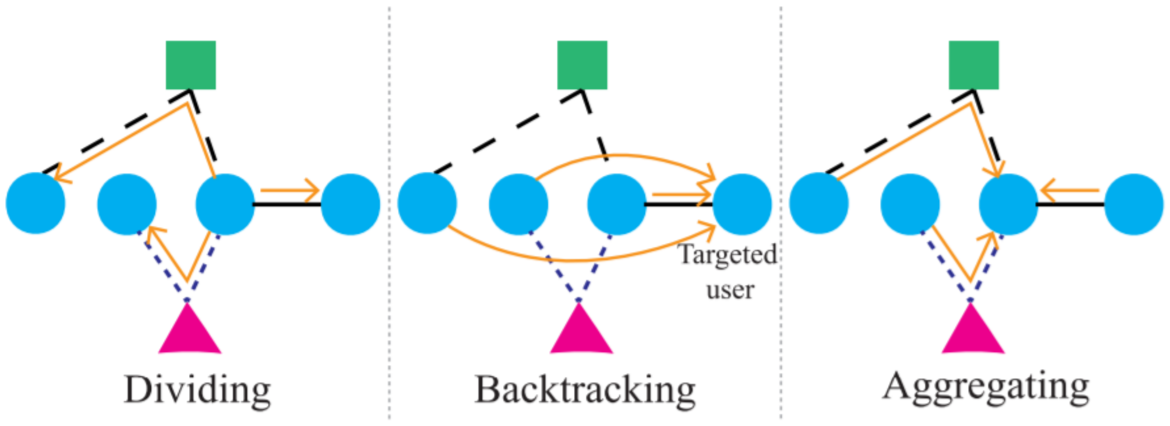


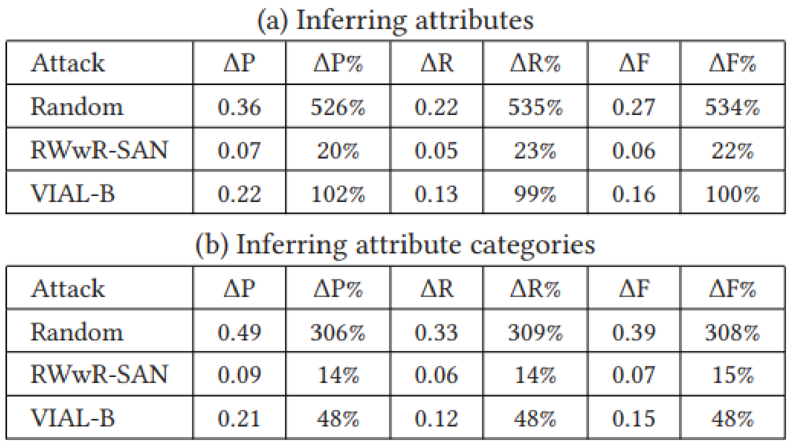
Figure: Illustration of the 3 local rules

Phase II

Here they divide the vote capacity of each user to its attribute values in proportion to the weights of the corresponding attribute links; and each attribute value sums the vote capacities that are divided to it by the users having the attribute value. They treat the summed votes of an attribute value as its similarity with v. Finally, they predict the target user has the attribute values that receive the highest votes.

Performance Evaluation

Performance was evaluated using **Precision, Recall,** and **F-Score.** As can be seen below, remarkable gains were seen across all performance metrics when compared to competing algorithms.



(a) Inferring attributes

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
Random	0.36	526%	0.22	535%	0.27	534%
RWwR-SAN	0.07	20%	0.05	23%	0.06	22%
VIAL-B	0.22	102%	0.13	99%	0.16	100%

(b) Inferring attribute categories

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
Random	0.49	306%	0.33	309%	0.39	308%
RWwR-SAN	0.09	14%	0.06	14%	0.07	15%
VIAL-B	0.21	48%	0.12	48%	0.15	48%

Figure: Performance gains and relative performance gain of VIAL and other inference methods

Weaknesses:

- Used over only one dataset
- Scalability untested. Time complexity of O(dmt) where d is the number of iterations, m the number of links, and t the number of target users.

Future Work:

- Learning link weights using machine learning
- Building techniques to defend against VIAL

Attributed Social Network Embedding

IEEE Transactions on Knowledge and Data Engineering, May 2017

Lizi Lao, Xianghan He, Et al. [5]

Introduction

This paper talks about the "homophily" effect, that is how birds of a similar kind flock together. This can be seen through the attribute information of social actors. The Attributed Social Media Networking Embedding (ASNE) framework will represent social actors by preserving both "structural proximity" and "attribute proximity". The structural proximity deals with representing the global network structure and the attribute proximity deals with representing the homophily effect. Through ASNE we can have better link prediction between social actors and better node classification where we can approximate with more precision the cluster or group where particular social actors belong to. By social networks we mean Facebook, Twitter, citation networks for academic papers, telephone caller-callee networks, etc. Machine Learning is used and people are assigned to particular clusters so that these clusters can be subject to targeted advertisement and recommendations. Besides link structures, there are lots of information about social actors and we refer to them as "attributes" which not only contain demographics like age, gender, etc but also affiliated texts and possible labels. ASNE works with real-valued feature vectors. Multi-layered neural network is used in this paper for good representation and generalization.

The social network is represented as a graph, G, where the nodes, U_i, are the social actors, the edges, $E_{i,j}$ mean the links between the social actors and nodes A_i mean the attributes of social actors. The edges can be given weights denoting the strength of the links. But for now we are using an unweighted graph.

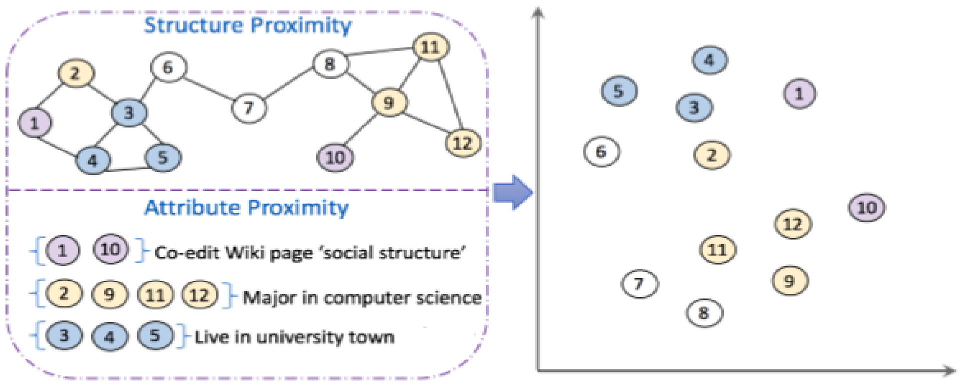

Structure Proximity
Attribute Proximity
Co-edit Wiki page 'social structure'
Major in computer science
Live in university town

Figure: An illustration of social network embedding

The aim of social network embedding is to project the social actors into a low-dimensional vector space (a.k.a. embedding space). Based on the link structure, a common assumption of network embedding methods is that closely connected users should be close to each other in the embedding space. For example, (U_1, U_2, U_3, U_4, U_5) should be close to each other, and similarly for (U_8, U_9, U_{11}, U_{12}). However, we argue that purely capturing structural information is far from enough. Taking the attribute homophily effect into consideration, (U_2, U_9, U_{11}, U_{12}) should also be close to each other since they have common attributes and so there is a potential connection. Structural Proximity denotes the proximity of nodes that is evidenced by links .For U_i and U_j ,if there exists a link $E_{i,j}$ between them, it indicates the direct proximity; on the other hand, if U_j is within the context of U_i, it indicates the indirect proximity. Attribute Proximity denotes the proximity of nodes that is evidenced by attributes. The attribute intersection of A_i and A_j indicates the attribute proximity of U_i and U_j.

Procedure:

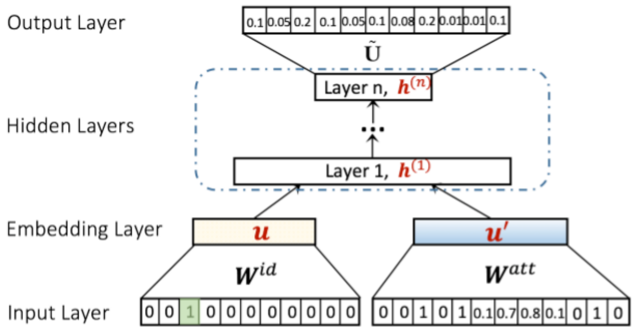


Figure: Attributed social network embedding (ASNE)

Here, a neural network is used and both the tuples containing information of structural proximity and attribute proximity are merged in an early fusion and then applied to the neural network and the output is transformed into a probability vector ‘o’, which contains the predictive link probability of U_i to all the nodes in U which we will use to cluster the users.

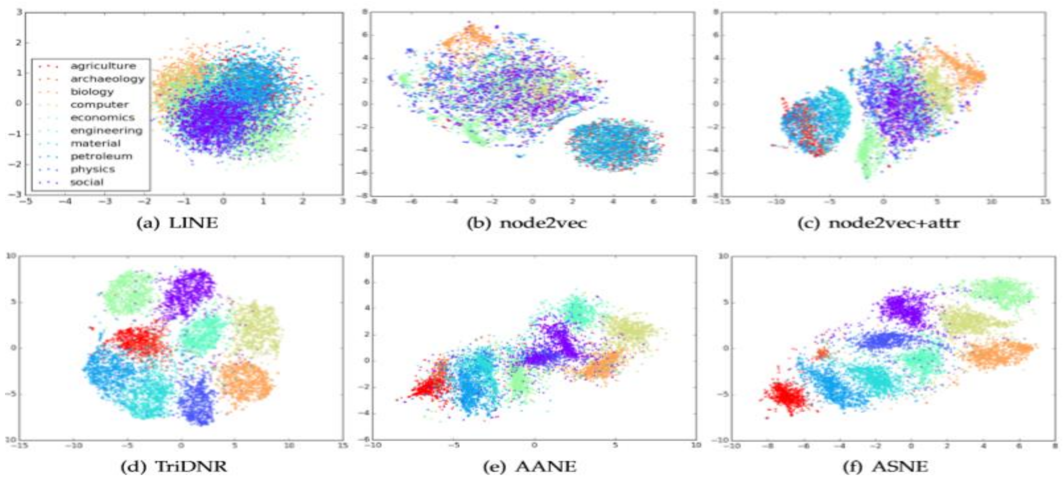


Figure: Results of clustering using different frameworks

WEAKNESS:

- To account for a node's *structural proximity* with respect to all its neighbors, we define the conditional probability of a node set by assuming conditional independence. By maximizing this conditional probability over all nodes, we can achieve the goal of preserving the global *structural proximity.* But, for a very large number of users preserving this is very difficult as the conditional probability becomes lesser and lesser with the increased number. So, clustering becomes difficult.

- When we are trying to fill up the tuple preserving *attribute proximity,* we come across several continuous values that are obtained from bag-of-words extraction from documents, raw image values, however, selecting the correct values to represent the attributes is a difficult task and this may lead to erroneous output tuple that would suggest wrong clustering.

Future Work:

- ASNE framework will be enhanced by fusing data from multiple modalities. It is reported that over 45% tweets contain images in Weibo.
- New ways will be explored to know how to capture the evolutionary nature of social networks, such as new users and social relations by using temporal-aware recurrent neural networks.
- We will consider improving the efficiency of ASNE by learning to hash techniques to make it suitable for large-scale industrial use.

GMove: Group-Level Mobility Modeling Using Geo-Tagged Social Media

Knowledge Discovery and Data Mining 2016, ACM

Chao Zhang, Keyang Zhang et al [4]

Using human mobility information, we can do things like urban planning, traffic scheduling, etc. Geotagged social media can be used to find patterns of mobility of people. We assume that people of a particular group have similar mobility patterns and this would help us predict the movements of many people provided we can identify which groups they belong to. GMove is a group level mobility modeling method which uses GeoSM data. GeoSM data stands for Geo-Social Media data. An example of such data is Facebook Check-in. In this paper they assume that people of a particular group have similar mobility patterns and this would help predict the movements of many people provided that the groups people belong to can be identifiable. GMove alternates between user grouping and mobility modeling and generates an ensemble of Hidden Markov Models to characterize group-level regularity. GeoSM data not only holds information about the places a person visits but also gives information of what a person does in those places as it also includes texts and GeoSM has a big size and coverage and provides lots of meaningful information. Using GeoSM answers to questions like where is the user, what is the user doing and when does the activity happen are answered as the data are spatiotemporal as well.

Data is taken as a tuple containing (*u,t,l,e*) where *u* is the user id, *t* is the time of activity *l* is the location containing latitude and longitude information and *e* is the bag of keywords that will determine the activity the user is associated with. For finding patterns a time threshold like 3 hours is taken, where activities that take place within 3 hours after the initial activity are grouped together and are called trajectory. We do this because long gaps in time would give us very unreliable patterns of transition between activities.

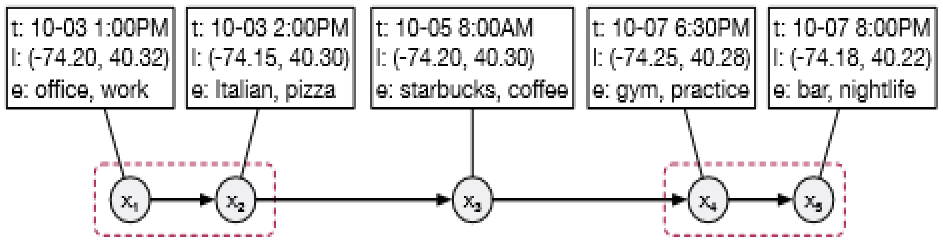


Figure: An illustration of trajectory

Given the set *U* of users and their trajectories extracted from historical GeoSM records, the task of group-level mobility modeling consists of two sub-tasks:

(1) User grouping: Softly group the users in U such that the members in the same group have similar moving behaviors.

(2) Mobility modeling: For each user group, discover the latent states that reflect the users' activities from a multidimensional (where-what-when) view, and find out how the users move between those latent states.

GMove is basically built by using Hidden Markov Model where given an observed sequence S ={$x_1 x_2 \ldots x_n$}, HMM defines K latent states Z = {1,2,...,K} that are not observable. Each record x_i (1 ≤ i ≤ n) corresponds to a latent state z_i ∈ Z, which has a probabilistic distribution that governs the generation of x_i. In addition, the latent state sequence $z_1 z_2 .. z_n$ follows the Markov process, namely, the state z_i only depends on the previous state z_{i-1}.

Grouping the users in such a way that the people in the group have similar movement patterns is quite a challenge. However, the sub-tasks of user grouping and mobility modeling can mutually enhance each other:

(1) Better user grouping leads to more consistent movement data within each group, which can improve the quality of the HMM.

(2) High-quality HMMs more accurately describe the true latent states and the transitions between them, which helps infer which group a user should belong to.

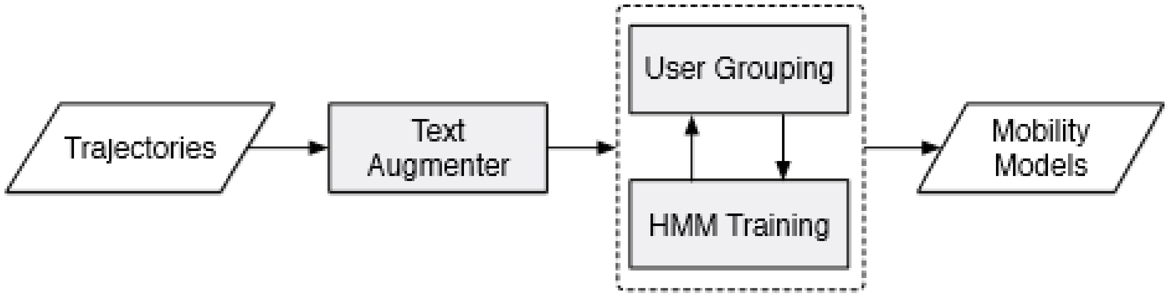
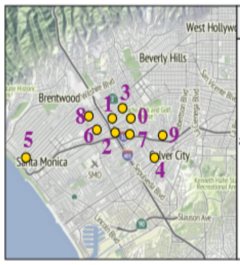
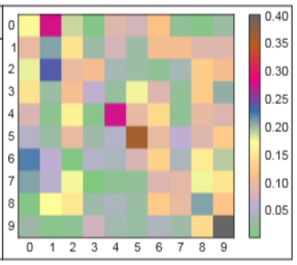


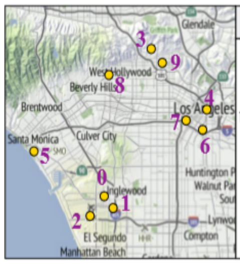
Figure: Framework of GMove



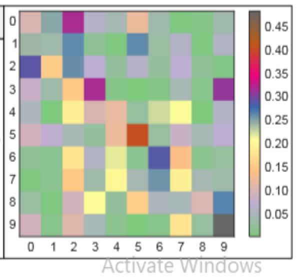
State 0	State 1	State 2	State 3	State 4	State 5	State 6	State 7	State 8	State 9
home	school	food	friend	buy	santa	wait	suck	home	night
morning	ucla	restaurant	hahaha	shop	monica	dinner	gym	rest	music
coffee	course	dinner	dream	mall	beach	food	football	sleep	tonight
dude	midterm	grill	song	beautiful	sunset	worst	team	live	bar
apartment	class	love	child	happy	dinner	grill	game	game	tbh
day	food	deli	birthday	mine	baby	restaurant	play	drink	nightlife
hall	drink	worst	drink	apparel	venice	delicious	buster	bed	evening
run	damn	come	kiss	bag	shore	amazing	let	show	movie
house	friend	waiter	bday	clothes	life	gather	beat	time	beer
sleep	stay	chipotle	snapchat	day	please	love	boy	miss	like



(a) The mobility model for the first user group (students).



State 0	State 1	State 2	State 3	State 4	State 5	State 6	State 7	State 8	State 9
hotel	restaurant	lax	universal	food	venice	shopping	night	food	club
sleep	food	travel	studio	restaurant	beach	buy	hotel	table	night
miss	amazing	airport	night	chinese	people	shop	live	taste	music
building	eat	international	party	mexican	life	clothes	lounge	salad	girl
center	lunch	come	hollywood	spicy	spa	cheap	place	quiet	nightclub
plaza	grill	flylax	buster	hometown	sunshine	sales	taxi	deli	money
flight	american	arrival	friend	taste	beautiful	nice	shower	hot	drink
home	deli	terminal	happy	china	santa	women	sleep	steak	sleep
city	together	train	haloween	town	monica	mall	queen	dining	dance
inn	pancake	depart	club	delicious	food	men	terminal	fame	rock

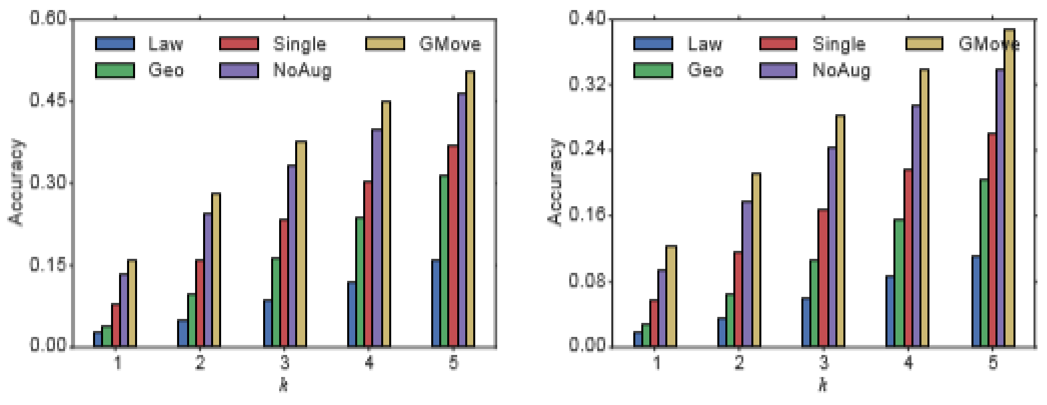


(b) The mobility model for the second user group (tourists).

Figure: Results of G-Move

As we can see, G-Move provides us matrices that give us the probability of moving across from one state to the next. This allows us to make predictions about where a user in a particular group will go next.

Prediction accuracy of GMove with other mobility models:



(a) LA (b) NY

Figure: Prediction accuracy vs k

Here we can observe GMove outperforming competing algorithms consistently across all values of K and over both the cities of Los Angeles and New York City. The accuracy ranged from 15% to 50% in Los Angeles and 12% to 38% in New York.

Weakness:

- GMove works well in such groups where the members of the group have some pattern in mobility, e.g. students in a university. However, there may be such groups like taxi drivers where mobility is pretty random and we cannot find good patterns in their movements.

Future Work:

- Adapting GMove such that it can automatically infer the reliability of the transitions and extract trajectories from the raw data.
- Besides context-aware location prediction, they plan to apply GMove for other tasks in the urban computing context.

A New Framework for Privacy-Preserving Aggregation of Time-Series Data

ACM Transactions on Privacy and Security, January 2018

Fabrice Benhamouda et al. [3]

Aggregator-oblivious encryption allows an untrusted aggregator to periodically compute an aggregate value over encrypted data contributed by a set of users. Such encryption schemes find numerous applications, particularly in the context of privacy-preserving smart metering. The goal of this article is to mitigate the privacy issues that arise from smart metering by computing aggregates rather than individual energy consumption. This is because, smart meters may send too much information to external people who may misuse information that reveal habits of users like when does a user watch TV. The work is based on smart meter encrypting its actual energy consumption using personal keys and sending the result at regular intervals to an aggregator (which can be an entity different from the energy provider). Upon receiving the encrypted values from a predetermined set of users, the aggregator combines the received values and from there deduces the total energy consumption over the population of these users for the current time period. This operation involves a secret key known only to the aggregator. Further, computing the sum over the predetermined set of users is the only operation the aggregator can perform and so it knows nothing else. This type of process is known as aggregator-oblivious encryption scheme.

This paper proposes a generic framework for the privacy-preserving aggregation of time-series data featuring a tighter reduction. This framework is based on smooth projective hash functions (SPHFs) with an extra additively homomorphic property over the key space.

Definition of the aggregator-oblivious encryption-

An aggregator-oblivious encryption scheme is a tuple of three algorithms, (Setup,Enc,AggrDec), defined as follows:

Setup(λ) : On input a security parameter λ, a trusted dealer generates the system parameters param, the aggregator's private key sk_0, and the private key sk_i for each user i (1≤i ≤n).

Enc (param, sk_i,τ, $x_{i,\tau}$): At time period τ, user i encrypts a value $x_{i,\tau}$ using her private encryption key sk_i to get $c_{i,\tau}$ = Enc(param, sk_i,τ, $x_{i,\tau}$).

AggrDec(param, sk_0,τ, $c_{i,\tau}$,..., $c_{n,\tau}$): At time period τ, the aggregator using sk_0 obtains x_τ = $\sum_{i=1}^{n} x_{i,\tau}$ mod M as the evaluation of x_τ=AggrDec(param, sk_0,τ, $c_{i,\tau}$,..., $c_{n,\tau}$). M is some fixed integer contained in the system parameters param.

Core Inspirations

Predicting future locations with Hidden Markov Models

Wesley Mathew, Ruben Raposo et al. [1]

ACM Conference on Ubiquitous Computing, 2012

Geo-positioning technologies like the GPS allows us to track and analyze human movement patterns like never before. In this paper, they have used a person's GPS 'footprints' to find particular patterns in his/her movements in order to predict the next location of the aforementioned individual provided we have access to the current sequence of locations visited. In order to predict the next location, an inference model was required and for that they are using "The Hidden Markov Model(HMM)" where the observation is the sequence of locations visited and the goal is to generate a sequence which would give the predicted future location in the last index of the sequence. In order to train the HMM efficiently, they have separated the location data from Geo-Life Project dataset and created sequences in virtue of weekdays day-time (7 AM - 7 PM), weekdays night-time (7 PM – 7 AM), weekends day-time and weekends night-time. However, the locations were in the form of {latitude, longitude, altitude} which were later represented as members of a triangular mesh. So, further discretized representation was achieved as the previous continuous dataset has now been reduced to a triangular mesh structure and thus the observable sequence becomes a stream of mesh IDs on which the location points laid.

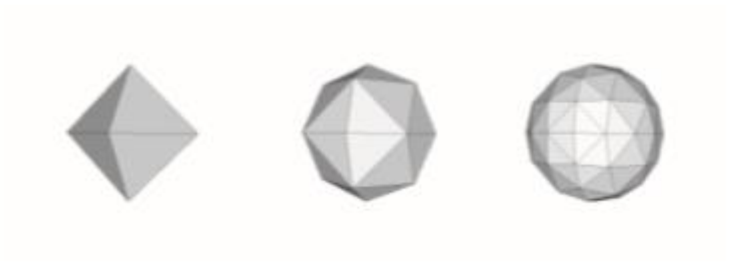


Figure 1. Decomposition of the Earth's surface for triangular meshes with resolutions of zero, one, and two.

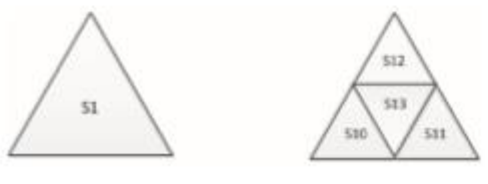


Figure 2. Decomposition of circular triangles.

Basically, in the HMM there are two random variables x and y where x(t) is the hidden state at time t, and y(t) is the observed location at time t which is now nothing but a mesh ID. The basic Markov property is that x(t) only depends on x(t-1) and y(t) only depends on x(t). In this paper, the Baum-Welch algorithm was implemented on an HMM described by $\boldsymbol{\lambda} = \{\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}\}$ where A represents the transition matrix, B being the emission matrix and **π** being the initial probability distribution. Given an observation sequence corresponding to Y = <y(1), y(2), y(3),… y(n)> the Baum-Welch algorithm computes **$\lambda^* = max_\lambda$ P(Y| λ),** which maximizes the probability of the observed sequence Y. So, in the end we are left with another sequence Y = <y(1), y(2), y(3),… y(L),y(Lnext)> where the first L positions correspond to places visited already in sequential format and the Lnext is the predicted possible location to be visited next.

Strengths:

- The emphasis on the discretization of the locations for training the HMM led to very satisfactory results
- The approach of the paper on preserving the human location characteristics by clustering location histories serves as an inspiration.

Weaknesses:

- During implementation, they removed all duplicate consecutive locations from the trajectory in the dataset and this hurts in extraction of features like for how long an individual has stayed in a particular location.
- Although they achieved a remarkable 13% accuracy for predicting the mesh id (among numerous meshes representing quite small areas) on which an individual can be found next, however, the computational time is quite expensive.

Future Work:

- Improving HMM through posterior regularization where facts like consecutive locations that are located closely should be more probable for visits than others.
- Experimenting with discriminative models like sliding window classifiers, structured Support Vector Machines (SVMs), Max Margin Markov Model (M3Ns) and Search-based Structured Prediction (SEARN) for addressing related sequence analysis methods, such as the classification of human location histories according to semantic categories.

Detecting Meaningful Places and Predicting Locations Using Varied K-Means and Hidden Markov Model

Ramez Elmasri, Neelabh Pant et al. [2]

17th SIAM International Conference on Data Mining, April 2017

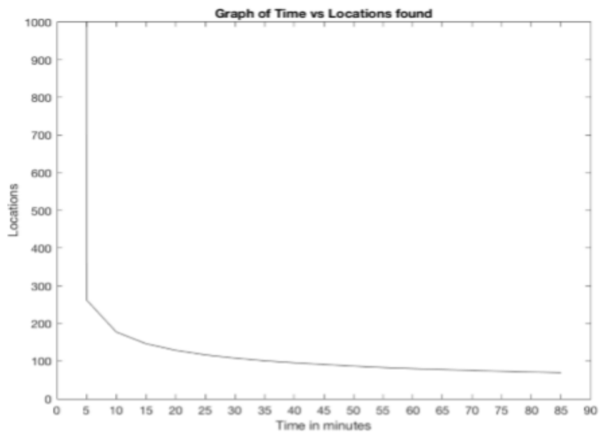


Figure : Graph to identify meaningful locations.

In this paper, they have used a modified version of the K-Means algorithm to cluster the dataset. The traditional algorithm takes k number of centroids to form k clusters by comparing distances and the closest points to the centroid form the clusters and mean location point is calculated. The mean point is later used as the centroid for the next clustering iteration. The process is repeated until we get a fixed mean point and the cluster is created involving all the points encompassed and these points are removed from consideration. The process repeats until no centroids remain. Their approach was to find meaningful locations that the user went to and by that they mean locations where the individual has remained for at least a "threshold, T" amount of time. The threshold was not randomly decided rather a graph was plotted of location vs time spent on those locations where the threshold of 10 minutes was decided as it was the turning point of the graph. During the clustering phase, a good radius had to be selected such that the cluster encompasses just the right number of relevant places. Too big a radius leads to too many irrelevant locations in the cluster, too small leads to very few locations selected in cluster. The best value of the radius was determined using the distances between each significant centroid, δ and it was calculated using the Haversine formula:

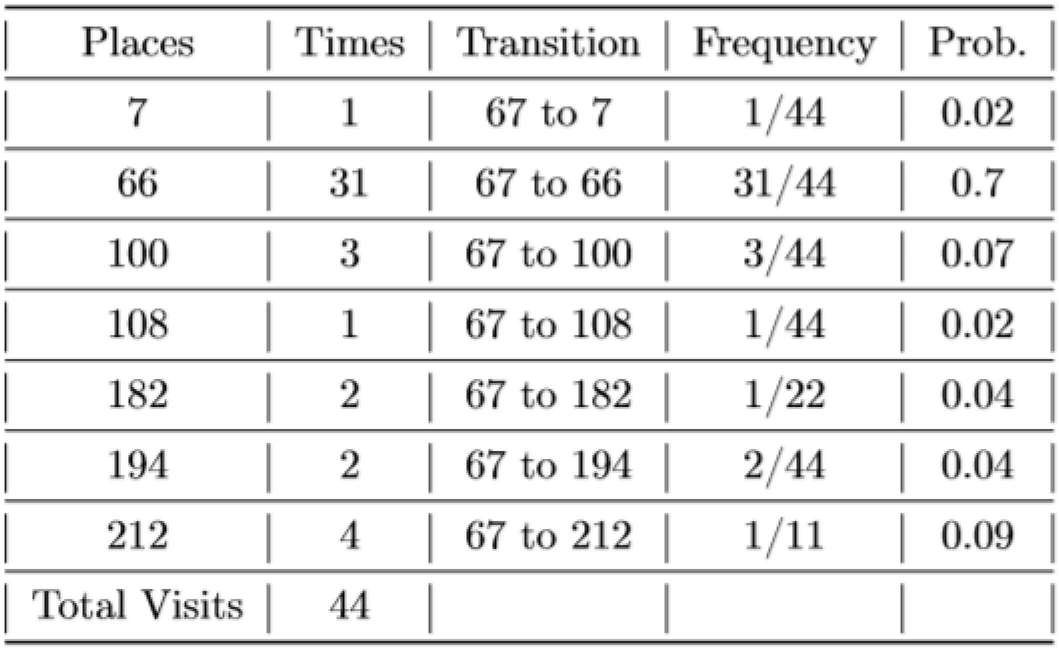
$$hav(\frac{d}{r}) = hav(\varphi_2 - \varphi_1) + cos(\varphi_1)cos(\varphi_2)hav(\lambda_1 - \lambda_2)$$

where *hav* is haversine function which is defined as:

$$hav(\theta) = sin^2(\frac{\theta}{2}) = (\frac{1 - cos\theta}{2}),$$

d = distance between two points,
r = radius of sphere,
φ_1, φ_2 = latitude of point 1 and point 2,
λ_1, λ_2 = longitude of point 1 and point 2, and
$\frac{d}{r}$ = central angle in radians.

They calculated δ for all sites and the minimum δ was used as the radius of the clusters. Later on, a Markov Model was created for each location containing transitions between the clusters indicating mobility. A naïve Bayes approach was used for generating the following transition table:



Places	Times	Transition	Frequency	Prob.
7	1	67 to 7	1/44	0.02
66	31	67 to 66	31/44	0.7
100	3	67 to 100	3/44	0.07
108	1	67 to 108	1/44	0.02
182	2	67 to 182	1/22	0.04
194	2	67 to 194	2/44	0.04
212	4	67 to 212	1/11	0.09
Total Visits	44			

Figure: Table showing the transition probabilities in the Markov Chain

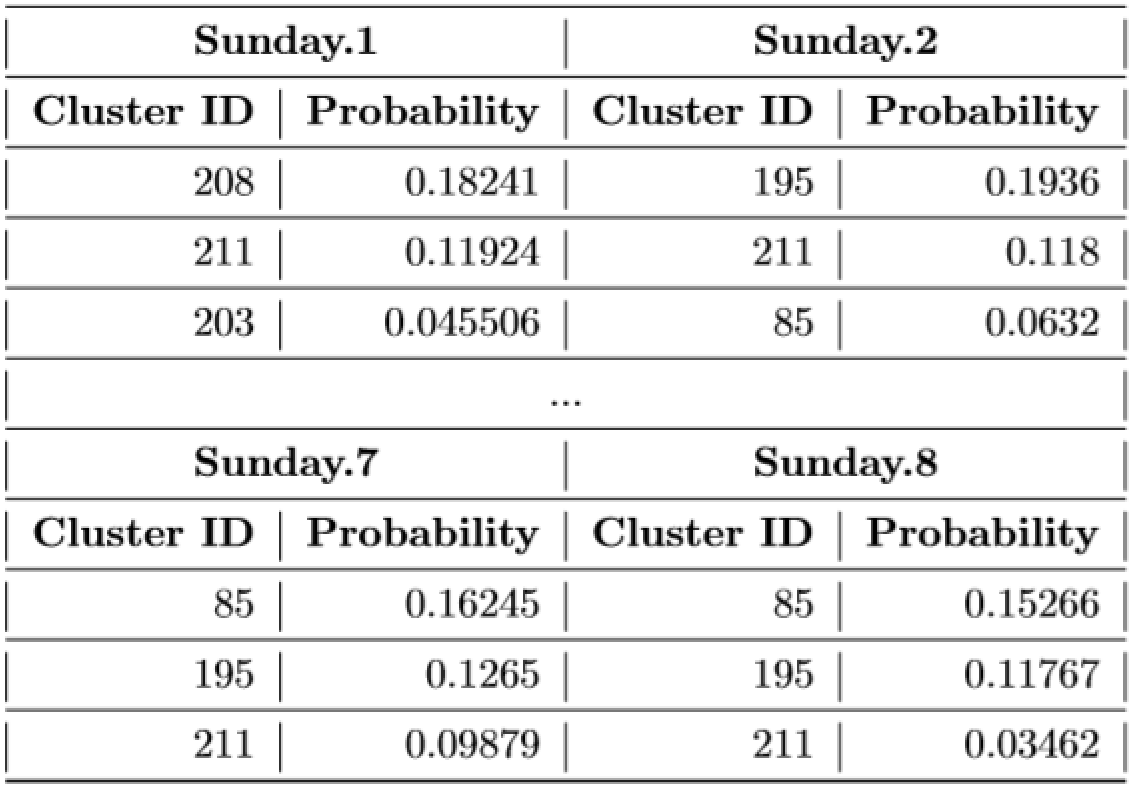
For getting the temporal behavior of the data they calculated the following probabilities like:

$$\mathbf{P(x|Sunday)} = \frac{\mathbf{P(Sunday|x) * P(x)}}{\mathbf{P(Sunday)}},$$

$$\mathbf{x} = \text{ClusterID}.$$

The query result is delivered by finding the x with the maximum probability. **P(Sunday|x)** , can be calculated by finding out the total number of visits to cluster **x** on Sunday divided by the total number of visits to cluster **x**. **P(x)** is the ratio of total points in cluster **x** over the total number of points in all the clusters. Finally, if **X** = set of all clusters, then: **P(Sunday)=[P(Sunday|x)*P(x)]+[P(Sunday|y) * P(y)] + ...+ [P(Sunday|n) * P(n)], where (x, y, ...,**

n) ∈ X. Furthermore, queries like **P(x|Sunday.1)** are calculated in a similar fashion where Sunday.1 represents the first three-hour period for Sunday i.e. 12 AM-3 AM. So, basically the day has been divided into 8 periods which are of 3 hours each.



Sunday.1		Sunday.2	
Cluster ID	Probability	Cluster ID	Probability
208	0.18241	195	0.1936
211	0.11924	211	0.118
203	0.045506	85	0.0632
...			
Sunday.7		**Sunday.8**	
Cluster ID	**Probability**	**Cluster ID**	**Probability**
85	0.16245	85	0.15266
195	0.1265	195	0.11767
211	0.09879	211	0.03462

Figure: Table showing the areas with highest probabilities for a particular day and time period

Strengths:

- The use of K-means clustering to obtain clusters containing meaningful places inspired us to use it for discretization of location in our HMM.

Weaknesses:

- The naïve Bayes approach is quite inefficient and it does not find or predict patterns of sequences containing next locations in a temporal period given the current mobility sequence.

Introduction to Markov Models and HMM

In order to work with sequential data like time-series data, we require a model and one of the most useful of models is the Hidden Markov Model (HMM). We use the HMM whenever we cannot use the Markov Model directly because the underlying data or the actual states are "hidden" or cannot be observed directly as in other cases of Markov Model. We only have information of the observed data but no idea on the information of the states. Now, in order to understand HMM fully, we need understanding of a Markov Model.

To understand a Markov Model, let us have an example with dice and balls. What we require is:

- A six-sided red dice
- A ten-sided black dice
- A red bag containing 9 red balls and 1 black ball
- A black bag containing 19 black balls and 1 red ball

Procedure of the Experiment

Step 1: Emission: Roll a dice of a color chosen randomly and record the number produced.

Step 2: Transition: The color of the dice rolled determines from which bag we pick up the ball from. The color of the ball determines whether we shift from the current state or not. The states being "red" and "black".

Let us consider the initial probability of selecting the red dice or the black dice be 0.5. The red dice was selected and rolled. In the next iteration, the red bag was selected and a black ball came up. Now, we must select the black dice and roll it. So, basically there was a transition from the red state to the black state. The following diagram explains the Markov Model:

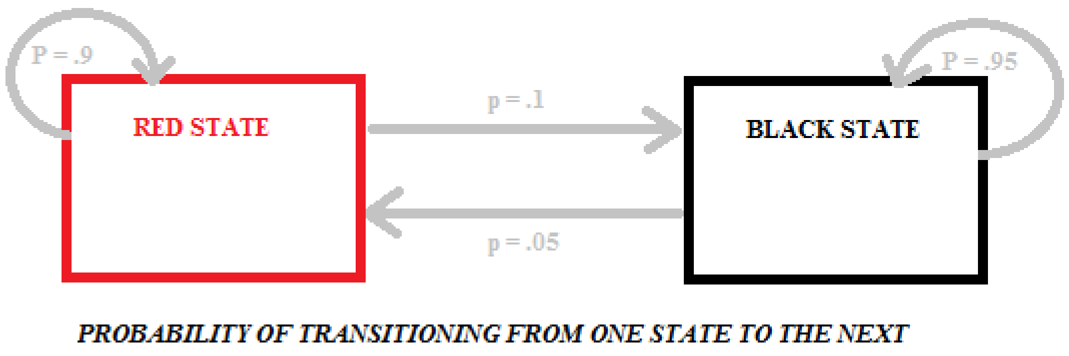


Figure: Markov Model Transitions [10]

A possible sequence of length 10 could be {2,3,6,1,1,8,7,3,9,6}. The process of transitioning from one state to the next is called the Markov process.

An HMM is expressed by $\lambda = (A,B,\pi)$. Other notation is used in Hidden Markov Models:

- A = transition matrix containing the "hidden"state transition probabilities (a_{ij})
- B = observation probability matrix ($b_j(k)$) also known as emission matrix
- N = number of "hidden" states in the model {1,2…N} or the state at time t $\rightarrow s_t$
- M = number of distinct observation symbols per state
- $Q = \{q_0, q_1, . . . , q_{n-1}\}$ = distinct states of the Markov process
- T = length of the observation sequence
- $V = \{0, 1, . . . , M - 1\}$ = set of possible observations
- $O = (O_0, O_1, . . . , O_{T-1})$ = observation sequence
- π = initial state distribution (π_i)
- s = state or state sequence ($s_1, s_2… s_n$)
- x_k = hidden state
- y_k = observation state

There are 3 basic types of problems solved by HMM:

(1) **The Evaluation Problem**
Given an HMM λ and a sequence of observations $O = o_1, o_2, … , o_T$ what is the probability that the observations are generated by the model, $P\{O|\lambda\}$? This problem can be solved in a naïve way or by using the **Forward** or **Backward** algorithm which are computationally much faster.

(2) **The Decoding Problem**
Given a model λ and a sequence of observations $O = o_1, o_2, … , o_T$ what is the most likely "hidden" state sequence in the model that produced the observations? This problem is solved using the **Viterbi** algorithm.

(3) **The Learning Problem**
Given a model λ and a sequence of observations $O = o_1, o_2, … , o_T$, how should we adjust the model parameters $\{A,B,\pi\}$ in order to maximize $P\{O|\lambda\}$. This problem is solved using the **Baum-Welch** algorithm.

Evaluation problem Solution

Naïve approach:

- We calculate the following quantity which is:

$$P(O \mid q, \lambda) = \prod_{i=1}^{T} P(o_t \mid q_t, \lambda) = b_{q1}(o_1) b_{q2}(o_2) \ldots b_{qT}(o_T)$$

- We calculate:

$$P(q \mid \lambda) = \pi_{q1} a_{q1q2} a_{q2q3} \ldots a_{qT-1qT}$$

- And finally we use them to get:

$$P(O \mid \lambda) = \sum_{q} P(O \mid q, \lambda) P(q \mid \lambda)$$

However, the entire process is quite computationally expensive with a time complexity of O(TN^T) where there are N^T "hidden" state paths each costing O(T) calculations. However, we can use much efficient algorithms like **Forward** and **Backward** algorithms which use dynamic programming.

Forward algorithm:

- Finding auxiliary forward variable α:

$$\alpha_t(i) = P(o_1, \ldots, o_t \mid q_t = i, \lambda)$$

- Initialize:

$$\alpha_1(i) = \pi_i b_i(o_1)$$

- Calculate:

$$\alpha_{t+1}(j) = [\sum \alpha_t(i) a_{ij}] b_j(o_{t+1})$$

- Obtain:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The complexity drops to O(N^2T) from O(TN^T). $\alpha_t(i)$ is the probability of observing a partial sequence of observables $o_1, \ldots o_t$ such that at time t, state $q_t = i$

Backward algorithm (Alternative algorithm) :

- Finding auxiliary forward variable ***β*:**

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T \mid q_t = i, \lambda)$$

- Initialize:

$$\beta_T(j) = 1$$

- Calculate:

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1})$$

- Obtain:

$$p(O \mid \lambda) = \sum_{i=1}^{N} \beta_1(i) \qquad t = T-1, \dots, 1$$

The complexity drops to O(N^2T) from O(TN^T). $\beta_t(i)$ – the probability of observing a sequence of observables $o_{t+1},\dots,o_T$ given state $q_t = i$ at time *t,* and λ.

Decoding problem Solution

Viterbi algorithm:

- It is an inductive algorithm that keeps the best state sequence at each instance. State sequence to maximise P(O,Q|λ):

$$P(q_1, q_2, \dots q_T \mid O, \lambda)$$

- Define auxiliary variable δ:

$$\delta_t(i) = \max_q P(q_1, q_2, \dots, q_t = i, o_1, o_2, \dots o_t \mid \lambda)$$

- Recurrent property:

$$\delta_{t+1}(j) = \max(\delta_t(i) a_{ij}) b_j(o_{t+1})$$

- Initialize:

$$\delta_1(i) = \pi_i b_i(o_1) \qquad 1 \le i \le N$$
$$\psi_1(i) = 0$$

- Recursion:

$$\delta_t(j) = \max_{1\le i\le N}(\delta_{t-1}(i)a_{ij})b_j(o_t)$$

$$\psi_t(j) = \arg\max_{1\le i\le N}(\delta_{t-1}(i)a_{ij}) \qquad 2 \le t \le T, 1 \le j \le N$$

- Termination:

$$P^* = \max_{1\le i\le N} \delta_T(i)$$
$$q_T^* = \arg\max_{1\le i\le N} \delta_T(i)$$

- Backtrack state sequence:

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t + T - 1, T - 2, \ldots, 1$$

Learning Problem Solution

Baum-Welch algorithm:

- The objective of the algorithm is to train using sequence of observations that occurred so that the HMM can identify similar sequences in future should they occur.
- An HMM, λ=(A,B,π) is generated by maximising P(O| λ).
- General algorithm:
    1) Initialising λ_0.
    2) Compute new refined model λ using previous model, λ_0 and observed sequence O.
    3) The model is updated λ_0 **<-** λ.
    4) Repeat until:

$$\log P(O \mid \lambda) - \log P(O \mid \lambda_0) < d$$

- Let *ξ*(i,j) be a probability of being in state i at time t and at state j at time t+1*,* given λ and O sequence:

$$\xi(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{P(O \mid \lambda)} \qquad = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}$$

- Let $\gamma_t(i)$ be a probability of being in state *i* at time *t*, given *O* :

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)$$

- We calculate the expected number of transitions:

$$\sum_{t=1}^{T-1} \gamma_t(i)$$

- We calculate the expected number of transitions i -> j:

$$\sum_{t=1}^{T-1} \xi_t(i)$$

- The expected frequency of state i at time t = 1:

$$\hat{\pi} = \gamma_1(i)$$

- Ratio of expected no. of transitions from state *i* to *j* over expected no. of transitions from state i:

$$\hat{a}_{ij} = \frac{\sum \xi_t(i, j)}{\sum \gamma_t(i)}$$

- Ratio of expected no. of times in state j observing symbol k over expected no. of times in state j:

$$\hat{b}_j(k) = \frac{\sum_{t, o_t = k} \gamma_t(j)}{\sum \gamma_t(j)}$$

- Baum-Welch algorithm uses the forward and backward algorithms to calculate the auxiliary variables α,β. The Baum-Welch algorithm is a special case of the Estimation Maximisation (EM) Algorithm.
  - E-step calculate ξ and γ.
  - M-step calculates $\bar{\pi}$, $a_{i,j}$ and $b_j(k)$.
  - The practical issues with Baum Welch include getting stuck in local maxima and numerical problems like log and scaling.

Explaining HMM using a simple example

Suppose we have a robot dog who would try to guess your emotion (only happy or sad) after a sequence of observed events. The observed events could be you “watching a tv series(w)”, “sleeping(sl)”, “crying(c)”, “browsing social media(f)” for simplicity. Now, in this case the robot can only observe your activities, however your emotions are completely hidden from it. So, it has to make an educated guess of your current emotion based on the activities done.

X = {Sad(s), Happy(h)}

Y = {w,sl,c,f}

Step 1: Set up the prior and likelihood models (Initial distribution and Emission matrix respectively):

Probability	S	H
P(X)	0.2	0.8

Probability	Y = w	Y = sl	Y = c	Y = f
P(Y|X=s)	0.1	0.3	0.5	0.1
P(Y|X=h)	0.4	0.4	0.2	0

Goal is to find P(X|Y).

P(X=h|Y=w) = ?

Using Bayes’ Rule:

$$P(X{=}h|Y{=}w) = \frac{P(Y{=}w|X{=}h)*P(X{=}h)}{P(Y{=}w|X{=}h)*P(X{=}h)+P(Y{=}w|X{=}s)*P(X{=}s)} = \frac{0.4*0.8}{0.4*0.8+0.1*0.2} = 0.94$$

Step 2: Set up the transition prior model:

	X2 = sad	X2 = happy
X1 = sad	0.99	0.01
X1 = happy	0.1	0.9

Here, X2 is the current mood and X1 is the previous mood. So, the transition matrix will give us the probability of what is to happen after a certain number of steps (for e.g. after every hour).

P(X2=s|X1=s) = 0.99.

We also set up an initial prior which states the probability of how often a person is sad or happy at the beginning. In this case,

	X = sad	X = happy
$P(X_0)$	0.2	0.8

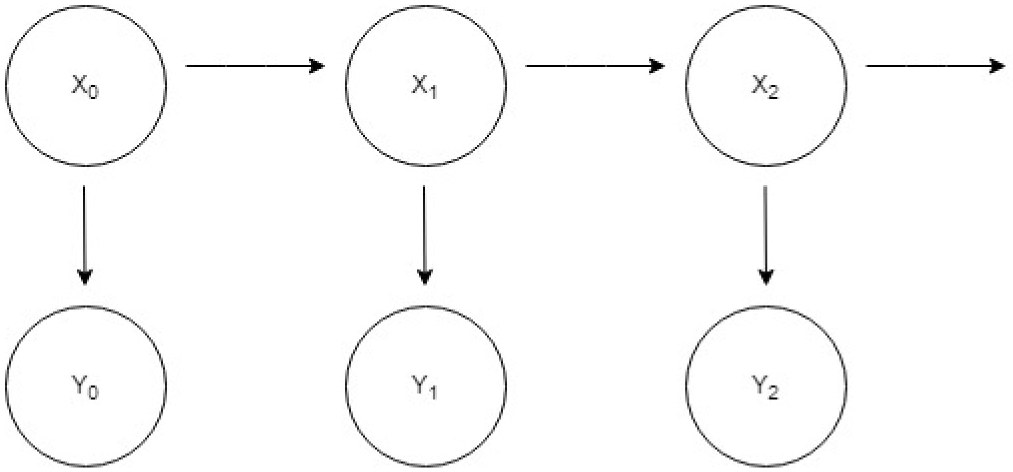


Figure: Hidden state X and visible state Y

Here, X_t are the emotions felt at time t and this is the hidden variable that needs to be inferred whereas Y_t is the activity done at time t and this is the observed variable.

Step 3: Breaking down the HMM algorithm in 2 sections: Prediction and Bayesian Update.

Now our goal is to find P(X3 = h | Y1 = w, Y2 = f, Y3 = c) = ?

Prediction Step:

P(X(t)|Y(1:t-1)) = $\sum_{X(t-1)}(\mathrm{P}(\mathrm{X}(\mathrm{t})|\mathrm{X}(\mathrm{t}-1)) * \mathrm{P}(\mathrm{X}(\mathrm{t}-1)|\mathrm{Y}(1{:}\,\mathrm{t}-1))$

Bayesian Update Step:

P(X(t)|Y(1:t)) $= \frac{\mathrm{P}\left(\mathrm{Y}(\mathrm{t})\middle|\mathrm{X}(\mathrm{t})\right)*\mathrm{P}(\mathrm{X}(\mathrm{t})|\mathrm{Y}(1{:}\mathrm{t}-1))}{\sum_{X(t)}\mathrm{P}(\mathrm{Y}(\mathrm{t})|\mathrm{X}(\mathrm{t}))*\mathrm{P}(\mathrm{X}(\mathrm{t})|\mathrm{Y}(1{:}\mathrm{t}-1))}$

To solve the problem mentioned above:

P(X(1) = h|Y(0)) = 0.9*0.8+0.01*0.2 = 0.722

P(X(1) = h|Y(1) = w) $= \frac{0.4*0.722}{0.4*0.722+0.1*0.722+0.4*0.278+0.1*0.278} = 0.5776$

P(X(2) = h|Y(1)=w) =0.9*0.5766+0.01*(1-0.5766) = 0.5231

P(X(2) = h|Y(1)=w,Y(2)=f) $= \frac{0*0.5231}{0*0.5231+0.1*0.5231+0*4769+0.1*0.4769} = 0$

P(X(3) = h|Y(1)= w,Y(2)=f) = 0.9*0+0.01*1 = 0.01

P(X(3) = h|Y(1)=w,Y(2)=f,Y(3)=c) $= \frac{0.2*0.01}{0.2*0.01+0.5*0.01+0.2*0.99+0.5*0.99} = 0.0029$

So, the probability that the person is happy after the sequence 'w,f,c' is 0.0029. So, the probability that the person is sad after the sequence is 0.9971. So, it is much more likely that the person is sad at the end of the activities.

HMM makes several assumptions about the data it models:

1. The observations $Y_1, Y_2, ..., Y_t$ come from a known, finite sets V, called the observation space.

2. The hidden states $X_1, X_2, ..., X_t$ come from a known, finite sets Q, called the hidden state space.

3. The HMM assigns a probability to any given sequence $X_1, X_2, ..., X_t$. The chain rule says that $P(X_1, X_2, ..., X_t) = \prod_{t=1}^{T} \mathrm{P}(\mathrm{Xt}|\mathrm{X0}, \mathrm{X1}, \ldots, \mathrm{X}(\mathrm{t}-1)) = \prod_{t=1}^{T} \mathrm{P}(\mathrm{Xt}|\mathrm{X} < \mathrm{t}) = \prod_{t=1}^{T} \mathrm{P}(\mathrm{Xt}|\mathrm{Xt}-1)$

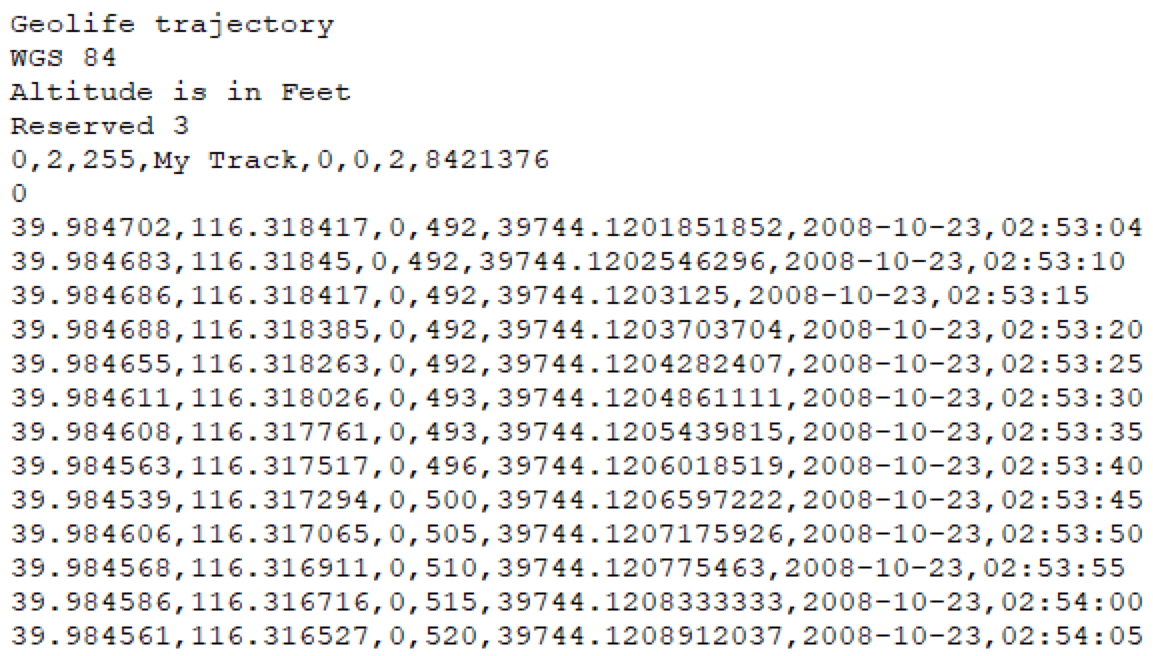
4. Each observation Y_t only depends on the immediate hidden state X_t. Formally $P(Y_t|Y<t\ X<t) = P(Y_t|X_t)$.

Dataset used in implementation

The dataset [8] used is a GPS trajectory dataset and was collected from Microsoft Research Asia. It contains trajectory data of 182 different users for a period of over 5 years (Apr 07-Aug 12). The data was recorded using different GPS loggers and GPS phones, using different sampling rates. The dataset gives us information regarding outdoor movements, consisting of activities like going home, going for work, shopping, hiking, cycling, etc.

The data is kept in PLT files, however we can read it as text. Each PLT file contains a single trajectory and is named by its starting time. The time format is in GMT. The data is formatted in the following way:

- Line 1…6 are useless in this dataset, and can be ignored.
- Field 1: Latitude in decimal degrees.
- Field 2: Longitude in decimal degrees.
- Field 3: All set to 0 for this dataset.
- Field 4: Altitude in feet (-777 if not valid).
- Field 5: Date - number of days (with fractional part) that have passed since 12/30/1899.
- Field 6: Date as a string.
- Field 7: Time as a string.



```
Geolife trajectory
WGS 84
Altitude is in Feet
Reserved 3
0,2,255,My Track,0,0,2,8421376
0
39.984702,116.318417,0,492,39744.1201851852,2008-10-23,02:53:04
39.984683,116.31845,0,492,39744.1202546296,2008-10-23,02:53:10
39.984686,116.318417,0,492,39744.1203125,2008-10-23,02:53:15
39.984688,116.318385,0,492,39744.1203703704,2008-10-23,02:53:20
39.984655,116.318263,0,492,39744.1204282407,2008-10-23,02:53:25
39.984611,116.318026,0,493,39744.1204861111,2008-10-23,02:53:30
39.984608,116.317761,0,493,39744.1205439815,2008-10-23,02:53:35
39.984563,116.317517,0,496,39744.1206018519,2008-10-23,02:53:40
39.984539,116.317294,0,500,39744.1206597222,2008-10-23,02:53:45
39.984606,116.317065,0,505,39744.1207175926,2008-10-23,02:53:50
39.984568,116.316911,0,510,39744.120775463,2008-10-23,02:53:55
39.984586,116.316716,0,515,39744.1208333333,2008-10-23,02:54:00
39.984561,116.316527,0,520,39744.1208912037,2008-10-23,02:54:05
```

Figure: Snapshot of the dataset of a user

The Proposed Method

We were inspired by the work of Mathew and Raposo [1] to use HMM for predicting future locations. However, we involve ourselves with little to no preprocessing because it would require lots of contextual filtering. In order to train the HMM, we initially cluster the trajectory data of a person from the Microsoft GeoLife Project. We cluster the data by taking inspiration from Elmasri and Pant's [2] approach in order to discretize the observed locations. We used K-means and K-medoids clustering, which in our case, produces 40 unique clusters. [We have tested and received valid results between 20 to 200 clusters as well]. So, an individual could only remain in any of these clusters. Hence the observation sequence that is fed to the HMM consists of nothing but a sequence of cluster numbers portraying mobility.

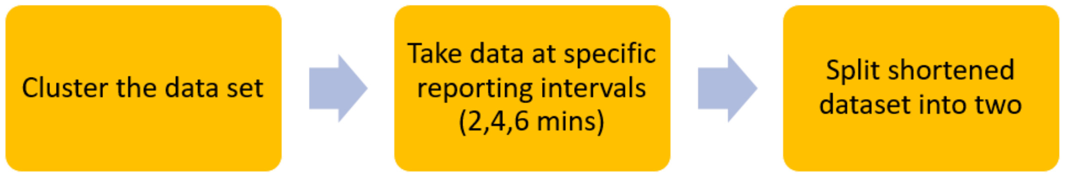

Cluster the data set
Take data at specific reporting intervals (2,4,6 mins)
Split shortened dataset into two

Figure: Depiction of the initial steps before training and evaluation of the HMM.

After clustering, we took data from specific reporting intervals (2,4,6 minutes). For training purpose, we have split the dataset into 2 parts, roughly 50-50. The first portion was used to train the HMM (one HMM for each reporting interval), and the accuracy of prediction was tested using the other half. In order to train the HMM, we have used the Baum-Welch algorithm, a variation of Estimation Maximization (EM), on half the dataset. In the testing phase, we take 100 sequences of same reporting interval. We use the trained HMM for predicting the "l+1"th element, "l" being the observation sequence length. Later on, the observed sequences in the evaluation portion and are compared with the actual "l+1"th element and in the process we find the accuracy percentage. We carry out "n" number of iterations for each of those sequences for getting multiple accuracy rates which we can later average for analysis. So, what we end up with is the **net average accuracy for each reporting frequency** and **the relationship between trends in reporting frequency and accuracy**. We have investigated further on by varying the length of the test sequences for the same reporting frequency and we saw **another relationship involving testing sequence length and accuracy.**

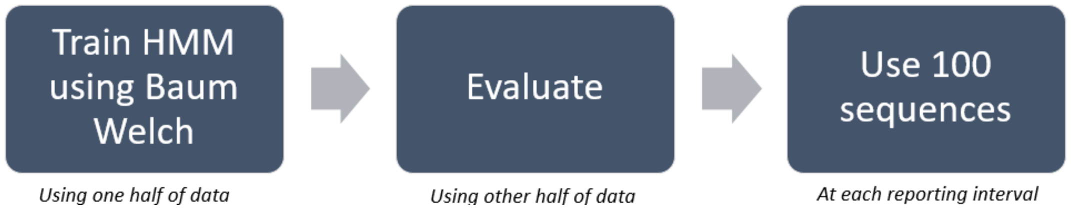


Figure: Depiction of training and evaluation of the HMM in detail.

When using HMMs, comparing location prediction accuracy across different systems is incredibly difficult because of the number of variables.

Variables include:

- Level of location discretization: In our case, this is the number of clusters used. For our testing we kept cluster number constant at 40, an arbitrary number. However, tested from 30 to 200 clusters and the general relationships remained consistent.
- Length of testing sequence: This is the input sequence we give to the trained HMM, whereby the HMM then tells us where the next possible location after the sequence should be. We vary the length of the input sequence to identify local maxima or minima in terms of the predicted accuracy.
- Reporting intervals: This is the “time gap” between the consecutive location points used to train our HMM. We vary this time gap and check the subsequent drops or rises in accuracy. This is the training **information level**.
- Number of abstract hidden states (kept constant at 12, the relationships remain consistent at other arbitrarily chosen hidden state numbers.

Results

As we increase the reporting interval (in minutes) the accuracy drops. The reporting interval was created by skipping intermediate values. Also note the variance in accuracy.

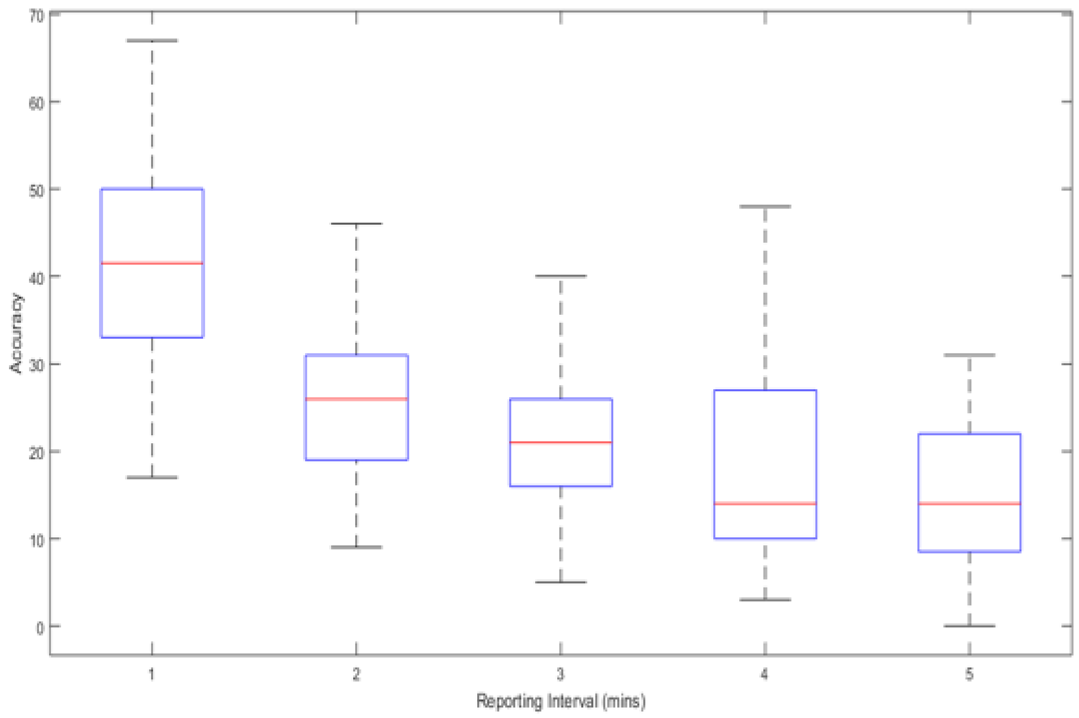


Figure: Accuracy against Reporting Interval (Boxplots)

The box-plots depict the median, first quartile and third quartile for each reporting interval (in minutes) and we can clearly see how the mean, and the quartiles are approximately exponentially decreasing with the increase in reporting interval. With increasing gaps the HMM is less able to decipher the mobility pattern. Listed below are the accuracies with 40 clusters using k-means(left) and k-medoids(right), and a testing sequence length of 10 last locations.

K-Means

Interval(mins)	Accuracy
1	41.15
2	26.13
3	20.93
4	18.48
5	15.09

K-Medoids

Interval(mins)	Accuracy
1	40.97
2	28.59
3	20.47
4	19.03
5	14.48

The exponential decline of accuracy can be more clearly demonstrated in the following diagram where we have done extrapolation (within 10% error) in plotting of accuracy vs reporting intervals using some points. The relationship demonstrated is of a third order polynomial nature.

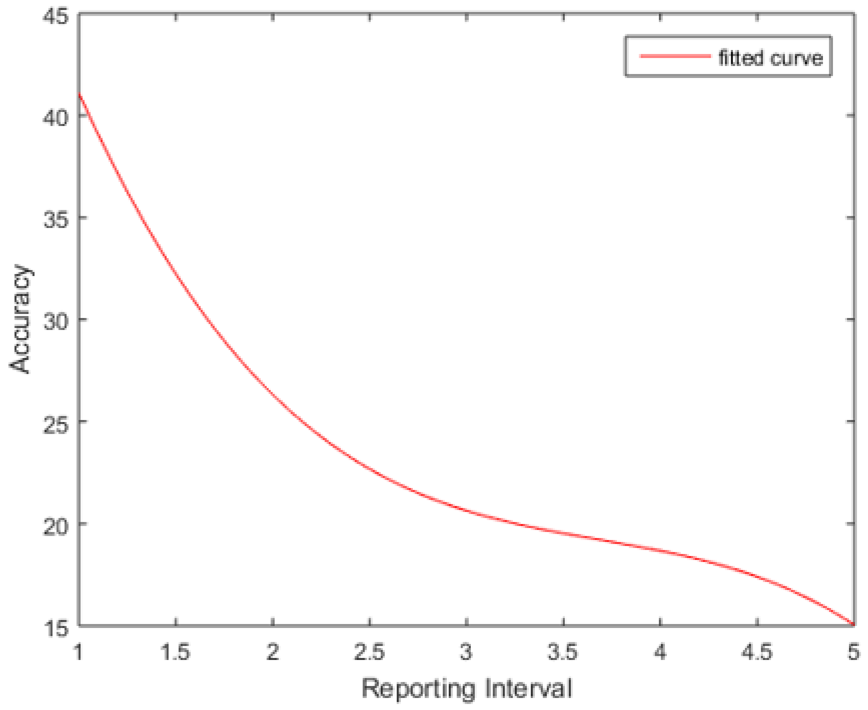


Figure: Relationship between Accuracy and Reporting Interval

Investigating further, we have also come across a relationship between average accuracy and test sequence length. The graph below shows us how a particular test sequence length is responsible for giving us the highest accuracy.

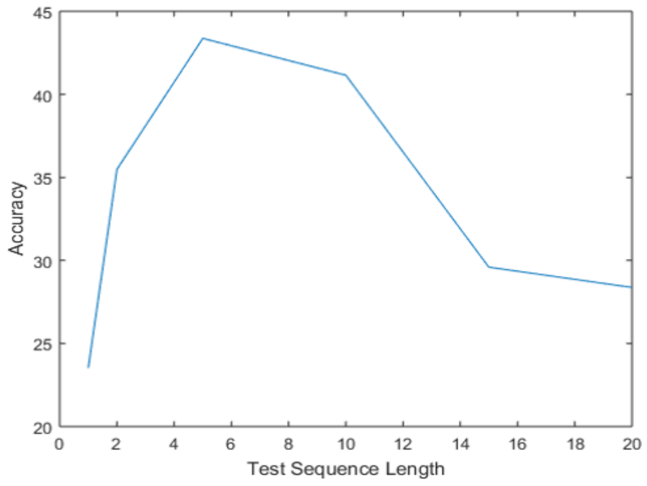


Figure: Demonstration of how test sequence length affects accuracy

We can see clearly that when the sequence length is low, like 2, we get relatively much less accuracy. This is due to the fact that a sequence length of 2 is not enough to capture the mobility patterns of people. However, we see a maximum when dealing with sequence lengths of 5. This could be because when the sequence length is 5, a person's daily mobility pattern is captured, in other words, sequence length of 5 is a good estimate of the "natural" sequence length for a day. Increasing sequence length later on, gives worse accuracy as there is too much information to work with when we are inferencing.

We can also see that testing is consistent across different reporting intervals by varying the test sequence lengths and checking the accuracy for the different lengths as well as the different reporting intervals for each of those test sequences.

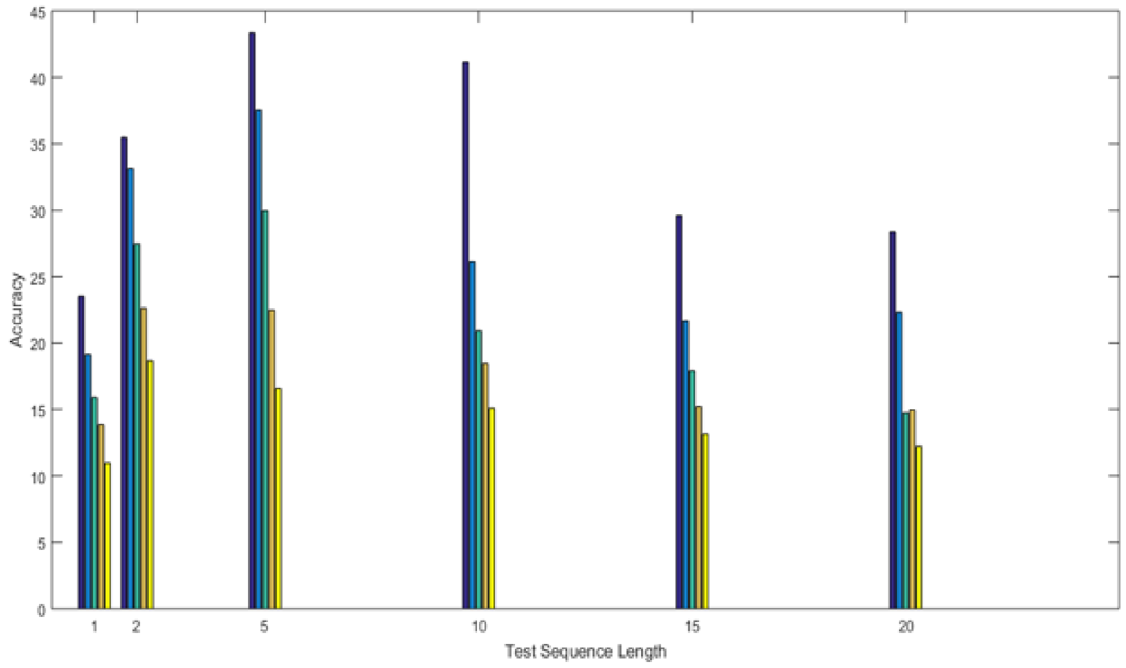


Figure: Demonstration of how the test sequence length and reporting intervals affect accuracy

Comparisons to existing work

Summarizing the contribution of Mathew and Raposo[1], we see how they have divided an immense amount of area into millions of blocks. Furthermore, they got rid of consecutive data that contained the same values, which indicated that individuals were stationary. They have also converted the **time-space** data in the context of **day/night** and **weekday/weekend**. What they strived for was to increase the accuracy across varying number of hidden states.

Mathew and Raposo[1] have achieved the highest accuracy of 13% in some cases and it is remarkable in the sense that they have achieved this accuracy in a precision space of millions of blocks. However, the process is computationally quite expensive and the precision is unnecessarily too high. We can get a much general prediction with a decent accuracy ranging from 16%-40% when using K-means clustering for discretization which was inspired by Elmasri and Pant[2].

However, our approach was not to shoot for high-end specificity like Mathew and Raposo[1], rather we want to obtain a respectable accuracy in the general sense. This is because we feel that if we are able to guess in what area (Uttara, Banani, etc) the person will go to next, and we can do this quite fast computationally, it will be much beneficial than predicting the next block of Uttara where the person will move to next. In short, we do not need “millimeter” range of accuracy in cases of human mobility where we are much happy with “meter” range of accuracy that can be obtained 10 times faster!

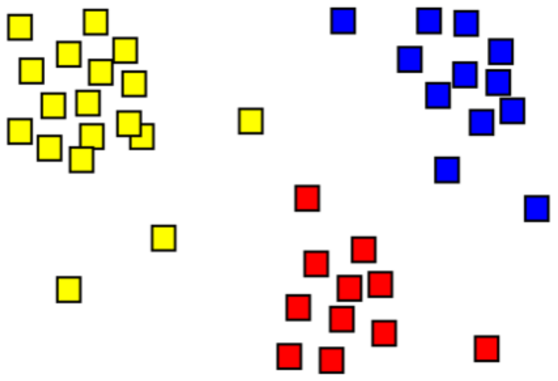


Figure: Clustering specific visiting locations into general clusters [9]

General locations would be preferable in a multitude of scenarios, because exact location consists of a large number of **redundant** information. **Clusters** would give us enough specificity to work with because if you are moving from Dhanmondi to Uttara (one general cluster to another), the mobility pattern consists of enough information to be utilized. Now, if it is possible to work with mobility patterns involving more specificity like moving from one specific block of Dhanmondi to another specific block of Uttara, we would get lots of information in the mobility pattern, however, the computation would be much more time-consuming, and all that information due to extreme specificity would be regarded irrelevant.

Mathew and Raposo [1] have obtained accuracies ranging from 1% to 13% over millions of blocks. When we discretize over 40 clusters, we obtain an accuracy ranging from 16% to 40%. The accuracy is highly dependent on how frequent the people report their geo-location data. Unlike Mathew and Raposo [1], we rejected the idea of dividing **time** and **space**. We reduced time by running a single HMM over all the un-preprocessed data. We wanted to capture the relationship between reporting intervals, testing length and the resulting accuracy.

Furthermore, the picture below is an illustration of how clusters capture the general location of a user. This is a graphical plot of the user's trajectories. Along straight lines, the user is most likely moving. At the vertex between two lines, the user has probably stopped at a location. These are the "meaningful locations". In the picture below the orange "*" marks represent the clusters. Observe how there are always clusters around these vertices between lines, capturing the meaningful locations.

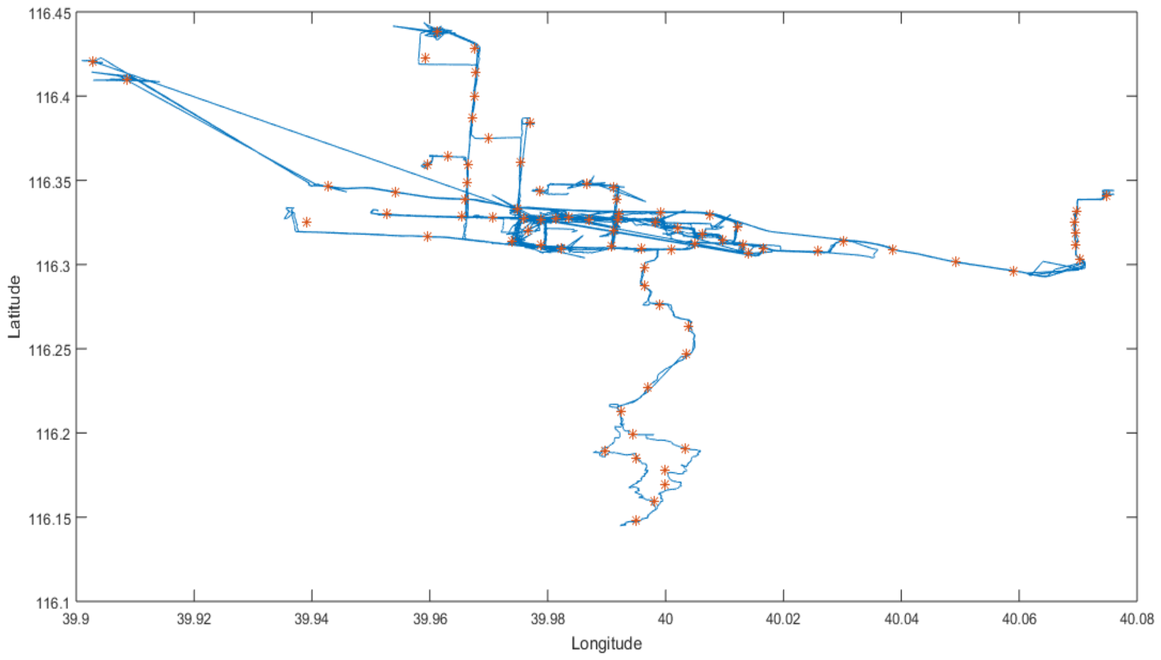


Figure: Graph of Longitude against Latitude, portraying the user's location. The clusters are also represented using the "*" marks

Limitations

So far we have worked with HMM which greatly depends on the discretization of observation variables. So, continuous data must be discretized for HMM to work, however, we would like to utilize the continuous nature of the dataset for which we could work with other methods like **LSTM** which would give us prediction with respect to a continuous domain, for example, in the form of latitude and longitudes. The current procedure, therefore, is limited by the dependence on discretization of the location space.

Also, since we are working using clustering, it is a very difficult task to assign a “good” number of clusters which can be used to model the mobility of an individual as well as providing us with a respectable accuracy of prediction.

Some of the variance in HMM prediction is partly due to the randomness in clustering. This is an unavoidable part of the clustering method.

Future Work

We have shown multiple relationships, accuracy vs reporting intervals, accuracy vs different test sequence lengths. We were able to merely estimate the relationship between these as exponential, third order polynomial or quadratic relationships, without any real mathematical co-relation. However, in future we would like to delve deeper in this matter and figure out and establish an exact mathematical model that satisfies the relationship.

We have gone through some different techniques for location prediction and one of the techniques that has caught our eyes is **LSTM** (Long-Short term memory) network, which are units of a **RNN** (Recurrent Neural Network).

For now, we cannot produce predictions of long sequences with much reliability. So, we want to investigate in producing **reliable long-term prediction** and movement patterns using HMM.

References

[1] Wesley Mathew, Ruben Raposo et al. : “Predicting future locations with Hidden Markov Models”, *ACM Conference on Ubiquitous Computing, 2012*

[2] Neelabh Pant, Ramez Elmasri et al. : “Detecting Meaningful Places and Predicting Locations Using Varied K-Means and Hidden Markov Model”, *17th SIAM International Conference on Data Mining, April 2017*

[3] Fabrice Benhamouda et al. : “A New Framework for Privacy-Preserving Aggregation of Time-Series Data” , *ACM Transactions on Privacy and Security, January 2018*

[4] Chao Zhang, Keyang Zhang et al. : “GMove: Group-Level Mobility Modeling Using Geo-Tagged Social Media” , *Knowledge Discovery and Data Mining 2016, ACM*

[5] Lizi Lao, Xianghan He, Et al. : “Attributed Social Network Embedding” , *IEEE Transactions on Knowledge and Data Engineering, May 2017*

[6] Neil ZG, Bin L.: “Attribute inference attacks on online social networks”, *ACM Transactions on Privacy and Security, January 2018*

[7] “Fitness tracking app Strava gives away location of secret US army bases”, Accessed: 10/09/2018. Available: https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-bases#img-1

[8] “GeoLife GPS Trajectories”. Dataset available at: https://www.microsoft.com/en-us/download/details.aspx?id=52367&from=https%3A%2F%2Fresearch.microsoft.com%2Fen-us%2Fdownloads%2Fb16d359d-d164-469e-9fd4-daa38f2b2e13%2F

[9] “Machine Learning – Clustering”, Accessed: 13/10/2018. Available: https://web.stanford.edu/class/cs102/notes/Clustering.pdf

[10] “Hidden Markov Model: Simple Definition & Overview”, Accessed: 13/10/2018. Available: https://www.statisticshowto.datasciencecentral.com/hidden-markov-model/