**Design and Development of Software Agents for Location**

**Privacy-risk estimation**

Shashank Sharma

August 2018

**Abstract**

The usage of mobile devices has become ubiquitous in today’s world. The human location is shared with different application on a mobile device. This location data can be used by different third-party applications to predict user’s future locations. This is a privacy attack which makes use of user’s past location data.

In this thesis we present an algorithm which predicts user future movements with confidence percentages. This algorithm is first implemented on python using Microsoft Geolife data. This data contains 182 user trajectories data for 5 years. The same algorithm is then implemented on Android device.

The raw trajectories are used from Microsoft Geolife data. This data is first used to find the stay points. The stay points are the places where user spent at least 20 minutes within the radius of 200 m. Once the stay points are found, we form states from these stay points using snapping algorithm. Once this process is done, we start creating Markov chains. This Markov chain is then used to predict user future locations.

**Acknowledgement**

Contents

[**1 Introduction** 4](#_Toc522025522)

[**2 Related Work** 5](#_Toc522025523)

[**3 System Model** 5](#_Toc522025524)

[3.1 Components 5](#_Toc522025525)

[3.2 Assumptions 5](#_Toc522025526)

[3.3 Problem Statement 5](#_Toc522025527)

[**4 Algorithms** 5](#_Toc522025528)

[4.1 Markov Model for Location Prediction 5](#_Toc522025529)

[Assume locations are available as states. 5](#_Toc522025530)

[**5 Implementation** 5](#_Toc522025531)

[5.1 Stay-points 5](#_Toc522025532)

[5.2 States from Stay-points 5](#_Toc522025533)

[5.3 Hourly weights of states 5](#_Toc522025534)

[5.4 Forming Markov chain on states 5](#_Toc522025535)

[5.5 Algorithms 5](#_Toc522025536)

[5.5.1 Variables: 5](#_Toc522025537)

[5.5.2 Algorithms 6](#_Toc522025538)

[5.5 Android implementation 10](#_Toc522025539)

[**6 Evaluation** 10](#_Toc522025540)

[6.1 User data analysis 10](#_Toc522025541)

[6.2 Discussion and Summary 10](#_Toc522025542)

[**7 Conclusion and Future Work** 10](#_Toc522025543)

[7.1 Summary 10](#_Toc522025544)

[7.2 Future Work 10](#_Toc522025545)

# **1 Introduction**

This algorithm is designed to predict human locations in a real-world scenario. The GPS data is taken as input and the processed using the below algorithm.

The Algorithm has several steps:

* \_ Detect stay-points (also detect start or end of the trajectory)
* \_ Group stay-points to form states
* \_ Calculate hourly weights for the states
* \_ Apply Markov chain for the data available

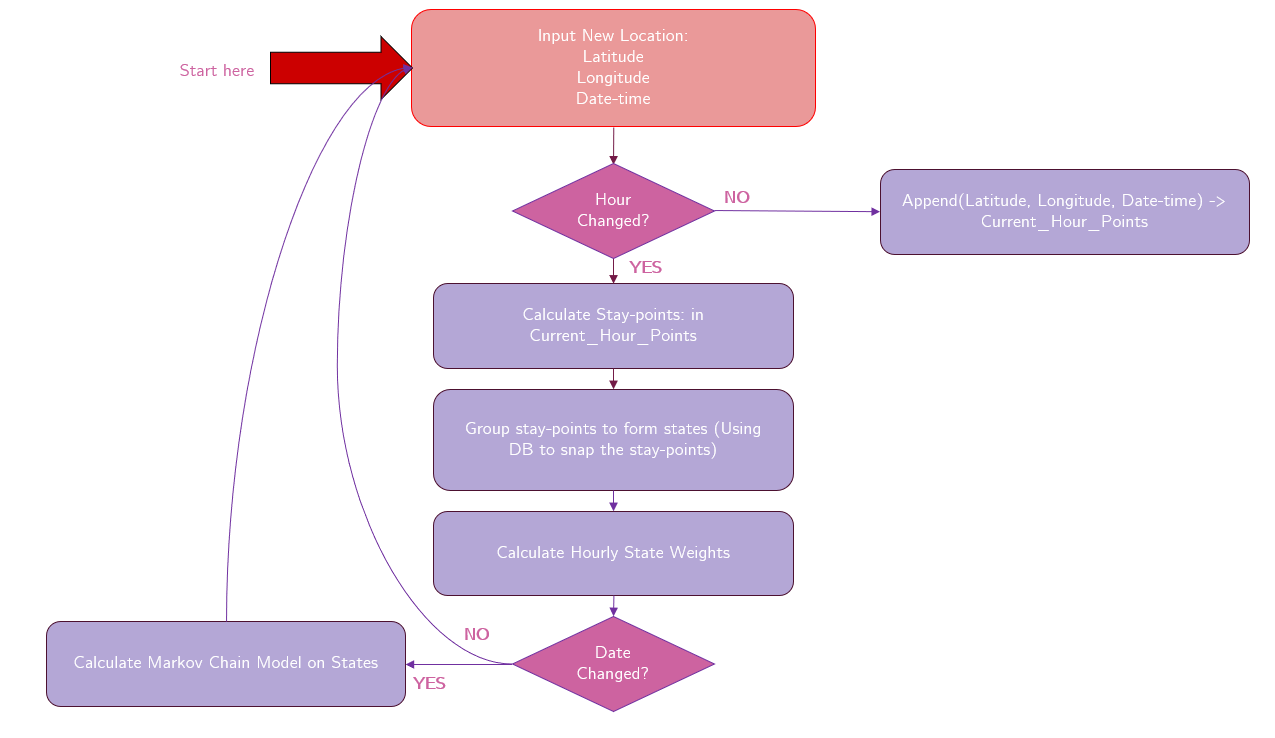


Figure 1: Algorithm Flow-chart

# **2 Related Work**

# **3 System Model**

## 3.1 Components

## 3.2 Assumptions

## 3.3 Problem Statement

# **4 Algorithms**

## 4.1 Markov Model for Location Prediction

Assume locations are available as states. Provide pseudo code.

# **5 Implementation**

## 5.1 Stay-points

Stay-points are any points which are stayed by the user in

user trajectories or it is the start or the end of the trajectory. For example,

if user start at his home, the home itself is a stay-point. Now he moves

towards work, but he visits a cafe in between for breakfast. The cafe is

also, a stay-point and then he finishes his trajectory at work, where work

is again a stay-point. The places like cafe in this case is identified using

distance and time-based clustering. For example, a set of points within

200m with total duration of stay greater than 20 minutes can be regarded

as a stay-point within the trajectory.

## 5.2 States from Stay-points

A state is formed using a group of stay-points. This is done using

a distance threshold for states. All the stay-points within this threshold

distance is grouped together as a single state. This is called snapping

stay-points to the states. The mean of all location latitudes and longitudes

from stay-points within a state are stored per state. Finally, Markov Chain

model is applied to the states. Note: A new stay-point is only added to the

state if after calculating the mean of the new state, all the existing stay-

points still stay within the distance threshold from this mean. This is done

to avoid drifting problem while aggregating the stay-points into states.

## 5.3 Hourly weights of states

## 5.4 Forming Markov chain on states

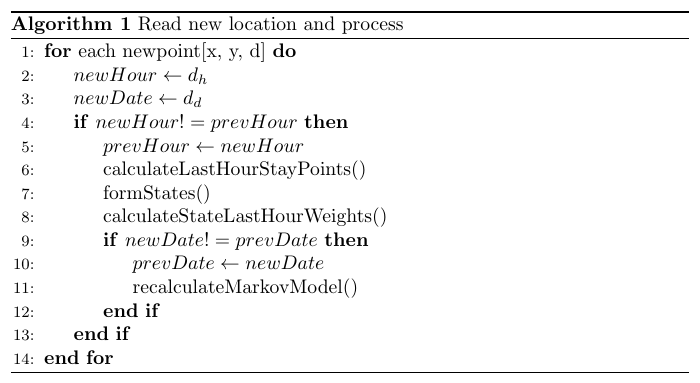
## 5.5 Algorithms

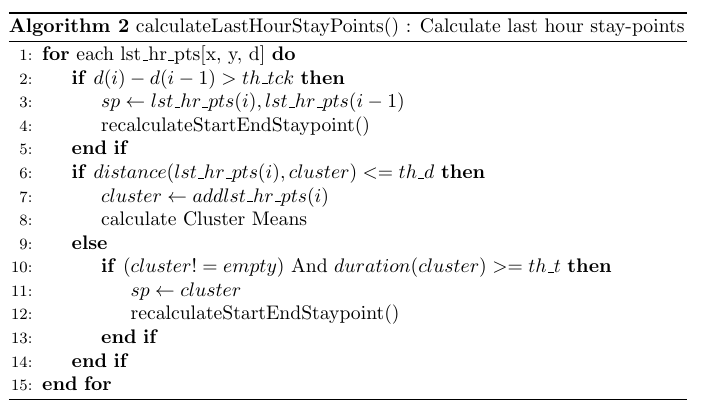
### 5.5.1 Variables:

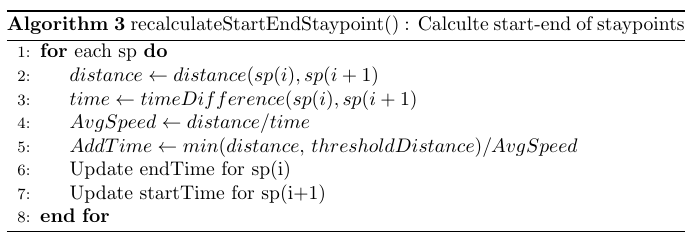
|  |  |
| --- | --- |
| Variable | Description |
| newpoint(x, y, d) | point is a tuple:  (Latitude, Longitude, Datetime) |
| dh | Hour of d datetime |
| dd | Date of d datetime |
| th\_tck | Threshold time for tracking GPS location data |
| th\_d | Threshold distance for staypoints |
| th\_t | Threshold time for staypoints |
| spi(x, y, ds, de) | ith Staypoint  (Staypoint Latitude, Stapoint Longitude, Start Datetime, End Datetime) |
| sti(x, y) | ith State  (State Latitude, State Longitude) |
| lst\_hr\_pts(x, y, d) | Last hour points only  (Latitude, Longitude, Datetime) |
| st\_hr\_wt(id, x, y, 0-24) | State hour weights  (State ID, State Latitude, State Longitude, 0Hrs, 1Hrs..24Hrs) |
| mc | Markov Chain for st(i) to st(i+1) for hour h to h+1 |

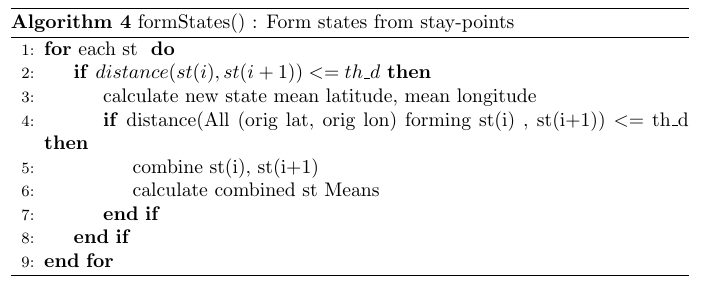
### 

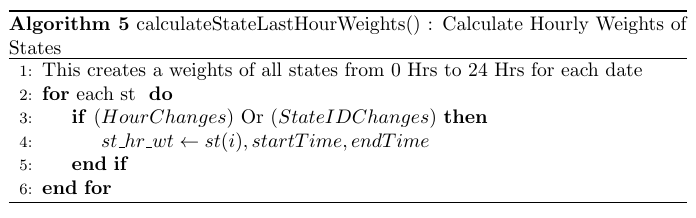
### 5.5.2 Algorithms

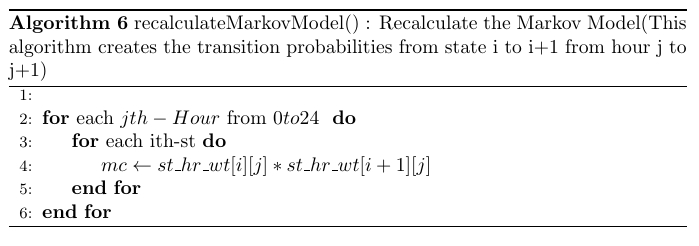












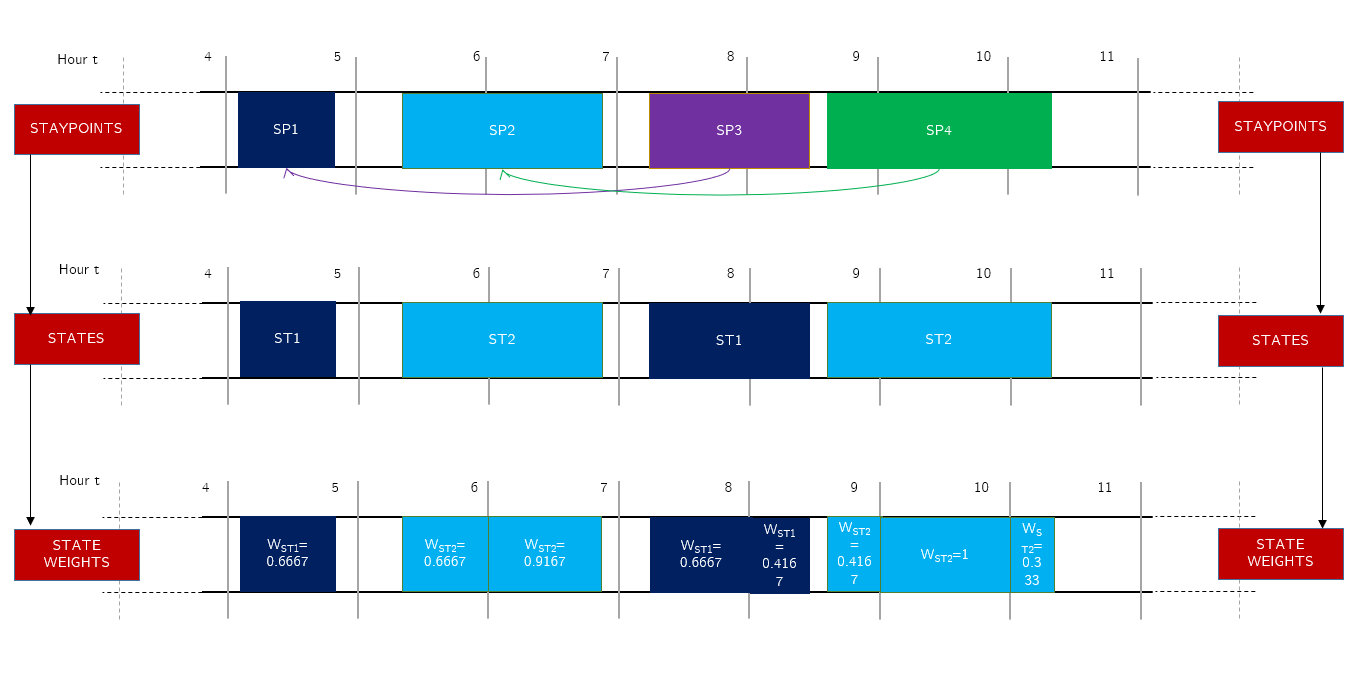


Figure 2: Stay-points to States

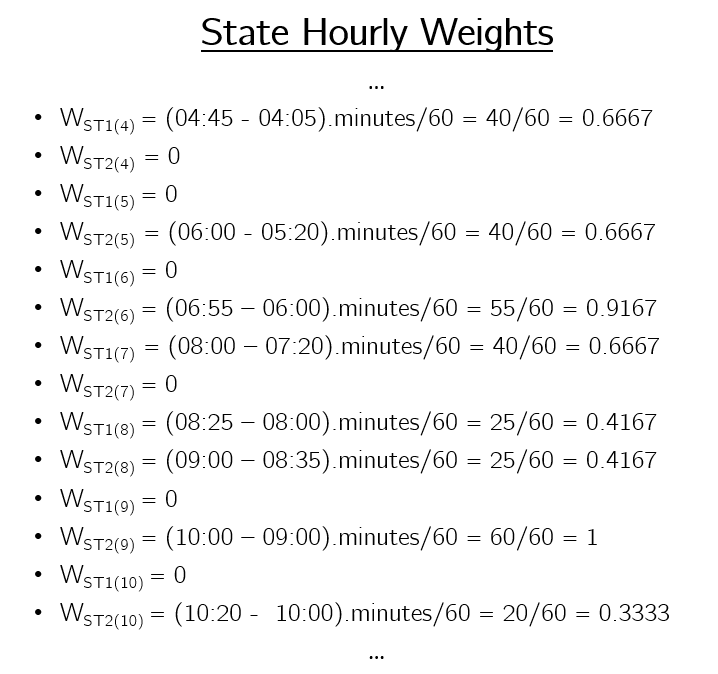


Figure 3: States Hourly Weights

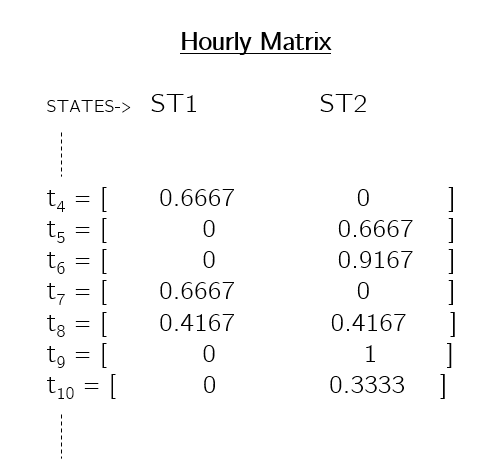


Figure 4: States Hourly Matrices



Figure 5: States Transition Matrix

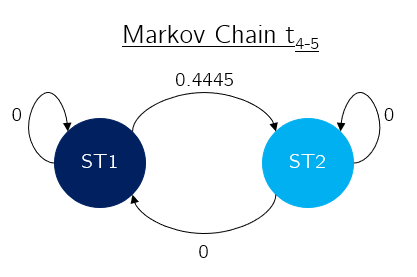


Figure 6: States Markov Chain

## 5.5 Android implementation

# **6 Evaluation**

## 6.1 User data analysis

## 6.2 Discussion and Summary

# **7 Conclusion and Future Work**

## 7.1 Summary

## 7.2 Future Work