Location Prediction Algorithm

Shashank Sharma

August 2018

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

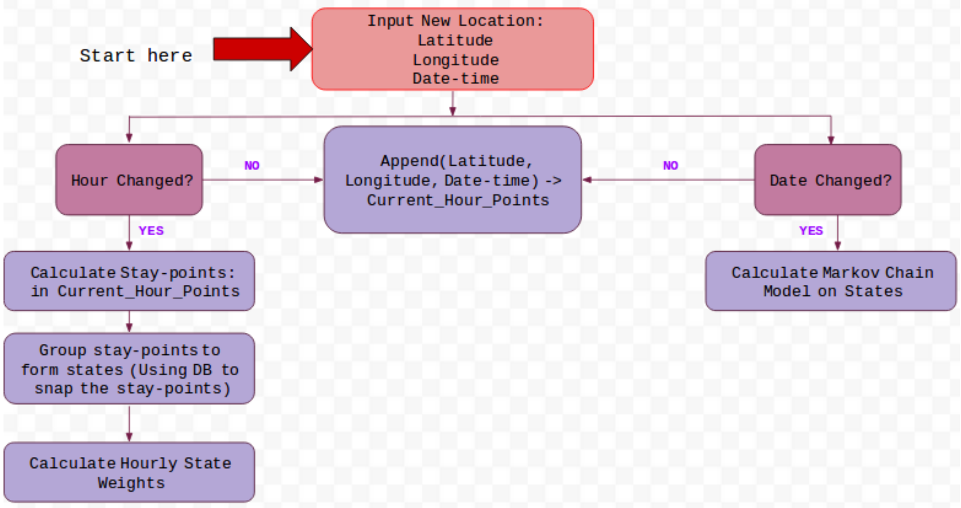
Start here
Input New Location:
Latitude
Longitude
Date-time
Hour Changed?
NO
YES
Append(Latitude, Longitude, Date-time) -> Current_Hour_Points
NO
Date Changed?
YES
Calculate Stay-points: in Current_Hour_Points
Group stay-points to form states (Using DB to snap the stay-points)
Calculate Hourly State Weights
Calculate Markov Chain Model on States

Figure 1: Algorithm Flow-chart

2 Definitions

- **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 move towards work, but he visit 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.

- **State:** 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 are 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.*

3 Algorithms

Algorithm 1 Read new location and process

```
Read new latitude, new longitude, new datetime information and process these new points
while $NewLocationDetected == True$ do
    Set $newHour = datetime.hour$
    Set $newDate = datetime.date$
    if $newHour! = prevHour$ then
        $prevHour \leftarrow newHour$
        calculateLastHourStayPoints()
        formStates()
        calculateStateLastHourWeights()
    end if
    if $newDate! = prevDate$ then
        $prevDate \leftarrow newDate$
        recalculateMarkovModel()
    end if
end while
```

Algorithm 2 calculateLastHourStayPoints() : Calculate last hour stay-points

```
Calculate stay-points
*trackingThreshold:* Maximum time distance between two points
*thresholdDistance:* Stay-point threshold distance covered
*thresholdTime:* Stay-point threshold time spent
**for** $eachLastHourPoint$ **do**
    **if** $(point(i).datetime - point(i-1).datetime) >= trackingThreshold$ **then**
        Add point(i), point(i-1) as stay-points
        recalculateStartEndStaypoint()
    **end if**
    **if** $distance(point(i), cluster) <= thresholdDistance$ **then**
        add point i to cluster
        calculate Cluster Means
    **else**
        **if** $(cluster! = empty)$ And $duration(cluster) >= thresholdTime$ **then**
            Add this cluster to stay-points
            recalculateStartEndStaypoint()
        **end if**
    **end if**
**end for**
```

Algorithm 3 recalculateStartEndStaypoint() : Calculte start-end of staypoints

```
**for** $eachStaypoint$ **do**
    Set $distance \leftarrow distance(staypoint(i), staypoint(i+1)$
    Set $time \leftarrow timeDifference(staypoint(i), staypoint(i+1)$
    Set $AvgSpeed \leftarrow distance/time$
    Set $AddTime \leftarrow min(distance, thresholdDistance)/AvgSpeed$
    Set $endTimeofStaypoint(i) \leftarrow endTimeStaypoint(i) + AddTime$
    Set $startTimeofStaypoint(i+1) \leftarrow startTimeStaypoin(i+1) + AddTime$
**end for**
```

Algorithm 4 formStates() : Form states from stay-points

```
Form each unique stay-point as individual state
Now, within this loop, start combining the states
*thresholdState:* State distance threshold
**for** $eachState$ **do**
    **if** $distance(state(i), state(i+1)) <= thresholdDistance$ **then**
        calculate new state mean latitude, mean longitude
        **if** $distance(allExitingState(i)Staypoints, NewMeanLatLong) <= thresholdState$ **then**
            combine state i, i+1
            calculate State Means
        **end if**
    **end if**
**end for**
```

Algorithm 5 calculateStateLastHourWeights() : Calculate Hourly Weights of Statesudl

```
This creates a weights of all states from 0 Hrs to 24 Hrs for each date
**for** $eachState$ **do**
    **if** $(HourChanges)$ Or $(StateIDChanges$ **then**
        Calculate the start and end of the state i
    **end if**
**end for**
```

Algorithm 6 recalculateMarkovModel() : Recalculate the Markov Model

```
This algorithm creates the transition probabilities from state i to i+1 from hour h to h+1
**for** each $Hth - hour$ from $0 - 24$ **do**
    **for** each $ith - state$ in $state - hourly - weight$ **do**
        $state(i) - > state(i+1)$ transition for $H - hour = Matrix[state] * Matrix[State + 1]$
    **end for**
**end for**
```