ADAPTIVE K-MEANS CLUSTERING FOR HUMAN MOBILITY MODELING AND PREDICTION
Anu Sharma(axs3617@rit.edu)
Advisor: Prof. Peizhao Hu

ABSTRACT

Human movement follows repetitive trajectories. There has been extensive research going on about how we can use the location data of individuals. There is a potential of predicting the next location of an individual based on the past mobility traces. This paper is related to analytics of clustering which can provide help in the above mentioned research area. The objective of clustering is to group GPS data points into appropriate circular clusters of radius not more than given Wi-Fi range. We are considering Wi-Fi range because GPS coordinates are very sensitive to movements, which can cause error in prediction algorithm. This improved version of K-Means clustering algorithm can help in range of applications that are based on human mobility traces, for example traffic regulation.

INTRODUCTION

The collection of GPS coordinates from various user machines like mobiles, tablets, laptops, etc. can provide rich data about the user mobility. There has been some extensive research going on about how to use this data. One of the main area of the research is to predict the next location of an individual based on the past mobility traces. The study in this paper is about how we can group these GPS coordinates into circular clusters so that we can achieve higher accuracy in prediction algorithm.

The objective is to implement an adaptive K-Means clustering algorithm that will form clusters of radius not more than given range. Why do we need this improved version of K-Means clustering? First, the value of K is not known so we need to find out the value of K iteratively. Second, we want to form the clusters of radius not more than given Wi-Fi range. Also, we want to form clusters in circular shape because we want to mimic the shape of Wi-Fi range. As clusters are circular, there is going to be overlapping between the clusters. In this implementation we also tried to reduce the overlapping that exists between the clusters.

Paper[2] shows that clustering can affect accuracy of prediction algorithm. Therefore, this algorithm can help in improving the accuracy of prediction algorithms that can be used in many applications. For example, it can help in contributing to many location-based services that will make the users aware of any important information related to their next anticipated location. It could also help in Traffic regulation by navigating user towards the desired predicted location and make him aware of any unusual activity that has happened on the expected route.

The data collected for our experiments is the location data of taxi drivers from Rome(Italy)[4]. The ultimate goal of this study is to model human mobility traces into graphs to make better
inference. The detailed explanation of the algorithm and modeling of the dataset is provided in the later sections.

BACKGROUND

The problem of data clustering has been widely researched in data mining and machine learning. Here, we want to perform clustering based on the basis of distance function. Two of distance based clustering categories are: Flat and Hierarchical. Flat clustering algorithms includes K-Means, K-Medians etc. and Hierarchical clustering includes Agglomerative etc. We will use Iterative K-Means clustering as our base implementation. We are using iterative version because the value of K is not known. Also, we want to form circular clusters because we wish to mimic the dome shape of Wireless connections (Wi-Fi). In Paper[1] and Paper[2] authors implemented Prediction algorithm using Naive clustering approach and Iterative K-Means Approach. The basic iterative approach is explained below in the flowchart (Figure 1). Paper[1] and [2] showed that the accuracy of prediction algorithm depends on clustering of GPS coordinates. Therefore, it is necessary to have an efficient clustering algorithm. The clustering algorithm in [3] was able to provide the basic features of clustering algorithm but it was not time efficient. It took a very long time to find the optimum value of K as the initial value of K was set to 2. Therefore, it took much more iterations to find the appropriate K value. Second, there was too much of overlapping between the clusters formed. Thus, the accuracy of prediction algorithm was not efficient. We try to address solutions to these problems in our algorithm.

MODELING

The input data we are using is the dataset about the taxi trajectories from the city Rome(Italy)[4] which was recorded over a month. The raw input data file is a comma-separated file that has taxid, timestamp and GPS coordinates as data. For example: 156;2014-02-01
This file was processed and data for each taxi was stored into separate file using the logic from [1]. The data for taxid 2 is shown in Figure 2.

These files are then provided as input to the Adaptive K-Means clustering algorithm that divides the dataset for each taxi into clusters of a given Wi-Fi range. The data is then modeled into graphs. The vertex of a graph is a cluster of GPS coordinates and edge is the movement between two clusters. Edge-weight is the frequency of the movement between the clusters (Figure 3).

ALGORITHM

This research includes modeling of the mobility data into graphs. The approach to solve the problem discussed in the paper includes following steps: First, divide the data set into clusters. A cluster is a group of GPS coordinates with a given specific range. Second step is to form a graph of clusters.

The adaptive K-Means algorithm is developed to provide the solutions for the following problems: First, the value of K is not known. Also, we have to consider the radius of the clusters as we don’t want any cluster’s radius more than the given Wifi range R. Second, we want the algorithm to be time efficient. Third, we want to reduce the number of clusters by removing the clusters with very small radius. Fourth, we want to minimize the overlapping of clusters.
To provide the solution for the first two problems, we came up with the following approaches:

1. Compute the initial value of $K$ according to the dataset. In basic iterative K-Means clustering, the initial value of $K$ is set to 2. This approach takes too many iterations to converge the appropriate $K$ value. So, we are finding the value of $K$ according to the dataset. In order to find the initial value of $K$, we find the bounding box which covers all the GPS coordinates and then compute the value of $K$ using the formula: $K = (\text{length}/2*R) \times (\text{breadth}/2*R)$ where length and breadth are the dimensions of the bounding box and $R$ is the given WiFi range. Then we find the value of $K$ using binary search technique. After performing clustering, we check if the radius of any cluster $C_i > R$ then $K = 2*K$ else if ($r(C_i) < \text{minimum} R$) then $K = K/2$. Example is shown in following figure 4.

![Figure 4: Finding value of K](image)

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Value of K</th>
<th>If $C_i &gt; R$</th>
<th>If $C_i &lt; \text{min}R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>100</td>
<td>True</td>
<td>-</td>
</tr>
<tr>
<td>#2</td>
<td>200</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>#3</td>
<td>150</td>
<td>True</td>
<td>-</td>
</tr>
<tr>
<td>#4</td>
<td>175</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>#5</td>
<td>163</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

2. Apply cluster refining techniques to the neighboring clusters. We have implemented two cluster refining techniques: Shuffle and Merge (explained below) which help in refinement of the clusters by decreasing overlapping. Before shuffling and merging, we identify the neighboring clusters to improve the efficiency of the algorithm.

![Figure 5: Cluster i with neighboring cluster j](image)

Before shuffling the points, we find the neighbors based on the following conditions.
Let's say we have cluster $C_i$ with $r(C_i) > R$ and we want to find the neighbors for this cluster.

```python
foreach(cluster $C_j$ in clusters)
    if($r(C_j) > R$), then ignore $C_j$ because $r(C_i) + r(C_j) > 2R$
    else if ($r(C_j) == R$)
        Check if clusters are overlapping:
        if($r(C_i) + r(C_j) >= d(C_i, C_j)$)
            Yes clusters are overlapping hence $C_j$ is neighbor of $C_i$
        else
            Ignore $C_j$
    else if ($r(C_j) < R$)
        if($d(C_i, C_j) < r(C_i) + R$)
            $C_i$ is a neighbor of $C_j$
        else
            Ignore $C_j$
```

Before merging the points we find the neighbors based on the following conditions.

Let's say we have cluster $C_i$ with $r(C_i) < R$ and we want to find the neighbors for this cluster.

```python
foreach(cluster $C_j$ in clusters)
    if($r(C_j) < R$)
        if($d(C_i, C_j) < 2*R$)
            $C_i$ and $C_j$ are neighbors
```

To improve the prediction algorithm accuracy we introduced the following two techniques.

1. Shuffling
   After finding the initial value of $K$ we select $K$ random data points from the input data file as centroids of $K$ clusters and then add points to these $K$ clusters. Then we shuffle the points from cluster $C_i$ if $r(C_i) > R$ to neighboring cluster in order to reduce the radius of $C_i$. We stop shuffling if no data point moves to the neighboring cluster. This whole process of clustering repeats for every value of $K$ till we find the appropriate $K$ value.

2. Merging
   In this process, we merge cluster $C_i$ with neighbor cluster $C_j$ (if $r(C_i) < R$ & $r(C_j) < R$) into cluster $C_i$ if $r(C_i) <= R$. This process helps in reducing the number of clusters by removing the clusters with small radius.
The complete algorithm is explained below:

Input - GPS coordinates with timestamp
Output - Clusters with GPS coordinates

1. Read all \(N\) coordinates from the input file.
2. Initialization of \(K\):
   1. Find the bounding box of all the GPS coordinates (min/max values of latitude, longitude).
   2. Calculate the length and breadth of the box.
   3. Given the range \(R\), calculate the value of \(K\)
      \[ K = \left(\frac{\text{length}}{2R}\right) \times \left(\frac{\text{breadth}}{2R}\right) \]
3. Perform Clustering:
   1. Randomly select the data points from the dataset as the centroids for \(k\) clusters.
   2. Add all points to these clusters. Calculate the centroid of each cluster while adding data point to it.
   3. Identify neighbors (explained below)
   4. Shuffling:
      - For each Cluster \(C_i\)
        Sort the \(d(p, C_i)\) in descending order (\(p\) is a point in a cluster)
        for each data point \(p\):
          if \(d(p, C_i) > R\)
            Move \(p\) to neighboring cluster with shortest distance
          else
            break;
4. After performing clustering, check
   for each cluster \(C_i\)
   if \(r(C_i) > R\)
      needMoreClusters = true
      break;
   else
      needMoreClusters = false
5. while (needMoreClusters)
   1. \(K = 2^k\)
   2. Perform step 3 and step 4
6. if \(r(C_i) <= R\)
   1. Check if the radius of any cluster is smaller than the given minimum wifi range.
   2. If \(r(C_i) < \text{min} R\)
      lowerbound = \(K/2\);
      upperbound = \(K\);
      while \(r(C_i) < \text{min} R \lor r(C_i) > R\)
        \(K = \text{lowerbound} + ((\text{upperbound} - \text{lowerbound})/2)\)
        Perform clustering (step 3)
        if (any cluster's radius > wifi range)
          Set lowerbound = \(K\)
        Else if (all cluster's radius are smaller than wifi range && any cluster's radius < given minimum wifi range)
          Set upperbound = \(K\)
7. Find neighbors
8. Merge clusters based on neighboring clusters
   while \(d(C_i, C_j) < R\)
      Find neighbors
      Add all the data points from the two clusters into temp cluster
      Calculate radius of temp cluster
      If (radius of temp cluster is less than wifi range)
        Add temp to the clusters list
      Remove those two clusters from the clusters list.
Figure 6: Flowchart of Adaptive K-Means Clustering Algorithm

During research we also came up with a different approach: Find the bounding box of all GPS coordinates and divide the box into smaller square boxes. Then perform clustering on the points available in the smaller sq. unit area. But after further investigation we figured out that this approach will not work because if two points are really close but fall in different boxes then they would be assigned to 2 different clusters instead of a single cluster. This approach would be faster but accuracy would be compromised.

RESULTS AND EXPERIMENTS

Using the methods in paper [3], we were able to plot the resulting clusters on google maps using google API. Figure 7 shows the GPS coordinates with color coded clusters for taxi Id 2. The centroid of the the clusters are marked by blue color.

![Figure 7: GPS coordinates with color coded clusters](image)

The algorithm has been evaluated after performing the experiments using taxi driver dataset. The dataset was first processed using the methods from [1]. Then clustering was performed on few taxi Ids with the given Wifi range of 400 meters. Figure 8 shows the plotting of number of clusters per iteration for taxi Id 2. It shows that merging reduced the number of clusters substantially by removing the clusters with very small radius.
Figure 9 shows the cumulative frequency distribution of radius of all clusters formed for taxi Id 2. It shows that the radius of around 50% of the clusters is less than 300 meters. Figure 10 shows the distribution of number of data points per cluster. It shows that there is a probability that almost 70 percent of the data points will fall under clusters which having total number of data points around 50.

Following figure 11 shows the stability of the algorithm. We ran the algorithm on taxi id 2 for six times. The resulting number of clusters lie between 485 to 495 with the average of 490 clusters and standard deviation of 4.7. This shows that the algorithm is pretty stable even after the randomness involved.
CONCLUSION

In this research we analyzed that the clustering algorithm plays a very important role in improving the accuracy of prediction algorithm. Experiments showed the new clustering algorithm increased the prediction accuracy from 30% to 65%. Thus, we can conclude that the new cluster refining techniques (Shuffle and Merge) helped in forming clusters with less overlapping. The algorithm is also time efficient as it is able to find the optimum value of K in a very less time when compared to basic iterative K-Means algorithm. The clustering algorithm used in [2] was taking more than two hours for clustering for one taxi Id whereas our algorithm is taking around twenty minutes for the same taxi Id. The time complexity of this algorithm is O(n^2) which is better than the complexity of agglomerative clustering i.e. O(n^3).

FUTURE WORK

For future work, there is a potential of improving the time efficiency by using Geo hashing technique. For this, we can place the data points into different bins by performing quantization using geo hashing technique. After dividing the dataset, perform k-means clustering on the bins instead of the entire dataset. This will increase the speed of the algorithm. Also, there is still a potential of decreasing the overlapping of clusters further.

REFERENCES

[2] NextMove: Modeling mobility trajectories for next location prediction, Sonam Padwal, RIT Capstone Project, Fall 2015
[3] Implement Clustering Algorithm for Contact Prediction from Human Mobility Traces, RIT May 2015