TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SYED ARSALAN AHMED SHAH

SECURE FORECASTING OF USER ACTIVITIES FOR DISTRIB-UTED URBAN APPLICATIONS

Master of Science Thesis

Examiner: Prof. Jose Luis Martinez Lastra
Examiner and topic approved on 07.09.2016

ABSTRACT

SYED ARSALAN AHMED SHAH: Secure forecasting of user activities for distributed urban applications
Tampere University of technology
Master of Science Thesis, 65 pages
April 2018
Master’s Degree Programme in Automation Technology
Major: Factory Automation and Industrial Informatics
Examiner: Prof. Jose Luis Martinez Lastra

Keywords: mobility traces, activity recognition, activity forecasting, datamining, smart cities, machine learning, Markov chains, adaptive-neuro-fuzzy-inference-system, MATLAB, web services

Modelling human mobility is an interesting yet challenging research topic. Such mobility models can give valuable insight into user behavior. Such models can be used to forecast movement of people. Even though an interesting problem, it was not studied as widely due to lack of available mobility data. But modern communication and digital infrastructure has solved this problem. Thus, as a result, over the past decade and a half, this topic has attracted a lot of attraction. The modelling and forecasting of human mobility has widespread applications from transportation to advertisement. Such models can be used to in a collaborative manner to segment people or used in isolation to bring better services to an individual.

Previous researches have presented different approaches for modelling human mobility. These range from neural networks to Markov chains. Some researchers have focused on location data while others have worked with accelerometer data. There are also recommendations to add more information to the data to understand the motive of mobility.

This thesis approaches the problem of forecasting human mobility in the form of activities. GPS data is analyzed to mine information and find patterns. The forecasting is done in a twostep process. The first step is to analyze the data to identify and label activities, that are done on a routine basis. This is achieved by using an Adaptive Neuro-Fuzzy Inference System. This additional information helps understand the motive of moving from one place to another. In the second and final step the Markov Chain model is built for the movement among visited locations. The forecasting is done with respect to current time and location, keeping in view the motive of movement. The proposed system is implemented in JAVA and deployed as a combination of RESTful web services. Finally, accuracy tests are made on different datasets which show promising results.

PREFACE

This thesis is based on the research I carried out in FAST-Lab of Tampere University of Technology. The research was done as part of an EU2020 Horizon project, MUSA. Even though this thesis work does not directly relate to the domain of factory automation. I believe that as we move towards digitalization, similar techniques will come more into play within the factory automation domain as well.

Completing this thesis turned out to be more challenging than I had initially anticipated. It required a lot of background study as I had jumped, by my own liking, into an unknown domain. The biggest hurdle for me was the lack of defined goals from the beginning. This resulted in me losing motivation many times. But finally I was able to finish what I had started.

A lot of help came on the way. First of all I would like to thank Professor Dr. Jose Luis Martinez Lastra for giving me the opportunity to work at FAST. It was indeed a pleasure working there and I will carry those nice memories from the lab. I also have to acknowledge Dr. Andrei Lobov for his guidance during my studies. Luis and Anne were the most cooperative supervisors anyone can have and I would like to thank them for their help. Dr. Borja Ramis deserves a special mention here for his motivation and quick response to all my queries and for reviewing my thesis.

I would like to thank my family for their support, especially my parents for their prayers and trust in my capabilities. My friends Ahsan, Ali, Amir, Aneeb and Umair who provided moral support, counsel and good company during my studies.

Finally I dedicate this thesis to my lovely wife who has always supported me in whatever way possible.

Tampere, 25.4.2018

Arsalan Shah

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF SYMBOLS AND ABBREVIATIONS

GSM	Global system for mobile communication
GPS	Global positioning system
POI	Points of interest
ML	Machine learning
KDD	Knowledge discovery in databases
HMM	Hidden Markov Model
MMM	Mixed Markov Model
ANN	Artificial neural network
ANFIS	Adaptive neuro fuzzy inference system
GI	Geographic-triggered Intentions
TI	Temporal-triggered Intentions
SI	Semantic-triggered Intentions
MDC	Mobile data challenge
MC	Markov chains
FIS	Fuzzy inference system
API	Application programming interface
CA	Certificate authority

1. INTRODUCTION

This thesis is divided into seven chapters. The first chapter introduces the problem at hand. It gives the motivation behind the thesis followed by presentation of the problem statement. The chapter ends with a brief discussion about the practical aspects of the work, such as expectations and objectives; and setting forth the limitations of the scope of work. The second chapter attempts to discuss the work done so far on the subject, with more emphasis on the latest findings and developments. The next two chapters build on the work discussed in chapter two and provides insight into the approach taken in this thesis to solve the problem and how it has been implemented. The chapters after that state the results of the implementation, discuss those results and draws conclusions based on them.

1.1 Motivation

Human beings can be categorized on the basis of what they do. Our physical activities, no matter how big or small, play a big role in defining who we are. Although there are many activities that we undertake as experiences and do not continue with them on a regular basis, this combined with the fact that we as humans have a certain amount of randomness in our decision making; makes it seem that our daily routines are completely overshadowed by randomness and there is no easily identifiable pattern to what we do. But we can also be certain that even if the patters are not easily visible they must be present. This topic of identifying patterns in daily human activities has interested many researchers because identifying these patterns is actually identifying the person itself.

Almost all the activities we do require movement from one place to another and thus studying a person's geospatial information provides insight into what a person might be doing at a specific time. For example, if it is known that the position of a person points out to a school on a daily basis then it can be inferred, with a certain level of confidence, that the person can either be a student studying there or an employee working at the school. So, analyzing human mobility patterns, provides us with an opportunity to identify and understand patterns in activities that we undertake as humans. Modern telecom infrastructure has enabled collection of geospatial data of users relatively easily. This data has been used in identifying the aforementioned patterns.

Identifying the activities that a person does as their routine can tell us a lot about that person and it can be used in many ways. Identifying individuals with similar activities can lead to identifying groups of people with similar routines and bring customizes services to them; for example, knowing what percentage of people prefer fitness activities over leisurely activities in their free time can help relevant businesses offer better services. Similarly sending targeted advertisements to these groups or even individuals can be made easier.

Not only does this information hold value for commercial organizations but being able to identify routines and predict activities of individuals can greatly enhance their experiences. Suggestions based on what a person does or likes to do are generally welcomed by users. Take, as an example, a user who travels on a certain route every day. If we can know in advance that the user will go to a certain place and we have information on other users as well we can build models to predict the traffic situation at a specific time of the day and advise the users to take alternate routes if a traffic blockage is expected. The city municipalities can better plan public transport routes and timings if the information of enough users is available. Carpooling can be made much more effective and efficient based on this information.

It has to be understood that the activity recognition and forecasting does not hold much value as a standalone application. But such a recognition and prediction system can be used to develop *smart* applications. The modern applications that utilize this concept and build smart applications are moving towards a distributed architecture. In such architectures, one part of the application is not affected by the other. The applications are built with a modular approach where the smartness is introduced on a component level. This gives the developers the flexibility of how and when to use a certain component. It also provides the opportunity to use third party services.

Although this distributed approach has its advantages it also brings forwards new challenges. These challenges are amplified when sensitive user data is involved. The matter has become worse with cloud and multi-cloud-based applications where distributed services are provided with different cloud platforms interacting with each other, thus increasing the surface of attack.

Due to its huge area of application it can be said confidently that mobility data it is valuable not only for the owner of the data but also commercial organizations. This leads to a difficult situation, where people do not want to share their data due to privacy concerns and also due to the fear that the data will end up in the wrong hands. The discussion of privacy concerns is very lengthy and somewhat academic and hence no general agreement is seen on the topic in the near future. But the concern and threat of data theft is very real and unanimously agreed. As a general practice, developers of technology focus on the functionality and do not emphasize on the security aspect as much. And even if they focus on the security in the past we have seen countless examples of data breaches in big technology companies. This has made users aware of the importance of their personal data and potential of the hackers thus making the users less confident in storing their personal data on third-party application.

The above discussion highlights the importance of building a system that is accurate in forecasting user activities, which is easy to use and is safe from external threats.

1.2 Problem Statement

The problem of this thesis is to develop a system which is able to forecast activities by analyzing the history of user mobility traces. The system upon identifying such activity patterns from user history should be able to forecast what activity the user will perform given his/her contextual information. Not only the system has to be accurate in forecasting the users next activity but the system as a whole should be secure in a way that during no phase of the information flow or data storage the system is vulnerable to external attacks. The system has to have the ability to be integrated with-in an existing system and deployed independently.

The forecasting part of the thesis will focus on developing an algorithm which is able to first identify an activity and based on the activity database and current context forecasts the next activity in real-time. The deployment and security of the system will have to take into account many aspects and cope with external threats.

1.3 Objectives

There can be many ways and approaches to solve the problem of forecasting the next activity. In this thesis the approach undertaken is, to forecast the coordinates of the location the person is likely to visit provided their current and past context. Information about the activity can be inferred based on the forecasted coordinates.

The primary objective is to find an efficient algorithm that can accurately forecast the location a person is likely to visit given their history and current context. Defining the context is also part of the problem. The algorithm developed will not only be the best one from the accuracy point of view but it has to scale well for any number of mobility traces of a given user. It has to meet the requirements of a real-time system, so it can be integrated into any mobility related application. It also needs to be secure from external attacks. It is important that the final implemented system is capable of independent deployment. This will make it possible for the system to be integrated with other apps.

It is important that the final implemented system is capable of independent deployment. This will make it possible for the system to be integrated with other apps.

1.4 Limitations

As stated in the first section of this chapter, the user mobility information can be interpreted in many ways and there are humongous applications where this information can be utilized. Most of the applications would involve a collaborative approach where the information from all the users is combined to make large impact forecasts or conclusions. In this thesis the focus is to solve the problem of forecasting activity or location of an individual only keeping his/her history in consideration. This may restrict the advantages of this work but future work can be built on this thesis.

2. THEORETICAL BACKGROUND

In this chapter, first basic terminology is defined and explained. This will lay a foundation to better understanding the contents of this chapter as well the following chapters. Furthermore, the chapter discusses State-Of-The-Art (SOTA) methods and techniques used in activity recognition and forecasting; followed by a discussion on distributed systems and how these systems are utilized in making modern applications. The chapter ends with a brief overview of standard approaches and frameworks which make possible building up distributed systems or applications.

2.1 Activity Recognition and Forecasting

Activity can most simply be defined as anything that people do [1]. Human beings may do many activities, from drinking water to running a marathon; from a once in a lifetime experience to countless monotonous tasks. In general, human activities involve a series of actions. These actions make an activity unique. One of the most form of action involved in performing an activity is movement. In this thesis all activities that are considered, involve movement in a *spatiotemporal* plane. This means that this thesis tries to identify and forecast only those activities that involve movement from one point to another. It also only focuses on activities that are undertaken on a repeated basis. Human mobility data captures the major activities performed by them. This helps in finding a lot of information by focusing on the mobility data only.

Mobility data can be collected with different means and varying forms e.g. the most common source of collecting mobility data is telecommunication infrastructure comprising of mobile phones and global system for mobile (GSM) towers. A mobile phone can be connected to one GSM tower at a time and based on the movement of the person the connection to the tower change. The information about a certain mobile phone's connection to different towers (geospatial data), when combined with the time information (temporal data) can enable construction of the trajectory followed by the owner of the mobile phone. Similarly, Global Positioning System satellites (GPS), Wi-Fi and cellular data connections can record locations. These locations are available as, or can be converted into, coordinate-points i.e. latitudes and longitudes. As in the case of GSM towers, when this geospatial information is combined with timestamps; it enables generation of the actual path of motion. Each set of geospatial and temporal data is called a *mobility trace*. So, it can be said that a path or trajectory can be reconstructed by combining many mobility traces. [2], [3]

Depending upon the amount of mobility traces and their sampling time, trajectories with different level of information can be obtained. In the daily routine people commute from one place to another or they stay at a place. This information of travelling and staying is captured in mobility traces. If the mobility traces are analyzed it is possible to separate the traces that indicate motion from those that indicate no or very little motion [4]. Since the places where people stay are significant because people spend their time in these locations, these locations can be called as *stay point(s)*. If the stay points are further filtered based on some criterion the remaining stay points are called *points of interest* (POI) [4].

Research done in [5] studied a large set of mobile phone records of users. They discovered that people have a pattern to their mobility. Based on this finding they proposed a mathematical model for human mobility. Their conclusion was that, opposed to previous understanding that human mobility followed levy flight and random walk. In addition to that, they claimed that it is possible to estimate the likelihood of finding somebody in a certain location. One explanation for this finding was that in spite of large travel histories humans, on a regular basis, travel to a limited number of locations and follow reproducible patterns. This is a very intuitive and logical explanation for their finding.

Assuming that human activities are strongly dependent on their geospatial location and having established that it is possible to identify patterns in human mobility. We can say that an activity taken by someone can be recognized and predicted given that their mobility data is available. The problem at hand is that we want to know where the user will go to, called the point-of-interest (POI), given his/her current location and/or current time and/or day. Different approaches have been put forward to model the problem. The most common approaches either involve modelling the system as a Markov chain [6]–[8] or using machine learning (ML) techniques [2], [9]. The next section takes us through these techniques.

2.2 Knowledge Discovery in Databases (KDD) and Data mining

Even though selecting the modeling approach for activity recognition and forecasting is an important step; pre-processing of data to extract information from raw data and feeding it to the model is an equally crucial part of solving the problem.

Coined in 1989, the term KDD is used to define the process of extracting useful knowledge from voluminous data [10]. Data mining is one of the steps of the KDD process and can be defined as can be defined as the process of applying specific algorithms for inspecting and sorting data to identify patterns of interest [11]. One of the most essential task of the KDD process is to transform the low-level raw data into other useful forms. Both the terms data mining and KDD are used alternatively but it is essential to understand the difference between the two. Figure 2.1 shows the four steps of the KDD process, where step 4 is the data mining step.

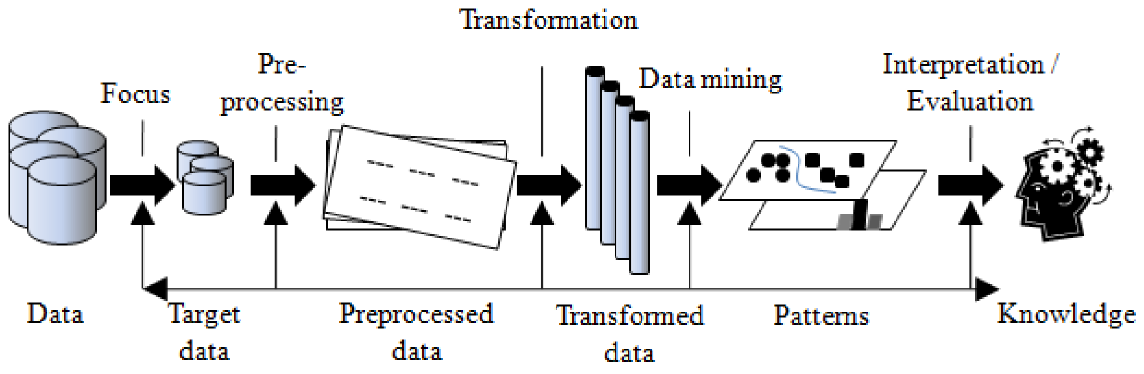


Figure 2.1** The knowledge discovery process **[12]

In todays *connected* world huge amounts of data is generated on a daily basis. In 2015 Frobes [13] reported that in the past 2 years more data has been generated than in the entire history of the human race, and it is predicted that by 2020, 1.7 MB of new data will be generated every second. This does not seem to be surprising, given the fact that there are more than 3.8 billion internet users in the world and this number is increasing by the minute [14]. Figure 2.2 shows the internet users by region of the world. Every day close to 3 billion emails are sent, around 4 billion google searches are made and 500 million tweets are made [15].

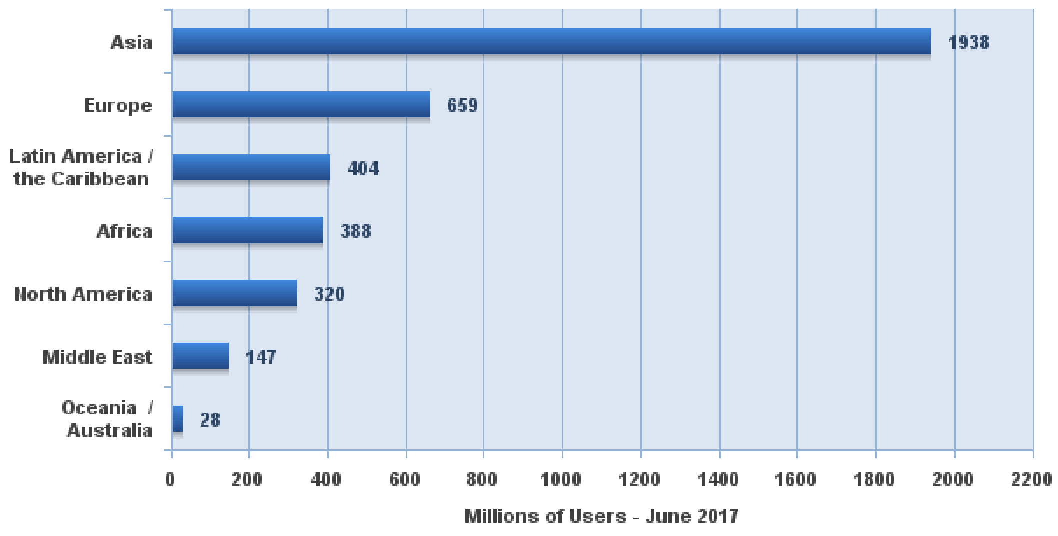


***Figure 2.2** Internet Users in the World by Geographic Regions - June 30,2017 [35]*

Similarly, devices generating data are also increasing steadily. This thesis is focused on human mobility patterns, and smart phones are the most important source of gathering this data. It can be seen from Figure 2.3, that the number of smart phone users is also on a rise.

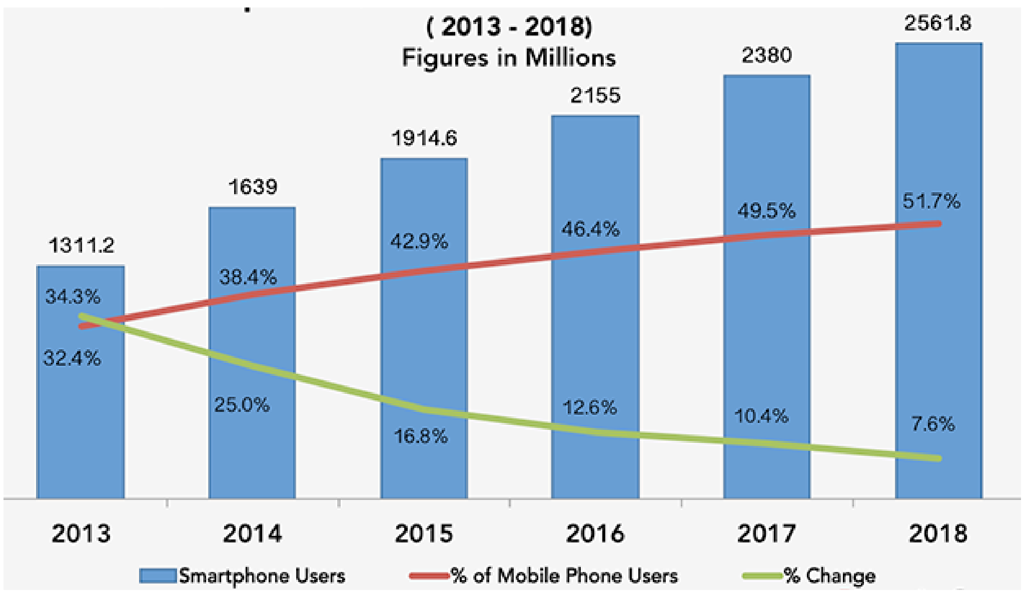


Figure 2.3 *Smartphone Users Worldwide (2013 - 2018) [37]*

Since the number of people generating data and the devices through which data is generated is both increasing we see that the growth of data is two dimensional. Consider n is the number of users whose data is available and d is the types of data available for each n-th user and this data is recorded in a database. We get a table with dimensions n x d. In the simplest case if both n and d increase linearly in a simultaneous manner, the database size increases quadratically.

As it is clear from the statistics above the data generated in today's world is humongous and it makes it a daunting task to extract relevant information from raw data. Data mining techniques help us to retrieve relevant information from big datasets. The algorithms are designed to process raw data, discard useless information and provide information relevant to the problem in study. Depending upon the algorithms data mining techniques are also used to identify patterns in data.

KDD in itself is a distinct branch of computer science. It is a broad term encompassing all the steps starting from cleaning of data, to processing it and finding knowledge as an outcome [10]. The processing of data to identify patterns is the data mining step and is considered to lay at the thresholds of machine learning, statistics and database management. Algorithms and techniques used in data mining are also used in machine learning in a similar or slightly different way. This has made it harder to draw a boundary between the two fields. But it is essential to distinguish the concepts of machine learning from that of data mining for this text.

In the scope of this thesis, the stage where pre-processing of data is done is classified as a knowledge discovery problem and the forecasting of future activities is classified as a separate problem (modeled as a MC or a ML problem). The pre-processing of data means to extract information and get rid of the excess data irrelevant to the problem. This first step can also be called as feature extraction, where the goal is to use some data mining techniques during the KDD process and extract information from the datasets. The outcome of this process is the information which is *features* for the ML methods or nodes for a Markov Chain. Certain techniques such as cluster analysis and neural networks are common to both ML and data mining, hence in order to avoid confusion it is essential to differentiate the terms in this literature review.

The schema of the datasets used for forecasting will be discussed in further detail in the next chapter. But since the thesis deals with forecasting next location of travel the variables common to each dataset are *GPS coordinates* and the *timestamps* associated with them. This raw information especially the coordinates alone do not provide any intelligible information to us. But when looked at together with the time, they can give some insight into the mobility patterns of an individual. For example the time of the day associated with a set of coordinates, time spent at a certain location and number of times an activity can give more information about a set of GPS coordinates. To answer such questions about the mobility of a user, KDD can be used and fortunately this information can be extracted from the minimal amount of data available i.e. the coordinates and the timestamps.

After the data has been pre-processed the answers to the above questions can be extracted using certain data mining algorithms. Given below are the top 10 data mining algorithms listed by IEEE Conference on Data Mining [16].

1. C4.5
2. K-means Clustering
3. Support Vector Machines
4. Apriori Algorithm
5. EM Algorithm
6. Page Rank Algorithm
7. Ada Boost
8. kNN- Clustering
9. Naive Bayes
10. CART

As can be seen from this list, some of these algorithms which are classified as data mining algorithms are already discussed in the machine learning section of this text. So instead reviewing all the algorithms we review those clustering algorithms in detail, that are of interest in the knowledge discovery step.

2.2.1 Cluster Analysis

In statistics cluster analysis is a set of algorithms that are used to classify different objects in groups based on some mathematical criteria. This grouping ensures that the similarity between objects of the same cluster is maximum [17].

Based upon the algorithm in use, cluster analysis can be further sub-divided into many categories. Presenting a literature review for all the variations of clustering techniques and algorithms is beyond the scope of this work. Hence only the most important sub-categories of cluster analysis are listed and explained [18] below. Furthermore, the most common algorithms of each technique are presented.

I. Hierarchical clustering
II. Partitioning clustering
III. Density-based clustering
IV. Grid-based clustering

I. Hierarchical Clustering

This clustering method involves building a hierarchy of clusters. This technique is also referred to as connectivity-based clustering. The [18] outcome of such a clustering is a tree which is referred to as a dendrogram. The dendrogram is not a single set of clusters rather it is a hierarchical connection of clusters at different level. At its most basic level one cluster consists of one object which is connected to another object or a cluster of objects that are closest to it. Figure 2.4 shows a sample dendrogram for a sample set of 10 objects. Hierarchical clustering is applied to it using some similarity criteria, and the objects that are closest to each other according to that criteria are joined with a common node. For example, object 2 and 10 are the closest, approximately 0.1 units away from each other, so they are grouped together (assume) as cluster A. The cluster which is closest to cluster A is the cluster of objects 5, 8 and 9 (cluster B). The distance between cluster A and cluster B is approximately 0.6 units. The advantage of hierarchical clustering is that it is easy to define a cutoff for the clustering criteria.

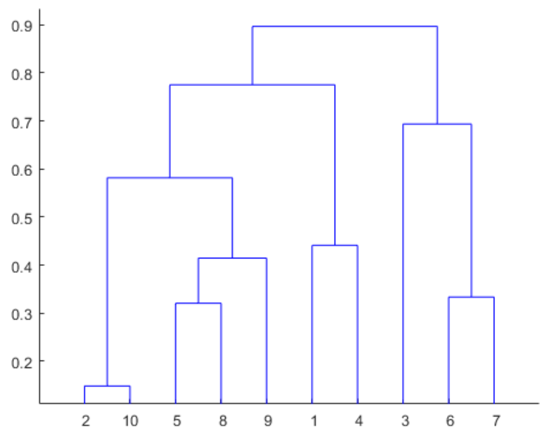


***Figure 2.4** A denrogram for a set of 10 objects [19]*

The approach described above is called the *Agglomerative* clustering. This is a bottom-to-top approach where each object starts off as a cluster in itself. Hence in the start there are as many clusters as the number of objects. Another approach known as *Devisive* clustering is similar to agglomerative clustering in principle but works in the top-to-bottom fashion. So in devisive clustering the algorithm starts with a one big cluster of all objects and these objects are split up into smaller clusters until each object is in its separate cluster.[18]

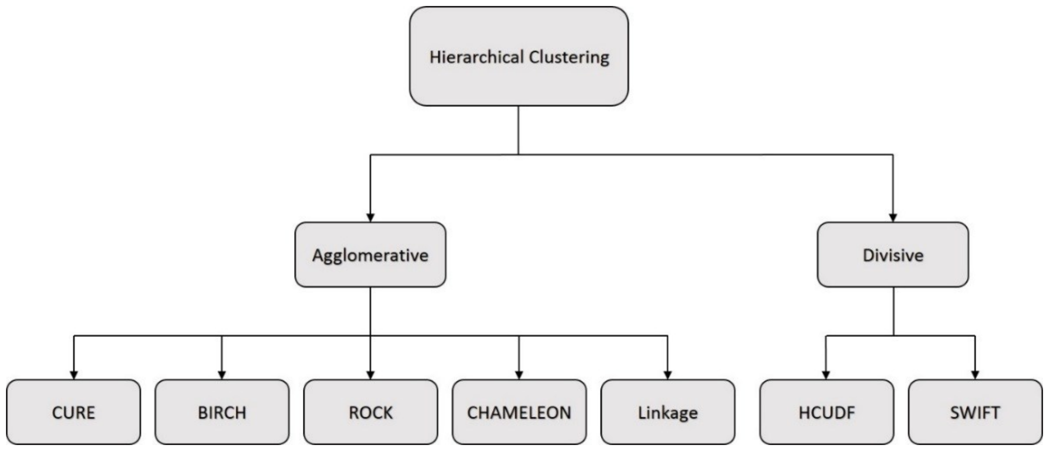


***Figure 2.5** Some hierarchical clustering algorithms [39][41]*

Further details for these algorithms can be found in the text of [18], [20]–[23]

II. Partitioning Clustering

In this clustering technique, a dataset of n objects is split into k clusters where $k \leq n$. This splitting or portioning is done based on some criteria. This ensures that all similar objects become part of a similar cluster. The criteria for portioning is [18]:

1. Each cluster contains at least one object
2. Every object is only present in one cluster

The most common algorithm for partition-based cluster analysis is *k-means* clustering. There are many other algorithms available for partitioning clustering, most of these are variations of k-means clustering. [18]

The variations are mostly in the way the algorithm is initialized. As can be seen from Figure 2.6 [24] the step 2 of the algorithm is initializing of the *k* number of clusters with centroids *c*. This is an iterative step and can be done for all the *n* objects of the dataset. This can however lead to unnecessary complexity and consumption of large computational resources as the number *n* increases. In order to avoid this problem many procedures have been proposed for the initialization of clusters. These can be random selection of points or selecting points based on some criteria.

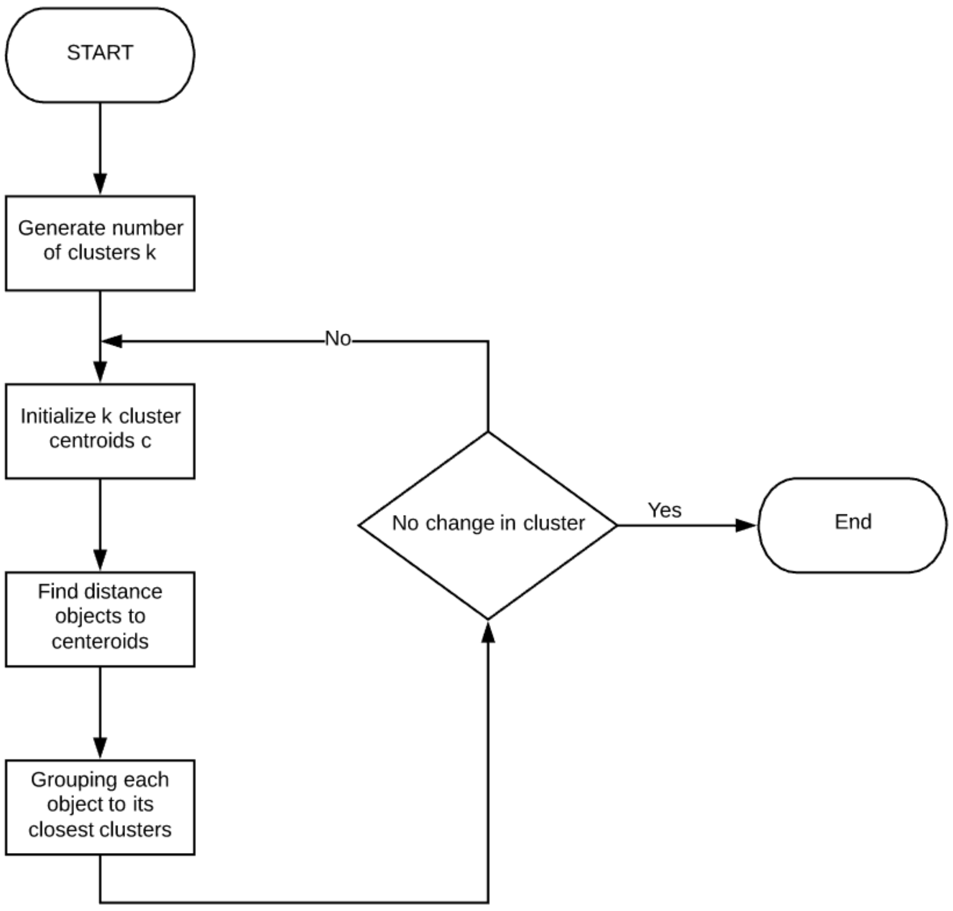

START
Generate number of clusters k
No
Initialize k cluster centroids c
Find distance objects to centeroids
Grouping each object to its closest clusters
No change in cluster
Yes
End

***Figure 2.6** Flowchart of a typical k-means clustering algorithm [24]*

Figure 2.7 shows output of applying *k-means* clustering to image data. The different colors represent different clusters and the X represents the center of each cluster. These centers are the most optimal solution to solving the problem of identifying clusters which result in all points having the closest distance to their respective clusters. This is not a trivial task and requires great amount of computation. The biggest drawback of this technique is that it needs the number of clusters specified as an input to the algorithm.

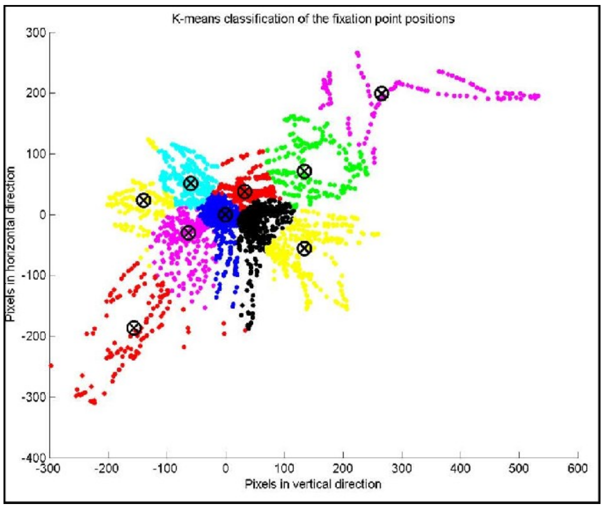


Figure 2.7 *An example of k-means clustering applied for pixel segregation [25]*

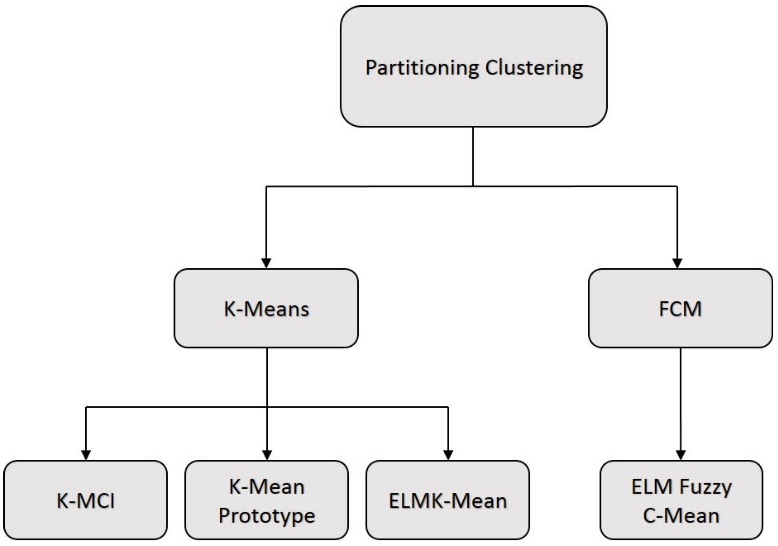


Figure 2.8 *Most common partitioning clustering algorithms [20]*

III. Density-based Clustering

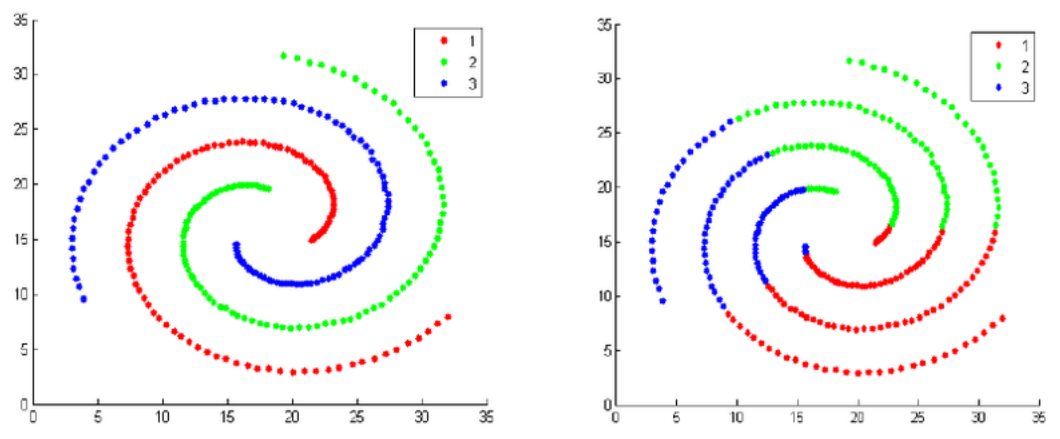
As indicated by the name, this clustering technique is used to form clusters based on density of data points in a region. This is done by finding high density regions and identifying them as clusters and using sparse regions to separate the identified clusters. This means in such clustering the points in the sparse regions are treated as noise and are ignored. [18]

Figure 2.9 shows the pseudocode of the conventional DBSCAN algorithm. It can be seen that it requires three inputs in addition to the dataset. The radius ε is the maximum distance a point can be away from the center of the cluster and still be a part of that cluster. The *minPts* that are required to form a cluster and a *dist* function which is used to calculate the distance between the points. This function can also be called a similarity function.

```
ALGORITHM 1: Pseudocode of Original Sequential DBSCAN Algorithm
Input: $DB$: Database
Input: $\varepsilon$: Radius
Input: $minPts$: Density threshold
Input: $dist$: Distance function
Data: $label$: Point labels, initially $undefined$
foreach $point\ p$ in $database\ DB$ do                         // Iterate over every point
    if $label(p) \neq undefined$ then continue                   // Skip processed points
    Neighbors $N \leftarrow$ RangeQuery($DB, dist, p, \varepsilon$)   // Find initial neighbors
    if $|N| < minPts$ then                                      // Non-core points are noise
        label($p$) $\leftarrow$ Noise
        continue
    $c \leftarrow$ next cluster label                            // Start a new cluster
    label($p$) $\leftarrow$ c
    Seed set $S \leftarrow N \setminus \{p\}$                   // Expand neighborhood
    foreach $q$ in $S$ do
        if $label(q) = Noise$ then $label(q) \leftarrow c$
        if $label(q) \neq undefined$ then continue
        Neighbors $N \leftarrow$ RangeQuery($DB, dist, q, \varepsilon$)
        $label(q) \leftarrow c$
        if $|N| < minPts$ then continue                         // Core-point check
        $S \leftarrow S \cup N$
```

Figure 2.9 *Pseudocode for DBSCAN algorithm [26]*

The DBSCAN algorithm is good in identifying dense clusters and drawing boundaries between them. As can be seen from Figure 2.10(a) , this algorithm is very good at finding patterns or clusters that are arbitrarily shaped. This is due the reason that the algorithm sniffs for high density regions and follows them. This leads to the result that all the objects that are connected are sorted into one cluster. This is different from other algorithms like *k-means*, where the distance of each point is calculated from a mean point, which can lead to inaccurate results; demonstrated in Figure 2.10 (b).



(a) DBSCAN applied to a sample dataset (b) k-means clustering result for the same dataset

***Figure 2.10** Variations in results based on clustering technique used [27]*

There are many variation of density-based clustering algorithms available. The basic idea of finding dense regions remains the same in all these algorithms but the procedure adopted to do so varies from one algorithm to another. BRIDGE, for example divides the dataset into pre-clusters and then applies DBSCAN to those clusters [18]. But the other algorithms mentioned in Figure 2.11 use different techniques for forming the clusters. Further detail of each of these algorithms can be found in [18], [20], [23].

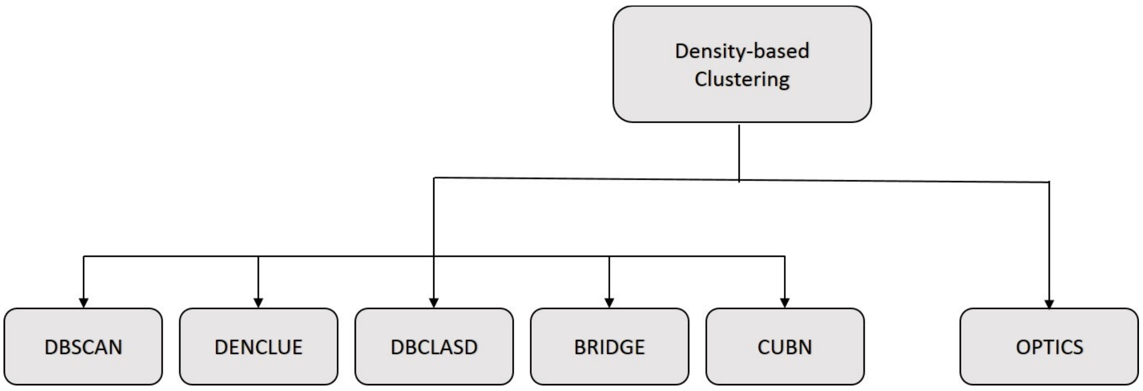


***Figure 2.11** Types of density-based clustering algorithms [18], [20]*

IV. Grid-based Clustering

These algorithms utilize a grid data structure which has a multi-resolution. The outcome is number of cells which are finite in number. This *grid* structure is then applied with the clustering techniques [21]. This quantization gives the advantage of fast processing and improves performance times, and utilizes fewer computation resources. The basic idea is to make the computation of the algorithm independent of the number of objects in the dataset. The grid dimensions are fixed at the start of the algorithm. This gives the advantage of fixing the number of points to be processed along with the option to choose grid size based on the accuracy that is required. This technique is particularly useful if a dataset is multi-dimensional and standard distance functions cannot be applied as intended.

The main focus is on how to identify clusters in multi-dimensional sparse datasets where clustering in all dimensions is not achieved but many sub-clusters are present. Some common grid-based clustering algorithms are given in Figure 2.12. SUBCLUS (density-connected Subspace Clustering) is one of many grid-based clustering algorithms. As shown in Figure 2.13, this algorithm uses the density-based connectivity approach of DBSCAN but it is adapted for datasets with many dimensions. The difference is that this algorithm finds the sub-clusters in the sub-spaces and can thus help greatly with high-dimension sparse datasets [28]. Further literature with details of these algorithms can be found in [18], [21], [23], [28].

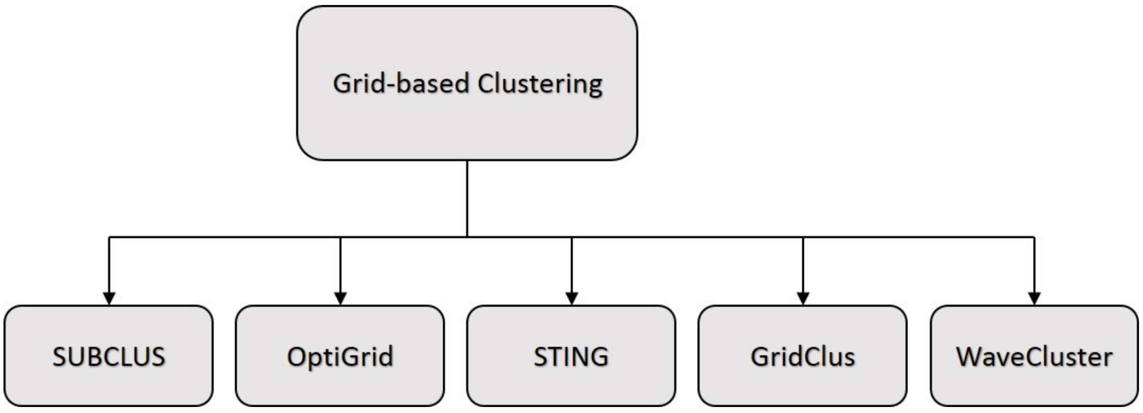
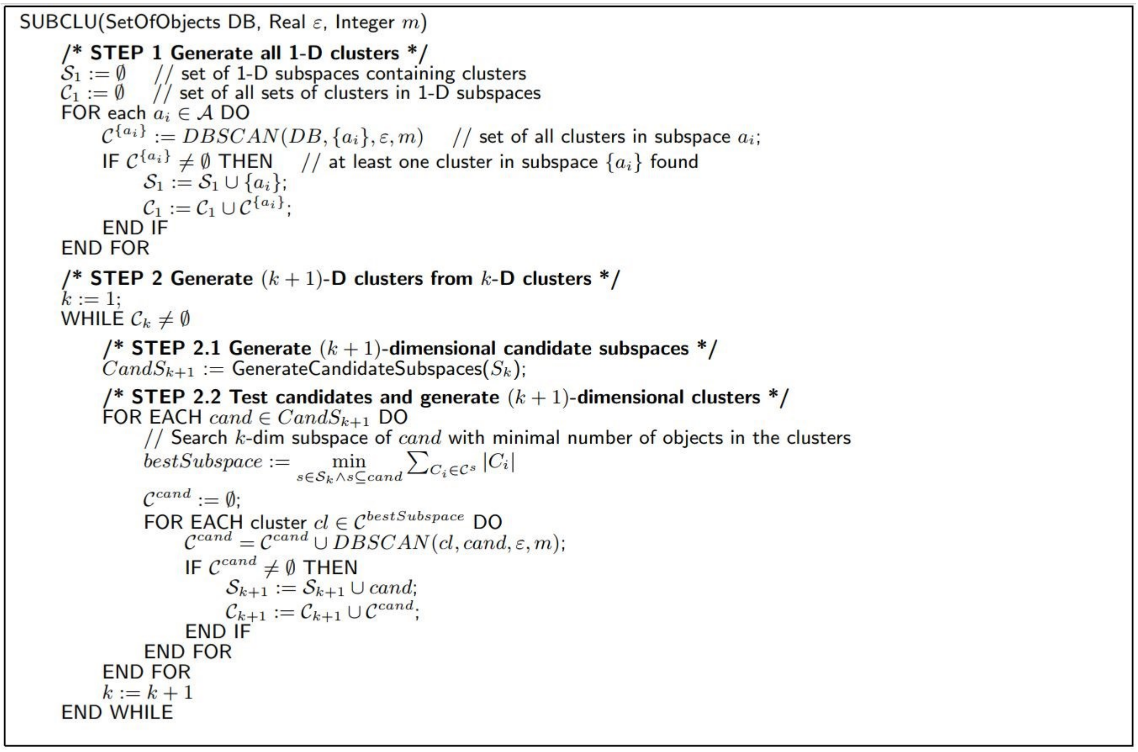


Figure 2.12 *Grid-based clustering algorithms*



```
SUBCLU(SetOfObjects DB, Real $\varepsilon$, Integer $m$)
    /* STEP 1 Generate all 1-D clusters */
    $\mathcal{S}_1 := \emptyset$    // set of 1-D subspaces containing clusters
    $\mathcal{C}_1 := \emptyset$    // set of all sets of clusters in 1-D subspaces
    FOR each $a_i \in \mathcal{A}$ DO
        $\mathcal{C}^{\{a_i\}} := DBSCAN(DB, \{a_i\}, \varepsilon, m)$    // set of all clusters in subspace $a_i$;
        IF $\mathcal{C}^{\{a_i\}} \neq \emptyset$ THEN    // at least one cluster in subspace $\{a_i\}$ found
            $\mathcal{S}_1 := \mathcal{S}_1 \cup \{a_i\}$;
            $\mathcal{C}_1 := \mathcal{C}_1 \cup \mathcal{C}^{\{a_i\}}$;
        END IF
    END FOR
    /* STEP 2 Generate $(k+1)$-D clusters from $k$-D clusters */
    $k := 1$;
    WHILE $\mathcal{C}_k \neq \emptyset$
        /* STEP 2.1 Generate $(k+1)$-dimensional candidate subspaces */
        $CandS_{k+1}$ := GenerateCandidateSubspaces($\mathcal{S}_k$);
        /* STEP 2.2 Test candidates and generate $(k+1)$-dimensional clusters */
        FOR EACH $cand \in CandS_{k+1}$ DO
            // Search $k$-dim subspace of $cand$ with minimal number of objects in the clusters
            $bestSubspace := \min_{s \in \mathcal{S}_k \wedge s \subseteq cand} \sum_{C_i \in \mathcal{C}^s} |C_i|$
            $\mathcal{C}^{cand} := \emptyset$;
            FOR EACH cluster $cl \in \mathcal{C}^{bestSubspace}$ DO
                $\mathcal{C}^{cand} = \mathcal{C}^{cand} \cup DBSCAN(cl, cand, \varepsilon, m)$;
                IF $\mathcal{C}^{cand} \neq \emptyset$ THEN
                    $\mathcal{S}_{k+1} := \mathcal{S}_{k+1} \cup cand$;
                    $\mathcal{C}_{k+1} := \mathcal{C}_{k+1} \cup \mathcal{C}^{cand}$;
                END IF
            END FOR
        END FOR
        $k := k + 1$
    END WHILE
```

Figure 2.13 SUBCLUS Algorithm ***[28]***

2.2.2 Point of Interest Extraction

As discussed in earlier sections the modelling step of activity forecasting is preceded by extracting points of interest (POI) from the mobility data of a person. These POI can then be used as a parameter to a ML model or they can be used to construct Markov chains. The first step for extracting a point of interest is to find the stay points from the data.

Stay Point Extraction

Stay points are the points where a person spends some time during the day. Spending time at a stay point implies that this specific point is of some interest to a person. The mobility data from which these stay points are extracted, is a time-series data the number of mobility points collected can be huge. Differentiating points where an individual spent their time from other numerous passing points can be a challenging task. The nature of data (timestamps and coordinates) available for analysis is also limited increasing the level of challenge. For this purpose researchers have used different KDD methods to extract stay points.

Clustering techniques ranging from hierarchical clustering to density-based clustering and many more are used with various success rate. Work of [2] gives a good overview of how different clustering techniques affect the outcome of stay points. Their work applies different variations of graph and grid-based clustering algorithms to identify the stay points. Hierarchical clustering is good for visualizing the spread of data and understanding the connectivity of different points but it is affected by the 3-dimensional nature of the data. Partitioning clustering faces the challenge that it requires the number of clusters as an input and since one user's stay points can be different in size and number from another user. Hence it is not practical to use partitioning clustering to solve this problem.

Most of the researchers have tried to avoid this problem by forming their own versions of density-based or grid-based algorithms. They first define the term "stay point" this helps in defining the distance functions for a cluster. One such example can be seen in [29], where the researchers have defined and constrained the term stay point by a set of conditions. According to the authors a stay point is a geographical region where a person spends a certain amount of time. They define each stay point as dependent on two threshold values, time and distance.

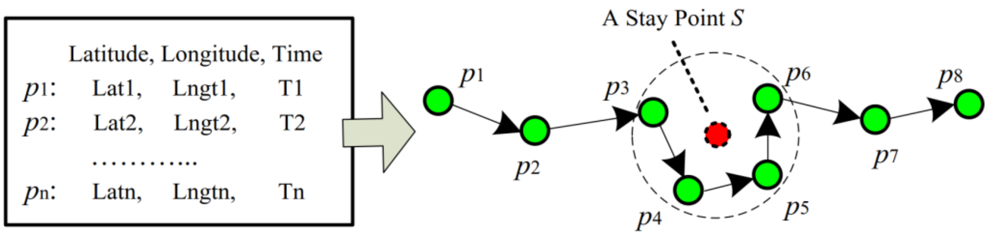


Figure 2.14 *A visual representation of a stay point [29]*

As can be seen from points p_3 to p_6 which are a group of consecutive GPS points are characterized as a stay point *S*. Hence a stay point according to [29] is a collection of consecutive GPS points *P* (where P = $\{p_m, p_{m+1}, \ldots, p_n\}$), where $\vee\ m < i \leq n$, Distance $(p_m, p_i) \leq D_{\text{threshold}}$ and $|p_n.\mathrm{T} - p_m.\mathrm{T}\ | \geq \mathrm{T}_{\text{threshold}}$. The point *S* can be defined as $S = (Lat, Lng, T_{arv}, T_{dep})$, where *Lat* and *Lng* is the average of all the points from *m* to *n* and T_{arv} and T_{dep} represent the times of arrival and departure to the point *S*.

This method of defining and extracting a stay point has been reused by many other researchers with their own variations. The same technique was found in the works of [30]–[32]. All these works focus on extracting stay points from a sequence of GPS coordinates and timestamps. The main difference is deciding the threshold values for time and distance. These two parameters have a significant impact on the outcome but unfortunately they can only be found by a trial and error approach [30]. Furthermore, one threshold value once fixed will also not be suitable for different user or a different situation, but this is a topic for later discussion.

After extracting the stay points any of the standard clustering techniques can then be used to group all the identified stay points as activity clusters. The most similar stay points are grouped in one activity cluster. DBSCAN has been successfully used by some researchers in [31]. It shows promising results but due to the 3-dimensional data, it can be struck by inaccuracies specially since all the three or two of the three dimensions are unrelated to each other. In most cases clustering the activities is not as much of a challenge because for every user there are a limited number of stay points found.

2.3 Modelling Human Mobility Patterns

Building up or finding a mathematical model for human mobility is the most critical task. As in the case of any other problem, the accuracy of the analysis or prediction of human mobility heavily depends on the way the problem is modeled. An accurate model can lead to superior results but such models can be complex and computationally difficult to build, while on the other hand a simple model will be more practical but compromise the accuracy of the results. This limitation should be kept in mind before building up a model for any problem.

2.3.1 Markov Chains

Exploiting the Markovian nature of the problem is an intuitive modelling approach due to the problems' *memoryless* nature. The idea is to first identify all the POIs, this is done by clustering together the same locations, visited from the data; we can call this step as data-mining. These POIs extracted from a collection of mobility traces can be treated as states of a Markov chain.

Markov chains first appeared in a publication of a Russian mathematician Andrei Andreevich Markov in 1906 [33]. A process is defined as a Markov process when, given the *r*-dimensional normal density of random variables *$(X_1, X_2, \dots, X_r)$* ***if*** for, $k \leq r$ the conditional density of X_k for given $X_1, X_2, \dots, X_{k-1}$ is identical with the conditional density of X_k for givenX_{k-1}. The POIs are a finite set, and such a *finite-state* Markov processes is also called Markov chains. In simpler words a Markov chain is a stochastic process whose conditional probability function satisfies the *memoryless* property [34, p. 194]. The term *memoryless* means that at any time instant all the future states of the system are only dependent on the current state and are independent of all the previous (or historical states). Mathematically a system can be considered *memoryless* given the random variables $X_1, X_2, \dots, X_r$ **if and only if**

I. $E(X_k \mid X_1, X_2, \dots, X_{k-1}) = E(X_k \mid X_{k-1})$ for all $k \leq r$ …(i)
II. $\rho_{jk} = \rho_{j,k-1}\rho_{k-1,k}$ for $j \leq v \leq k \leq r$, where ρ_{jk} is the correlation coefficient for bivariate random variables …(ii)

Equation (*i*) uses the conditional probability [35, p. 97]. It states that the probability of moving to the state *k,* given that the system previously went through k-*1* states where *k-1th* state is the latest state, is the same as the probability of moving to the next state *k,* given that the system is in the state *k-1*. This in essence describes a *memoryless system* where all the previous states, except the most recent state, can be ignored since they do not provide any additional information in determining the next state of the system.

Figure 2.15 is an example of a memoryless system, thus a simple Markov chain. The system has three *states* A, B and C. The arrows represent *state transitions* and the number adjacent to each state is the *probability* of transitioning from one state to another. It can be seen that the sum of all possible transition probabilities is equal to 1 and it is also possible to end up in the same state. The system is *memoryless* because every *next* transition is only dependent on the current state and the transition probabilities associated with that state. The next state transition is independent of the starting time, the starting state and the number of previous state transitions.

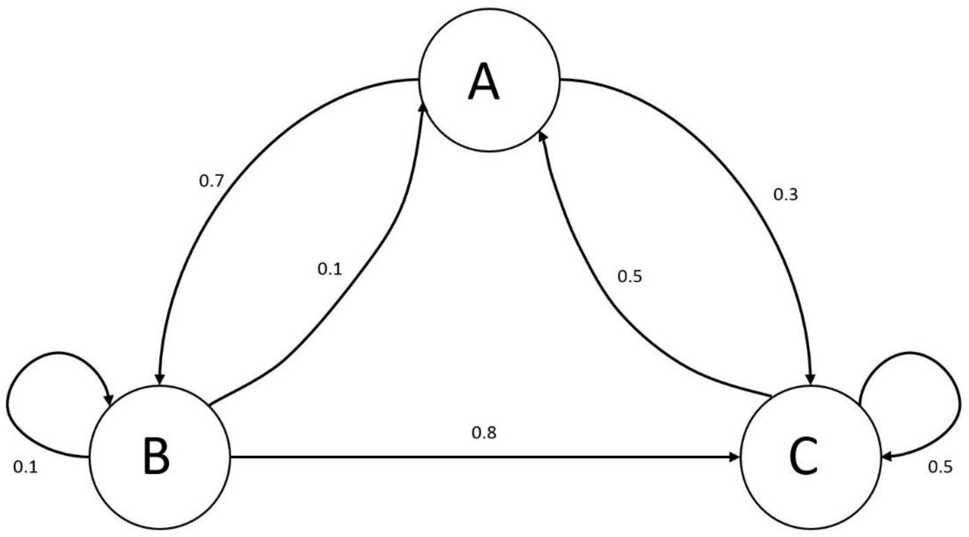


***Figure 2.15** A Markov Chain model*

Hence to form a simple Markov chain three parameters need to be defined. These parameters are the *states* of the system, the *transition probabilities* and the starting point or the *initial state*. These three parameters can be used to calculate the next *immediate state* or *n-future states* of the system with a certain degree of confidence based on the calculated probability values. In the case of mobility traces, the previous data can be used to formulate similar Markov chains for every user.

From the available data the POIs are extracted. These POIs will be the *states* of the model, similarly *transition probabilities* of going from one state to another are calculated, this can be thought of as training of the model. Now having the complete Markov model we can calculate the probabilities of ending up in any state (a POI) and use it to predict the next point of interest of the user. This approach is very intuitive because the human mobility can be thought of as a Markov process as people normally have only few frequented locations (POIs), making the transitions discrete and the transition to the next state is independent of the previous state. Since most of the times information about previous locations are of little or no use in predicting the next POI. Take the example of a university teacher if the teacher leaves for the university every day at 7:00 am then given the history, it can be said that the most probable point of interest for prediction on Thursday 7:00 am is the university. It is irrelevant where the teacher was at last night.

Markov chains can be classified into different types based upon how a problem is translated to a Markov problem and vice versa. The simple Markov chain of figure 1 is the most basic form and also referred to as Markov-Chain Model. Some complications may be present in this model but the basic requirement for a model to be Markov-Chain Model is that all the states are directly observable. Hidden Markov Model(HMM) is another type of Markov model where the system is not directly observable[35, p. 256]. In such models indirect observation of a state is possible by observing some other random variable which is then used to estimate the state of the system. Figure 2.16 is an example of a HMM; the states are represented as random variables $X_0, X_1, X_{2,}X_3$ and these states are not directly observable. The state can be known by making observations on the system e.g. if Y_2 is observed it is known then that the system is in stateX_2.

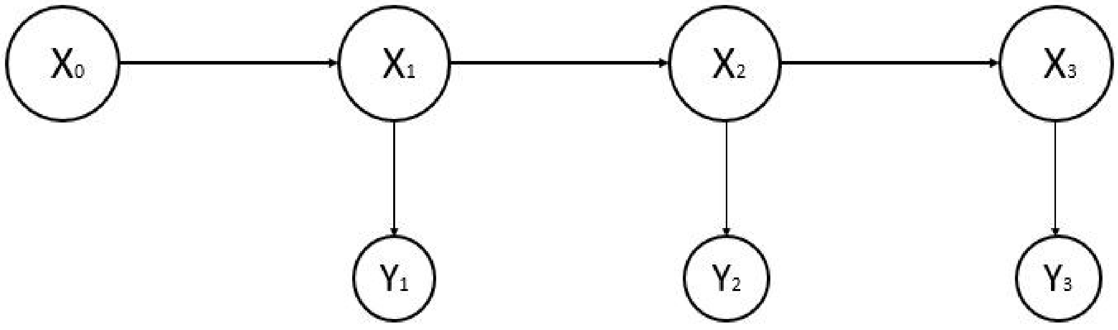


Figure 2.16 *A Hidden Markov chain model*

Another variation of the Markov model, which in essence is an extension of the HMM, is the Mixed Markov model (MMM). This model has an unobservable parameter similar to a HMM but the unobservable parameter is fixed at the time of transition. This means that unlike a HMM where the next transition is affected by only a specific observed state, in a MMM the next state can be affected by a *combination* or *mixture* of observed states. A MMM such as one presented in Figure 2.17 can better model a process since there is some flexibility in which model generates the state transition. Hence it gives more flexibility at the cost of greater complexity in modeling a process.[8]

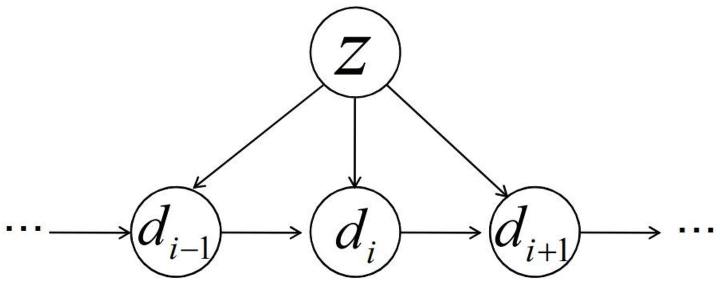


Figure 2.17 *Mixed Markov Chain Model [8]*

These Markov models have been analyzed and tested to find the best suited model to solve the problem. The work of [6], [31], [36] uses the most basic Markov-Chain model. Hidden Markov Model are used in [7] and Mixed Markov Model are implemented in [8]. As clustering of POIs is the first step for using a Markovian approach the results can vary with type of clustering technique and assumptions about the data set. Although Markov Models perform satisfactorily for available data sets but they will fail to predict the next point of interest if it is the first time a user visits the location. This, combined with the fact that huge amounts of data preparation or data mining is required on datasets brings a certain amount of disadvantage to these predictors. Overall such models are mathematically simple but discretization of states and transitions can be challenging in cases where discrete boundaries are not easily identifiable. This can greatly affect accuracy and results of the model.

2.3.2 Machine Learning

Another approach to analyze the human mobility which has been presented by researches can broadly be termed as *Machine Learning* (ML). ML compared to Markov chains [33] is a relatively new field. The term *machine learning* was coined by Arthur Samuel in 1959 [37]. He defined machine learning as *the study of making the computers learn without explicitly programming them*. The term machine learning encompasses a broad range of mathematical methods and computing techniques but all of these methods and techniques have their basis in statistical theory [38]. In general ML helps in building mathematical models without any prior knowledge of a system or process for which the model is constructed. This is a data driven approach where available data is used to generate models. The accuracy of such models is highly dependent on the amount and quality of the data. The ML based models can range from very simple to very complex based on type of technique used in building that model. Some of the most common machine learning techniques which are relevant to this thesis are introduced below.

I. Decision Tree Learning

A decision tree can most simply be defined as an acyclic graph [39]. It is represented graphically and has a tree like structure. The starting point which is also known as the *root* is always fixed. It consists of *nodes* and *edges*, where each node describes an attribute while the edges are the decisions based on each attribute. Figure 2.18 shows a simple decision tree. In this tree the first node which is blue in color is the root. Each nodes corresponds to an input variable value whereas the arrows are decisions or observations. The last node also known as the *leaf node* is the target value given that the input variables have the values in the path from the root to that specific leaf.

As can be seen from Figure 2.18 the risk of having a heart attack is predicted with a simple decision tree. There are two input variables *age* and *medical history*. At each node a decision is taken based on the value of an input variable to arrive at the *leaf node*. The training of this classifier is mostly done in a recursive manner and the tree is itself constructed by splitting data at each splitting step. The attribute or variable to split can be selected in a number of ways and to achieve certain criteria such as accuracy or tree depth or a combination of both. [40]

Decision tree based learning is a form of supervised learning. Conceptually they are very intuitive, training is not complicated, meaning that they are easy to troubleshoot and the classification is very fast [41]. But these work well with binary classification and can run into problems such as overfitting if the depth of a tree increases too much.

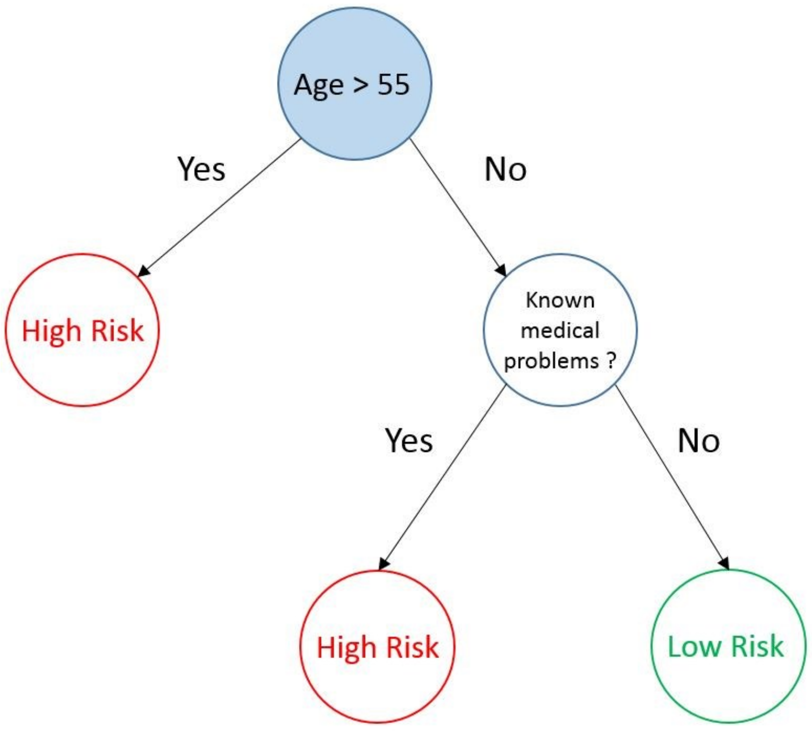


Figure 2.18 *Decision Tree*

II. Bayesian Networks

Another type of graphical model this type of classification is based on the Bayes' formula and uses probability theory at its core. Also known as the *directed graphical model* [42].

Such networks are based on cyclic graphs which are constructed from the probability density functions of variables.

Figure 2.19 shows a simple directed graphical model. Let the joint probability of the three variables *a*, *b* and *c* be given by equation (2-1). For each condition distribution arrows can be drawn connected the nodes [42].

$$P(a,b,c) = P(c|a,b)P(b|a)P(a) \qquad \text{(2-1)[42]}$$

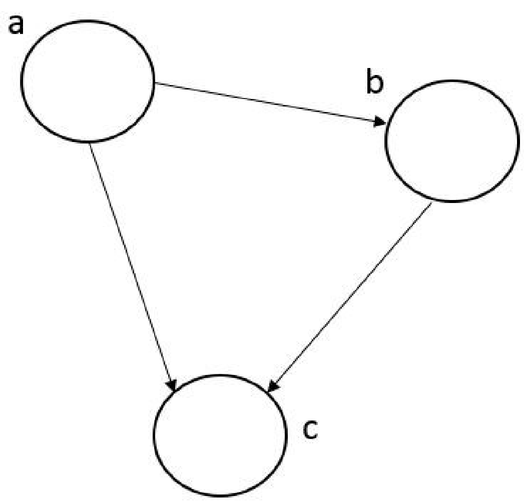


Figure 2.19 *A Directed Graphical Model*

In such networks the probability of being in a certain node can be obtained if the probabilities of the variables are known and their joint probabilities are calculable. The training of the model gives the possibility of directly finding the probabilities of the variables which in the case of an ML problem is the input variables. The joint probabilities can be estimated from the data using the maximum likelihood approach [43].

III. Cluster Analysis

As indicated by its name this technique involves partitioning data into groups or segments [44]. This is a form of unsupervised learning where no target outputs are known but the data is only clustered based on the input variables. Figure 2.20 shows an example of clustering; where data is plotted against two variables and then plotted data is split into two segments based on some cost function.

There are different algorithm developed for clustering of data. Two most common ones are *k-Means clustering* and *hierarchical clustering*. The difference between these two clustering techniques is that in k-Means the data is split such that the distance from the centroid (mean) of each cluster to the data points in that cluster is minimized [45]. While in hierarchical clustering the clustering is done at each layer and distance from one data point to its closest data point are calculated. In the end a tree or dendrogram is formed [46].

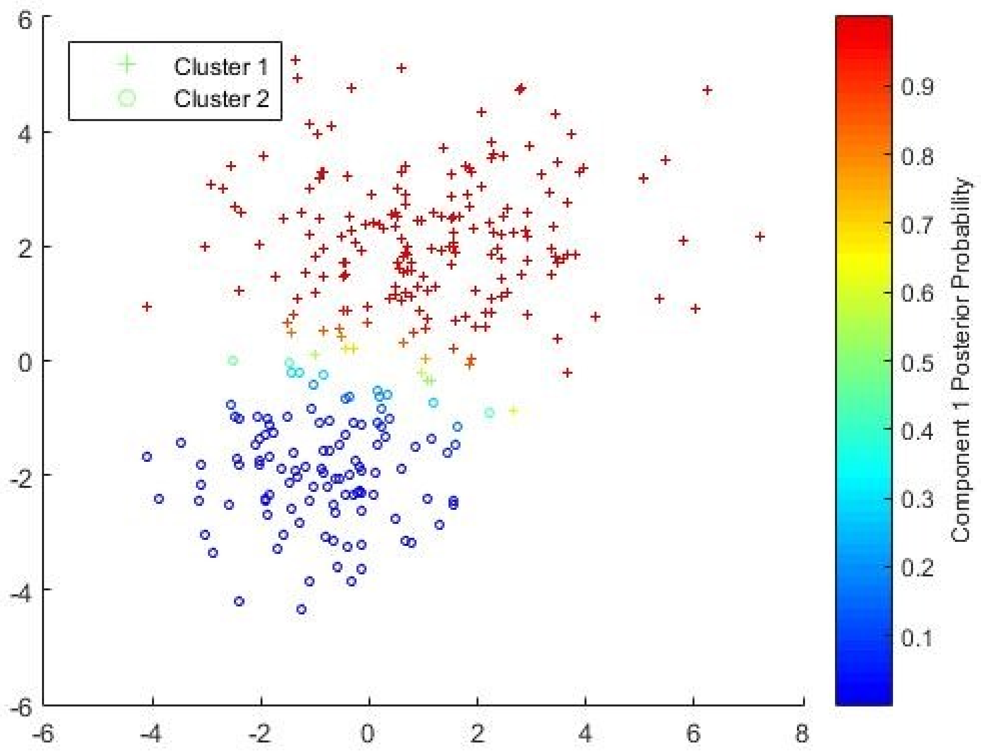


Figure 2.20 *Clustering of data [44]*

Clustering is also commonly used in data mining problems. In machine learning application its use is mostly restricted to unsupervised learning. But clustering can also be used as an initial step in a supervised learning problem.

IV. Artificial Neural Networks (ANN)

Artificial neural networks were inspired by the neural networks. The work on ANNs started in 1940s when a learning hypothesis, also known as Hebbian Learning, was developed by D.O. Hebb [47]. The basic idea of an ANN is to mimic the function of human brain, which is a combination of billions of neurons [48]. But for each action or reaction that humans display only a limited number of neurons are stimulated. Thus every neuron in the brain is trained to be stimulated for a specific input.

In the similar manner an ANN can contain numerous neurons but they are trained in such a manner so that for every input to the network the neurons react differently. The training of an ANN is responsible for each neuron to output a certain value based on the input it receives. When training an ANN, data is provided to it and with the help of a cost function the training is done such that each neuron has a weight associated to it. By many training epochs these weights are tuned to provide a target output value.

Figure 2.21 shows an ANN which has 4 input variables. Each of these variables is fed into the neurons of the hidden layers. There can be many hidden layers based on the complexity of the network. The outputs of the final hidden layer is summed up and given as the output layer. The output layer can have more than one neurons, this number is equal to the number of classes.

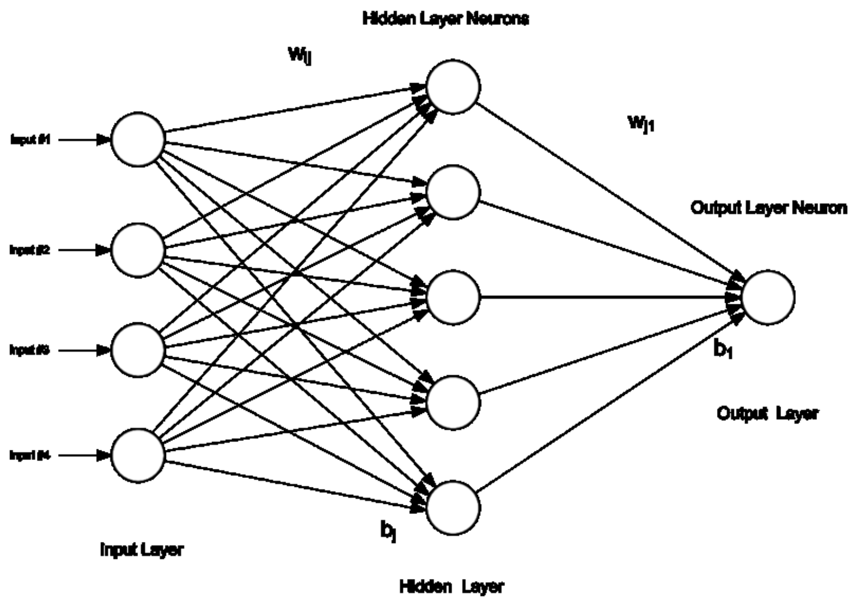


Figure 2.21 *An Artificial Neural Network [49]*

V. Adaptive Neuro Fuzzy Inference System (ANFIS)

An adaptive neuro fuzzy inference system is an extension of a neural network but since it combines concepts of fuzzy reasoning, this technique deserves a separate explanation. Contrary to binary logic fuzzy logic gives the possibility to have many values between 0 and 1. This gives fuzzy reasoning the strength of having many levels of truth and false e.g. there is a possibility to have partial truths where the value of the truth can be any value between entirely true and entirely false [50]. There are three steps involved in fuzzy reasoning [50].

1. The first step is to fuzzify the input variables. This is done by converting the variables into membership functions.
2. The second step is to use the fuzzified inputs and with the help of predefined rules obtain the fuzzy outputs.
3. The final step is to de-fuzzify the fuzzy output functions to get *crisp* output values.

The above concept of fuzzy reasoning when combined with ANNs give rise to ANFIS. In such systems the three steps of fuzzy reasoning are followed but the neural networks are introduced in step 2. Generally fuzzification of input variables can be done using fixed membership functions and a rule-base but in ANFIS step 2 is altered so that the rules base is adaptive to the output variables. The training step in ANFIS is used to fit the rules so that the desired training outputs can be obtained. It can be said that instead of tuning the weights of a neural network in ANFIS the tuning of rules takes place such that an error threshold for the training output can be obtained.

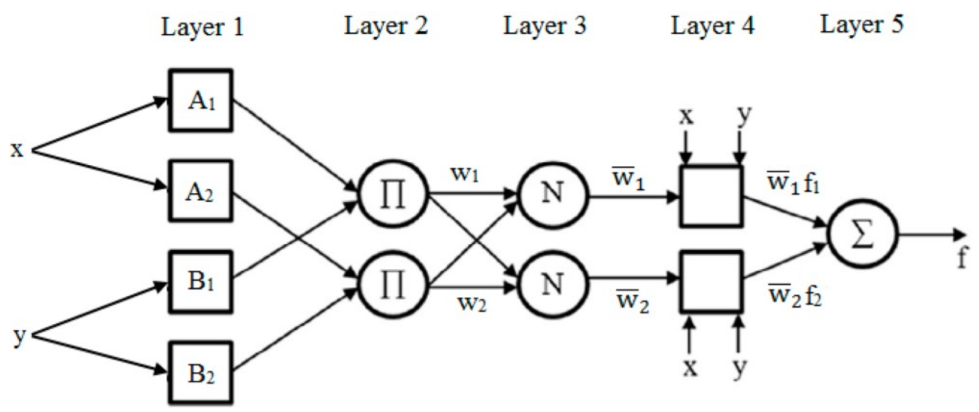


Figure 2.22 *A typical ANFIS structure for two inputs, two rules and one output* ***[51]***

Figure 2.22 shows an ANIFS structure. The input variables are *x* and *y*, the output *f* is given for a single class problem and can be mathematically be calculated from equation (2-2) [51].

$$f = \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \qquad \text{(2-2)}$$

The de-fuzzification of the output function as represented in equation (2-2) is done by the Takagi-Sugeno-Kang method and can be better explained by Figure 2.23.

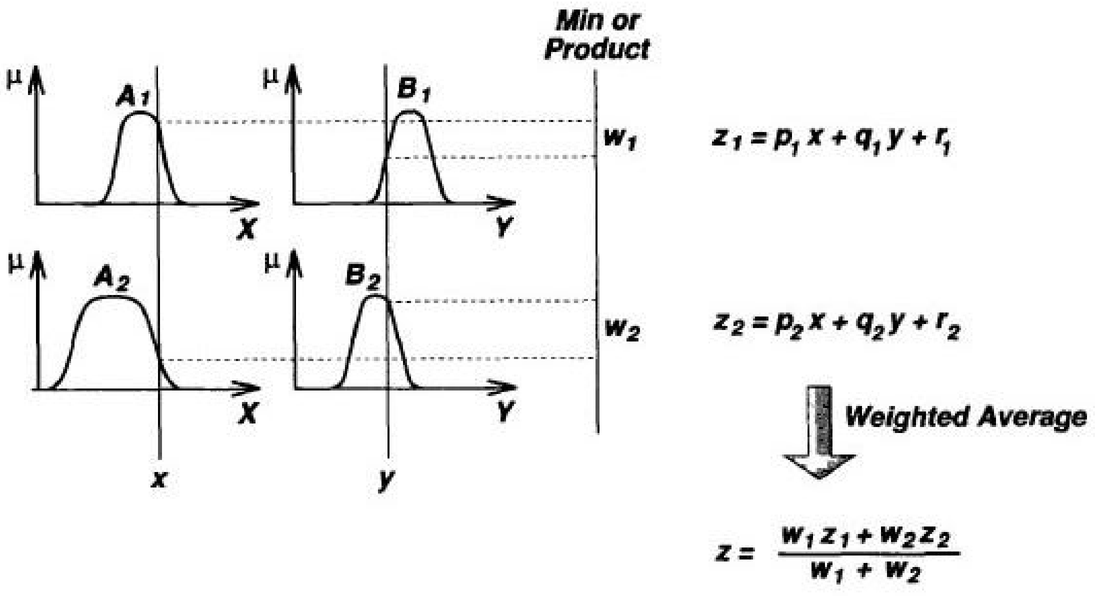

Min or Product
μ
A1
μ
B1
w1
$z_1 = p_1 x + q_1 y + r_1$
X
Y
μ
A2
μ
B2
w2
$z_2 = p_2 x + q_2 y + r_2$
X
Y
Weighted Average
x
y
$z = \frac{w_1 z_1 + w_2 z_2}{w_1 + w_2}$

Figure 2.23 *1st Order Sugeno Fuzzy Model [52]*

Compared to Markov-chains based approach, not too much work has been done where classical machine learning techniques are employed to solve the problem of predicting the next point of interest of a user. Some studies have taken a machine learning based approach to identify patterns in user mobility but they tend to focus on other application areas like finding people with similar interests [53]. The most extensive work done in this domain that is by [54]. In their research the authors have developed and applied machine learning based solution to the problem of predicting the next place a user will visit based on instantaneous information and previous trajectory data. They were the winning team in Nokia's Mobile Data Challenge for the Next-Place Prediction task. Their research shows application of various machine learning techniques like decision trees, Bayesian networks and Neural networks to solve the problem and then also suggested possibility of making hybrid-predictors. Majority of the researchers have used the hybrid approach where they combine different machine learning techniques with Markov chains to obtain next place predictors. Works of [2], [9], [53] show this approach.

2.4 Semantic Information

As a final outcome each point of interest can be defined as a tuple P (*coordinates*, T_{arv}, T_{dep}). This is the least amount of information required to define a POI. In order to improve the results the researchers have tried to add semantic information to the extracted POI. In most cases the motivation to move from one place to another is always inspired by the context. In [55] the researchers present the problem of location prediction separately from location recommendation no matter how similar both seem. They categorize the movement of people into three categories. Geographic-triggered Intentions (GI), Temporal-triggered Intentions (TI) and Semantic-triggered Intentions (SI). As the name indicates, GI intentions are those where a user's next point of interest is dictated by his/her current location only, for example if a user is travelling in a train their next predicted location will be the next train station on the way. In Temporal-triggered Intentions the user's activity or mobility is dependent on the time of the day; for example on a weekday evening, a person is more likely to go home. The third type, SI are the intentions that are actually physical geographical locations but the cause for visiting those locations can be better explained as semantic information. For example, if a person leaves their office every day at noon for lunch and go to a different restaurant. By observing the data we can know in advance what time the user is most likely to leave the office but we cannot be certain about the activity that person undertakes after leaving the office. This may result in an inaccurate prediction or recommendation. But if we can find semantic information about their place of visit, which in this example would be to that the user visits a restaurant at noon, we can make a better prediction of the next POI.

The earlier discussed Markov Models were mostly GI based with different ways of clustering but attempts have been made to add more dimensions to these models, in [31] researchers tried to combine GI and TI in their predictors and reported better performance.

Similarly, [55] and [32] tried to combine all three categorizes in their spatiotemporal-geographic predictors. [55] claimed to outperform existing state-of-the-art predictors, whereas [31] focused on understanding and categorizing user types based on the data which can be used for better and targeted recommendations. It should be mentioned here that [55] used a tree based approach to find similarity between a user's own previous trajectories and/or others users trajectories to predict the next location with maximum probability but they had to first carry out extensive data-mining to prepare the data. It can be easily concluded that the more intentions (GI, TI, and SI) we add to the system the more data-mining or preparation is required.

2.4.1 Activity Recognition

Obtaining this semantic information is not a trivial task. Semantic information can be anything ranging from the emotional state of the person to the type of location visited. Since we are interested in the activities a user performs so we limit the semantic information to the nature of activity a user performs. Knowing the activity type provides relevant semantic information. In most cases this semantic information is correlated to the geographic information as well. Hence by knowing the semantic information we can cover both GI and SI triggered movements. Apart from that having this semantic information can widen the range of recommendations that can be provided to a user. This is a very important use, since today's technology is very focused on customizing user experience.

Once again there are a huge number of possibilities available on how to recognize a person's activity. The research in the activity recognition domain is primarily focuses on data gathered from sensors that can closely monitor a human being. This data can be gathered from wearables or from smart phone accelerometers. This sort of activity recognition shows promising results. But when we try to identify activities from timestamped GPS coordinates it can pose many challenges. The greatest challenge is that a person can perform activity without any movement or with minimum movement. Such activities are impossible to detect by monitoring GPS data. So in this text the focus in only on activities that involve moving from one place to another. This makes the task somewhat easier since the focus is to identify only the activities performed at the extracted POI.

The most common approach used for all type of activity recognitions is based on ML techniques. Activity recognition from GPS coordinates only is not done by many researchers. The most similar work is done in [2] where the researchers have used clustering on GSM tower based location data. Most researchers working with activity recognition and GPS data also use other data sources to obtain better estimates. A brief introduction of ML techniques has already been provided. In the coming chapters same techniques will be demonstrated and used for activity recognition and activity forecasting.

3. APPROACH

As discussed in chapter two, there are two main approaches used to solve the problem of forecasting user activities. First one is the classical approach of Markov Chains and the second one is the relatively new data-driven approach of Machine Learning. In the initial phase both of these techniques are employed to identify the most suitable method for activity forecasting. This section details the approach employed to solve the problem.

3.1 Point of Interest Extraction

The initial step for the above-mentioned techniques is the datamining. It is used to extract certain information or features from the datasets. The most important information is the point of interests of a user.

While logging GPS data we receive hundreds of thousands of traces of a user. Finding the points where a user actually spends their time can be a challenging task. Especially with time series data where there is no segregation of different trajectories. In order to overcome this problem, first the traces are split into trajectories. This is done based on some threshold value of time. So for example if two consecutive traces have a time difference of a few hours then clearly they represent two different trajectories.

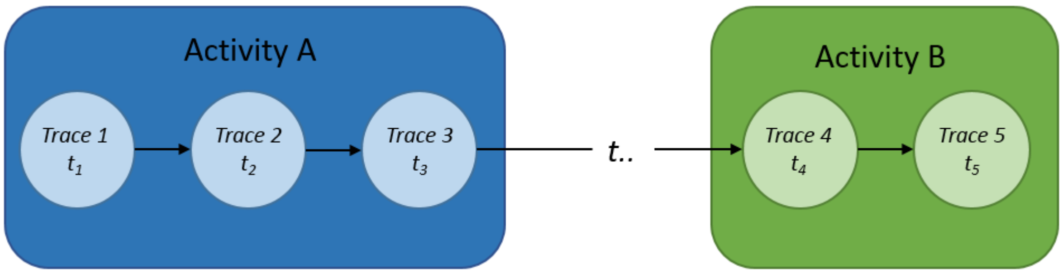


***Figure 3.1** Splitting traces into trajectories.*

Figure 3.1 shows an example of 5 consecutive traces of a user. It can be seen that there is a significant time difference between t_3 and t_4. Even though the data is continuous, this implies that either the user spent a significant amount of time at point 3 or there was a break in data collection. In both the cases *trace 3* marks the end of one trip or trajectory and *trace 4* the start of a new one. The time threshold to split trajectories can be set to any desired value and is calculated as $t_4 – t_3$. This step helps us to reduce the number of processing points while calculating the points of interest.

To extract points of interests from the trajectories the most common approach is to find clusters within the dataset. There are many clustering techniques available and these are presented in chapter 2 of this text. For finding points of interest the approach used is presented here. Figure 2.14represents a series of consecutive points p_1, p_2 ,…. p_8, these are the GPS traces with a time associated with each point. It can be seen that the first two and last two points are merely part of a trajectory or in other words fly-by points; they do not hold any significance to us apart from telling us the path a user took to reach the stay point and the path they took after leaving the stay point.

How to identify points $p_3 - p_6$ can be a trivial but computationally exhaustive task. In order to identify a location as a stay point the following two criteria need to be met:

A stay point *S,* is a collection of traces *P* (where $P = \{p_m, p_{m+1}, ..., p_n\}$), where $\vee\ m < i \leq n$,

1. Distance $(p_m,p_i) \leq D_{\text{threshold}}$ and
2. $|p_n.T - p_m.T\ | \geq T_{\text{threshold}}$

The point *S* can be defined as $S = (Lat,Lng,T_{arv},T_{dep})$, where *Lat* and *Lng* is the average of all the points from *m* to *n* and T_{arv} and T_{dep} represent the times of arrival and departure to the point *S.*

The $D_{\text{threshold}}$ and $T_{\text{threshold}}$ are the input parameters for the algorithm to extract stay points from a series of GPS traces. Distance (p_m,p_i) is a function written to find the distance between two points. The latitude and longitude values are not linear across the globe, so in order to find the distance between two points with given latitudes and longitudes a conversion to polar coordinates is done.

This customized algorithm is a variant of DBSCAN algorithm. Since we are dealing with two parameters, *time* and *distance*, a 2-dimensional DBSCAN is implemented. The complexity of this algorithm is not the best and it can be in the $O(n^2)$ range in this implementation. This is not a good option in computational terms since we are dealing with thousands of data points. But DBSCAN still offers an advantage over other clustering algorithms.

Other clustering algorithms such as hierarchical clustering and partitioning clustering have also been tested but they give poor results due to their inherited constraints. For example, hierarchical clustering is hard to implement for two dimensional data and also cannot handle noise as well as density based clustering algorithms. Partitioning based clustering poses a big challenge that it requires us to specify the number of clusters we want to split a dataset into. This means that the number of POI that we identify are same for every user, which is not possible. Grid-based clustering has certain advantages when dealing with large datasets, because in such algorithms some discretization of data is done. This can reduce the computational time by a great amount. But it can also lead to loss of important information.

In conclusion, all the algorithms require specifying some sort of threshold values and the performance of the algorithm is greatly affected by those threshold values. So based on the outcome of the custom DBSCAN algorithm and also the fact that it is most commonly used technique in the problem of POI extraction, it was decided to use this algorithm. Certain parameters are calculated for each point of interest, these are the time a certain point was entered, the time spent at that point and the radius of that point. These values are used in the later stages.

3.2 Grouping Similar Points of Interest

After finding all the points of interest for one user we need to identify the similar points of interest. One challenge while working with GPS trajectories is that the coordinates logged by a GPS device can never be 100 % precise or accurate. Hence, we can have slightly different GPS readings from a similar point. This along with variation in sampling rate can cause differences in readings even though they are taken from a same point. In order to overcome this problem, the two POI are thought to be similar if the user visited them within a specified time difference and they lie only a specific distance from each other.

This can be explained better from an example. Consider Activity A, this activity represents a work place, the user started this activity at 8:15 am and the radius of the activity is 9 m and the center point is the mean of all the GPS traces. Similarly, on another day Activity B is detected which occurred at 8:48 am and the mean calculated for the activity is 10 m away from that of center of Activity A. This can happen due to the inaccuracy of the GPS device and human behavior. So keeping the precision of the GPS device in view both activities A and B are identified as similar activities or the points are classified as similar a POI. The new point of interest is calculated as a mean of the GPS traces from activity A and B and the start of the activity is the earliest start time of the two activities. Calculation of a new mean by merging two or more POI brings forward another problem. The radius of the new point of interest can grow by a very large amount and this growing radius can cause other POI to merge with each other. This causes loss of information since in the end we are left with one big point of interest with little information.

To solve this problem a new approach is used. Since it is observed that most GPS points are concentrated close to the mean of a point of interest and a sparse population of points farther away from the mean is contributing to very large POI radii; an activity radius reduction method is used. This is done by using the Chebyshev's inequality. A 50% confidence interval is selected for each point and the new radius is calculated. This enables in pointing to a location with more accuracy than before. Equation (3-1) shows the Chebyshev's inequality.

$$\boldsymbol{P}(|\boldsymbol{x}| \geq \boldsymbol{\varepsilon}) \leq \frac{\boldsymbol{E}(\boldsymbol{x}^2)}{\boldsymbol{\varepsilon}^2} \qquad \textit{(3-1)}$$

It can be seen from Figure 3.2 that after calculating the 50% confidence ellipse the points outside the ellipse are discarded but the mean is not affected to any significant amount.

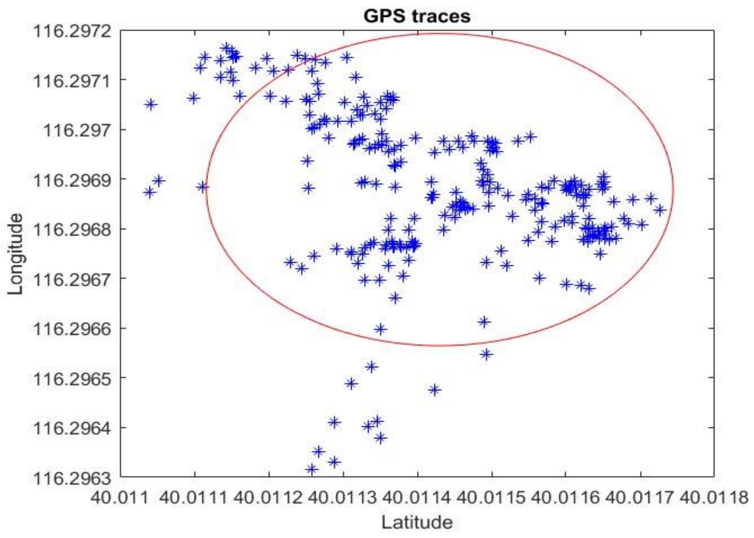


***Figure 3.2** The 50% confidence ellipse plotted for the traces of one stay point. [4]*

After grouping similar activities or POI some information is also obtained for each activity. This information is utilized in further analysis of the points. Consider an activity point *P*, containing *n* number of latitude and longitude readings and equal number of time values. The information that is collected from this activity point is given in Table 1.

***Table 1.** Extracted information for an activity point*

Mean of each activity (Center point of *P*)	CP$(lat,lng) = \left(\frac{\sum_{i=1}^{n} lat_i}{n}, \frac{\sum_{i=1}^{n} lng_i}{n}\right)$
Start time of an activity	$T_{start} = \min(t_1,t_2,\ldots,t_n)$
Total time spent at the activity point	$T_{total} = T_{start} - \max(t_1,t_2,\ldots,t_n)$
Frequency of visit	Initialize i = 0 For each visit i = i +1
Radius of activity point	$R = max(Dist^*(lat_i , lat_{cp}, lng_i , lng_{cp})$ where $i \leq 1$ *to n)* $^*Dist(lat_1,lat_2,lng_1,lng_2)$ is a function that calculates the displacement between the coordinates of two points

3.3 Activity Recognition

As discussed in chapter 2, having some semantic information about a certain set of coordinates can better help in understanding user mobility behavior. Better understanding of mobility patterns can in-turn help in forecasting the next activity or location of interest.

Keeping this in mind some sort of activity labels can be attached to each point of interest. These labels are based on the activity a user might perform at a point. It is worth mentioning that, one point of interest can have more than one labels based on different users mobility patterns.

Since the information available is only about time and geographical location it is hard to have accurate semantic information about a certain place. Keeping the sparsity of data in view a very simple approach is adopted for activity recognition. The first step is to decide on the most common activity labels. Based on literature and intuition four to five activity labels are selected. These are given in below in Table 2**Error! Reference source not found.**

Table 2. *List of activity labels*

Activity 1	At work
Activity 2	At home
Activity 3	Leisurely activity
Activity 4	Shopping/chores
Activity 5	Dine out

The approach used for activity recognition is very intuitive. It tries to mimic human interpretation in a rule-based manner. For example if it is known that a person spends on average eight to twelve hours of their day in a place, then most likely that place is their home. Similarly if someone spends 30 to 60 minutes at a point of interest during noon, then that place is most likely where they eat lunch. These assumptions can go wrong in certain cases but based on a weighted average methodology it is easy to identify majority of the activities. In some cases activity 3 (leisurely activity) and activity 4 (chores) are very similar since they are performed during same time of the day and can take equal amount of time hence for final implementation they are grouped into one label.

Once these labels are finalized a rule based identification technique is used. Theory of fuzzy sets offers great help in implementing such an approach. One advantage of using a fuzzy approach is that no point is 100% labelled with one label and over time the most frequent labels get more weightage. Another advantage is in order to train an Adaptive

Neuro Fuzzy System if no training data is available. Some generic rules can be used to train the system. The outcome of this generic fuzzy model is further discussed in the next chapter. It is also easy to use different models and rules for each user since the parameters are fairly less and easy to train on. So in an ideal case each user has a set of POI where each point has a label associated with it. These labels and the available data can be used to train the ANFIS model which can then output the activity that user might undertake at a certain time of the day. This system can scale well for many users and once every user has their trained ANFIS model the generic model can be discarded.

Having identified user activities it can be easier to forecast which POI the user travels to next. For example, if it is known that a certain user goes out at 9:00 pm on Fridays; in order to forecast their activity for Friday 9:00 pm all the activities labelled as leisure can be given high weightage. This approach will improve the chances of accurate forecasting and can improve user experience by suggesting other similar leisure places.

It is important to point out that for activity recognition only two parameters are used as input. These are the start time of an activity and the time spent at the activity point. The output label is associated to the mean of each activity point.

3.4 Building Mobility Model with Training Data

The next step after extracting the information and attaching labels to different POI is to use that data to model the forecasting problem. As discussed in chapter two, there are two approaches that can be taken to set up the problem solution. Both of the approaches were studied and are discussed below.

3.4.1 Markov Chains

Theory of Markov Chains (MC) was presented in chapter two. This section describes the way the MC are built. In order to form MC we need states and transitions. The most basic MC can be formed by setting the coordinates of POI as states and calculate the transitions from one state to another. Calculating transition probabilities from one state to another requires the discretization of the states. This discretization is required so that all the POI that are within a few meters from each other are clustered into one point. Without discretization there is a chance that POI that are a few meters apart are taken as different states. This results in incorrect calculation of transition probabilities.

To solve this problem, a hierarchical clustering algorithm is used to discretize the origins (start) and destinations (end) of each trajectory. After this step the probabilities of moving from one state to another are calculated. These probabilities can be represented in the form of a matrix and are also known as state transition matrix. Table 3, shows an example of the state transition matrix.

Table 3. *State transition matrix of traveling from one location to another*

	Location 1	Location 2	Location 3
Location 1	0	3/10	7/10
Location 2	4/6	0	2/6
Location 3	3/5	2/5	0

In order to test if there are better ways to make the MC, another approach is also tested. In the second approach the discretization of times is done and the states are formed based on time and the destination. Hence the transition probabilities are calculated based on the time a transition is made to the destination state. Table 4 shows the transition probabilities of travelling to a specific location against each time value.

Table 4. *State transition matrix of travelling to a location at a given time*

	Location 1	Location 2	Location 3
t1	0	4/10	6/10
t2	5/6	0	1/6
t3	3/5	2/5	0

The approach of MC offers the option to customize certain parameters as per requirement, it also makes easy to calculate the conditional probabilities and add weights to all the states.

3.4.2 Neural Networks

When using neural networks there is no need to calculate the transition probabilities in any way. In this approach only the discretization of the start and end points needs to be done. After that the training is performed so that the neural network can train itself and the results of the training can be checked to see how well the system performed.

Using neural networks offers a slight disadvantage that once the network is trained it will provide the output and adding further weight or rule based parameters is not so easy. But on the other hand it offers faster performance. The results of both the approaches are discussed in detail in the next two chapters.

3.5 Activity Forecasting

After the modeling step is completed the actual forecasting of the activity or next POI needs to be done. This is fairly simple compared to the previous steps. In case of neural networks we just need to have the current location and time. These values are discretized to the nearest point in the time-space grid and input it to the trained neural network and an output is obtained.

In case of MC, we have two options on how to proceed. The first option is to simply use the transition probabilities of Table 3 or 4. Knowing the current location and time, the maximum likelihood can be estimated for the next transition. For this purpose using Bayes' Theorem of equation (3-2) is a natural choice.

$$\boldsymbol{P(A|B) = \frac{P(B|A).P(A)}{P(B)}} \qquad \text{(3-2)[56]}$$

This is also called conditional probability and it gives a relationship between the probability of an event occurring before an observation is made *P(A)* and after an observation is made *P(A|B)*. So in this case *P(A|B)* is the probability of transitioning to state *A* when we know that the previous observed state was *B*, *P(B|A)* is the probability of a transition happening, which is computed from the transition probability matrix and *P(A)* which is the prior probability or the probability of being in state *A* when no other information is available. This approach is somewhat simple but the forecast accuracy can be further improved by utilizing the semantic information that was obtained about a user and their mobility patterns.

In order to utilize that data, another table can be constructed for the most likely places a person visits at a certain time of the day. For example if it is known that a user performs some leisurely activity during the evening hence all similar POI can be given more weightage while making the forecast. This will improve the accuracy of the system as well as improve user experience.

4. IMPLEMENTATION

This chapter explains building of a use case, the datasets selection and implementation of the approach of the previous chapter.

4.1 Use Case and Datasets

The type of data selected for a *data-driven analysis* can significantly affect the results. It is generally believed that bigger amount of data will result in higher accuracy. Although this is true to some extent but the statement is not absolutely correct. There are other factors that also affect the accuracy of the results. The diversity of data plays an important role in data-driven analysis. It is important that outliers are represented in the dataset. Due to this reason, due importance should be given to selection of a dataset.

The development of use case requires selection of a dataset and testing of the implemented approach on that dataset. A rich dataset of users with their GPS tracks is not very easy to find. Such data is either not shared by people or if shared, it is not made publically available due to privacy concerns. An ideal dataset should be collected by round the clock location tracking of users. This is not possible in practice but some datasets are available which contain enough data to analyze a user's mobility pattern.

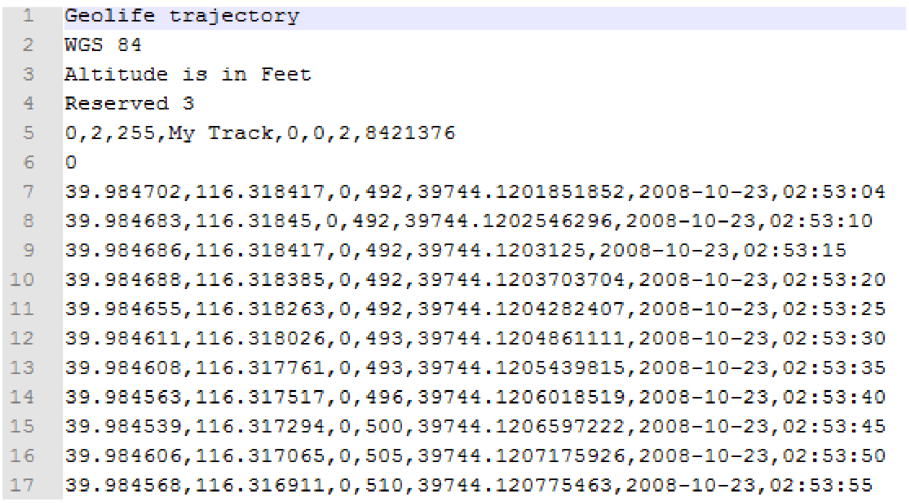
The use case involved reading in a dataset, pre-processing it and splitting it into training and testing data, with a percentage of 80% and 20% respectively. The training data is used to train the predictor. The testing data features are used to forecast an output location. This forecasted value is compared to the actual output of test data. If it is close to the actual value the forecast is considered as accurate. This is done for all the test data and in the end a percentage accuracy is calculated.

Due to constraints a single dataset might offer three datasets were used for testing the proposed system. This also provides a chance to better understand the forecasting system and validate how well it can generalize for different datasets. The problem of using different datasets is that the separate sub-routines need to be written for pre-processing and cleaning each dataset. This is because the data collection process is different for all the three datasets.

4.1.1 GeoLife GPS Trajectories

This GPS trajectories dataset was collected by Microsoft Asia for a project called *Geolife*. It contains trajectories of users based in China mostly Beijing. It was collected over a period of over 3 years and contains 182 users [57]. It is made publically available by Microsoft.

In total the dataset contains 17,621 trajectories which represent data collection of more than 48,000 hours. When combined together the data represents 1.2 million kilometers of travel logs. The data was collected using different devices including GPS loggers and AGPS supported phones. Due to variation of recording devices the sampling rates vary but 91% of the trajectories are dense i.e. data is logged every 1 – 5 seconds or 5 – 10 meters. The trajectories are available in *.plt* file format and contains the *latitude*, *longitude*, *altitude* (if available), *number of days passed since 30.12.1899*, *date* and *time*. Some users have also labelled the mode of transport in their journeys but that information is redundant. [57]



```
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
```

Figure 4.1 *An example of GeoLife dataset [3], [29], [58]*

4.1.2 Mobile Data Challenge (MDC) Dataset

This dataset was collected as part of a research initiative of IDIAP and Nokia Research. This dataset is a huge collection of mobile data which includes logs of different mobile processes including call logs, Bluetooth and Wi-Fi connections, media files accessed, calendars, GPS data and many other events. 80 users took part in the data collection phase and all of them were based in Switzerland, focused around the city Lausanne.

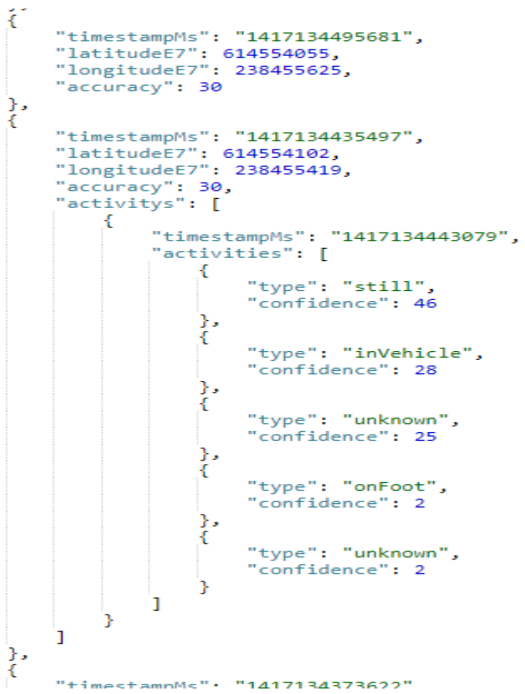
Due to the privacy of users the administrators of this dataset are careful in making the dataset available for research purposes only. Due to this precaution the dataset was provided upon a personal request to the administrators and only relevant data was provided. This means that only GPS traces of users with timestamps and accuracy of the traces was provided. The data available was in the Matlab supported file format, *.mat*. Further details of this dataset can be found at [59], [60].

4.1.3 Google Maps

Google collects data from users whenever Google services are used. To use Google Maps[1] services from a smart phone it becomes essential to share the user location. Google provides its users to save the location history on a cloud and watch the mobility history in a feature called *timeline*. This data can also be downloaded by the data owners from the web[2].

The data is available for download in two formats *.kml* and *.json*. The data can be very rich with Google's own analytics added to it. But the basic data consists of timestamped GPS coordinates and altitude. There might be other data available such as speed, acceleration or activity. The sampling time of this data is not constant as it changes based on the activity. The system is smart enough to not log GPS data when there is no change in position. This along with the fact that the service runs in the background helps in conserving battery and logging only important data.

For this thesis location data of some researchers of FAST-Lab[3] was used with their consent to make analysis of their mobility histories.



```
{
    "timestampMs": "1417134495681",
    "latitudeE7": 614554055,
    "longitudeE7": 238455625,
    "accuracy": 30
},
{
    "timestampMs": "1417134435497",
    "latitudeE7": 614554102,
    "longitudeE7": 238455419,
    "accuracy": 30,
    "activitys": [
        {
            "timestampMs": "1417134443079",
            "activities": [
                {
                    "type": "still",
                    "confidence": 46
                },
                {
                    "type": "inVehicle",
                    "confidence": 28
                },
                {
                    "type": "unknown",
                    "confidence": 25
                },
                {
                    "type": "onFoot",
                    "confidence": 2
                },
                {
                    "type": "unknown",
                    "confidence": 2
                }
            ]
        }
    ]
},
{
    "timestampMs": "1417134373622"
```

Figure 4.2 Sample from Google Maps data

The implementation was carried out in two parts. The first part was the development and testing of the algorithm and the second part was the deployment of the system.

[1] https://www.google.fi/maps
[2] https://takeout.google.com/settings/takeout
[3] http://www.tut.fi/en/fast/

4.2 Algorithm Development

In the initial phase, there was a need to find a convenient way to implement the approach explained in the previous chapter. The reason to do this is that, it helps in comparing results of different algorithms. It also makes it possible to focus on designing algorithms rather than getting stuck in the details of implementation. MATLAB fits to these requirements perfectly. Its free license for students and educational institutes also makes it a good choice for analyzing and improving the actual algorithms. MATLAB has certain built-in toolboxes that take away the problems of computations and low level implementations. This helps in focusing on the original problem.

The implementation of this thesis was started in MATLAB[4] version R2016b and later completed in version R2017b. There was no significant difference in the two versions for the toolboxes used but the newer version promises better implementation of built-in functions. Below a brief description of how the algorithms were implemented is given.

4.2.1 Activity Recognition

As explained in the last section, the most important part of activity recognition is to identify activities from a huge amount of GPS coordinates. This part is also called points of interest extraction. To extract this information from the set of timestamped GPS coordinates a subroutine is written. Algorithm 1 gives an overview of the subroutine.

The first step is to clean the data from irrelevant information. Depending upon the data set there is a lot of useless information that can be discarded. Doing so helps in having smaller data arrays and matrices which makes data handling easer and processing faster. The outcome of this cleaning is that only GPS coordinates and timestamps are stored in a matrix. This matrix is then sorted with respect to time to ensure that the data is in a sequence. Next two loops are initiated to check if the traces qualify two conditions. These two conditions are:

1. Distance between two consecutive GPS coordinates is less than 20m
2. Time difference of two consecutive traces is more than 20 mins.

If both the conditions are met, another loop is initiated to find all the consecutive traces where this condition is satisfied. Once all such points are found the index of those traces are saved in an array to identify a stay point. This is repeated for all the traces.

[4] https://www.mathworks.com/products/matlab.html

Algorithm 1 extract_points_of_interest(user_traces, deltT, delD)

```
clean_data(user_traces)
sort data w.r.t. time
stay_points_array
i = 1
repeat till i is less than the number of traces
    j = i+1
    repeat till j is equal to number of traces
        if distance between trace i and j is less that delD else break loop
            if difference between timestamp of trace i and j is greater that delT
                repeat till distance between trace i and j is less than delD and timestamp difference is less than delT
                    j = j+1
                    if j is greater than number of traces then break loop
                j = j-1
                stay_point = traces from i till j
                i = j
                break loop
    add stay_point to stay_points_array
    i = i +1
return stay_points_array
```

Algorithm 1. *Extracting stay points*

The next step is to find some information or properties of these identified stay points. These are mean of latitude and longitude values (center of the stay point), time of arrival at a stay point, time of departure from a stay point, number of GPS points recorded at a stay point, and radius of the stay point (distance from the center to the farthest point).

After this all the similar stay points are grouped together by using the information from the previous step. From Algorithm 2, it can be seen how this is done. The main idea here is to identify the stay points which are overlapping with each other in terms of coordinates and have almost similar start and end time. For an activity to be classified as similar to another one the start time and end time of both can vary up to 1.5 hrs as a maximum.

Once such stay points are identified their corresponding indexes are marked into a matrix of boolean values. Then, a masking array is created which finds indices of similar activities by applying the AND operator over the boolean matrix. The final outcome is obtained by taking a UNION of all indices from similar stay points and merging them.

```
Algorithm 2      group_similar_stay_points(stay_points_info)
1.  i = 1
2.  repeat till i is equal to total number of stay points -1
3.      j = i+1
4.      repeat till j is equal to total number of stay points
5.          if distance between center of stay point i and stay point j is less than radius of stay point j
6.              if difference between time of entry and time of departure to/from stay point i and j is less than 1.5 hrs
7.                  bolean_matrix(i,i) = 1
8.                  bolean_matrix(j,i) = 1
9.  i = number of stay points
10. repeat till i is equal to 1
11.     mask = boolean_matrix(i,all) = 1
12.     j = i-1
13.     repeat j till j is equal to 1
14.         if ANY value of (mask AND boolean_matrix(all,j) is true)
15.             find index of boolean_matrix with true
16.             add the index to array
17.     combine all activity indexes
18. return grouped_activities_info
```

Algorithm 2. *Grouping similar activity points*

After the similar activities are identified, the properties of stay points are recomputed. These recomputed properties give important information about the stay points and the user behavior. Simply by knowing at what time a person goes to a certain place and how much time he/she spends there, we can start building a profile of the user. But in order to extract semantic information from the stay points we need certain specific labels for each stay point. This makes the process of activity recognition more quantitative.

For this step a machine learning approach is used. Fuzzy Logic is a MATLAB toolbox which is used to formulate this problem as a fuzzy reasoning problem. The two variables used in this case are:

1. time of arrival at a stay point and
2. the time spent at a stay point.

The stay points are assigned activity labels as according to Table 2.

Based on these activity labels four outputs classes are defined, and an ANFIS is trained to predict the class label based on the two inputs provided to it. Due to lack of labelled data, another approach was used for predicting output labels using fuzzy reasoning. A generic FIS (fuzzy inference system) structure was created with some rules written based on most common human behavior. The membership functions were also adjusted, keeping in view the output results. This FIS was capable of predicting output label classes based on the inputs provided. Once the output data was labelled using the fuzzy approach, these labels were fed back to the ANFIS model to train another FIS. This FIS model was then used for other researchers activity prediction. This technique is presented in [4].

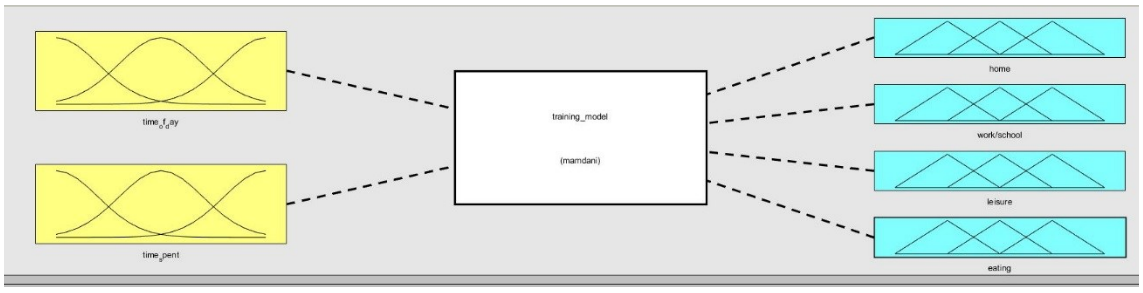
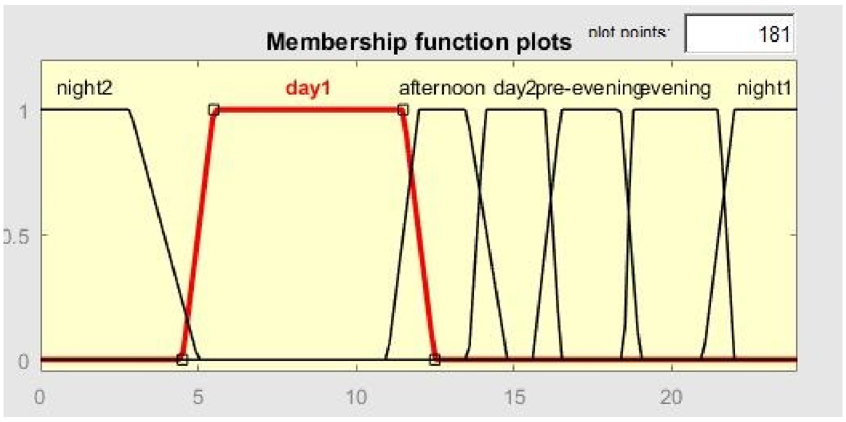
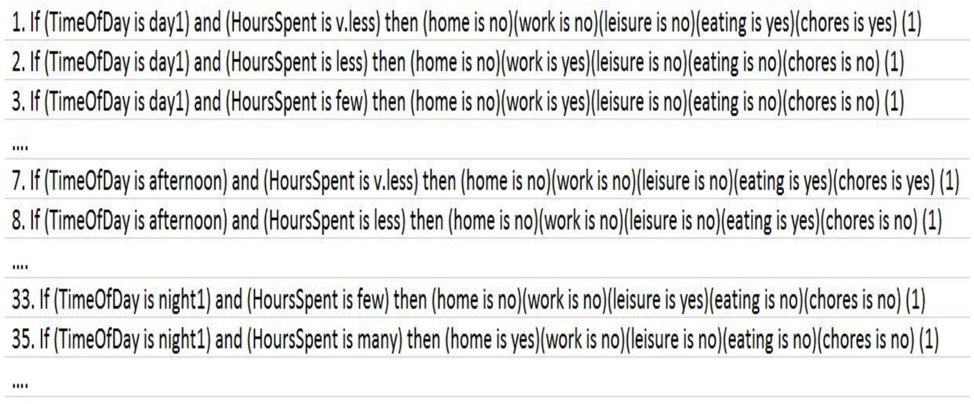


Figure 4.3 *Fuzzy logic controller with two inputs and four outputs [4]*



(a) Membership functions for the generic FIS model



1. If (TimeOfDay is day1) and (HoursSpent is v.less) then (home is no)(work is no)(leisure is no)(eating is yes)(chores is yes) (1)
2. If (TimeOfDay is day1) and (HoursSpent is less) then (home is no)(work is yes)(leisure is no)(eating is no)(chores is no) (1)
3. If (TimeOfDay is day1) and (HoursSpent is few) then (home is no)(work is yes)(leisure is no)(eating is no)(chores is no) (1)

....

7. If (TimeOfDay is afternoon) and (HoursSpent is v.less) then (home is no)(work is no)(leisure is no)(eating is yes)(chores is yes) (1)
8. If (TimeOfDay is afternoon) and (HoursSpent is less) then (home is no)(work is no)(leisure is no)(eating is yes)(chores is no) (1)

....

33. If (TimeOfDay is night1) and (HoursSpent is few) then (home is no)(work is no)(leisure is yes)(eating is no)(chores is no) (1)
35. If (TimeOfDay is night1) and (HoursSpent is many) then (home is yes)(work is no)(leisure is no)(eating is no)(chores is no) (1)

....

(b) an example of rules for the generic FIS model

Figure 4.4 (a)-(b) *Setting up of a generic FIS model [4]*

To visualize the output of the activity recognition algorithms a JavaScript code is written. This code plots the stay points, and the calculated radii on google maps. The information about the activity labels and the probability values are also shown on the map. Google maps API (application programming interface) is used for this purpose.

4.2.2 Activity Forecasting

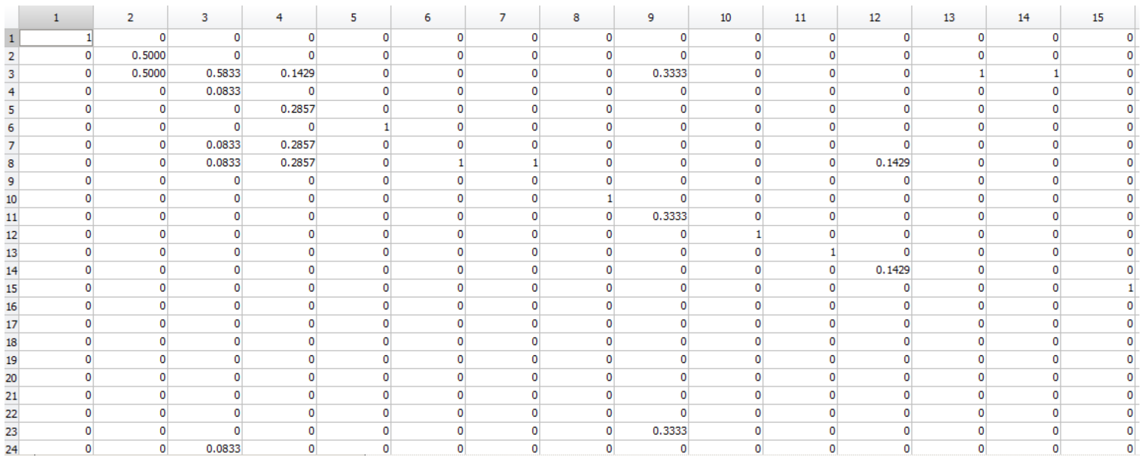
The next stage is to forecast the activity or point of interest of the user based on the context. The context is the current time, location or both. For forecasting activities two approaches were put forward in the previous chapter.

The algorithm for Markov chains based prediction is given below as Algorithm 3.

```
Algorithm 3     forecasting_with_markov_chains(user_traces,current_ location)
 1. clean_data(user_traces)
 2. sort data w.r.t. time
 3. identify unique trajectories
 4. find start and end points of the trajectories
 5. Discretize start points [heirarchical_clustering(points, cutoff)]
 6. Discretize end points [heirarchical_clustering(points, cutoff)]
 7. Discretize timestamps [heirarchical_clustering(points, cutoff)]
 8. Calculate probabilities of being at an end point (priori)
 9. Form transition probability matrices
10. a = find closest start point to current_location
11. Pa_bs = transition probabilities from point a to all end points
12. Pbs = probabilities to be at an end point
13. probAtoB = Pa_bs x Pbs
14. maxInd = find index of maximum(probAtoB)
15. forecsted_location = discretized_end_points(maxInd)
16. return forecsted_location
```

Algorithm 3 *Predictions by using Markov Chains*

The unique trajectories are found by checking the timestamps of the traces. If the timestamps are a few hours apart it means that they represent two different journeys. Once these trajectories are found their first points can be thought of as starting points for a new journey and the last points as the destination or ending point of a journey. Once these points are obtained they are discretized using a built-in function from MATLAB for hierarchical clustering. The input to the function are points and the cutoff distance, which is the minimum distance to form a cluster. This discretizing is necessary for reducing the number of end points since the inaccuracy of GPS readings can cause one start point to be identified as many different end points. The cut-off value chosen in this case in approximately 20m. Once all the points are discretized the next step is to find the probability of being at a certain end point. This is also called the prior probability and it helps in computing the final probabilities. Once we have the probability values for being at every end point, we make the transition probability matrix for all the states (start and end points). This matrix represents the probabilities of transitioning from one start point to all the end points. Most of the values in this matrix are 0 as can be seen from Figure 4.5.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0.5000	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0.5000	0.5833	0.1429	0	0	0	0	0.3333	0	0	0	1	1	0
4	0	0	0.0833	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0.2857	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0.0833	0.2857	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0.0833	0.2857	0	1	1	0	0	0	0	0.1429	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0.3333	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0.1429	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0.3333	0	0	0	0	0	0
24	0	0	0.0833	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.5 *Transition probability matrix*

After completion of this step the prediction system is trained to make predictions. In order to make a prediction the current context is input to the predictor. The predictor finds the closest discretized start point and obtains the transition probabilities from that start point to other locations. Then using the approach of Bayes' Theorem, final probabilities are calculated. In the end, the maximum value of probability is found and the index for that value is noted. The index represents the discretized end point, which is then displayed as a prediction.

The current context can be time or location or both. The algorithm given in Algorithm 3 only shows predictions based on the current location. Similar approach can be used for a time-based predictor, by comparing the time current time to all the discretized time points.

The other approach that was implemented and tested was that of Neural Networks. The data preparation algorithm is very similar to that used in the MC predictor. But it varies in the end. This can be seen from Algorithm 4. MATLAB offers an easy to use GUI for the neural networks toolbox which can be used to train a network with any number of layers or epochs and the code is auto generated. The neural network approach was not used in the final implementation due to its unimpressive results.

```
Algorithm 4    forecasting_with_neural_networks(user_traces,curent_loc, current_time)
 1. clean_data(user_traces)
 2. sort data w.r.t. time
 3. identify unique trajectories
 4. find start and end points of the trajectories
 5. Discretize start points [heirarchical_clustering(points, cutoff)]
 6. Discretize end points [heirarchical_clustering(points, cutoff)]
 7. Discretize timestamps [heirarchical_clustering(points, cutoff)]
 8. X = [discretized start points, discretized timestamps]
 9. Y = discretized end points
10. train a neural network with X and Y
11. Input current_loc and curr_time to neural network
12. Obtain forecasted location
13. return forecsted_location
```

Algorithm 4 *Prediction using neural networks*

4.3 Deployment of Algorithms as Web Services

After designing and analyzing different algorithm, the next step is to identify the best predictor and deploy it as a web service. The advantage of doing this is that a web service can be deployed as a standalone application which can be integrated with other applications. This is truly how a distributed application should work.

Having implemented the code in MATLAB there are two options. First is to use a licensed software from MATLAB known as a MATLAB production server. This software can be installed on any OS and can run MATLAB at runtime. The MATLAB code can be hosted in the server without any problems and exposed and consumed as a RESTful web service.

The second option is to export the code using MATLAB Library Compiler to any other programming language such as C++ or Java and use it as a library in that language. The second approach seemed more feasible for the deployment as it offered more flexibility and a more standardized approach. The testing of the deployed system is done through Postman[5] and Google Chrome[6].

4.3.1 Java and Spring

It was decided to use Java for the final deployment. Using the MATLAB option to export the code as Java was used. The exported package was run and a jar was created ready for deployment. The jar was then installed using maven which helped in the dependency management. It is to be noted that using MATLAB code in Java requires importing the jar file for MATLAB Runtime as well. Online tutorials helped a lot in this phase.

[5] https://www.getpostman.com/

[6] https://www.google.com/chrome/

Once the packages were imported successfully to a Java project, the next step was to expose it as a RESTful web service. Spring offers an excellent framework to write RESTful web services. It uses an easy annotation-based approach to achieve this. It offers excellent security and integrity features as well. So it was decided to use the Spring framework for the deployment.

Based on the results it was decided to implement three web services. All these are POST methods. Where the payload is the traces of a user and different threshold values. Depending upon the service the result is then returned as a JSON string. The three implemented web services are:

1. Identify activities
2. Train Markov Chains predictor
3. Predict next location

The web services were checked on a local host and generated results similar to those in MATLAB.

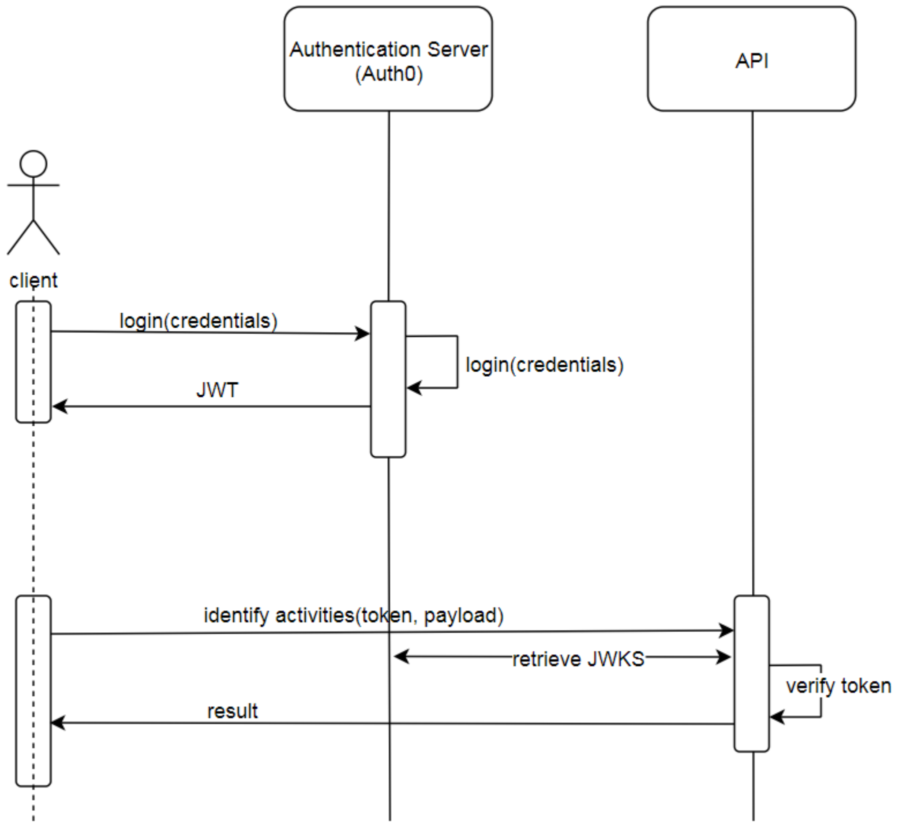


Figure 4.6 *Sequence diagram for making an HTTP request to a web service*

4.3.2 Authentication

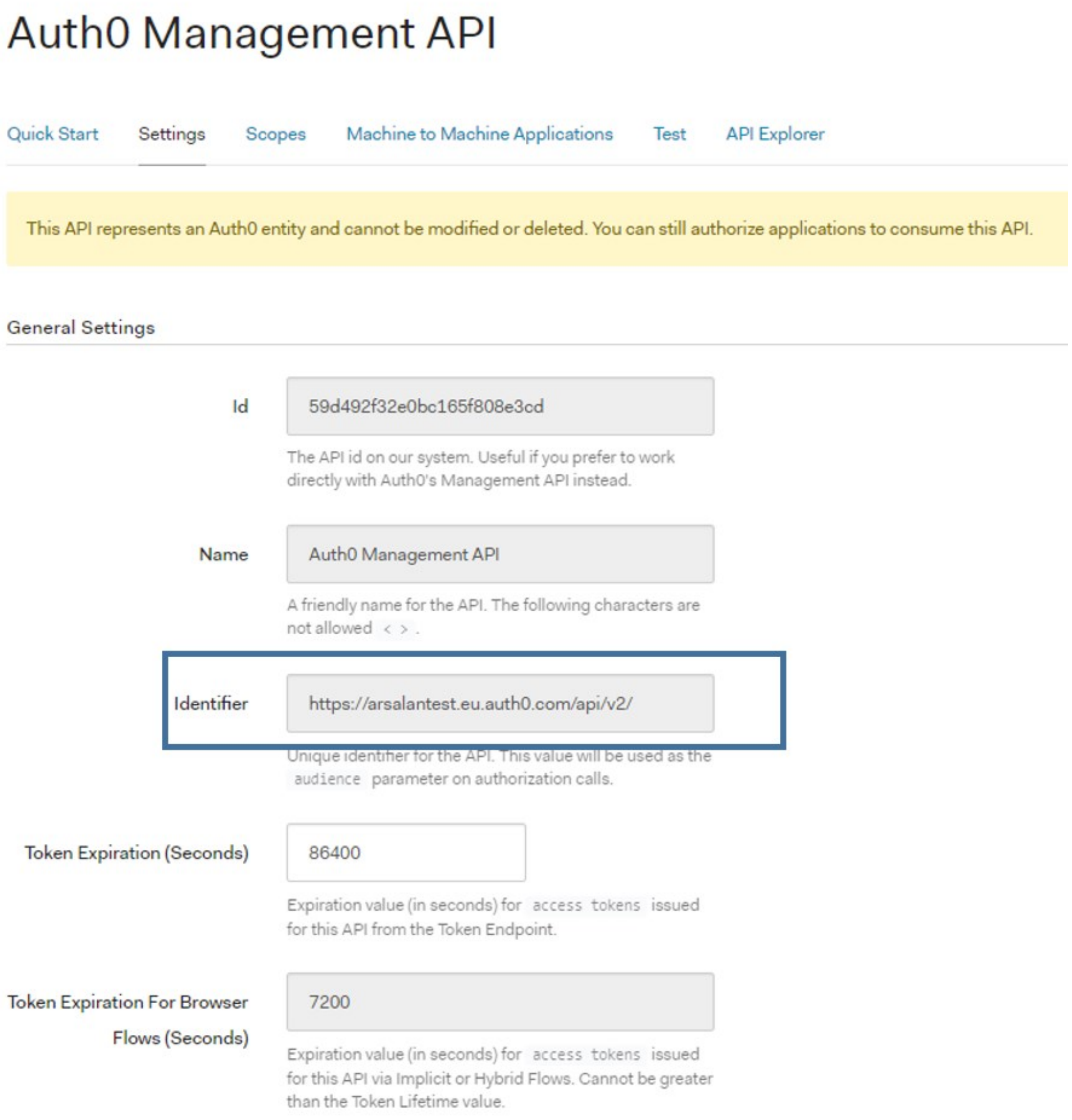
While dealing with sensitive data it is very necessary to make sure that the right people have access to the data and only people authorized to use the web services can do so. If the prediction system is to be integrated to other applications it also needs to support the latest authentication methods used in the industry.

To implement this authentication and authorization [61]mechanism there are two options. The first option is to develop a personal authentication and authorization server based on OpenID and OAuth standards [62]. The other option is to use an existing implementation of the authentication and authorization process. The second option is favored as it relies on specialized service providers to implement the standards and reduces chances of errors or loopholes. Another advantage is, as the authentication and authorization mechanisms advance and new standards are created, these service providers have to make sure that there solutions are up to date. Keeping this in view a third-party solution is used. Auth0[7] was selected because it's a well-known Identity Manager and offers free of charge service for up to 7000 active users. It provides a standalone authentication server which can be used to authenticate users and provide JSON Web Tokens [63] for making requests to an API. JSON Web Tokens are a safe way of transferring claims between two parties.

There is a multiple number of options how to set up the authentication flow. Depending upon the application type and intended use Auth0 offers different authentication flows and software SDKs which can be used for easy setup of the authentication process. There is option for setting up flows for web apps and single page applications. But since the web services are only intended for consumption by an already existing software, the most suitable option is to use the backend/API option. This option is used to secure an existing API using Auth0 authentication server.

Once the configuration is done from the Auth0 dashboard, the endpoints are configured in the Java. The Spring controller is already set up and only a properties file needs to be added to the project, which contains the domain and API credentials.

[7] https://auth0.com/


Auth0 Management API

This API represents an Auth0 entity and cannot be modified or deleted. You can still authorize applications to consume this API.

General Settings

Id: 59d492f32e0bc165f808e3cd

The API id on our system. Useful if you prefer to work directly with Auth0's Management API instead.

Name: Auth0 Management API

A friendly name for the API. The following characters are not allowed `<` `>`.

Identifier: https://arsalantest.eu.auth0.com/api/v2/

Unique identifier for the API. This value will be used as the `audience` parameter on authorization calls.

Token Expiration (Seconds): 86400

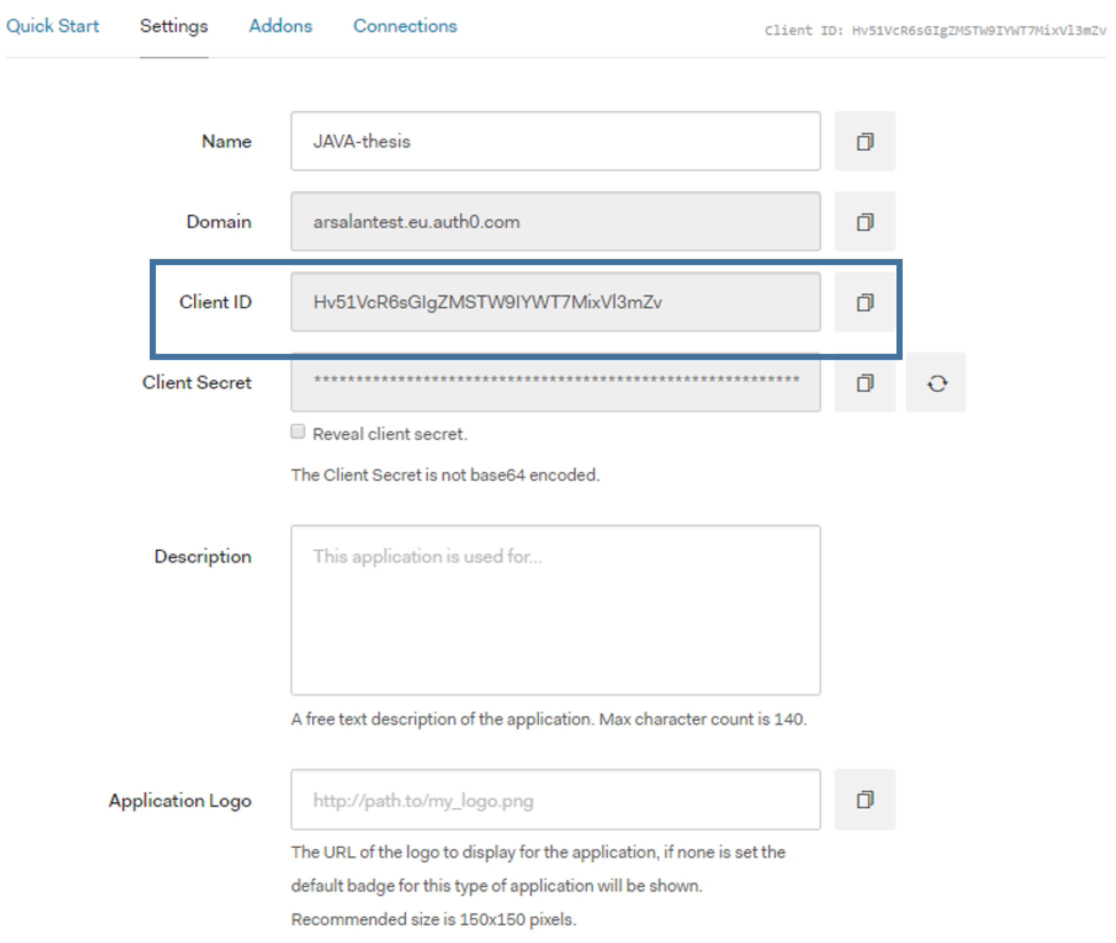
Expiration value (in seconds) for `access tokens` issued for this API from the Token Endpoint.

Token Expiration For Browser Flows (Seconds): 7200

Expiration value (in seconds) for `access tokens` issued for this API via Implicit or Hybrid Flows. Cannot be greater than the Token Lifetime value.

Figure 4.7 *Auth0 dashboard view for setting up an API, highlighted value is input in server configuration*

On starting the server and making request to the end points gives an Authentication Error, which means that the end points can only be accessed with a token. In order to receive the token a non-interactive client is setup from the dashboard and request is made to receive the token. Once the token is added to the POST request the desired response is received.



Client ID: Hv51VcR6sGIgZMSTW9IYWT7MixVl3mZv

Name	JAVA-thesis
Domain	arsalantest.eu.auth0.com
Client ID	Hv51VcR6sGIgZMSTW9IYWT7MixVl3mZv
Client Secret	**

Reveal client secret.

The Client Secret is not base64 encoded.

Description: This application is used for...

A free text description of the application. Max character count is 140.

Application Logo: http://path.to/my_logo.png

The URL of the logo to display for the application, if none is set the default badge for this type of application will be shown. Recommended size is 150x150 pixels.

Figure 4.8 *Auth0 dashboard for (non-interactive) client set-up, Client ID is sent with every request*

4.3.3 Encryption

JSON web tokens are a good way to authenticate and authorize users but they do not ensure total security. If a token is intercepted on the way it can still be used by someone to impersonate a person. The best way to overcome this problem is to secure the communication flow between the client and the server. Using HTTPS instead of HTTP is the standard way to secure end to end communication. Since sensitive user data is also sent as a payload so it makes it even more important to encrypt the communication using HTTPS.

Configuring this encrypted HTTPS communication is easily achievable with Spring. Spring offers the opportunity to run a server with HTTP or HTTPS option simply by writing a few configuration lines and adding the key store file to the project.

For this implementation a private certificate was generated using Java's built-in key store. The generated certificate when added to the request made to the API returns the correct response. If the certificate is not added it returns a warning about proceeding forward and the risks associated with it. This warning is caused due to the fact that a private certificate is used and it cannot be verified by the certification authority (CA).

Figure 4.9 shows a sequence diagram for making a request to the web service for identifying activities. In the first step a request is made to the Auth0 authentication server to provide a access token. For this request the credentials are provided. The authentication server validates that the credentials are correct and sends an access token in the response. This token is then used in the header of all the requests made to the API for using any of the three web services. In this case, the request goes to the service for identifying activities from the user traces. First the service validates that the token is valid by checking the signature and expiry time of the access token. Once it is sure that the token is valid, it process's the traces and returns the results.

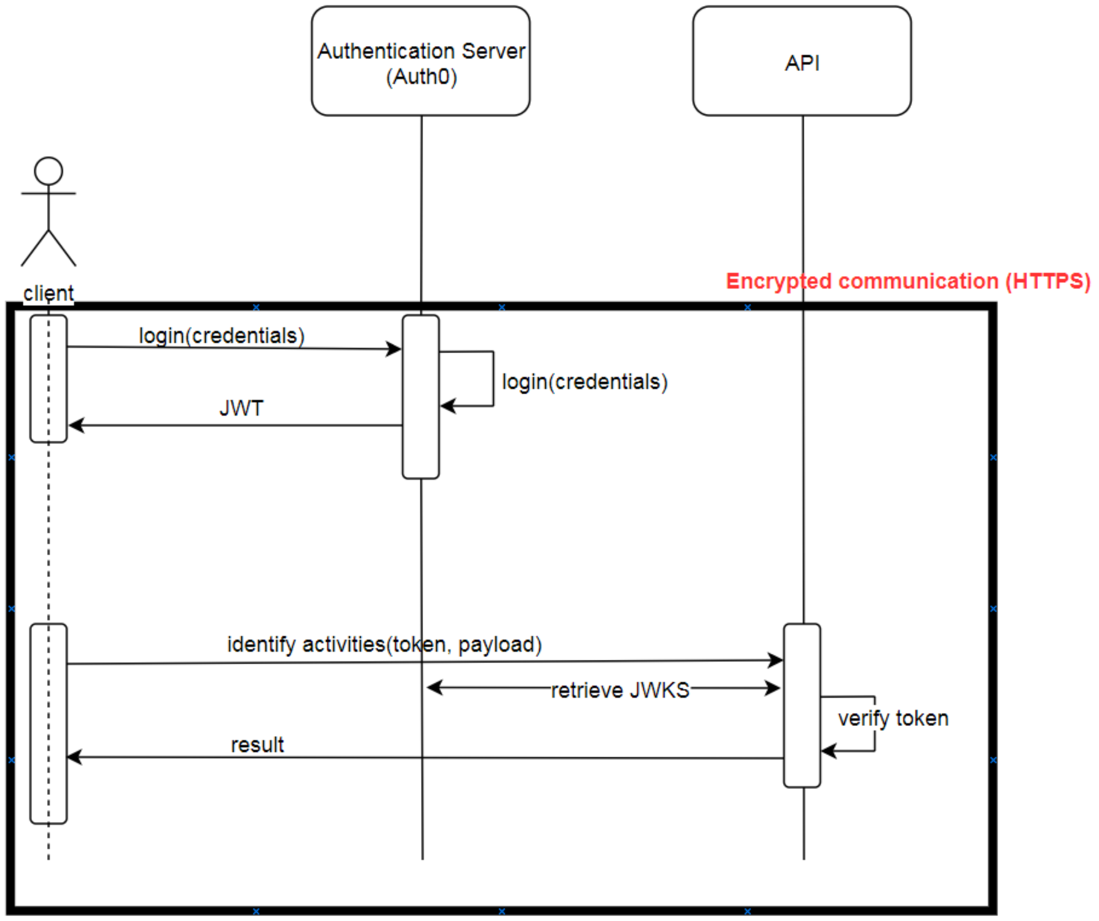

Authentication Server (Auth0)
API
client
Encrypted communication (HTTPS)
login(credentials)
login(credentials)
JWT
identify activities(token, payload)
retrieve JWKS
verify token
result

Figure 4.9 *Sequence diagram for making an HTTPs request to a web service*

5. RESULTS & DISCUSSION

This chapter presents results of the implementation described in the previous chapter. It also gives a brief explanation of how the results were obtained.

5.1 Activity Recognition

A small JavaScript code is used along with Google Maps API to plot the results of the activity recognition algorithm on Google Maps using HTML. Once the HTML file is opened in the browser, a map of the said location is shown. The map is zoomed in to the area where the activities have occurred. As can be seen from Figure 5.1 the map has markers on it in different colours. Each colour represents a different activity, and the circle around it represents the radius of that activity. In other words the circle represents the area within which the GPS points are recorded. By clicking the markers an info window appears which lists the activity label that is predicted and the percentage value associated to that label.

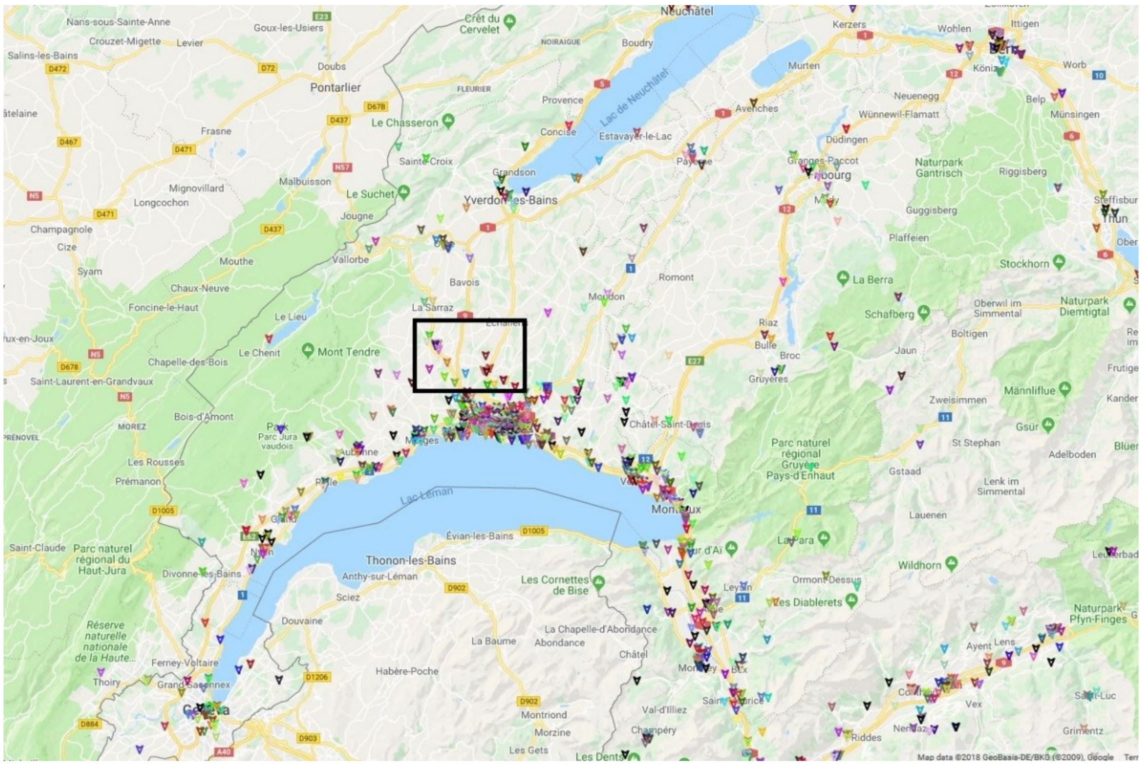


Figure 5.1 *Recognized activities visualized on Google maps*

A portion of Figure 5.1 is enlarged and displayed in Figure 5.2. The activity labels along with the probability are shown in the info window. The labels are accurate with respect to the place they represent on the map, e.g. work represents a TCS office and home represents a house in a residential area.

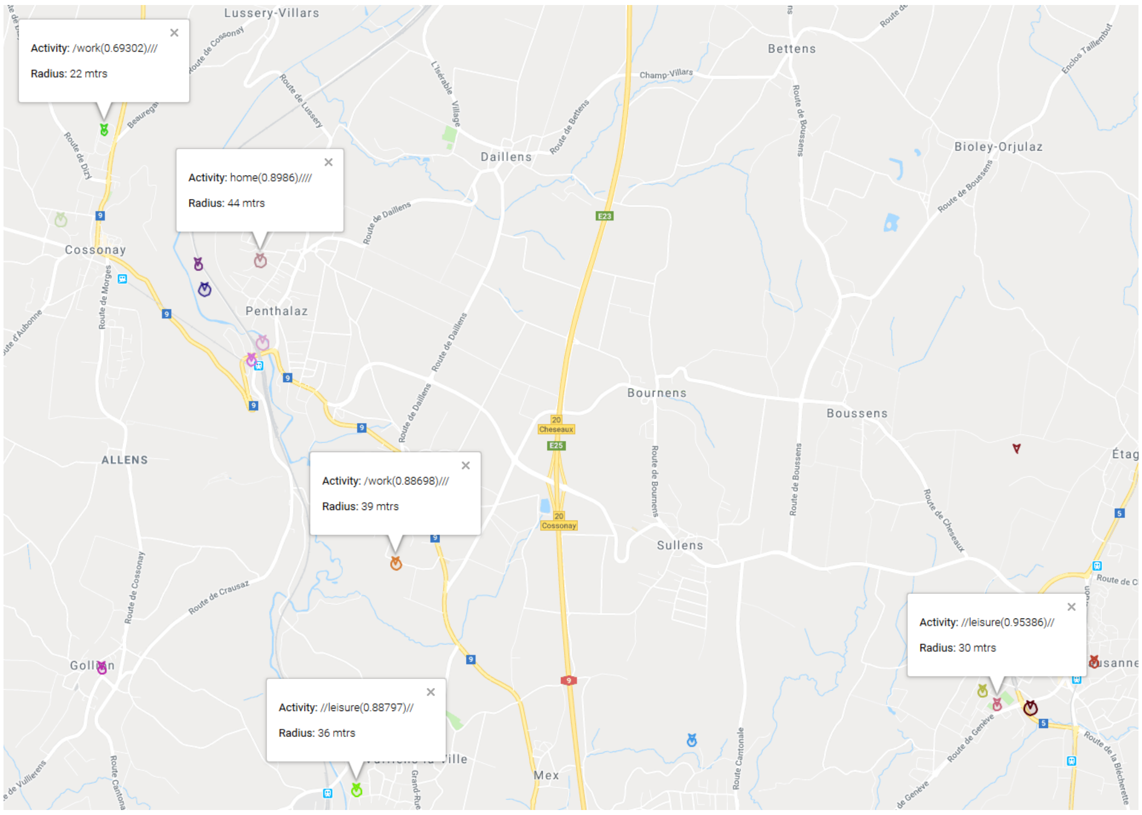
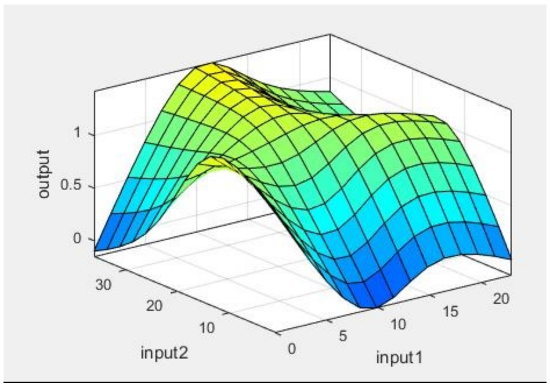
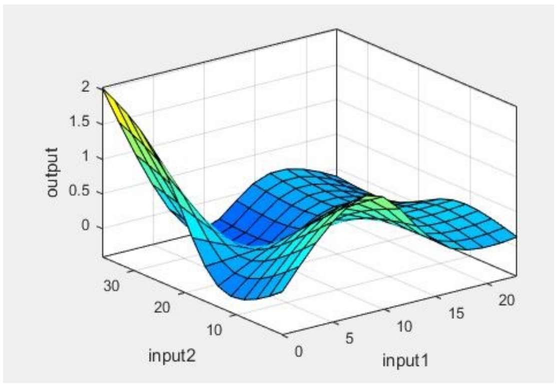


Figure 5.2 *Activity labels with probability values*

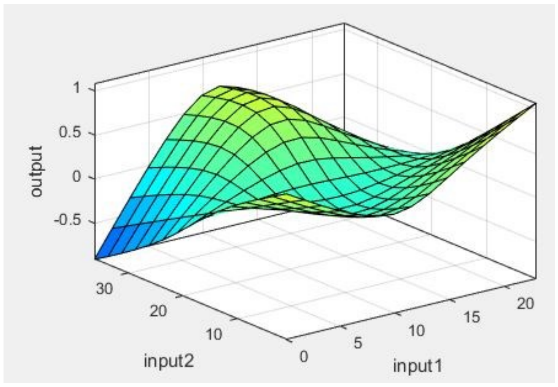
The surface plots for the fuzzy controller can be seen from Figure 5.3. The surfaces are seen to be smooth. They appear to be non-linear but the smoothness indicated the absence of any abrupt non-linear changes. It shows that the trained model adapted well to the data provided. But in some cases, the flat surfaces indicate lack of data for a specific range of input values. It also indicates a lack of outliers. This lack of outliers can be attributed to the fact that the labels were generated automatically from code. In case of actual data the outliers will be represented and analyzed.



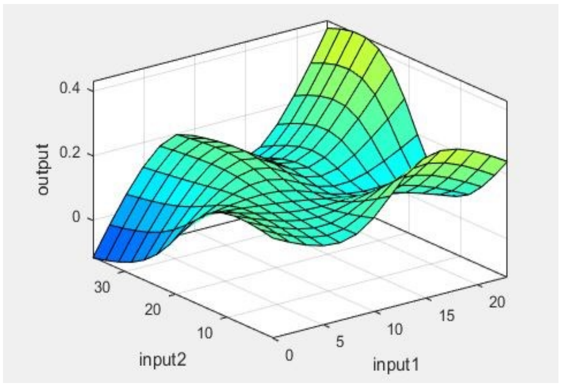
(a) FIS surface for activity *home*



(b) FIS surface for activity *work*



(c) FIS surface for activity *leisure*



(d) FIS surface for activity *eating*

Figure 5.3(a)-(d) *Surface plots for the fuzzy inference system [4]*

Another thing to observe is that the surfaces are relatively linear or flat towards the edges and bumpy in the middle of the plot. This points to the fact that the controller finds it hard to differentiate activities that are performed in the middle of the day and take less than eight hours to be performed.

When observing the visualization of the results in a browser, it can be seen that certain labels can be visually verified. For example, activities happening in a residential area contain the label of home. Activities happening in the city are mostly shopping, or leisurely and those in commercial areas are labelled as work. This shows promising results, but none the less it cannot be qualified as a successful result based on the visual validation only. Unfortunately, no extensive dataset with location and activity labels was found during this thesis work to make a quantitative analysis of this recognition system.

But as discussed earlier the main purpose of the activity recognition was to aid in activity forecasting. So the results of this activity recognition are not the main goal of the thesis. This recognition is used to nudge the forecasting in the right direction by providing some sort of semantic information about the stay points. So even if the accuracy of the recognition is not very high it is still able to help us in understanding a user, and forecasting the next place of movement.

One of the advantages of using fuzzy theory for activity recognition is that it helps to think of the problem intuitively. The answer is not always a binary one but there can be a certain level of surety assigned to the outcome. This mimics human thinking process. It also helps to have easy to train FIS models for any number of individual users or a few specific models based on the personality of users. This can further lead into interesting research on user behaviors. Another advantage is that, in case no labelled data is available, like in the case of this implementation, a generic FIS model can still be used to extract semantic information of a user. The MATLAB tool box for fuzzy logic also assists in understanding behavior of the model. This is a big advantage over a neural network where the process of prediction is very complicated to understand.

The activity recognition algorithm had certain drawback. Since the recognition is done with so few parameters it fails to understand the complicated activities. The activity time and radius are cut off at certain values. This makes it hard to identify activities which take slightly less time or are spread on a bigger area. An activity can vary from person to person. One person might take 10 minutes to do grocery shopping while another might take half an hour. Similarly, if the radius of performing an activity is large it might not be detected as an activity. So certain physical activities such as jogging or cycling cannot be identified. Hence coming up with a generalized identifier can be a challenging task. Some suggestions on how to improve this are discussed in the next chapter. But overall the results of the algorithm are fairly good, given the lack of labelled data and few training parameters.

5.2 Activity Forecasting

Although the neural network approach was easy to implement but the results from it were not promising. So it was not used in the final implementation. The Markov Chain approach was tested with a variety of parameters. For example, an implementation was tested where only the transition based on start location were considered, then another implementation was tested with only time as the deciding factor for transition. All these tests yield different results. This can be mainly attributed to the personality of a user. A person who is punctual will have better results when time centric transitions are taken account. Similarly a person who likes to try different experiences will visit different locations and perform poorly if the current location based transitions are considered. But a person who is consistent in their choices of places where to go will perform better if the current location based transitions are considered.

Still, to better understand the working of the algorithm, many variations of the algorithm were tested. To check the accuracy of the algorithms a quantitative test was set up and performed.

A percentage of data was randomly selected for testing the results of the predictor. This subset is called the test data and it is typically 20% of the original data. The test data inputs and outputs are discretized with respect to the discretized training data. The inputs are fed to the predictor and the predictor makes a prediction on the output. The predicted output is compared to the original output of the test data. The distance between the prediction and actual value is computed. If this distance is less than 20m, the prediction is a hit. If the distance in more than 20m the prediction is a miss. Finally the total hits are divided by the size of the testing data to get an accuracy percentage of the predictor.

Table 5 gives accuracy values of different predictors. The accuracy values vary greatly from user to user but the ones given here are average prediction accuracies for these predictors. These numbers can be compared to the work of previous researchers. One significant work was published by the winners of MDC mobile data challenge. They reported an average accuracy of 56% for the MDC dataset [9]. Hence even though the performance of the predictor seems to be low, it compares well with work done on similar problems.

Table 5. *Markov Chains algorithm variations and their prediction accuracy*

Algorithm Type	Accuracy
Time Dependent Transitions	53.70%
Location Dependent Transitions	57.40%
Combined Transitions	51.85%
Hybrid	33.33%

This test also is based on a pre-decided threshold values for discretization and errors, which can be increased or decreased. These changes can lead to variations in accuracy rates. For the implementation these values were decided based on a typical distance between two significant places in a crowded area of a city.

Understanding the motivation for movement is as complex as understanding the human brain. It is not easy to understand the motivation of a human being for going to a place, merely by looking at their past history. This makes activity recognition and forecasting is a complicated problem to solve. Even if the best predictor is found for one individual, it might not give the same results for another individual. This combined with inconsistent data collection further complicates the problem. In the datasets that were studied, people had used different devices to collect data, they also had different sampling times. There are many other parameters like weather and outdoors temperature that play a significant role in human mobility that were not considered or recorded.

Discretization is also a big challenge encountered in this thesis. In a congested city a shop and an office can be 10m apart from each other. While in a rural area two farms can be kilometers apart. What is the ideal distance to discretize two significant locations from one another? Too many points can lead to unnecessary computational problems and too few points can lead to loss of important data.

All these problems pose challenges in understanding and forecasting human mobility. But in spite of these challenges the predictor presented in this thesis performed sufficiently and met the required criteria.

6. CONCLUSION AND FUTURE WORK

A life time can be spent finding the best forecasting method for human activity and there is potential for major improvements to this work. This chapter contains some concluding remarks on the work of done in this thesis and suggestions for improvement.

Understanding human mobility patterns is a complicated task. A lot of work has been done in this area but the field still has potential for improvement. In this thesis an approach is adopted to forecast human mobility in terms of significant locations of a user. GPS traces and timestamps are used to analyze the history of mobility traces of a user. This approach though not novel is unique in the sense that the data used for solving the problem is limited.

A data-driven approach is used to learn the mobility behavior of users and build up a mathematical model of their mobility. The work done is an extension of previous research done on the subject, where the main approach of extracting semantic information is used to assist in making the predictions. Different training methods are tested, including Markov Chains and Neural Networks to find the best predictor. Another aim of the thesis is to make available such a prediction system as a web service which can be integrated with other applications and assist in understanding user behaviors. This can enable building intelligent apps by integrating the forecasting system in the apps as a third party solution.

Both goals of the thesis are achieved. But as any other system there is margin for improvement of the system. Some suggestions for improvement and future work are discussed here.

The data collection needs to be improved in order to understand the nature and motivation for human mobility. Smart phones offer an excellent method of large scale data collection. Google like services are making this possible, where there is also chances to attach labels to locations and activities.

There also needs to be consistency in how the data is collected. For example collecting GPS traces while a person is not moving drains battery without any additional advantage. Smarter algorithms for sampling the coordinates can help gather more data from users. Besides this more data or information can also be added to the timestamps and GPS values. For example accelerometer data along with GPS values can greatly improve the activity recognition. Linking semantic information to the datasets can also help better understand the motive of travel. All this information can help in producing better estimates on user's mobility.

Additionally consistency is required in storing this data. During this thesis lots of time was spent in reading and cleaning the datasets. Since all the datasets have different storage formats, different items that are stored and different way of representing them. So separate adapters had to be written to read and process the data. Standardization of data collection can help save a lot of time and energy.

Activity detection and forecasting algorithms can be further improved by taking certain things into account. It is a highly possible scenario where a user's home or work address changes. The current work does not take those changes into account. This can lead to redundant labels, which in turn leads to errors in forecasting activities of a user. In addition to this, if more data is available for analysis, this can be used in adding more learning features to the system. This will give interesting insights into user behavior and help in getting better results. In fact a probable future work can be collection of a richer dataset which will indeed open up interesting possibilities for analysis. There is also a need to make the algorithms more efficient. For example the datamining or activity extraction from hundreds and thousands of traces is a computationally exhaustive task. If efficient datamining and clustering algorithms are used, the processing time can be improved to a large extent.

Another area of improvement is the deployment of the system as web services. MATLAB files are exported as Java libraries and used in a Java project. This causes the system to be very slow. The whole system can be developed in another environment like Python which will improve performance. While working with user mobility history, an important thing to notice was the sensitivity of the data. It is impressive, but at the same time, alarming to see how much can be known about a human being by looking at their mobility history. This highly private data should be handled with care and it should be made sure that it is anonymized and stored in the most secure manner and only authorized people have access to this data.

REFERENCES

[1] “activity - definition of activity in English | Oxford Dictionaries,” *Oxford Dictionaries | English*. [Online]. Available: https://en.oxforddictionaries.com/definition/activity. [Accessed: 05-Sep-2017].

[2] P. Nurmi and J. Koolwaaij, “Identifying meaningful locations,” in *2006 3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops*, 2006, pp. 1–8.

[3] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, “Understanding Mobility Based on GPS Data,” in *Proceedings of the 10th International Conference on Ubiquitous Computing*, New York, NY, USA, 2008, pp. 312–321.

[4] A. Shah, P. Belyaev, B. R. Ferrer, W. M. Mohammed, and J. L. M. Lastra, “Processing mobility traces for activity recognition in smart cities,” in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017, pp. 8654–8661.

[5] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, “Understanding individual human mobility patterns,” *Nature*, vol. 453, no. 7196, pp. 779–782, Jun. 2008.

[6] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, “Next Place Prediction Using Mobility Markov Chains,” in *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, New York, NY, USA, 2012, p. 3:1–3:6.

[7] W. Mathew, R. Raposo, and B. Martins, “Predicting Future Locations with Hidden Markov Models,” in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, New York, NY, USA, 2012, pp. 911–918.

[8] A. Asahara, K. Maruyama, A. Sato, and K. Seto, “Pedestrian-movement Prediction Based on Mixed Markov-chain Model,” in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, New York, NY, USA, 2011, pp. 25–33.

[9] V. Etter, M. Kafsi, E. Kazemi, M. Grossglauser, and P. Thiran, “Where to go from here? Mobility prediction from instantaneous information,” *Pervasive and Mobile Computing*, vol. 9, no. 6, pp. 784–797, Dec. 2013.

[10] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From Data Mining to Knowledge Discovery in Databases,” *AI Magazine*, vol. 17, no. 3, p. 37, Mar. 1996.

[11] B. Shirley, *Data Warehousing and Web Engineering*. Idea Group Inc (IGI), 2001.

[12] D. Snider, *Knowledge Discovery in Fetal Activity Data*. 2011.

[13] B. Marr, “Big Data: 20 Mind-Boggling Facts Everyone Must Read,” *Forbes*. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/. [Accessed: 10-Dec-2017].

[14] “World Internet Users Statistics and 2017 World Population Stats.” [Online]. Available: http://www.internetworldstats.com/stats.htm. [Accessed: 10-Dec-2017].

[15] “Internet Live Stats - Internet Usage & Social Media Statistics.” [Online]. Available: http://www.internetlivestats.com/. [Accessed: 10-Dec-2017].

[16] X. Wu and V. Kumar, *The Top Ten Algorithms in Data Mining*, 1st ed. Chapman & Hall/CRC, 2009.

[17] J. Gallestey, “Cluster analysis | statistics,” *Encyclopedia Britannica*. [Online]. Available: https://www.britannica.com/topic/cluster-analysis. [Accessed: 22-Dec-2017].

[18] Nisha and P. J. Kaur, “A survey of clustering techniques and algorithms,” in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2015, pp. 304–307.

[19] “Dendrogram plot - MATLAB dendrogram - MathWorks Nordic.” [Online]. Available: https://se.mathworks.com/help/stats/dendrogram.html. [Accessed: 25-Dec-2017].

[20] S. Arora and I. Chana, “A survey of clustering techniques for big data analysis,” in *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, 2014, pp. 59–65.

[21] I. Bhumika, K. Minal, and Dr. S.P. Khandait, “Review of Clusterization Techniques for Random Data,” *International Journal of Research in Advent Technology*, vol. 2, no. 2, pp. 1–3, Feb. 2014.

[22] R. Yogita and R. Harish, “A Study of Hierarchical Clustering Algorithm,” *International Journal of Information and Computation Technology*, vol. 3, no. 10, pp. 1115–1122, Nov. 2013.

[23] F. Murtagh and P. Contreras, “Methods of Hierarchical Clustering,” *arXiv:1105.0121 [cs, math, stat]*, Apr. 2011.

[24] S. P. Karthik and A. Sheshasaayee, *Clustering of user behaviour based on web log data using improved K-means clustering algorithm*, vol. 8. 2016.

[25] E. Erdemir, D. M. Wilkes, K. Kawamura, and A. Erdemir, *Learning structural affordances through self-exploration*. 2012.

[26] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN,” *ACM Trans. Database Syst.*, vol. 42, no. 3, p. 19:1–19:21, Jul. 2017.

[27] K. P. Agrawal, S. Garg, and P. Patel, *Performance Measures for Densed and Arbitrary Shaped Clusters*, vol. 6. 2015.

[28] K. Kailing, H. Kriegel, and P. Kröger, “Density-Connected Subspace Clustering for High-Dimensional Data,” in *Proceedings of the 2004 SIAM International Conference on Data Mining*, 0 vols., Society for Industrial and Applied Mathematics, 2004, pp. 246–256.

[29] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, “Mining Interesting Locations and Travel Sequences from GPS Trajectories,” in *Proceedings of the 18th International Conference on World Wide Web*, New York, NY, USA, 2009, pp. 791–800.

[30] D. Ashbrook and T. Starner, “Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users,” *Personal Ubiquitous Comput.*, vol. 7, no. 5, pp. 275–286, Oct. 2003.

[31] W. Huang, S. Li, X. Liu, and Y. Ban, “Predicting human mobility with activity changes,” *International Journal of Geographical Information Science*, vol. 29, no. 9, pp. 1569–1587, Sep. 2015.

[32] W. Huang and S. Li, “Understanding human activity patterns based on space-time-semantics,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 121, pp. 1–10, Nov. 2016.

[33] E. Seneta, “Markov and the Birth of Chain Dependence Theory,” *International Statistical Review / Revue Internationale de Statistique*, vol. 64, no. 3, pp. 255–263, 1996.

[34] W. J. Stewart, *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. Princeton University Press, 2009.

[35] S. Ross, *Introduction to Probability Models*, 9th ed. Elsevier.

[36] D. Ashbrook and T. Starner, “Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users,” *Personal Ubiquitous Comput.*, vol. 7, no. 5, pp. 275–286, Oct. 2003.

[37] R. Khanna and M. Awad, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Apress, 2015.

[38] W. S. Sarle, *Neural Networks and Statistical Models*. 1994.

[39] P. Hagenlocher, “Decision Tree Learning,” Fakultat fur Informatik, Technische Universitat Munchen, Proseminar Data Mining.

[40] J. R. Quinlan, “Induction of Decision Trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[41] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, vol. 1, pp. 278–282 vol.1.

[42] *Pattern Recognition and Machine Learning | Christopher Bishop | Springer*. .

[43] J. Pfanzagl and R. Hamböker, *Parametric Statistical Theory*. Walter de Gruyter, 1994.

[44] “Cluster Analysis - MATLAB & Simulink - MathWorks Nordic.” [Online]. Available: https://se.mathworks.com/help/stats/cluster-analysis.html. [Accessed: 25-Sep-2017].

[45] “k-Means and k-Medoids Clustering - MATLAB & Simulink - MathWorks Nordic.” [Online]. Available: https://se.mathworks.com/help/stats/k-means-clustering-12.html. [Accessed: 25-Sep-2017].

[46] “Hierarchical Clustering - MATLAB & Simulink - MathWorks Nordic.” [Online]. Available: https://se.mathworks.com/help/stats/hierarchical-clustering-12.html. [Accessed: 25-Sep-2017].

[47] R. G. Morris, “D.O. Hebb: The Organization of Behavior, Wiley: New York; 1949,” *Brain Res. Bull.*, vol. 50, no. 5–6, p. 437, Dec. 1999.

[48] R. Mirman, *Our Almost Impossible Universe: Why the Laws of Nature Make the Existence of Humans Extraordinarily Unlikely*. iUniverse, 2006.

[49] M. Alam, *Codes in MATLAB for Training Artificial Neural Network using Particle Swarm Optimization*. 2016.

[50] *Mathematical Principles of Fuzzy Logic | Vilém Novák | Springer*. .

[51] J. Vieira, F. Dias, and A. Mota, “Comparison between artificial neural networks and neurofuzzy systems in modelling and control: a case study,” *IFAC Proceedings Volumes*, vol. 36, no. 12, pp. 249–255, Jul. 2003.

[52] “Sugeno Fuzzy Model.” [Online]. Available: http://researchhubs.com/post/engineering/fuzzy-system/takagi-sugeno-fuzzy-model.html. [Accessed: 26-Sep-2017].

[53] N. Eagle, A. (Sandy) Pentland, and D. Lazer, “Inferring friendship network structure by using mobile phone data,” *PNAS*, vol. 106, no. 36, pp. 15274–15278, Sep. 2009.

[54] "Where to go from here? Mobility prediction from instantaneous information." [Online]. Available: https://dl.acm.org/citation.cfm?id=2563750.2564485. [Accessed: 14-Nov-2017].

[55] J. J.-C. Ying, W.-C. Lee, and V. S. Tseng, "Mining Geographic-temporal-semantic Patterns in Trajectories for Location Prediction," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 1, p. 2:1–2:33, Jan. 2014.

[56] D. G. Fryback, "Bayes' theorem and conditional nonindependence of data in medical diagnosis," *Computers and Biomedical Research*, vol. 11, no. 5, pp. 423–434, Oct. 1978.

[57] "GeoLife GPS Trajectories," *Microsoft Download Center*. [Online]. Available: 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. [Accessed: 07-Mar-2018].

[58] Y. Zheng, X. Xie, and W.-Y. Ma, "GeoLife: A Collaborative Social Networking Service among User, location and trajectory," *IEEE Data(base) Engineering Bulletin*, Jun. 2010.

[59] "The Mobile Data Challenge Initiative — MDC." [Online]. Available: https://www.idiap.ch/project/mdc/. [Accessed: 07-Mar-2018].

[60] "Mobile Data Challenge (MDC) Dataset — DDP." [Online]. Available: https://www.idiap.ch/dataset/mdc. [Accessed: 07-Mar-2018].

[61] S. O. Afolaranmi, B. R. Ferrer, W. M. Mohammed, J. L. M. Lastra, M. Ahmad, and R. Harrison, "Providing an access control layer to web-based applications for the industrial domain," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, 2017, pp. 1096–1102.

[62] N. Naik and P. Jenkins, "Securing digital identities in the cloud by selecting an apposite Federated Identity Management from SAML, OAuth and OpenID Connect," in *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, 2017, pp. 163–174.

[63] J. Bradley, N. Sakimura, and M. Jones, "JSON Web Token (JWT)." [Online]. Available: https://tools.ietf.org/html/rfc7519. [Accessed: 25-Apr-2018].