

Spatio-Temporal Data Mining to Detect Changes and Clusters in Trajectories

by

Anirudh Kondaveeti

A Dissertation Presented in Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

Approved November 2012 by the  
Graduate Supervisory Committee:

George C. Runger, Chair  
Rong Pan  
Pitu Mirchandani  
Ross Maciejewski

ARIZONA STATE UNIVERSITY

December 2012

## ABSTRACT

With the rapid development of mobile sensing technologies like GPS, RFID, sensors in smart-phones, etc., capturing position data in the form of trajectories has become easy. Moving object trajectory analysis is a growing area of interest these days owing to its applications in various domains such as marketing, security, traffic monitoring and management, etc. To better understand movement behaviors from the raw mobility data, this doctoral work provides analytic models for analyzing trajectory data.

As a first contribution, a model is developed to detect changes in trajectories with time. If the taxis moving in a city are viewed as sensors that provide real time information of the traffic in the city, a change in these trajectories with time can reveal that the road network has changed. To detect changes, trajectories are modeled with a Hidden Markov Model (HMM). A modified training algorithm, for parameter estimation in HMM, called m-BaumWelch, is used to develop likelihood estimates under assumed changes and used to detect changes in trajectory data with time. Data from vehicles are used to test the method for change detection.

Secondly, sequential pattern mining is used to develop a model to detect changes in frequent patterns occurring in trajectory data. The aim is to answer two questions: Are the frequent patterns still frequent in the new data? If they are frequent, has the time interval distribution in the pattern changed? Two different approaches are considered for change detection, frequency-based approach and distribution-based approach. The methods are illustrated with vehicle trajectory data.

Finally, a model is developed for clustering and outlier detection in semantic trajectories. A challenge with clustering semantic trajectories is that both numeric and categorical attributes are present. Another problem to be addressed while clustering is that trajectories can be of different lengths and also have missing values. A tree-based ensemble is used to address these problems. The approach is extended to outlier detection in semantic trajectories.

## ACKNOWLEDGMENTS

I owe my gratitude to all those people who have made this dissertation possible and because of whom my graduate experience has been one that I will cherish forever.

First, I would like to express my deepest sense of gratitude to my advisor, Dr. George Runger, for giving me an opportunity to work in the exciting area of trajectory data mining. His inspiration and guidance have motivated me throughout my dissertation work. The intellectual and personal growth that I have experienced under his supervision cannot be overstated. I express my gratitude to Dr. Kurt Vanlehn for giving me the opportunity to work on the ANDES project. I am also grateful to Dr. Roger Berger for having appointed me as the statistics lab manager.

I would also like to thank my committee members Dr. Pitu Mirchandani, Dr. Rong Pan and Dr. Ross Maciejewski for reading previous drafts of this dissertation and providing many valuable comments that improved the presentation and contents of this dissertation. I would also like to sincerely thank my colleagues Wandaliz Torres Garcia, Amit Shinde, Houtao Deng, Erik, Anshuman and Mustafa for their help and the technical discussions. I also sincerely thank Arizona State University for selecting me for the University Graduate Fellowship.

Special thanks to my brother Ashwin Kondaveeti and my friends, Ashkar Ali, Ashish Gandhe, Sachin Kadloor, Vighnesh Venkatesan. Despite the distance, they have been very supportive in my graduate studies. I would also like to thank my wing-mates and friends in IIT Madras, Manohar Seetharam, Karthik Tripurari, Amrutayan Pati, Adarsh Sharma, Santosh Suram and others for their belief in me. I like to thank my room-mates and friends in Arizona, Pramod Raman, Karthikeyan Dhandapani and others, who have made my stay in Arizona a memorable experience.

I like to thank my parents, Chandra Sekhar Rao and Bhavani, who raised me very well with a strong sense of personal responsibility, courage and resilience, and prepared me for success in life. Their love, support and understanding has always motivated me to strive for excellence. Finally,

and most importantly, I would like to thank my wife Eshwari. Her encouragement, patience and unwavering love have certainly been the cornerstones on which the past few years of my life have been built. To my wife, parents and brother, I dedicate this dissertation.

# TABLE OF CONTENTS

	Page
LIST OF TABLES . . . . .	viii
LIST OF FIGURES . . . . .	ix
CHAPTER	
1 INTRODUCTION . . . . .	1
1. CONTRIBUTIONS . . . . .	5
2 BACKGROUND . . . . .	8
1. PREPROCESSING TRAJECTORY DATA . . . . .	8
2. ANALYTICS FOR TRAJECTORIES . . . . .	11
2.1. TRAJECTORY CLUSTERING . . . . .	12
2.2. TRAJECTORY CLASSIFICATION . . . . .	15
2.3. TRAJECTORY OUTLIER DETECTION . . . . .	20
2.4. DISCOVERY OF TRAJECTORY SWARM OR CONVOY PAT- TERNS . . . . .	22
2.5. MINING PERIODIC BEHAVIOR IN TRAJECTORIES . . . . .	23
2.6. PROBABILISTIC MODELING OF TRAJECTORIES . . . . .	25
3. HIDDEN MARKOV MODEL . . . . .	27
3.1. BAUM-WELCH ESTIMATION . . . . .	30
3.2. CHANGE DETECTION IN HIDDEN MARKOV MODEL . . . . .	32
4. SEMANTIC TRAJECTORIES . . . . .	34
4.1. IDENTIFYING STOPS AND MOVES OF TRAJECTORIES . . . . .	35
4.2. ANALYTICS FOR SEMANTIC TRAJECTORIES . . . . .	41
5. SEQUENTIAL PATTERN MINING . . . . .	45

CHAPTER	Page
5.1. ANALYTICS FOR SEQUENTIAL PATTERNS . . . . .	49
6. RANDOM FORESTS . . . . .	51
6.1. RANDOM FOREST PROXIMITY MEASURE . . . . .	54
6.2. UNSUPERVISED LEARNING WITH RANDOM FORESTS . . . . .	55
6.3. CONDITIONAL INFERENCE TREE FORESTS . . . . .	56
6.4. OUTLIER DETECTION USING RANDOM FORESTS . . . . .	57
7. ISOLATION FOREST . . . . .	57
8. CONTROL CHARTS . . . . .	58
3 DETECTING CHANGES IN TRAJECTORIES WITH TIME . . . . .	60
1. MODELING TRAJECTORIES USING HIDDEN MARKOV MODELS FOR CHANGE DETECTION . . . . .	62
2. CHANGE DETECTION OF A HMM WHEN THE PARAMETER AF- TER CHANGE IS KNOWN . . . . .	68
2.1. EXPERIMENTS FOR CHANGE DETECTION WHEN THE PA- RAMETER AFTER CHANGE IS KNOWN . . . . .	71
3. CHANGE DETECTION OF A HMM WHEN THE PARAMETER AF- TER CHANGE IS UNKNOWN . . . . .	73
3.1. EXPERIMENTS FOR CHANGE DETECTION WHEN THE PA- RAMETER AFTER CHANGE IS UNKNOWN . . . . .	75
4. DETECTING SPECIFIC PARAMETER CHANGES IN A HMM . . . . .	76
4.1. m-BAUMWELCH ALGORITHM . . . . .	78
4.2. CHANGE DETECTION USING m-BAUMWELCH ALGORITHM	80

CHAPTER	Page
4.3. EXPERIMENTS FOR CHANGE DETECTION USING m-BW ALGORITHM . . . . .	81
5. CHANGE DETECTION IN TRAJECTORY DATA . . . . .	85
5.1. DATA GENERATION . . . . .	88
5.2. EXPERIMENTS . . . . .	91
6. CONCLUSIONS . . . . .	98
4 DETECTING CHANGES IN PATTERNS OCCURRING IN TRAJECTORY DATA . . . . .	100
1. CHANGE DETECTION IN SEQUENTIAL PATTERNS . . . . .	105
1.1. FREQUENCY-BASED APPROACH . . . . .	109
1.2. DISTRIBUTION-BASED APPROACH . . . . .	111
2. DETECTING CHANGES IN TRAJECTORY PATTERNS . . . . .	115
3. EXPERIMENTS . . . . .	117
3.1. FREQUENCY-BASED APPROACH: LOW LEVEL OF DIS- CRETIZATION . . . . .	118
3.2. FREQUENCY-BASED APPROACH: HIGH LEVEL OF DIS- CRETIZATION . . . . .	131
3.3. DISTRIBUTION BASED APPROACH: HIGH LEVEL OF DIS- CRETIZATION . . . . .	132
3.4. DISTRIBUTION BASED APPROACH: LOW LEVEL OF DIS- CRETIZATION . . . . .	140
5 CLUSTERING AND OUTLIER DETECTION IN SEMANTICALLY EN- RICHED TRAJECTORIES . . . . .	144

CHAPTER	Page
1. SEMANTICALLY ENRICHED TRAJECTORIES . . . . .	151
2. CLUSTERING SEMANTICALLY ENRICHED TRAJECTORIES . . . . .	151
2.1. CLASSIFICATION USING RANDOM FORESTS . . . . .	157
3. OUTLIER DETECTION IN SEMANTICALLY ENRICHED TRAJEC- TORIES . . . . .	160
3.1. WHY RANDOM FORESTS FAIL? . . . . .	161
3.2. OUTLIER DETECTION USING CONDITIONAL INFERENCE FOREST . . . . .	162
4. EXPERIMENTS . . . . .	163
4.1. EXPERIMENTS FOR CLUSTERING . . . . .	163
4.2. EXPERIMENTS FOR OUTLIER DETECTION . . . . .	167
A APPENDIX . . . . .	173
1. Modified Baum Welch Algorithm Implementation (MATLAB CODE) . . . . .	174
2. Creating daily trajectories of cabs (R CODE) . . . . .	176
3. Compressing trajectory data using DP algorithm (R CODE) . . . . .	177
4. Discretizing trajectory data using BIRCH clustering (R CODE) . . . . .	179
5. Clustering trajectories using mixture of regression (R CODE) . . . . .	180
6. HMM for change detection in trajectories (R CODE) . . . . .	180
7. Sequential Pattern Mining (R CODE) . . . . .	185
8. Distribution Based Method (R CODE) . . . . .	187
9. Clustering and Outlier detection using random forests (R CODE) . . . . .	190
REFERENCES . . . . .	192



## LIST OF TABLES

Table	Page
1 Applications of various trajectory techniques . . . . .	4
2 Notation table . . . . .	61
3 Clusters of trajectories of 50 cabs. . . . .	92
4 Notation table . . . . .	101
4 Notation table . . . . .	102
4 Notation table . . . . .	103
4 Notation table . . . . .	104
5 Support count and % support of some patterns in training and testing data. .	119
6 Number of training and testing points used to build each replicate of the control chart using low level of discretization. . . . .	130
7 Support count and % support of some patterns in training and testing data. .	131
8 Number of training and testing points used to build each replicate of the control chart with high level of discretization. . . . .	131
9 Notation table . . . . .	147
9 Notation table . . . . .	148
9 Notation table . . . . .	149
9 Notation table . . . . .	150
10 Numeric variables used in clustering experiments . . . . .	165
11 Categorical variables used in clustering experiments . . . . .	165

## LIST OF FIGURES

Figure	Page
1	Representation of trajectory in a 2D space as points sampled at equal time intervals [1]. . . . . 9
2	Motif based feature extraction of trajectories [2]. . . . . 10
3	Approximate representation of a trajectory using DP algorithm [3]. . . . . 11
4	Examples of 2D trajectories [4]. . . . . 12
5	Trajectories of estimated vertical position of moving hand as a function of time, estimated from six different video sequences [5]. . . . . 13
6	An example of partition and group framework for trajectory clustering [6]. . 14
7	A sample ASL dataset showing one representative trajectory for each class [7]. 16
8	Trajectory representation in 2D image coordinates [8]. . . . . 17
9	Classifying vessel trajectories based on the types of features extracted [9]. . 18
10	An example of an outlying sub-trajectory [10]. . . . . 20
11	Example depicting the issues related to finding periods from movement data [11]. . . . . 24
12	Real bald eagle data [11]. . . . . 25
13	Example of a Markov Process [12]. . . . . 27
14	Example of a Hidden Markov model [12]. . . . . 29
15	An example explaining the SMoT algorithm [13]. . . . . 35
16	Example showing unknown stops identified by CB-SMOT [14]. . . . . 38
17	An example explaining the CB-SMoT algorithm [14]. . . . . 38
18	Semantic annotation of raw trajectories [15]. . . . . 39
19	The location category hierarchy graph [16]. . . . . 40

Figure	Page
20	Examples of three semantic trajectories [17]. . . . . 42
21	Sequence database of patients visiting a hospital [18]. . . . . 47
22	A typical control chart [19]. . . . . 58
23	Translation of trajectory 2 so as to start from time 0. . . . . 63
24	Modeling trajectories using HMM. The regions $R_1, R_2, R_3, R_4$ and $R_5$ are the dense regions in space traversed by the four objects. . . . . 64
25	Learning the parameters of the HMM using the trajectories. . . . . 65
26a	Trajectory of daily commute of a person in May . . . . . 66
26b	Trajectory of daily commute of the person in June . . . . . 66
27a	Transition probabilities in May. The transition probability from state 1 to state 2 is 0.95 and that from state 1 to state 3 is 0.01. . . . . 67
27b	Transition probabilities in June. The transition probability from state 1 to state 2 is 0.01 (compared to 0.95 in May) and that from state 1 to state 3 is 0.95 (compared to 0.01 in May). . . . . 68
28a	Transition probability matrix in terms of $\Theta$ for the case when parameter after change in known. . . . . 71
28b	Emission probabilities for the case when parameter after change in known. . 72
29	Likelihood ratio statistic $L(t)$ versus observation number or $t$ . The first 50 observations correspond to null case parameters while the next 50 observa- tion correspond to shifted case. The statistic $L(t)$ begins to decrease after 50th observation confirming the change. . . . . 72
30a	Transition probability matrix under null case (Left) and shifted case (Right) for change detection when parameter after the change is known. . . . . 75

30b	Emission probability matrix under null & shifted case for change detection when parameter after the change is known. . . . .	76
31a	Likelihood ratio statistic $L(t)$ versus observation number with a window size $w = 25$ . The first 1000 observations correspond to null case parameters while the next 1000 observation correspond to shifted case. The statistic $L(t)$ begins to increase after 1000th observation confirming the change. . .	77
31b	Likelihood ratio statistic $L(t)$ versus observation number with a window size $w = 50$ . The first 1000 observations correspond to null case parameters while the next 1000 observation correspond to shifted case. The statistic $L(t)$ begins to increase after 1000th observation confirming the change. . .	77
32a	TPM: Null case (Left) & Emission: Null and Shifted cases (Right) for change detection using m-BW algorithm with two hidden states. . . . .	81
32b	TPM: Shifted case I (Left) & Shifted case II (Right) for change detection using m-BW algorithm with two hidden states. . . . .	82
33	Likelihood ratio statistic $L_r(t)$ versus observation number for detecting case I shift. The first 100 observations correspond to null case parameters while the next 100 observation correspond to shifted case I. The statistic $L_1(t)$ begins to increase after 100th observation confirming the change in the first row of the TPM. The statistic $L_2(t)$ doesnot change for the 200 observations because the second row of the TPM doesnot change. . . . .	82

Figure	Page	
34	Likelihood ratio statistic $L_r(t)$ versus observation number for detecting case II shift. The first 100 observations correspond to null case parameters while the next 100 observation correspond to shifted case II. The statistic $L_2(t)$ begins to increase after 100th observation confirming the change in the second row of the TPM. The statistic $L_1(t)$ doesnot change for the 200 observations because the first row of the TPM does not change. . . . .	83
35a	TPM: Null case TPM (Left) & Shifted case TPM (Right) for change detection using m-BW algorithm with four hidden states. . . . .	83
35b	TPM: Null & Shifted case Emission probabilities for change detection using m-BW algorithm with four hidden states. . . . .	84
36	Likelihood ratio statistic $L_r(t)$ versus observation number for $r = 1, 2, 3, 4$ . The first 100 observations correspond to null case parameters while the next 100 observation correspond to shifted case. The plot shows the $L_r(t)$ values for only 20 observations before the change and 20 observations after the change. The statistic $L_r(t)$ when $r = 1$ is higher for the 20 observations after the change compared to the 20 before the change, thus confirming the change. . . . .	85
37	Modeling framework for change detection in trajectory data. . . . .	86
38	Variation of likelihood of trajectory data with number of states of the HMM.	89
39	Sample trajectories belonging to cluster 2. Clustering is done using the mixture of regressions model. . . . .	93

40a	Change detection of trajectories from cluster 1 to cluster 2 using window size of 10. The first 59 trajectories belong to cluster 1 while the next 79 trajectories belong to cluster 2. An increase in the likelihood ratio value after the 59th trajectory confirms the change. . . . .	94
40b	Change detection of trajectories from cluster 1 to cluster 2 using window size of 25. The first 59 trajectories belong to cluster 1 while the next 79 trajectories belong to cluster 2. An increase in the likelihood ratio value after the 59th trajectory confirms the change. . . . .	94
41	Sample likelihood ratio i.e. $L(t)$ vs $t$ plots using trajectories from cluster 1 as the training data and trajectories from other clusters at the testing data. . . . .	95
42	Run length values to get an out-of-control signal using control charts for different combination of training and testing trajectories with window size of 10. . . . .	96
43	Run length values to get an out-of-control signal using control charts for different combination of training and testing trajectories with window size of 25. . . . .	97
44	Autocorrelation between $L(t)$ values of in-control data for cluster 1 trajectories for a window size of 25. . . . .	98
45	Sequence database consisting of 10 sequences and the set of frequent sequential patterns with minimum support of 0.5 extracted from them. . . . .	105
46	An example depicting a trajectory which is a sequence of GPS points being denoted as a sequence of items. . . . .	107

Figure	Page
47	Modeling framework of the frequency-based method for change detection in sequential patterns. Subsets are extracted from the training data by sampling without replacement. Frequency based statistics obtained from these subsets are compared with the test data to detect changes. . . . . 108
48	Modeling framework of the distribution-based method for change detection in sequential patterns. Subsets are extracted from the training data by sampling without replacement. Time-interval based statistics obtained from these subsets are compared with the test data to detect changes. . . . . 113
49a	Relative support versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with $R = 25$ . The change in relative support from the training to testing data illustrates the pattern change. . . . . 120
49b	Relative support versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with $R = 50$ . The change in relative support from the training to testing data illustrates the pattern change. . . . . 121
49c	Relative support versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with $R = 100$ . The change in relative support from the training to testing data illustrates the pattern change. . . . . 122
50a	Information gain versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with $R = 25$ . The change in information gain from the training to testing data illustrates the pattern change. . . . . 123
50b	Information gain versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with $R = 50$ . The change in information gain from the training to testing data illustrates the pattern change. . . . . 124

Figure	Page
50c	Information gain versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with $R = 100$ . The change in information gain from the training to testing data illustrates the pattern change. . . . . 125
51	Growth rate versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with $R = 100$ . The change in growth rate from the training to testing data illustrates the pattern change. . . . . 126
52	Average run length and standard errors calculated for NULL case (no change in the support of the pattern) using pattern $P2$ . Both the $2\sigma$ and $3\sigma$ limits are reported using the different measures. The $3\sigma$ limits are calculated by truncating the run length to 400, if an out-of-control signal doesn't occur for 400 testing points. . . . . 128
53	Average run length values for different $(pattern, R, measure)$ combinations. All the run lengths are calculated using $3\sigma$ limits. . . . . 129
54	Average run length values for different $(pattern, R, measure)$ combinations and high level of discretization. All the run lengths are calculated using $3\sigma$ limits. . . . . 132
55a	Hotelling $T^2$ chart of group statistic versus sample index for pattern $P1$ using $\delta = 1.2$ and $R = 100$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change. . . . . 134



- 55b Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P1$  using  $\delta = 1.3$  and  $R = 100$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change. . . . . 135
- 55c Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P1$  using  $\delta = 1.5$  and  $R = 100$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change. . . . . 136
- 56 Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P1$  using  $\delta = 1.2$  and  $R = 50$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change. . . . . 137
- 57a Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P2$  using  $\delta = 1.2$  and  $R = 100$ . The change in the statistic value from the train data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change. . . . . 138
- 57b Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P2$  using  $\delta = 1.3$  and  $R = 100$ . The change in the statistic value from the train data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change. . . . . 139

58	Hotelling $T^2$ chart of single observation versus sample index for pattern $P1$ using $\delta = 1.2$ and $R = 100$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change. . . . .	140
59	The ARL values obtained for pattern $P1$ using the distribution based method for different combinations of ( $R$ , Control chart method, % Change in time between items in the pattern). . . . .	141
60	The ARL values obtained for pattern $PR1$ and $PR2$ with repeating items, using the distribution based method for different levels of % change in time between items in the patterns. . . . .	142
61	Modeling framework for clustering and outlier detection in semantic trajectories. . . . .	152
62	Two similar semantic trajectories belonging to a cluster. . . . .	153
63	Representation of a semantic trajectory using the variables $s_{ij}$ and $m_{uv}$ corresponding to the stops and moves of the trajectory. . . . .	154
64	Transforming the trajectory data to a two class problem. Data $C_0$ belonging to class 0 consists of 100 records and data $C_1$ belonging to class 1 consists of 100 records. The values for the variables in data $C_1$ depend on the corresponding variables in $C_0$ . . . . .	155
65	Heat map of the trajectory data represented using the variables $s_{ij}$ and $m_{uv}$ . The rows corresponds to the trajectories and the columns correspond to the variables. Only first 20 trajectories belonging to three clusters are shown, where each cluster consists of 100 trajectories. . . . .	164

66	Dendrogram of the clusters obtained by hierarchical clustering. The three clusters in the trajectory data are clearly visible in the dendrogram. . . . .	166
67a	Breiman outlier scores of 207 trajectories with $\delta = 0.5$ and $r = 1$ . The scores for the 7 outlying trajectories with IDs 201 to 207 are higher than the rest of the trajectories. . . . .	168
67b	Breiman outlier scores of 207 trajectories with $\delta = 2.5$ and $r = 1$ . Comparing with Fig. 67a, higher noise in the trajectories ( $\delta = 2.5$ ), makes it difficult to identify the outlying trajectories with IDs 201 to 207, compared to lower noise ( $\delta = 0.5$ ). . . . .	169
68a	10 trajectories with highest outlier scores for $\delta = 0.5$ and $r = 1$ . The outlying trajectories with IDs 201 to 207 are among the 10 trajectories with highest scores. . . . .	169
68b	10 trajectories with highest outlier scores for $\delta = 2.5$ and $r = 1$ . The outlying trajectories with IDs 201 to 207 are among the 10 trajectories with highest scores. . . . .	170
69a	Breiman outlier scores for 207 trajectories with $\delta = 0.5$ and $r = 2$ . Comparing with Fig. 67a, higher value of $r$ ( $r = 2$ ), makes it difficult to identify the outlying trajectories with IDs 201 to 207, compared to lower value of $r$ ( $r = 1$ ). . . . .	171
69b	10 trajectories with highest outlier scores with $\delta = 0.5$ and $r = 2$ . Out of the 7 outlying trajectories with IDs 201 to 207, only 6 are among the top 10 trajectories with high outlier scores. . . . .	172

70 TPR and FPR values obtained for different thresholds and varying parameter values  $r$  and  $\delta$ . . . . . 172

## CHAPTER 1

### INTRODUCTION

Trajectories are collected by recording the location of motion in two or three dimensional space of human beings, animals etc., at multiple instants of time. Moving object trajectory analysis is a growing area of interest these days owing to its applications in various domains. Some of the areas where trajectory analysis plays an important role are marketing, production systems, surveillance for security purposes, traffic monitoring and management, social media etc.

Analysis of movement data in security applications like analysis of data from the sensors, analysis of images produced by security cameras can be useful to detect anomalies among the various trajectories for intrusion detection. RFID technology installed in goods can improve the service quality of e-business with better tracking of shipment. Collecting urban trajectories of vehicles can be used to derive useful knowledge for optimizing traffic management like predicting areas of high traffic so as to reroute the traffic for lesser congestion in these areas. Trajectory analysis can also be used to derive useful knowledge from social media like popular photo sharing websites like Facebook, Flickr etc. Spatio-temporal information is captured in the form of GPS coordinates and time the photograph was taken by the low-cost GPS chips in cell phones and cameras and is saved in the header of image files. Trajectory pattern mining of this user generated spatio-temporal data can be used to explore the wisdom of the crowd embedded in the social media. Another application of trajectory analysis is relating to non-physical movement in a metaphorical sense where an object is moving in an abstract space whose points change with different values of a time-varying attribute. For example, time series data of price attribute of a product can be modeled as a trajectory.

Trajectories can be viewed as a line that a moving object traces in the geometric space with time. This notation of a trajectory in space and time focuses on the spatio-temporal features of the trajectory. Apart from the geometric features of a trajectory, there are high level semantic features also associated with it. Modern GPS, mobiles and wireless sensing technologies allow us to record the continuous movement of an object as a function of time, in a specific time interval of the object's lifespan. The travelling object thereby produces countless spatio-temporal trajectories during its lifespan e.g. employees commuting from office to home on a daily basis. From an application perspective, a trajectory has semantic features associated with it apart from the geometric features. For example, the semantic information associated with the daily trip of employees would be related to the transportation means used during the trip and whether any carpool facilities were used in the trip. A conceptual model is needed for describing the trajectories which would include both trajectory as a sequence of spatio-temporal records of a moving object as well as associating semantic annotations to trajectories like attributes of the trajectory, links between trajectory and other objects (like obstacle in the path) etc.

Research on trajectory analysis has enabled us to develop tools and techniques for various tasks. Some of the areas of significant research include

- Trajectory classification: Model construction for predicting the class labels of moving objects based on their trajectories and other features.
- Trajectory clustering: Grouping a set of moving objects into clusters based on the similarity of their trajectories, thus discovering common trajectories.

- Trajectory anomaly detection: Automatic identification of abnormal or suspicious moving objects from a massive set of moving object trajectories.
- Trajectory ranking and diversification: Ranking the frequent trajectory patterns to identify the important trajectories and diversification to explore diverse routes in trajectories.
- Trajectory convoy detection: Finding moving object clusters i.e to find a group of moving objects that are travelling together sporadically.
- Trajectory anonymization: Properly anonymizing the trajectories before being released to public use to secure sensitive information as removing personally identifying information would not be enough.
- Semantic annotation: Enriching trajectories with semantic annotations allowing users to attach semantic data to specific parts of the trajectory.
- Mining periodic behavior: Finding repeating activities at certain locations with regular time intervals.
- Trajectory prediction: Predicting with certain accuracy the next location of moving object.
- Trajectory location correlations: Learning location correlation from human trajectories that states the relations between geographic locations in the space of human behavior.

Trajectory analysis finds applications in a wide range of areas from bioengineering to traffic management. Understanding the existing techniques and developing new methodol-

**TABLE 1.** Applications of various trajectory techniques

Method	Applications
Trajectory clustering	Finding similar multiple attribute response curves in drug therapy, Common behavior of hurricanes near coastline, Common behavior in animal movement data with varying traffic rate, Vehicle position data.
Trajectory classification	Vessel classification from satellite images, Classification of trace gas measurements, Video surveillance, Pattern recognition e.g in sign language and gesture recognition, Bioengineering for gene expression classification.
Trajectory outlier detection	Detection of credit card fraud, Monitoring criminal activities in electronic commerce, Hurricane tracking , Animal movement tracking, Detect breaks or delays in supply chain
Trajectory pattern ranking	Trip planning, Search result diversification, Recommendation systems.
Location correlation	Location recommendation, Sales promotion (Supply chain), Bus routes design, Mobile tour guides.
Swarm pattern detection	Animal tracking, Carpooling, Throughput planning of trucks in supply chain, Military applications to monitor the troop that move in parallel and merge/evolve over time, Intelligence and counterterrorism services to identify suspicious activity of individuals moving similarly.
Semantic annotation	Mining user similarity based on location history, Tourism applications like finding frequent visited places, relation between visited places, sequence of tourist places visited.
Trajectory pattern mining	Traffic management, Homeland Security (e.g., border monitoring), Location-based service, Recommendation systems (e.g., suggesting trajectories of frequently visited places to tourists), Law enforcement (e.g. video surveillance).

ogy for trajectory analysis are gaining increasing importance due to the numerous applications. Table 1 summarizes the different trajectory analysis methods and their applications in various domains.



## 1. CONTRIBUTIONS

This section provides an overview of the scientific contributions of the thesis to trajectory analysis.

**Contributions to trajectory change detection:** We have proposed a novel algorithm, m-BaumWelch for parameter estimation in Hidden Markov Models. The Baum-Welch algorithm is a class of expectation-maximization algorithms, which computes the maximum likelihood estimates of the parameters (transition and emission probabilities) of a HMM, when given only emissions as training data. The m-BaumWelch algorithm that we have proposed enforces constraints for parameter estimation in a HMM. Unlike the original algorithm which estimates all the parameters of the HMM, the m-BaumWelch algorithm fixes the specific parameters which have to be estimated and leaves the remaining parameters unaltered. The m-BaumWelch algorithm is based on the principles of the original Baum-Welch algorithm. It uses the expectation-maximization principle and computes the parameter estimates by maximizing the likelihood values for the parameters of the HMM given the observation sequence. We have integrated the m-BaumWelch algorithm to develop a model for detecting changes in trajectory data with time. For the purpose of change detection in trajectories, we have modeled trajectories using a Hidden Markov Model (HMM). The frequent regions in the trajectories are modeled as hidden states and the observation sequences are the observations of the HMM. Real trajectory data obtained from vehicles was used to test the method for change detection.

**Contributions to frequent pattern mining:** We have proposed a technique, by making use of various statistics like support, confidence etc. for detecting changes in frequent trajectory patterns occurring in streaming data. Our technique is used to detect if the frequent

sequential patterns occurring in a data stream change when new data arrives. We aim to answer two important questions: Are the frequent patterns still frequent in the new data? If they are frequent, has the distribution of items in the pattern changed? We have illustrated our method to detect changes in frequent trajectory patterns occurring in vehicle data. We have addressed this problem using the following two approaches. In the frequency based approach, the statistics such as relative support, relative confidence etc. for each frequent sequential pattern in the original data are used to distinguish the patterns in the new data. This method is used to detect if the previously found frequent patterns are still frequent in the new data. In the distribution based approach, the distribution of the time between the items in the frequent patterns is used to detect changes in the new data. Using the distribution based approach, we can detect if the time interval distribution between items in the patterns in the new data has changed from the distribution in the original data.

**Contributions to semantic trajectories:** A trajectory is typically represented as a discrete sequence of points. Recently a new trajectory concept, called semantic trajectory, has been introduced in [20] which defines trajectories from a semantic point of view. A semantic trajectory is defined as a sequence of stops and moves. Stops are the important parts of a trajectory where the moving object has stayed for a minimal amount of time. Moves are the sub-trajectories describing the movements between two consecutive stops. Based on the concept of stops and moves, the user can enrich trajectories with semantic information according to the application domain. In this thesis, we propose a modeling framework for analyzing semantic trajectories. We have used classification techniques such as decision trees for developing a framework to cluster semantic trajectories. The most important challenge with clustering semantic trajectories is that they have both numeric and categorical

attributes. Another problem to be addressed while clustering such trajectories is that trajectories can be of different lengths and also have missing values. Our method was able to overcome these challenges as decision trees can efficiently handle mixed attributes and missing values. We had extended this method for outlier detection in semantic trajectories. Decision trees were not able to detect outliers in trajectories due to the bias introduced in selecting variables while splitting. Alternate tree growing strategies were explored to detect the outlying trajectories.

## CHAPTER 2

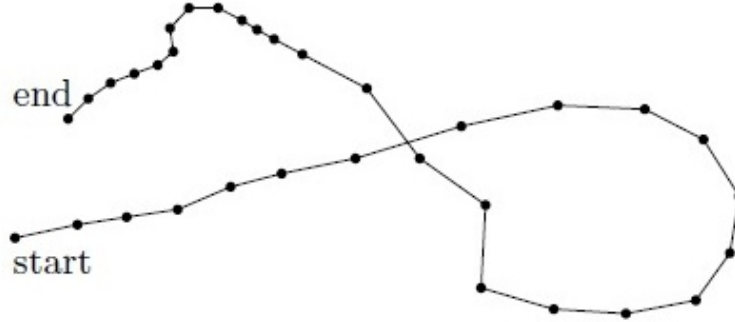
### BACKGROUND

In this chapter, we give an overview of the related work on trajectory analysis. With a constant increase of moving object data, a lot of work has been done for modeling, processing and mining trajectories, to understand and find patterns in this data.

#### 1. PREPROCESSING TRAJECTORY DATA

Trajectories can be viewed as a line that a moving object traces in the geometric space with time. This notation of a trajectory in space and time focuses on the spatio-temporal features of the trajectory. The trajectories are thus represented as a sequence of points ordered in time i.e.  $(x_t, y_t)$  where  $t = 1, 2, \dots, n$ . A lot of data mining tasks can be performed on the trajectory data which is represented as sequence of points ordered in time. Some of them include trajectory clustering, trajectory classification, trajectory outlier detection, trajectory pattern mining and mining periodic behavior in trajectories. To better understand mobility data, applications need to apprehend the semantics of the trajectory. The geometric points  $(x, y)$  in a trajectory can be transformed using the concept of stops and moves. For example, a move from  $(x_1, y_1)$  to  $(x_2, y_2)$  can be transformed to a move from *(Tempe)* to *(Phoenix)* by incorporating the geographic knowledge into the trajectory data. Preprocessing of trajectories is required for analyzing the data depending on the application.

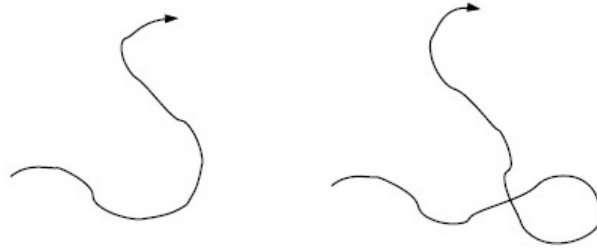
A trajectory may have a long and complicated path. Hence, even though some portions of trajectories show a common behavior, the whole trajectories might not. Discovering common sub-trajectories is very useful in many applications, especially if we have regions of special interest for analysis. Spatio-temporal trajectories can be segmented to partition the trajectory into a number of sub trajectories, so that the movement characteristics in each segment are uniform in some sense. In [1], Buchin et al. have addressed the problem of



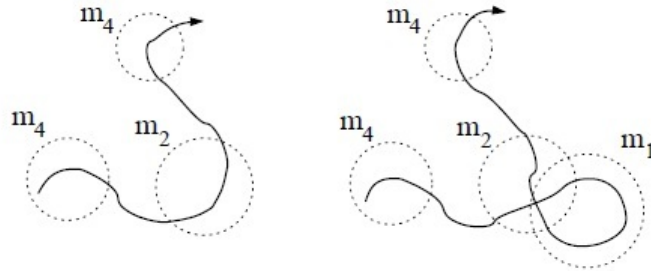
**Figure 1.** Representation of trajectory in a 2D space as points sampled at equal time intervals [1].

segmenting a trajectory into homogeneous segments based on velocity, speed, curvature, sinuosity, and curviness. The idea of the segmentation is to obtain segments where movement characteristics like speed, curviness etc. are uniform in some sense. They have represented the trajectory as a sequence of points sampled with equal time intervals as shown in Fig.1 [1].

Li et al. [2] have proposed a framework for anomaly detection in trajectories, where object trajectories are expressed using discrete pattern fragments called motifs. A motif is a prototypical movement pattern, which include additional spatiotemporal attributes. Examples of motifs include e.g. *right turn*, *u-turn*, and *loop*. Fig.2 [2] explains their approach of motif extraction from trajectories represented in a two dimensional space. Consider the two trajectories in Fig.2(a) [2], which have similar shapes except for an extra loop for the one on the right. Such trajectories would be difficult to be differentiated by distance measures like Euclidean distance. Once the motifs are extracted from the trajectories, they are used as features to develop a rule based classifier to classify the trajectories. This classifier is



(a) Two similar trajectories. The loop in the right trajectory is difficult to handle in holistic approaches.

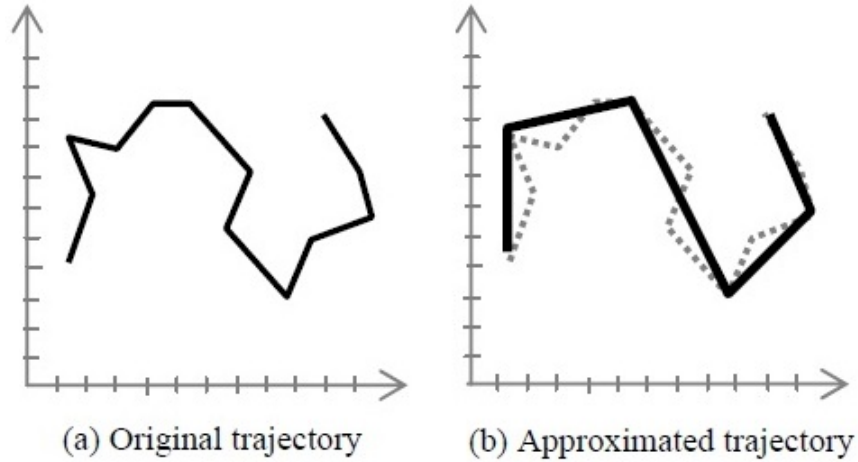


(b) Same two trajectories after motif extraction. The right trajectory has an extra  $m_1$ .

**Figure 2.** Motif based feature extraction of trajectories [2].

used to classify the trajectories into two classes - *normal* and *abnormal*, thereby detecting the anomalous trajectories.

Another important preprocessing step for trajectory data is to compress the trajectory data for data reduction. Given a trajectory or a curve composed of specific number of points, the DouglasPeucker (DP) [3] algorithm tries to find a similar curve with fewer subset of points. The algorithm recursively divides a trajectory. The input to the algorithm is all the points between the first and the last point of the trajectory. The details are as follows: the algorithm first marks the first point  $f$  and last point  $l$  of the trajectory. It then finds the point  $p$  that is furthest from the line segment formed by the first and the last points as the end points. The distance of the farthest point  $p$ , is compared to a threshold, and if this distance

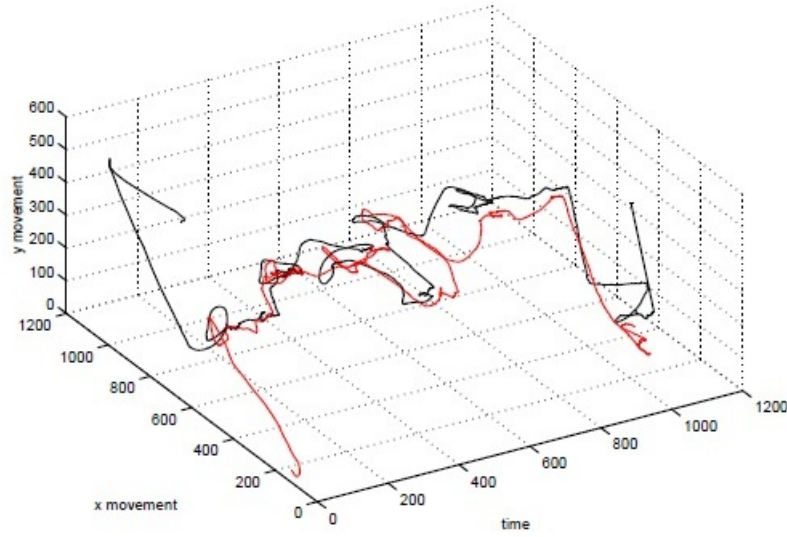


**Figure 3.** Approximate representation of a trajectory using DP algorithm [3].

is greater than the threshold, the algorithm recursively calls itself by joining the first point  $f$  and the worst point  $p$ , and then with the worst point  $p$  and the last point  $l$ . At the end of the recursion, a new curve is outputted which consists of only a subset of the initial points. An example of reducing the size of a trajectory using the DP algorithm is shown in Fig. 3 [3], where Fig. 3(a) [3] shows the original trajectory and Fig. 3(b) [3] shows the approximated trajectory after the DP algorithm is applied.

## 2. ANALYTICS FOR TRAJECTORIES

A lot of techniques have been discovered in data mining literature for performing various tasks like clustering, anomaly detection, classification etc. These techniques have been applied to trajectory data to extract useful information and patterns in the data. Most of these techniques were developed by considering trajectories as a sequence of points ordered in time i.e.  $(x_t, y_t)$  where  $t = 1, 2, \dots, n$ , of a moving object. In this section, we discuss the analytic models developed for the trajectories which are represented as a se-



**Figure 4.** Examples of 2D trajectories [4].

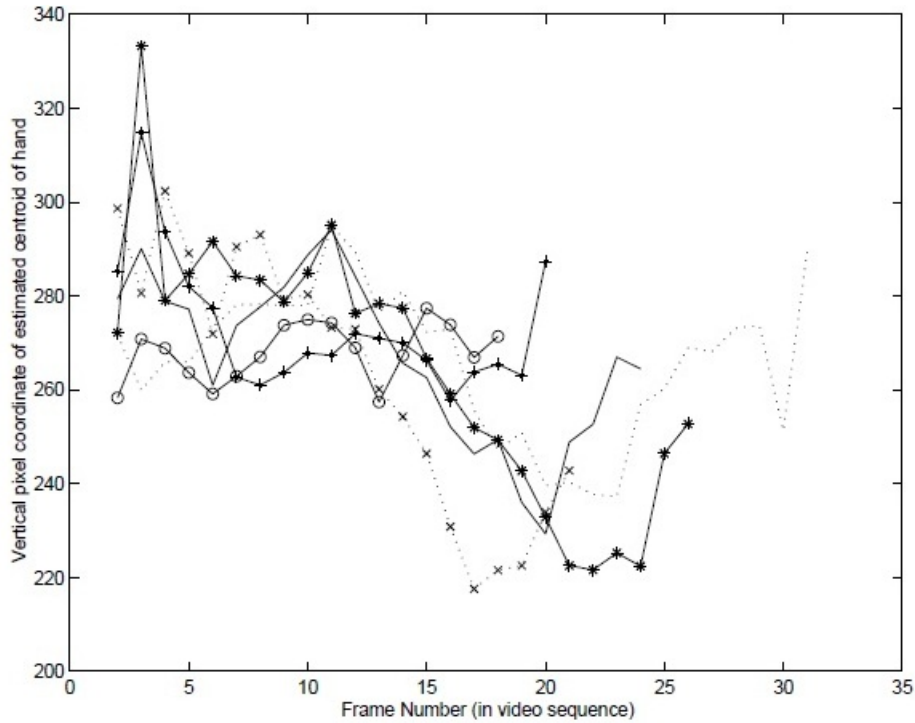
quence of geometric points. Detailed discussion about semantic trajectories will follow in the later sections.

## 2.1. TRAJECTORY CLUSTERING

Clustering is a process of grouping a set of similar objects together. There has been a lot of research in the clustering algorithms which include  $k$ -means [21], BIRCH [22], DBSCAN [23], OPTICS [24], STING [25] etc. Preliminary research on clustering dealt with clustering of point objects. There is an increasing interest to cluster trajectories of moving objects so as to find groups of objects that moved in a similar way.

Vlachos et al. [4] have investigated techniques for discovering similar multidimensional trajectories, where they have considered trajectories as a sequence of points in a three-dimensional space i.e.  $(x_t, y_t, z_t)$  where  $t = 1, 2, \dots, n$ . They modelled the trajectory as a sequence of consecutive locations in a multidimensional (generally two or three dimen-

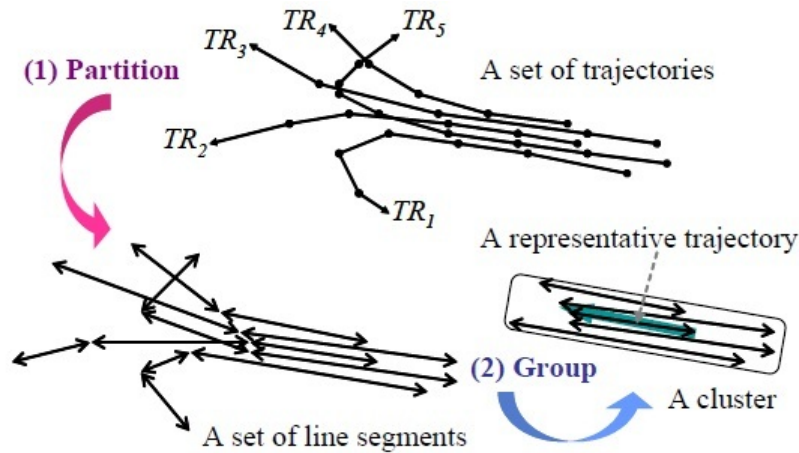




**Figure 5.** Trajectories of estimated vertical position of moving hand as a function of time, estimated from six different video sequences [5].

sional) Euclidean space as shown in Fig.4. Trajectories of the form represented in Fig.4 [4] arise in many applications where an object motion is recorded over time. Examples include features extracted from video clips, tracking animal motion, mobile phone usage data etc. Since trajectory data consists a lot of noise, they formalized non-metric similarity functions based on the Longest Common Subsequence (LCSS), which are very robust to noise, to measure similarity between trajectories. They also compared their method with widely used *Euclidean* and *Dynamic Time Warping (DTW)* [26] distance.

Gaffney et al. [5] proposed a model-based clustering algorithm for trajectories where a set of trajectories is represented using a mixture of regression model. The measurements of a specific quantity (e.g. vertical position of a moving hand) is considered to be the response



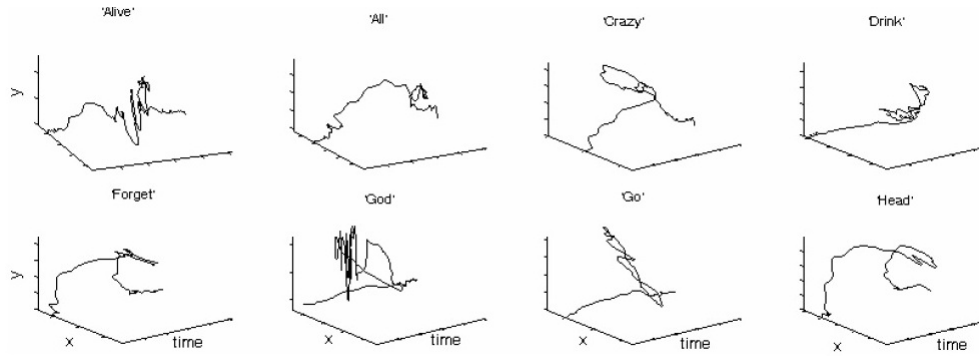
**Figure 6.** An example of partition and group framework for trajectory clustering [6].

variable  $y$ , which is measured as a function of an independent variable  $t$  (e.g. time). An example of such data is shown in Fig.5 [5] for a set of 6 different  $y$  measurements. Each curve represents the estimated trajectory of a particular individual performing a particular hand movement. This type of data can occur in variety of applications where repeated measurements are available on individual objects over time. Some of the complications that arise in this kind of data which make it difficult to apply standard clustering techniques are that the trajectories are of different length. Hence, because of the varying length of the trajectories, one cannot simply convert the trajectories to fixed length vectors and apply a clustering technique such as the  $K$ -means algorithm in a fixed dimensional space. Gaffney et al. [5] have introduced a probabilistic mixture regression model for such data and specifically used EM algorithm [27] to cluster the trajectories. Given the measurements  $y$  which are a function of some known  $t$ , each trajectory is described using a conditional regression model  $f_k(y|t, \theta_k)$  which gives the probability that  $y$  belongs to the  $k$ th group with parameters  $\theta_k$ .

The existing trajectory clustering techniques group similar trajectories as a whole. But Lee et al. [6] proposed that clustering trajectories as a whole could miss common sub-trajectories. Clustering sub-trajectories finds applications in many areas e.g. meteorologists are interested in common behavior of hurricanes near the coastline or at the sea which requires analyzing only a part of the hurricane trajectory in the regions of interest. They have represented trajectories as a sequence of multidimensional points i.e  $TR = \{p_t\}$  for  $t = 1, 2, \dots, n$ , where  $p_t \forall t$  is a  $d$ -dimensional point i.e. they can be either two or three dimensional. The length of the trajectory  $len$  can be different for various trajectories. Lee et al. [6] have developed a *partition-and-group* framework as shown in Fig.6 [6] for clustering trajectories, which partitions a trajectory into a set of line segments, and then groups similar line segments together into a cluster. They developed a trajectory clustering algorithm *TRACCLUS* which consists of two phases: *partitioning* and *grouping*. In the first phase, partitioning is done using Minimum Description Length (MDL) principle [28]. In the second phase, a density based line-segment clustering algorithm based on DBSCAN is presented for clustering the trajectories. This algorithm discovers common sub-trajectories from trajectory data unlike the previous work where trajectories are clustered as a whole. They have employed a density based clustering technique as density based methods are suitable for line segments because they can discover clusters of arbitrary shape and filter out noises.

## **2.2. TRAJECTORY CLASSIFICATION**

Trajectory classification has many important, real-world applications. It is broadly defined as model construction for predicting the class labels of moving objects based on their



**Figure 7.** A sample ASL dataset showing one representative trajectory for each class [7].

trajectories and other features. As an example consider Fig.7 [7] where trajectories are classified into different classes from an Australian Sign Language (ASL) dataset. The figure depicts one representative trajectory for each class for 8 different classes (e.g. ‘Alive’, ‘Crazy’ etc.) in the dataset.

Most proposed methods employ the Hidden Markov Model(HMM) [29] and use whole trajectories for classification. Bashir et al. [7] presented a framework of classifying human motion trajectories, which uses the HMM with a mixture of Gaussians. In their classification system, trajectories are segmented at points of change in curvature and the subtrajectories are represented by their Principal Component Analysis (PCA) [30] components. They have addressed two major issues with regard to post processing of the trajectory data. In most cases e.g. video tracking applications, full trajectory information is often unavailable due to occlusions. As a result of this limitation, trajectory representation methods should be able to perform well even in the case of partial trajectory information. Bashir et al. [7] have addressed this problem by segmenting the trajectories at points of perceptual discontinuities. The discontinuities in the trajectory are detected with the help of velocity (first derivative) and acceleration (second derivative). The other concern while modeling

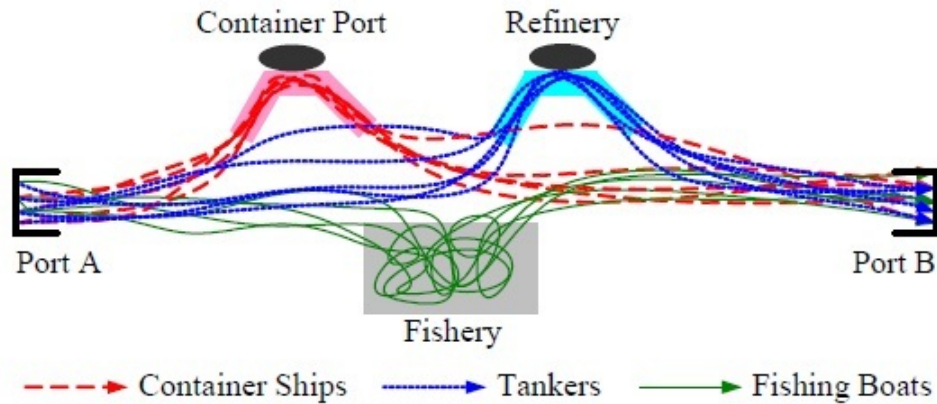


**Figure 8.** Trajectory representation in 2D image coordinates [8].

trajectories is its compact representation for efficient distance computation. For this purpose, they use PCA to represent trajectories using a compact set of features to obtain a reduced-dimensional space.

Fraile et al. [31] classify the trajectories of vehicles using HMM to detect accidents or abnormal events. Their method aims at finding low curvature approximations to segments of the trajectories of vehicles. In each segment, the trajectory is approximated by a smooth function and then assigned to one of the four categories: *ahead*, *left*, *right* or *stop*. In this way the list of segments is reduced to a string of symbols drawn from the set  $\{a, l, r, s\}$ , where each letter corresponds to the appropriate category. Once a sequence of measurements is obtained, the trajectory is classified using a HMM. The Viterbi algorithm [32] is then used to find the sequence of internal states for which the observed behaviour of the vehicle has the highest probability.

Trajectory classification has been an active research topic in the fields of bioengineering and video surveillance. Many of proposed methods employ the neural network such as the



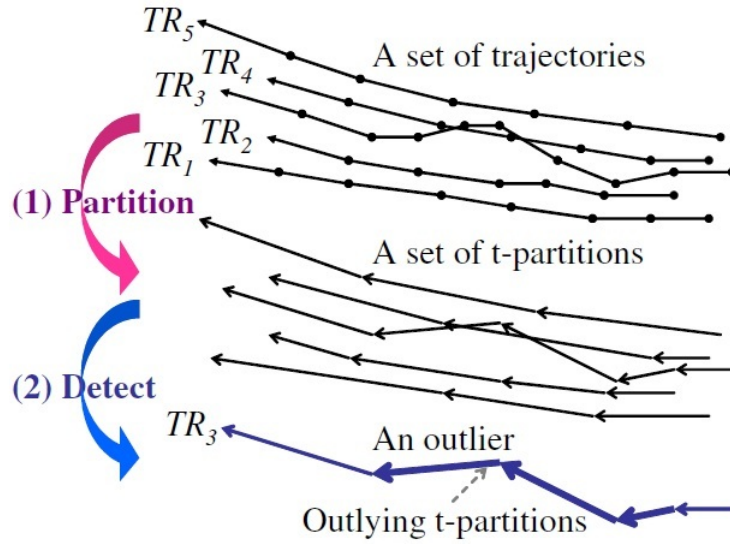
**Figure 9.** Classifying vessel trajectories based on the types of features extracted [9].

Self-Organizing Map (SOM) [33] and use whole trajectories for classification. Sbalzarini et al. [34] have compared various machine learning techniques used for classifying biological motion trajectories. In their work, machine learning techniques,  $k$ -nearest neighbors(KNN), support vector machines and HMMs, were applied to the task of classifying trajectories of moving keratocyte cells. They have represented trajectories of moving cells as position readings at equidistant sampling intervals.

Owens and Hunter [8] proposed a method of detecting suspicious behaviors of pedestrians using a video surveillance system. Trajectories are represented in 2D image coordinates as shown in Fig.8 [8]. The input to the decision maker module is the trajectory of an object, which consists of a series of centroid positions. Each trajectory is encoded to a feature vector using its summary information (e.g., the maximum speed). The trajectory sequence is recorded into a trajectory description vector, and this vector is used as an input for the self organizing feature map neural network. If a trajectory is very complicated, some valuable information could be lost due to this encoding. They have developed a “model-free” ap-

proach i.e. there is no explicit modeling of normal or abnormal behavior, which is instead learned by the neural network.

Recent work on trajectory classification by Lee et al. [9] classifies trajectories by extracting a hierarchy of features by partitioning trajectories. Unlike the previous methods in the literature which use the shapes of whole trajectories for classification, their method addresses classification when discriminative features appear at *parts* of the trajectories (not at whole trajectories). Also, they address the issue of classification when discriminative features appear not only as common movement patterns but also as regions e.g. trajectories which visit a specific region like a fishery in their path belong to the same class even though their trajectories may not have a long common path. Features are extracted by performing region-based as well as trajectory-based clustering. Region-based clustering is used to discover regions that have trajectories mostly of one class regardless of their movement patterns. Trajectory-based clustering is used to discover sub-trajectories that indicate common movement patterns of each class. An example of classification using vessel trajectories that move from port A to port B is shown in Fig.9 [9] where (1) parts of the trajectories near the container port and near the refinery enable us to distinguish between container ships and tankers even if they share common long paths; (2) the trajectories in the fishery enable us to recognize fishing boats even if they have no common path there. They have used the MDL [28] principle for extracting region-based features. The region extraction algorithm based on MDL principle is computationally expensive and further research lies in exploring algorithms for effective classification.



**Figure 10.** An example of an outlying sub-trajectory [10].

### 2.3. TRAJECTORY OUTLIER DETECTION

For many applications like detecting criminal activities in E-commerce, detecting rare instances or outliers would be more interesting than detecting the common or frequent patterns. There are many outlier detection algorithms reported in literature. They can be classified into distribution-based [35], distance-based [36] [37] [38], density-based [39], and deviation-based [40] algorithms. But most of these algorithms have been designed to detect outliers from multidimensional point data.

Markus et al. [39] have introduced a quantitative measure called *local outlier factor* (LOF) for each object in multidimensional dataset, indicating its degree of outlier-ness. Their method is loosely related to density based clustering to detect outliers, where outliers are objects (noise) not belonging to any cluster of the dataset. In [40], Aggarwal et al. developed methods to detect outliers in high-dimensional data. In high dimensional data the notion of proximity is meaningless and every point can be considered to be an outlier



from the perspective of proximity based definitions. They have defined outliers as points which behave very differently or deviate from the average behavior. Their method works by finding lower dimensional projections of the data which are locally sparse.

Very few attempts for trajectory outlier detection exist in the literature. Knorr et al. [37] have developed a method for trajectory outlier detection for 2D motion tracking in video surveillance. They represented trajectory by a set of key features instead of a sequence of points. That is, trajectory is summarized by the coordinates of the starting and ending points; the average, minimum, and maximum values of the directional vector; and the average, minimum and maximum velocities. The distance function is defined as the weighted sum of the difference of these values. A distance-based algorithm is then applied for detecting trajectory outliers. This method however compares the trajectory *as a whole*.

Li et al. [2] have proposed a trajectory outlier detection algorithm based on classification. Trajectories are represented as a sequence of spatiotemporal records of a moving object e.g. GPS records. They extracted common patterns of movement e.g. *right turn, u-turn, and loop* from trajectories using clustering. These patterns form a feature space for the trajectories, which are then fed into a classifier. This algorithm needs a training set to build a classification model, and a new trajectory is classified into “normal” or “outlier” based on this model. It is not always practical to obtain a good training set.

Lee et al. [10] proposed a partition-and-detect framework for trajectory outlier detection. This framework allows us to discover outlying sub-trajectories, whereas previous works do not. A trajectory is partitioned into a set of line segments, and the outlying line segments are detected as trajectory outliers as shown in Fig.10 [10]. Their trajectory outlier detection algorithm takes advantage of both distance-based [36] [37] [38] and density-

based [39] approaches for outlier detection. However they have considered only the spatial information to detect outliers and ignored the temporal information.

#### **2.4. DISCOVERY OF TRAJECTORY SWARM OR CONVOY PATTERNS**

An important analysis of trajectory data involves finding moving objects that travel together. The discovery of such a close cluster of moving objects can find applications in study of animal behaviors, routes planning, and vehicle control. A moving object cluster is defined as a group of moving objects that are geometrically close to each other and are together for some minimum time duration. The difference between finding moving clusters compared to clustering trajectories is that the identity of a moving cluster remains unchanged while its location and content may change over time, e.g. a group of animals migrating in search of food can be thought of as a moving cluster. While searching for food, the group of animals move from place to place, hence the location of the moving cluster of animals changes with time. Also, some animals may leave the group or new animals may enter the group and hence the content of the moving cluster of animals changes i.e. increases or decreasing over time.

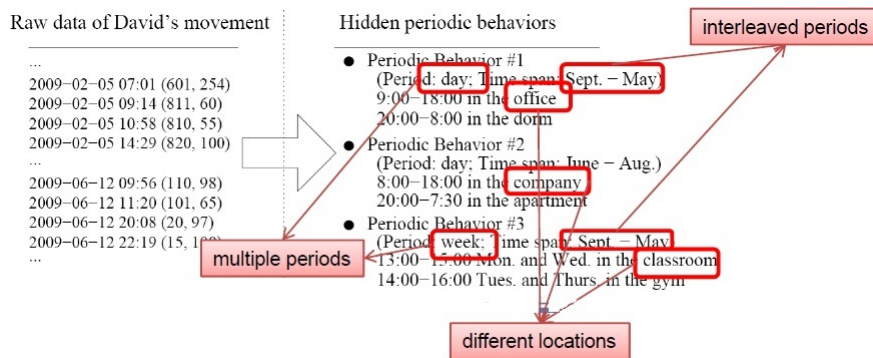
Mining moving object clusters has gained attention recently. Research has been done in find moving object clusters including moving clusters [41], flocks [42], and convoys [43]. A moving cluster as defined earlier is a set of objects that move close to each other for a long time interval. A flock is a group of objects that move together for a long time interval within a disk of some user-specified size. The chosen disk size has a substantial effect on the results of the discovery process. Also, the selection of a proper disc size turns out to be difficult, as situations can occur where objects that intuitively belong together or do not belong together are not quite within any disk of the given size or are within such a disk.

A convoy is used to discover a group objects that move together for a long time interval, by avoiding the rigid restrictions (e.g. disk size as in the case of flocks) on the sizes and shapes of the trajectory patterns to be discovered. Convoys are thus able to capture generic trajectory patterns of any shape and any extent. However, all these patterns require the group of moving objects to be together for at least  $k$  consecutive timestamps, which might not be practical in the real cases. Enforcing the consecutive time constraint may result in loss of interesting moving object clusters.

Li et al. [44] have proposed a general movement pattern of moving object clusters called swarm. A swarm is a group of moving objects that move within arbitrary shape of clusters for certain timestamps that are possibly nonconsecutive. They enable the discovery of interesting moving object clusters with relaxed temporal constraint. The constraint that the objects should stick together for consecutive timestamps is relaxed. This is the case in real life, where a set of moving objects (e.g., birds, flies, and mammals) hardly stick together all the time- they actually diverge temporarily and congregate at certain timestamps. They have represented moving objects as a set of points in a 2D space at different timestamps. As time progresses, the clusters rearrange themselves due to movement of objects to different locations. Discovering swarm patterns in such a dataset involves finding objects that travel together (or belong to same cluster) for most of the time.

## ***2.5. MINING PERIODIC BEHAVIOR IN TRAJECTORIES***

Periodicity is a frequently happening phenomenon for moving objects. A periodic behavior can be loosely defined as repeating activities at specific locations for certain time intervals. Such periodic behavior provide an insightful and concise explanation over the long moving history. Periodic behaviors are also useful for compressing movement data

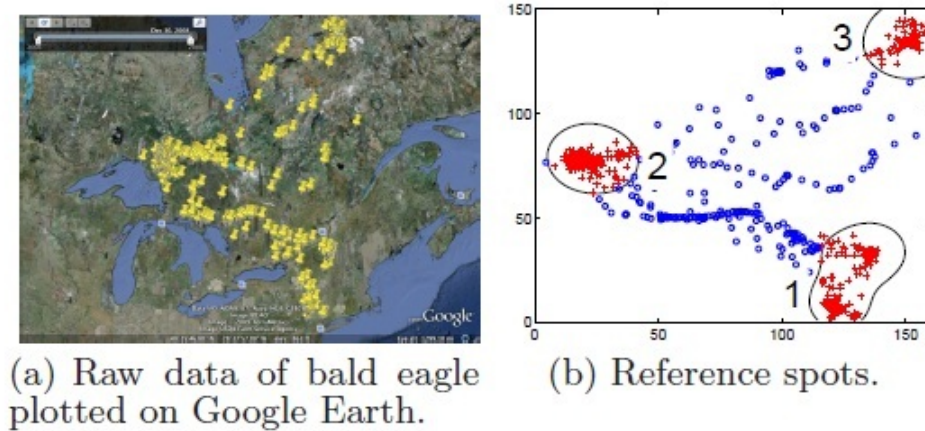


**Figure 11.** Example depicting the issues related to finding periods from movement data [11].

and hence save space. They also are useful for future movement prediction and anomaly detection i.e if an object fails to follow a regular periodic behavior it could be an anomaly caused by an accident.

Li et al. [11] have addressed the problem of mining periodic behaviors for moving objects. They have considered various issues while mining periodicity in movement. As an example consider the Fig.11 [11] which translates the raw movement of a person called David into periodic behavior. There are many complications that arise from the periodic behavior. There are multiple periods like “day” and “week” for David’s movement as shown in Fig.11 [11]. Also, periodic behaviors may interleave with each other like the time span from Sept to May which appears in both the periodic behaviors 1 and 2 in Fig.11 [11]. Prior to their work, there has been no previous work to detect multiple periods from noisy moving object data, except for Bar-David et al. [45], who directly applied the fourier transform on moving object data by transforming a location onto a complex space, to detect periods.

In [11], Li et al., have used the movement data of the form  $(x_t, y_t)$  where  $t = 1, 2, 3, \dots, n$ . The raw data is linearly interpolated with a constant time gap such as hour or



**Figure 12.** Real bald eagle data [11].

day. Their method for detecting periods involves first finding the *observation spots*, which are the frequently visited locations with higher density than a random location. At each observation spot the movement is transformed into a binary sequence, e.g. giving a value of 1 for the time period the object is present at that particular spot and a value of 0 when the object is not at that spot. Then a fourier transform and autocorrelation measure is used to detect the periods for each observation spot. The overall movement is then partitioned into segments by the period detected. Each segment is then clustered to form a behavior. As an example, consider the movement data of bald eagle in Fig.12 [11] where the observation spots are first extracted. By applying period detection to each observation spot, the periods for each observation spot were found to be 363, 363 and 364 days, respectively.

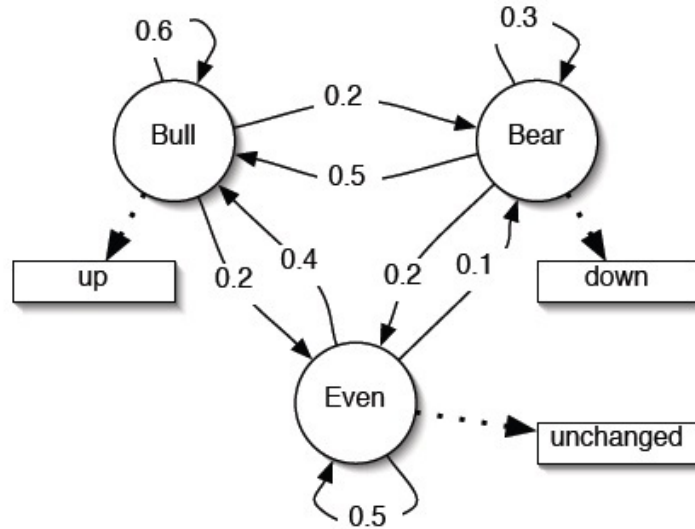
## 2.6. PROBABILISTIC MODELING OF TRAJECTORIES

Jueng et al. [46] have used a HMM based representation of trajectories for mining trajectory patterns. They have modeled the frequent regions in the trajectories where an object frequently visits, as hidden states, and the observed positions as the observed states. They

have employed grid based partitioning of the data space to partition the various positions into disjoint cells and assign each position to a distinct cell. Each frequent region is then associated with a set of one or more possible partitioned cells. A HMM based on this observed and hidden states is used to explain the relationships between the frequent regions and partitioned cells. Thus, the discovery accuracy of trajectory patterns according their approach doesnot depend on the space granularity of the partitioning method used.

Bashir et al. [47] have segmented the trajectories into subtrajectories which are represented in a principal component analysis subspace. Once the trajectories are represented using PCA-based method, the underlying class distribution is modeled as a mixture of Gaussian using the training dataset. Once the Gaussian Mixture Models(GMM) for all classes are trained, the classification of new trajectories can be performed by computing the likelihood for each GMM and the class with the highest likelihood represents the trajectory. Since all the subtrajectories of a specific class are modeled as a mixture of Gaussian, the order of occurrence of subtrajectories in a trajectory is not considered. Therefore, the GMM based modeling of trajectories only allows to model classes where contents are time-invariant.

Modeling trajectories using subtrajectory-based representation should emphasize the temporal ordering of subtrajectories in a trajectory. A first order Markov chain is used to model this temporal ordering. A HMM is used, where the number of states for each class is equal to the maximum number of subtrajectories in all the training set trajectories for that class. Once the HMM's for all classes are trained the classification of new trajectories is performed by computing the maximum likelihood that a particular HMM best describes the test trajectory. More details about HMM and modeling trajectories using HMM are provided in Chapter 3.



**Figure 13.** Example of a Markov Process [12].

### 3. HIDDEN MARKOV MODEL

A Hidden Markov Model (HMM) is a statistical tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence. Consider a HMM with a finite state space  $S$ , with  $N$  states i.e.  $S$  is the set containing the symbols for all  $N$  states. Let  $O$  be the observation space i.e.  $O$  is the set containing the symbols for all the  $M$  observations:

$$S = (s_1, s_2, \dots, s_N)$$

$$O = (o_1, o_2, \dots, o_M)$$

Let the pair  $(x_t; y_t)$  where  $t = (1, 2, \dots, n)$  be the Hidden Markov process where  $x_t$  is a homogenous Markov chain such that  $x_t \in S$  and  $y_t$  is the observation sequence such that  $y_t \in O$ .

Let the parameters of the HMM be  $\theta = (A, B, \pi)$ . The transition probability matrix,  $A$ , is the matrix consisting of transition probabilities from state  $i$  to state  $j$  in one step  $a_{ij}$ :

$$A = [a_{ij}], a_{ij} = P(x_t = s_j | x_{t-1} = s_i)$$

Each state in a HMM can generate an observation according to specified probabilities. For each state  $s_i$  and each possible output  $o_k$ ,  $b_i(k)$  gives the probability that observation  $o_k$  is emitted in state  $s_i$ .  $B$  is the observation array, storing the emission probabilities:

$$B = [b_i(k)]$$

$$b_i(k) = P(y_t = o_k | x_t = s_i)$$

$\pi$  is the initial probability array:

$$\pi = [\pi_i]$$

$$\pi_i = P(q_1 = s_i)$$

Two assumptions are made by the model. The first, called the Markov assumption, states that the current state is dependent only on the previous state, this represents the memory of the model:

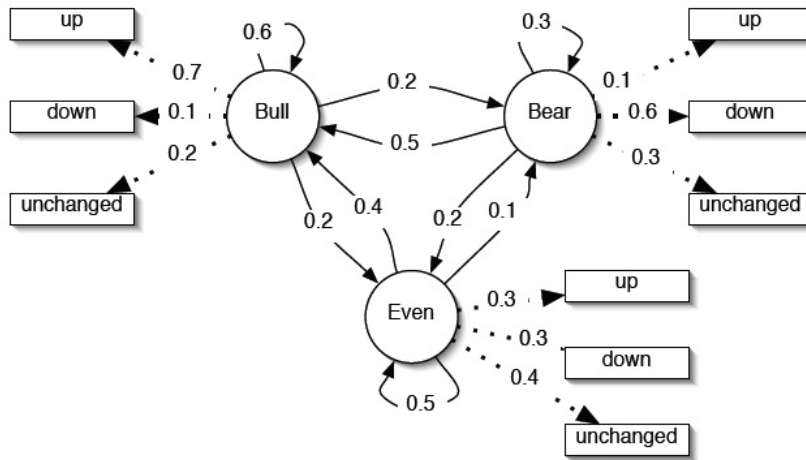
$$P(x_t | x_{t-1}, x_{t-2}, \dots, x_1) = P(x_t | x_{t-1})$$

The independence assumption states that the output observation at time  $t$  is dependent only on the current state, it is independent of previous observations and states:

$$P(y_t | y_{t-1}, y_{t-2}, \dots, y_1, x_t, x_{t-1}, \dots, x_1) = P(y_t | x_t)$$

Figure 13 [12] depicts an example of a Markov process, which presents a simple model for stock market index. There are three states in the model - *Bull*, *Bear* and *Even*, and there





**Figure 14.** Example of a Hidden Markov model [12].

are three index observations - *up*, *down*, *unchanged*. Given a sequence of observations, example: *up-down-down* we can easily verify that the state sequence that produced those observations was: *Bull-Bear-Bear*. However, a HMM is one where the states are unknown or hidden. Figure 14 shows an illustration of converting the previous model into a HMM. The new model now allows all observation symbols to be emitted from each state with a finite probability. They key difference is that now for a given observation sequence, say *up, down, unchanged*, one cannot directly tell the state sequence.

Only the observations are the visible to an external observer and the states are “hidden” to the observer, hence the name Hidden Markov Model. A HMM is therefore characterized by the number of states, the transition probabilities between the states and the emission probabilities corresponding to the observations. The likelihood of a set of parameters  $\lambda$  of a HMM given some observations  $O$ , is the probability  $p(O|\lambda)$  of observation sequence given the model.

### 3.1. BAUM-WELCH ESTIMATION

The BaumWelch (BW) algorithm is used to estimate the unknown parameters of a HMM. In general, for a given observation sequence  $Y = (y_1, y_2, \dots, y_n)$ , if the state sequence is known, we can obtain the parameters of a HMM using maximum likelihood estimation. But for a HMM, the state sequence is unknown. The BW algorithm is used to estimate the model parameters in a HMM, where the state sequence is unknown and only observation sequence is given. Given an observation sequence,  $Y$ , BW algorithm estimates the parameters  $\theta$  of the HMM.

The BM algorithm used for estimating the parameter values belongs to a family of algorithms called Expectation Maximization (EM) [27] algorithms. They all work by guessing initial parameter values, then estimating the likelihood of the data under the current parameters. These likelihoods can then be used to re-estimate the parameters, iteratively until a local maximum is reached.

The steps involved in the algorithm are as follows:

1. Choose some initial values for  $\theta$  i.e for  $(\pi_i, a_{ij}, b_i(k)) \forall i, j \in \{1, 2, \dots, N\}$  and  $k \in \{1, 2, \dots, M\}$ .
2. Choose an initial probable state path  $X = (x_1, x_2, \dots, x_n)$
3. Count the expected number of transitions,  $\bar{a}_{ij}$ , from state  $s_i$  to state  $s_j$ , given the current estimate of  $\theta$ .
4. Count,  $\bar{b}_i(k)$ , the expected number of times the observed value is  $o_k$ , given the hidden state is  $s_i$ .
5. Re-estimate  $\theta$  from  $\bar{a}_{ij}$  and  $\bar{b}_i(k)$ .

6. If not converged, go to step 2.

For a given observation sequence,  $Y = (y_1, y_2, \dots, y_n)$ , probability of transiting from state  $s_i$  to  $s_j$  at time  $t$  is

$$P(x_t = s_i, x_{t+1} = s_j | Y, \theta) = \frac{P(x_t = s_i, x_{t+1} = s_j, Y)}{P(Y)} = \frac{\alpha_t(i) a_{ij} b_j(y_{t+1}) \beta_{t+1}(j)}{P(Y)}$$

The term  $\alpha_t(i)$  is the probability that the model has emitted symbols  $y_1, y_2, \dots, y_t$  and is in state  $s_i$  at time  $t$ . This probability can be obtained using the Forward algorithm [29].

The Forward algorithm is used to calculate the probability of being in a particular state at certain time, given the history of the observations upto that time. Similarly, the Backward algorithm [29] yields  $\beta_{t+1}(j)$  the probability of emitting the rest of the sequence if we are in state  $s_j$  at time  $t + 1$ . The remaining two terms,  $a_{ij}$  and  $b_j(o_{t+1})$  give the probability of making the transition from  $s_i$  to  $s_j$  and emitting the  $t + 1^{st}$  character.

By summing up  $P(x_t = s_i, x_{t+1} = s_j | Y, \theta)$  over all values of  $t$ , we can estimate  $\bar{a}_{ij}$  i.e.

$$\bar{a}_{ij} = \frac{\sum_t \alpha_t(i) a_{ij} b_j(y_{t+1}) \beta_{t+1}(j)}{P(Y)}$$

The probability  $P(Y)$  can be estimated using current parameter values using the Forward algorithm.

Similarly,

$$\bar{b}_i(k) = \frac{\sum_{\{t|y_t=o_k\}} \alpha_t(i) \beta_t(i)}{P(Y)}$$

From  $\bar{a}_{ij}$  and  $\bar{b}_i(k)$  we can re-estimate the parameters.

### 3.2. CHANGE DETECTION IN HIDDEN MARKOV MODEL

LeGland et al. [48] have proposed statistical tests for fault detection in HMMs. They considered the case where the parameters of the HMM after the change are known. Their method specifically is used for detecting a change in the transition probability matrix of a HMM. Hypothesis testing is used to design a test to decide, on the basis of the observations  $(y_0, y_1, \dots, y_n)$ , between the following two hypothesis:

$$H_0 : \theta = \theta_0$$

$$H_1 : \theta \in \theta_1, \text{ where } \theta_0 \notin \theta_1$$

If  $l_n(\theta)$  denotes the log-likelihood function for the estimation of the parameter  $\theta$  based on the observations  $(y_0, y_1, \dots, y_n)$ , the score function,  $s_n(\theta)$ , is defined as the derivative of the log-likelihood function  $l_n(\theta)$  w.r.t the parameter  $\theta$  i.e.

$$s_n(\theta) = \frac{dl_n(\theta)}{d\theta}$$

. The central limit theorem yields that the score function evaluated at the nominal value  $\theta_0$  follows a Gaussian distribution. The original problem of designing a test for change detection in the transition probability matrix of a HMM was replaced by the simpler problem of designing a test to detect the change in mean of the score function which is a Gaussian random variable.

Gerencser et al. [49] consider the problem of change point detection for HMMs when the parameter after change is unknown by using fixed gain or forgetting rate. The transition probability matrix and emission probabilities of finite state HMM with a state space  $S$  and observation space  $O$ , where  $|S| = N$  and  $|O| = M$ , was parameterized by  $\theta$ . They consider a system in which the dynamics change slowly in time. Hence the estimation

procedure was modified as, instead of cumulating past data, it is gradually forgotten by using an exponential forgetting in the off-line case. Let the initial parameter of the HMM be denoted as  $\theta_0$ . In order to detect change in the parameter  $\theta_0$ , an MLE estimate of  $\theta_0$  is used, which gives more weight to the recent observations. Let  $\hat{\theta}_\lambda$  denote the MLE of  $\theta_0$  with fixed gain or forgetting rate  $\lambda$ . Consider a specific time  $t$  at which the parameter of the HMM changes. Before the time  $t$ , the difference between the estimated value of  $\theta_0$  i.e.  $\hat{\theta}_\lambda$  and the actual value  $\theta_0$  would be small. But after time  $t$ , this difference would start increases. Gerencser et al. [49], have analyzed the variation of the error term  $\theta_0 - \hat{\theta}_\lambda$  with different forgetting rates and established an explicit formula for the same.

In [50], Gerencser et al. considered the problem of change detection in the statistical pattern of a hidden markov process when the parameters of the HMM, before and after the change are known, unlike in [49] where the dynamics of the HMM were unknown. The specific change detection problem they address is as follows:

$$\begin{aligned}\theta^* &= \theta_0 \text{ for } n \leq \tau^* - 1 \\ &= \theta_1 \text{ for } n \geq \tau^*\end{aligned}$$

for an unknown  $\tau^*$ , but for given  $\theta_0$  and  $\theta_1$ . Their goal was to estimate  $\tau^*$ . They proposed an upper bound for the false alarm frequency assuming that there is no change in the parameters of the HMM. A basic method for detecting temporal changes in an independent sequence of observations called the Cumulative Sum algorithm or Hinkley-detector [51] was adapted by them for the change detection process.

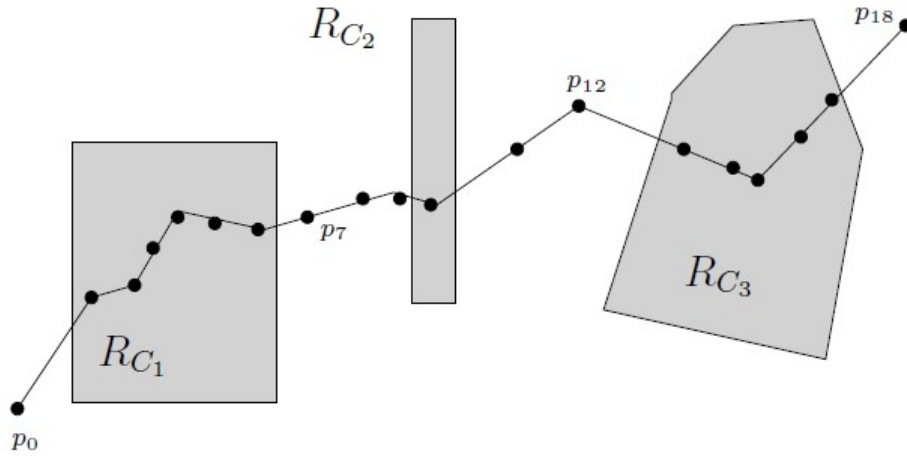
Fuh et al. [52] considered the problem of change detection in HMMs when the parameter after change is given. Let  $y_1, y_2, \dots, y_{w-1}$  be the observations from a HMM with probability distribution  $P^{\theta_0}$ , and let  $y_w, y_{w+1}, \dots$  be observations from a HMM with probability

distribution  $P^{\theta_1}$ . They consider that parameters  $\theta_0$  and  $\theta_1$  are given, while the change point  $w$  is unknown. They addressed the problem of raising an alarm as soon as possible after the distribution changes from  $P^{\theta_0}$  to  $P^{\theta_1}$ , but by avoiding the false alarms. They investigated the performance of the Shirayayev-Roberts-Pollak (SRP) [52] rule for change point detection in the dynamic system of HMMs.

We propose a novel method for change detection in the parameters of HMM, which is different from the previous methods proposed in the literature. We address the problem of change detection of specific parameters in a HMM. Our method allows specific elements of the transition probability matrix and the emission probabilities to remain unchanged, while others can change. In order to detect such changes, we have proposed a new modified Baum-Welch (m-BaumWelch) algorithm. The m-BaumWelch algorithm that we have proposed enforces constraints for parameter estimation in a HMM. Unlike the original algorithm which estimates all the parameters of the HMM, the m-BaumWelch algorithm fixes the specific parameters which have to be estimated and leaves the remaining parameters unaltered.

#### **4. SEMANTIC TRAJECTORIES**

A trajectory is typically represented as a discrete sequence of points. Recently a new trajectory concept, called semantic trajectory, has been introduced which defines trajectories from a semantic point of view [20]. They have incorporated the semantic data by decomposing trajectories into a sequence of *stops* and *moves*. Their conceptual model allows one to associate any kind of semantic annotations to trajectories like attributes of the trajectory and links between trajectories and any other object in the database. Many applications such as daily trips of employees going from home to work and back, weekly journeys of trucks



**Figure 15.** An example explaining the SMoT algorithm [13].

delivering goods to customers distributed within a given region, annual migrations of birds in search of longer daylight, need a more structured recording of movement. For daily trips of employees applications may wish to know which transportation means have been used during the trip and whether the trip used carpool facilities. For bird migrations an important information is which weather conditions the birds faced during their flight, and where, why and how long birds stopped on their way. Spaccapietra et al. [20] developed a conceptual model for trajectories that allows us to associating any kind of semantic annotations to trajectories, be it as attributes of the trajectory or via links between the trajectory and any other object in the database. They segment the trajectories to identify stops and moves within a trajectory which is described later, thereby representing a trajectory as a sequence of stops and moves.

#### **4.1. IDENTIFYING STOPS AND MOVES OF TRAJECTORIES**

A lot of research has been done to transform trajectory data into a sequence of stops and moves. Luis et al. [13] proposed a preprocessing method for trajectories to integrate

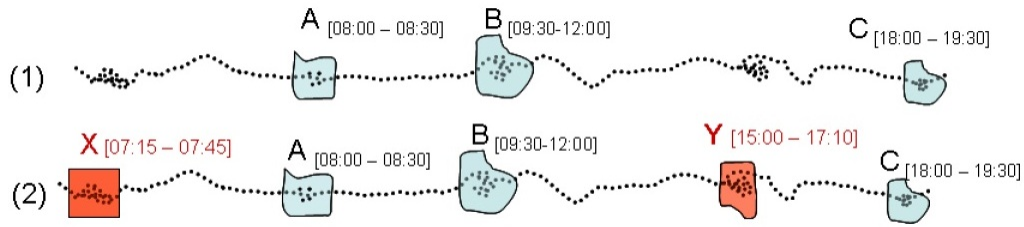
trajectories with geographic information. They have built upon the model proposed by [20] where trajectories are modeled as sequence of *stops* and *moves*. They have focussed on *stops* aspect of the trajectory by identifying points of interest. They have proposed an algorithm called *SMoT*(Stops and Moves of Trajectories) to identify stops and moves in a trajectory. The outline of the algorithm is as follows: Each point of a trajectory  $T$  is checked for if it intersects the geometry of a relevant feature type  $C$ , such as a restaurant, building etc. The geometries of the features are represented as polygons. If it intersects, the algorithm checks if the duration of intersection is at least equal to a given threshold  $\delta_C$  and if that is satisfied, the intersected candidate stop is considered as a stop, and is recorded. Moves are those parts of the trajectory that donot intersect a candidate stop for  $\delta_C$ .

Fig. 15 [13] illustrates the concept of *SMoT* algoritim. There are three candidate stops which are regions in  $x - y$  space with geometries  $R_{C_1}$ ,  $R_{C_2}$  and  $R_{C_3}$  respectively. The candidate stops are specified by the user or depend on the application. Consider the trajectory  $T$  to be represented a points in a two dimensional space ordered with respect to time as points  $p_0, p_1 \dots p_{18}$ . At first,  $T$  is outside any candidate stop, so we designate it as a move. Then  $T$  enters region  $R_{C_1}$  at point  $p_1$ , corresponding to the time  $t_1$ . The time for which the trajectory  $T$  is inside  $R_{C_1}$  is computed and since the time is long enough,  $R_{C_1}, t_1, t_6$  is the first stop and  $\langle p_0, p_1 \rangle$  is its first move. Next,  $T$  enters region  $R_{C_2}$  at point  $p_9$  but for a time interval shorter than  $\delta_{C_2}$ , hence this is not considered as a stop. The path from  $\langle p_6, p_7, \dots p_{13} \rangle$  is considered to be a move and the same condition is checked if  $R_{C_3}$  is qualified to be a stop when  $T$  enters region  $R_{C_3}$ . Since it fulfills the time constraint to be a stop,  $R_{C_3}, t_{13}, t_{17}$  is the second stop, and the trajectory ends with a move. Hence, the two stops of the trajectory are  $R_{C_1}, t_1, t_6$  and  $R_{C_3}, t_{13}, t_{17}$ .

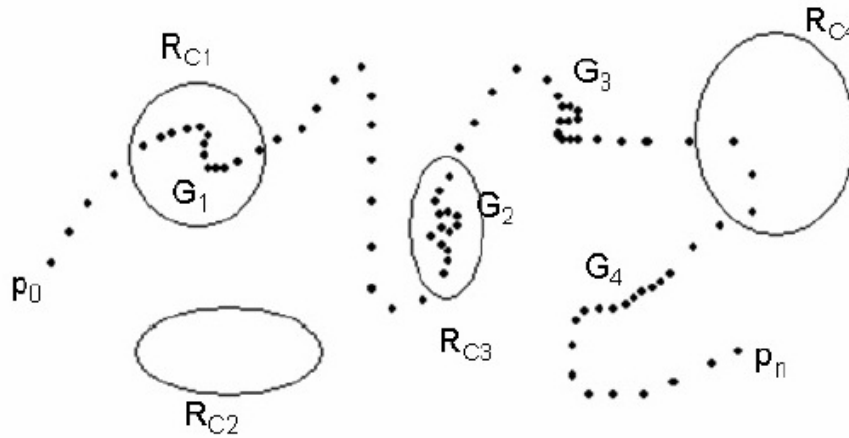


The *SMoT* algorithm proposed by Luis et al. [13], however has a drawback that it computes only the stops and moves which are expected by the user. In [14], Palma et al. have proposed a speed-based spatio-temporal clustering approach to find important places of trajectories. As an example consider Fig.16 [14] which shows a trajectory and its intersections with geographical objects in the  $x - y$  space, corresponding to A, B and C. If the time spent at each of the geographic locations A, B and C is greater than the threshold (say 30 minutes), *SMoT* algorithm identifies them as stops as shown in Fig.16(1) [14]. But the method proposed by Palma et al. [14] also identifies X and Y as stops, as shown in Fig.16(2) [14], which are not identified by *SMoT*. The algorithm proposed in [14], called *CB-SMoT*, identifies important places as those parts of the trajectory in which the speed is lower than in other parts of the same trajectory. In a first step the algorithm discovers the low speed clusters using a variation of the DBSCAN [53] algorithm. Instead of searching for a minimal amount of points inside a neighborhood to find clusters, the modified DBSCAN algorithm searched for a minimal duration, thereby taking speed into account to find slow moving points in a trajectory. In the second step, the algorithm compares the clusters identified in the previous step with the candidate stops. If a cluster doesnot intersect any of the given candidate stops, it can still be an interesting place and hence the algorithm labels such places as *unknown stops*.

Fig.17 [14] illustrates the method *CB-SMoT*. Consider a trajectory  $T$ , which is denoted by the points  $\langle p_0, p_1 \dots, p_n \rangle$  as shown in Fig.17 [14]. Firstly, the clusters of slow moving regions within a trajectory are computed using a variation of DBSCAN [53] algorithm. Let the clusters be denoted as  $G_1, G_2, G_3$  and  $G_4$ , as shown in the Fig.17. There are four candidate stops (C1,C2, C3 and C4), denoted by the ellipses  $R_{C_1}, R_{C_2}, R_{C_3}$  and  $R_{C_4}$ . The



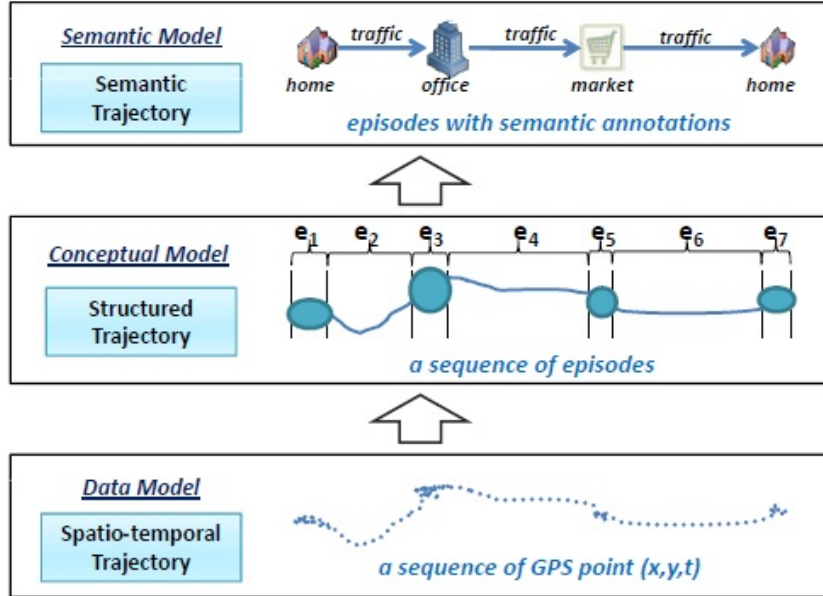
**Figure 16.** Example showing unknown stops identified by CB-SMOT [14].



**Figure 17.** An example explaining the CB-SMOT algorithm [14].

cluster  $G_1$  intersects the geometry of the candidate stop  $C_1$  for a time greater than  $\delta_{C1}$ , hence the first stop is  $R_{C1}$ . Similarly cluster  $G_2$  is labeled as  $R_{C3}$  and is the second stop of the trajectory. The clusters,  $G_3$  and  $G_4$ , do not intersect any candidate stops and are labeled as unknown stops. If SMoT algorithm was used for this trajectory,  $G_3$  and  $G_4$ , would not be identified as stops since SMoT algorithm only considers regions in trajectory intersecting a candidate stop to be a potential stop.

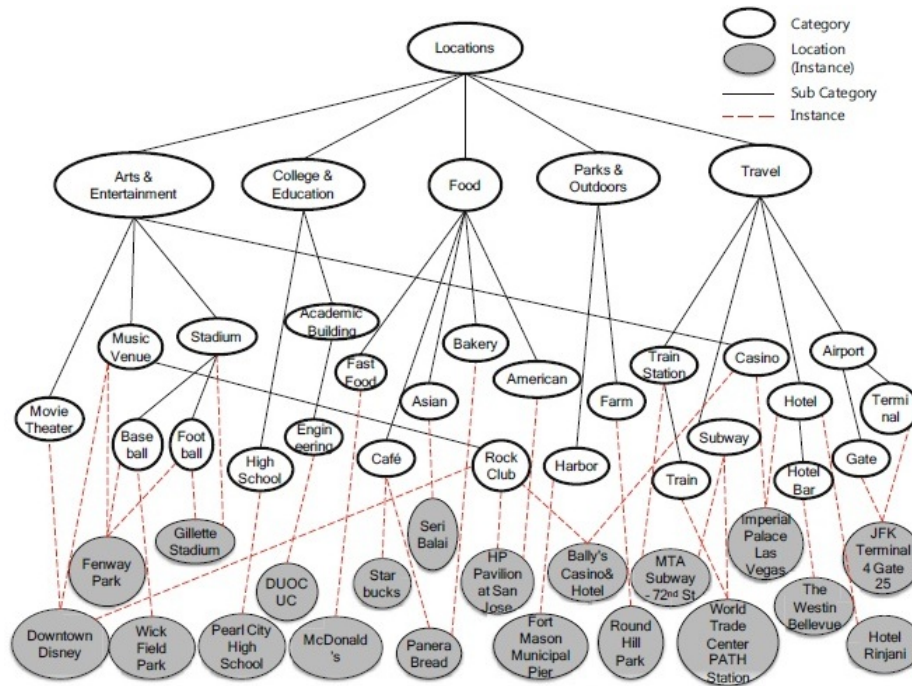
The two methods SMoT and CB-SMoT have varied areas of applications. Since speed is not a consideration in SMoT, it is important in applications like tourism and urban planning where the presence or absence of a moving object in relevant places is more important. In



**Figure 18.** Semantic annotation of raw trajectories [15].

other applications like traffic management etc. where the speed is an important factor, CB-SMoT, would be more appropriate, where CB-SMoT would find roundabouts, traffic lights and velocity controllers even if they are not given as candidate stops by the user. CB-SMoT would also be able to generate clusters in some parts of the trajectory where some points are missing, as the average speed between two points of the trajectory corresponding to such points would be minimal. This is very common in areas like hotels, buildings etc. where the GPS signal would be lost.

A body of work exists in the area of semantic annotation of trajectories [54] [15]. In [55], Yan et al. presented a framework called SeMiTri which is a multi-tiered approach towards semantic enrichment of raw trajectories. SeMiTri framework enables annotating trajectories for any kind of moving objects. The framework and its algorithms have been



**Figure 19.** The location category hierarchy graph [16].

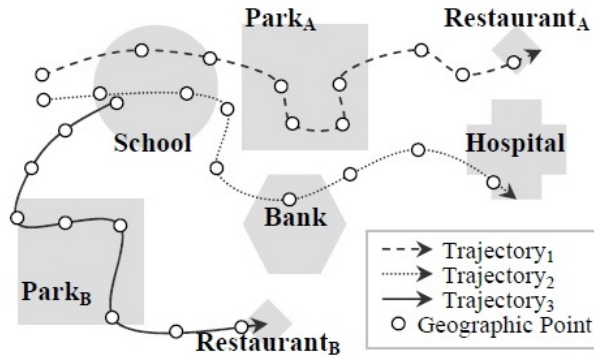
designed to work on trajectories with varying data quality and different structures, with the objective of covering abstraction requirements of a wide range of applications. Algorithms for integrating information from geographic objects (with the spatial extent of point, line or regions) were designed to be generic and accommodate most existing geographic information sources. Fig.18 [15] depicts the semantic trajectory computation methodology. The stop and move computation in SeMiTri framework is explained in [15], where velocity-based approach is used to compute the episodes(stops and moves). They have computed the instant speed for each GPS point  $p$ ,  $(x,y)$ , and if the instant speed of  $p$  is lower than  $\delta_{speed}$ , it is a part of a stop, otherwise it belongs to a move.

After the trajectory is structured into a sequence of stops and moves, SeMiTri [55] exploits the geographical context to annotate stops and moves with geographic objects(such

as regions, lines and points) as shown in Fig.18 [55]. The semantic annotation framework consists of three layers, *semantic region annotation layer*, *semantic line annotation layer* and *semantic point annotation layer*. The semantic region annotation layer forms a coarse grained view of the trajectory by using a spatial join algorithm to pick up *regions* that the trajectory has passed through. As shown in Fig.18 [55], the stop and move episodes of the trajectory are transformed into regions like *residential area*, *business area*, *market area* and *residential area* after being annotated with semantic region layer. The move episodes are processed in the semantic line annotation layer where apart from mapping the move segments to road networks, the transportation modes (such as *on foot*, *by bus*, *by metro*) are also inferred by exploiting the geometric properties (e.g velocity, acceleration) of the segment. The stops are then funneled to the semantic point annotation layer which annotates them with information about suitable points of interest(POIs). Examples of POIs are restaurant, bars, shops, movie theaters etc. A HMM is used for semantic annotation of stops. The novelty of this approach is that it works for densely populated area with many possible POI candidates. It also enables identifying the activity(behavior) behind the stop thus providing information related to the purpose of the stop.

#### **4.2. ANALYTICS FOR SEMANTIC TRAJECTORIES**

The semantic information attached to the spatio-temporal path of a moving object reveals useful application knowledge. Consider the trajectory data of tourists visiting various locations during a trip. The kind of semantic knowledge from these trajectories would allow us to answer questions like *which are the places most frequently visited by tourists in the morning?*. A pattern of the form [*Hotel Cascade*] (s= 90%) would imply that 90% of the tra-



**Figure 20.** Examples of three semantic trajectories [17].

jectories have hotel cascade in them. Similarly, a pattern  $[TouristPlace, Hotel](s = 80\%)$  would mean that 80% of the trajectories stop at a tourist place as well as a hotel.

Association rule mining [56] can be applied over such *stops* data to find a relation between the various *stops* in the semantically enriched data to answer questions like *is there any relation between visited touristic places and hotels?* A rule of the form  $\{[Hotel_1 \Rightarrow [TouristPlace_1]](s = 20\%)(c = 70\%)$  expresses that tourists that stop at  $Hotel_1$  also stop at  $TouristPlace_1$ . This happens in 20% of the trajectories, and with a confidence of 70% [57]. The confidence measure implies that for 70% of the times a person visits  $Hotel_1$ , he visits  $TouristPlace_1$  as well.

Sequential pattern mining [58], which is discussed later in Section 2.5, can be used to extract patterns from the *stops* dataset. *Which is the sequence of tourist places most frequently visited and when these visits occur.* A pattern of the form  $\{[Louvre]_{morning}, [NotreDame]_{afternoon}\} (s=8\%)$  would mean that visitors go to the Louvre in the morning and goto the Notre Dame church in the afternoon. Sequential pattern mining can also be applied over the *move* data set. A pattern of the form  $\{[Orsay-EiffelTower], [Invalides-$

*NotreDame}}* (s= 5%) would imply that trajectories that have a move from Orsay museum to Eiffel tower also have a move from Invalides to Notre Dame church in this order [57].

Clustering techniques can be applied to semantically enriched trajectories to find user similarity. Consider Fig. 20 [17] where trajectories are tagged with a number of semantic labels like school, park etc. We can see that both *Trajectory<sub>1</sub>* and *Trajectory<sub>3</sub>* can be represented as the sequence {School, Park, Restaurant}. The semantic behavior of *Trajectory<sub>1</sub>* and *Trajectory<sub>3</sub>* are quite the same and they are more similar to each other than that of *Trajectory<sub>2</sub>*. But if we only considered the geographic distance, *Trajectory<sub>1</sub>* is more geographically close to *Trajectory<sub>2</sub>*.

In [16], Lee et al. proposed a method to calculate the user similarity based on the semantics of the location. In their method, locations and their categories are used to form a hierarchical graph structure as shown in Fig. 19. By considering only the relevant nodes and computing similarity at necessary nodes, the proposed method generates the similarity results. Only the top-*k* visited locations of each user are considered. An example of similar users by their method is as follows: user A and user B live in very different locations, but they are similar because they are both students (since they visit the university frequently) and they like to go to shopping (as they often visit the mall). However, their method did not consider the sequential knowledge of the trajectories to cluster the users.

In [17], Josh et al. have defined a novel similarity measure, *Maximal Semantic Trajectory Pattern Similarity (MSTP-Similarity)*, which measures the semantic similarity between trajectories. This similarity measure is used as a basis for recommending potential friends to a user. To transform GPS trajectories into semantic trajectories, they use a cell based approach. To deal with cell trajectories, they treat a cell station as a geographic region. Then,

the stay time can be derived by calculating the difference between the time a user arrives and leaves the cell. A user-specified time threshold is used to filter the cells with stay time shorter than the threshold. The remaining cells (i.e., their stay time is equal or greater than the threshold) are called *stay cell*. Therefore, we can transform each cell trajectory as a stay cell sequence. Then, a geographic information database was used to assign semantic terms to the discovered stay cells. After transforming each geographic trajectory to a semantic trajectory, each users geographic trajectory set was transformed as a semantic trajectory dataset. The semantic trajectories of a user may be quite diverse since the user movements may change time to time. Sequential pattern mining algorithm *Prefix-Span* [59] was used on each user's semantic trajectory dataset to mine maximal frequent semantic trajectory patterns. The Longest Common Sequence (LCS) of these two patterns to represent their longest common part. Techniques related to information theory like TF-IDF [60] are used to find the similarity between users by treating a pattern set of a user as a document and each pattern in a pattern as a word.

Semantic trajectory similarity techniques have been used in many recommendation systems [17] [16] [61] for recommending friends based on users semantic trajectories for location-based social networks. In [61], Zheng et al. propose a personalized friend and location recommendation system. To explore users similarity, the system considers users movement behaviors in various location granularities. Based on the notion of stay points which are the geographic regions mobile users stay for over a time threshold, the system discovers all of the stay points in trajectories and then employ a density-based clustering algorithm to organize these stay points as a hierarchical framework. To measure two users similarity, some common sequences, named similar sequence, are discovered by matching



their stay region sequences in each level of the hierarchical graph. However, this approach treats every stay region in the similar sequence independently, i.e., without considering the sequential property of stay regions in the similar sequence.

## 5. SEQUENTIAL PATTERN MINING

Several classes of data change continuously with time. Hence a time stamp is becoming essential while collecting data. As an example, consider a transactional database which consists of transactions made by customers visiting a store. Each transaction consists of a list of items purchased by the customer during a visit to the shop. A transaction of the form  $(A, B, C)$  implies that the customer has bought items  $A, B$  and  $C$  in that particular transaction. Consider a rule of the form  $A \rightarrow B$ , which implies that a person who buys the product  $A$  also buys product  $B$ . However it would have been more useful for organizations to make sound decisions if it was known that a person who buys product  $A$  buys product  $B$  within a week.

Different mining techniques have been designed for mining time series data, and have identified four kinds of patterns we can get from timeseries data:

- Trend Analysis: Trend analysis is a technical analysis approach that tries to predict the future trend or pattern based on the past data.
- Similarity search: Unlike normal queries, which find data that match the normal query exactly, a similarity search find data sequences that differ only slightly (within a user-defined parameter) from the given query sequence.
- Sequential patterns.

- Periodic patterns: Periodical patterns are those recurring patterns in the time series database. Periodicity can be daily, weekly, monthly, seasonal or yearly. Obviously periodic pattern mining can be viewed as sequential pattern mining.

Sequential Pattern Mining (SPM) [58] is a technique used to find the relationships between occurrences of sequential events, to find if there exist any specific order of the occurrences. It can be thought of as an extension of association rule mining [56], with a time stamp associated with each transaction, thereby emphasizing on the order of the occurrence of the events. An example of sequential patterns is that every time Microsoft stock drops atleast 5%, IBM stock will also drop at least 4% within three days. Business organizations use sequential pattern mining to study customer behaviors. Sequential patterns can be extracted from web log analysis, which are very useful to better structure a companys website for providing easier access to the most popular links [62]. It is also used in intrusion detection [63] and DNA sequence analysis [64].

SPM was first introduced by Agrawal and Srikant [58] based on their study of customer purchase sequence as follows: “Given a set of sequences, where each sequence consists of a list of events (or elements) and each event consists of a set of items, and given a user-specified minimum support threshold of  $\text{min\_sup}$ , sequential pattern mining finds all frequent subsequences, that is, the subsequences whose occurrence frequency in the set of sequences is no less than  $\text{min\_sup}$ .”

A subject or customer does various transactions, with each transaction comprising single or many items. A collection of item defines an itemset. The collection of transactions related to a subject can be thought of as a sequence. If we have a set of items denoted by

Patient ID	Date	Treatment Codes	
Joe	1/3/2005	J0102	
Joe	1/18/2005	90012, <b>G1234</b>	}
Joe	6/5/2005	<b>J0502</b> , 52113	
Joe	7/1/2005	49991, V2025, <b>90012</b>	
Jane	2/28/2005	21589, <b>G1234</b>	
Jane	12/24/2005	B5859, 22123, 20.50	}
Jane	4/14/2006	J0125, <b>J0502</b>	
Jane	4/15/2006	<b>90012</b> , X8502, DX234, V2025	
Bob	7/8/2005	90011, <b>G1234</b> , C0155	
Bob	8/15/2005	<b>J0502</b> , 88226	}
Bob	8/18/2005	23889, <b>90012</b> , Q1258	
Sue	9/18/2005	90011, C6025	
Sue	10/26/2005	<b>G1234</b>	}
Sue	11/18/2005	D1118, Z5589	
Sue	11/19/2005	<b>J0502</b>	
Sue	11/28/2005	<b>90012</b>	

**Figure 21.** Sequence database of patients visiting a hospital [18].

$(i_1 i_2 \dots i_m)$ , then an itemset  $s_j$  is a non-empty set of items. A sequence is thus an ordered list of such itemsets denoted by  $\langle s_1, s_2 \dots s_n \rangle$ .

The sequence  $\langle a_1 a_2 \dots a_n \rangle$  is said to be contained in the sequence  $\langle b_1 b_2 \dots b_m \rangle$  if there exists integers  $i_1 < i_2 < \dots < i_n$  such that  $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_n \subseteq b_{i_n}$ . The transactions are generally ordered by increasing transaction time  $T_1, T_2, \dots, T_n$ . To represent a sequence in terms of the transactions, consider the set of items in transaction  $T_i$  to be denoted as  $\text{itemset}(T_i)$ . Let the set of transactions of a particular subject be  $T_1, T_2, \dots, T_n$  the sequence corresponding to the particular subject is therefore  $\langle \text{Itemset}(T_1) \text{Itemset}(T_2) \dots \text{Itemset}(T_n) \rangle$ .

To illustrate the above concepts, consider the example in Fig. 21 [18], which is a sequence database storing the information of the treatments undergone by different patients during their visits to a hospital. The sequence corresponding to the patient “*Joe*” in Fig.

21 [18] is  $\langle J0102, (90012, G1234), (J0502, 52113), (49991, V2025, 90012) \rangle$ . It can be seen that the pattern  $\{G1234, J0502, 90012\}$  exists in the sequences of all the patients.

Support is a measure of importance of a sequence. The support of the sequence  $s$  in a sequence database  $D$ , is the number of sequences of  $D$  which contain  $s$  as a subsequence. Referring to the previous example, sequence  $s$  corresponds to sequence of treatments undergone by the patients, and the database  $D$  consists information of treatments undergone by different patients during their visits to a hospital. If the support of  $s$  is greater than a threshold, then  $s$  is a frequent sequence. For the example in Fig. 21 [18] the pattern  $\langle G1234, J0502, 90012 \rangle$  occurs as a subsequence in all the sequences. The support of the pattern  $\langle G1234, J0502, 90012 \rangle$  is computed as the proportion of the patients containing that pattern i.e support =  $4/4$ .

Numerous algorithms have been proposed for SPM, some of which include GSP(Generalized Sequential Patterns) [65], PrefixSpan [59], FreeSpan [66], SPADE [67] etc. In our work, we have used PrefixSpan as it can be used with larger data sets. PrefixSpan filters the database making it much smaller by employing database projection, to make the algorithm faster. It also can handle long sequential patterns unlike GSP.

Besides mining sequential patterns in a single dimension, mining multiple dimensional sequential patterns can further reveal important patterns. Multi-dimensional sequential pattern mining was first introduced by Pinto et al. [68]. For example, consider a market database which identifies the following pattern: most people who buy product A also purchase product B within a specific time, by employing general sequential pattern mining. However, multiple dimensional sequential pattern mining can also reveal important purchasing patterns among various groups of individuals. For example, while students often

buy A within a week after B, this pattern does not hold true for individuals of other age groups.

### **5.1. ANALYTICS FOR SEQUENTIAL PATTERNS**

Most of the work related to SPM dealt with finding sequential patterns in static databases. Some of the works dealt with filtering the large amount of frequent patterns extracted by imposing constraints like time between the consecutive items in a frequent sequential pattern, length of the sequential patterns etc [69]. Recently research focused on extracting sequential pattern in dynamic streaming data where storing the large stream of incoming data was a concern [70]. Han et al. [71] have used a lexicographic tree to store the sequences seen in the data stream. They consider the incoming stream of data as arriving one batch at a time. The data stream is broken into fixed-size batches where each batch contains  $L$  sequences. For each arriving batch, PrefixSpan algorithm was used to extract frequent sequential patterns corresponding to a specific support. Each frequent sequence is then inserted into the lexicographic tree and the properties like batchCount and count values of the corresponding node are updated.

Tsai et al. [72] have addressed the issue of change detection of sequential patterns between two different datasets. The datasets are obtained by collecting data like customer purchase data etc. over different time periods. They have compared the sequential patterns extracted from two datasets based on the similarity of the sequential patterns extracted from the two datasets. They considered sequential patterns as strings of data and used techniques from information theory like Levenshtein edit distance [73] to find the difference between the two strings. Based on a specific threshold they have categorized the patterns as emerging sequential patterns, unexpected sequence changes and added or perished patterns. Their

work however dealt with static databases. Also, they compared datasets of approximately similar size. In our work, we compare datasets of considerably different sizes and the data is also dynamic.

Laur et al. [74] have addressed the problem of comparing sequential patterns based on statistical measures like support when the original data has been incrementally updated by incoming data. The knowledge of the stream is only partial at any time as the whole data stream cannot be analyzed at the same time. The data stored from the stream is only a representation at that instant of the data stream and thus the information mined from this stored data should take into account the uncertainty generated by partial observation of the whole stream. They address issues related to longer observation of the data stream like some patterns observed as frequent might become infrequent while some patterns observed as not frequent might become frequent from a longer history of the data stream. They have introduced two statistical borders (upper and lower) which would be useful in choosing sequential patterns in an incremental mining process.

Jacquemont et al. [75] have analyzed SPM from a statistical point of view. They consider the sequential pattern mining process as extracting frequent sequences from a finite sample set of sequences that have been drawn from an unknown target distribution. They analyze the statistical bias associated with the sequential patterns extracted from the sample set by considering the underlying statistical distribution from which the sample set has been drawn. They have used statistical procedures like hypothesis testing to verify the constraints under which a sequential pattern mining process is statistically relevant. They have also provided a lower bound on the number of sequences needed to guarantee the discovery of significant knowledge.

Dong et al. [76] introduced a new class of patterns called Emerging Patterns(EP) for knowledge discovery from different databases. Emerging patterns are patterns whose support changes significantly from one dataset to another. Such patterns can be used to capture emerging trends in timestamped databases and also significant changes and differences between datasets. When applied to datasets with classes like male vs female etc. such patterns can be used to capture useful contrasts between classes. Boulesteix et al. [77] have developed an approach using decision trees for inferring emerging patterns and used them for classification of micro-array data. They have suggested an alternative definition of a statistical EP. In the method proposed by Boulesteix et al. [77], EPs are extracted by growing decision trees. A decision tree [78] is a statistical model, used for classification, that recursively partitions the measurements space into subsets by splitting on a variable. The agglomerated splitting on several variables is used to generate rules for a leaf in a tree which distinguishes one class from another. This rule has a high support in one class (which it classifies) while a lower support in all other classes. These decision rules are thus equivalent to statistical EPs whose supports vary with different classes or datasets. The EPs inferred by the tree based approach were subsequently used for classification using linear discriminant analysis (LDA) [79].

## 6. RANDOM FORESTS

A decision tree [78] is a model used for classification, which predicts the value of a target variable based on several input variables. Consider a training set  $S = (X, Y)$ , consisting of instances  $(x_i, y_i), i = 1 \cdots n$ , where  $x_i \in X$  is an input vector and  $y_i \in Y$  is its corresponding class label.  $x_i$  is made up of a number of features  $f_1, f_2 \cdots f_m \in M$ . A decision tree splits each of the input vector  $x_i$  based on the values of the features

$f_1, f_2 \dots f_m$ . At each node of the decision tree, the input data is partitioned into subsets, such that the impurity of the resulting child node is decreased. This process is repeated to form a tree.

Let  $p(c|t)$  denote the fraction of records belonging to a class  $c$  at a given node  $t$ . Some of the impurity measures include

$$Entropy(t) = - \sum_{c=0}^{C-1} p(c|t) \log_2 p(c|t)$$

$$Gini(t) = 1 - \sum_{c=0}^{C-1} [p(c|t)]^2$$

$$Classificationerror(t) = 1 - \max_c [p(c|t)]$$

where  $C$  is the number of classes and  $0 \log_2 0 = 0$  in entropy calculations.

To determine how well a test condition performs, we need to compare the degree of impurity of the parent node (before splitting) with the degree of impurity of the child nodes (after splitting). The larger their difference, the better the test condition. The gain,  $\delta$ , is a criteria that can be used to determine the goodness of a split:

$$\delta = I(parent) - \sum_{j=1}^k \frac{N(v_j)}{N} I(v_j)$$

where  $I(.)$  is the impurity measure of a given node,  $N$  is the total number of records at the parent node,  $k$  is the number of attribute values, and  $N(v_j)$  is the number of records associated with the child node,  $v_j$ . Decision tree induction algorithms often choose a test condition that maximizes the gain  $\delta$ .

The top most node of the tree is called the root node and the bottom nodes are called leaf nodes. In a decision tree each leaf node is assigned a class label. The non-terminal nodes, which include the root and other internal nodes, contain attribute test conditions to



separate records that have different characteristics. Tree nodes are used to determine how to propagate a given attribute set down the tree. In order to classify a test record, it is propagated down the tree and decision is made based on the terminal node that is reached.

Random forest [80] uses a collection of decision trees instead of one tree. A random forest is a collection of trees, where each node in a tree is split based on the greatest information gain obtained from only a random sample of attributes. The outline of the algorithm used to construct a random forest is as follows: Assume the full data set consists of  $N$  observations

1. Take a random sample of  $N$  observations from the data set with replacement (this is called bagging). Some observations will be selected more than once, and others will not be selected. On average, about  $2/3$  of the rows will be selected by the sampling. The remaining  $1/3$  of the rows are called the “out of bag (OOB)” rows. A new random selection of rows is performed for each tree constructed.
2. Using the rows selected in step 1, construct a decision tree. Build the tree to the maximum size, and do not prune it. As the tree is built, allow only a subset of the total set of predictor variables to be considered as possible splitters for each node. Select the set of predictors to be considered as a random subset of the total set of available predictors. Perform a new random selection for each split. Some predictors (possibly the best one) will not be considered for each split, but a predictor excluded from one split may be used for another split in the same tree.
3. Repeat steps 1 and 2 a large number of times constructing a forest of trees.

4. To score a row, run the row through each tree in the forest and record the predicted value (i.e., terminal node) that the row ends up in. For a classification analysis, use the predicted categories for each tree as “vote” for the best category, and use the category with the most votes as the predicted category for the row.

Decision tree forests have two stochastic (randomizing) elements: (1) the selection of data rows used as input for each tree, and (2) the set of predictor variables considered as candidates for each node split. One of the main reasons random forests perform better than a single decision tree is their ability to utilize redundant features. The generalization error for forests converges to a limit as the number of trees in the forest becomes large [80]. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them [80].

### 6.1. RANDOM FOREST PROXIMITY MEASURE

Random forests can also be used to obtain a proximity or similarity measure [81] between any two cases in the data. For a given forest  $F$ , we compute the similarity between two instances  $x_1$  and  $x_2$  in the following way. For each of the two instances, we first propagate their values down all trees within  $F$ . Next, the terminal node position for each instance in each of the trees is recorded. Let  $z_1 = (z_{11}, z_{12} \cdots z_{1T})$  be the tree node positions in the  $T$  trees for  $x_1$  and similarly define  $z_2$ . Then the similarity between pair  $x_1$  and  $x_2$  is set to: ( $I$  is the indicator function)

$$S(x_1, x_2) = \frac{1}{T} \sum_{i=1}^T I(z_{1i} == z_{2i})$$

The  $(i, j)$  element of the proximity matrix produced by random forest is the fraction of trees in which instances  $i$  and  $j$  fall in the same terminal node. The intuition is that similar

instances should be in the same terminal nodes more often than dissimilar ones. The proximity matrix can be used to identify structure in the data or for unsupervised learning with random forests. This proximity matrix obtained was used for clustering the trajectories.

## **6.2. UNSUPERVISED LEARNING WITH RANDOM FORESTS**

Random forests are ensembles of decision trees that output the class that is the mode of the classes output by individual trees. They are mostly used for classification. However, random forests can also be used for unsupervised learning such as clustering [81]. The first step is to call the original data “class 0” and construct a “class 1” synthetic data. A label i.e. 0 or 1, is assigned for each instance based on the type of the data. The combined data is then classified using the random forest.

There are two ways to simulate the “class 1” data:

1. The “class 1” data are sampled from the product of the marginal distributions of the variables (by independent bootstrap of each variable separately).
2. The “class 1” data are sampled uniformly from the hypercube containing the data (by sampling uniformly within the range of each variables).

The idea is that real data points that are similar to one another frequently end up in the same terminal node of a tree. This similarity is measured by the proximity matrix. If two instances fall in the same leaf node of a tree, their proximity is increased by one. Thus the proximity matrix can be taken as a similarity measure, and clustering or multi-dimensional scaling using this similarity can be used to divide the original data points into groups for visual exploration.

### **6.3. *CONDITIONAL INFERENCE TREE FORESTS***

Random forests select variables for splitting so as to decrease the entropy in the resulting child nodes. If the child node is pure and consists of a single class, the entropy would be zero. This method of growing trees induces bias in the variable selection [82]. Random forests tend to select variables with many splitting values e.g. numeric variables are selected compared to categorical variables. For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. Alternate tree growing strategies can be used to overcome the bias problem.

Conditional inference trees estimate a regression relationship by binary recursive partitioning in a conditional inference framework [83]. The algorithm works as follows:

1. Test the global null hypothesis of independence between any of the input variables and the response. Stop if this hypothesis cannot be rejected. Otherwise select the input variable with strongest association to the response. This association is measured by a p-value corresponding to a test for the partial null hypothesis of a single input variable and the response.
2. Implement a binary split in the selected input variable.
3. Recursively repeat steps 1 and 2.

The conditional inference trees thus reduces the bias in variable selection by separating the variable selection and splitting step unlike decision trees where variables are selected based on the best split obtained which reduces the entropy in the child nodes.

#### **6.4. OUTLIER DETECTION USING RANDOM FORESTS**

The proximity measure obtained from the random forests can be used to detect outliers in the data. Outliers are those instances whose proximity to rest of the data is small (or distance is large). Let the average proximity from instance  $n$  to the rest of the data be

$$\bar{P}(n) = \sum_k prox^2(n, k)$$

The raw outlier measure or Breiman outlier score [80] for instance  $n$  is defined as

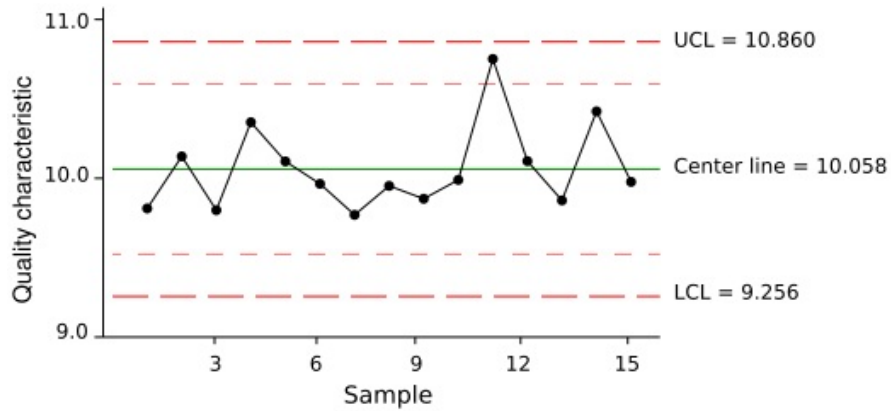
$$N/\bar{P}(n)$$

where  $N$  is the total number of instances in the sample. This value is large if the average proximity is small.

#### **7. ISOLATION FOREST**

The anomaly detection method called isolation forest proposed by Liu et al [84], uses an ensemble of binary trees to detect anomalies. Unlike the other anomaly detection techniques, which construct a profile of normal instances and identify instances that donot confirm to the normal profile as normal, their method isolates anomalies rather than profiling normal instances.

Given a data set, a binary tree which randomly partitions by splitting on the variables in the data, is used to isolate all the instances in the data. The tree is grown fully so that each leaf node of the tree contains only one instance. The path length of any instance is defined as the number of nodes it takes to traverse from the root node to the leaf node to which the instance belongs to. Since, anomalies are sparse in space, they would need lesser number of partitions to be isolated from the rest of the data. Hence, the path lengths of anomalies would be shorter compared to normal instances.



**Figure 22.** A typical control chart [19].

A forest of trees are grown and the average path length of each instance is calculated using these trees. Anomalies are those instances which have shorter path lengths for most of the trees in the forest. An anomaly score  $s$  which lies between 0 and 1 is proposed, which aggregates the path length across all the trees in the isolation forest. The instances which have  $s$  very close to 1 are definite anomalies. If instances have  $s$  much smaller than 0.5, than they are quite safe to be regarded as normal instances. If all the instances have  $s \approx 0.5$ , than the entire sample doesnot have any distinct anomaly.

## 8. CONTROL CHARTS

A control chart is a graphical display of a quality characteristic that has been measured or computed from a sample versus time or sample number. A typical control chart is shown in Fig. 22 [19]. The chart contains a center line which represents the average value of a quality characteristic when the process is in-control. Two other horizontal lines upper control limit (UCL) and lower control limit (LCL), called the control limits, are used to detect an out-of-control signal. If the process is in-control, nearly all of the sample points

will fall between the UCL and LCL. However, if a point is outside the control limits, there is considerable evidence that the process is out-of-control.

The control limits are calculated as follows:

$$UCL : \mu + k\sigma$$

$$LCL : \mu - k\sigma$$

where  $\mu$  is the mean and  $\sigma$  is the standard deviation of the in-control data. The value of  $k$  is generally chosen to be 3.

Control charts can also be used to monitor more than one variable at a time. The Hotelling  $T^2$  control [85] chart is a multivariate extension of the control chart, which can be used to monitor multiple variables and also takes the correlation among the variables into account. While plotting a Hotelling  $T^2$  chart, the data can consist of subgroups or individual observations. For subgroup data, given  $p$  response variables, the subgroup Hotelling control chart plots the following quantity

$$T^2 = n(\bar{x} - \bar{\bar{x}}_0)'S^{-1}(\bar{x} - \bar{\bar{x}}_0)$$

where  $n$  is the sample size,  $\bar{x}$  is the  $p$ -dimensional vector of subgroup means,  $\bar{\bar{x}}$  is the  $p$ -dimensional vector of means of the subgroup means,  $s^{-1}$  is the inverse of the pooled covariance matrix. The  $T^2$  statistic is plotted against the sample number. The upper control limit for the subgroup Hotelling control chart is

$$UCL = \frac{kn p - kn - np + p}{kn - k - p + 1} F_{\alpha, p, kn - k - p + 1}$$

where  $n$  is the sample size,  $k$  is the number of subgroups and  $p$  is the number of variables.

## CHAPTER 3

### **DETECTING CHANGES IN TRAJECTORIES WITH TIME**

With recent advances in GPS, sensing and communication technologies, the position data of people and vehicles is available easily and increasing rapidly. The vehicles in cities, such as taxis, are equipped with GPS devices which provide information regarding the position of the vehicles by analyzing the GPS traces left by them. Analyzing this spatio-temporal data can be used to gain knowledge about human behavior and the dynamics of the city.

Detecting changes in the trajectories of vehicles like taxis in a city, have many potential applications. In developing cities, the road networks change over time. It is important to update these changes in a digital map. If the taxis moving in a city are viewed as sensors that provide real time information of the traffic in the city, a change in these trajectories with time would reveal that the road network has changed with time i.e. either a new road has been constructed or an existing road has been blocked. Change detection in trajectories of taxi or cabs in a city can also be used to prevent taxi fraud. Most tourists are victims of taxi frauds where the taxi drivers overcharge the customers by taking unnecessary detours. If a mechanism to detect these detours in real time is available, these frauds can be avoided. Other applications of trajectory change detection include maritime surveillance of ports and waterways for the purpose of safety in navigation and collision avoidance.

The rest of the chapter is organized as follows. In Section 1, we explain how the trajectories are modeled using a Hidden Markov Model (HMM). In Section 2, experiments are conducted to illustrate a change point detection method for observations from a HMM, when the parameter after change is known. Experiments for the case when the parameter after change are unknown are shown in Section 3. A statistical process control based method using the modified Baum-Welch algorithm, to detect specific parameter changes in



TABLE 2. Notation table

Notation	Description
$\theta_0$	Null case parameters of a HMM
$\theta_1$	Shifted case parameters of a HMM
$\hat{\theta}$	Estimated parameters of a HMM
$r$	Input parameter denoting row number for change detection using m-BW algorithm
$\hat{\theta}_r$	Estimated parameters of a HMM using m-BW with input parameter $r$
$N$	No.of states of HMM
$M$	No.of observations of HMM
$A$	Transition probability matrix (TPM) of HMM with elements $a_{ij}$
$B$	Estimation probabilities of HMM
$\pi$	Initial probabilities of HMM
$S$	State space of HMM i.e $S = (s_1, s_2 \dots s_N)$
$O$	Observation space of HMM i.e $O = (o_1, o_2, \dots o_M)$
$x_t$	Variable denoting the state of HMM at time $t$
$y_t$	Variable denoting the observation of HMM at time $t$
$\theta$	Parameters of HMM initially provided as input to m-BaumWelch(m-BW) algorithm
$\bar{\theta}$	Parameters of HMM in each iteration of the m-BW algorithm
$\bar{A}$	TPM of HMM in each iteration of m-BaumWelch algorithm
$\bar{B}$	Emission probabilities of HMM in each iteration of m-BW algorithm
$\bar{\pi}$	Initial probabilities of HMM in each iteration of m-BW algorithm
$\delta$	Threshold used for convergence in m-BW algorithm
$L(t)$	Log-likelihood ratio statistic used for change detection
$t_0$	Time at which the shift occurs
$W(t)$	Sliding window of observations of size $w$
$L_0(t)$	Log-likelihood of observations in window $W(t)$ w.r.t $\theta_0$
$L_1(t)$	Log-likelihood of observations in window $W(t)$ w.r.t $\theta_1$
$T_i$	Trajectory denoted as a sequence of time-ordered observations i.e. $T_i : (x_t^i, y_t^i)$
$m_i$	Length of trajectory $T_i$

a HMM, by analyzing the observations from a HMM is presented in Section 4. In Section 5, we explain how the HMM based modeling can be extended for change detection in a group of trajectories, each of which consists of a set of time-ordered observations. In Section 5, the change detection method is evaluated by finding run lengths to obtain an out-of-control

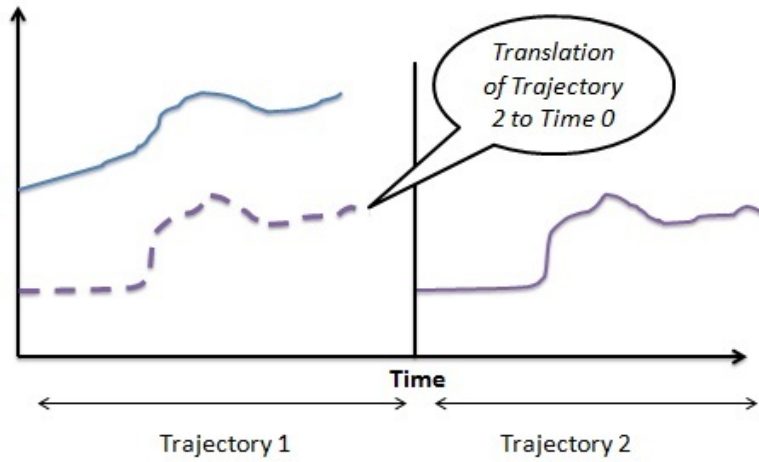
signal using control charts. The autocorrelation values for the in-control data are also calculated. These metrics are explicitly calculated in Section 5, but are also applicable to the preceding Sections 1 through 4. The various notations used in this chapter along with their description are shown in Table 2.

## **1. MODELING TRAJECTORIES USING HIDDEN MARKOV MODELS FOR CHANGE DETECTION**

We aim to detect two types of changes in trajectories. The first change is change within a trajectory and the other is change between trajectories. An example of change within trajectory is the case of taxi-fraud where taxi drivers overcharge customers by taking unnecessary detours. By monitoring the GPS points in a trajectory, which moves from a particular source to destination, we aim to detect if the observed GPS points donot follow the regular pattern of normal behavior.

An example for change between trajectories is the case of road network change. The road networks in a city change with time and if the taxis moving in a city can be considered as sensors to monitor the road network, a change in the trajectories of taxis with time indicates a new road being built or an existing road being blocked. We thus aim to detect an emerging cluster of trajectories with time.

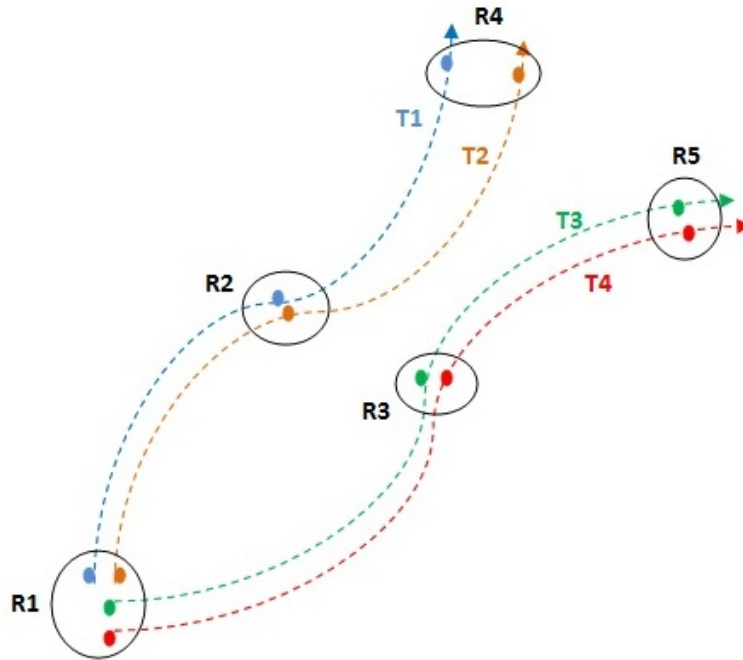
Consider a trajectory  $T_i$  which is represented as a sequence of GPS points i.e.  $T_i = (x_t^i, y_t^i)$  where  $t = 1, 2, \dots, m_i$  denotes the times at which the GPS points in the trajectories are recorded and  $m_i$  is the length of trajectory  $T_i$ . While detecting the changes within a trajectory, we aim to find the time  $t$  at which GPS points in a particular trajectory show an abnormal behavior. On the other hand, while detecting the changes between trajectories, we typically have a sequence of trajectories  $T_1, T_2 \dots$  etc. and each one has a different start



**Figure 23.** Translation of trajectory 2 so as to start from time 0.

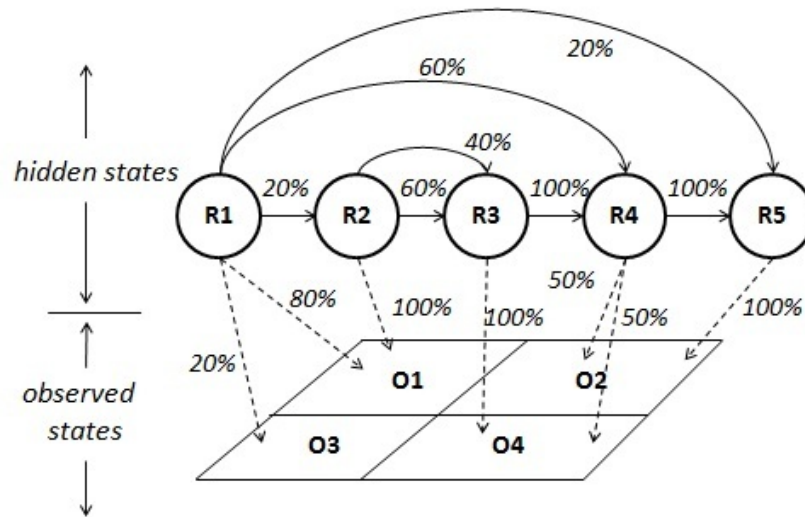
time. For example,  $T_j$  may represent the trajectory of a cab in day  $j$ , for  $j = 1, 2, \dots$ . In this example, each trajectory occurs in a separate day. In order to focus on similarities (or dissimilarities) between trajectories, we translate each trajectory along the time axis so that each is considered to start at time  $t = 0$ . That is, if trajectory  $T_j = (x_t^j, y_t^j)$  starts at time  $t_0$ , we translate it to  $(x_{t-t_0}^j, y_{t-t_0}^j)$ . Figure 23 shows two trajectories 1 and 2, where trajectory 2 is translated so as to start from time  $t = 0$ . We aim to detect a change in the sequence of trajectories. This is the time at which a new cluster of trajectories begins to emerge.

For the purpose of change detection in trajectories, we have modeled trajectories using a Hidden Markov Model (HMM). A HMM is a statistical tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence and satisfy the Markovian property. The system being modeled using a HMM is assumed to have hidden variables or states. Each hidden state in a HMM can generate an observation according to specified probabilities. In order to model the trajectory data using HMM, we need to define the states and observations of the HMM.



**Figure 24.** Modeling trajectories using HMM. The regions  $R1$ ,  $R2$ ,  $R3$ ,  $R4$  and  $R5$  are the dense regions in space traversed by the four objects.

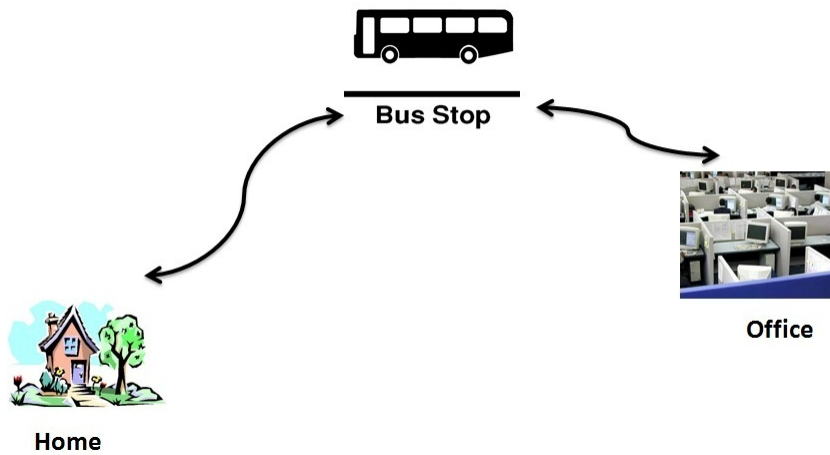
To model a set of trajectories using HMM, the dense regions occurring in a set of trajectories are found. The dense regions are clusters of points, which are frequent in most of the trajectories. These regions are labeled and the labels define the finite states of the trajectories. Fig. 24 gives an illustrative example of sample trajectories being modeled using a HMM. Consider the four objects with trajectories  $T_1$ ,  $T_2$ ,  $T_3$  and  $T_4$ , which have moved in the space for three time stamps as shown in the figure. Let there be some dense regions occurring in the space transversed by the four trajectories. These dense regions are clusters of points from various trajectories which move closer to each other. The dense regions  $R1$ ,  $R2$ ,  $R3$ ,  $R4$  and  $R5$  are shown in the Fig. 24.



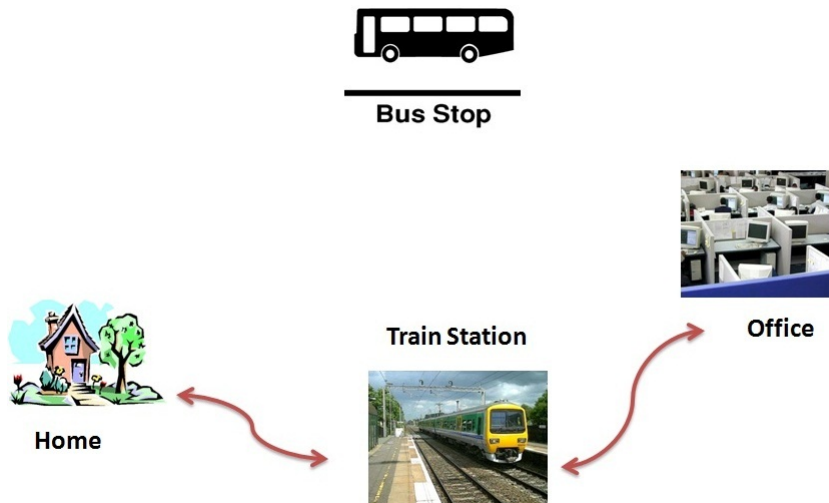
**Figure 25.** Learning the parameters of the HMM using the trajectories.

Each hidden state emits an observation based on a Gaussian distribution. To learn the transition probability matrix and the emission probabilities of the HMM, the observations  $(x_t, y_t)$  from a trajectory are mapped to corresponding hidden state (calculating the probability using emission probability of each Gaussian distribution). Fig. 25 shows the HMM modeled using trajectories when the observation space is discretized using grid-based discretization. The points denoted by  $(x_t, y_t)$  are the points of a trajectory in the observation space, which are discretized using the grids  $O1, O2, O3, O4$ . The regions  $R1, R2, R3, R4, R5$  shown in Fig. 25 show the dense regions, which are the hidden states. Each hidden state, gives an observation  $O1, O2, O3, O4$ . The transition probabilities and emission probabilities are learnt from the trajectories.

Once the trajectories are modeled using HMM, the change detection problem in trajectories is mapped to change detection in parameters of a HMM. To illustrate this, consider Fig. 26 shows the trajectory of a person who commutes from home to work. Fig. 26a

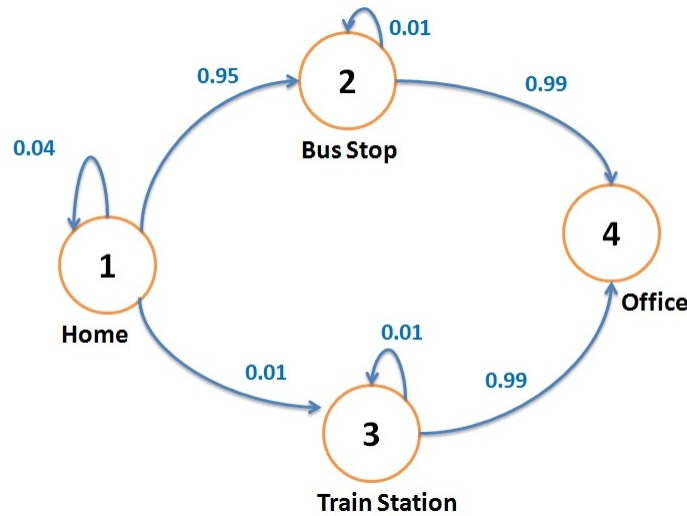


**Figure 26a.** Trajectory of daily commute of a person in May



**Figure 26b.** Trajectory of daily commute of the person in June

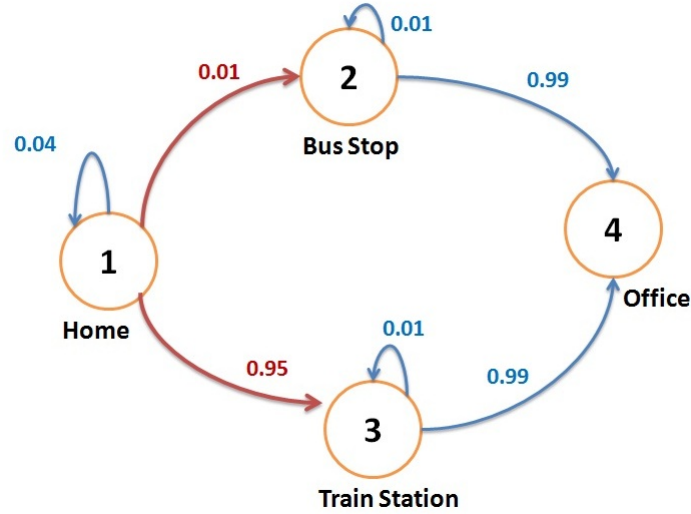
shows the trajectory of daily commute of the person in May where he travels from home to bus stop and to office. However, the person takes the train station instead of the bus in June, thus leading to a different trajectory as shown in Fig. 26b. For the example mentioned in Fig. 26, to model the trajectories using HMM, the various locations the person visits such as home, bus stop, train station and office are modeled as the states of the HMM.



**Figure 27a.** Transition probabilities in May. The transition probability from state 1 to state 2 is 0.95 and that from state 1 to state 3 is 0.01.

The transition probabilities for various states change as shown in Fig. 27, where home is considered to be state 1, bus stop as state 2, train station as stop 3 and office as state 4. As seen in Fig. 27a, the probability of transition from state 1 to state 2 in May is 0.95 while the same probability in June is 0.01 in Fig. 27b. Similarly, the transition probability from state 1 to state 3 in May is 0.01 (Fig. 27a) which raises to 0.95 in June (Fig. 27b). The original problem of change detection of trajectories thus translates to the change detection in the parameters of a HMM.

The problem of detecting changes within a trajectory is addressed in Sections 2, 3 and 4. The points within a trajectory before the change occurs are assumed to be from a null case HMM, while the points after the change are from a shifted case HMM. We aim to detect the time at which the parameters of a HMM change. The change detection between trajectories is addressed in Section 5. The parameters of a HMM are learnt using a set of training trajectories which are used to detect the changes in the test trajectories.



**Figure 27b.** Transition probabilities in June. The transition probability from state 1 to state 2 is 0.01 (compared to 0.95 in May) and that from state 1 to state 3 is 0.95 (compared to 0.01 in May).

## 2. CHANGE DETECTION OF A HMM WHEN THE PARAMETER AFTER CHANGE IS KNOWN

This section deals with detecting changes within a trajectory. The change detection problem of trajectories is translated to change detection in the parameters of a HMM. We aim to detect the time at which the parameters of a HMM change, when the parameters after change are known.

In [50], Gerencser et al. considered the problem of change detection in the statistical pattern of a hidden markov process when the parameters of the HMM, before and after the change are known. The specific change detection problem they address is as follows:

$$\begin{aligned} \theta^* &= \theta_0 \text{ for } n \leq \tau^* - 1 \\ &= \theta_1 \text{ for } n \geq \tau^* \end{aligned}$$



for an unknown  $\tau^*$ , but for given  $\theta_0$  and  $\theta_1$ . Their goal was to estimate  $\tau^*$ . Hence, this is a change-point problem, where the main interest is to find the time at which the change occurs.

A basic method for detecting temporal changes in an independent sequence of observations called the Cumulative Sum algorithm or Hinkley-detector [51] was adapted by them for the change detection process. A cumulative score  $S_n$  was calculated from a statistic  $R_t$ , which is used to monitor the process, i.e.

$$S_n = \sum_{t=1}^n R_t$$

The Hinkley-detector  $g_n$  defined in terms of  $S_n$  is  $g_n = S_n - \min_{0 \leq k \leq n} S_k$ . An alarm is generated if  $g_n$  exceeds a specified threshold.

We aim to detect the time at which the parameter of a HMM changes. We used a concept similar to Hinkley-detector for change detection. The statistic  $R_t$  used by them [50] for change detection is

$$R_t = -\log P(y_t | y_{t-1}, \dots, y_0; \theta)$$

We used a similar statistic

$$R_t = \log P(y_t, y_{t-1}, \dots, y_0; \theta)$$

which calculates the logarithm of likelihood of a set of observations given the parameter values. The statistic is cumulated across the various observations to exactly detect the change point. The details of the exact statistic  $L(t)$  used for change detection are discussed below.

To detect the change in the parameter of a HMM assuming the value of the parameter after the change is known, we adopted the following method. Consider a HMM with a finite

state space  $S$ , with  $N$  states i.e.  $S$  is the set containing the symbols for all  $N$  states. Let  $O$  be the observation space i.e.  $O$  is the set containing the symbols for all the  $M$  observations:

$$S = (s_1, s_2, \dots, s_N)$$

$$O = (o_1, o_2, \dots, o_M)$$

Let the pair  $(x_t; y_t)$  where  $t = 1, 2, \dots, n$  be the HMM where  $x_t \in S$  and  $y_t$  is the observation sequence such that  $y_t \in O$ .

Let the parameters of the HMM be  $\theta = (A, B, \pi)$ . The transition probability matrix,  $A$ , is the matrix consisting of transition probabilities from state  $i$  to state  $j$  in one step  $a_{ij}$ :

$$A = [a_{ij}], a_{ij} = P(x_t = s_j | x_{t-1} = s_i)$$

Each state in a HMM can generate an observation according to specified probabilities. For each state  $s_i$  and each possible output  $o_k$ ,  $b_i(k)$  gives the probability that observation  $o_k$  is emitted in state  $s_i$ .  $B$  is the observation array, storing the emission probabilities:

$$B = [b_i(k)], b_i(k) = P(y_t = o_k | x_t = s_i)$$

$\pi$  is the initial probability array:

$$\pi = [\pi_i], \pi_i = P(x_1 = s_i)$$

Let  $y_1, y_2, y_3, \dots, y_n$  be the  $n$  observations obtained by the HMM. Let  $\theta_0$  denote the parameters of the HMM under the null case and  $\theta_1$  denote the shifted case, when the parameters of the HMM have changed. Some of the changes of interest are the change in the probabilities of the transition probability matrix. The statistic,  $L(t)$ , used for detecting the change in the parameter of HMM is

$$L(t) = \sum_{l=1}^t \log[p(y_l; \theta_0)] - \sum_{l=1}^t \log[p(y_l; \theta_1)] \quad (3.1)$$

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>1</b>	$\Theta$	$\Theta^2$	$\Theta - \Theta^2$	$1-2\Theta$
<b>2</b>	0	0	$\Theta$	$1-\Theta$
<b>3</b>	$\Theta^2 + \Theta$	0	0	$1-\Theta^2-\Theta$
<b>4</b>	0	$1-\Theta$	$\Theta$	0

**Figure 28a.** Transition probability matrix in terms of  $\Theta$  for the case when parameter after change is known.

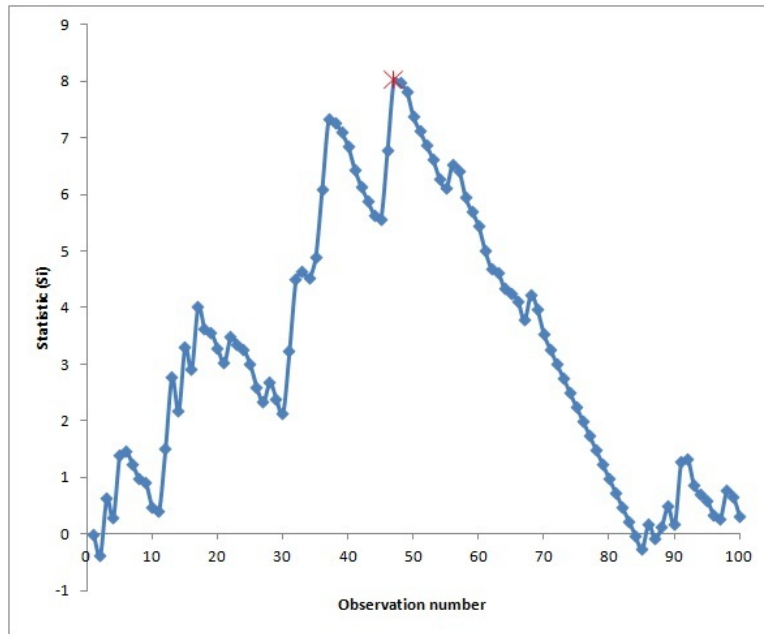
where  $L(t)$  is the difference of log-likelihood value of the observations from 1 to  $t$  under the null case  $\theta_0$  and the log-likelihood value of the observations from 1 to  $t$  under the shifted case  $\theta_1$ . The variation of statistic  $L(t)$  with the observation number  $t$  is used for change detection. The statistic  $L(t)$  increases with  $t$  if no change occurs. If a change in the parameter of HMM occurs at time  $t_0$ ,  $L(t)$  increases till  $t \leq t_0$  and decreases thereafter. This happens because for  $t \leq t_0$ , the value of  $\sum_{i=1}^t \log[p(y_i; \theta_0)]$  would be larger than  $\sum_{i=1}^t \log[p(y_i; \theta_1)]$  and the difference increases as more points are available for estimation. Beyond  $t_0$ , as the observations are more likely to occur from a HMM with parameters  $\theta_1$ , for  $t \geq t_0$ ,  $\sum_{i=t_0}^t \log[p(y_i; \theta_1)]$  would start increasing and hence the value of  $L(t)$  starts decreasing.

## **2.1. EXPERIMENTS FOR CHANGE DETECTION WHEN THE PARAMETER AFTER CHANGE IS KNOWN**

We had parameterized the HMM with a single parameter  $\Theta$ , which determines the transition probability matrix. For a fixed state HMM with  $N = 4$  states and  $M = 6$  observations, the parameters of a HMM are shown in Fig. 28, where the transition probability matrix is shown in Fig. 28a which is dependent on  $\Theta$ . To simplify the number of pa-

	a	b	c	d	e	f
1	0	0	0.25	0.25	0	0.5
2	0.3	0	0.3	0.2	0.2	0
3	0	0.5	0.2	0	0	0.3
4	0.6	0.1	0	0.2	0.1	0

**Figure 28b.** Emission probabilities for the case when parameter after change is known.



**Figure 29.** Likelihood ratio statistic  $L(t)$  versus observation number or  $t$ . The first 50 observations correspond to null case parameters while the next 50 observation correspond to shifted case. The statistic  $L(t)$  begins to decrease after 50th observation confirming the change.

rameters, the emission probabilities are assumed to be independent of  $\Theta$  as shown in Fig. 28b.

To illustrate the above method, 100 points (or observations) were simulated. Let these 100 points (or observations) be denoted as  $(y_1, y_2, y_3 \dots y_{100})$ . Consider the case with a  $\Theta$  value of 0.3 to be the null case, while the case with  $\Theta$  value of 0.1 to be the shifted case. The

first 50 points  $(y_1, y_2, y_3 \cdots y_{50})$  were generated from the HMM with emission probability matrix as shown in Fig. 28b, transition probability matrix as shown in Fig. 28a and a  $\Theta$  value of 0.3. The initial probabilities are  $(0.25, 0.25, 0.25, 0.25)$  for the 4 states. The next 50 points  $(y_{51}, y_{52}, y_{53} \cdots y_{100})$  were generated from a similar HMM with a  $\Theta$  value of 0.1.

The statistic  $L(t)$ , as described above, is plotted against the observation number  $t$  as shown in Fig. 29. It can be seen from the figure that the statistic  $L(t)$  for detecting the change in the parameter  $\Theta$  reaches a maximum value at value of  $t$  approximately equal to 50 and then starts decreasing thereafter. Hence, the statistic is able to detect the change in the parameter value  $\Theta$  and the time at which the change occurs.

### 3. CHANGE DETECTION OF A HMM WHEN THE PARAMETER AFTER CHANGE IS UNKNOWN

This section deals with detecting changes within a trajectory. The change detection problem of trajectories is translated to change detection in the parameters of a HMM. We aim to detect the time at which the parameters of a HMM change, when the parameters after change are unknown.

Gerencser et al [49] consider the problem of change point detection for HMMs when the parameter after change is unknown. Let the pair  $(x_t; y_t)$  where  $t = 1, 2, \cdots n$  be the HMM where  $x_t$  is a homogenous Markov chain such that  $x_t \in S$  and  $y_t$  is the observation sequence such that  $y_t \in O$ . Their estimation procedure was as follows, instead of cumulating past data, it is gradually forgotten by using an exponential forgetting. If  $\lambda$  is the forgetting rate and  $\theta$  are the parameters, the quantity they have used for change detection is

$$\sum_{t=1}^n (1 - \lambda)^{n-t} \lambda \log p(y_t | y_{t-1}, \cdots, y_0; \theta)$$

The factor  $(1 - \lambda)^{n-t}$  is used for weighing the observations, so that the past observations have lower weights while the most recent ones have higher weight.

Our method differs from [49] in the way we choose the points and the statistic used for change detection. Instead of using an exponential forgetting rate, we choose a moving window of points with fixed window size. We also investigated the effect of window size on detecting the change.

In the previous section, we have considered the simpler case where the parameter after change is known. However, in most cases (e.g while dealing with trajectory data), the parameter after change is generally unknown. In this section, we adopted the following method for detecting the change in the parameter of a HMM when the parameter after change is unknown.

Let the parameters of the HMM be  $\theta = (A, B, \pi)$ . Let  $y_1, y_2, y_3, \dots, y_n$  be the  $n$  observations obtained by the HMM. Consider a window of observations of size  $w (\leq n)$  from the HMM:

$$W(t) = (y_{t-w+1}, y_{t-w+2}, \dots, y_t)$$

where  $t$  varies from  $w$  to  $n$ . Let  $\theta_0$  denote the parameters of the HMM under the null case. Let  $L_0(t)$  denote the log-likelihood of observations in window  $W(t)$  with respect to the parameters of the HMM under null case i.e

$$L_0(t) = \log[p(y_{t-w+1}, y_{t-w+2}, \dots, y_t; \theta_0)]$$

Similarly let  $L_1(t)$  be the log-likelihood of the observations in window  $W(t)$  with respect to the parameters obtained by estimation using Baum-Welch algorithm [29] i.e.

$$L_1(t) = \log[p(y_{t-w+1}, y_{t-w+2}, \dots, y_t; \hat{\theta})]$$

	1	2
1	0.7	0.3
2	0.1	0.9

	1	2
1	0.3	0.7
2	0.9	0.1

**Figure 30a.** Transition probability matrix under null case (Left) and shifted case (Right) for change detection when parameter after the change is known.

The statistic  $L(t)$  used for detecting the change is:

$$L(t) = L_1(t) - L_0(t)$$

where  $L(t)$  measures the difference between the log-likelihood of the window of observations under null case  $\theta_0$  and the log-likelihood of the window of observations using the estimated parameters  $\hat{\theta}$ . The statistic  $L(t)$  is plotted against  $t$ , as  $t$  varies from  $w$  to  $n$  to detect the changes.

If there is a change in the parameters of the HMM, the statistic  $L(t)$  would show a shift as  $t$  changes, since the likelihood values are expected to differ significantly, hence the difference between  $L_1(t)$  and  $L_0(t)$  increases. Consider a time  $t_0$  at which the parameters of the HMM change. For  $t \leq t_0$ , the estimated value  $\bar{\theta}$  and the null value  $\theta_0$  would be close to each other. Hence the value of  $L(t)$  would be small. For  $t \geq t_0$ , the estimated value  $\bar{\theta}$  would be significantly different from the null value  $\theta_0$ . Hence the value of  $L(t)$  would be large.

### **3.1. EXPERIMENTS FOR CHANGE DETECTION WHEN THE PARAMETER AFTER CHANGE IS UNKNOWN**

Experiments were conducted to illustrate the above method. Consider a fixed state binary HMM with  $N = 2$  states and  $M = 2$  observations. The transition probability matrix and the emission probability matrix for the null case are shown in Fig. 30a and Fig. 30b

Observations	States	
	1	2
a	0.9	0.2
b	0.1	0.8

**Figure 30b.** Emission probability matrix under null & shifted case for change detection when parameter after the change is known.

respectively. The transition probability matrix and the emission probability matrix for the shifted case are shown in Fig. 30a and Fig. 30b respectively. It can be seen that the emission probability matrix doesn't change from the null to the shifted case.

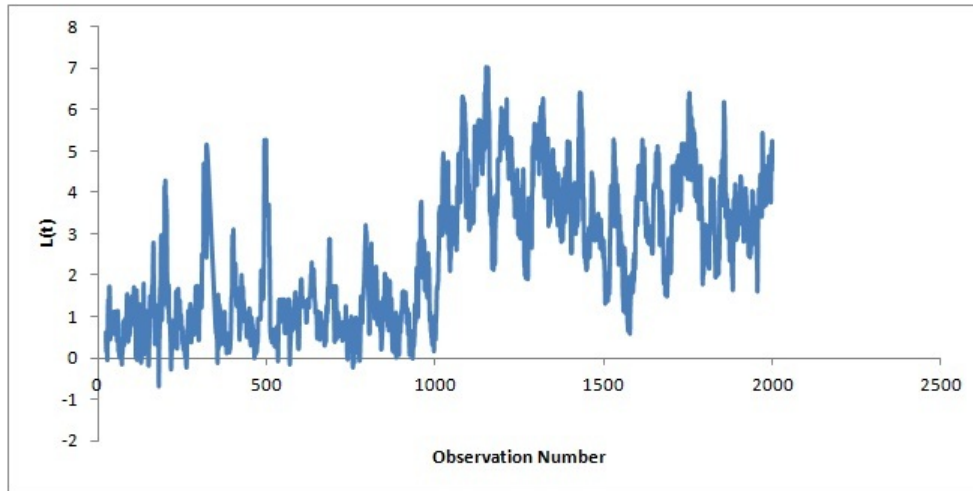
To illustrate the performance of the method with increased number of observations, 2000 points (or observations) were simulated. The first 1000 points simulated with parameters of HMM as in the null case and the next 1000 points simulated with parameters of HMM under the shifted case. The window size is varied from 25 to 50 observations. For each window size, the statistic  $D_t$  as described before, is plotted against the observation number  $t$ .

The results for window sizes of 25 and 50 are shown in Fig. 31a and Fig. 31b. These plots resemble control charts used for process control. There is a clear indication of process change at 1000th observation in all the plots. The signal is more stronger but delayed as the window size increases, since more observations would be available for estimating the parameters of the HMM if window size is more.

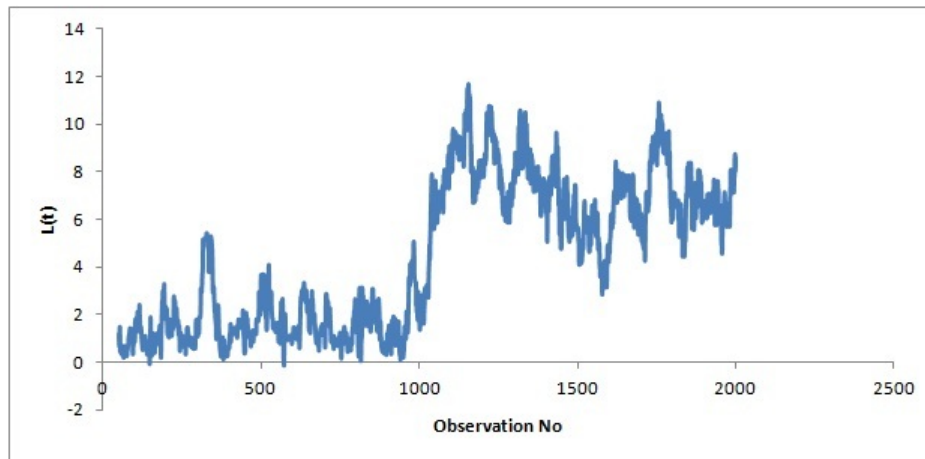
#### 4. DETECTING SPECIFIC PARAMETER CHANGES IN A HMM

This section deals with detecting changes within a trajectory. While detecting changes within a trajectory, it is of interest to detect specific regions of the trajectory where the change could have occurred. The change detection problem of trajectories is translated to





**Figure 31a.** Likelihood ratio statistic  $L(t)$  versus observation number with a window size  $w = 25$ . The first 1000 observations correspond to null case parameters while the next 1000 observation correspond to shifted case. The statistic  $L(t)$  begins to increase after 1000th observation confirming the change.



**Figure 31b.** Likelihood ratio statistic  $L(t)$  versus observation number with a window size  $w = 50$ . The first 1000 observations correspond to null case parameters while the next 1000 observation correspond to shifted case. The statistic  $L(t)$  begins to increase after 1000th observation confirming the change.

change detection in the parameters of a HMM. To detect such changes, we need to detect specific parameter changes of a HMM i.e. if the state transition probability from only one state has changed.

In order to detect such changes, we have proposed a new modified Baum-Welch (m-BaumWelch) algorithm. To illustrate this change, consider a trajectory being modeled using a HMM which is described by the pair  $(x_t; y_t)$  where  $t = 1, 2, \dots, n$ ,  $x_t$  is a Markov chain with state space of four states  $(s_1, s_2, s_3, s_4)$  and  $y_t$  is the observation sequence belonging to four finite observations  $(o_1, o_2, o_3, o_4)$ . Let the transition probability matrix of this HMM be given by a matrix

$$A = [a_{ij}], a_{ij} = P(x_t = s_j | x_{t-1} = s_i)$$

such that

$$\sum_j a_{ij} = 1 \quad \forall i \quad \text{where } i, j \in \{1, 2, 3, 4\}$$

The proposed method is used to detect specific parameter changes in the transition probability matrix of HMM. For the above example, if  $a_{ij}$  for  $i = 1$  changes, while those for  $i = 2, 3, 4$  remain unchanged, we would be able to detect this change. Since  $a_{ij}$ 's for a fixed  $i$  are dependent, if  $a_{ij}$  changes, the other values have to change to satisfy the equation  $\sum_j a_{ij} = 1$  i.e if  $a_{11}$  changes,  $a_{12}, a_{13}, a_{14}$  also have to change. Hence, the method is used to detect the particular row in the transition probability matrix of the HMM that has changed.

#### **4.1. *m*-BAUMWELCH ALGORITHM**

The Baum-Welch (BW) algorithm is a class of expectation-maximization algorithms, which computes the maximum likelihood estimates of the parameters (transition and emission probabilities) of a HMM, when given only emissions as training data [86]. The m-BaumWelch(m-BW) algorithm that we have proposed enforces constraints for parameter estimation in a HMM. Unlike the original algorithm which estimates all the parameters of

the HMM, the m-BW algorithm fixes the specific parameters which have to be estimated and leaves the remaining parameters unaltered.

The m-BW algorithm is based on the principles of the original BW algorithm. It uses the expectation-maximization principle and computes the parameter estimates by maximizing the likelihood values for the parameters of the HMM given the observation sequence. The details of the algorithm including the input and the steps involved are described.

Let  $y_1, y_2, y_3, \dots, y_n$  be the observation sequence that is known, containing  $n$  observations, from which the specific parameters of the transition probability matrix of a HMM have to be re-estimated. The algorithm takes as input a parameter  $r$  which denotes the row number of the transition probability matrix that has to be estimated. This is the main difference from the original BW algorithm, where we have the flexibility to change only specific row of the transition probability matrix while the remaining rows remains unchanged, unlike in the traditional BW algorithm, where all the rows of the transition probability matrix and the emission probabilities are estimated.

The steps involved in the algorithm are as follows:

1. Input the initial parameters of an  $N$  state HMM i.e.  $\theta = (A, B, \pi)$ .
2. Let the new parameter being estimated be  $\bar{\theta} = (\bar{A}, \bar{B}, \bar{\pi})$  where the transition probability matrix (TPM) is  $\bar{A}$ . Re-estimate  $\bar{A}$  as follows

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{n-1} P(x_t = s_i, x_{t+1} = s_j | Y, \theta)}{\sum_{t=1}^{n-1} P(x_t = s_i | Y, \theta)} \quad \forall i = r, j \in 1, 2, \dots, N$$

$$\bar{a}_{ij} = a_{ij} \quad \forall i \neq r, j \in 1, 2, \dots, N$$

3. The values of  $\bar{B}$  and  $\bar{\pi}$  are as follows

$$\bar{B} = B$$

$$\bar{\pi} = \pi$$

4. Check for convergence:

$$|\log[p(y_1, y_2, y_3, \dots, y_n|\theta)] - \log[p(y_1, y_2, y_3, \dots, y_n|\bar{\theta})]| \leq \delta$$

i.e. loglikelihood ratio of the observations given the new estimate of  $\bar{\theta}$  and the old value of  $\theta$  is less than a threshold  $\delta$ . If converged stop.

5. Else  $\theta \leftarrow \bar{\theta}$ , go to step 2.

#### **4.2. CHANGE DETECTION USING *m*-BAUMWELCH ALGORITHM**

In this section, we describe how the new m-BW algorithm is used to detect specific parameter changes of the transition probability matrix of a HMM. Let  $y_1, y_2, y_3, \dots, y_n$  be the  $n$  observations obtained by the HMM. Consider a window of observations of size  $w$  ( $\leq n$ ) from the HMM

$$W(t) = (y_{t-w+1}, y_{t-w+2}, \dots, y_t)$$

where  $t$  varies from  $w$  to  $n$ . Let  $\theta_0$  denote the parameters of the HMM under the null case. Let  $L_0(t)$  denote the log-likelihood of observations in window  $W(t)$  with respect to the parameters of the HMM under null case

$$L_0(t) = \log[p(y_{t-w+1}, y_{t-w+2}, \dots, y_t; \theta_0)]$$

Let  $\hat{\theta}_r$  denote the parameters of the HMM estimated using the m-BW algorithm with input parameter as  $r$ , where  $r$  varies from 1 to  $N$ . For each value of  $r$ , let  $L_{1r}(t)$  be the log-likelihood of the observations in window  $W(t)$  with respect to the parameter  $\hat{\theta}_r$  i.e.

$$L_{1r}(t) = \log[p(y_{t-w+1}, y_{t-w+2}, \dots, y_t; \hat{\theta}_r)]$$

States	1	2
1	0.95	0.05
2	0.05	0.95

States	1	2	3	4
1	0.45	0.55	0	0
2	0	0	0.9	0.1

**Figure 32a.** TPM: Null case (Left) & Emission: Null and Shifted cases (Right) for change detection using m-BW algorithm with two hidden states.

The statistic  $L_r(t)$  used for detecting the change is:

$$L_r(t) = L_{1r}(t) - L_0(t)$$

where  $L_r(t)$  is the log-likelihood ratio of the window of observations. The statistic  $L_r(t)$  is plotted against  $t$ , as  $t$  varies from  $w$  to  $n$ , for each value of  $r$  to detect the specific changes. If there is a change in a specific row  $l$  of the transition probability matrix of the HMM, the statistic  $L_r(t)$  would show a shift as  $t$  changes, when  $r = l$ . But for the rest of the values of  $r$ , the statistic  $L_r(t)$  would not show any significant shift.

### 4.3. EXPERIMENTS FOR CHANGE DETECTION USING *m*-BW ALGORITHM

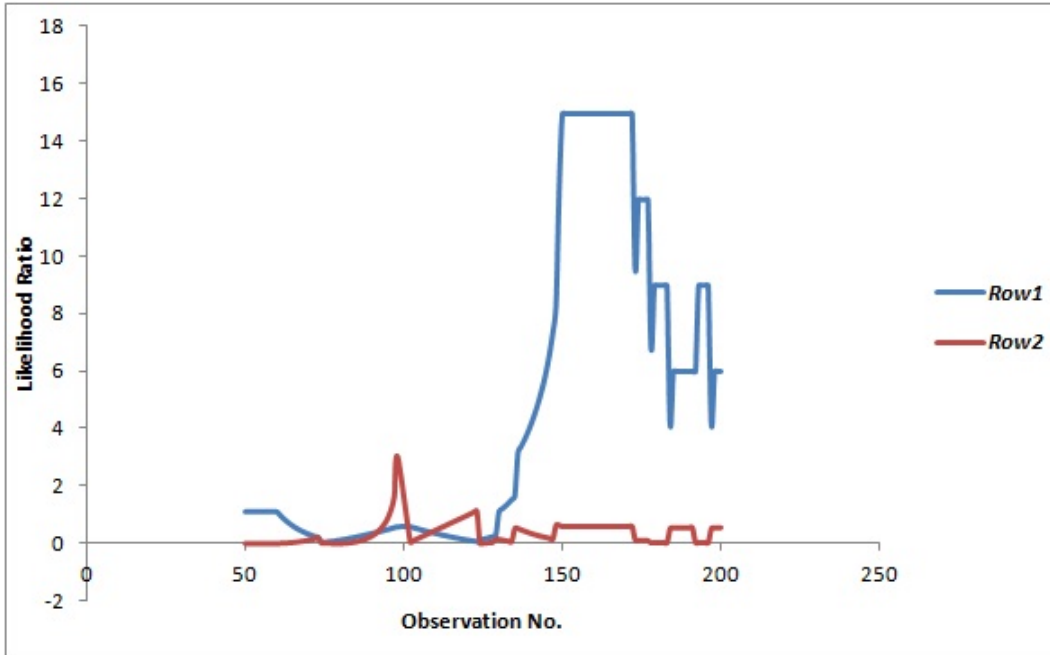
Experiments were conducted to illustrate the m-BW algorithm. Consider a fixed state binary HMM with  $N = 2$  states and  $M = 4$  observations. The null case transition probability matrix and the emission probabilities are shown in Fig. 32a. Two cases of shift in the transition probability matrix are considered. In case I, as shown in Fig. 32b, only the transition probabilities from state 1 change while those from state 2 remain unchanged. On the other hand, in case II shift, as shown in Fig. 32b, only the transition probabilities from state 2 change while those from state 1 don't change. We have incorporated a rather large shift in the transition probabilities to illustrate how our method works.

For case I shift, 200 points were simulated with the first 100 points from the HMM with parameters under null case, while the next 100 points from the HMM with TPM under

States	1	2
1	0.05	0.95
2	0.05	0.95

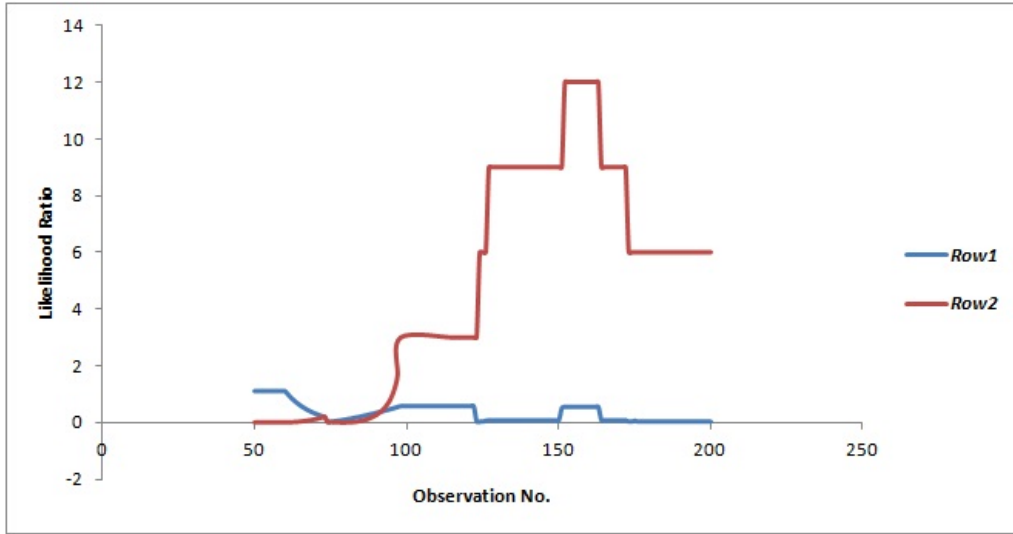
States	1	2
1	0.95	0.05
2	0.95	0.05

**Figure 32b.** TPM: Shifted case I (Left) & Shifted case II (Right) for change detection using m-BW algorithm with two hidden states.



**Figure 33.** Likelihood ratio statistic  $L_r(t)$  versus observation number for detecting case I shift. The first 100 observations correspond to null case parameters while the next 100 observation correspond to shifted case I. The statistic  $L_1(t)$  begins to increase after 100th observation confirming the change in the first row of the TPM. The statistic  $L_2(t)$  doesnot change for the 200 observations because the second row of the TPM doesnot change.

shifted case I. A sliding window of size  $w = 50$  points is chosen. For each position of the sliding window, the m-BW algorithm is run two times using the input parameters as  $r = 1$  and  $r = 2$  to estimate only the first or second row of the TPM, respectively. The statistic  $L_r(t)$  for  $r = 1, 2$ , as mentioned earlier, is plotted against the observation number or  $t$ , as  $t$  varies from 50 to 200.



**Figure 34.** Likelihood ratio statistic  $L_r(t)$  versus observation number for detecting case II shift. The first 100 observations correspond to null case parameters while the next 100 observation correspond to shifted case II. The statistic  $L_2(t)$  begins to increase after 100th observation confirming the change in the second row of the TPM. The statistic  $L_1(t)$  does not change for the 200 observations because the first row of the TPM does not change.

States	1	2	3	4
1	0.994	0.006	0	0
2	0	0.56	0.436	0.004
3	0.062	0.083	0.755	0.101
4	0.2	0	0	0.8

States	1	2	3	4
1	0.006	0	0.994	0
2	0	0.56	0.436	0.004
3	0.062	0.083	0.755	0.101
4	0.2	0	0	0.8

**Figure 35a.** TPM: Null case TPM (Left) & Shifted case TPM (Right) for change detection using m-BW algorithm with four hidden states.

The results for case I shift are shown in Fig. 33. From the plot in Fig. 33, it can be seen that  $L_r(t)$  values for  $r = 1$  show a change as the position of the sliding window changes, while the values for  $r = 2$  remain unchanged.

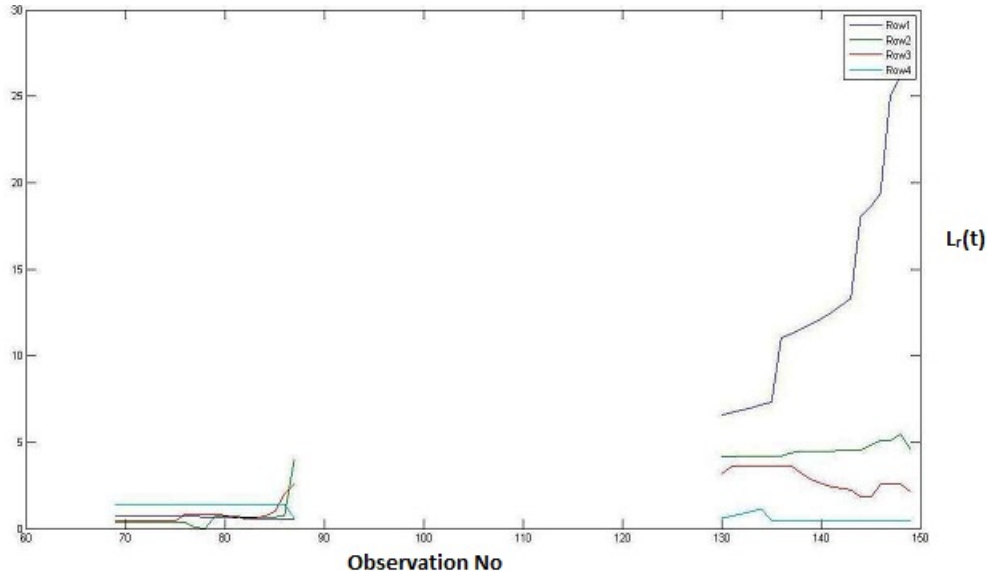
The results for case II shift, where only the parameters of the second row of TPM change are shown in Fig. 34. The plot shows the  $L_r(t)$  values changing for  $r = 2$  while those for  $r = 1$  remain unchanged, thus confirming the change.

States	Emissions				
	1	2	3	4	5
1	0.45	0.55	0	0	0
2	0	0	0.8	0.2	0
3	0	0	0	0.9	0.1
4	0.1	0	0	0	0.9

**Figure 35b.** TPM: Null & Shifted case Emission probabilities for change detection using m-BW algorithm with four hidden states.

Experiments were conducted to test the m-BW algorithm for increasing number of hidden states and the observations of the HMM. Consider a HMM with with  $N = 4$  states and  $M = 5$  observations. The null case TPM and the shifted case TPM are shown in Fig. 35a, while the emission probabilities used in both cases is shown in Fig. 35b. From Fig. 35a, it can be seen that only the transition probabilities from state 1 i.e. row 1 change in the TPM while the other values remaining values remain unchanged. We generated 100 observations using the HMM with null parameters, followed by the next 100 observations from the HMM with shifted parameters. The m-BW algorithm was used to detect the change in the parameters from these 200 observations using a window size of 50. The results are shown in Fig. 36, where  $L_r(t)$  is plotted for  $r = 1, 2, 3, 4$  against the observation number. The observation number in the figure corresponds to  $t$  in  $L_r(t)$ . The results are shown for 20 observations before the change and 20 observations after change. It can be seen from the Fig. 36 that the  $L_r(t)$  value for  $r = 1$  has a higher value for  $t = 130, 131, \dots, 150$  compared to when  $t = 70, 71, \dots, 90$ , thus confirming that the first row of TPM has changed. The values of  $L_r(t)$  for  $r = 2, 3, 4$  remain unchanged, hence confirming that the second, third and fourth row probabilities of the TPM did not change.



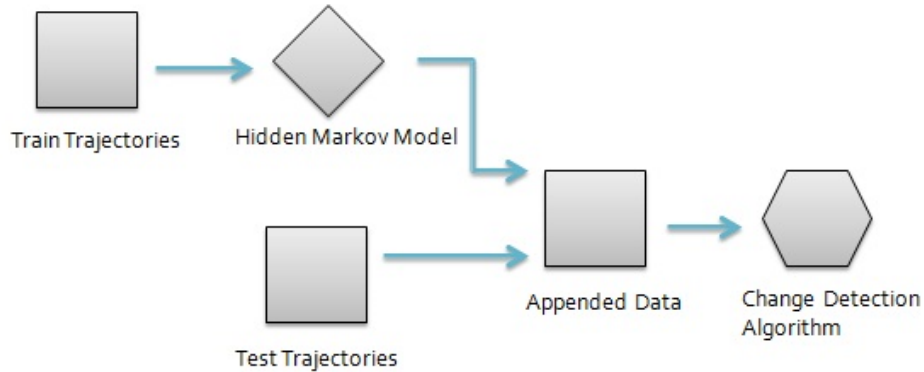


**Figure 36.** Likelihood ratio statistic  $L_r(t)$  versus observation number for  $r = 1, 2, 3, 4$ . The first 100 observations correspond to null case parameters while the next 100 observations correspond to shifted case. The plot shows the  $L_r(t)$  values for only 20 observations before the change and 20 observations after the change. The statistic  $L_r(t)$  when  $r = 1$  is higher for the 20 observations after the change compared to the 20 before the change, thus confirming the change.

## 5. CHANGE DETECTION IN TRAJECTORY DATA

The previous sections dealt with detecting changes within a trajectory i.e. changes in observations from a HMM. In this section, we describe how the change detection framework for a HMM, explained in the previous sections, is used to detect changes between trajectories. The modeling framework used in change detection of trajectories is shown in Fig. 37.

The difference between the change detection framework in this section compared to other sections is that, in the earlier sections the observations from a single HMM, which can be considered to be GPS points of a single trajectory, are used for change detection. In this section, we consider observations from a set of different trajectories and aim to



**Figure 37.** Modeling framework for change detection in trajectory data.

detect changes in trajectories as a whole. The difference in the type of HMM used in this section compared to earlier sections, is that we use a continuous state HMM unlike the finite state HMM used earlier. Since the trajectories are represented as a sequence of continuous GPS points ordered in time, we use a continuous state HMM to model the trajectory data. A set of trajectories  $T_1, T_2, \dots, T_n$  are modeled using a HMM, where the GPS points in the trajectories are considered to be the observations of the HMM. The dense regions or clusters occurring in the trajectories are considered to be the unknown or hidden states of the HMM. The number of states of the HMM are learnt from the training trajectories. A detailed explanation of the modeling framework is given in Section 3.1. Each trajectory  $T_i$  is represented as follows,  $T_i = (x_t^i, y_t^i); t = 1, 2, \dots, m_i$  where  $m_i$  is the length of trajectory  $T_i$  for  $i = 1, 2, \dots, n$ . This modeling framework allows the trajectories  $T_1, T_2, \dots, T_n$  to be of varying lengths.

The method used for change detection is as follows. A set of training trajectories  $T_1, T_2, \dots, T_k$  for  $k \leq n$  are used to build a continuous HMM. Let  $\theta_0$  denote the parameters of the continuous HMM modeled using the training trajectories. Consider a new set of test trajectories  $T_{k+1}, T_{k+2}, \dots, T_n$ . A new data set consisting of  $n$  trajectories is created by appending the training and testing trajectories. Consider a window of size  $w \leq n$ , consisting of  $w$  trajectories i.e.

$$W(t) = (T_{t-w+1}, T_{t-w+2}, \dots, T_t)$$

where  $t$  varies from  $w$  to  $n$ . This window is different from the earlier sections, as the window in the earlier sections consists of a set of observations but here we use a set of trajectories. Let  $L_0(t)$  denote the log-likelihood of the trajectories in window  $W(t)$  with respect to the parameters of the HMM obtained from the training trajectories i.e

$$L_0(t) = \log[p(T_{t-w+1}, T_{t-w+2}, \dots, T_t; \theta_0)]$$

Similarly, let  $L_1(t)$  be the log-likelihood of the trajectories in sliding window  $W(t)$  with respect to the parameters of the HMM obtained by estimation using Baum-Welch algorithm:

$$L_1(t) = \log[p(T_{t-w+1}, T_{t-w+2}, \dots, T_t; \hat{\theta})]$$

The statistic  $L(t)$  used for change detection is:

$$L(t) = L_1(t) - L_0(t)$$

where  $t \geq w$ .

To test the efficiency of our method a control chart is built to monitor  $L(t)$ . The control charts can be used to test the change detection models used in previous sections as well.

However, a detailed explanation of how to use the control charts for change detection is described only in this sections. The control limits were chosen using the training trajectories.

The control limits are

$$UCL : \mu + 3\sigma/\sqrt{w}$$

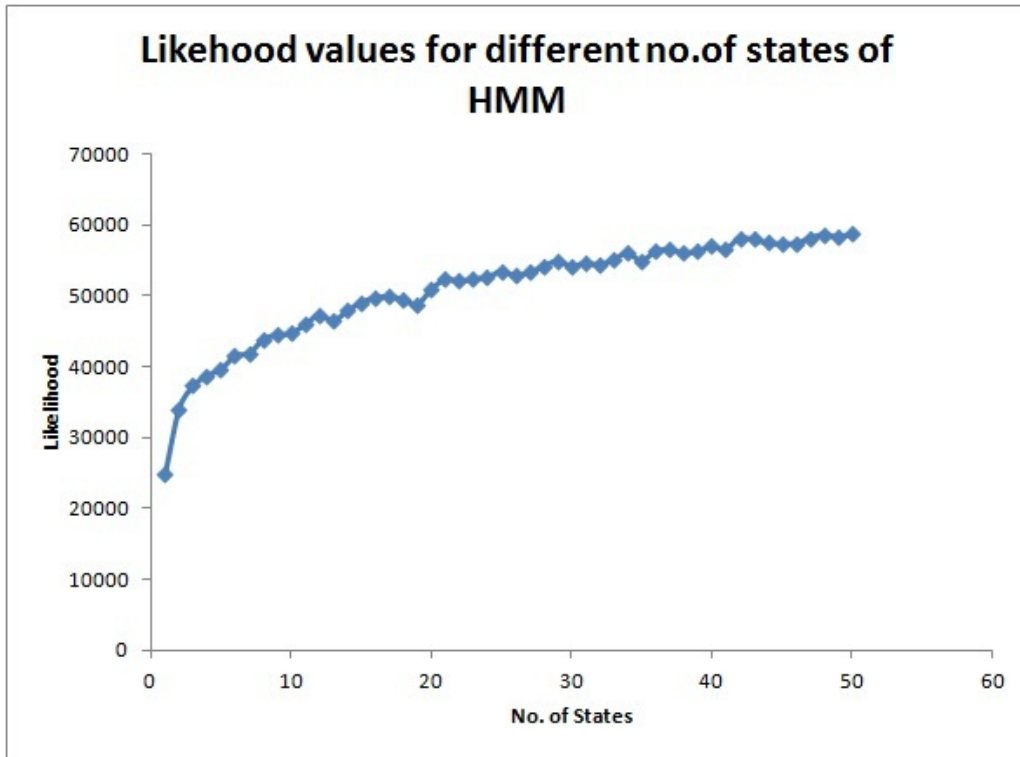
$$LCL : \mu - 3\sigma/\sqrt{w}$$

where  $\mu$  is the mean of the  $L(t)$  values of the in-control data or training trajectories,  $\sigma$  is the standard deviation of the  $L(t)$  values of the training trajectories and  $w$  is the window size. Once the control limits are set, the run length for the first out-of-control signal to occur is found. These run lengths are calculated for different combinations of training and testing trajectories and different window sizes.

We have calculated the autocorrelation function [85] of the statistic  $L(t)$  at different time lags and found that the autocorrelation for the data used in the experiments was low. However, if there is autocorrelation in the data, one approach to deal with autocorrelation is to sample from the process data stream less frequently i.e. by avoiding data from adjacent windows in the trajectory data. But this would lead to an inefficient use of available data. Another approach in dealing with autocorrelated data is to directly model the correlative structure with an appropriate time series model, use that model to remove the autocorrelation from the data, and apply control charts to the residuals [87].

## **5.1. DATA GENERATION**

To illustrate the applicability of our method with real trajectory data, we have used a dataset that contains mobility traces of taxi cabs in San Francisco, USA [88]. The dataset contains GPS coordinates of approximately 500 taxis collected over 30 days in the San Francisco Bay Area with a total of 10990 trajectories. Each trajectory consists of approxi-



**Figure 38.** Variation of likelihood of trajectory data with number of states of the HMM.

mately 1200 to 1500 GPS points. Each San Francisco based yellow cab vehicle is outfitted with a GPS tracking device that is used by dispatchers to efficiently reach customers. The data is transmitted from each cab to a central receiving station, and then delivered in real-time to dispatch computers via a central server.

The daily trajectories of the cabs in the dataset consist of a large amount of points i.e. long trajectories. As a first step in analyzing the trajectories, we reduced the size of the dataset by compressing the original trajectories with minimum loss of information. To compress the trajectory data we used the line simplification method, specifically, the DP (Douglas-Peucker) algorithm [89]. Details of the DP algorithm are provided in Chapter 2. After compression of the dataset using DP algorithm, each trajectory consisted of 150

to 200 GPS points on an average, with a total of 87,2600 points for the 500 cabs. These compressed trajectories of the cabs were used for further analysis. The first step in change detection is to train a continuous HMM using the cab trajectories.

To decide on the number of hidden states to be used for a continuous HMM with Gaussian distribution, one month trajectory data of a single cab was used. The states of the continuous HMM were varied from 2 to 50 and the likelihood of the data was calculated for each of the state configurations of the HMM. The plot of variation of the likelihood values with the states is shown in Fig. 38. From the figure, it can be seen that the likelihood value increases drastically up to 20 states and stabilizes thereafter. Hence, the number of states of the HMM was chosen to be 20 based on the plot. The emission probability distribution i.e. mean and variance of the Gaussian distribution are learnt from the observations.

Our trajectory change detection method is used to detect changes in the trajectories of the cabs with time. Consider the trajectories of a cab for a month. If the trajectories in the first half of the month follow a specific path, while the trajectories start shifting after that, our method can prove useful to detect if and when this change occurs. However, the trajectories of the cabs in the SFO dataset do not follow this pattern of change. We created a simulated dataset from the cab trajectory data to induce the change, which we aim to detect later.

As a first step to generate trajectory data with the induced change, we clustered the trajectories of the cabs. Clustering was done to obtain groups of similarly moving trajectories. The trajectories were clustered using the mixtures of regression model [5]. The details of the clustering framework are provided in Chapter 2. Trajectory data have certain complications which make it difficult to apply standard clustering techniques. Trajectories can be of

different length, hence they cannot be converted to fixed length vectors to apply standard clustering techniques. Trajectories are also a function of the time variable, hence the clustering algorithm should take this kind of dependance into account. Also, the trajectories can be measured at different time points. Hence a mixture of regressions approach is used for clustering the trajectories, where each cluster is modeled as a prototype regression function with some variability around that prototype.

The representative data for change detection was created as follows. Let two clusters from the trajectory data be denoted as cluster 0 and 1 respectively. Consider  $n_0$  trajectories belonging to cluster 0 and  $n_1$  trajectories belonging to cluster 1. Let the trajectories belonging to cluster 0 be denoted as  $T_1^0, T_2^0 \dots T_{n_0}^0$  respectively and the trajectories belonging to cluster 1 as  $T_1^1, T_2^1 \dots T_{n_1}^1$  respectively. A new dataset is created with a total of  $n_0 + n_1$  trajectories  $T_1^0, T_2^0 \dots T_{n_0}^0, T_1^1, T_2^1 \dots T_{n_1}^1$  such that the first  $n_0$  trajectories belong to cluster 0 and the next  $n_1$  trajectories belong to cluster 1. The order of trajectories in the dataset is important to train the HMM. For the purpose of this experiment, the order of trajectories in each cluster is random, but in general time-ordered trajectories should be used. The time to detection would vary with a change in the order of the trajectories in the cluster, which is explained later.

## **5.2. EXPERIMENTS**

The first step was to obtain clusters of trajectories with induced change. The trajectories of first 50 cabs were used for clustering. Each cab had approximately 20 to 24 trajectories, with a total of 1142 trajectories for the 50 cabs. Each trajectory has 90 GPS points on an average. Hence, the total size of the data set is of the order of approximately 100,000 data points. The mixture of regressions method was used to cluster the 1142 trajectories into 20

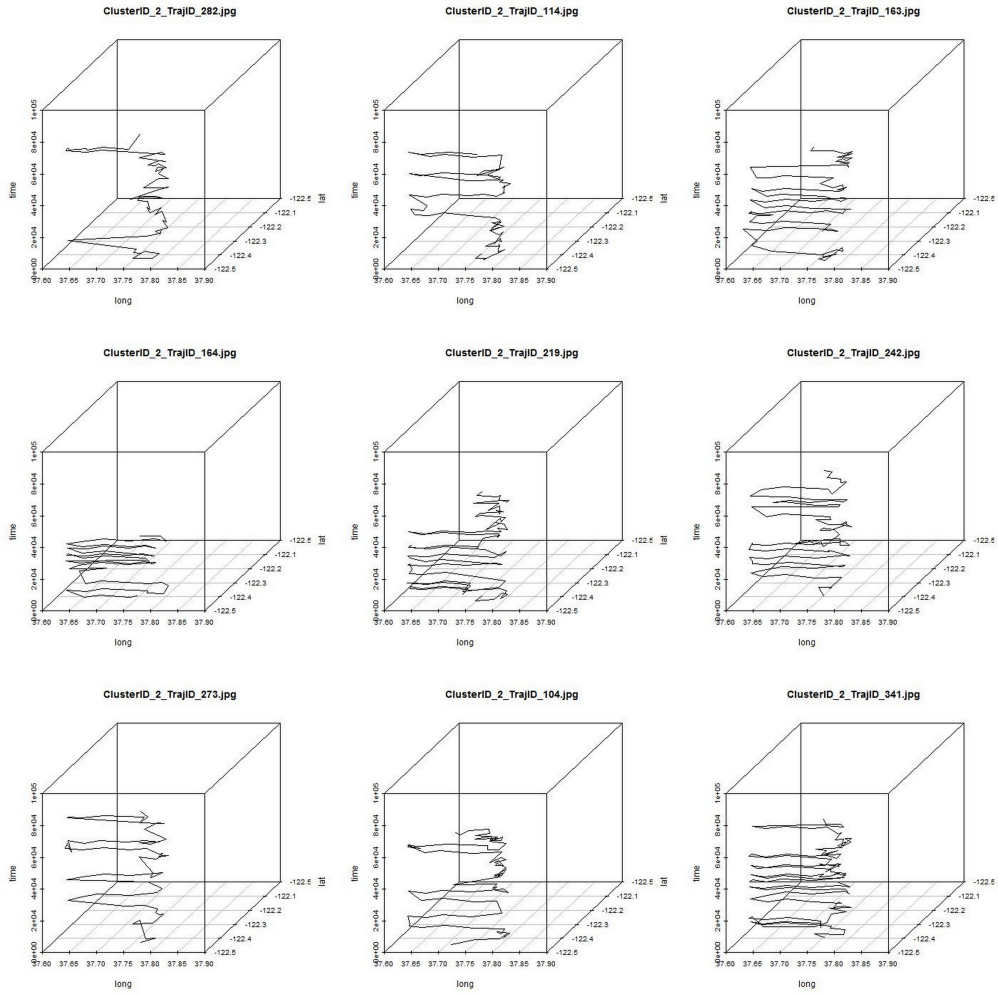
**TABLE 3.** Clusters of trajectories of 50 cabs.

Cluster	Count of Trajectories
1	59
2	79
3	4
4	39
5	148
6	44
7	52
8	131
9	44
10	41
11	24
12	30
13	69
14	58
15	50
16	81
17	79
18	56
19	49
20	5

clusters. The number of clusters were chosen to be 20, as increasing the number beyond 20 would produce clusters with very few trajectories in them and decreasing the number below 20 would produce fewer replicates to conduct experiments for finding the run lengths which is discussed later. The count of trajectories belonging to each cluster is shown in Table 3. It can be seen from the table that some clusters are very dense while the clusters 3 and 20 are very sparse. The 3D plot of a sample of 9 trajectories belonging to cluster 2 is shown in Fig. 39. It can be seen from the figure that the trajectories can be confined to a cube of specified dimensions. This behavior was observed in the trajectories from other cabs as well.

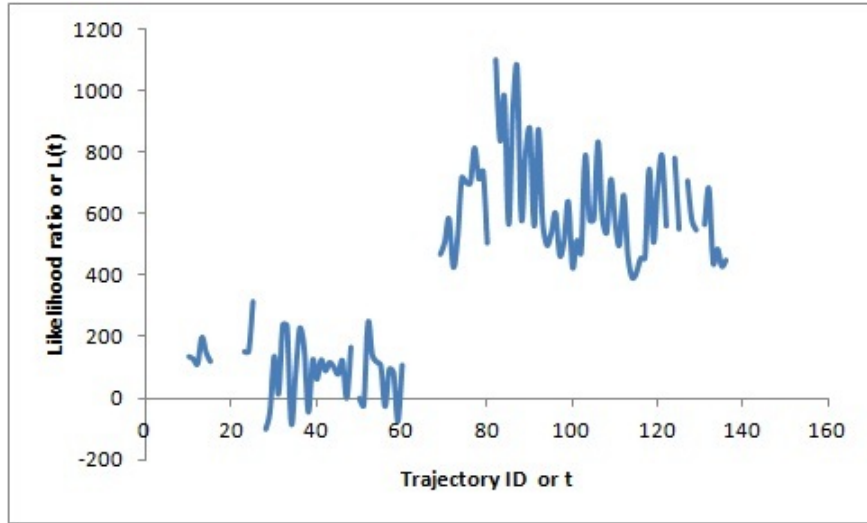
Using the 20 clusters, datasets with induced change as described earlier were generated, by taking two clusters at a time. However, clusters with IDs 3 and 20 were ignored as they



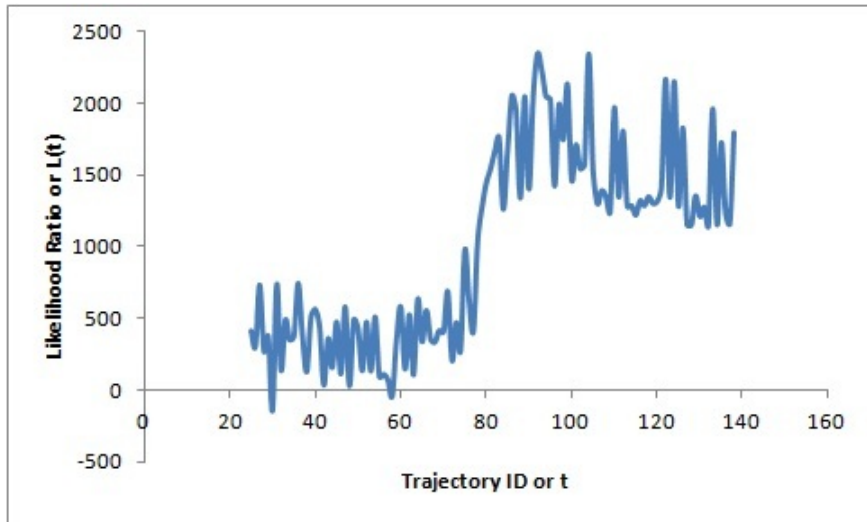


**Figure 39.** Sample trajectories belonging to cluster 2. Clustering is done using the mixture of regressions model.

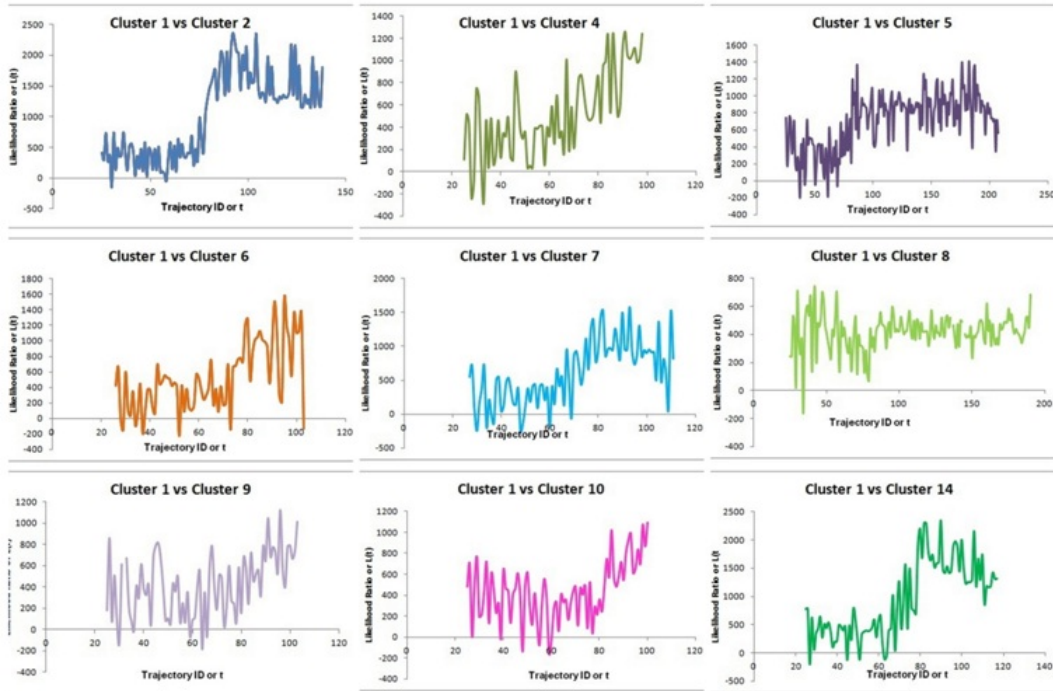
were very sparse. To illustrate the method, let the 59 trajectories belonging to cluster 1 be the training set. A HMM with 20 states is trained using the 59 trajectories belonging to cluster 1 and is considered to be the null case HMM. The trajectories belonging to each of the clusters 2 to 20 (excluding clusters 3 and 20) are used as the testing trajectories to detect the change. Experiments were conducted for two different window sizes - 10 and 25. The likelihood ratio plot for cluster 1 trajectories as the training set and cluster 2 trajectories



**Figure 40a.** Change detection of trajectories from cluster 1 to cluster 2 using window size of 10. The first 59 trajectories belong to cluster 1 while the next 79 trajectories belong to cluster 2. An increase in the likelihood ratio value after the 59th trajectory confirms the change.



**Figure 40b.** Change detection of trajectories from cluster 1 to cluster 2 using window size of 25. The first 59 trajectories belong to cluster 1 while the next 79 trajectories belong to cluster 2. An increase in the likelihood ratio value after the 59th trajectory confirms the change.



**Figure 41.** Sample likelihood ratio i.e.  $L(t)$  vs  $t$  plots using trajectories from cluster 1 as the training data and trajectories from other clusters at the testing data.

as test set for different window sizes is shown in Fig. 40. It can be seen from the figure that the likelihood ratio starts increasing after ID the 59th trajectory and the mean value of the likelihood ratio shifts thereafter. Some values of the likelihood ratio are unavailable for window size of 10 because of the inability of the estimation algorithms such as BW to converge due to insufficient data in the window. However, as the window size increases from 10 to 25 it can be seen that lesser values are missing for the likelihood ratio. Some sample likelihood ratio plots using cluster 1 trajectories as training data and the trajectories from other clusters as the testing data is shown in Fig. 41. It can be seen from the Fig. 41, that the general trend of the plots is that  $L(t)$  value starts to increase after the 59th point, thus depicting a change in the nature of the trajectories.

Cluster ID	1	2	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1		7	1	4	5	5	10	1	8	2	3	11	6	2	5	5	10	3
2	4		2	1	1	3	5	2	1	1	2	2	2	4	3	2	10	4
4	2	1		6	4	2	1	1	1	1	2	4	2	5	4	2	5	2
5	1	1	1		2	3	1	3	1	1	2	2	2	4	5	1	2	3
6	3	1	1	4		2	3	2	3	3	10	4	1	6	7	13	1	1
7	2	10	2	1	5		4	1	2	3	2	1	3	1	1	2	1	2
8	1	1	1	1	1	1		2	1	1	3	1	2	3	1	2	2	1
9	2	2	2	3	1	1	2		1	1	1	1	1	2	1	2	2	2
10	1	1	1	1	4	2	1	1		4	7	1	1	3	1	7	1	1
11	2	4	1	5	3	9	1	1	1		1	9	5	1	1	4	1	2
12	1	3	1	2	1	1	4	1	1	1		2	6	1	1	2	6	2
13	2	1	1	1	1	1	1	1	1	1	1		1	2	1	1	2	1
14	31	3	1	4	6	1	1	1	2	2	1	2		5	6	2	6	2
15	1	3	3	6	1	3	7	2	6	1	2	4	3		2	1	9	5
16	2	10	4	5	1	2	34	3	2	1	1	9	7	3		4	10	33
17	1	1	5	2	1	2	3	1	1	1	2	4	1	1	2		2	1
18	2	1	3	2	1	1	2	1	1	2	3	2	1	1	3	1		1
19	1	6	2	3	2	1	6	3	3	1	5	4	1	5	2	7	5	

**Figure 42.** Run length values to get an out-of-control signal using control charts for different combination of training and testing trajectories with window size of 10.

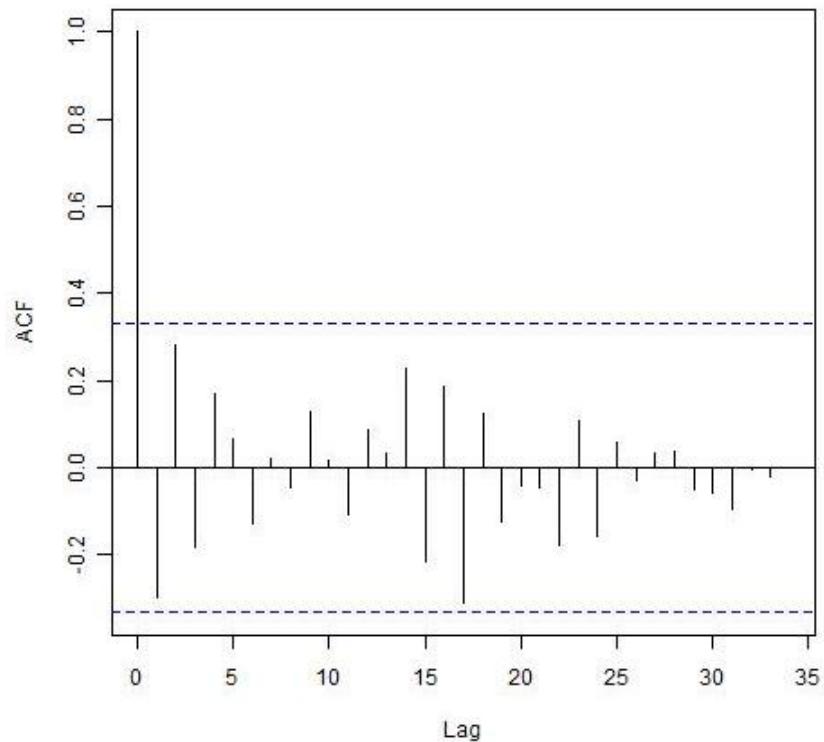
Similar experiments were conducted by varying the training set i.e choosing cluster 1 through cluster 20 (excluding cluster 3 and 20) trajectories as the training set. For each of the dataset consisting of a set of training and testing trajectories, a control chart is built using the training trajectories as the in-control data. The run length of the control chart is the number of points in the testing data till which an out-of-control signal occur i.e. the point is outside the control limits. The run length is calculated using the testing trajectories for different window sizes. The results of the run lengths for window size of 10 are shown in Fig. 42. Similar results for a window size of 25 are shown in Fig. 43. From the figures it can be seen that the run lengths for a window size of 10 are greater than that for a window size of 25. The average value of the run lengths for cluster 1 across the clusters 2 through

Cluster ID	1	2	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1		1	2	2	1	1	6	1	1	2	3	3	1	1	1	1	1	3
2	2		3	1	1	1	1	2	1	1	3	1	1	1	1	1	1	1
4	3	5		3	1	5	4	4	1	1	1	3	1	1	2	3	3	1
5	1	1	3		2	1	1	1	1	1	1	1	1	1	1	9	3	2
6	1	1	1	2		6	1	2	4	1	1	1	1	2	1	1	2	3
7	1	3	2	3	1		2	1	1	2	2	2	4	3	1	3	3	4
8	1	3	1	1	1	1		1	1	1	1	1	1	1	1	2	1	1
9	1	1	4	1	1	1	1		2	1	2	1	1	1	1	1	1	1
10	1	2	1	1	1	1	3	1		6	1	1	2	1	1	2	5	1
11	NA	NA	NA	NA	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA	NA	NA	NA
12	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA	NA	NA
13	1	2	1	2	1	2	3	1	2	1	1		2	4	2	1	1	2
14	1	1	1	1	1	1	5	2	1	1	1	1		1	1	1	2	1
15	1	3	1	2	1	1	1	2	3	1	1	1	1		1	3	2	4
16	5	1	2	1	1	1	6	4	3	1	2	2	4	2		1	2	2
17	1	1	2	1	2	2	1	1	2	1	1	3	1	1	2		4	1
18	1	1	1	2	1	2	2	1	1	1	1	1	1	2	1	2		1
19	1	1	1	1	1	1	1	1	1	2	1	2	1	1	2	1	1	

**Figure 43.** Run length values to get an out-of-control signal using control charts for different combination of training and testing trajectories with window size of 25.

19 is 5.17 using window size of 10, while the same value is 1.82 for window size of 25. The maximum value of the run lengths using window size of 10 is 34, while it is only 9 using window size of 25. Also, in Fig. 43 the run length values for cluster ID 11 and 12 are not available because the number of trajectories in the clusters are 24 and 30 which are comparable to the window size of 25. Hence, the in-control data for these clusters could not be obtained.

The autocorrelation between  $L(t)$  values of in-control data for cluster 1 trajectories shown in Fig. 40b, for a window size of 25, is shown in Fig. 44. By examining the plot in Fig. 44, it can be seen that there is low autocorrelation in the data. The autocorrelation values for rest of the clusters of trajectories was also calculated. It was found that, the auto-



**Figure 44.** Autocorrelation between  $L(t)$  values of in-control data for cluster 1 trajectories for a window size of 25.

correlation between the  $L(t)$  values for the training data was low, for all the data sets where the training data was cluster 2 through cluster 19 trajectories

## 6. CONCLUSIONS

A change detection method was developed to detect changes in trajectories with time. HMMs were used to model the trajectories, where the GPS points in the trajectories were considered to be the observations of the HMM. The problem of detecting changes in trajectories with time, was mapped to detecting the changes in the parameters of the HMM. Real trajectory data obtained from the cabs was used to test the method. Trajectories with

induced change were simulated using the cab trajectory data. Our method when applied to these trajectories, was able to successfully detect the changes.

### **DETECTING CHANGES IN PATTERNS OCCURRING IN TRAJECTORY DATA**

With the advent of technologies like GPS, sensors etc., the movement of people and vehicles can be observed from the digital traces left by them which are being collected by wireless devices. For example, the trajectories of moving people can be collected by analyzing the positioning logs left by the mobile phones. Similarly, vehicles such as cabs in a region can be traced by the GPS systems present in the vehicles which record the latitude-longitude position at different time instants. Such information can be useful to discover usable knowledge about the movement behavior.

Discovering frequent sequential patterns in trajectory data, called trajectory patterns, would be extremely useful in various domains like traffic management, tourism recommendations etc. Such trajectory patterns which show the cumulative behavior of moving objects in a specific region can help us understand various mobility-related phenomenon. A trajectory pattern represents a set of individuals that visit the same sequence of places. Knowledge gained from such trajectory patterns, can help promote tourism by providing banners at the frequently visited places about other famous places in the region to visit.

Trajectory patterns extracted from a set of trajectories can change with time. For example, the sequence of places visited by tourists last year might change this year. Detecting such changes in trajectory patterns is important to identify the obsolete patterns and new patterns that are emerging with time. We have proposed two different methods for detecting changes in the trajectory patterns. We have conducted experiments using real trajectory data of cabs in San Francisco, to illustrate our method. Our method can be applied more generally to any sequential data from which frequent sequential patterns can be extracted e.g. customer market data.



**TABLE 4.** Notation table

Notation	Description
$D_0$	Original data consisting of $R_0$ sequences
$P_0$	Set of frequent sequential patterns extracted from $D_0$
$D_1$	Test data consisting of $R_1$ sequences
$R$	Number of sequences in a sample obtained from $D_0$ or $D_1$
$D_0^n$	$n$ th replicate containing $R$ sequences, where each of the $R$ sequences are obtained by sampling from the $R_0$ sequences in $D_0$ without replacement
$N$	Number of replicates obtained from the training data $D_0$
$D_1^m$	$m$ th replicate containing $R$ sequences, where each of the $R$ sequences are obtained by sampling from the $R_1$ sequences in $D_1$ without replacement
$M$	Number of replicates obtained from the test data $D_1$
$p$	Frequent sequential pattern in $P_0$ , denoted by $(p_1, p_2, \dots, p_Q)$ where $p_q \in I$ for $q = 1, 2 \dots Q$
$Q$	Number of items in pattern $p$
$I$	Set of items denoted by $\{i_1, i_2, \dots, i_W\}$
$W$	Total no.of items in the database $D_0$ or $D_1$

**TABLE 4.** Notation table

Notation	Description
$s$	Sequence denoted by $(s_1, s_2, \dots, s_L)$ where $s_l \in I$ for $l = 1, 2 \dots L$
$S_0^n(j)$	$j$ th statistic calculated for a pattern $p$ in replicate $D_0^n$
$S_1^m(j)$	$j$ th statistic calculated for a pattern $p$ in replicate $D_1^m$
$C_j$	Control chart built using the $j$ th measure
$f(p_{q'}, p_q, s)$	Function which calculates the average time between occurrence of the items $p_{q'}$ and $p_q$ of a pattern $p$ in a sequence $s$
$t_0^n(q', q)$	Average time between occurrence of an item $p_q$ after item $p_{q'}$ of pattern $p$ in a replicate $D_0^n$ calculated using all sequences in $D_0^n$ where $p$ occurs
$t_1^m(q', q)$	Average time between occurrence of an item $p_q$ after item $p_{q'}$ of pattern $p$ in a replicate $D_1^m$ calculated using all sequences in $D_1^m$ where $p$ occurs
$T_0^n$	$Q - 1$ dimensional vector representing the average times between the $Q$ items of pattern $p$ in a replicate $D_0^n$ calculated using all sequences in $D_0^n$ where $p$ occurs

**TABLE 4.** Notation table


Notation	Description
$T_1^m$	$Q - 1$ dimensional vector representing the average times between the $Q$ items of pattern $p$ in a replicate $D_1^m$ calculated using all sequences in $D_1^m$ where $p$ occurs
$t_0$	Sampling time between items in sequences $D_0$
$t_1$	Sampling time between items in sequences $D_1$ , equal to $\delta t_0$
$\epsilon$	Threshold used in BIRCH clustering
$TR$	Trajectory represented as a sequence of points $(x_t, y_t)$ , where $t = 1, 2, \dots, T$
$TR_u$	$u$ th trajectory among $U$ trajectories represented as a sequence of points $(x_t^u, y_t^u)$ where $t = 1, 2, \dots, T_u$ and $u = 1, 2, \dots, U$
$v_t^u$	Feature vector extracted from a trajectory $TR_u$ i.e. $v_t^u = [x_t^u, y_t^u, x_{t+1}^u, y_{t+1}^u]$
$V_u$	Set of $T_u - 1$ feature vectors extracted from a trajectory $TR_u$ i.e. $V_u : \{v_t^u\}$ where $t = 1, 2, \dots, T_u - 1$
$V$	Set containing feature vectors extracted from all $U$ trajectories i.e. $V : \{V_u\}$ where $u = 1, 2, \dots, U$
$C$	Set of $K$ clusters obtained by clustering feature vectors belonging to $V$ i.e. $C : \{c_1, c_2, \dots, c_K\}$

**TABLE 4.** Notation table

Notation	Description
$D_0^n(p)$	Set of all sequences in $D_0^n$ in which the pattern $p$ appears
$g$	Subgroup size used in hotelling $T^2$ control chart

In Section 1, we have defined a sequential pattern and the problem of change detection in sequential patterns. The algorithm for frequency-based approach is described in Section 1.1, and the algorithm for distribution-based approach is described in Section 1.2. In Section 2, we describe the preprocessing technique used for trajectories. The trajectories are first compressed to reduce the size of the trajectory data. The compressed trajectories are then discretized to convert the continuous valued GPS points in trajectories to discrete clusters. The procedure used for converting a trajectory into a sequence of items is described in Section 2. The real trajectory data that has been used and the discretization parameters are described in Section 3. Experiments for the frequency-based approach with low level of discretization are shown in Section 3.1 and those with high level of discretization are shown in Section 3.2. Experiments for the distribution-based approach with high level of discretization are shown in Section 3.3 while those with low level of discretization are shown in Section 3.4. The notation used in the chapter is shown in Table 4.

ID	Sequence
1	{a, b, c}
2	{a}
3	{c, d, e}
4	{a, b, c, e}
5	{a, b, b, c}
6	{d, c}
7	{a, e, b}
8	{a, b, c, d}
9	{d, e}
10	{a, b, c, b}



Pattern	Support
{a}	7/10
{c}	6/10
{a, b}	5/10
{a, b, c}	5/10

**Figure 45.** Sequence database consisting of 10 sequences and the set of frequent sequential patterns with minimum support of 0.5 extracted from them.

## 1. CHANGE DETECTION IN SEQUENTIAL PATTERNS

Sequential Pattern Mining (SPM) [59] is a technique used to find the relationships between occurrences of sequential events, to find if there exist any specific order of the occurrences. An example of sequential patterns is that every time Microsoft stock drops 5%, IBM stock will also drop at least 4% within three days. SPM finds applications in a wide range of areas since many types of data have a time-stamp associated with it. Business organizations use SPM to study customer behaviors. Sequential patterns can be extracted from web log analysis, which are very useful to better structure a company's website for providing easier access to the most popular links [62]. It is also used in intrusion detection [63] and DNA sequence analysis [64].

In our present work we propose a change detection framework for sequential patterns in streaming data. The specific problem that we address is to detect if a new stream of

data has the same sequential patterns as that of the original data from which the sequential patterns are extracted. Consider a dataset consisting of thousands of records from which sequential patterns are extracted. The frequent sequential patterns are those patterns whose frequency count is greater than a specified minimum threshold. When a new stream of sequential data arrives the sequential patterns from this data could be similar to the original sequential patterns or significantly different. It is of interest to know if the discovered frequent sequential patterns in the original data are still frequent. Also, if the patterns continue to exist in the new data, do they occur at similar time intervals as the original patterns or has the time interval distribution changed. If the patterns are changing in the new data stream than it would be a concern as the sequential patterns from the original data would be obsolete and would not provide sufficient knowledge for decision making.

We have addressed this problem using the following two approaches. In the frequency based approach, the statistics such as relative support, information gain etc. for each frequent sequential pattern in the original data are compared with those from the new data to detect changes. This method is used to detect if the previously found frequent patterns are still frequent in the new data. In the distribution based approach, the distribution of the time between the items in the frequent patterns is used to detect changes in the new data. Using the distribution based approach, we can detect if the time of occurrence of items in the patterns in the new data have the same distribution as the original patterns.

Consider a database  $D_0$  consisting of  $R_0$  sequence. Each sequence in  $D_0$  is an ordered set of items e.g. the sequence with ID 1 in the figure is the ordered set of items  $\{a, b, c, e, a, c, f\}$ . Let  $I$  be the set of items denoted by  $\{i_1, i_2, \dots, i_W\}$  where  $W$  represents the total number of items in  $D_0$ . Let  $s$  be a sequence in  $D_0$ , denoted by  $(s_1, s_2, \dots, s_L)$

Traj ID	Time	GPS_x	GPS_y	Discretized ID
1	12:00	123.51	67.81	L7
1	12:05	123.8	66.7	L11
1	12:10	124.2	69.5	L15
1	12:15	128.8	65.4	L22
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

**Figure 46.** An example depicting a trajectory which is a sequence of GPS points being denoted as a sequence of items.

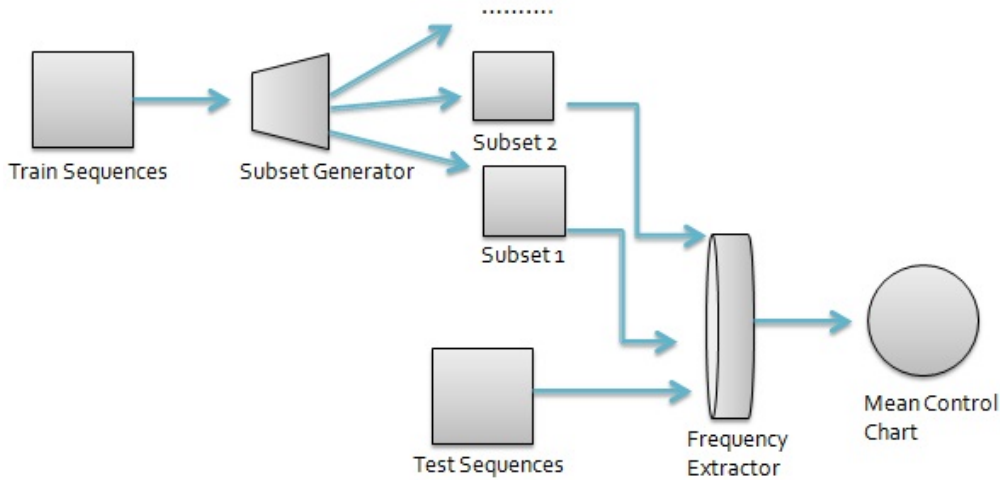
where  $s_l \in I$  for  $l = 1, 2 \dots L$ . An example of a sequence database consisting of 10 sequences and the frequent sequential patterns extracted from it are shown in Fig. 45.

Let  $p$  be a frequent sequential pattern extracted from  $D_0$  and be denoted by  $(p_1, p_2, \dots, p_Q)$  where  $p_q \in I$  for  $q = 1, 2 \dots Q$ . The support of the pattern  $p$  in the sequence database  $D_0$  is the fraction of sequences,  $s$ , that contain the pattern  $p$  as a subsequence i.e.

$$support(p) = \frac{\sum_i I(p \subset s_i)}{R_0}$$

The pattern  $p$  is frequent if it's support is greater than a user specified minimum support. As an example, the pattern  $\{a, b, c\}$  in Fig. 45 is a frequent sequential pattern. It occurs in 5 out of 10 sequences in the database. Hence the support of pattern  $\{a, b, c\}$  is 5/10.

The general notion of a sequence as a set of items is described here. A trajectory can also be represented as a sequence of items. A trajectory is a time-ordered sequence of GPS points. To convert a trajectory which is an ordered sequence of points to a sequence of items, the GPS points in the trajectory were discretized. A detailed explanation of the



**Figure 47.** Modeling framework of the frequency-based method for change detection in sequential patterns. Subsets are extracted from the training data by sampling without replacement. Frequency based statistics obtained from these subsets are compared with the test data to detect changes.

representation of trajectory as a sequence of items, and the notion of an item in the case of a trajectory is given in Section 2. Fig. 46 shows an example of a trajectory being represented as a sequence of items. At each timestamp, the x-coordinate and the y-coordinate of the trajectory are recorded i.e. at time 12:00, the trajectory with ID 1 in Fig. 46 is at location (123.51, 67.81). The location corresponding to (123.51, 67.81) is discretized and assigned an ID  $L7$ . The trajectory in Fig. 46 is thus denoted as a sequence of discretized locations  $(L7, L11, L15, L22 \dots)$ , where each of them can be considered to be an item. All the models developed in this chapter are applicable to data that can be represented as a sequence of items.



### 1.1. FREQUENCY-BASED APPROACH

The frequency-based method is used to detect if the sequential patterns which were frequent in the original data are still frequent in the new data. There is no restriction on the size of the new data as the size of the new data can be different from the original data. A threshold obtained from the control charts is used to make the decision if the sequential patterns in the new data are significantly different from the original data.

Each row in the database  $D_0$  is a sequence. Let  $D_1$  be the new stream of data containing a different set of  $R_1$  sequences. An overview of the modeling framework for the frequency-based method is shown in Fig. 47. The basic idea behind the frequency based approach is that the frequency based statistics like relative support, information gain etc. of a pattern obtained using any replicate of the original data do not differ much. A replicate of the data  $D_0$ , denoted by  $D_0^r$ , is a subset of data  $D_0$  containing  $R$  sequences, where each of the  $R$  sequences are obtained by sampling from the  $R_0$  sequences in  $D_0$  without replacement. These statistics can be used to detect how the patterns in the new stream of data differ from the original data. Original data is sampled multiple times without replacement to obtain replicates from the data. Statistics such as relative support, information gain etc. obtained from these replicates are used for change detection in the new data stream. The detailed steps involved in the frequency-based approach for change detection in sequential patterns are as follows:

1. Frequent sequential patterns are extracted from the original data using PrefixSpan algorithm [59]. Let the set  $P_0$  denote the set containing frequent sequential patterns from the original data  $D_0$  with a support greater than *minsup*.

2. Let  $D_0^n$  where  $n = 1, 2, \dots, N$ , be the  $n$ th replicate containing  $R$  sequences, such that each of the  $R$  sequences are obtained by sampling from the  $R_0$  sequences in  $D_0$  without replacement.
3. Consider a sequential pattern  $p$ , where  $p \in P_0$ . For the pattern  $p$ ,  $J$  statistics  $S_0^n(1), S_0^n(2), \dots, S_0^n(J)$  are calculated for each replicate  $D_0^n$  where  $n = 1, 2, \dots, N$ . Given the data  $D_0$ , some of the measures that can be extracted for a pattern  $p$  from replicate  $D_0^n$  are relative support, information gain, growth rate etc. [90]. The relative support of pattern  $p$  in sample  $D_0^n$ , is the ratio of the support of the pattern  $p$  in sample  $D_0^n$  to the number of sequences in the original data  $D_0$ .

$$\text{Relative Support (RS)} = \frac{\text{support}(p, D_0^n)}{R_0}$$

Information Gain (IG) [90] of a pattern  $p$  w.r.t to a replicate  $D_0^n$  is the logarithm of ratio of the support of a pattern in the replicate  $D_0^n$  to the support of the pattern in the original data  $D_0$  weighted by the number of sequences in both

$$\text{Information Gain (IG)} = \log \frac{\text{support}(p, D_0^n) \times R_0}{\text{support}(p, D_0) \times R}$$

The Growth Rate (GR) [90] of a pattern  $p$  w.r.t to a replicate  $D_0^n$  is proportional to the rate of change of the support of the pattern  $p$  in the replicate  $D_0^n$  compared to the support of pattern in  $D_0$

$$\text{Growth Rate (GR)} = \frac{R_0 - R}{R_0} \times \frac{\text{support}(p, D_0^n)}{\text{support}(p, D_0) - \text{support}(p, D_0^n)}$$

4. Let  $D_1^m$  where  $m = 1, 2, \dots, M$ , be the  $m$ th replicate containing  $R$  sequences, such that each of the  $R$  sequences are obtained by sampling from the  $R_1$  sequences in  $D_1$

without replacement. Similarly  $J$  statistics,  $S_1^m(1), S_1^m(2), \dots, S_1^m(J)$ , for the pattern  $p$  are calculated relative to the new data stream  $D_1^m$ .

5. Each of the measures  $S_1^m(j)$  where  $j = 1, 2, \dots, J$  is compared with the corresponding values of the measure  $S_0^n(j)$  where  $n = 1, 2, \dots, N$  obtained from  $D_0^n$ . If there is a considerable difference between the two, a change has occurred. To detect the change, a 3-sigma control chart [85] is built for each of the  $J$  measures i.e. there are  $J$  different control charts. For each chart  $C_j$ , where  $j = 1, 2, \dots, J$ , the control limits are calculated using  $S_0^n(j)$ , where  $n = 1, 2, \dots, N$ , as the in-control data. The value  $S_1^m(j)$  from new data  $D_1^m$  is plotted on the control chart to check if it falls within the control limits or is an out-of-control signal thus indicating a change.

## **1.2. DISTRIBUTION-BASED APPROACH**

The distribution-based method is used to detect if the time-interval distribution between items in the patterns from the new data is different from those in the original data. This method takes into account the multiple occurrence of items of pattern in the observed sequences, while finding an average value of time between items in a pattern. Also, there is no restriction on the number of items in a pattern, since we use a vector to store the time intervals between different items in the patterns. There is also no restriction on the number of times an item can occur in the patterns. To check for changes in multiple patterns, each pattern needs to be checked for a change in the distribution of items in that specific pattern.

Consider a sequential database  $D_0$  as shown in Fig. 45. The sequential database consists of  $R_0$  sequences, where each sequence is an ordered list of items. Here we explain the distribution-based method with respect to items in general, which can be extended to tra-

jectory data. Consider  $I = \{i_1, i_2, \dots, i_W\}$  to be a set of  $W$  items, as defined earlier. Let  $s$  be a sequence denoted by  $(s_1, s_2, \dots, s_L)$  where  $s_l$  is an item, i.e.  $s_l \in I$  for  $1 \leq l \leq L$ . Let  $D_1$  be the new stream of data consisting of  $N_1$  sequences.

Let the set  $P_0$  denote the set containing frequent sequential patterns from the original data  $D_0$  with a support greater than *minsup*. Let  $p$  be a sequential pattern in the set  $P_0$ . Let  $p$  be denoted by  $(p_1, p_2, \dots, p_Q)$  where  $p_q \in I$  for  $1 \leq q \leq Q$ . Let a sequence  $s$  of  $D_0$  contain the sequential pattern  $p$ . The time between occurrence of the items  $p_{q'}$  and  $p_q$  in  $p$  with respect to sequence  $s$  is calculated as follows: sequence  $s$  is scanned to find the item  $s_l = p_{q'}$ , let this occur at  $l = a$  and at time  $t_a$ . The remaining items in the sequence from  $l = (a + 1)$  to  $l = L$  are scanned to find the item  $s_l = p_q$ , let this occur at  $l = b$  and at time  $t_b$ . The time between occurrence of items  $p_{q'}$  and  $p_q$  in sequence  $s$  is therefore  $(t_b - t_a)$ .

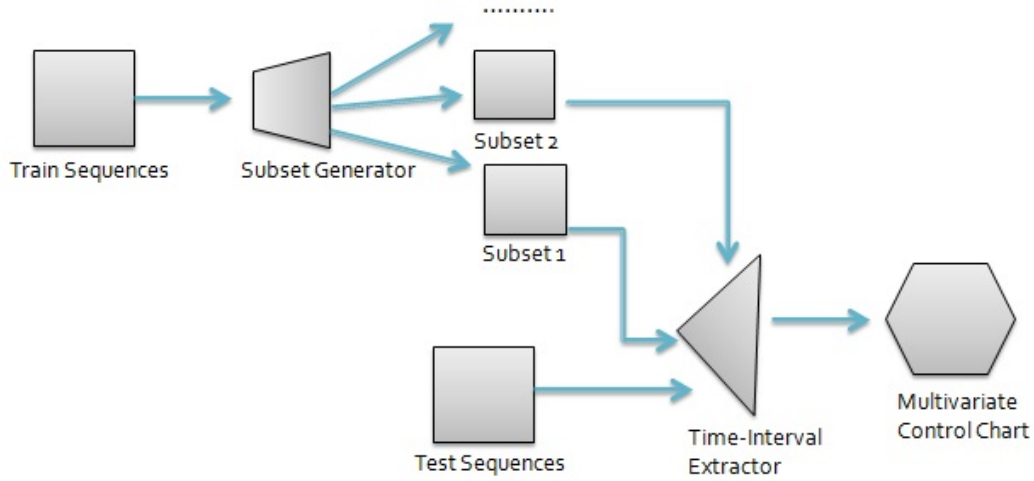
In some cases, there might be multiple occurrences of items  $p_q$  or  $p_{q'}$  in the sequence  $s$ . For example, say  $s_a = p_{q'}$  and  $s_b = p_q$  as well as  $s_c = p_q$  where  $a \leq b \leq c \leq L$ . In such cases, the time between occurrence of items  $p_{q'}$  and  $p_q$  with respect to sequence  $s$  would be

$$\frac{(t_b - t_a) + (t_c - t_a)}{2}$$

Similarly if  $p_{q'}$  occurs multiple times i.e.  $s_x = p_{q'}$ ,  $s_y = p_{q'}$  and  $s_z = p_q$  where  $x \leq y \leq z \leq L$ . The time between occurrence of items  $p_{q'}$  and  $p_q$  with respect to sequence  $s$  would be

$$\frac{(t_z - t_y) + (t_z - t_x)}{2}$$

Let the average time between occurrence of items  $p_{q'}$  and  $p_q$  of a pattern  $p$ , in a replicate  $D_0^n$ , be denoted as  $t_0^n(q', q)$ , which is calculated as follows. Let  $f(p_{q'}, p_q, s)$  denote the function which calculates the average time between occurrence of the items  $p_{q'}$  and  $p_q$  of



**Figure 48.** Modeling framework of the distribution-based method for change detection in sequential patterns. Subsets are extracted from the training data by sampling without replacement. Time-interval based statistics obtained from these subsets are compared with the test data to detect changes.

a pattern  $p$  in a sequence  $s$ . Let  $D_0^n(p)$  denote the set of all sequences of  $D_0^n$  in which the pattern  $p$  appears. Hence, the average time of occurrence  $t_0^n(q', q)$  is

$$\frac{\sum_{s \in D_0^n(p)} f(p_{q'}, p_q, s)}{|D_0^n(p)|}$$

where  $|D_0^n(p)|$  denotes the number of sequences in  $D_0^n(p)$ .

An overview of the modeling framework for the distribution-based method is shown in Fig. 47. The distribution-based approach for change detection in sequential patterns is as follows:

1. Frequent sequential patterns are extracted from the original data  $D_0$  using PrefixSpan algorithm [59].

2. Let  $D_0^n$  where  $n = 1, 2, \dots, N$ , be the  $n$ th replicate containing  $R \leq R_0$  sequences, such that each of the  $R$  sequences are obtained by sampling from the  $R_0$  sequences in  $D_0$  without replacement.
3. Consider a frequent sequential pattern  $p$  denoted by  $(p_1, p_2, \dots, p_Q)$  where  $p_q \in I$  for  $1 \leq q \leq Q$ . Let  $t_0^n(q-1, q)$ , where  $q = 2, 3, \dots, Q$ , denote the average time between occurrence of an item  $p_q$  after item  $p_{q-1}$  in pattern  $p$ , calculated using  $D_0^n(p)$ , for  $n = 1, 2, \dots, N$ . The values  $t_0^n(q-1, q)$  are calculated using the function  $f(p_{q-1}, p_q, s)$  as described previously.
4. Similarly let  $t_1^m(q-1, q)$ , where  $q = 2, 3, \dots, Q$ , be the average time between occurrence of an item  $p_q$  after item  $p_{q-1}$  in the pattern  $p$ , calculated using all the sequences from the data  $D_1^m$  where  $p$  occurs. The values  $t_1^m(q-1, q)$  are calculated for all the  $Q$  items of the pattern  $p$ . Multiple test sets  $D_1^m$ , can be obtained by varying  $m$ , where  $m = 1, 2, \dots, M$ .
5. Let  $T_0^n$  be a  $Q - 1$  dimensional vector representing the average times between the  $Q$  items of pattern  $p$  in a replicate  $D_0^n$  i.e.  $T_0^n = [t_0^n(2), t_0^n(3), \dots, t_0^n(Q)]$ . Similarly, let  $T_1^m$  be a  $Q - 1$  dimensional vector representing the average times between the  $Q$  items of pattern  $p$  in a replicate  $D_1^m$  i.e.  $T_1^m = [t_1^m(2), t_1^m(3), \dots, t_1^m(Q)]$ . The values of the vectors  $T_0^n$ , for  $n = 1, 2, \dots, N$  are compared with the corresponding vector  $T_1^m$  from the test data. A multivariate Hotelling  $T^2$  control chart is built using  $T_0^n$ , for  $n = 1, 2, \dots, N$  as the in-control data and calculating the control limits. The ordering among the  $T_0^n$  is not important while constructing the control chart as each of  $T_0^n$  is obtained from a replicate  $D_0^n$ , which are not ordered. The mean vector  $\bar{T}_0$  of

the in-control data is estimated as follows

$$\bar{T}_0 = \frac{\sum_{n=1}^N T_0^n}{N}$$

Similarly, the covariance matrix  $CV$  is estimated as follows

$$CV = \frac{\sum_{n=1}^N (T_0^n - \bar{T}_0)(T_0^n - \bar{T}_0)^t}{N - 1}$$

The vector  $T_1^m$  from new data  $D_1^m$  is plotted on the control chart to check if it falls within the control limits or is an out-of-control signal thus indicating a change.

## 2. DETECTING CHANGES IN TRAJECTORY PATTERNS

Our objective is detect changes in the frequent sequential patterns occurring in trajectories. Let  $TR$  be a trajectory denoted as a sequence of time ordered GPS points  $(x_t, y_t)$  for  $t \in \{1, 2, \dots, T\}$ . In order to represent trajectories as sequences, we discretize the trajectories, where the continuous GPS points  $(x_t, y_t)$  are represented by discrete clusters in space. The details of the discretization are discussed below. Even prior to discretization, the trajectory data needed to be compressed owing to its huge size.

The trajectory data was first compressed using the DP algorithm [3]. DP algorithm is a line-simplification algorithm. The input to the algorithm is a trajectory composed of sequence of points. The output of the algorithm is a similar simplified trajectory with only a subset of the original points. The details of the algorithm are discussed in Chapter 2.

The details of representing trajectories as a sequence of items is discussed here. Consider a dataset consisting of  $U$  trajectories. Let each trajectory  $TR_u$  in the dataset be represented as sequence of points  $(x_t^u, y_t^u)$  where  $t = 1, 2, \dots, T_u$  and  $u = 1, 2, \dots, U$ , where  $T_u$  denotes the number of points in the trajectory  $TR_u$ . Let  $v_t^u$  be a feature vector extracted

from the trajectory  $TR_u$  i.e.  $v_t^u = [x_t^u, y_t^u, x_{t+1}^u, y_{t+1}^u]$ , where  $t = 1, 2, \dots, T_u - 1$ . The vector  $v_t^u$  gives information of the path traversed by the trajectory  $TR_u$  between times  $t$  and  $t + 1$ . It says the trajectory starts from  $(x_t^u, y_t^u)$  at time  $t$  and ends at  $(x_{t+1}^u, y_{t+1}^u)$  at time  $t + 1$ . The order of points is thus important in the vector  $v_t^u$ . Hence, for a trajectory  $TR_u$  consisting of  $T_u$  points,  $T_u - 1$  feature vectors are extracted.

Let  $g(TR_u)$  be a function which outputs all the feature vectors of  $TR_u$  i.e.  $g(TR_u) = v_1^u, v_2^u, \dots, v_{T_u-1}^u$ . Let  $V$  denote the set of feature vectors obtained from all the  $U$  trajectories, which can be obtained by applying the function  $g(TR_u)$  for each  $u = 1, 2, \dots, U$ . The order of the feature vectors in  $V$  is not important. All the feature vectors belonging to  $V$  are clustered using BIRCH clustering [22]. The details of the BIRCH clustering algorithm are provided in Chapter 2. Given the various feature vectors belonging to the set  $V$  as inputs, BIRCH generates groups of the feature vectors which are close to each other. Since each feature vector is a representative of a line segment in a trajectory, the clustering phase indirectly groups the segments of the trajectories which have similar positions in space. A threshold  $\epsilon$  can be provided to the BIRCH algorithm which determines the closeness threshold between the feature vectors that are being clustered. If the value of  $\epsilon$  is small, the number of clusters obtained would be more compared to the number of clusters obtained if the  $\epsilon$  is large.

Let  $C : \{c_1, c_2, \dots, c_K\}$  denote the set of  $K$  cluster IDs obtained after clustering the feature vectors belonging to set  $V$  using BIRCH. Let the feature vector  $v_t^u$  of a particular trajectory  $TR_u$  belong to cluster  $c_k \in C$ . The trajectory  $TR_u$  is thus represented as the sequence of cluster IDs  $c_k$ , the corresponding feature vectors  $v_t^u$  belong to, where  $t = 1, 2, \dots, T_u - 1$ .



The original trajectory data is thus discretized by clustering the feature vectors extracted from the segments of the trajectories, and the original continuous values in the trajectories are replaced by the discrete values representing the cluster IDs the feature vectors belong to. Alternate methods of discretization are (a) grid based discretization, where the two or three dimensional space is divided into grids and (b) clustering the GPS points  $(x_t, y_t)$  of the trajectories directly to obtain clusters in space, and representing the trajectories as a sequences of clusters the corresponding points fall into [3]. An advantage of using our method over other discretization methods is that when data space is discretized into very fine granularities, two very similar trajectories may fall into different cells. Also, since the grid based discretization uses interval splits on each attribute i.e.  $x$  or  $y$  axis independently, the spatial correlation among the attributes may be lost. Also, our method of discretization is proven effective for sequential pattern mining of spatio-temporal data, since it generates partitions without losing the distributions of patterns in data [3].

The trajectory  $TR_u$  obtained after discretization, can therefore be represented as a sequence of cluster IDs the feature vectors  $v_t^u$  belong to. Once the trajectories are represented as sequences of items (cluster IDs), the frequency and distribution based methods, as discussed earlier, were used to detect changes in frequent patterns occurring in the discretized trajectories.

### 3. EXPERIMENTS

We have used real trajectory data from San Francisco cabs dataset which contains mobility traces of taxi cabs in San Francisco. The trajectory data consists of trajectories of 500 cabs over 30 days with a total of 10990 trajectories. These trajectories were first compressed

using the DP [3] algorithm, to reduce the size of the dataset to a total of 87,2600 GPS points. The compressed trajectory data is then discretized using BIRCH [22] clustering.

The trajectories are represented as a sequence of items as discussed in Section 2. The parameter  $\epsilon$  used in BIRCH clustering as discussed earlier can be varied to control the level of discretization. Experiments were conducted by choosing two different values for  $\epsilon$ : 0.1 and 0.0001, thereby leading to two levels of discretization: low and high. At low level of discretization, the number of items in the sequences are fewer. Hence the noise in the sequences is greater due to more number of repeating items in the sequences. This allows us to test the performance of our method to increased level of noise in sequences. At high level of discretization, the number of distinct items in the sequences are greater.

The discretized data from the cabs was divided into training and testing sets so as to have sufficient number of trajectory sequences in both of them. Frequent patterns were extracted from the training set. The testing set was used to test for changes in the frequent trajectory patterns extracted from the training set.

In Section 3.1, we discuss the experiments using the frequency-based approach for low-level of discretization. The experiments for frequency-based approach with high-level of discretization are discussed in Section 3.2. Section 3.3 discusses the experiments using distribution-based approach for high-level of discretization and Section 3.4 for distribution-based approach with low-level of discretization.

### **3.1. FREQUENCY-BASED APPROACH: LOW LEVEL OF DISCRETIZATION**

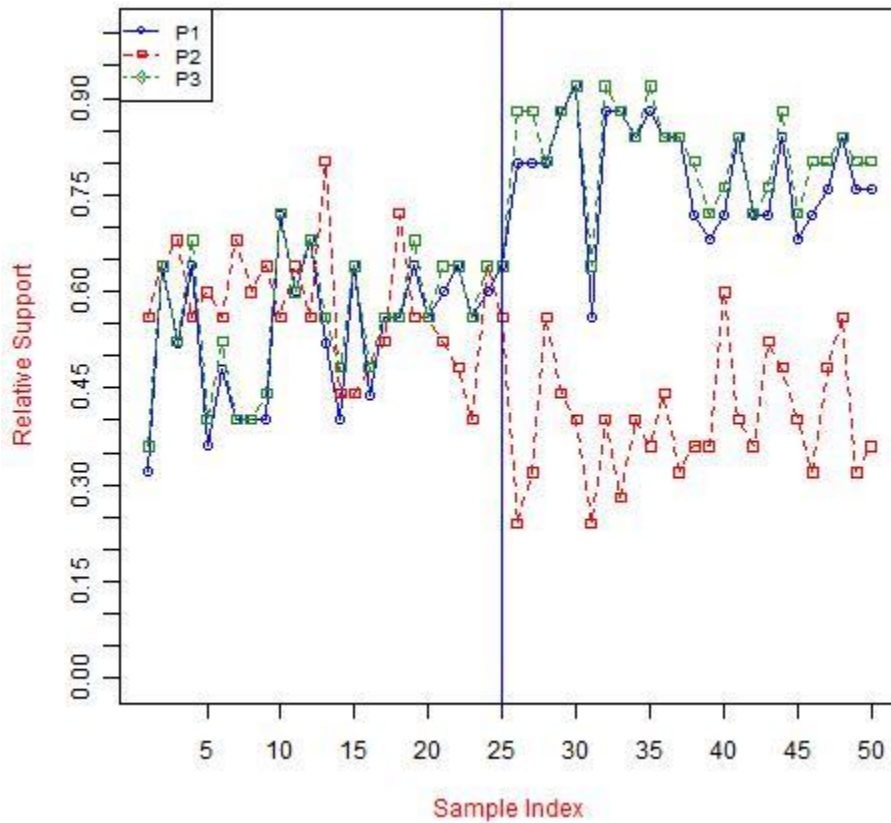
Thousand trajectories belonging to cabs with IDs 1 to 50 were used as training data  $D_0$ . The test set  $D_1$ , comprised of 3000 trajectories belonging to cabs with IDs 51 to 150. The discretization process as mentioned earlier was used with an  $\epsilon$  value of 0.1, to dis-

**TABLE 5.** Support count and % support of some patterns in training and testing data.

Pattern	Train Support Count	Train % Support (of 1000)	Test Support Count	Test % Support (of 3000)
(170-150-3)	511	0.511	2334	0.788
(6-150-144)	565	0.565	1186	0.395
(173-3)	533	0.533	2479	0.826

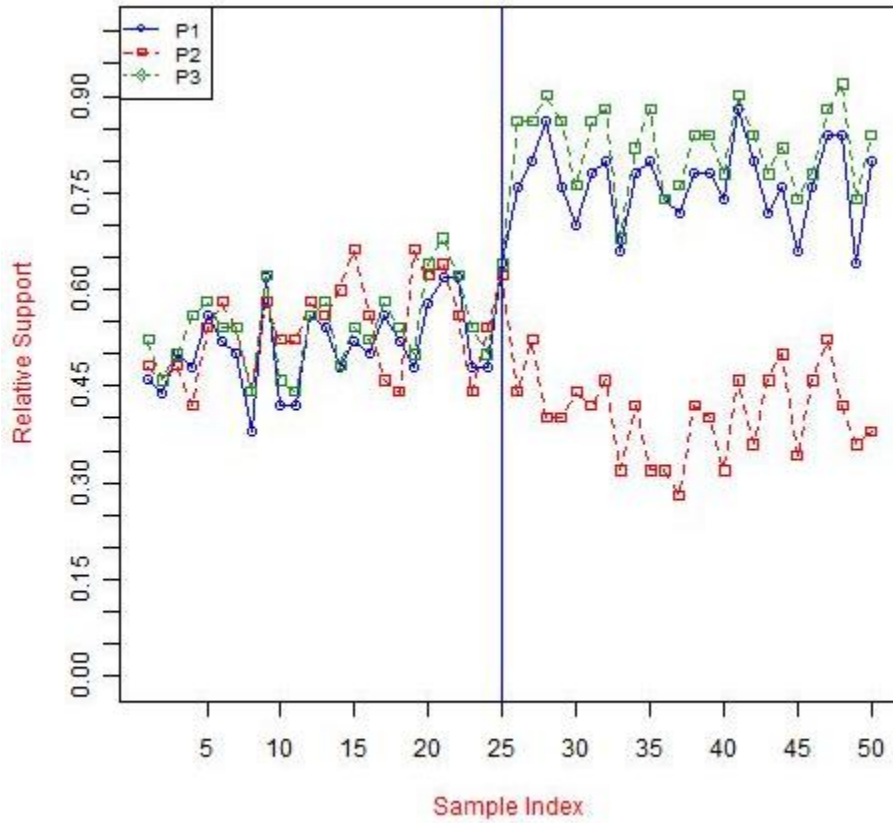
cretize the training and testing trajectories together. The discretization process led to 341 clusters for the data consisting of both training and testing trajectories. The number of clusters could have been further reduced by increasing the value of  $\epsilon$ , but this decreases the number of distinct items in the database. As a result the sequences have fewer number of distinct items. As an example, the trajectories would be discretized to the following sequence  $(a, a, a, a \dots)$  for higher values of  $\epsilon$ , leading to sequences with lesser distinct items. Prefixspan algorithm was used to extract frequent trajectory patterns from the training data  $D_0$  with a minimum support of 0.4. The minimum support is a parameter in the prefixspan algorithm which specifies a threshold for the support of the sequential patterns being extracted. The value of minimum support was varied in later experiments. The choice of the minimum support doesn't affect the results, but if the patterns chosen for analysis have very low support, there might not be enough sequences in the replicates to obtain an appropriate value of a statistic using the replicate. Also, we are interested in detecting changes in frequent sequential patterns occurring in the data whose support is significantly high.

Among all the frequent trajectory patterns extracted from the training set, a few patterns were chosen which showed sufficient change in support from training to testing set. These patterns were used for further analysis to illustrate our method. The three frequent trajectory patterns that were chosen from the training data are  $P1 : (170 - 150 - 3)$ ,  $P2 : (6 -$



**Figure 49a.** Relative support versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with  $R = 25$ . The change in relative support from the training to testing data illustrates the pattern change.

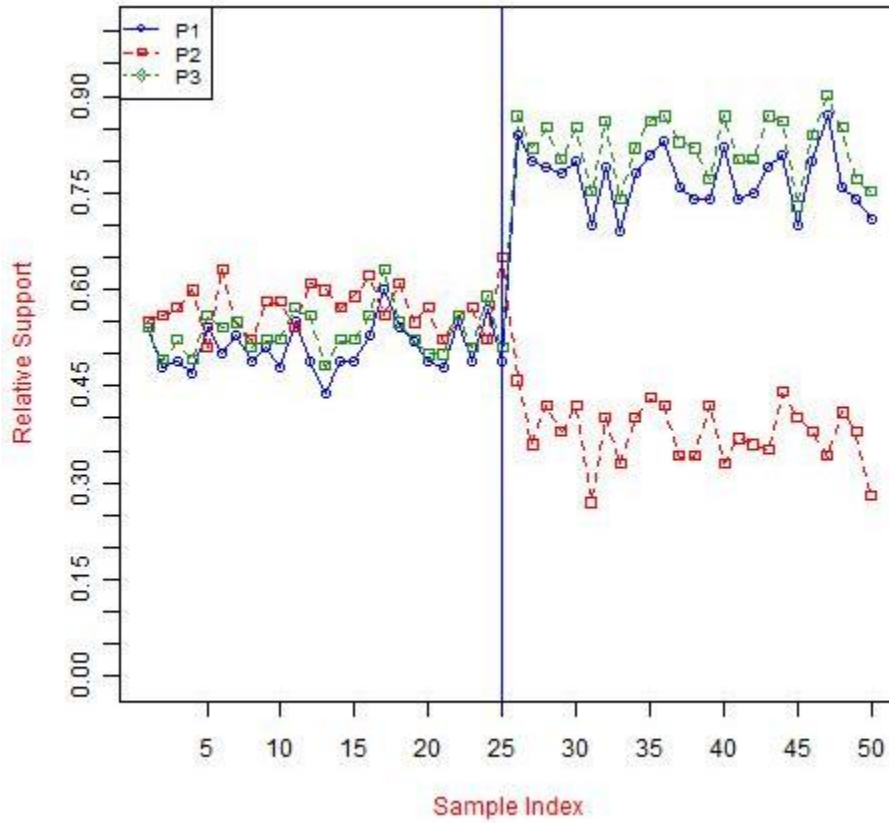
150 – 144),  $P3 : (173 - 3)$ . To understand what these patterns mean, consider pattern  $P1 : (170 - 150 - 3)$ , which implies that in more than atleast 40% of the trajectories out of the total trajectories, the segments designated by cluster IDs 170, 150 and 3, appear in that order. The support counts of these patterns in the training and testing data is shown in Table 5. From the table, it can be seen that the change in the support for the patterns from training data to testing data is relatively small (less than 0.3). For patterns P1 and P3, there is an increase in the support while for pattern P2, there is a decrease in the support.



**Figure 49b.** Relative support versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with  $R = 50$ . The change in relative support from the training to testing data illustrates the pattern change.

The first step in the algorithm is to sample the training data  $D_0$  containing  $R_0 (= 1000)$  trajectory sequences  $N$  times, such that each replicate consist of  $R$  sequences. Let these replicates be designated as  $D_0^n$  where  $n = 1, 2, \dots, N$ . The test data  $D_1$  consists of  $R_1 (= 3000)$  sequences is sampled  $M$  times to obtain replicates  $D_1^m$  where  $m = 1, 2, \dots, M$ , such that each replicate consists of  $R$  sequences.

3.1.1. *EXPERIMENTS WITH VARYING  $R$ .* These set of experiments were conducted to check how our method performs for variation in  $R$ . Experiments were conducted for different values of  $R = (25, 50, 100)$ . Three patterns  $P1$ ,  $P2$  and  $P3$  as described earlier

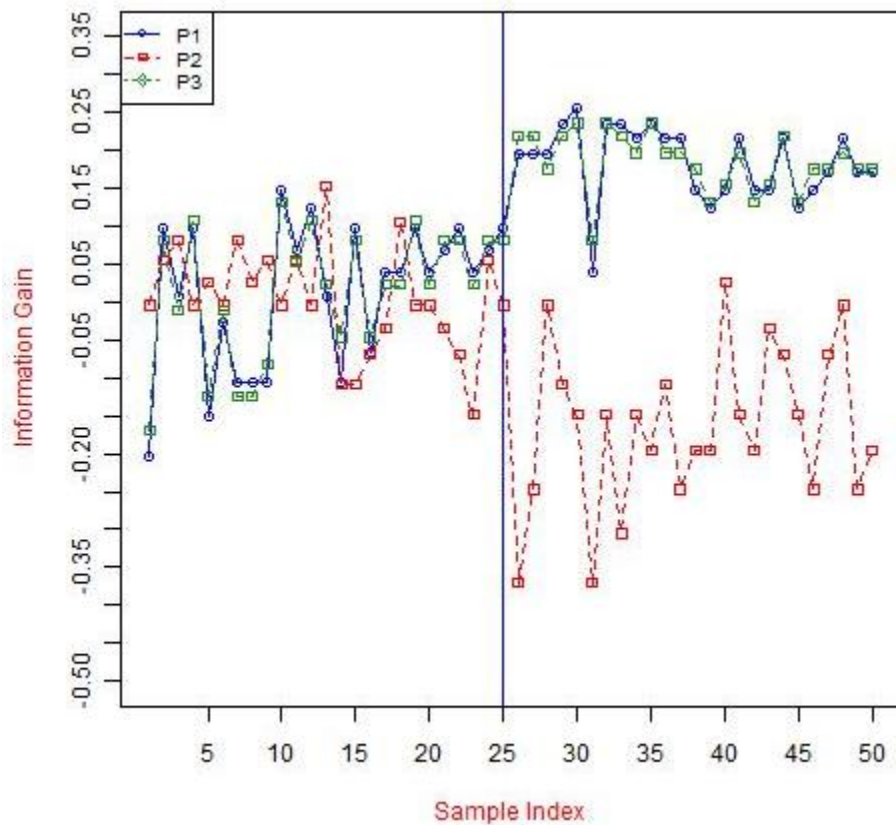


**Figure 49c.** Relative support versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with  $R = 100$ . The change in relative support from the training to testing data illustrates the pattern change.

were used to test the method. The number of training and testing replicates were fixed to be 25 in each experiment i.e  $N = M = 25$ . At each value of  $R$ , the variation of each of the statistics - relative support, information gain, growth rate - is discussed from the training to testing samples.

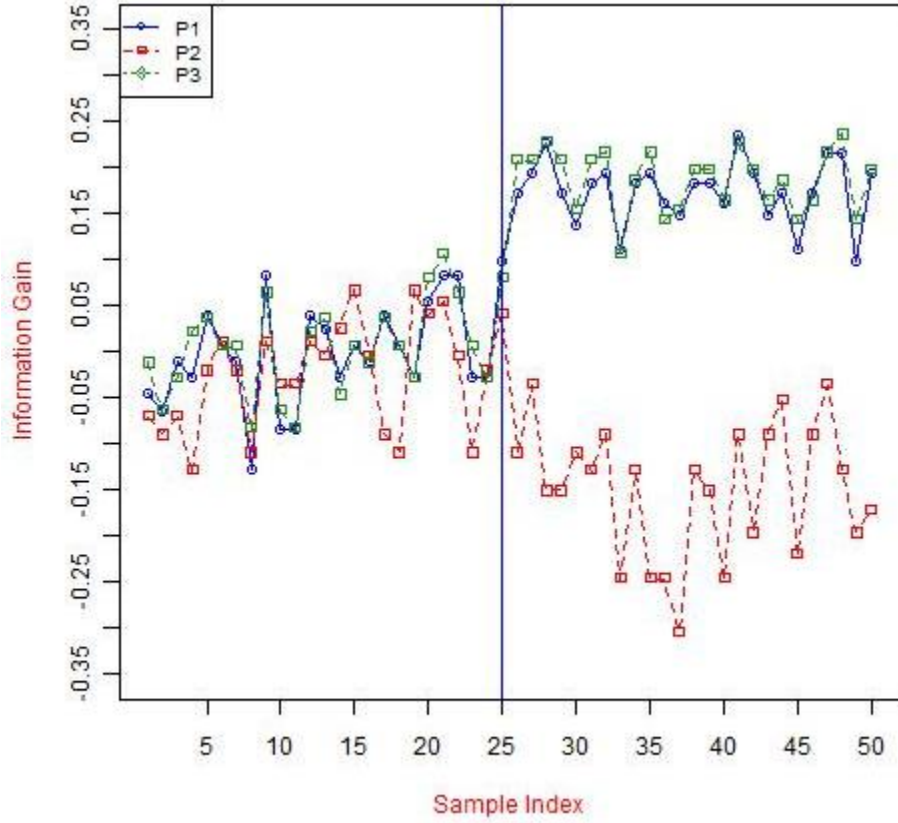
**Variation of relative support:** Relative support of a pattern  $p$ , which is defined earlier, in a replicate  $D_0^n$  is calculated as:

$$\frac{\text{support}(p, D_0^n)}{R_0}$$



**Figure 50a.** Information gain versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with  $R = 25$ . The change in information gain from the training to testing data illustrates the pattern change.

The relative support is plotted against the sample ID for the 25 training samples and 25 testing samples for different values of  $R$ . The variation of relative support from training to testing samples are shown in Fig. 49a, Fig. 49b and Fig. 49c for  $R = 25, 50, 100$  respectively across all the three patterns. It can be seen from the figures that the shift after the 25th sample is detectable for different values of  $R$ . However, the shift is more apparent for larger values of  $R$  i.e when  $R = 50, 100$  compared to when  $R = 25$ .



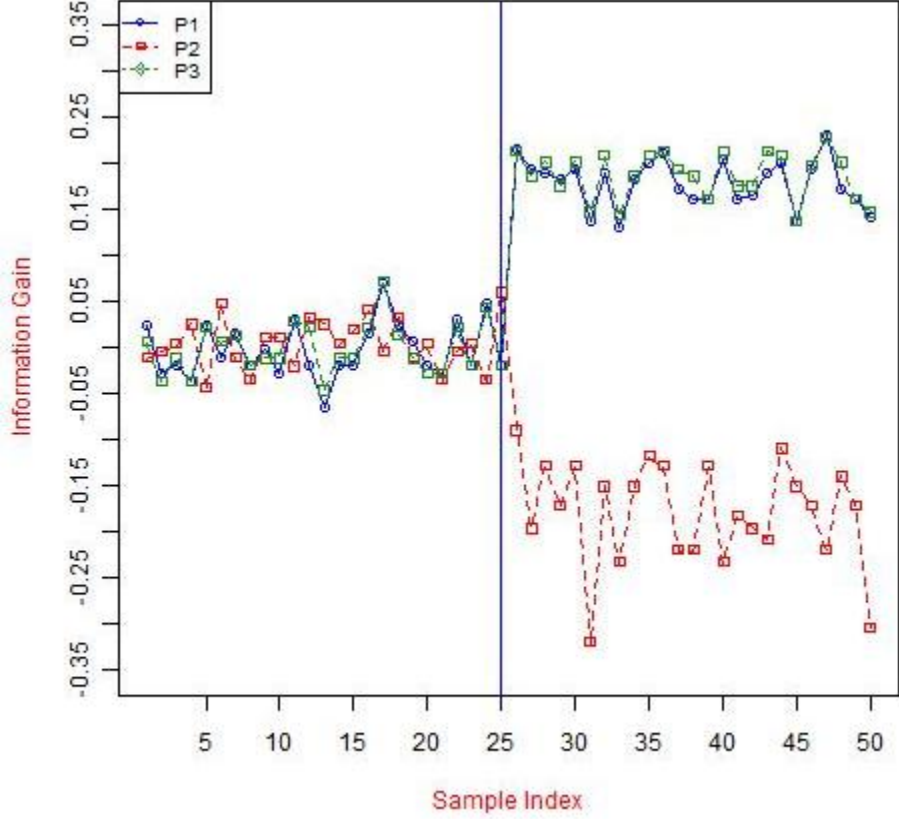
**Figure 50b.** Information gain versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with  $R = 50$ . The change in information gain from the training to testing data illustrates the pattern change.

**Variation of information gain:** Information Gain (IG) [90] of a pattern  $p$  w.r.t to a replicate  $D_0^n$ , which is defined earlier, is calculated as:

$$\log \frac{\text{support}(p, D_0^n) \times R_0}{\text{support}(p, D_0) \times R}$$

The variation of information gain from training to testing samples are shown in Fig. 50a, Fig. 50b, Fig. 50c respectively for  $R = 25, 50, 100$  across all the three patterns. The behavior of information gain is similar to the relative support with increasing  $R$  i.e the shift is more detectable with larger values of  $R$ .





**Figure 50c.** Information gain versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with  $R = 100$ . The change in information gain from the training to testing data illustrates the pattern change.

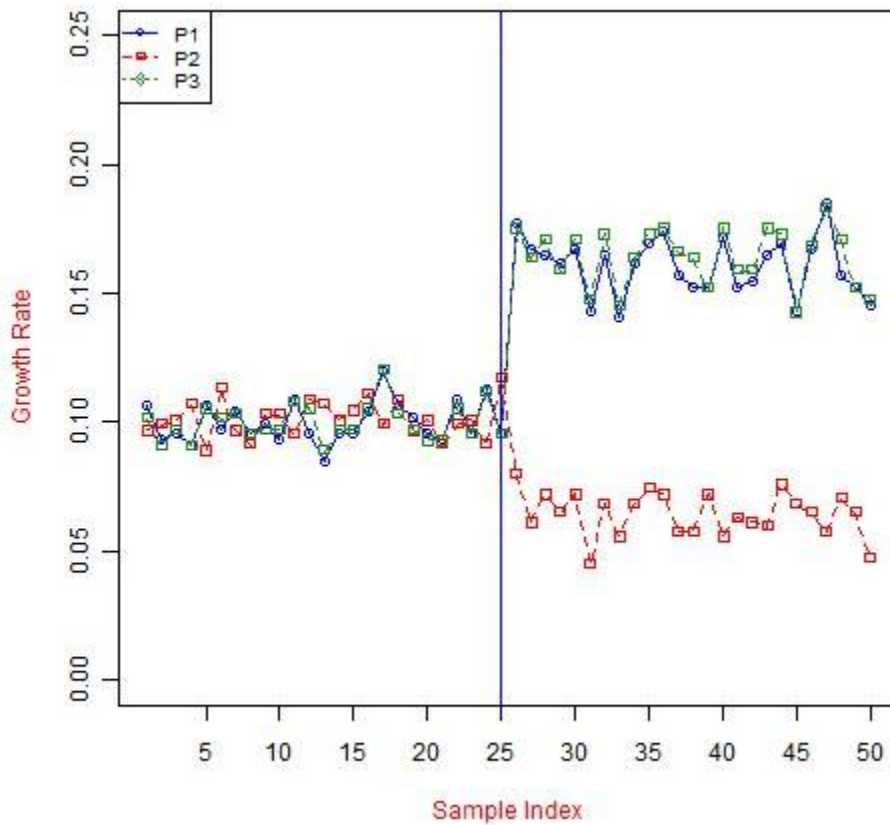
**Variation of growth rate:** The Growth Rate (GR) [90] of a pattern  $p$  w.r.t to a replicate  $D_0^n$  is

$$\frac{N_0 - R}{N_0} \times \frac{\text{support}(p, D_0^n)}{\text{support}(p, D_0) - \text{support}(p, D_0^n)}$$

The variation for growth rate from training to testing samples for  $R = 100$  is shown in Fig. 51. It can be seen from the figure that the shift is detectable after the 25th sample for all the three patterns.

### 3.1.2. EXPERIMENTS TO TEST THE SENSITIVITY BY OBTAINING RUN LENGTHS.

Several simulations were run to test the sensitivity of the method. Control charts were built



**Figure 51.** Growth rate versus sample index for training data (indices 1 to 25) and testing data (indices 26 to 50) with  $R = 100$ . The change in growth rate from the training to testing data illustrates the pattern change.

using the statistics obtained from the replicates  $D_0^n$  as the in-control or training data. The control limits set using the training data were used to obtain the run length with the statistics obtained from the replicates  $D_1^m$ . The run length for a specific control chart is the number of plotted points until a control chart signals i.e. an out-of-control signal occurs. Several run length measures were obtained by using different replicates of training and test data.

The  $3\sigma$  control limits were used to obtain the run lengths. The control limits are

$$UCL : \mu + 3\sigma$$

$$LCL : \mu - 3\sigma$$

where  $\mu$  is the mean and  $\sigma$  is the standard deviation of the training data. The Average Run Length (ARL) value is calculated by averaging the run lengths from the various control charts.

We conducted these experiments for the patterns  $P1$ ,  $P2$  and  $P3$ ; using different values of  $R = (25, 50, 100)$ ; and across the various measures i.e. relative support, information gain and growth rate. For each of the  $(pattern, R, measure)$  combination, hundred control charts were built using different replicates of training and testing data. Each control chart gave a value for the run length, which was used to estimate the ARL by averaging the hundred run lengths.

**NULL case ARL:** The average run length for NULL case, when there is no change in the pattern is calculated as follows. A particular pattern  $P2$  was chosen. 5000 Replicates  $D_0^n$ , where  $n = 1, 2, \dots, 5000$  were obtained from data  $D_0$  (consisting of 1000 sequences) by sampling without replacement, with each replicate consisting of 100 sequences i.e.  $R = 100$ . For each of the replicate  $D_0^n$ , statistical measures such as relative support, information gain and growth rate for the pattern  $P2$  were calculated relative to  $D_0$ . Control charts were plotted for each measure  $j$  and an average run length value calculated as described. For each measure  $j$ , the 5000 values of  $S_0^n(j)$  where  $n = 1, 2, \dots, 5000$  obtained using the replicates  $D_0^n$  are divided into training and test sets. The first 1000 values were divided into 10 training sets of 100 values. Similarly, the next 4000 values were divided into 10 test sets, where each test set consists of 400 values. A control chart was plotted for each combination

	2 $\sigma$ Limits	3 $\sigma$ Limits*
Relative Support	34.5 $\pm$ 4.98	243.2 $\pm$ 13.78
Information Gain	33.51 $\pm$ 3.82	246.55 $\pm$ 14.03
Growth Rate	34.97 $\pm$ 4.99	228.72 $\pm$ 14.29

**Figure 52.** Average run length and standard errors calculated for NULL case (no change in the support of the pattern) using pattern  $P2$ . Both the  $2\sigma$  and  $3\sigma$  limits are reported using the different measures. The  $3\sigma$  limits are calculated by truncating the run length to 400, if an out-of-control signal doesn't occur for 400 testing points.

of the training set and the test set, to obtain a total of 100 control charts. The run lengths obtained from these charts was averaged to obtain the ARL.

In order to get an estimate of the ARL value using the  $3\sigma$  limits for the NULL case, the run lengths for the individual charts were calculated as follows. Since in each control chart, 400 test points were used, if an out-of-control signal occurs within the 400 test points, the run length for that particular chart would be the number of the test point at which the signal occurs. On the other hand, if an out-of-control signal doesnot occur within the 400 test points, the run length for that chart is assumed to be 400. The ARLs (as well as standard errors) with  $3\sigma$  control limits calculated for the pattern  $P2$  across the three measures is shown in Fig. 52. This value under estimates the true ARL calculated using the  $3\sigma$  control limits. Fig. 52 also has the ARLs (and standard errors) with  $2\sigma$  control limits. The large values for ARL using the  $3\sigma$  limits suggest that there is no change in the statistics of the patterns between the training and testing data.

**ARL for non-null case:** Statistics obtained from the replicates  $D_1^m$  were used as the testing data. The ARL and standard errors were calculated using control charts with statistics  $S_0^n(j)$  from replicates  $D_0^n$  as the training data and statistics  $S_1^m(j)$  from replicates  $D_1^m$  as the testing data. The number of training ( $N$ ) and testing ( $M$ ) points to be used in each

R	Measure	Pattern1	Pattern2	Pattern3
25	Relative Support	2.81 ( $\pm 0.25$ )	7.59 ( $\pm 0.723$ )	1.63 ( $\pm 0.1021$ )
	Information Gain	33.84 ( $\pm 4.466$ )	4.08 ( $\pm 0.43$ )	32.29 ( $\pm 7.608$ )
	Growth Rate	2.62 ( $\pm 0.238$ )	9.69 ( $\pm 0.976$ )	1.63 ( $\pm 0.102$ )
50	Relative Support	1 ( $\pm 0.001$ )	3.97 ( $\pm 0.311$ )	1 ( $\pm 0.001$ )
	Information Gain	1.48 ( $\pm 0.095$ )	2.4 ( $\pm 0.208$ )	1.1 ( $\pm 0.03$ )
	Growth Rate	1 ( $\pm 0.001$ )	3.97 ( $\pm 0.311$ )	1 ( $\pm 0.001$ )
100	Relative Support	1.03 ( $\pm 0.017$ )	1.53 ( $\pm 0.134$ )	1 ( $\pm 0.001$ )
	Information Gain	1.06 ( $\pm 0.024$ )	1.32 ( $\pm 0.094$ )	1 ( $\pm 0.001$ )
	Growth Rate	1 ( $\pm 0.001$ )	1.56 ( $\pm 0.139$ )	1 ( $\pm 0.001$ )

**Figure 53.** Average run length values for different (*pattern, R, measure*) combinations. All the run lengths are calculated using  $3\sigma$  limits.

replicate of the control chart for a specific value of  $R$ , were chosen so as to obtain a run length value for that chart using  $3\sigma$  limits. The ARL value is calculated by averaging the run length values for 100 control charts. The number of training and testing points in each replicate of the control chart for a specific value of  $R$  are shown in Table 6. For  $R = 25$ , in order to obtain the run lengths using information gain measure for patterns  $P1$  and  $P3$ , we had to use a large number of testing points (500) in each replicate of the control chart as shown in Table 6.

Experiments were conducted for different combinations of (*pattern, R, measure*). For each (*pattern, R, measure*), the ARL is calculated by averaging the run lengths from 100 control charts. The standard errors were calculated by first calculating the standard deviation of the hundred run length values and then dividing it by square root of sample size i.e.

$$StdError = \frac{\sqrt{\sum_{i=1}^{100} (RL_i - \bar{RL})^2}}{10}$$

**TABLE 6.** Number of training and testing points used to build each replicate of the control chart using low level of discretization.

R	N	M
25	75	500
50	50	50
100	50	50

where  $RL_i$  is the run length obtained from the  $i$ th control chart and  $\bar{RL}$  is the average value of the 100 run lengths. The ARL values for each combination of  $(pattern, R, measure)$  are shown in Fig. 53.

- It can be seen from Fig. 53, that as the value of  $R$  increases from 25 to 100, the average run length value decreases. Hence, our method performs better for larger values of  $R$ .
- The ARL value for a particular value of  $R$ , using information gain measure is considerably lower than that obtained using relative support or growth rate for pattern  $P2$ . This means, for patterns with a negative change (i.e. decrease in support) from the train set to test set (e.g. pattern  $P2$ ), information gain measure is a better measure to detect the changes quickly.
- The ARL value for a particular value of  $R$ , using information gain measure is considerably higher than that obtained using relative support or growth rate for patterns  $P1$  and  $P3$ . This means, for patterns with a positive change (i.e. increase in support) from the train set to test set (e.g. pattern  $P1, P3$ ), relative support or growth rate are better compared to information gain to detect early changes.

**TABLE 7.** Support count and % support of some patterns in training and testing data.

Pattern	Train Support Count	Train % Support (of 3363)	Test Support Count	Test % Support (of 5703)
(1130-1123-1138)	1695	0.504	2132	0.374
(85-1130-1123)	1669	0.496	2271	0.398
(1130-1138)	2143	0.637	2811	0.493
(1123-1130-1123)	1728	0.514	2236	0.392

**TABLE 8.** Number of training and testing points used to build each replicate of the control chart with high level of discretization.

R	N	M
25	100	250
50	100	100
100	100	100

### 3.2. FREQUENCY-BASED APPROACH: HIGH LEVEL OF DISCRETIZATION

The trajectory data consists of trajectories of 500 cabs over 30 days with a total of 10990 trajectories were discretized using a BIRCH parameter  $\epsilon$  of 0.0001 to obtain 1538 clusters. Larger training and test sets were used to check how our method scales to larger data sets. The discretized data from cabs 1 to 150 consisting of  $R_0 = 3363$  trajectories was used as training data  $D_0$ . Prefixspan algorithm was used to extract frequent trajectory patterns from the training data  $D_0$  with a minimum support of 0.4. The test data  $D_1$ , comprised of  $R_1 = 5703$  trajectories from cabs 152 to 414.

The patterns chosen for analysis were  $P1 : (1130 - 1123 - 1138)$ ,  $P2 : (85 - 1130 - 1123)$ ,  $P3 : (1130 - 1138)$  and  $P4 : (1123 - 1130 - 1123)$ . The support count of these patterns in the training  $D_0$  and testing data  $D_1$  are shown in Table 7.

The ARL and standard errors were calculated using control charts with statistics  $S_0^n(j)$  from replicates  $D_0^n$  as the training data and statistics  $S_1^m(j)$  from replicates  $D_1^m$  as the

R	Measure	Pattern1	Pattern2	Pattern3	Pattern4
25	Relative Support	22.85 ( $\pm 2.499$ )	40.17 ( $\pm 4.052$ )	14.65 ( $\pm 1.903$ )	35.66 ( $\pm 5.055$ )
	Information Gain	6.51 ( $\pm 0.645$ )	12.46 ( $\pm 1.361$ )	5.31 ( $\pm 0.41$ )	8.4 ( $\pm 0.927$ )
	Growth Rate	22.85 ( $\pm 2.499$ )	45.34 ( $\pm 4.25$ )	14.65 ( $\pm 1.903$ )	37.08 ( $\pm 5.028$ )
50	Relative Support	7.74 ( $\pm 0.823$ )	21.1 ( $\pm 2.703$ )	2.33 ( $\pm 0.22$ )	8.23 ( $\pm 0.983$ )
	Information Gain	2.15 ( $\pm 0.184$ )	8.99 ( $\pm 1.016$ )	1.88 ( $\pm 0.142$ )	3.76 ( $\pm 0.495$ )
	Growth Rate	7.74 ( $\pm 0.823$ )	21.1 ( $\pm 2.703$ )	2.33 ( $\pm 0.22$ )	8.23 ( $\pm 0.983$ )
100	Relative Support	2.28 ( $\pm 0.14$ )	4.85 ( $\pm 0.353$ )	1.36 ( $\pm 0.052$ )	2.95 ( $\pm 0.21$ )
	Information Gain	2 ( $\pm 0.1$ )	2.76 ( $\pm 0.17$ )	1.28 ( $\pm 0.045$ )	2.27 ( $\pm 0.104$ )
	Growth Rate	2.35 ( $\pm 0.15$ )	5.2 ( $\pm 0.37$ )	1.36 ( $\pm 0.052$ )	2.99 ( $\pm 0.206$ )

**Figure 54.** Average run length values for different (*pattern*, *R*, *measure*) combinations and high level of discretization. All the run lengths are calculated using  $3\sigma$  limits.

testing data. The number of training ( $N$ ) and testing ( $M$ ) points to be used in each control chart for a specific value of  $R$ , were chosen so as to obtain a run length value for that chart using  $3\sigma$  limits. The number of training ( $N$ ) and testing ( $M$ ) points for different values of  $R$  are shown in Table 8. The ARL value is calculated by averaging the run length values for 100 control charts. The average run length values and the standard errors for the four patterns are shown in Fig. 54.

Similar conclusions can be drawn for high-level of discretization as was the case with low-level of discretization. As it can be seen Fig. 54, information gain measure performs better compared to other measures for all the patterns, when the support decreases for all the patterns. Also, the change is better detectable with larger values of  $R$ .

### 3.3. DISTRIBUTION BASED APPROACH: HIGH LEVEL OF DISCRETIZATION

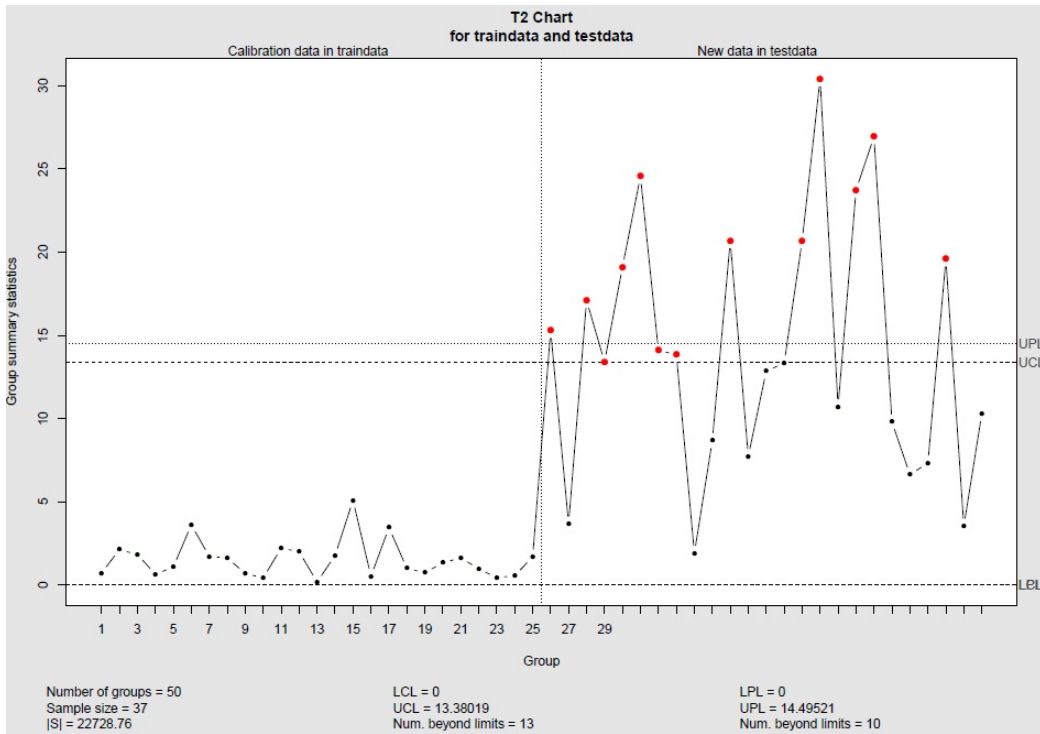
One thousand discretized trajectories belonging to cabs with IDs 1 to 50 were used as training data  $D_0$ . The discretization process as mentioned earlier was used with an  $\epsilon$  value of 0.0001. Prefixspan [59] algorithm was used to extract frequent trajectory patterns from the training data  $D_0$  with a minimum support of 0.3.



In order to induce change in the distribution of the times in the trajectory sequences, synthetic data was created by changing the sampling frequency of the items in the sequences of  $D_0$ . Let the items in the sequences from the training data  $D_0$  be sampled at regular intervals of time  $t_0$ . New data sets were created using the same 1000 sequences from  $D_0$  but by changing the interval of sampling times for items in the sequences of the original data  $D_0$ . For example, consider a sequence  $(a, b, c, d)$  in the original data  $D_0$  with time intervals  $(t_{ab}, t_{bc}, t_{cd})$  between the items  $a, b, c, d$  respectively. In order to create a testing data set  $D_1$ , the times between the items in the sequence  $(a, b, c, d)$  were changed to  $(\delta \times t_{ab}, \delta \times t_{bc}, \delta \times t_{cd})$  respectively. Let the test data be  $D_1$ , with the sampling time between items in the sequences being  $t_1$ , where  $t_1 = \delta t_0$ ,  $\delta$  denoting the amount of change in the sampling frequency between items in the sequences in the test data  $D_1$  compared to the original data  $D_0$ . The purpose of the synthetic data creation is to check if our method is able to detect the change in the time interval distribution of the patterns which continue to be frequent in the test data.

In the distribution-based method we are mainly interested to detect if the distribution of times between items in a pattern have changed. The number of statistics monitored in this method depend on the number of items in the pattern as described earlier. Hence we have chosen patterns with varying length i.e. patterns with 3 items and 4 items. The two frequent trajectory patterns that were chosen from the training data are  $P1$ : (85-1130-1123) and  $P2$ : (523-1445-1442-1123) with supports of 0.48 and 0.39 respectively.

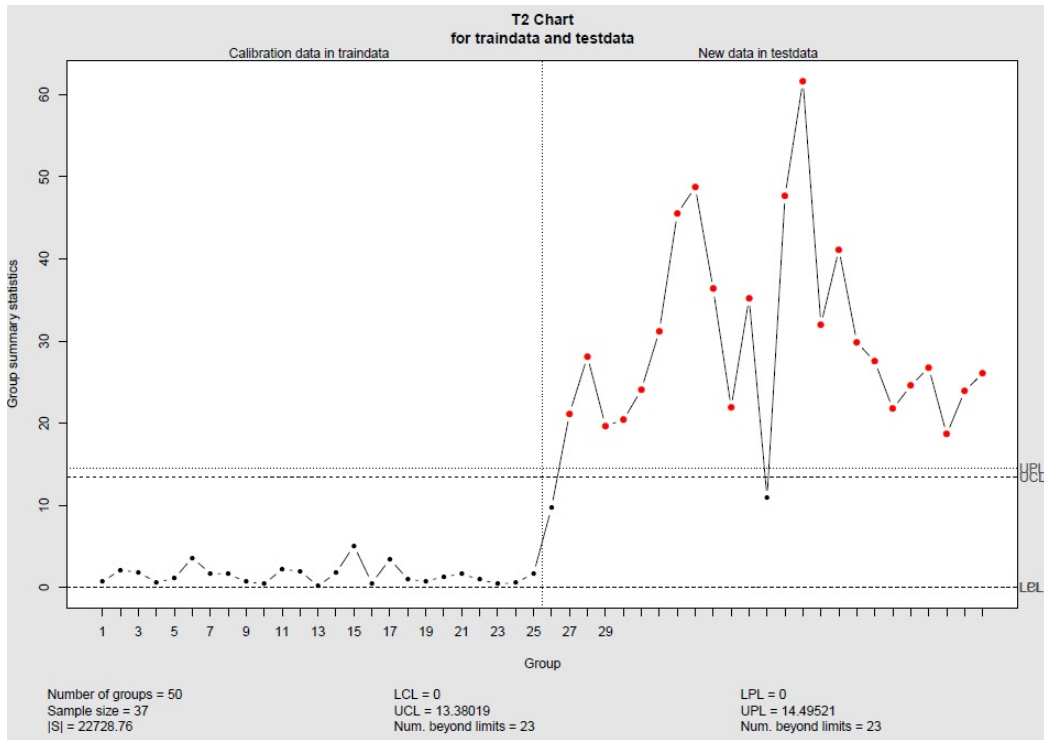
3.3.1. *EXPERIMENTS WITH VARYING  $\delta$* . Experiments were conducted for different values of  $\delta = 1.2, 1.3, 1.5$ , to check the sensitivity of our method to different levels of change in the time interval distribution between items in the patterns. Also, the value of  $R$



**Figure 55a.** Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P1$  using  $\delta = 1.2$  and  $R = 100$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change.

was varied i.e.  $R = 50, 100$ . Different combinations of  $N$  and  $M$  were used for various control charts to obtain an out-of-control signal. The results for varying values of  $\delta$  and  $R$  are discussed below.

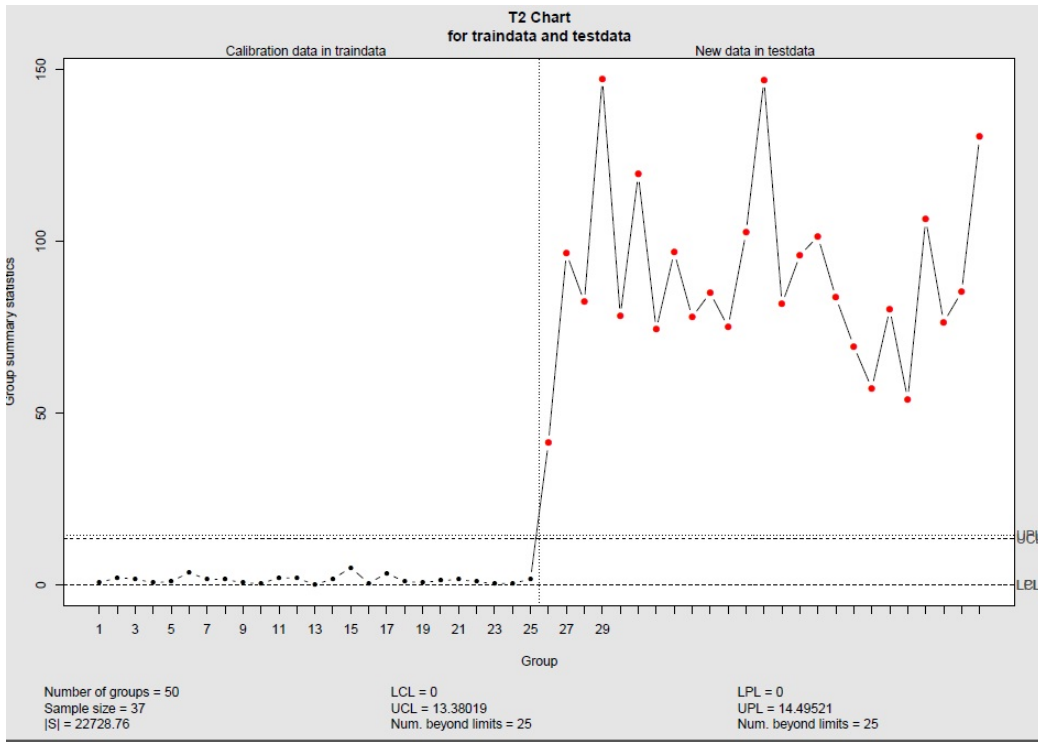
We have used a Hotelling  $T^2$  control chart to detect changes in the distribution of the time between occurrence of various items in the pattern. We have used a confidence level of 0.999 to determine the control limits for all the charts. In a Hotelling  $T^2$  chart, we can have more than one response variable as it is a multivariate control chart. Consider the pattern  $P1$ : (85-1130-1123), since there are three items in the pattern, we have two variables to be monitored. Similarly for the four item pattern  $P2$ , three variables need to be monitored.



**Figure 55b.** Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P1$  using  $\delta = 1.3$  and  $R = 100$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change.

We used Hotelling  $T^2$  control charts with subgroup data as well as individual observations for the purpose of change detection. First set of experiments were conducted by plotting Hotelling  $T^2$  control charts with subgroup data. Experiments with individual observations are discussed later.

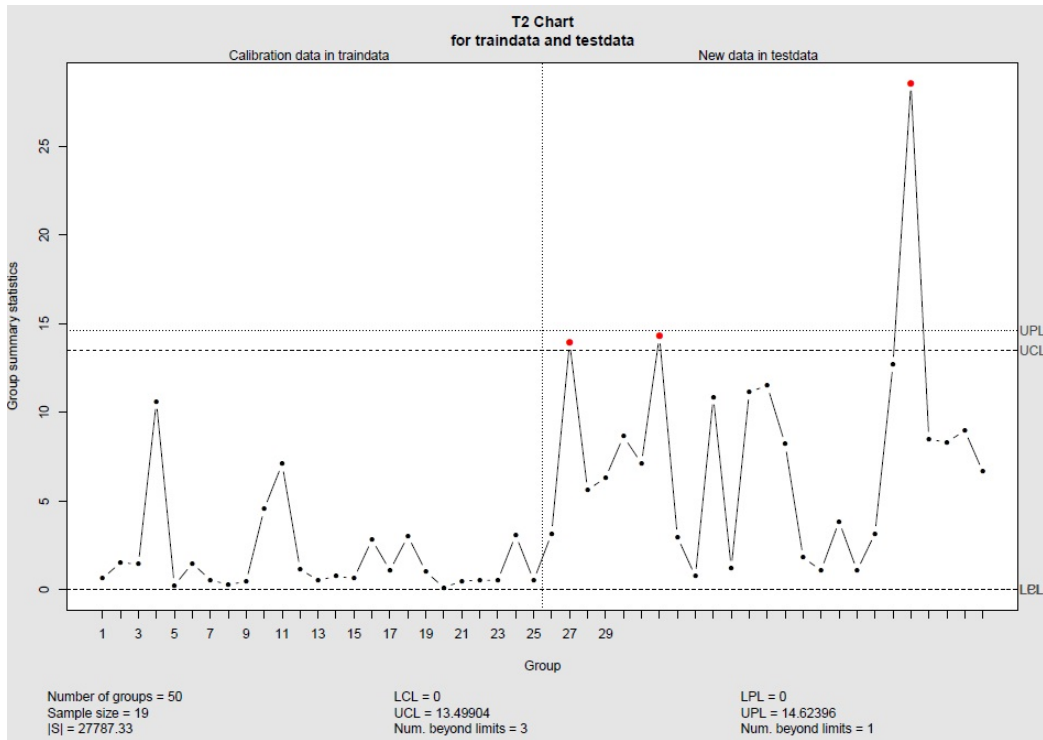
The subgroup size  $g$  for plotting the control chart is chosen as follows. Each sequence in  $D_0^n$ , where a pattern  $p$  occurs, is considered to be a subgroup. The number of subgroups in a replicate  $D_0^n$  therefore depend on the support count of the pattern  $p$  in the replicate. The subgroup size  $g$  is therefore chosen to be the minimum support count of the pattern  $p$  among all the replicates  $D_0^n$  where  $n = 1, 2, \dots, N$ . The average time interval values from



**Figure 55c.** Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P1$  using  $\delta = 1.5$  and  $R = 100$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change.

the first  $g$  sequences where the pattern  $p$  occurs in  $D_0^n$ , are considered to be belong to that subgroup. For a replicate  $D_0^n$ , the average times between occurrence of items in the pattern  $p$  are calculated for each sequence in  $D_0^n$ , which contains the pattern  $p$ .

**Hotelling  $T^2$  chart with sub-group data:** For the pattern  $P1$ , the Hotelling  $T^2$  chart with sub-group data is shown in Fig. 55a, which is built using  $N = 25$  training points and  $M = 25$  test points. The value of  $\delta$  is 1.2 i.e. sampling times between the items in the trajectory sequences are increased by 20% in the testing data and the value of  $R = 100$ .

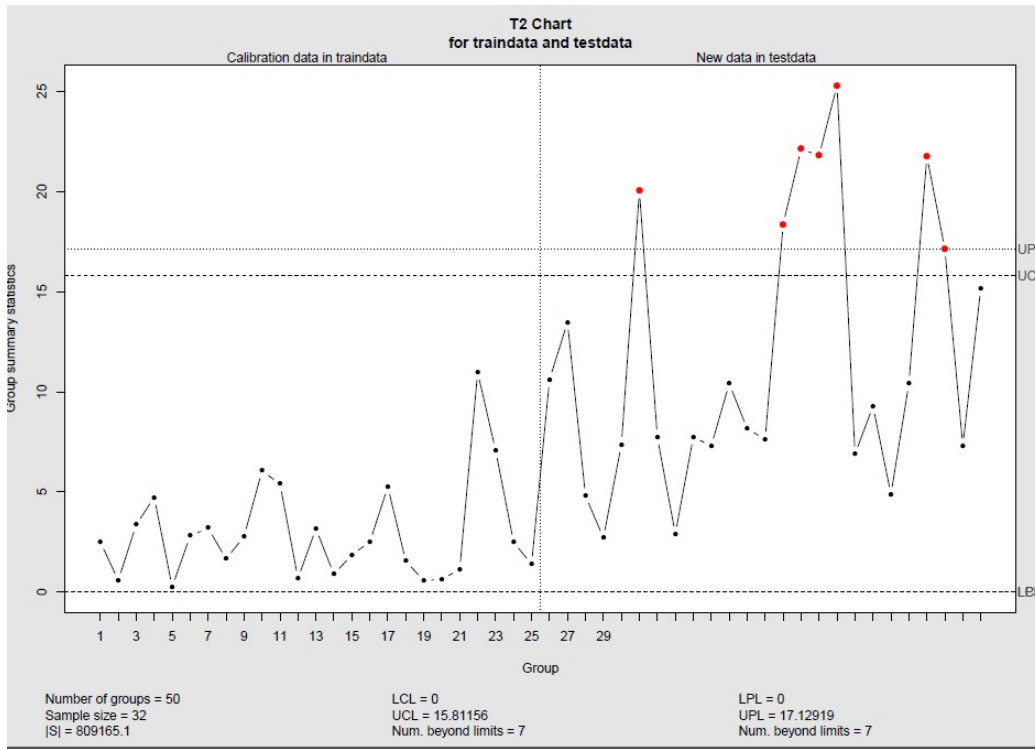


**Figure 56.** Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P1$  using  $\delta = 1.2$  and  $R = 50$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change.

For the pattern  $P1$ , the Hotelling  $T^2$  chart with sub-group data using the same parameter values of  $R = 100$ ,  $N = 25$  and  $M = 25$  but a different value of  $\delta = 1.3$  is shown in Fig. 55b.

The Hotelling  $T^2$  chart with sub-group data using the same parameter values of  $R = 100$ ,  $N = 25$  and  $M = 25$  but  $\delta = 1.5$  for pattern  $P1$  is shown in Fig. 55c

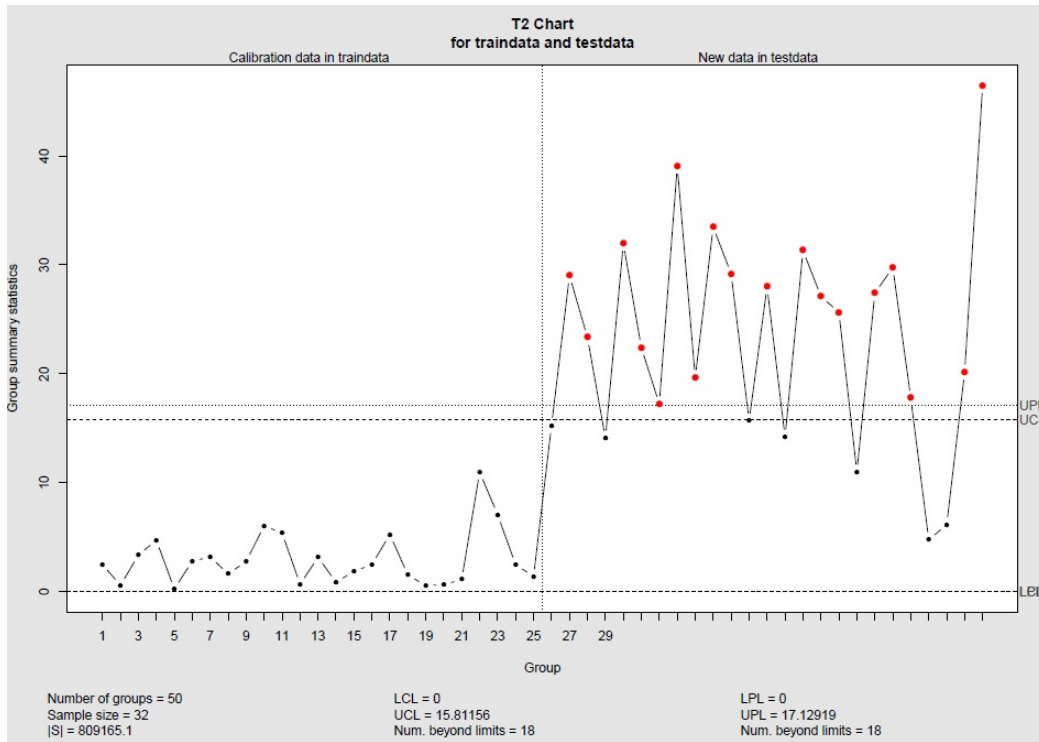
It can be seen from the figures that our method is able to detect the changes in the distribution for all values of  $\delta$  in the test data. The more the change in the time interval distribution i.e. the higher the value of  $\delta$ , the better the detection capacity.



**Figure 57a.** Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P2$  using  $\delta = 1.2$  and  $R = 100$ . The change in the statistic value from the train data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change.

The value of  $R$  was decreased from 100 to 50 to check the sensitivity of our method to lower sampling sizes. The control chart for  $N = 25$  training points and  $M = 25$  testing points using  $R = 50$  and  $\delta = 1.2$  is shown in Fig. 56. Comparing it with Fig. 55a, it can be seen that larger the value of  $R$ , the better the detection capacity.

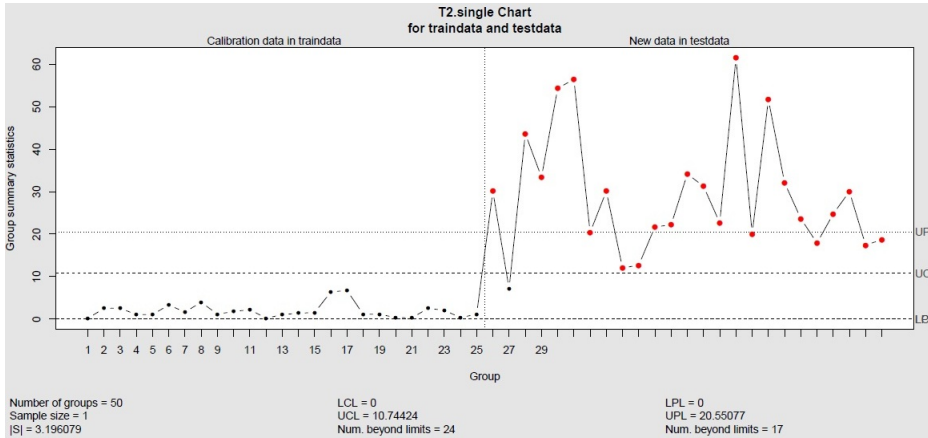
**Hotelling  $T^2$  chart with single observations:** Another set of experiments were conducted by plotting the  $T^2$  control chart using single observation from  $D_0^n$ . In order to plot these control charts, an average value for the time interval  $t_0^n(q-1, q)$  between items in a pattern  $p$  obtained from  $D_0^n$  was used to derive the statistic to be plotted on the control chart.



**Figure 57b.** Hotelling  $T^2$  chart of group statistic versus sample index for pattern  $P2$  using  $\delta = 1.3$  and  $R = 100$ . The change in the statistic value from the train data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change.

This average value  $t_0^n(q - 1, q)$  is calculated using the function  $f(p_{q-1}, p_q, s)$  as described earlier.

The Hotelling  $T^2$  control chart plotted using single observations is shown in Fig. 58. This chart is plotted for pattern  $P1$  using a 20% increase in the the time interval distribution between the items i.e.  $\delta = 1.2$ . The value of  $R = 100$  and  $N = 25$  training,  $M = 25$  testing points were used. It can be seen from the figure that, our method was able to detect the changes using single observations as well.



**Figure 58.** Hotelling  $T^2$  chart of single observation versus sample index for pattern  $P1$  using  $\delta = 1.2$  and  $R = 100$ . The change in the statistic value from the training data (indices 1 to 25) to the test data (indices 26 to 50) indicates the time interval distribution change.

### 3.4. DISTRIBUTION BASED APPROACH: LOW LEVEL OF DISCRETIZATION

We used 1000 discretized trajectories belonging to cabs with IDs 1 to 50 as training data  $D_0$ . The discretization process as mentioned earlier was used with an  $\epsilon$  value of 0.1. Prefixspan [59] algorithm was used to extract frequent trajectory patterns from the training data  $D_0$  with a minimum support of 0.3.

3.4.1. *PATTERNS WITH NON-REPEATING ITEMS.* The pattern chosen for analysis was  $P1 : (170 - 150 - 3)$ . Time-interval statistics obtained from the replicates  $D_0^n$  were used as the training data and statistics from the replicates  $D_1^m$  were used as the test data for a hotelling  $T^2$  control chart. Hundred control charts were built using different training and testing points for each control chart. The ARL is calculated by averaging the hundred run length values. The values of  $N$  (number of training points) was fixed to 50 for each control chart. The values of  $M$  were varied for each control chart, until we obtained an out-of-control signal. The ARL values were obtained from 100 replicates of the Hotelling



R	Control Chart	% change	M	Average Run Length
50	Single Observation	35%	50	4.16 ± 0.6344
		20%	75	14.4 ± 1.232
	Subgroup Data	35%	75	12.81 ± 1.415
		20%	200	34.37 ± 3.45
100	Single Observation	20%	60	10.36 ± 0.647
	Subgroup Data	20%	65	18.9 ± 1.23

**Figure 59.** The ARL values obtained for pattern  $P1$  using the distribution based method for different combinations of ( $R$ , Control chart method, % Change in time between items in the pattern).

$T^2$  control chart for each combination of ( $R$ , Control chart method, % Change in time distribution).

The results are shown in Fig. 59 for different combinations of ( $R$ , Control chart method, % Change in time distribution). The column  $M$  in the figure, is the number of testing points in each replicate of the control chart to obtain an out-of-control signal. The ARL values shown in the Fig. 59, are obtained by averaging the 100 run length values obtained from the 100 replicates of the control chart. The standard errors, as shown in Fig. 59, were calculated as described earlier.

From the experiments and the run length values in Fig. 59, the following conclusions can be made.

- The patterns with higher % change are easily detectable compared with the ones with lower % change. For example, the ARL value using  $R = 50$  and single observation method for pattern with 20 % change is 14.4, while that for a pattern with 30 % change is only 4.16.

<b>%Change</b>	<b>Pattern PR1</b>	<b>Pattern PR2</b>
35%	1.04 ± 0.0197	1.13 ± 0.0338
20%	4.91 ± 0.3062	3.39 ± 0.438
-26%	5.41 ± 0.651	2.64 ± 0.365

**Figure 60.** The ARL values obtained for pattern *PR1* and *PR2* with repeating items, using the distribution based method for different levels of % change in time between items in the patterns.

- As the value of  $R$  increases, the same % change in the distribution becomes easy to detect. For example, the ARL value for a 20 % change using single observation method is 14.4 when  $R = 50$ , while it is 10.36 when  $R = 100$ .
- Control charts using single observations perform better compared to control charts using subgroup data. For example, the ARL value for a 20 % change using  $R = 50$  is 34.37 using subgroup data, while it is only 14.4 using single observations.

3.4.2. *PATTERNS WITH REPEATING ITEMS.* Further experiments were conducted, to test our method on patterns with repeating items. If the item that is repeating in a pattern, is also a repeating item in a sequence, the noise further increases while calculating the average time between the items of a pattern in a sequence. To illustrate this, consider the sequence (3-4-4-3-3), and the two patterns (3-3) and (3-4) that are contained in it. The pattern (3-3) occurs 3 items in the sequence, but the pattern (3-4) only occurs 2 items.

The following two patterns with repeating items and varying lengths were chosen, *PR1*: (144-3-144) and *PR2*: (144-144-3-3).

Experiments were also conducted by decreasing the time interval distribution between items in the test data to test the efficiency of the distribution-based method to detect such changes. The  $T^2$  control chart with single observations was used to detect the changes.

The results of the experiments are shown in Fig. 60. In each control chart,  $N = 50$  training and  $M = 50$  testing points were used, an run length value calculated. The ARL was calculated by averaging the run lengths from 100 control charts. A confidence level of 0.999 was chosen and the upper control limit was 12.22993. A value of  $R = 50$  was used for analysis.

## CLUSTERING AND OUTLIER DETECTION IN SEMANTICALLY ENRICHED TRAJECTORIES

Research in movement data is booming and is focusing on providing rich information service tailored for the application at hand. Most of these services build upon some semantic interpretation of movement data. Raw data stream (collected by GPS) is first turned into a set of trajectories, which are then enriched with semantic data from the application world. Trajectory semantics can be inferred from spatio-temporal properties of the raw data stream (e.g. *when and where the object stops or moves, its track orientation or movement pattern*), from the geographical information related to the region traversed by the trajectory (e.g. *its road network*), as well as from application objects stored in the application databases and related to the trajectory (e.g. *the list of customers visited by a company salespersons*).

Semantic annotation refers to the additional data attached to the spatio-temporal positions in the trajectory. Examples of annotation include recording the observed “*activity*” of a moving animal (with activity values “*feeding*”, “*resting*”, “*moving*” etc.), computing and recording the instant speed of the moving object, inferring and recording the “*means of transport*” used by a moving person (e.g. “*by foot*”, “*bus*”, “*metro*”, “*bicycle*” etc.).

Some of the examples of semantically enriched trajectories involve trajectories of trucks or other vehicles transporting goods in a supply chain. The attributes describing the flow of product in a vehicle include time at a stop (e.g. 60 mins), activity at a stop (e.g. Loading), speed of travel for a move (e.g. 25 mph), mode of transport used for the move (e.g. Truck) etc. Another example of a semantically enriched trajectory is the web browsing pattern of a person, where the various websites visited by the person can be thought of the stops. The attributes that describe the trajectory of web browsing pattern of a person who is trying to purchase goods from website like Ebay, include activity of the person on the webpage (e.g.

Logging in), product purchased (e.g. Shoe), time between clicks from one page to another, price of the product purchased (e.g. \$50) etc.

Semantic enrichment of trajectories benefits many applications compared to raw trajectory obtained from GPS feeds. A few examples include for daily trip of employees, it would be important to know the transportation means used during the trip and also if the trip used carpool facilities. Similarly, analyzing the migratory patterns of birds, it would be important to have information about the weather conditions the bird faces during the flight and also where and how long the bird stopped during the journey. The flow of goods in a supply chain can be modeled as a semantic trajectory i.e. a sequence of stops and moves. The trajectory of objects (like vehicles, goods etc.) in a supply chain has a lot of semantic information associated with it. Hence apart from representing these objects as a point moving in space, there is a need for semantically enriched trajectories. Detecting outliers during the flow of goods in a supply chain using trucks enables us to detect accidents or breakdown occurring during transportation. The information to semantically enrich the flow of goods in a supply chain involves weather conditions like wind direction, speed of wind, sky condition (rainy, cloudy, foggy) while transporting goods using ships. Also shipment involving trucks would require information about the natural objects (e.g. higher elevation due to mountains) or artificial objects (e.g highways, bridges over water bodies) that the truck would encounter in its trajectory.

Consider a case where we are analyzing people trajectories, rather than using the raw GPS data, it would be more informative to view a trajectory as the following semantic encoded sequence of triples (let us call it as reference trajectory):

(home, -10am, )

- (road, 10am-11am, on car)
- (office, 11am-5pm, work)
- (road, 5pm-5:30pm, on bus)
- (market, 5:30pm-6pm, shopping)
- (road, 6pm-6:20pm, on foot)
- (home, 6:20pm-, ) [55]

It can be observed that the first and last triples respectively denote the first (Begin) and last (End) spatio-temporal positions delimiting the trajectory. The spatial coordinates  $(x_t, y_t)$  do not appear in the triples as they are encoded with semantic information like “home”, “office”, “road”, “park”, giving detailed information about the location. The second element denotes the time period when the other two elements (location and annotation) remain constant e.g. consider the second triple which says that person was on road traveling in a car from 10 am to 11 am. The third element in the triple provides additional semantic annotation, related to the *activity* performed (work, shopping) or the *mode of transportation* used (on car, on bus, on foot). Such information can be extracted based on statistics like amount of time spent at a location like market, speed of motion while traveling on road e.g. higher speeds might imply traveling by bus or car, while lower speed would imply that the person is walking. Such semantic representation of trajectories would enable a better understanding of the semantic behavior. Analyzing such semantically enriched trajectories would further help to discovery knowledge like semantic similarity, clustering semantically enriched trajectories to find user similarity, semantic pattern mining, mobility analysis/statistics.

Analyzing semantically enriched trajectories is a challenging area of research due to the various factors to be considered. The trajectory data may not be continuous e.g. when a person enters an indoor location the GPS signal would be lost. The algorithms used for trajectory analysis should not have any application specific data encoded in them e.g. movement of cars is constrained by road network, while walking is unconstrained and follows unplanned paths. Also, the algorithms should be able to handle variations in data quality. The framework also needs to filter out the data which is not required for annotation and should be able to select the most relevant kind of annotation data for the various segments of the trajectory e.g. while annotating a walking person it doesn't make sense to annotate all the restaurants which he quickly passes by, unless he does an activity like dining at one of these restaurants and spends considerable amount of time at one place.

**TABLE 9.** Notation table

Notation	Description
$N$	Number of semantic trajectories
$T$	Semantic trajectory represented as a finite sequence $\langle I_1, I_2, \dots, I_K \rangle$ where $I_k$ for $k = 1, 2, \dots, K$ is either a stop or move
$T_n$	$n$ th semantic trajectory out of the $N$ trajectories represented as finite sequence of stops or moves i.e. $T_n := \langle I_1^n, I_2^n, \dots, I_{K_n}^n \rangle$
$K_n$	Number of stops and moves in a trajectory $T_n$

**TABLE 9.** Notation table

Notation	Description
$I_k^n$	$k$ th element of the trajectory $T_n$ such that $I_k^n \in \{S_1, S_2 \dots S_P, M_1, M_2, \dots M_Q\} \forall k \in \{1, 2, \dots K_n\}$
$P$	Number of stops that can be used to represent all the $N$ trajectories
$S$	Set of $P$ stops $S : \{S_1, S_2, \dots S_P\}$
$Q$	Number of moves that can be used to represent all the $N$ trajectories
$M$	Set of $Q$ moves $M : \{M_1, M_2, \dots M_Q\}$
$A$	Number of attributes describing each stop of a trajectory
$B$	Number of attributes describing each move of a trajectory
$s_{ij}$	Variable to denote the $j$ th attribute of the $i$ th stop
$m_{uv}$	Variable to denote the $v$ th attribute of the $u$ th move
$T_n'$	Trajectory $T_n$ after transformation using variables corresponding to the stops and moves i.e. $s_{ij}$ and $m_{uv}$
$C_0$	Data consisting of $N$ records from trajectories $T_n'$ where $n = 1, 2 \dots N$ belonging to class 0
$C_1$	Synthetic data consisting of $N$ records created using $C_0$ and belonging to class 1



**TABLE 9.** Notation table

Notation	Description
$C_T$	Data consisting of $N$ records from $C_0$ and $N$ records from $C_1$
$r$	Parameter used for creating synthetic data $C_1$
$RF$	Random forest consists of $L$ trees constructed using the data $C_T$ i.e. $RF : \{rf_1, rf_2, \dots, rf_L\}$
$z_n$	The terminal node positions in the $L$ trees for $T'_n$ i.e. $z_n = (z_{n1}, z_{n2} \dots z_{nL})$
$S(T'_{n_1}, T'_{n_2})$	Similarity between pair of trajectories $T'_{n_1}$ and $T'_{n_2}$
$PR$	Proximity matrix obtained from the random forest $RF$ where the $ij$ th element is $S(T'_i, T'_j)$
$CF$	Conditional inference forest (cForest) consisting of $L$ trees constructed using the data $C_T$ i.e. $CF : \{cf_1, cf_2, \dots, cf_L\}$
$PC$	Proximity matrix obtained from the cForest $CF$
$\bar{PC}(n)$	Average proximity from trajectory $T'_n$ to the rest of the trajectories calculated using the proximity matrix $PC$
$O_n$	Outlier score for trajectory $T'_n$

**TABLE 9.** Notation table

Notation	Description
$\delta$	<p>A parameter used to simulate trajectories belonging to a cluster.</p> <p>The values of the numeric variables for trajectories belonging to the cluster, deviate from the mean value of the cluster by an amount <math>\delta</math></p>

Considering the semantic information attached to the spatio-temporal path of a moving object would reveal useful application dependant knowledge. Consider the trajectory data of tourists visiting various locations during a trip. The kind of semantic knowledge from these trajectories would allow us to answer questions like *which are the places most frequently visited by tourists in the morning?* A pattern of the form  $[Hotel\ Emperor] (s= 90\%)$  would imply that 90% of the trajectories have Hotel Emperor in them. Similarly, a pattern  $[TouristPlace, Hotel](s = 80\%)$  would mean that 80% of the trajectories that have a stop at a tourist place also have a stop at a hotel.

The trajectories have two facets: a geometrical facet and a semantic facet. The geometrical facet is representation as a continuous function of time in a geographical space i.e.  $(x_t, y_t)$  where  $t = 1, 2, \dots T$ . The semantic facet includes two types of the semantic characteristics of the trajectories, the first type is specific to requirements of the application

(like arrival time, weight of a product etc.) and the second includes standard semantics and uses the components of the trajectory like stop, move, begin and end of the trajectory. The notation table used in the chapter is shown in Table 9.

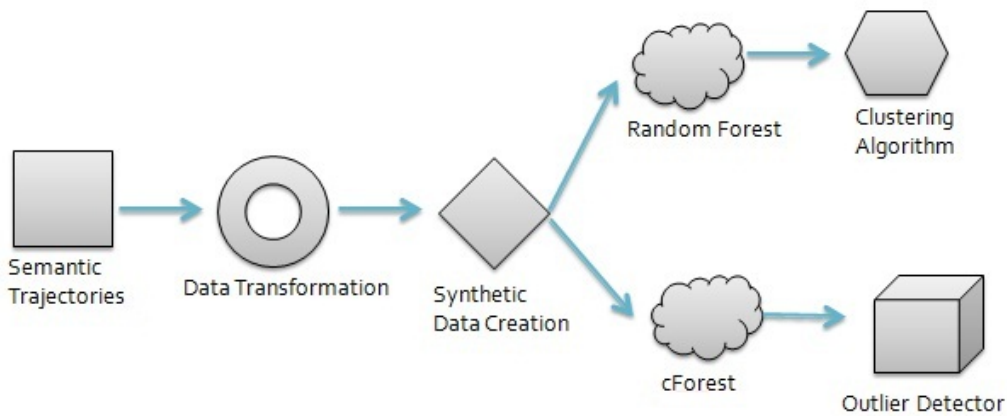
## 1. SEMANTICALLY ENRICHED TRAJECTORIES

A semantic trajectory  $T$  is a finite sequence  $\langle I_1, I_2, \dots, I_K \rangle$  where  $I_k$  for  $k = 1, 2, \dots, K$ , is either a stop or a move. A semantically enriched trajectory is a semantic trajectory where each of the stops and moves have attributes describing them. Some examples of semantic trajectories are, trajectories of trucks transporting goods in a supply chain, trajectories of web browsing pattern of a person etc. which can be modeled as a sequence of stops and moves.

We propose a model for clustering and outlier detection in semantically enriched trajectories. Clustering aims to group trajectories that have the same movement behavior. Outlier detection can be used to detect trajectories that donot follow the common pattern as most of the trajectories. The details of the method used for clustering semantically enriched trajectories are in Section 2. The method used for outlier detection in semantically enriched trajectories is explained in Section 3. The experiments are shown in 4, with specific experiments for clustering in 4.1 and those for outlier detection in Section 4.2. An overview of the method used for clustering and outlier detection in sematic trajectories is shown in Fig. 61.

## 2. CLUSTERING SEMANTICALLY ENRICHED TRAJECTORIES

Clustering semantic trajectories aims to group similar trajectories having the same movement behavior. It is an unsupervised learning technique, and hence deals with finding structure in a collection of unlabeled data. A cluster is a collection of trajectories which

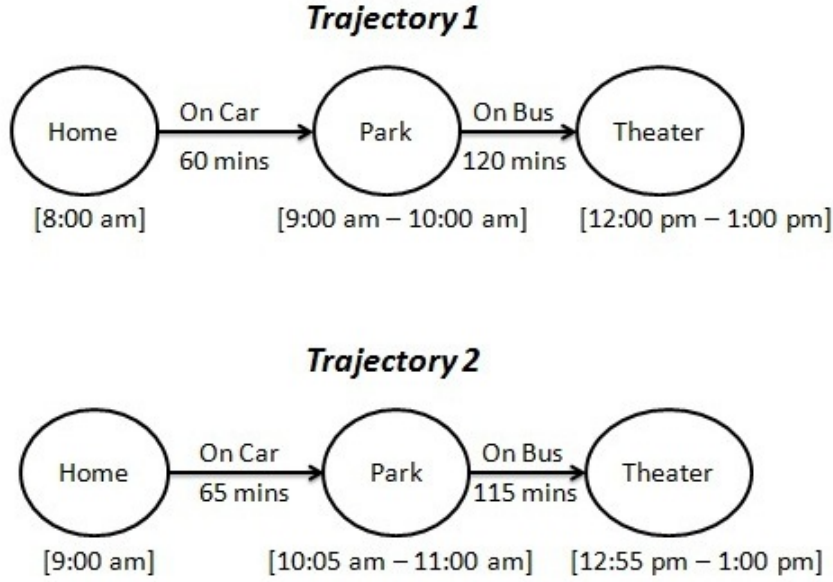


**Figure 61.** Modeling framework for clustering and outlier detection in semantic trajectories.

are more “similar” to each other and are “dissimilar” to the trajectories belonging to other clusters. Clustering therefore requires a similarity measure.

The example in Fig. 62 shows the semantic trajectories of two people, which are considered to be similar. The sequence of the stops and moves in the two trajectories have to be same i.e.  $[home \rightarrow park \rightarrow theater]$ . It can be seen that in trajectory 1 the person starts from home at 8:00 am, while in trajectory 2 he starts from home at 9:00 am. Hence, the starting time of the two trajectories need not be same but the time taken for travel, mode of transport, time at each of the stops and the sequence of stops and moves have to be similar for the trajectories to be considered similar. Such similarity can be used in recommendation systems to recommend friends based on similar movement behavior.

Each trajectory, which is a sequence of stops and moves, is transformed using a set of variables to capture the important characteristics of the trajectory. To cluster the tra-



**Figure 62.** Two similar semantic trajectories belonging to a cluster.

jectory data, which is now represented using a set of variables, we start by converting the unsupervised learning problem to a supervised learning problem.

Consider a set of  $N$  semantic trajectories where each trajectory is denoted as  $T_n$  where  $n = 1, 2, \dots, N$ . Let the path taken by all these  $N$  trajectories be represented by a set of  $P$  stops and  $Q$  moves. Let these  $P$  stops be denoted as  $S = \{S_1, S_2, \dots, S_P\}$  and the  $Q$  moves be denoted as  $M = \{M_1, M_2, \dots, M_Q\}$ . Each trajectory can be represented using a subset of the  $P$  stops and  $Q$  moves. Each of the trajectory  $T_n$  can be represented as sequence of the stops and moves visited in its path i.e. each semantic trajectory  $T_n$  is a finite sequence  $\langle I_1^n, I_2^n, \dots, I_{K_n}^n \rangle$  where  $I_k^n$  is either a stop or move i.e.  $I_k^n \in \{S_1, S_2, \dots, S_P, M_1, M_2, \dots, M_Q\}$  for  $k \in \{1, 2, \dots, K_n\}$ ,  $n \in \{1, 2, \dots, N\}$ . There are algorithms in literature, SMoT [13], CBSMoT [14] to decompose a set of trajectories into a sequence of stops and moves which are discussed in Chapter 2.

TrajID	$s_{11}$ (Time)	$s_{12}$ (Activity)	$m_{11}$ (Time)	$m_{12}$ (Speed)	$s_{21}$	$s_{22}$
1	500 (secs)	Loading	1700 (secs)	60 (mph)	200 (secs)	
2	800 (secs)	Quality Check	4000 (secs)	70 (mph)	500 (secs)	.
3	.	.	.	.	.	.
4	.	.	.	.	.	.

**Figure 63.** Representation of a semantic trajectory using the variables  $s_{ij}$  and  $m_{uv}$  corresponding to the stops and moves of the trajectory.

At each stop of a trajectory, let  $A$  attributes be recorded, e.g. time at the stop, activity at the stop etc. Similarly, for each move of a trajectory let  $B$  attributes be recorded e.g. speed of travel, mode of transport etc. The number of attributes recorded at a stop or a move can differ depending on the stop or move, but we explain using constant number of attributes for simplicity. Let  $s_{ij}$  for  $i \in \{1, 2, \dots, P\}, j \in \{1, 2, \dots, A\}$  denote the  $j$ th attribute of the  $i$ th stop. Similarly, let  $m_{uv}$  for  $u \in \{1, 2, \dots, Q\}, v \in \{1, 2, \dots, B\}$  denote the  $v$ th attribute of the  $u$ th move. Each trajectory  $T_n$  is represented by the variables  $s_{ij}$  and  $m_{uv}$  that correspond to the stops and moves in the path of the trajectory  $T_n$ . As an example, suppose the path of a particular trajectory consists of the stops  $S_1, S_2, S_3, S_4, S_5, S_6$  and moves  $M_1, M_2, M_3, M_4, M_5$  in a specific order. The variables which have a non-null value for this trajectory are  $s_{ij}$  where  $i = 1, 2, \dots, 6$  and  $j = 1, 2, \dots, A$  and  $m_{uv}$  where  $u = 1, 2, \dots, 5$  and  $v = 1, 2, \dots, B$ , while the rest of the variables have a null value.

An example of this transformation for the trajectories of trucks transporting goods in a supply chain is shown in Fig. 63. Consider the trajectory of a vehicle which starts from  $S1$  and moves to  $S2$ . Let the vehicle stop at stop  $S1$  for 12 mins to load goods and then move to stop  $S2$  with the time for the move  $M1$  being 70 min while traveling at a speed of 60 mph. The first row in Fig. 63, shows this trajectory being represented using variables

Traj ID	$s_{11}$	$s_{12}$	$m_{11}$	$s_{21}$	$s_{22}$	.	Class
1	500	Loading	70	Quality Check	80	.	0
2	90	Unloading	100	Loading	75	.	0
3	850	Loading	83	Distribution		.	0
.	.	.	.	.	.	.	.
100	.	.	.	.	.	.	.
101	98	Loading	85	Quality Check	.	.	1
102	812	Unloading	78	Distribution	.	.	1
103	534	Loading	97	Distribution	.	.	1
.	.	.	.	.	.	.	.
200	.	.	.	.	.	.	.

Random value between  
 $90 - \sigma$  and  $850 + \sigma$

Synthetic Class

**Figure 64.** Transforming the trajectory data to a two class problem. Data  $C_0$  belonging to class 0 consists of 100 records and data  $C_1$  belonging to class 1 consists of 100 records. The values for the variables in data  $C_1$  depend on the corresponding variables in  $C_0$ .

$s_{11}, s_{12}, m_{11}, m_{12}$  etc. which correspond to the stop  $S1$  and move  $M1$ , respectively. This kind of representation of the trajectories can capture the important characteristics like time, speed etc. of a trajectory, which are stored as attributes of the respective stops or moves. The representation can also capture the direction of motion, e.g., a move from stops  $S1$  to  $S2$  would be different from a move from stops  $S2$  to  $S1$ , as new variables for the moves would be created. The special case of circles in trajectories i.e. trajectories starting from a particular stop and returning to the same stop, was not considered was analysis.

The trajectories are thus represented in terms of the variables  $s_{ij}$  and  $m_{uv}$  as described above. Let each trajectory after transformation using the variables be denoted by  $T'_n$  where  $n = 1, 2, \dots, N$  and the set of  $N$  trajectories be denoted by  $C_0$ . New synthetic data denoted by  $C_1$  was created of same size as  $C_0$  i.e. the synthetic data  $C_1$  also consists of  $N$  rows.

The problem is converted to a supervised learning problem by creating a target variable  $y$ .

The value of the target variable  $y$  for each row is defined as follows

$$y_i = 0 \text{ if row } i \in C_0$$

$$y_i = 1 \text{ if row } i \in C_1$$

The target variable  $y$  thus assigns a class label to each row. The records in  $C_0$  belong to class 0 while those in  $C_1$  belong to class 1.

Each record in data  $C_1$  contains the same number of variables as those in data  $C_0$ . The values for variables in  $C_1$  are generated from the corresponding variables in  $C_0$  as described. Consider a specific numeric variable  $X \in \{s_{ij}, m_{uv}\}$ , which could be related to either a stop or move in the data  $C_0$ . Let the values of  $X$  range from  $x_{min}$  (minimum) to  $x_{max}$  (maximum) across all the trajectories in  $C_0$ , and the standard deviation be  $x_{std}$ . The values for the corresponding variable  $X$  in the synthetic data  $C_1$  are chosen to be uniformly distributed in the interval  $\{x_{min} - r \times x_{std}, x_{max} + r \times x_{std}\}$ , where  $r$  is an integer and can take values like 1, 2, 3 etc. Consider a specific categorical variable  $Y \in \{s_{ij}, m_{uv}\}$ , which can be related to either a stop or move in the data  $C_0$ . Let the values of  $Y$  belong to one of  $|Y|$  categories among all the trajectories in  $C_0$ . The values for the variable  $Y$  in the synthetic data  $C_1$  are randomly chosen to be one among the  $|Y|$  categories. An example to illustrate this is shown in Fig. 64. The value of  $r$  chosen to generate the synthetic data is  $r = 2$ . Consider the variable  $s_{11}$  Fig. 64, and let  $\sigma$  be the standard deviation of  $s_{11}$  across all the rows in  $C_0$ . The value for the numeric variable  $s_{11}$  in data  $C_1$  is therefore uniformly distributed between  $[90 - \sigma, 850 + \sigma]$ , assuming 90 and 850 are the minimum and maximum values of the  $s_{11}$  in data  $C_0$ . Similarly, the values for the categorical variable  $s_{21}$  in data  $C_1$  are cho-



sen randomly to be one of the categories [“*Loading*”, “*QualityCheck*”, “*Distribution*”], assuming the values for variable  $s_{21}$  in  $C_0$  belong to one of these categories.

The purpose of creation of synthetic data  $C_1$  is to break the dependency structure between variables in the original data  $C_0$ . As an example, consider the variable  $s_{11}$  which denotes the activity at a stop and another variable  $s_{12}$  denoting the time taken at that particular stop. For most trajectories in the original data  $C_0$ , if the activity of *Loading*, takes a specific time i.e. approximately *1 hr*. This means there is a dependency between the variables  $s_{11}$  and  $s_{12}$ , because whenever the value of  $s_{11} = \textit>Loading$ , the value of  $s_{12} = 1hr$ . However, since the values for variables  $s_{11}$  and  $s_{12}$  in data  $C_1$  are obtained by randomly choosing among the corresponding values of the variables in  $C_0$ , the dependency between the variables no longer exists.

Let the combined data containing  $N$  records from  $C_0$  and  $N$  records from synthetic data  $C_1$ , be denoted by  $C_T$ . The data  $C_T$  containing  $2N$  records is thus a two class problem, with the target variable  $y$  denoting the class trajectory belongs to, which can be either class 0 or 1.

## **2.1. CLASSIFICATION USING RANDOM FORESTS**

We have a chosen decision trees to classify the two class data  $C_T$ . A decision tree [78] is a model used for classification, which predicts the value of a target variable based on several input variables. Alternate classification techniques such as support vector machines [91], neural networks [91] etc. could have been used to classify the two class data  $C_T$ . We chose decision trees taking into consideration the challenges encountered while clustering the trajectory data.

Traditional clustering techniques cannot be applied to cluster semantic trajectories due to various reasons. The most important challenge with clustering semantic trajectories is that they have both numeric and categorical attributes e.g. consider the semantic trajectories arising in supply chain while transporting goods in trucks, the attributes such as time for the transport are numeric, while the attributes such as mode of transport are categorical. Distance measures such as Euclidean distance cannot be used for this kind of data, since Euclidean distance can be calculated only between numeric values.

Another problem to be addressed while clustering such trajectories is that trajectories can be of different lengths. In order to use distance measures such as Euclidean distance to compute distance between multi-dimensional vectors, the vectors need to be of equal size.

Trajectories can also have a large number of variables depending on the number of stops and moves in a trajectory. As a result, the trajectory data can be high dimensional. As the dimensionality of the data increases, the volume of space increases so fast that the available data becomes sparse. Hence, in high dimensional data all objects appear to be sparse and dissimilar in many ways. The sparsity can be problematic for any method that requires statistical significance. This phenomenon is referred to as curse of dimensionality [92].

The desirable characteristics of the supervised learner are to handle mixed data types, missing values, large feature space etc. Decision trees handle such data very well. Single tree classifiers are unstable due to high variance. Ensemble methods use multiple models to obtain better predictive performance [93]. We have used ensembles of trees to classify our data.

Let  $RF$  denote the random forest [78] that is constructed consisting of  $L$  trees  $rf_1, rf_2 \dots rf_L$  and using data  $C_T$ . A detailed explanation of random forests is provided in

Chapter 2. The purpose of classifying the two class data  $C_T$  using random forest  $RF$  is to obtain a similarity measure between the  $N$  trajectories in the original data as the traditional distance measures such as Euclidean distance cannot be used to measure the similarity between the trajectories. This similarity measure obtained from the random forest  $RF$  was used later for clustering the trajectories in the original data  $C_0$ .

The similarity measure can be calculated using the random forest  $RF$  for any two trajectories say  $T'_1$  and  $T'_2$  in the data  $C_0$  as follows. For each of the two trajectories, we first propagate their values down all the  $L$  trees within  $RF$  and a terminal node is assigned to them. Next, the terminal node position for each trajectory in each of the trees is recorded. Let  $z_{n_1} = (z_{n_11}, z_{n_12} \cdots z_{n_1L})$  be the terminal node positions in the  $L$  trees for  $T'_{n_1}$  and similarly define  $z_{n_2}$ . Then the similarity between pair  $T'_1$  and  $T'_2$  is set to:

$$S(T'_{n_1}, T'_{n_2}) = \frac{1}{l} \sum_{i=1}^L I(z_{n_1i} == z_{n_2i})$$

where  $I$  is the indicator function.

The similarity measure is calculated between all pairs of the  $N$  trajectories in  $C_0$ , and a proximity matrix  $PR$  is created. The  $ij$ th element of the proximity matrix  $PR$ , produced by random forest  $RF$  is the similarity measure  $S(T'_i, T'_j)$  between trajectories  $T'_i$  and  $T'_j$  in  $C_0$ . The intuition is that similar trajectories should be in the same terminal nodes more often than dissimilar ones. The proximity matrix  $PR$  can be used to identify structure in the data or for unsupervised learning with random forests. This proximity matrix obtained was used for clustering the trajectories.

The similarities obtained from the proximity matrix  $PR$ , were used to cluster the trajectories using hierarchical clustering [91] technique, as any clustering algorithm needs a similarity to group objects into clusters. The linkage criteria used for the hierarchical clus-

tering was average linkage. In the average linkage method the distance between the two clusters is defined as the average of the distances between all pairs of objects, where one member of the pair is from each of the cluster. This type of linkage was preferred over other methods like single and complete linkage, since it uses information on all pairs of distance, not merely the minimum or maximum distances [91].

### **3. OUTLIER DETECTION IN SEMANTICALLY ENRICHED TRAJECTORIES**

Outliers in semantic trajectories can be of various types. One kind of outliers is a trajectory which has a different path i.e. a different stop and move sequence compared to the rest of the trajectories. These outliers are extreme, since they follow a different path altogether compared to the rest of the trajectories. The other kind of outliers are those trajectories which have the same stop and move sequence but the attributes describing the motion vary at these stops or moves. These outliers are more subtle in nature and are difficult to identify. An example of this outlier is as follows: consider trucks which are transporting goods from point A to point B, if most of the trucks take approximately one hour to transport the goods, an outlying trajectory is one which takes a considerably larger amount of time to transport the goods. We focus on detecting such outlying trajectories, which are more subtle in nature.

To detect the outliers, the model should be able to capture the interactions between variables e.g. the interaction between the numeric and categorical variable. For example, if a particular activity (categorical variable) takes a specific amount of time (numeric variable) in most trajectories, an outlying trajectory would be one where that particular activity takes more or lesser time than the usual.

### 3.1. WHY RANDOM FORESTS FAIL?

Outliers are those trajectories whose proximity to rest of the trajectories is small (or distance is large). Let the average proximity from trajectory  $T'_n$  to the rest of the trajectories be

$$\overline{PC}(n) = \sum_{r=1}^N prox^2(T'_n, T'_r)$$

where  $prox(T'_n, T'_r)$  is the proximity between trajectory  $T'_n$  and trajectory  $T'_r$ . The raw outlier measure  $O_n$  or Breiman outlier score [80] for trajectory  $T'_n$  is defined as

$$O_n = \frac{N}{\overline{PC}(n)}$$

The model developed using random forests was used to obtain a proximity matrix  $PR$ . The  $ij$ th element of the matrix gives the proximity between trajectory  $T'_i$  and trajectory  $T'_j$ , which was used to calculate the Breiman outlier score as described earlier. However, the score obtained using the matrix  $PR$  did not prove useful to detect outliers in the trajectories.

Consider a training set  $S = (X, Y)$ , consisting of instances  $(x_i, y_i), i = 1 \cdots n$ , where  $x_i \in X$  is an input vector and  $y_i \in Y$  is its corresponding class label.  $x_i$  is made up of a number of features  $f_1, f_2 \cdots f_m \in M$ . A decision tree splits each of the input vector  $x_i$  based on the values of the features  $f_1, f_2 \cdots f_m$ . At each node of the decision tree, the input data is partitioned into subsets, such that the impurity of the resulting child node is decreased. Some of the impurity measures that can be used are discussed in Chapter 2. This process is repeated to form a tree.

In order to select the best variable to be used to split at a node, the degree of impurity of the parent node (before splitting) is compared with the degree of impurity of the child

nodes (after splitting). The gain,  $\delta$ , is a criteria used to determine the goodness of a split:

$$\delta = I(\text{parent}) - \sum_{j=1}^k \frac{N(v_j)}{N} I(v_j)$$

where  $I(\cdot)$  is the impurity measure of a given node,  $N$  is the total number of records at the parent node,  $k$  is the number of attribute values, and  $N(v_j)$  is the number of records associated with the child node,  $v_j$ . Decision tree algorithms often choose a variable that maximizes the gain  $\delta$ .

This introduces a bias while selecting variables for splitting, as the variables with more number of values when selected lead to a larger value for  $\delta$ . The trees in the random forest thus tend to select variables with many splitting values e.g. numeric variables are selected compared to categorical variables. Due to this bias introduced by random forest, they cannot capture the interaction between categorical and numeric variables, which is essential for detecting outliers in trajectories.

### **3.2. OUTLIER DETECTION USING CONDITIONAL INFERENCE FOREST**

For the purpose of outlier detection, alternate ways of growing the trees were explored. The conditional inference trees [83], described in detail in Chapter 2, use a different criteria for tree growing which reduces the bias in variable selection. They separate the variable selection and splitting step unlike decision trees where variables are selected based on the best split obtained which reduces the entropy in the child nodes. We build a forest of conditional inference trees [83] using the trajectory data. The proximity measure obtained from the trees in the conditional inference forest (cForest) is used for outlier detection.

The details of the method are as follows. The data  $C_T$ , consisting of  $N$  records from  $C_0$  and  $N$  records from  $C_1$  was classified using a forest of  $L$  conditional inference trees

denoted by  $CF : \{cf_1, cf_2, \dots, cf_L\}$ . A proximity matrix  $PC$  is obtained from  $CF$ , for the  $N$  trajectories in  $C_0$  by recording the terminal node positions in each of the  $L$  trees of  $CF$ , as described in the previous section.

The proximities obtained from the matrix  $PC$ , were used to calculate the outlier scores  $O_n$ , as described earlier. Once the Breiman scores  $O_n$  were obtained for all the  $N$  trajectories, a threshold was chosen to distinguish outliers from normal trajectories. This threshold was chosen using two methods - Isolation Forest (IF) and Control Chart (CL) method.

The threshold for CL method was chosen as follows. Let  $\mu_N$  denote the mean of the  $N$  Breiman scores  $O_N$  and  $\sigma_N$  denote their standard deviation. The threshold  $\epsilon$  used for outlier detection was  $\mu_N + 3\sigma_N$ . Any trajectory  $T_n$  with score  $O_n$  greater than  $\epsilon$  was considered to be an outlier.

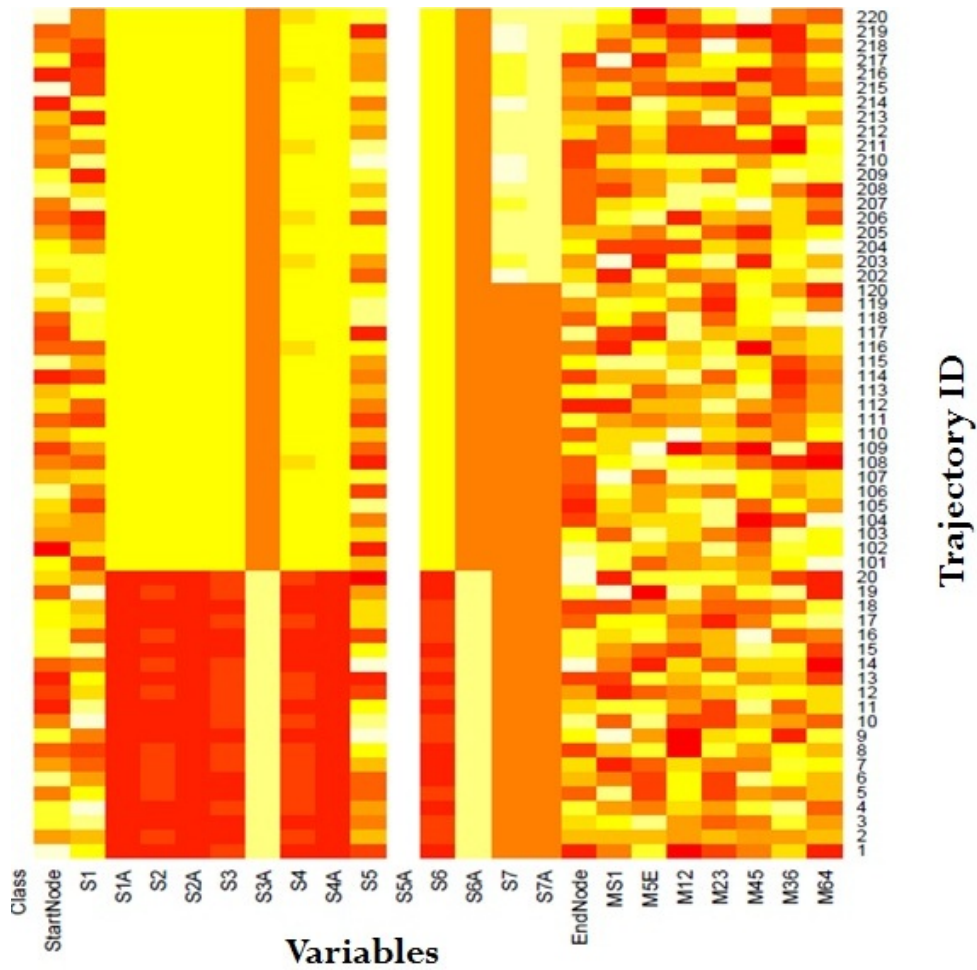
In IL method, a forest of isolation trees were grown using the scores  $O_n$ . The details of isolation forest are provided in Chapter 2. Each of the  $O_n$  values were mapped to an equivalent value  $s_n$ , where  $0 \leq s_n \leq 1$  by the isolation forest. The trajectories  $T_n$  for which  $s_n \geq 0.6$  were considered to be outliers.

## 4. EXPERIMENTS

The experiments for clustering semantic trajectories are shown in Section 4.1, while the experiments for outlier detection in semantic trajectories are shown in Section 4.2

### 4.1. EXPERIMENTS FOR CLUSTERING

Data was simulated similar to semantic trajectories containing a sequence of stops and moves. Variables were created corresponding to the the stops and moves of the trajectories as described in the previous section. A dataset consisting of 300 trajectories and 23



**Figure 65.** Heat map of the trajectory data represented using the variables  $s_{ij}$  and  $m_{uv}$ . The rows corresponds to the trajectories and the columns correspond to the variables. Only first 20 trajectories belonging to three clusters are shown, where each cluster consists of 100 trajectories.

variables, which correspond to eight stops and seven moves, was used for clustering. The numeric variables used are shown in Table 11, while the categoric variables are shown in Table ???. This method is easily scalable with trajectories consisting of larger number of stops and moves. If the stops and moves increase, the number of variables would increase and hence the feature space to be used for classification increases. Decision trees can han-



**TABLE 10.** Numeric variables used in clustering experiments

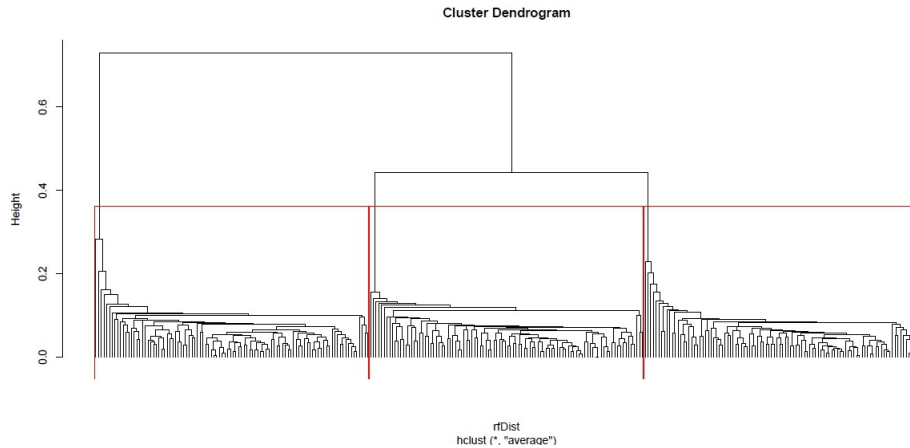
Variable	Mean in cluster 1	Mean in cluster 2	Mean in cluster 3
$s_{11}$	31.13	31.13	31.13
$s_{21}$	81.44	81.44	81.44
$s_{31}$	21.63	59.7	59.7
$s_{41}$	51.34	101.3	101.3
$s_{51}$	70.96	96.7	96.7
$s_{61}$	100.85	100.85	100.85
$s_{71}$	30.5	78.6	78.6
$s_{81}$	-	-	12.45
$s_{91}$	26.3	26.3	26.3
$m_{11}$	16.99	16.99	16.99
$m_{21}$	29.8	29.8	29.8
$m_{31}$	51.1	51.1	51.1
$m_{41}$	61.4	61.4	61.4
$m_{51}$	27.3	27.3	27.3
$m_{61}$	73.8	73.8	73.8
$m_{71}$	49.1	49.1	49.1

**TABLE 11.** Categorical variables used in clustering experiments

Variable	Category in cluster 1	Category in cluster 2	Category in cluster 3
$s_{22}$	Loading	Packing	Packing
$s_{32}$	Eating	Fuelling	Fuelling
$s_{42}$	OrderProcessing	Neworder	Neworder
$s_{52}$	QualityCheck	ReManufacturing	ReManufacturing
$s_{62}$	Unloading	Unloading	Unloading
$s_{72}$	OrderAcknowledgement	Finance Assesment	Finance Assesment
$s_{82}$	-	-	OrderCheck

dle data with large number of features very well, since only a subset of features are used while splitting at each of the tree.

The trajectory data was simulated as to have three clusters. The first and the second clusters of trajectories have the same sequence of stops and moves but they differ in the activities and the time at the stops. The third cluster differs from the first and second in the sequence of places (or stops) visited, but these trajectories have approximately simi-



**Figure 66.** Dendrogram of the clusters obtained by hierarchical clustering. The three clusters in the trajectory data are clearly visible in the dendrogram.

lar times at the similar stops visited. The values of the numeric variables, for trajectories belonging to the same cluster, were simulated as follows: the values are chosen to be uniformly distributed between  $\{\mu_j^C - \delta, \mu_j^C + \delta\}$ , where  $\mu_j^C$  is the mean value of the variable  $x_j$  corresponding to  $C$ th cluster and  $\delta$  is amount of the noise induced. We had conducted experiments with different values of  $\delta$ . The mean values of the numeric variables in the three clusters are shown in Table 11.

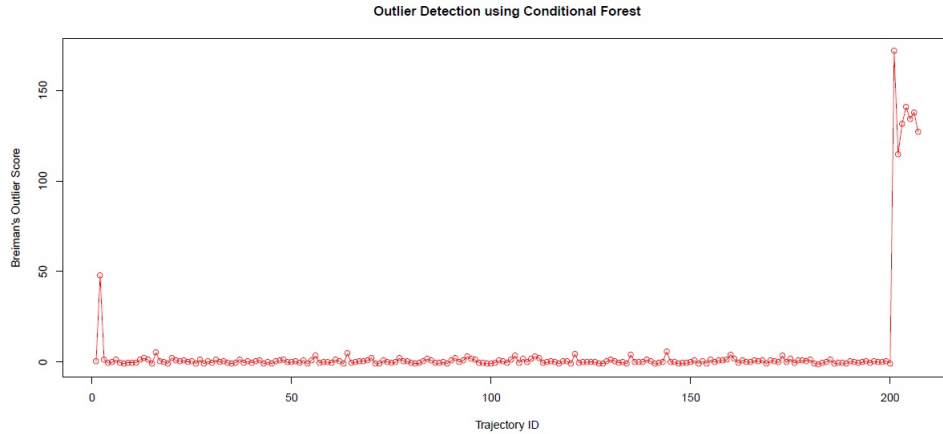
A heat map of the trajectory data, consisting of the first 20 trajectories from the three clusters is shown in Fig. 65. The rows in the figure correspond to the trajectories and the columns correspond to the variables in each trajectory. A heat map is a graphical representation of data where each individual value in the data matrix is represented as a color. The first 20 trajectories with trajectory IDs 1 to 20 in the heat map shown in Fig. 65 correspond to trajectories from the first cluster, the next 20 trajectories with IDs 101 to 120 belong to second cluster and the last 19 trajectories with IDs 202 to 220 belong to the third cluster.

Let the data consisting of 300 trajectories after variable creation using  $\delta = 0.5$  be denoted as  $C_0$ , and the class label be 0. Synthetic data  $C_1$  was created consisting of 300 rows using  $r = 1$  as described earlier, and a class label of 1 was assigned to them. RFs were used to classify the two class data  $C_T$ . The proximity measures obtained from the RFs were used to cluster the trajectories in the data  $C_0$ . Hierarchical clustering method was with average linkage was used to cluster the trajectories. Hierarchical clustering does not require us to prespecify the number of clusters and also it outputs a hierarchical tree-like structure which is more informative than the unstructured set of clusters. The cluster dendrogram is shown in Fig. 66. The dendrogram in the figure shows the three clusters present in the trajectory data. Hence our method, is able to identify the clusters in the data.

#### **4.2. EXPERIMENTS FOR OUTLIER DETECTION**

To detect outlying trajectories, data was created similar to the trajectories generated while transporting goods in a supply chain. Here, 100 trajectories were created having eight stops and seven moves. The activities at the the six stops excluding the first and last are loading, eating, order processing, order acknowledgement, quality checking and unloading respectively. Each activity is associated with its corresponding time. Another 100 trajectories were created having the same stop sequence as the previous trajectories but with different activities at the corresponding stops. The activities at the six stops excluding the first and last are packing, fueling, new order loading, re-manufacturing, finance assessment, unloading respectively.

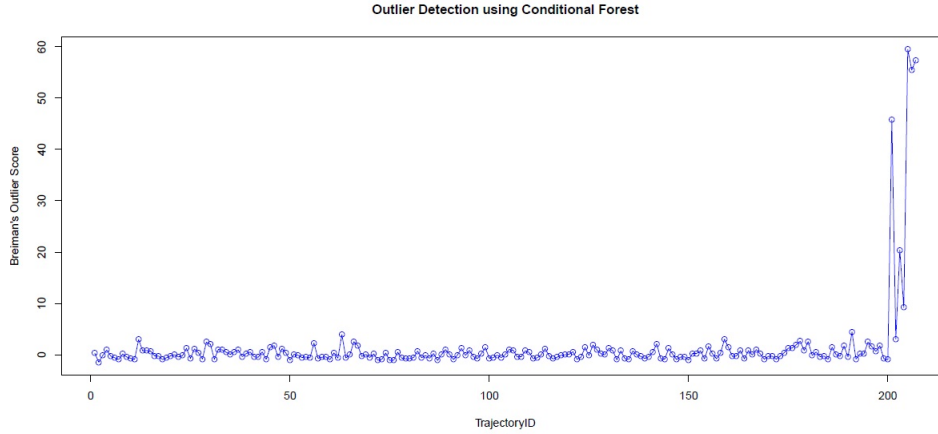
Seven outlying trajectories with IDs 201 to 207 were added to the 200 trajectories. These outliers were created so as to vary the time duration (increase or decrease) at the stops or moves of the trajectories. Quality check activity at stop 4 generally takes around 70



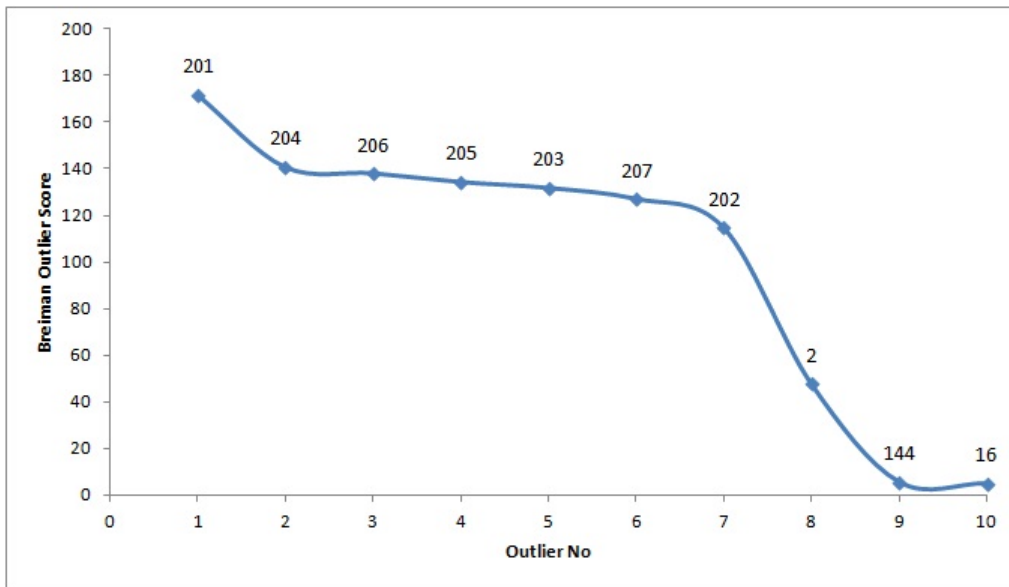
**Figure 67a.** Breiman outlier scores of 207 trajectories with  $\delta = 0.5$  and  $r = 1$ . The scores for the 7 outlying trajectories with IDs 201 to 207 are higher than the rest of the trajectories.

minutes. Two outliers were generated with time for quality check activity being increased to 85 and 120 minutes respectively. Order processing activity at stop 3 generally takes around 50 minutes. Two outliers were generated with time for order processing activity decreased to 45 and 15 minutes respectively. The time for moving from stop 2 to stop 3 generally takes approximately 60 minutes. Three other outliers were generated with the time for move from stop 2 to stop 3 increased in steps of 10 minutes to 70, 80 and 90 minutes respectively.

The method to detect outlying trajectories using conditional inference trees, as described earlier, was tested at different levels of noise  $\delta$  in the data  $C_0$ , and for values of  $r$  for generating the synthetic data  $C_1$ . The noise in the trajectory data  $C_0$  was added at two different levels relative to the average value i.e.  $\delta = 0.5, 2.5$ . At each of the noise levels  $\delta$ , the Breiman outlier score was plotted for the 207 trajectories. A plot of the Breiman outlier scores for the two noise levels  $\delta = 0.5$  and  $\delta = 2.5$  are shown in Fig. 67a and Fig. 67b respectively. The trajectories are ordered in terms of decreasing outlier score, and the top 10 trajectories with highest outlier scores were plotted for the different noise levels in the

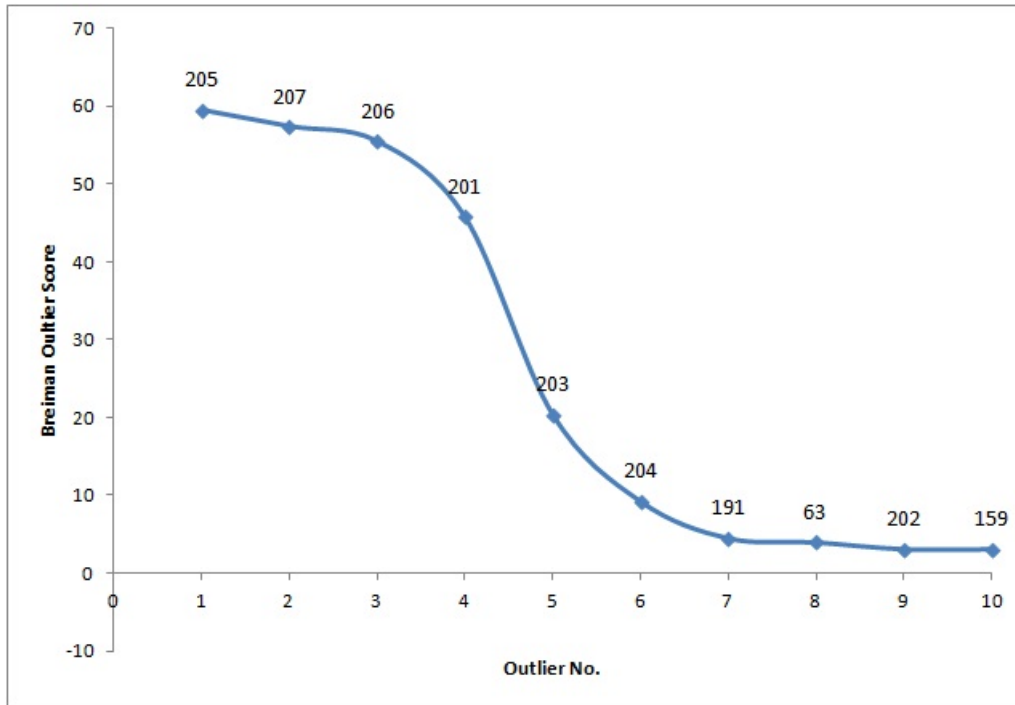


**Figure 67b.** Breiman outlier scores of 207 trajectories with  $\delta = 2.5$  and  $r = 1$ . Comparing with Fig. 67a, higher noise in the trajectories ( $\delta = 2.5$ ), makes it difficult to identify the outlying trajectories with IDs 201 to 207, compared to lower noise ( $\delta = 0.5$ ).



**Figure 68a.** 10 trajectories with highest outlier scores for  $\delta = 0.5$  and  $r = 1$ . The outlying trajectories with IDs 201 to 207 are among the 10 trajectories with highest scores.

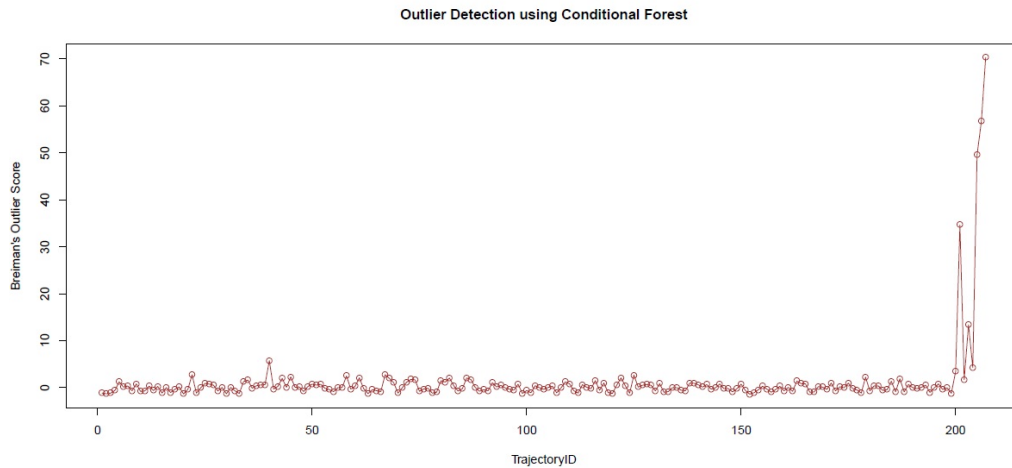
data. The plots for the ordered outlier scores for two noise levels  $\delta = 0.5$  and  $\delta = 2.5$  are shown in Fig. 68a and Fig. 68b respectively. It can be seen from Fig. 67 and Fig. 68 that our method was able to detect outliers at both the noise levels  $\delta$ . However, our method is



**Figure 68b.** 10 trajectories with highest outlier scores for  $\delta = 2.5$  and  $r = 1$ . The outlying trajectories with IDs 201 to 207 are among the 10 trajectories with highest scores.

sensitive to the noise in the data, and the results are better with lesser noisy data i.e when  $\delta = 0.5$ .

Further experiments were conducted to check how our method performs with the variation in noise in the synthetic data  $C_1$  created i.e.  $r$ , as described previously. A value of  $r = 2$  was used, hence the variables in the data  $C_1$  are now chosen such that they are between minimum  $- 2\sigma$  and maximum  $+ 2\sigma$  of the corresponding variables in data  $C_0$ . The outlier scores for the 207 trajectories are shown in Fig. 69a. The outlier scores are ordered in decreasing order and the trajectories with 10 highest scores are shown in Fig. 69b. It can be seen from the Fig. 69 our method performs better with lower value of  $r$ . As the value of  $r$  increases, the synthetic data  $C_1$  created would have a higher variance and would be



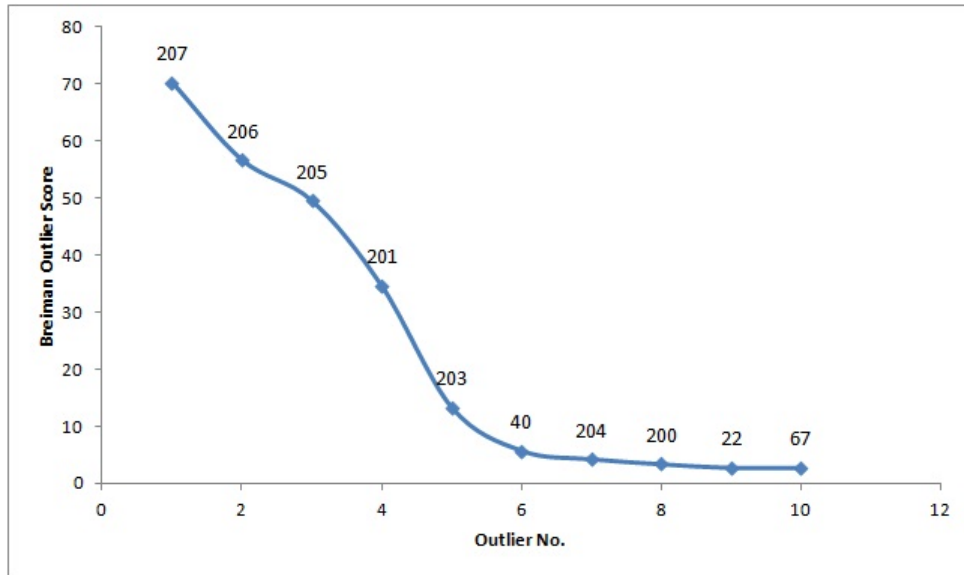
**Figure 69a.** Breiman outlier scores for 207 trajectories with  $\delta = 0.5$  and  $r = 2$ . Comparing with Fig. 67a, higher value of  $r$  ( $r = 2$ ), makes it difficult to identify the outlying trajectories with IDs 201 to 207, compared to lower value of  $r$  ( $r = 1$ ).

more sparse in space compared to when  $r$  is small. Hence, the capability of the classifier to differentiate between normal points and anomalies decreases.

4.2.1. *MODEL EVALUATION.* In order to evaluate the outlier detection method and the parameter settings of  $r$  and  $\delta$ , the True Positive Rate (TPR) and the False Positive Rate (FPR) were calculated. Two different thresholds using the IF and CL method as described earlier were used to detect the outliers. The results for various parameter setting are shown in Fig. 70.

The following conclusions can be drawn from the Fig. 70

- As the value of  $r$  or  $\delta$  increases, the FPR increases i.e. the tendency of outliers to be considered normal increases with higher values of  $r$  or  $\delta$ .
- The TPR using CL threshold is always higher than the IF threshold i.e. the tendency for normal points to be considered as outliers is lesser if CL threshold is used compared to IF threshold.



**Figure 69b.** 10 trajectories with highest outlier scores with  $\delta = 0.5$  and  $r = 2$ . Out of the 7 outlying trajectories with IDs 201 to 207, only 6 are among the top 10 trajectories with high outlier scores.

r	$\delta$	IF Threshold		CL Threshold	
		TPR	FPR	TPR	FPR
1	0.5	0.54	0	1	0
1	2.5	0.7	0	1	0.015
2	0.5	0.85	0.005	1	0.02

**Figure 70.** TPR and FPR values obtained for different thresholds and varying parameter values  $r$  and  $\delta$ .

- The FPR using IF threshold is always lower than the CL threshold i.e. the tendency for outliers to be considered normal is lesser in the case of IF threshold compared to CL threshold.



APPENDIX A

APPENDIX

## 1. Modified Baum Welch Algorithm Implementation (MATLAB CODE)

```
tol = 1e-6;
trtol = tol;
etol = tol;
maxiter = 5000;
% changeRow is the index of the row to be changed
changeRow = 1;
count = 1;
% guessTR is the initial transition probability matrix
[numStates, checkTr] = size(guessTR);
% guessE is the initial emission probability matrix
[checkE, numEmissions] = size(guessE);
[numSeqs, seqLength] = size(seqs);
TR = zeros(size(guessTR));
% loglik is the log likelihood of all sequences
given the TR and E
loglik = 1;
logliks = zeros(1,maxiter);
for iteration = 1:maxiter
    oldLL = loglik;
    loglik = 0;
    oldGuessE = guessE;
    oldGuessTR = guessTR;
```

```

seq = seqs(count, :);

[p, logPseq, fs, bs, scale] = hmmdecode(seq, guessTR, guessE);

loglik = loglik + logPseq;

logf = log(fs);

logb = log(bs);

logGE = log(guessE);

logGTR = log(guessTR);

seq = [0 seq];

    for k = 1:numStates
        for l = 1:numStates
            for i = 1:seqLength
                if k == changeRow
                    TR(k,l) = TR(k,l) + exp( logf(k,i)
                    + logGTR(k,l) + logGE(l,seq(i+1))
                    + logb(l,i+1))./scale(i+1);
                else
                    TR(k,l) = guessTR(k,l)
                end
            end
        end
    end

    end

totalTransitions = sum(TR,2);

guessTR = TR./(repmat(totalTransitions,1,numStates));

```

```

logliks(iteration) = loglik;

    if (abs(loglik-oldLL)/(1+abs(oldLL))) < tol
        if norm(guessTR - oldGuessTR,inf) ./
            numStates < trtol

            fprintf('Algorithm converged after
                    \ %d iterations.',iteration)

            fprintf('Loglikelihood value is
                    \ %d.',logliks(iteration))

            converged = true;

            break

        end

    end

end
end
end

```

## 2. Creating daily trajectories of cabs (R CODE)

```

cabid <- 1

while(cabid <= 100)

{

name <- sprintf("Cab_%d.txt",cabid)

d = read.table(name, sep=" ",

               col.names=c("long", "lat", "occupancy", "time"),

               fill=FALSE, strip.white=TRUE)

doc <- 1

```

```

while(doc <= 24)
{
  subname <- sprintf("Cab%d_%d.txt",cabid, doc)
  ref <- d[1,4]
  ref <- as.Date(ISOdatetime(1970,1,1,7,0,0,"MST") + ref)
  df1 <- subset(d, as.Date
                (ISOdatetime(1970,1,1,7,0,0,"MST")+
                d[,4]) == ref)
  d <- subset(d, as.Date
              (ISOdatetime(1970,1,1,7,0,0,"MST")+
              d[,4]) != ref)
  if(nrow(df1) != 0)
  {
    write.table(df1, file = subname)
  }
  doc <- doc + 1
}
cabid <- cabid + 1
}

```

### 3. Compressing trajectory data using DP algorithm (R CODE)

```

full = NULL

it <- 1

cabid <- 101

```

```

while(cabid <= 500)
{
doc <- 1
while(doc <= 24)
{
name <- sprintf("Cab%d_%d.txt",cabid, doc)
d = try(read.table(name,sep=" ",
                  col.names=c("long","lat","occupancy","time"),
                  fill=FALSE,strip.white=TRUE), silent = TRUE)
if(inherits(d, "try-error"))
{
//Do Nothing
}
else
{
points <- list(x=d$long,y=d$lat)
simpleLine <- dp(points, 0.01)
m1<- as.matrix(simpleLine$x)
m2 <- as.matrix(simpleLine$y)
newm <- cbind(m1,m2)
ind1 <- apply(d[,1:2], 1, paste, collapse = "/")
ind2 <- apply(newm, 1, paste, collapse = "/")
newtime <- as.matrix(d[match(ind2, ind1),4])

```

```

newtime <- newtime - min(newtime)
m3 <- cbind(m1,m2,newtime,it,cabid)
colnames(m3) <- c("long","lat", "time","trajID","CabID")
full <- rbind(full, m3)
it <- it + 1
}
doc <- doc + 1
}
cabid <- cabid + 1
}

```

#### **4. Discretizing trajectory data using BIRCH clustering (R CODE)**

```

obj <- birch(trajdata[,1:4],0.1, keeptree = TRUE)
obj <- birch.getTree(obj)
mem <- obj$members
size <- 1
appended = NULL
while(size <= length(mem))
{
  m1 <- unlist(mem[size])
  m2 <- cbind(m1,size)
  appended <- rbind(appended,m2)
  size = size + 1
}

```

```

appended <- appended[sort.list(appended[,1]), ]
newfull <- cbind(full,appended[,2])
colnames(newfull) <- c("long1","lat1",
"long2","lat2", trajID,"CabID","ClusterID")

```

## 5. Clustering trajectories using mixture of regression (R CODE)

```

trajdata <- data.frame(full)
ex1 <- flexmix(~time|trajID,
data=trajdata, k = 4,
model=list(FLXMRglm(long~.),FLXMRglm(lat~.)))
table(trajdata$trajID, clusters(ex1))
ex1@cluster

```

## 6. HMM for change detection in trajectories (R CODE)

```

*****Creating Learning Sample*****
sampnew <- vector("list", 24)
doc <-1
sampcount <-1
while(doc <=24){
name <- sprintf("Cab2_%d.txt",doc)
d = read.table(name,sep=" ",fill=FALSE,strip.white=TRUE)
sampnew[[sampcount]] <- as.matrix(d[,1:2])
doc <- doc + 1
sampcount <- sampcount + 1

```



```

}

*****Learning****

it <- 1

fit <- 1

learn <- matrix(rep("NA"), ncol=2,nrow=30)

while(fit <= 30)

{

fitted <- HMMFit(obs=fitsamp, nStates = fit,
                 asymptMethod = "optim")

learn[it,1] <- fit

learn[it,2] <- fitted$LLH

fit <- fit + 1

print(it)

it <- it + 1

}

plot(learn[,1],learn[,2], xlab = "No.of States",
     ylab = "Log-Likelihood",
     main = "Variation of likelihood with no.of states",
     col = "red")

*****Testing *****

tnum <- 2

```

```

while(tnum <= 20)
{
print(****tnum****)
tsubdata <- fullclust[[tnum]]
tuniquevals <- unique(tsubdata$trajID)
trainset <- vector("list", length(tuniquevals))
i <- 1
while (i <= length(tuniquevals))
{
trainset[[i]] <- subset(tsubdata, trajID ==
tuniquevals[i])
trainset[[i]] <- trainset[[i]][,1:2]
i <- i + 1
}

fitted <- HMMFit(obs=trainset, nStates = 20,
asymptMethod = "optim")

num <- 1
results <- matrix(rep("NA"),nrow = 250, ncol = 20)
while(num <= 20)
{
if(num == tnum)

```

```

{
  num = num + 1
}
else
{
print(num)

subdata <- fullclust[[num]]

uniquevals <- unique(subdata$trajID)

testset <- vector("list", length(uniquevals))

it <- 1

while (it <= length(uniquevals))

{

testset[[it]] <- subset(subdata, trajID == uniquevals[it])

testset[[it]] <- testset[[it]][,1:2]

it <- it + 1

}

fullset <- c(trainset,testset)

est <- 0

fbLog <- NULL

tfitted <- NULL

test <- 25

```

```

totalen <- length(uniquevals) + 59
while(test <= totalen)
{
  est <- 0
  low <- test - 24
  tfitted <- NULL
  tfitted <- HMMFit(obs=fullset[low:test], nStates = 20,
                    asymptMethod = "optim")
  while(low <= test)
  {
    fbLog <- NULL
    fbLog <- forwardBackward(fitted, fullset[[low]])
    est <- est + fbLog$LLH
    low <- low + 1
  }
  results[test,num] <- tfitted$LLH - est
  test = test + 1
}
num <- num + 1
}
}
outfile <- sprintf("Cluster_%d_vsRest.csv", tnum)
write.csv(results, file = outfile)

```

```
tnum <- tnum + 1  
}
```

## 7. Sequential Pattern Mining (R CODE)

```
***Preparing data for Prefixspan algorithm ***  
prefixdata <- NULL  
it <- 1001  
while(it <= 4000)  
{  
temp <- as.integer(trajfull[which(trajfull$trajID ==it),9])  
temp <- append(temp, c(-1,-2))  
prefixdata <- append(prefixdata,temp)  
it <- it + 1  
}  
write.table(prefixdata, file = "SampData.txt")  
x <- scan(file="SampData.txt",what="integer")  
x <- as.integer(x)  
x  
zz<-file("prefixinput.data", "wb")  
writeBin(x, zz, size=4)  
close(zz)  
zz<-file("prefixinput.data", "rb")  
readBin(zz, integer(), 1000, size=4)  
close(zz)
```

```

****CreatingSequences****

seqlist <- list()

count <- 1

it <- 1

while(it <= 3363)

{

temp <- as.integer(trajfull500_discrete

[which(trajfull500_discrete$trajID ==it),6])

seqlist[[count]] <- temp

it <- it + 1

count <- count + 1

}

maxLen <- max(sapply(seqlist, length))

newseqlist <- lapply(seqlist,

function(.ele){c(.ele, rep(NA, maxLen))[1:maxLen]})

seqs <- do.call(rbind, newseqlist)

write.csv(seqs, file="Seq_Cabs1To150_B0.0001_2.csv")

****CreatingSubsets****

mydata <- read.csv("Seq_traj4000_341clusters.csv")

mydata <- mydata[1001:4000,]

z <- 26

```

```

while(z <= 1000){
name <- sprintf("TestSample_25_%d.csv",z)
mysample<- mydata[sample(1:nrow(mydata), 25, replace=FALSE),]
write.csv(mysample, file = name)
z <- z + 1
}

```

## 8. Distribution Based Method (R CODE)

```

****Creating sequences with increased time
intervals between items ****
seqlist <- list()
count <- 1
it <- 1
while(it <= 1000)
{
temp_item <- as.integer(trajfull500_discrete
[which(trajfull500_discrete$trajID ==it),6])
full <- NULL
k <- 1
while(k <= length(temp_item))
{
z <- 1.2*k
temp_list <- append(2000+ z,temp_item[k])
full <- append(full, temp_list)

```

```

k <- k + 1
}
seqlist[[count]] <- full
it <- it + 1
count <- count + 1
}
maxLen <- max(sapply(seqlist, length))
newseqlist <- lapply(seqlist,
function(.ele){c(.ele, rep(10000, maxLen))[1:maxLen]})
seqs <- do.call(rbind, newseqlist)
write.csv(seqs, file="Seq_Test_Time1.1.csv")

***Constructing multivariate control chart*****
z <- 1
train1 <-NULL
train2 <- NULL
while(z <= 25)
{
name1 <- sprintf("1-2_TrainFile_%d.csv", z)
name2 <-  sprintf("2-3_TrainFile_%d.csv", z)
t1<- read.csv(file = name1, header = FALSE)
t2<- read.csv(file = name2, header = FALSE)
t1 <- mean(t1)

```



```

t2 <- mean(t2)

train1 <- rbind(train1, t1)
train2 <- rbind(train2, t2)

z = z + 1

}

traindata <- cbind(train1, train2)

z <- 1

test1 <-NULL

test2 <- NULL

while(z <= 25)

{

name1 <- sprintf("1-2_TestFile_%d.csv", z)
name2 <-  sprintf("2-3_TestFile_%d.csv", z)
t1<- read.csv(file = name1, header = FALSE)
t2<- read.csv(file = name2, header = FALSE)

t1 <- mean(t1)

t2 <- mean(t2)

test1 <- rbind(test1, t1)

test2 <- rbind(test2, t2)

z = z + 1

}

testdata <- cbind(test1, test2)

qq = mqqc(traindata, type = "T2.single",

```

```

newdata = testdata, confidence.level = 0.999,
pred.limits = TRUE)

```

## 9. Clustering and Outlier detection using random forests (R CODE)

```

**Random Forests****

FD <- read.csv("Traj_Clust.csv", header=TRUE)

FD$Class <- as.factor(FD$Class)

rf <- randomForest(formula = Class ~ .,
data = FD,nodesize=0.05, proximity = TRUE,ntree=500)

rfProx <- as.dist(rf$proximity[1:300,1:300])

rfDist <- as.dist(1-rfProx)

hc <- hclust(rfDist, "ave")

plot (hc, hang = -1, labels = FALSE)

rect.hclust(hc, k=3, border="red")

groups <- cutree (hc, k = 3)

plot(outlier(rf$proximity[1:42,1:42]),type="h",col=c("blue"),
main="Outlier Detection using Random Forest",
xlab="Data Points",
ylab="Breiman's Outlier Measure")

FD_matrix_2 <- data.matrix(FD[202:220,])

FD_heatmap <- heatmap(FD_matrix, Rowv=NA, Colv=NA,
col = cm.colors(1256), scale="column",margins=c(5,10))

FD_heatmap <- heatmap(FD_all, Rowv=NA,
Colv=NA,scale="column",margins=c(5,10))

```

```

FD.dist <-dist(FD_matrix)

mds <- isoMDS(FD.dist, k=2)

mds <- isoMDS(cfDist, k=2)

plot(mds$points[,1],mds$points[,2],
xlab='Dimension1',ylab='Dimension2', col = "red")

****Conditional Inference Forests****

cf <- cforest(Class ~ ., data = FD)

o <- outlier(proximity(cf)[1:207,1:207])

plot(o, type="o",col=c("brown"),
main="Outlier Detection using Conditional Forest",
xlab="TrajectoryID",ylab="Breiman's Outlier Score")

```

## REFERENCES

- [1] M. Buchin, A. Driemel, M. van Kreveld, and V. Sacristan, “An algorithmic framework for segmenting trajectories based on spatio-temporal criteria,” in *Proc. 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*. ACM, 2010, pp. 202–211.
- [2] X. Li, J. Han, S. Kim, and H. Gonzalez, “Roam: Rule- and motif-based anomaly detection in massive moving object data sets,” in *In Proceedings of 7th SIAM International Conference on Data Mining*, 2007.
- [3] J. Kang and H.-S. Yong, “Spatio-temporal discretization for sequential pattern mining,” in *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, ser. ICUIMC '08. New York, NY, USA: ACM, 2008, pp. 218–224. [Online]. Available: <http://doi.acm.org/10.1145/1352793.1352840>
- [4] M. Vlachos, D. Gunopulos, and G. Kollios, “Discovering similar multidimensional trajectories,” in *ICDE'02*, 2002, pp. 673–684.
- [5] S. Gaffney and P. Smyth, “Trajectory clustering with mixtures of regression models,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 1999, pp. 63–72.
- [6] J. gil Lee and J. Han, “Trajectory clustering: A partition-and-group framework,” in *In SIGMOD*, 2007, pp. 593–604.
- [7] F. I. Bashir, S. Member, A. A. Khokhar, S. Member, and S. Member, “Object trajectory-based activity classification and recognition using hidden markov models,” *IEEE Trans. Image Process*, vol. 2005.
- [8] J. Owens and A. Hunter, “Application of the self-organizing map to trajectory classification,” in *Proceedings of the Third IEEE International Workshop on Visual Surveillance (VS'2000)*, ser. VS '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 77–. [Online]. Available: <http://dl.acm.org/citation.cfm?id=832293.836186>

- [9] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "Traiclass: trajectory classification using hierarchical region-based and trajectory-based clustering," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 1081–1094, Aug. 2008. [Online]. Available: <http://dx.doi.org/10.1145/1453856.1453972>
- [10] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework." in *ICDE*, G. Alonso, J. A. Blakeley, and A. L. P. Chen, Eds. IEEE, 2008, pp. 140–149. [Online]. Available: <http://dblp.uni-trier.de/db/conf/icde/icde2008.html#LeeHL08>
- [11] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye, "Mining periodic behaviors for moving objects," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '10. New York, NY, USA: ACM, 2010, pp. 1099–1108. [Online]. Available: <http://doi.acm.org/10.1145/1835804.1835942>
- [12] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, Apr. 2001. [Online]. Available: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0130226165>
- [13] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. F. de Macedo, B. Moelans, and A. Vaisman, "A model for enriching trajectories with semantic geographical information," in *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, ser. GIS '07. New York, NY, USA: ACM, 2007, pp. 22:1–22:8. [Online]. Available: <http://doi.acm.org/10.1145/1341012.1341041>
- [14] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares, "A clustering-based approach for discovering interesting places in trajectories," in *Proceedings of the 2008 ACM symposium on Applied computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 863–868. [Online]. Available: <http://doi.acm.org/10.1145/1363686.1363886>
- [15] Z. Yan, C. Parent, S. Spaccapietra, and D. Chakraborty, "A hybrid model and computing platform for spatio-semantic trajectories," in *Proceedings of the 7th international conference on The Semantic Web: research and Applications - Volume Part I*, ser. ESWC'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 60–75. [Online]. Available: [http://dx.doi.org/10.1007/978-3-642-13486-9\\_5](http://dx.doi.org/10.1007/978-3-642-13486-9_5)
- [16] M.-J. Lee and C.-W. Chung, "A user similarity calculation based on the location for social network services," in *Proceedings of the 16th international conference*

*on Database systems for advanced applications - Volume Part I*, ser. DASFAA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 38–52. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1997305.1997313>

- [17] J. J.-C. Ying, E. H.-C. Lu, W.-C. Lee, T.-C. Weng, and V. S. Tseng, “Mining user similarity from semantic trajectories,” in *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, ser. LBSN '10. New York, NY, USA: ACM, 2010, pp. 19–26. [Online]. Available: <http://doi.acm.org/10.1145/1867699.1867703>
- [18] W. M. Bannister, “Associative and sequential classification with adaptive constrained regression methods,” Ph.D. dissertation, Tempe, AZ, USA, 2007, aAI3287911.
- [19] Wikipedia, “Control chart — wikipedia, the free encyclopedia,” 2007. [Online]. Available: <http://en.wikipedia.org/w/index.php?title=File:ControlChart.svg&page=1>
- [20] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. de Macedo, F. Porto, and C. Vangenot, “A conceptual view on trajectories,” *Data Knowl. Eng.*, vol. 65, no. 1, pp. 126–146, Apr. 2008. [Online]. Available: <http://dx.doi.org/10.1016/j.datak.2007.10.008>
- [21] S. P. Lloyd, “Least squares quantization in pcm,” *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.
- [22] T. Zhang, R. Ramakrishnan, and M. Livny, “Birch: An efficient data clustering method for very large databases,” in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, H. V. Jagadish and I. S. Mumick, Eds. ACM Press, 1996, pp. 103–114.
- [23] M. Ester, H. Peter Kriegel, J. S., and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.” AAAI Press, 1996, pp. 226–231.
- [24] M. Ankerst, M. M. Breunig, H. Peter Kriegel, and J. Sander, “Optics: Ordering points to identify the clustering structure.” ACM Press, 1999, pp. 49–60.
- [25] W. Wang, J. Yang, and R. Muntz, “Sting: A statistical information grid approach to spatial data mining,” 1997.
- [26] H. Sakoe and S. Chiba, “Readings in speech recognition,” A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. Dynamic programming algorithm optimization for spoken word recognition, pp. 159–165. [Online]. Available: <http://dl.acm.org/citation.cfm?id=108235.108244>

- [27] T. K. Moon, “The expectation-maximization algorithm,” *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, Nov. 1996. [Online]. Available: <http://dx.doi.org/10.1109/79.543975>
- [28] P. D. Grnwald, *The Minimum Description Length Principle*, ser. MIT Press Books. The MIT Press, 2007, vol. 1, no. 0262072815. [Online]. Available: <http://ideas.repec.org/b/mtp/titles/0262072815.html>
- [29] L. Rabiner and B. Juang, “An introduction to hidden Markov models,” *ASSP Magazine, IEEE*, vol. 3, no. 1, pp. 4–16, Apr. 2003. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs\\\_all.jsp?arnumber=1165342](http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=1165342)
- [30] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, Oct. 2002. [Online]. Available: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0387954422>
- [31] R. Fraile and S. J. Maybank, “Vehicle Trajectory Approximation and Classification,” in *British Machine Vision Conference*, P. H. Lewis and M. S. Nixon, Eds., 1998.
- [32] G. D. Forney, “The viterbi algorithm,” *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973. [Online]. Available: <http://dx.doi.org/10.1109/PROC.1973.9030>
- [33] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., *Self-Organizing Maps*, 3rd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.
- [34] I. F. Sbalzarinii, J. Theriot, and P. Koumoutsakos, “Machine learning for biological trajectory classification applications,” in *Proc. 2002 Summer Program, Center for Turbulence Research*, 2002, pp. 305–316.
- [35] V. Barnett and T. Lewis, *Outliers in Statistical Data*, ser. Wiley Series in Probability & Statistics. Wiley, Apr. 1994.
- [36] E. M. Knorr and R. T. Ng, “Algorithms for Mining Distance-Based Outliers in Large Datasets,” in *Proc. 24th Int. Conf. Very Large Data Bases, VLDB, FebApr–FebJul~ 1998*, pp. 392–403. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.8026>
- [37] E. M. Knorr, R. T. Ng, and V. Tucakov, “Distance-Based Outliers: Algorithms and Applications,” *VLDB Journal: Very Large Data Bases*, vol. 8, no. 3–4, pp. 237–253, 2000. [Online]. Available: <http://citeseer.ist.psu.edu/knorr00distancebased.html>

- [38] S. Ramaswamy, R. Rastogi, and K. Shim, “Efficient algorithms for mining outliers from large data sets,” 2000, pp. 427–438. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.947>
- [39] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying Density-Based Local Outliers,” in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, ser. SIGMOD ’00, vol. 29, no. 2. New York, NY, USA: ACM, Jun. 2000, pp. 93–104. [Online]. Available: <http://dx.doi.org/10.1145/335191.335388>
- [40] C. C. Aggarwal and P. S. Yu, “Outlier detection for high dimensional data,” in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, ser. SIGMOD ’01. New York, NY, USA: ACM, 2001, pp. 37–46. [Online]. Available: <http://doi.acm.org/10.1145/375663.375668>
- [41] P. Kalnis, N. Mamoulis, and S. Bakiras, “On discovering moving clusters in spatio-temporal data,” in *Proceedings of the 9th international conference on Advances in Spatial and Temporal Databases*, ser. SSTD’05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 364–381. [Online]. Available: [http://dx.doi.org/10.1007/11535331\\_21](http://dx.doi.org/10.1007/11535331_21)
- [42] J. Gudmundsson and M. van Kreveld, “Computing longest duration flocks in trajectory data,” in *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, ser. GIS ’06. New York, NY, USA: ACM, 2006, pp. 35–42. [Online]. Available: <http://doi.acm.org/10.1145/1183471.1183479>
- [43] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen, “Discovery of convoys in trajectory databases,” *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 1068–1080, Aug. 2008. [Online]. Available: <http://dx.doi.org/10.1145/1453856.1453971>
- [44] Z. Li, B. Ding, J. Han, and R. Kays, “Swarm: mining relaxed temporal moving object clusters,” *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 723–734, Sep. 2010. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1920841.1920934>
- [45] S. Bar-David, I. Bar-David, P. C. Cross, S. J. Ryan, C. U. Knechtel, and W. M. Getz.
- [46] H. Jeung, H. T. Shen, and X. Zhou, “Mining trajectory patterns using hidden markov models,” in *DaWaK*, ser. Lecture Notes in Computer Science, I. Y. Song, J. Eder, and T. M. Nguyen, Eds., vol. 4654. Springer, 2007, pp. 470–480. [Online]. Available: <http://dblp.uni-trier.de/db/conf/dawak/dawak2007.html#JeungSZ07>



- [47] F. I. Bashir, A. A. Khokhar, and D. Schonfeld, "Object Trajectory-Based Activity Classification and Recognition Using Hidden Markov Models," *Image Processing, IEEE Transactions on*, vol. 16, no. 7, pp. 1912–1919, 2007. [Online]. Available: <http://dx.doi.org/10.1109/TIP.2007.898960>
- [48] F. Legland and L. Mevel, "Fault detection in hidden Markov models : a local asymptotic approach," in *39th IEEE Conference on Decision and Control*, vol. 5, 2000, pp. 4686–4690 vol.5. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs/\\_all.jsp?arnumber=914667](http://ieeexplore.ieee.org/xpls/abs/_all.jsp?arnumber=914667)
- [49] L. Gerencseis and G. Molnar-Saska, "Change detection of hidden Markov models," in *Conference on Decision and Control*, vol. 2, 2004.
- [50] L. Gerencseis and C. Prosdocimi, "Change detection for hidden Markov models," 2005.
- [51] D. V. Hinkley, "Inference about the change-point from cumulative sum tests," *Biometrika*, vol. 58, no. 3, pp. 509–523, Dec. 1971. [Online]. Available: <http://dx.doi.org/10.1093/biomet/58.3.509>
- [52] C.-D. Fuh, "Asymptotic operating characteristics of an optimal change point detection in hidden markov models," *The Annals of Statistics*, vol. 32, no. 5, pp. 2305–2339, 2004. [Online]. Available: <http://www.jstor.org/stable/3448573>
- [53] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of 2nd International Conference on Knowledge Discovery and*, 1996, pp. 226–231.
- [54] Z. Yan, "Towards semantic trajectory data analysis: A conceptual and computational approach." in *VLDB PhD Workshop*, P. Rigaux and P. Senellart, Eds. VLDB Endowment, 2009. [Online]. Available: <http://dblp.uni-trier.de/db/conf/vldb/vldb2009phd.html#Yan09>
- [55] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer, "Semitri: a framework for semantic annotation of heterogeneous trajectories." in *EDBT*, A. Ailamaki, S. Amer-Yahia, J. M. Patel, T. Risch, P. Senellart, and J. Stoyanovich, Eds. ACM, 2011, pp. 259–270. [Online]. Available: <http://dblp.uni-trier.de/db/conf/edbt/edbt2011.html#YanCPSA11>
- [56] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference*

- on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645920.672836>
- [57] V. Bogorny, B. Kuijpers, and L. O. Alvares, “A spatio-temporal data mining query language for moving object trajectories,” Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil, 2008.
- [58] R. Agrawal and R. Srikant, “Mining sequential patterns,” in *Proceedings of the Eleventh International Conference on Data Engineering*, ser. ICDE '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 3–14. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645480.655281>
- [59] “Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth,” in *Proceedings of the 17th International Conference on Data Engineering*, ser. ICDE '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 215–. [Online]. Available: <http://dl.acm.org/citation.cfm?id=876881.879716>
- [60] S. E. Robertson and K. Sparck Jones, “Document retrieval systems,” P. Willett, Ed. London, UK, UK: Taylor Graham Publishing, 1988, ch. Relevance weighting of search terms, pp. 143–160. [Online]. Available: <http://dl.acm.org/citation.cfm?id=106765.106783>
- [61] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, “Mining user similarity based on location history,” in *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, ser. GIS '08. New York, NY, USA: ACM, 2008, pp. 34:1–34:10. [Online]. Available: <http://doi.acm.org/10.1145/1463434.1463477>
- [62] Kosala and Blockeel, “Web Mining Research: A Survey,” *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM*, vol. 2, 2000. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.5720>
- [63] W. Lee and S. J. Stolfo, “Data mining approaches for intrusion detection,” in *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*, ser. SSYM'98. Berkeley, CA, USA: USENIX Association, 1998, pp. 6–6. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1267549.1267555>
- [64] M. J. Zaki, “Mining data in bioinformatics,” in *Handbook of Data Mining*, N. Ye, Ed. Lawrence Earlbaum Associates, 2003, pp. 573–596.

- [65] R. Srikant and R. Agrawal, “Mining sequential patterns: Generalizations and performance improvements,” in *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, ser. EDBT '96. London, UK, UK: Springer-Verlag, 1996, pp. 3–17. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645337.650382>
- [66] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, “Freespan: frequent pattern-projected sequential pattern mining,” in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '00. New York, NY, USA: ACM, 2000, pp. 355–359. [Online]. Available: <http://doi.acm.org/10.1145/347090.347167>
- [67] M. J. Zaki, “Spade: An efficient algorithm for mining frequent sequences,” *Mach. Learn.*, vol. 42, no. 1-2, pp. 31–60, Jan. 2001. [Online]. Available: <http://dx.doi.org/10.1023/A:1007652502315>
- [68] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal, “Multi-dimensional sequential pattern mining,” in *Proceedings of the tenth international conference on Information and knowledge management (CIKM '01)*. New York, NY, USA: ACM, 2001, pp. 81–88. [Online]. Available: <http://dx.doi.org/10.1145/502585.502600>
- [69] Y. Chen, “Discovering time-interval sequential patterns in sequence databases,” *Expert Systems with Applications*, vol. 25, no. 3, pp. 343–354, Oct. 2003. [Online]. Available: [http://dx.doi.org/10.1016/S0957-4174\(03\)00075-7](http://dx.doi.org/10.1016/S0957-4174(03)00075-7)
- [70] L. Vincelas, J.-E. Symphor, A. Mancheron, and P. Poncelet, “Spams: A novel incremental approach for sequential pattern mining in data streams,” in *EGC (best of volume)*, 2009, pp. 201–216.
- [71] L. F. Mendes, B. Ding, and J. Han, “Stream sequential pattern mining with precise error bounds,” in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 941–946. [Online]. Available: <http://dx.doi.org/10.1109/ICDM.2008.154>
- [72] C.-Y. Tsai and Y.-C. Shieh, “A change detection method for sequential patterns,” *Decis. Support Syst.*, vol. 46, no. 2, pp. 501–511, Jan. 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.dss.2008.09.003>
- [73] L. Allison, “Dynamic Programming Algorithm (DPA) for Edit-Distance,” Department of Computer Science, UWA, 1984.

- [74] P.-A. Laur, J.-E. Symphor, R. Nock, and P. Poncelet, “Statistical supports for mining sequential patterns and improving the incremental update process on data streams,” *Intell. Data Anal.*, vol. 11, no. 1, pp. 29–47, Jan. 2007. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1367489.1367492>
- [75] S. Jacquemont, F. Jacquenet, and M. Sebban, “Mining probabilistic automata: a statistical view of sequential pattern mining,” *Mach. Learn.*, vol. 75, no. 1, pp. 91–127, Apr. 2009. [Online]. Available: <http://dx.doi.org/10.1007/s10994-008-5098-y>
- [76] G. Dong and J. Li, “Efficient mining of emerging patterns: discovering trends and differences,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD ’99. New York, NY, USA: ACM, 1999, pp. 43–52. [Online]. Available: <http://doi.acm.org/10.1145/312129.312191>
- [77] A.-L. Boulesteix, G. Tutz, and K. Strimmer, “A cart-based approach to discover emerging patterns in microarray data,” *Bioinformatics*, vol. 19, no. 18, pp. 2465–2472, 2003.
- [78] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, ser. Statistics/Probability Series. Belmont, California, U.S.A.: Wadsworth Publishing Company, 1984.
- [79] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals Eugen.*, vol. 7, pp. 179–188, 1936.
- [80] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: <http://dx.doi.org/10.1023/A:1010933404324>
- [81] A. Liaw and M. Wiener, “Classification and regression by randomforest.” *R News: The Newsletter of the R Project*, vol. 2, no. 3, pp. 18–22, 2002. [Online]. Available: <http://cran.r-project.org/doc/Rnews/>
- [82] H. Deng, G. Runger, and E. Tuv, “Bias of importance measures for multi-valued attributes and solutions,” in *Proceedings of the 21st international conference on Artificial neural networks - Volume Part II*, ser. ICANN’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 293–300. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2029604.2029642>
- [83] T. Hothorn, K. Hornik, A. Zeileis, W. Wien, and W. Wien, “Unbiased recursive partitioning: A conditional inference framework,” *Journal of Computational*

- and Graphical Statistics*, vol. 2006, pp. 651–674, 2005. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.92.9930>
- [84] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 413–422. [Online]. Available: <http://dx.doi.org/10.1109/ICDM.2008.17>
- [85] D. C. Montgomery, *Statistical quality control*. Wiley Hoboken, N.J., 2009. [Online]. Available: <http://opac.bibliothek.uni-kassel.de/DB=1/PPN?PPN=21627463X>
- [86] Wikipedia, “Baum-welch algorithm — wikipedia, the free encyclopedia.” [Online]. Available: [http://en.wikipedia.org/wiki/BaumWelch\\_algorithm](http://en.wikipedia.org/wiki/BaumWelch_algorithm)
- [87] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, 2nd ed. Springer, Mar. 2002. [Online]. Available: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387953515>
- [88] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, “CRAW-DAD data set epfl/mobility (v. 2009-02-24),” Downloaded from <http://crawdad.cs.dartmouth.edu/epfl/mobility>, Feb. 2009.
- [89] J. Hershberger and J. Snoeyink, “Speeding up the douglas-peucker line-simplification algorithm,” Vancouver, BC, Canada, Canada, Tech. Rep., 1992.
- [90] M. Plantevit and B. Crémilleux, “Condensed representation of sequential patterns according to frequency-based measures,” in *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, ser. IDA '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 155–166. [Online]. Available: [http://dx.doi.org/10.1007/978-3-642-03915-7\\_14](http://dx.doi.org/10.1007/978-3-642-03915-7_14)
- [91] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.
- [92] Wikipedia, “Curse of dimensionality — wikipedia, the free encyclopedia.” [Online]. Available: [http://en.wikipedia.org/wiki/Curse\\_of\\_dimensionality](http://en.wikipedia.org/wiki/Curse_of_dimensionality)
- [93] N. Stepenosky, R. Polikar, J. Kounios, and C. M. Clark, “Ensemble techniques with weighted combination rules for early diagnosis of alzheimer’s disease,” in *IJCNN'06*, 2006, pp. 1935–1941.