Time Series Online Empirical Bayesian Kernel Density Segmentation: Applications in Real Time Activity Recognition Using Smartphone Accelerometer

Shuang Na
University of South Florida, sna@mail.usf.edu

Time Series Online Empirical Bayesian Kernel Density Segmentation

Applications in Real Time Activity Recognition Using Smartphone Accelerometer

by

Shuang Na

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Mathematics and Statistics
College of Arts and Sciences
University of South Florida

Date of Approval:
June 20, 2017

Keywords: Online Empirical Bayesian, time series segmentation, kernel density,
Random Forests, human pattern recognition

Dedication

This doctoral dissertation is dedicated to my parents.

Acknowledgments

First of all, I would like to express my deepest gratitude to my advisor, Professor Kandethody M. Ramachandran for sharing his wealth of knowledge and wisdom with me. Not only his guidance of my Ph.D. dissertation, but also encouragement and advice from him. I am also obliged and thankful to my co-advisor Professor Ming Ji for his valuable suggestions and useful discussions.

I am also truly grateful to the other members of my research committee: Prof. Chris P. Tsokos, Prof. Yicheng Tu. They gave me many nice suggestions to improve my study. I also would like to thank other professors who taught me probability and statistics: Prof. Gangaram S. Ladde, Prof. Lu Lu, and Prof. Mingyang Li. I learned and obtained a lot from them about Statistical knowledge. I would like to thank the Mathematics and Statistics Department at the University of South Florida for their financial support.

I have no words to express my gratitude to my family for their endless support and love. I would like to acknowledge the sacrifices made by my father and mother for my better education and upbringing.

Finally, I am so very thankful to my friends Dr. Xing Wang, Jason Burgess and all the other fellow graduate students for their support during my Ph.D. time.

Table of Contents

List of Tables iii

List of Figures v

Abstract vi

1 Introduction 1

- 1.1 Motivation 1
- 1.2 Online Empirical Bayesian Kernel Segmentation Algorithm 4
- 1.3 Two-layer Classification and Human Pattern Recognition System . . 5
- 1.4 Structure of Dissertation 6

2 Previous Work Discussion 8

- 2.1 Hidden Markov Model 8
  - 2.1.1 Viterbi Algorithm 9
  - 2.1.2 Forward Probability 10
- 2.2 Estimation Result 10
- 2.3 Reason of Deficiency 12

3 Combine online Empirical Bayesian with Online Kernel Estimation density 16

- 3.1 Related Work 16
  - 3.1.1 Online Segmentation 16
  - 3.1.2 Kernel Density Estimation 21
  - 3.1.3 Unscented Transformation 26
- 3.2 Online Empirical Bayesian Kernel Segmentation 30
  - 3.2.1 Empirical Bayesian Online Segmentation 31
  - 3.2.2 Adjusted Online Empirical Bayesian Method 36
- 3.3 Online Multivariate Kernel Estimation 39

3.3.1 Bandwidth Selection 40
3.3.2 Compression of the Sample Model 46
3.3.3 Online Kernel Density Estimation 51
3.4 Contribution 55

4 Experiment Result of Online Empirical Bayesian Kernel Segmentation 57
4.1 Related Work 57
4.1.1 Metropolis Hastings Algorithm 57
4.1.2 Adaptive MCMC 60
4.1.3 Combining Delying Rejection (DR) and Adaptive MCMC . . 61
4.2 Simulated Experiment Result 62
4.3 Empirical Observation Experiment Result 67
4.4 Comparing OBKS with HMM 71
4.5 Contribution 72

5 Two-layer Classification and Experiment Result 74
5.1 Classification Algorithm 74
5.1.1 k-Nearest Neighbor 74
5.1.2 Decision Tree 75
5.1.3 Naive Bayes 78
5.1.4 Neural Network 79
5.2 Feature Extraction and Feature Selection 80
5.3 Feature Selection for Human Pattern Observation 81
5.4 Classification Comparison 83
5.5 Contribution 86

6 Human Pattern Recognition System 90
6.1 Human Pattern Recognition System 90
6.2 Empirical Experiment 93
6.3 Contribution 93

7 Conclusion and Future Work 95

References 98

List of Tables

3.1 The classical kernel functions 23

4.1 Detection accuracy of OB and OBKS on four different simulated data 65
4.2 MLE and MSE of OB and OKDE on four different simulated data . . 67
4.3 Online Empirical Bayesian Kernel Segmentation Confusion Matrix . . 69
4.4 SWAB overall accuracy 70
4.5 Online Empirical Bayesian Kernel Segmentation Confusion Matrix . . 71
4.6 Detection delay using OBKS method 71

5.1 Variable importance 88
5.2 Overall accuracy of six activities using four classification within one layer and two-layer approach 88
5.3 Experimental time consuming for each classification method 89

6.1 Online two-layer Classification Confusion Matrix 93

List of Figures

2.1 State estimation by Viterbi algorithm 11

2.2 State estimation by Forward probability 12

2.3 The probability function of six activities with respect to x-axis 13

2.4 The probability function of six activities with respect to y-axis 14

2.5 The probability function of six activities with respect to z-axis 15

3.1 Univariate Kernel Density Estimation at $\mu = 1$ and $\sigma = 1$ 22

3.2 “Run length” r_t path: Solid line represents $r_{t+1} = r_t + 1$, Dashed lines represent $r_{t+1} = 0$ and $\mathbf{x}_{t+1}$ is change point 32

3.3 Online Empirical Bayesian Kernel Segmentation 38

3.4 Sample distribution $p_s(\mathbf{x})$ with associated detail distributions $q_i(\mathbf{x})$. 39

3.5 Compression of four-component sample distribution $p_s(\mathbf{x})$ (1) into three-component sample distribution (2) 46

3.6 Online Multivariate Kernel Density Estimation 52

3.7 OBKS result with “run length” r_t 55

4.1 Detection accuracy of OB and OKDE on low covariance and low correlation bi-normal simulation data 65

4.2 Detection accuracy of OB and OKDE on low covariance and low correlation bi-normal simulation data 66

4.3 Detection accuracy of OB and OKDE on high covariance and low correlation bi-normal simulation data 66

4.4 Detection accuracy of OB and OKDE on high covariance and high correlation bi-normal simulation data 66

4.5 Acceleration plot of six activities 69

4.6 Acceleration plot of six activities 72

5.1 Decision tree partition 76

5.2 Structure of Bagging 77

5.3 Structure of Boosting 78

5.4 Structure of ANN 80
5.5 Two-layer Classification 84
5.6 Accuracy of six activities using four classification methods 86

6.1 Online Activity Recognition Process 92

Abstract

Time series analysis has been explored by the researchers in many areas such, as statistical research, engineering applications, medical analysis, and finance study. To represent the data more efficiently, the mining process is supported by time series segmentation. Time series segmentation algorithm looks for the change points between two different patterns and develops a suitable model, depending on the data observed in such segment. Based on the issue of limited computing and storage capability, it is necessary to consider an adaptive and incremental online segmentation method. In this study, we propose an Online Empirical Bayesian Kernel Segmentation (OBKS), which combines Online Multivariate Kernel Density Estimation (OMKDE) and Online Empirical Bayesian Segmentation (OBS) algorithm. This innovative method considers Online Multivariate Kernel density as a predictive distribution derived by Online Empirical Bayesian segmentation instead of using posterior predictive distribution as a predictive distribution. The benefit of Online Multivariate Kernel Density Estimation is that it does not require the assumption of a pre-defined prior function, which makes the OMKDE more adaptive and adjustable than the posterior predictive distribution.

Human Activity Recognition (HAR) by smartphones with embedded sensors is a modern time series application applied in many areas, such as therapeutic applications and sensors of cars. The important procedures related to the HAR problem include classification, clustering, feature extraction, dimension reduction, and segmentation. Segmentation as the first step of HAR analysis attempts to represent the time interval more effectively and efficiently. The traditional segmentation method of HAR is to partition the time series into short and fixed length segments. However, these segments might not be long enough to capture the sufficient information for the entire activity time interval. In this research, we segment the observations of a whole activity as a whole interval using the Online Empirical Bayesian Kernel Segmentation algorithm as the first step. The smartphone with built-in accelerometer generates observations of these activities.

Based on the segmenting result, we introduce a two-layer random forest classification method. The first layer is used to identify the main group; the second layer is designed to analyze the subgroup from each core group. We evaluate the performance of our method based on six activities: sitting, standing, lying, walking, walking_upstairs, and walking_downstairs on 30 volunteers. If we want to create a machine that can detect walking_upstairs and walking_downstairs automatically, it requires more information and more detail that can generate more complicated features, since these two activities are very similar. Continuously, considering the real-time Activity Recognition application on the smartphones by the embedded accelerometers, the first layer classifies the activities as static and dynamic activities, the second layer classifies each main group into the sub-classes, depending on the first layer result. For the data collected, we get an overall accuracy of 91.4% based on the six activities and an overall accuracy of 100% based only on the dynamic activity (walking, walking_upstairs, walking_downstairs) and the static activity (sitting, standing, lying).

1 Introduction

1.1 Motivation

Decades of development in technology have led to research and academic equipment that can generate a significant amount of data with high speed and high dimensions. The size of data is continuously growing without limitations. For example, the set-down devices on the surface of the ocean are set every 150 square miles of ocean to detect the temperature and the amplitude of a wave and keep sending the data back at a rate of ten times per second, which result in 3.5 terabytes data per day [47]. This equipment that is not a part of everyday life are not the only machines that generate real-time data, but many common devices that are closely bound up in our daily lives also automatically create data every second, such as smartphones and computers. Generally, real-time data is continuous, large in data size, high speed, and high dimensionality. It is more challenging to analyze this type of data, limited by the computing speed and storage of the current computer. Furthermore, large databases are involved in many fields, such as finance, biology, and engineering. A flexible method that can apply to different areas needs to be developed. Considering the specificities of big data and the intersection of different application areas, we come up with online data mining knowledge, which is a subfield of data mining. Data mining is the computational process of discovering patterns in big data related to these approaches at the intersection of statistics, machine learning, artificial intelligence, and database systems [14].

The target values of many online data mining applications such as human activity identification ([42],[51],[74]), voice recognition ([26],[27]) and sign language ([72],[73]) may change over time, such as the statistical properties of the probability function or the fitting model. These studies led to a collection of approaches for analyzing time series observation called time series mining. At this point, the aim of time series mining is to extract information from the observations of a particular period ([20],[59]), which considers all data points in an interval as a whole instead of the individual point. These important problems of mining a time series scenario are dimension reduction, time series classification, time series clustering, frequency counting, and time series segmentation ([3], [23], [24]), which supports the empirical analysis. For instance, time series mining can be used for weather prediction, stock price prediction, fraud detection, and health protection. Given computer limitations like computing time, memory size, and computing speed, representing the observations more effectively and more efficiently is the first step of mining time series data. One of the mining methods, time series segmentation algorithm, is applied to the input observations and returns a function representation, such as linear regression model, probability density function, and wavelet representation. Motivated by Empirical Bayesian Online Change Detection [2], which presents an online algorithm to detect the change points rather than using an off-line and retrospective segmentation method. However, the Empirical Bayesian method requires the assumptions of prior distribution and likelihood function, and it is not convenient to find a general pre-defined prior distribution and a likelihood function for these unknown and shifty observations. Online Multivariate Kernel Density Estimation method is more flexible than a pre-defined density function, and it can be fit into any type of data. Therefore, combining these two techniques yields Online Empirical Bayesian Kernel Segmentation method.

It has been estimated that there would be 6.1 billion smartphone users by 2020, which is about 70% of the population worldwide [1]. The smartphone is not just a communication device, it is also a powerful tool with a variety of software features, such as photography, radio, calculator, games, and navigation. In addition, the smartphone has internally installed hardware such as a camera, accelerometer, microphone, magnetic compasses, and GPS sensors. The Human Activity Recognition (HAR) problem is about monitoring human activities continuously (i.e. sitting, walking, running, etc.) using a smartphone with built-in sensors (i.e. GPS and accelerometer). The user can carry a smartphone every day and everywhere as the smartphone tracks their activities. For instance, some users are concerned about how many calories are burned every day through exercise or detecting falls. Compared with other types of wearable sensors, it is more convenient and the user is likely to carry the smartphone every day due to its small size and multi-functionality.

The first experiment of HAR is discussed in [21] in the late '90s. During the past 27 years, there were several approaches to improve the recognition process by placing several sensors set on multiple locations of a user's body ([8],[15],[19]). During Activity Recognition from the User-Annotated Acceleration data [8], Ling Bao and Stephen Intille indicate that the multiple accelerometers distributed at different locations on the body have more effective recognition ability. In addition, by applying complicated methods such as statistical learning and machine learning, they also improved the identification process ([6], [5], [13], [76]). An overview of the HAR research using wearable sensors is discussed in [46], which compares the HAR system, sensors, recognition method, and evaluation systems. The researchers claim that it was difficult to define the best detecting procedure because of considering different tasks, sampling rates, algorithms, computing speed, and the evaluation methods. Nevertheless, wearing more sensors gave better results in general. However, it is unreasonable for users to wear multiple pieces of equipment every day because of the expense, complexity, and inconvenience. The smartphone is a much used electronic product that has been widely applied for activity recognition with built-in sensors (e.g. [5],[32],[44],[64], [70]). Therefore, the smartphone-based HAR became a very important research field.

1.2 Online Empirical Bayesian Kernel Segmentation Algorithm

As we mentioned above, the time series segmentation is the first step of time series mining, followed by a classification algorithm. In this study, we first propose an Online Empirical Bayesian Kernel Segmentation (OBKS) method that modified Online

Empirical Bayesian Change Point Detection (OBCD)[?], the OKBS regards Online Kernel Density Estimation (OKDE) as a predictive function instead of the Empirical Bayesian predictive function. One of the advantages of the Empirical Bayesian approach is it considers all uncertainty as a prior distribution. Another of the advantages is that it does not require the asymptotic assumptions about test statistics that are presented in the frequentist algorithms, which can be problematic in the situations where the parametric models considered are restricted to a finite, possibly small time interval [16]. However, it is challenging to choose a general prior function that can be used for a multiple of cases. Also, it might cost more time to search the optimal parameters if the initial parameters are far away from the true parameters. In addition, there are two "prior functions" in [?], which results in a higher demand to select the correct prior distributions. The details will be given in Chapter 3. Meanwhile, the multivariate kernel density function replaces the posterior predictive function that is used to generate Empirical Bayesian predictive probability. There are a few articles that discuss Online Multivariate Kernel Density Estimation algorithm ([41],[40],[45]).

1.3 Two-layer Classification and Human Pattern Recognition System

After the OBKS method detecting the change points on a time interval, we extract the essential features (maximum, average, frequency, etc.) from each segment that are supposed to represent the characteristics of a segment. The features of every segment contain more information than the features of a short time interval that are generated by a fixed length time interval (100 observations). For the training procedure, different classification layers consider using different features. The first layer can detect the main groups; the second layer is to check these main groups further separately through more complicated features that can evaluate the sub-groups. In this research, we present a real-time Human Activity Recognition algorithm that combines Online Empirical Bayesian Kernel Segmentation (OBKS) algorithm and two-layer Random Forest classification. This algorithm can automatically identify the patterns at any particular time without the user intervening, such that the user does not need to mark the start time and the end time for every activity. For this application, the aim of the first layer classification is to distinguish the dynamic activities (walking, walking_upstairs and walking_downstairs) versus the static activities (sitting, standing, and lying). The main feature used in the first layer classifier is amplitude defined as $\sqrt{x_{i,1}^2 + x_{i,2}^2 + ... + x_{i,d}^2}$, x_i is i^{th} point with d-dimension. During the second layer classification processing, there are two separate classifiers: classifier A and classifier B that classify the different sub-groups. If we get a static activity on the first layer for a specified time interval, then we go through classifier A at the second layer to further classify the observations into sitting, standing, or lying; otherwise, we go through classifier B at the second layer to further identify the observations as walking, walking_upstairs or walking_downstairs.

1.4 Structure of Dissertation

This dissertation is organized as follows. Chapter 1 summarizes the proposed method and the experiments in this study. Chapter 2 discusses the previous work of using the Hidden Markov Model to analyze the time series pattern. Briefly, all background knowledge that relates to our task is introduced in the first subsection from Chapter 3 to Chapter 5. Chapter 3 proposes an Online Empirical Bayesian Kernel Segmentation method built on the Online Empirical Bayesian approach and Online Multivariate Kernel Density, which can detect the time series change points. The experimental analysis of this novel segmentation method is explained in Chapter 4, which compares the experimental result with the result of the Sliding Window Bottom Up segmentation algorithm. Chapter 5 discusses classification features selection and the two-layer classification that is applied in identifying human pattern application. Combining the research from Chapter 3 to Chapter 5, the human pattern recognition system is introduced in Chapter 6 and presents the final experiment result. Finally, conclusions and future work are given in Chapter 7.

2 Previous Work Discussion

2.1 Hidden Markov Model

The Hidden Markov Models are known for their application in speech recognition, handwriting identifying, gesture recognition, etc. A Hidden Markov Model (HMM) is a Markov process with state unknown and observation known that is generated by these unknown states. For example, a person gets a disease; what the disease really was, the doctor is not sure, but the symptom truly can be observed. In this case, the disease is regarded as a hidden state, and the symptoms are the observations. HMM can estimate the states through these different types of symptoms.

In a Markov Chain, the states are directly observed, and the state transition probabilities are the only parameters. In the HMM, the observation could be discrete (such as the outcomes of a diced experiment, the characters from a finite alphabet, symptom from a cold), or continuous (such as temperature, voice sample, sensor speed from a car). Each state has a probability distribution providing the possible outcomes at a given moment, which is an emission probability. Therefore, the sequence of observed data generated by an HMM gives hidden information about the sequence of states.

Through the Hidden Markov Model, we estimate the features of six human pattern activities, including sitting, standing, walking_upstairs, walking_downstairs, walking, and jogging by the Expectation Maximization algorithm. This data source is provided by Dr. Tu Yicheng's Lab at University of South Florida, which is generated by iPhone 6 built-in accelerometer with 50Hz. What we get is transmission probability $A = a_{ij}$, prior probability π, mean and variance of observation from each state, μ and σ^2. Let q_t represent the state at time t, $t = 1, 2, ..., T$, and o_t represents the observation at time t. s_i represents the possible state i, $i = 1, 2, ..., k$. k is the total number of types of states. In our sensor data generated with the built-in accelerometer in the smartphone (iPhone), the number of states is 6, $S = s_1, s_2, ..., s_6$. The sensor data is tri-dimensional: x-axis shows the acceleration rate on forth-back direction; y-axis shows the acceleration rate on left-right direction; and z-axis shows the acceleration rate on up-down direction.

2.1.1 Viterbi Algorithm

The traditional method to estimate state for every observation is called the Viterbi algorithm, which testifies which state the observation comes from at time t. This algorithm picks the state with respect to the highest conditional probability of observation given a certain state at time t, that is:

$$\begin{aligned}
&P_1 = maxP(o_1|q)\pi_q \\
&q_1 = argmax_{q \in S} P(o_1|q)\pi_q \\
&P_t = maxP(o_t|q)a_{q_{t-1}q}P(t-1) \\
&q_t = argmax_{q \in S} P(o_t|q)a_{q_{t-1}q}P(t-1) \qquad (2.1.1)
\end{aligned}$$

2.1.2 Forward Probability

Since the Viterbi algorithm just considers one trial associated with the highest probability, it may miss some information for other routines. Also, it is very sensitive to the prior probability. Forward probability is to pick the state with the highest probability at time t given all previous observations, which considers all possible routines that happened before. That is,

$$q_t = argmax_{q \in S} P(q|o_{1:t}) \qquad (2.1.2)$$

2.2 Estimation Result

To justify the methods mentioned above, the following is our result generated by these two methods. In Figure 2.1, the states are estimated by the Viterbi algorithm. Some points seem to belong to multiple states, but a point only comes from a state in fact. In addition, the state jumped from state i to state j at time $t-1$ to t; it is unreasonable for a person to change his/her activity in very short time. For example, the state is changed from walking to walking_upstairs, followed by jogging and sitting.

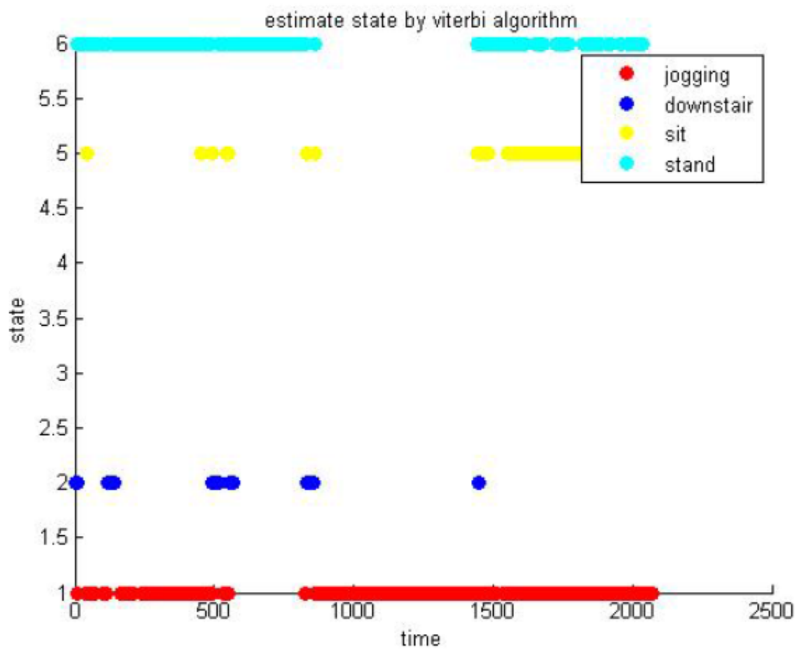


Figure 2.1: State estimation by Viterbi algorithm

In Figure 2.2, the states are estimated by Forwarding probability. Around the first 500 data, jogging and walking_upstairs almost mix, meaning the points generated from jogging contain the points generated by walking_upstairs. The reason could be both patterns have a similar acceleration rate on x-axis and z-axis.

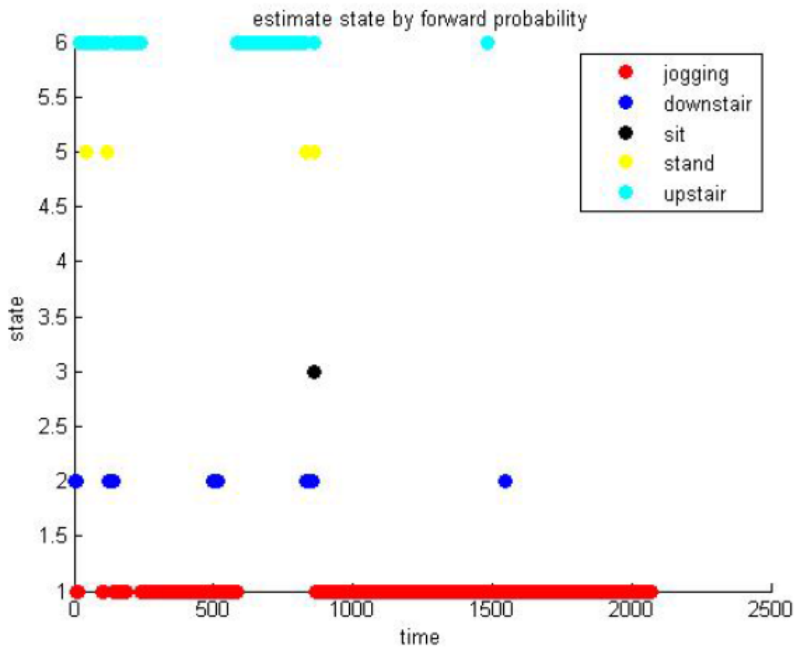


Figure 2.2: State estimation by Forward probability

2.3 Reason of Deficiency

The estimated result is not as good as we expect. The reason could be explained by following the density plot of these six patterns. The density of each state corresponding to the x-axis, y-axis, and z-axis overlap greatly. It is hard to identify the states for some specific points using these standard algorithms because the result of the tri-dimensional data points are generated by the given emission probability for each state.

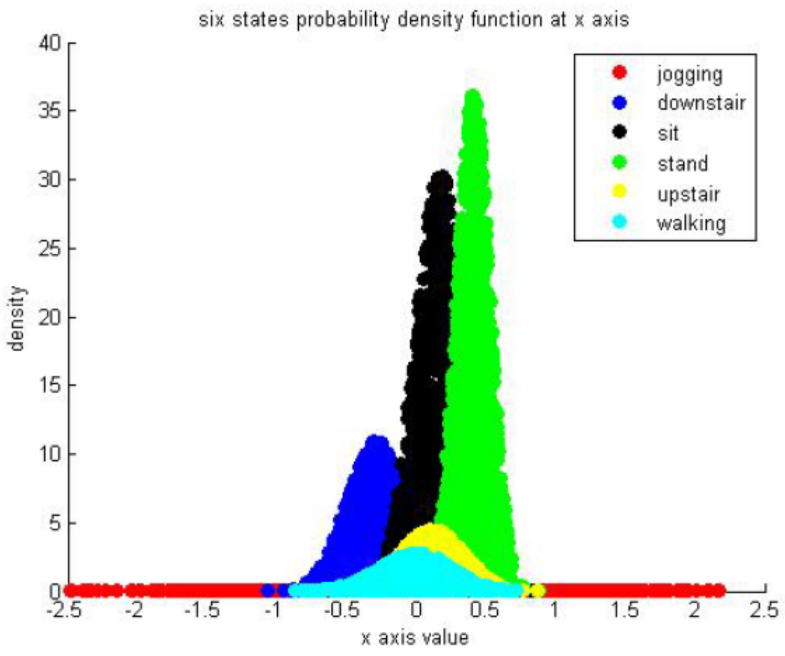


Figure 2.3: The probability function of six activities with respect to x-axis

Checking the probability density function corresponding to the x-axis of six patterns (Figure 2.3), the emission probabilities associated with walking_upstairs and walking almost overlap. The emission probabilities associated with sitting and standing overlap by a half. The jogging varies in the three directions, so it has the largest variance.

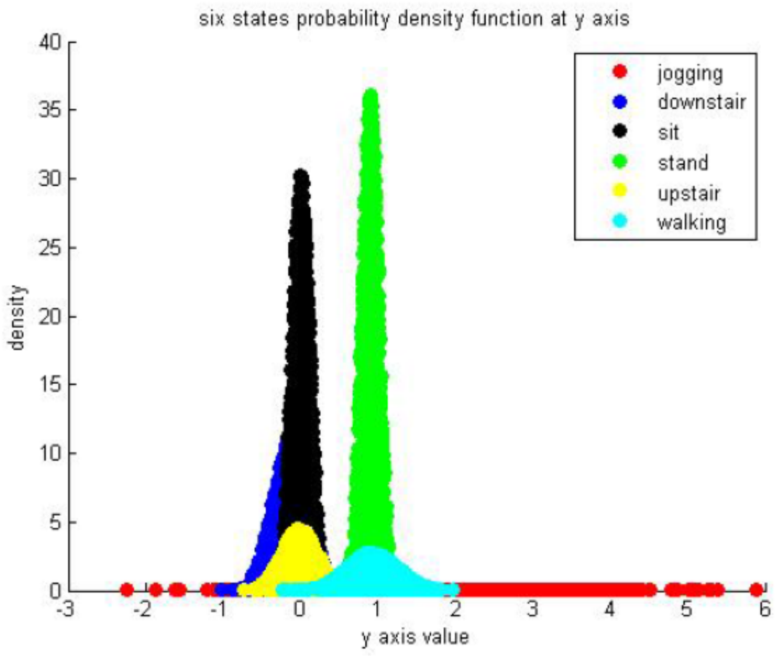


Figure 2.4: The probability function of six activities with respect to y-axis

Apparently, the possible result generated from the emission probability associated with walking pattern includes the data points of standing. The observations whichcome from the emission probability of walking_upstairs contains the sitting data and the part of walking_downstairs data.

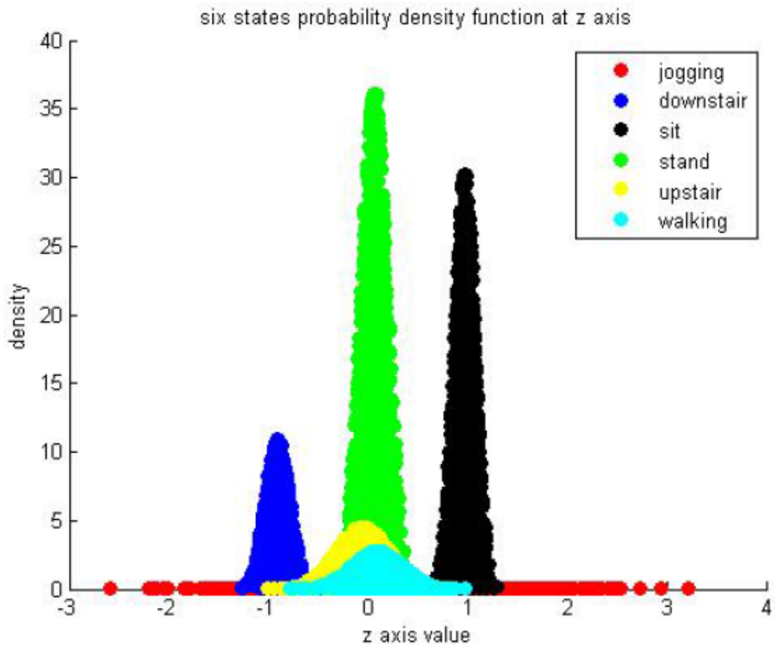


Figure 2.5: The probability function of six activities with respect to z-axis

In Figure 2.5, the outputs generated from the probability densities corresponding to sitting and walking_downstairs do not overlap. Based on the probability density functions of six patterns with respect to three axes as shown in above figures, different states overlap on the different axes, they also di not keep the same overlapping on the different axes. Because it is not efficient to analyze such data individually, we look for a method that could consider connective data, which regard these similar observations generated from the same functions as a whole.

3 Modified Online Empirical Bayesian Segmentation: Online Empirical Bayesian Kernel Segmentation

3.1 Related Work

3.1.1 Online Segmentation

Time series segmentation is looking for the change point between two different patterns and developing a suitable model which fits to the provided observations of every segment, and these observations between two change points are regarded as a subset of the entire time series. Furthermore, the segmentation algorithms can be divided into two groups: offline algorithm and online algorithm. The most common offline algorithms are top-down and bottom-up algorithms [36, 9, 10]. Many papers extend the two offline methods to improve the accuracy on the basis of different technical skills. [30] introduces a local iterative replacement method and a global iterative replacement method that both require and are processed by dynamic programming skill. The Empirical Bayesian method has been applied to discover change points by posterior probability [48]. [17] uses the Fisher information as the cost function rather than the error function (defined in Chapter 3). However, concerning the properties of continuously collecting the time series data, an adaptive and incremental algorithm is more suitable for dealing with the time series. For another category of segmentation, Sliding Window (SW) algorithm has been used for defining segments as an online method. Nonetheless, Sliding Window gives us the undesirable experimental results that are analyzed in [69]. We will discuss several online segmentation algorithms that improve the performance of an online algorithm.

To reach the aim of segmenting the entire time series into finite subsets, the aim usually is constructed in the following ways that are mentioned in [36]:

1. Consider a time series s, which generates m segments with optimal splitting, m is known.

2. Consider a time series s, which generates m segments with optimal splitting such that the maximum error of every segment is not larger than a pre-defined cut-off boundary.

3. Consider a time series s, which generate m segments with optimal splitting such that the total error for combining all segment errors is not larger than a pre-defined cut-off boundary.

First of all, let us define the segmentation frame, a time series s containing N samples $x_1, x_2, ..., x_N$. Assume m segments as $s_1 = s(1 : c_1), s_2 = s(c_1+1 : c_2), ..., s_k = s(c_{m-1} + 1 : N)$, where s_i is ith segment and $s(a, b) = \{x_a, x_{a+1}, ..., x_b\}, a \leq b$, $0 < c_1 < c_2 < ... < c_{m-1} < N$, and $c_0 = 1, c_m = N$, then $s = s_1 s_2, ..., s_m$.

The goal of segmentation is to form the segments such that these observations are homogeneous in a segment and heterogeneous in the different segments. This goal infers these observations that are generated by an individual function during this specified period, such as a probability distribution or a regression model. Depending on the construction process discussed above, the first step would be to define a cost function $F_i, i = 1, 2, .., m$ for an individual segment, and the aim is to minimize the overall cost function F. The cost function F_i can be an arbitrary function; the most general cost function is the sum of variance of the components of a segment. The number of data points of segment s_i is $n_i = c_i - c_{i-1} + 1$, assuming these data points are d dimensional:

$$
\begin{aligned}
F_i &= V(s_i) && (3.1.1)\\
&= \sum_{l=1}^{d} \Big[\frac{1}{n_i} \sum_{j=c_{i-1}+1}^{c_i} x_{jl}^2 - \Big(\frac{1}{n_i} \sum_{j=c_{i-1}+1}^{c_i} x_{jl}^2\Big)^2\Big]
\end{aligned}
$$

Thus, the cost function of combining all segmentations is:

$$
\begin{aligned}
F = V(s_1 s_2 ... s_m) &= \frac{1}{N} \sum_{[i=1]}^{m} n_i V(s_i)\\
&= \frac{1}{N} \sum_{[i=1]}^{m} \sum_{j=c_{i-1}+1}^{c_i} \|x_j - \mu_i\|^2 && (3.1.2)
\end{aligned}
$$

where $N = \sum_{i=1}^{m} n_i$, which infers finding the boundaries c_i of each segment by means of minimizing the cost function.

The advantage and disadvantage of the online segmentation algorithms are different according to the different requirements and the application purposes. It is hard to decide which the best and suitable segmenting algorithm is. Usually, less time consuming of processing an online segmentation is associated with the higher error and vice versa. The method selection should relate to the data type, the experiment requirements, and the task. We expect to build an algorithm with the higher accuracy that could immediately detect the change points in a time series. The two components of time consuming and accuracy should be improved simultaneously.

Because the Sliding Window (SW) method is the first method supported for the time series online segmentation, the first thought is that we could consider developing an online segmentation which extends from SW. SWAB algorithm in [36] discusses associating the Sliding Window with the Bottom-up method. Ten types of observations are used to analyze the performance of the SW, the Bottom-up and the SWAB. The experiments in [36] show that SWAB is almost as effective as the Bottom-up method. Meanwhile, the consuming time of the SWAB becomes an issue for a time series. We can consider a less time invoking Bottom-up algorithm by remembering the calculation of the previous innovations. Dima Alberg and Avner Ben-Yair [4] supply the Interval Sliding Window (ISW) method by adding the confidence level parameters. Different from the SWAB, the ISW method does not require a pre-defined threshold and performs as well as SWAB. Sometimes, a different pre-defined maximum error could result in different segmenting.

For Hidden Markov Model (HMM) based online segmentation, [38] introduces a dynamic HMM algorithm, which improves the Viterbi algorithm without the fixed number of states. It can automatically update the parameters and compare them with the prototype distribution in order to label the type of patterns. [38] experiments with a dynamic switching process with 2 out of 15 incorrect segments at a speed of 457 data points per second, a performance showing a fast online segmenting algorithm. Different from [38], the advantaged dynamic HMM algorithm [57] can label multiple patterns at the same time, and the applications of human motion have been demonstrated that can stably detect the change point with a reduced delay rate. Also, this system uses the SVM (Support Vector Machine) for classification rather than an integrated squared error of probability density function. However, the emission probability of HMM could be 0 in the high-dimensional observations, which is one of the biggest issues in HMM. The technical skill to solve this problem is the feature reduction by the principle component analysis for reducing dimensions.

Another distribution-based online segmenting method is Online Empirical Bayesian (OB) detecting algorithm. Unlike HMM, the pre-defined number of states is not necessary. Instead of generating k posterior distributions from k classes at time t, Online Empirical Bayesian Segmentation in [?] produces t posterior distributions from the previous $t-1$ posterior distributions and a new posterior distribution of a new coming observation at time t. It is more flexible than the HMM, but this procedure increases computational time. To handle this complex problem, [62] and [58] fix a constant number of particles at each time.

Piecewise Linear Approximation segmentation as an online segmentation method constructs segments by approximating a line from the Feasible Space Window (FSW) method and the Stepwise FSW (SFSW) method [49]. Because the SFSW algorithm considers the past observations with an overview of the recent two segments generated by the FSW algorithm, the SFSW has a better overall performance than the FSW method, the SWAB, and the SW. The overall performance includes the number of segments and the value of cost function. Due to the higher number of segments, this usually results in smaller margin of error. However, it is meaningless to create as many as possible segments. The worst performance comes from the SW, followed by the FSW. For online detecting process, the problem of time-consumption can not be ignored. [49] also shows the FSW along with much less computing time, followed by the SW. The experiments show that the SWAB costs the most time. Therefore, the FSW and the SFSW improve the process speed and the performance. At the same time, these two methods are only used for one-dimensional observation. However, a multidimensional time series is more common in real life. On the other hand, [49] and [80] only use a straight line for fitting time series, which is not a suitable method to adapt to changeable observations. [81] extends the traditional piecewise linear model to a polynomial function. It introduces the coefficient space based model instead of the space window, which supplies more choices of a function (different order of polynomial) and it is more practical to fit a nonlinear time series. The [81] proposed method displays less error and better performance than the FSW and is almost able to adapt to all types of time series data.

3.1.2 Kernel Density Estimation

Kernel Density Estimation (KDE) estimates the probability density function of a random variable based on a non-parametric approach, which is a fundamental data smoothing approach which makes inferences about the population. This method is used in many fields, such as engineering, economics, and biostatistics. Traditionally,the KDE requires the random sample is independent and identically distributed (i.i.d), and the time series observations are time dependent. However, if the observations used in the estimation are a stationary process within a specific interval, we can extend the most techniques of KDE to time-dependent variables [28].

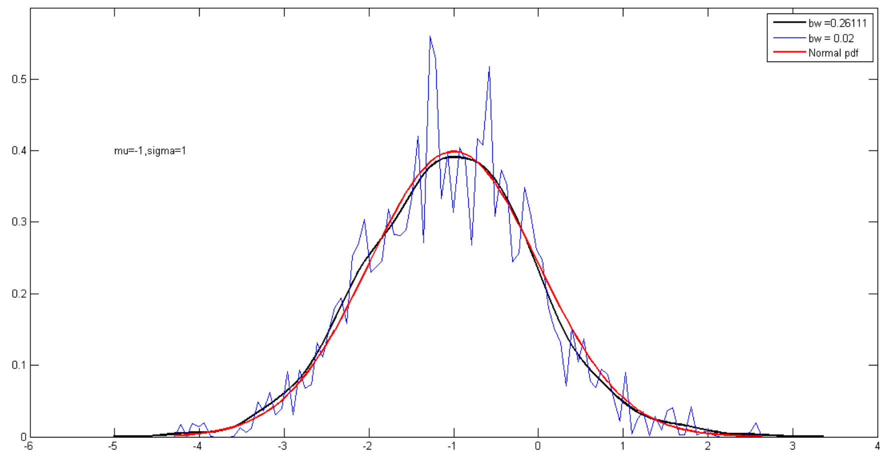


Figure 3.1: Univariate Kernel Density Estimation at $\mu = 1$ and $\sigma = 1$

Let $(x_1, x_2, ..., x_n)$ be the independent and identically distribution samples drawn from an unknown density f. The kernel density estimation in Figure 4.2 is defined as:

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{3.1.3}$$

where $K(\cdot)$ is called kernel and the kernel function satisfies following conditions:

$$
\begin{aligned}
&1. K(x) > 0, \int_R K(x)dx = 1.\\
&2. Meaniszero: \int_R xK(x)dx = 0.\\
&3. Finitesecondmonment: \int_R x^2 K(x)dx < \infty
\end{aligned}
\tag{3.1.4}
$$

The most widely used kernel function is the Gaussian density distribution. Following is listed a range of kernel functions in Table 3.1.

Table 3.1: The classical kernel functions

Kernel	Kernel Function $K(u)$
Uniform	$K(u) = \frac{1}{2}\,\mathbf{I}_{\{\lvert u\rvert\le 1\}}$
Gaussian	$K(u) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}u^2}$
Epanechnikov	$K(u) = \frac{3}{4}(1-u^2)\,\mathbf{I}_{\{\lvert u\rvert\le 1\}}$
Triangular	$K(u) = (1-\lvert u\rvert)\,\mathbf{I}_{\{\lvert u\rvert\le 1\}}$
Triweight	$K(u) = \frac{35}{32}(1-u^2)^3\,\mathbf{I}_{\{\lvert u\rvert\le 1\}}$
Tricube	$K(u) = \frac{70}{81}(1-\lvert u\rvert^3)^3\,\mathbf{I}_{\{\lvert u\rvert\le 1\}}$
Biweight	$K(u) = \frac{15}{16}(1-u^2)^2\,\mathbf{I}_{\{\lvert u\rvert\le 1\}}$
Cosine	$K(u) = \frac{\pi}{4}\cos\left(\frac{\pi}{2}u\right)\mathbf{I}_{\{\lvert u\rvert\le 1\}}$
Silverman	$K(u) = \frac{1}{2}e^{-\frac{\lvert u\rvert}{\sqrt{2}}}\cdot\sin\left(\frac{\lvert u\rvert}{\sqrt{2}}+\frac{\pi}{4}\right)$

Here, $h > 0$ called bandwidth is a smoothing parameter. Based on the different values of h, the estimated density curve could be over-smoothed or under-smoothed. Therefore, what value of h we should choose becomes very important. A few papers already mentioned the techniques of bandwidth selection, such as [18],[29]. The most used optimal bandwidth is extracted by minimizing the mean integrated squared error, defined as:

$$MISE(h) = E\left[\int (\hat{f}_h(x) - f(x))^2 dx\right] \tag{3.1.5}$$

The Asymptotic MISE (AMISE) is built on a weak assumption of f and K, $MISE(h) = AMISE(h) + o(1/nh + h^2)$, which generates the following terms:

$$AMISE(h) = \frac{R(K)}{nh} + \frac{1}{4}m_2(K)^2 h^4 R(f'') \tag{3.1.6}$$

where $R(g) = \int g(x)^2 dx$ for a function g, $m_2(K) = \int x^2 K(x) dx$ and f'' is the second derivation. The optimal h is derived from minimizing AMISE :

$$\begin{aligned} \frac{\delta}{\delta h} AMISE(h) &= -\frac{R(K)}{nh^2} + m_2(K)^2 h^3 R(f'') = 0 \\ h_{AMISE} &= \frac{R(K)^{1/5}}{m_2(K)^{2/5} R(f'')^{1/5} n^{1/5}} \end{aligned} \tag{3.1.7}$$

To face the problem of handling the multivariate data, in the 1990s and 2000s, the Multivariate Kernel Density Estimation has achieved a stronger estimating capability compare to its univariate counterparts [71]. Let $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ be a d-dimensional random sample that comes from a multivariate density function f. The multivariate kernel density function is defined as:

$$\hat{f}_H(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_H(\mathbf{x} - \mathbf{x}_i) = \frac{1}{n|H|^{1/2}}\sum_{i=1}^{n} K(H^{-1/2}(\mathbf{x} - \mathbf{x}_i)) \tag{3.1.8}$$

Similar to the Univariate Kernel Density function, H is the bandwidth that is $d \times d$ symmetric and positive definite matrix. The most common kernel function used in the multivariate case is the standard multivariate normal kernel, and H is a covariance matrix. At this point, matrix H is the most important factor since it controls the amount and the orientation of smoothing [78]. To consider the AMISE:

$$AMISE(H) = n^{-1}|H|^{-1/2}R(K) + \frac{1}{4}m_2(K)^2(vec^T H)\Psi_4(vecH) \tag{3.1.9}$$

where

$$\begin{aligned}
&R(K) = \int K(\mathbf{x})^2 dx \\
&m_2(K)I_d = \int \mathbf{x}\mathbf{x}^T K(\mathbf{x})d\mathbf{x} \text{ with } I_d \text{ is the identity matrix} \\
&\Psi_4 = \int (vecD^2 f)(vecD^2 f)^T d\mathbf{x}\text{is a } d^2 \times d^2 matrix \\
&D^2 f \text{ is the} d \times d \text{ Hessian matrix of second order partial derivative of } f \\
&vec \text{ is vector operator which}
\end{aligned} \tag{3.1.10}$$

There is a Plug-in method discussed in [78], a Smoothed Cross-validation method discussed in [18] and a Rule of thumb method, which are different types of methods used to select the optimal bandwidth matrix.

3.1.3 Unscented Transformation

The Unscented Transformation (UT) estimates the results of applying a given non-linear transformation to a probability distribution based on a mathematical function. If the state transition and the observation models that are the predict function f and the update function h respectively, which are highly non-linear with each other, then the extended Kalman Filter can give very poor performance [35]. The reason is due to the covariance which is propagated by the linearization of the underlying non-linear model. The Unscented Kalman Filter (UKF) ([35],[34]) picks a minimal amount of sample points known as sigma points, around the mean through using a deterministic sampling technique called the unscented transform. The sigma points are then produced through the non-linear functions, from which result in the mean and covariance of the estimate. The result of the filter as shown in [35] can more accurately capture the true mean and the true covariance.

For the prediction, UKF prediction can be used independently from the UKF update. The estimated states and covariance are augmented with the mean and covariance of the process noise.

$$\begin{aligned}\mathbf{x}^a_{k-1|k-1} &= [\hat{\mathbf{x}}^{\mathrm{T}}_{k-1|k-1} \quad E[\mathbf{w}^{\mathrm{T}}_k]\ ]^{\mathrm{T}} \\ \mathbf{P}^a_{k-1|k-1} &= \begin{bmatrix} \mathbf{P}_{k-1|k-1} & 0 \\ 0 & \mathbf{Q}_k \end{bmatrix}\end{aligned} \tag{3.1.11}$$

The augmented state and covariance can derive a set of $2L+1$ sigma points, where L is the dimension of the augmented state.

$$\begin{aligned}
\chi^0_{k-1|k-1} &= \mathbf{x}^a_{k-1|k-1} \\
\chi^i_{k-1|k-1} &= \mathbf{x}^a_{k-1|k-1} + \left(\sqrt{(L+\lambda)\mathbf{P}^a_{k-1|k-1}}\right)_i, \qquad i = 1, \ldots, L \\
\chi^i_{k-1|k-1} &= \mathbf{x}^a_{k-1|k-1} - \left(\sqrt{(L+\lambda)\mathbf{P}^a_{k-1|k-1}}\right)_{i-L}, \qquad i = L+1, \ldots, 2L
\end{aligned} \tag{3.1.12}$$

where $\left(\sqrt{(L+\lambda)\mathbf{P}^a_{k-1|k-1}}\right)_i$ is the ith column of the square root of matrix $(L+\lambda)\mathbf{P}^a_{k-1|k-1}$

The sigma points are generated through the transition function f. $\chi^i_{k|k-1} = f(\chi^i_{k-1|k-1}) \quad i = 0, \ldots, 2L$ where $f : R^L \rightarrow R^{|\mathbf{x}|}$. The weighted sigma points are recombined to produce the predicted state and covariance.

$$\begin{aligned}
\hat{\mathbf{x}}_{k|k-1} &= \sum_{i=0}^{2L} W^i_s \chi^i_{k|k-1} \\
\mathbf{P}_{k|k-1} &= \sum_{i=0}^{2L} W^i_c \; [\chi^i_{k|k-1} - \hat{\mathbf{x}}_{k|k-1}][\chi^i_{k|k-1} - \hat{\mathbf{x}}_{k|k-1}]^{\mathrm{T}}
\end{aligned} \tag{3.1.13}$$

where the weights for the state and covariance are given by:

$$\begin{aligned}
W^0_s &= \frac{\lambda}{L+\lambda} \\
W^0_c &= \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \\
W^i_s = W^i_c &= \frac{\lambda}{2(L+\lambda)} \\
\lambda &= \alpha^2(L+\kappa) - L
\end{aligned} \tag{3.1.14}$$

α and κ restrict the spread of the sigma points. β is correlated to the distribution of x. Normal values are $\alpha = 10^{-3}$, $\kappa = 0$ and $\beta = 2$. If the true distribution of x is a Gaussian distribution, $\beta = 2$ is the optimal choice[77].

The next phase is to update the step:

$$\begin{aligned}
\mathbf{x}^a_{k|k-1} &= [\hat{\mathbf{x}}^{\mathrm{T}}_{k|k-1} \quad E[\mathbf{v}^{\mathrm{T}}_k]\ ]^{\mathrm{T}} \\
\mathbf{P}^a_{k|k-1} &= \begin{bmatrix} \mathbf{P}_{k|k-1} & 0 \\ 0 & \mathbf{R}_k \end{bmatrix}
\end{aligned} \tag{3.1.15}$$

As mentioned before, the set of $2L + 1$ sigma points is generated from the augmented state and covariance where L is the dimension of the augmented state.

$$\begin{aligned}
\chi^0_{k|k-1} &= \mathbf{x}^a_{k|k-1} \\
\chi^i_{k|k-1} &= \mathbf{x}^a_{k|k-1} + \left(\sqrt{(L+\lambda)\mathbf{P}^a_{k|k-1}}\right)_i, \qquad i = 1, \ldots, L \\
\chi^i_{k|k-1} &= \mathbf{x}^a_{k|k-1} - \left(\sqrt{(L+\lambda)\mathbf{P}^a_{k|k-1}}\right)_{i-L}, \qquad i = L+1, \ldots, 2L
\end{aligned} \tag{3.1.16}$$

The sigma points are designed through the observation function h.

$$\gamma^i_k = h(\chi^i_{k|k-1}) \quad i = 0..2L \tag{3.1.17}$$

Recombining the weighted sigma points produce the predicted measurement and the predicted measurement covariance, which are defined as follows:

$$\begin{aligned}\hat{\mathbf{z}}_k &= \sum_{i=0}^{2L} W_s^i \gamma_k^i \\ \mathbf{P}_{z_k z_k} &= \sum_{i=0}^{2L} W_c^i \; [\gamma_k^i - \hat{\mathbf{z}}_k][\gamma_k^i - \hat{\mathbf{z}}_k]^{\mathrm{T}}\end{aligned} \tag{3.1.18}$$

The following state-measurement cross-covariance matrix is used to compute the UKF Kalman gain.

$$\begin{aligned}\mathbf{P}_{x_k z_k} &= \sum_{i=0}^{2L} W_c^i \; [\chi_{k|k-1}^i - \hat{\mathbf{x}}_{k|k-1}][\gamma_k^i - \hat{\mathbf{z}}_k]^{\mathrm{T}} \\ K_k &= \mathbf{P}_{x_k z_k} \mathbf{P}_{z_k z_k}^{-1}\end{aligned} \tag{3.1.19}$$

As with the Kalman filter, the updated state is the predicted state with the innovation weighted by the Kalman gain,

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{z}_k - \hat{\mathbf{z}}_k) \tag{3.1.20}$$

The updated covariance is the predicted covariance minus the predicted measurement covariance and weighted by the Kalman gain.

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - K_k \mathbf{P}_{z_k z_k} K_k^{\mathrm{T}} \tag{3.1.21}$$

3.2 Online Empirical Bayesian Kernel Segmentation

Considering many time series applications, such as speech recognition, stock price, and pattern detection, the first step of time series analysis is to separate the entire time series interval into disjoined smaller segments. This segmentation step is to pre-process the time series, because the raw individual observations cannot be applied directly in the classification methods. Usually, we partition the entire time series into the equal length segments, the length of the segment could be 1.5s, 2s, and 3s, etc. However, these type of intervals is so short that they do not capture enough information of activities. In this research, we discuss the step based on Online Empirical Bayesian Kernel Segmentation (OBKS) method that is introduced in section 2.2. This approach detects the change point between two different connected patterns, and it defines the observations between two change points as a segment. OBKS procedure results in different length time intervals rather than equal and small length. For the next step, we need to extract the features from each segment. The features widely used so far are categorized as time-domain features, frequency domain features, wavelet features and heuristic features [7]. Here, we mainly use the time-domain features and the frequency-domain features. The authors of [46] discusses the current classification methods including k-nearest neighbors, decision tree, empirical bayesian, neural network, support vector machines, Fuzzy logic, classifier ensembles, regression methods and Markov models that are applied widely in many publications and applications. Then, testing our method on the simulated data and the empirical observations, we use the accuracy and confusion matrix as the evaluation index. The application of the time series segmentation, feature extraction, and classification is shown in Figure 6.1, which displays the process of a real-time activity recognition.

3.2.1 Empirical Bayesian Online Segmentation

In this section, we consider d-dimensional data points $\mathbf{x} = (x_1, x_2, ..., x_d)$. The task of segmenting the time series is to group the homogeneous observations together and separate the heterogeneous observations. These observations are listed on a time line, and all observations between two change points construct a time series segment defined as a pattern. The pattern during period $t_1, t_2, ...t_i$ is different than the pattern during time period $t_{i+1}, t_{i+2}, ..., t_j$. Those homogeneous observations are assumed to follow a multivariate distribution, and the different patterns follow different multivariate distributions. Therefore, to find the change point between two patterns becomes a significant problem.

The Online Empirical Bayesian detecting method [2] can be used to prepare these segments automatically for classification. First of all, we consider the concept of "run length" r_t that is the length of the current run at time t and it is linearly increasing over time t. For example, if $r_t = 0$ at $t = 8$, $\mathbf{x}_8$ is a change point; if $r_t \neq 0$, we run once more and repeat the process. $\mathbf{x}_t^{(r)}$ is defined as the set corresponding to run length r_t. If r_t is zero, $\mathbf{x}^{(r)}$ is an empty set. For example, $t = 9$, $r_t = 1$, then $\mathbf{x}_9^{(r)} = \{\mathbf{x}_8, \mathbf{x}_9\}$. To find the posterior distribution $P(r_t|\mathbf{x}_{1:t})$, we need to generate a recursive joint distribution $P(r_t, \mathbf{x}_{1:t})$,

$$\begin{aligned} P(r_t|\mathbf{x}_{1:t}) &= \frac{P(r_t, \mathbf{x}_{1:t})}{P(\mathbf{x}_{1:t})} \propto P(r_t, \mathbf{x}_{1:t}) \\ &\propto \sum_{r_{t-1}} P(r_t, r_{t-1}, \mathbf{x}_t, \mathbf{x}_{1:t-1}) \\ &\propto \sum_{r_{t-1}} P(r_t, \mathbf{x}_t|r_{t-1}, \mathbf{x}_{1:t-1}) P(r_{t-1}, \mathbf{x}_{1:t-1}) \\ &\propto \sum_{r_{t-1}} P(r_t|r_{t-1}) P(\mathbf{x}_t|r_{t-1}, \mathbf{x}_{t-1}^{(r)}) P(r_{t-1}, \mathbf{x}_{1:t-1}) \end{aligned} \tag{3.2.22}$$

Here, $P(r_t|r_{t-1})$ is a prior probability, the joint distribution $P(r_t, \mathbf{x}_{1:t})$ is called growth probability and $P(\mathbf{x}_t|r_{t-1}, \mathbf{x}_{t-1}^{(r)})$ is a predictive probability. At every recursion, we pick the r_t that is associated with the largest posterior probability, which is the r_t that is also associated with the largest joint distribution in recent data. Next, we need get the prior distribution $P(r_t|r_{t-1})$ and the predictive distribution $P(\mathbf{x}_t|r_{t-1}, \mathbf{x}_t^{(r)})$.

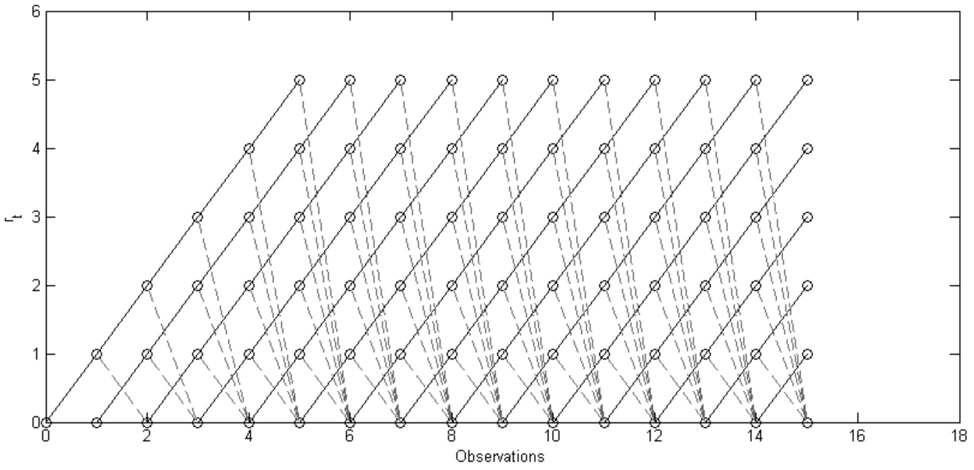


Figure 3.2: "Run length" r_t path: Solid line represents $r_{t+1} = r_t + 1$, Dashed lines represent $r_{t+1} = 0$ and $\mathbf{x}_{t+1}$ is change point

The run length has two directions. One direction is that no change point happens at time t and $r_t = r_{t-1} + 1$, which means the new data still stays in the same group with the previous observations and follows the same distribution. Another one is that a change point occurs at time t, r_t drops to 0 with probability $H(r_t) = 1/\lambda$. Here, $H(r_t)$ is a hazard function based on the geometric distribution with parameter λ [75]. The prior distribution is:

$$P(r_t|r_{t-1}) = \begin{cases} H(r_{t-1}) & \text{if } r_t = 0 \\ 1 - H(r_{t-1}) & \text{if } r_t = r_{t-1} + 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.2.23}$$

The predictive probability $P(\mathbf{x}_{t+1}|r_t, \mathbf{x}_t^{(r)})$ is the marginal distribution integral over the parameter vectors θ corresponding to current run length r_t, which only depends on the recent data set $\mathbf{x}_t^{(r)}$, since the assumption of the distribution stays the same in recent data. We define it as following:

$$P(\mathbf{x}_{t+1}|r_t, \mathbf{x}_t^{(r)}) = \int P(\mathbf{x}_{t+1}|\theta) P(\theta_t^{(r)} = \theta | r_t, \mathbf{x}_t^{(r)}) d\theta \tag{3.2.24}$$

Here, $\theta_t^{(r)}$ is the parameter of the current run length. Assuming the tri-dimensional data $\mathbf{x} = (x_1, x_2, x_3)^T$ follows a three-dimensional normal distribution with mean μ and inverse-covariance matrix $\Omega = \Sigma^{-1}$ and dimension $d = 3$, the likelihood function of t data points $\mathbf{X_{1:t}} = \mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_t}$ is described as follows:

$$\begin{aligned}
P(\mathbf{X_{1:t}}|\mu, \mathbf{\Omega}) &= \prod_{i=1}^{t}(2\pi)^{-d/2}|\Omega|^{1/2}exp\Big(-\frac{1}{2}(\mathbf{x}_i-\mu)^T\Omega(\mathbf{x}_i-\mu)\Big) \\
&= (2\pi)^{-td/2}|\Omega|^{t/2}exp\Big(-\frac{1}{2}\sum_{i=1}^{t}(\mathbf{x}_i-\mu)^T\Omega(\mathbf{x}_i-\mu)\Big) \\
&\propto |\Omega|^{t/2}exp\Big(-\frac{1}{2}\sum_{i=1}^{t}(\mathbf{x}_i-\mu)^T\Omega(\mathbf{x}_i-\mu)\Big) \qquad (3.2.25)
\end{aligned}$$

For the prior distribution $P(\mu, \Omega)$ in Eq(3.2.25), we assume $\mu \sim \mathcal{N}(\mu_0, (\kappa_0\Omega)^{-1})$ normal distribution and $\Omega \sim Wi_d(T_0, \nu_0)$ Wishart distribution. Under the assumption that μ and Ω are independent of each other, we simply multiply these two probabilitity functions as the prior distribution:

$$\begin{aligned}
&P(\mu, \Omega|\mu_0, \kappa_0, \nu_0, T_0) \\
&= \mathcal{N}(\mu|\mu_0, \kappa_0)Wi_d(\Omega|\nu_0, T_0) \\
&= (2\pi)^{-d/2}|\kappa_0\Omega|^{1/2}exp\Big(-\frac{1}{2}(\mu-\mu_0)^T(\kappa\Omega)(\mu-\mu_0)\Big)| \\
&\times \Omega|^{(\nu_0-d-1)/2}exp\Big(-tr(T\Omega)/2\Big)2^{-\nu_0 d/2}|T|^{-\nu_0/2}\Gamma_d(\nu_0/2) \\
&\propto |\Omega|^{1/2}exp\Big(-\frac{1}{2}(\mu-\mu_0)^T(\kappa\Omega)(\mu-\mu_0)\Big)|\Omega|^{(\nu_0-d-1)/2}exp\Big(-tr(T\Omega)/2\Big) \qquad (3.2.26)
\end{aligned}$$

Therefore, we can get the posterior distribution:

$$
\begin{aligned}
&P(\mu,\Omega|X_{1:t}) = \frac{P(\mu,\Omega,X_{1:t})}{P(X_{1:t})} = \frac{P(X_{1:t}|\mu,\Omega)P(\mu,\Omega)}{P(X_{1:t})} \propto P(X_{1:t}|\mu,\Omega)P(\mu,\Omega) \\
&\propto |\Omega|^{t/2} exp\Big(-\frac{1}{2}\sum_{i=1}^{t}(x_i-\mu)^T\Omega(x_i-\mu)\Big) \\
&\times |\Omega|^{1/2} exp\Big(-\frac{1}{2}(\mu-\mu_0)^T(\kappa\Omega)(\mu-\mu_0)\Big)|\Omega|^{(\nu_0-d-1)/2} exp\Big(-tr(T\Omega)/2\Big) \\
&\propto |\Omega|^{1/2}|\Omega|^{(\nu_0+t-d-1)/2} exp\Big(-\frac{1}{2}(\mu-\frac{\kappa_0\nu_0+t\bar{X}}{\kappa_0+t})^T(\kappa\Omega)(\mu-\frac{\kappa_0\nu_0+t\bar{X}}{\kappa_0+t})\Big) \\
&\times |\Omega|^{(\nu_0-d-1)/2} exp\Big(-tr(T\Omega)/2\Big) \\
&\times exp\Big((T_0+\sum(X_i-\bar{X})(X_i-\bar{X})^T+\frac{\kappa_0 t}{\kappa_0+t}(\mu_0-\bar{X})(\mu_0-\bar{X})^T)\Big) \\
&\propto |\Omega|^{1/2}|exp(-\frac{1}{2}(\mu-\mu_t)(\kappa_t\Omega)(\mu-\mu_t)^T)|\Omega|^{(\nu_t-d-1)/2} exp\Big(T_t\Omega\Big) \\
&\propto \mathcal{N}(\mu|\mu_t,\kappa_t)Wi_d(\Omega|\nu_t,T_t) \qquad (3.2.27)
\end{aligned}
$$

where,

$$
\begin{aligned}
&\kappa_t = \kappa_0 + t \\
&\mu_t = \frac{\kappa_0\mu_0 + t\bar{\mathbf{X}}_t}{\kappa_0+t} \\
&\nu_t = \nu_0 + t \\
&T_t = T_0 + \sum(\mathbf{X_i}-\bar{\mathbf{X}}_\mathbf{t})(\mathbf{X_i}-\bar{\mathbf{X}}_\mathbf{t})^\mathbf{T} + \frac{\kappa_\mathbf{0}\mathbf{t}}{\kappa_\mathbf{0}+\mathbf{t}}(\mu_\mathbf{0}-\bar{\mathbf{X}}_\mathbf{t})(\mu_\mathbf{0}-\bar{\mathbf{X}}_\mathbf{t})^\mathbf{T} \qquad (3.2.28)
\end{aligned}
$$

Therefore, the updated step for every new incoming data as shown in Eq(3.2.29).

$$\begin{aligned}
\kappa_{t+1} &= \kappa_t + 1 \\
\mu_{t+1} &= \frac{\kappa_t \mu_t + \mathbf{X_{t+1}}}{\kappa_t + 1} \\
\nu_{t+1} &= \nu_t + 1 \\
T_{t+1} &= \frac{\kappa_t (\mathbf{X_{t+1}} - \mu_\mathbf{t})(\mathbf{X_{t+1}} - \mu_\mathbf{t})^\mathbf{T}}{\kappa_t + 1}
\end{aligned} \tag{3.2.29}$$

Finally, the posterior predictive probability is

$$P(\mathbf{X_{t+1}} | \mathbf{X_t^r}, \mathbf{r_t}) = \mathbf{t}_{\nu_\mathbf{t} - \mathbf{d} + \mathbf{1}}(\mu_\mathbf{t}, \frac{\mathbf{T_t}(\kappa_\mathbf{t} + \mathbf{1})}{\kappa_\mathbf{t}(\nu_\mathbf{t} - \mathbf{d} + \mathbf{1})}) \tag{3.2.30}$$

When new data is observed, the algorithm updates the parameters and the joint distribution $P(r_t, \mathbf{x}_{1:t})$, which approximates the posterior distribution $P(r_t|\mathbf{x}_{1:t})$. This Empirical Bayesian method creates t posterior distributions $\{P(r_t|\mathbf{x}_{i:t})\})\}_{i=1}^{t}$ at every iteration time t, to pick up the r_t associated with the highest posterior probability. If r_t changes to 0 that means a new segment is formed, with $\mathbf{x}_t$ defined as a change point.

3.2.2 Adjusted Online Empirical Bayesian Method

Time-consuming is one of the chief issues of the Online Empirical Bayesian algorithm because the computing time is linear with the number of observations, which makes the computing time linearly increase with time. Motivated by Sliding Window and

Bottom-up (SWAB) [36] that combines the Sliding Window (SW) algorithm with the Bottom-up algorithm, we introduce the SW method with a fixed window size N, and N is so large that it includes at least a few pattern segments. In this experiment, we take $N = 10000$ that is equivalent to 200s. The adjusted Online Empirical Bayesian method re-initializes all parameters when starting a new sliding window. The algorithm of an adjusted Online Empirical Bayesian Segmenting is shown in Figure 3.3.

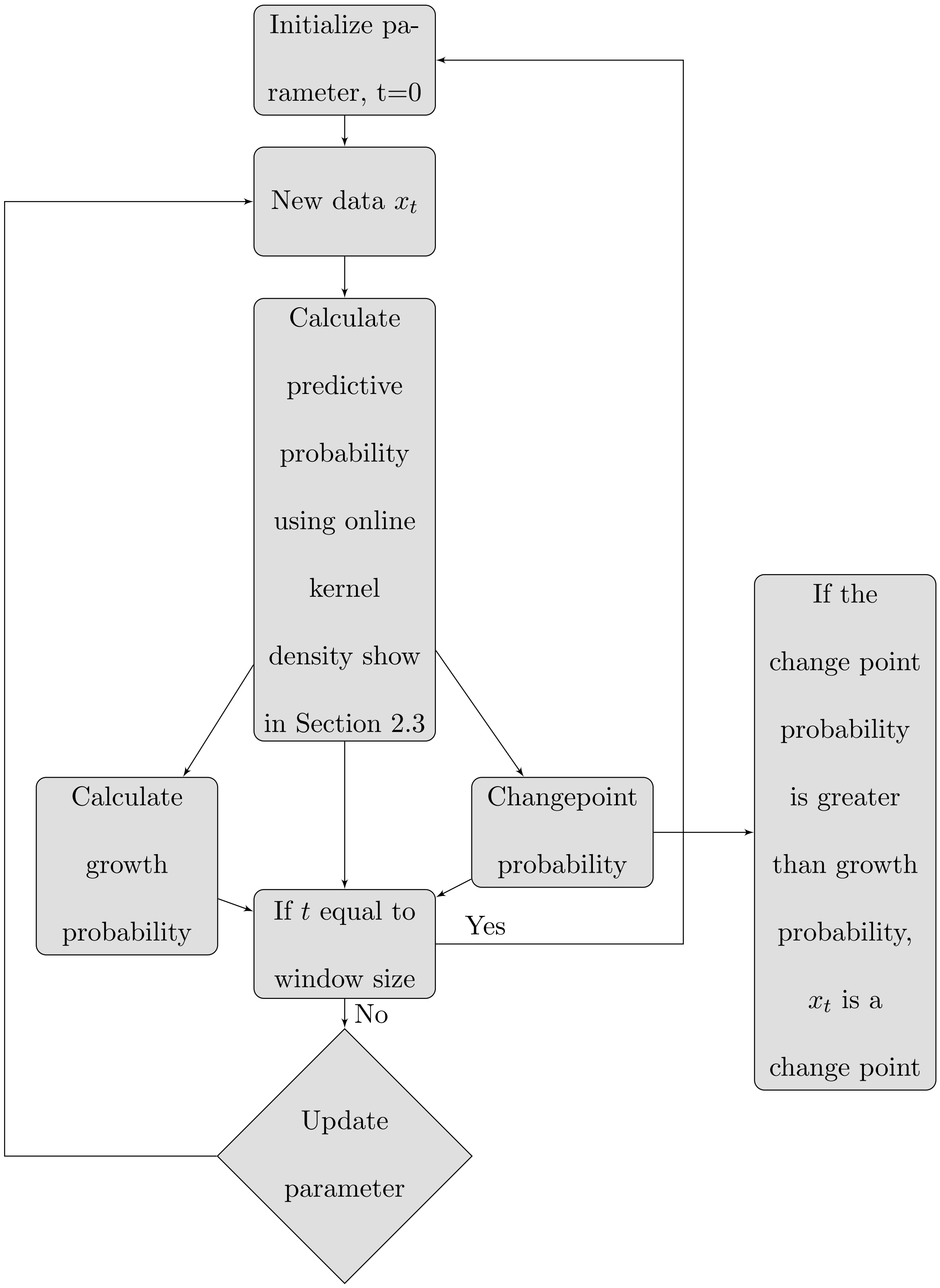

Initialize parameter, t=0
New data x_t
Calculate predictive probability using online kernel density show in Section 2.3
Calculate growth probability
Changepoint probability
If t equal to window size
Yes
No
Update parameter
If the change point probability is greater than growth probability, x_t is a change point

Figure 3.3: Online Empirical Bayesian Kernel Segmentation

3.3 Online Multivariate Kernel Estimation

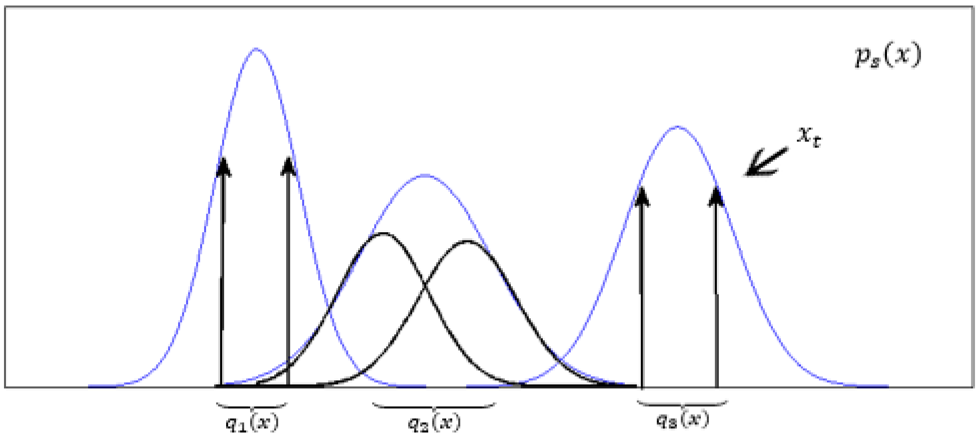


Figure 3.4: Sample distribution $p_s(\mathbf{x})$ with associated detail distributions $q_i(\mathbf{x})$

We apply online kernel function as the predictive function to get rid of the assumption of multi-normality rather than considering the posterior predictive function. At this point, the normality of observations is not a strict requirement anymore. As a nonparametric density estimation, the Multivariate Kernel Density estimation is more flexible and more adaptive and is suitable to fit any distributions. [41] has proposed an Online Multivariate Kernel Density estimation algorithm, which creates an online bandwidth estimation method and designed a compression model that reduces the KDE's complexity. The compressed model of d-dimensional data as an N-component Gaussian mixture model (Figure 3.4) is defined as:

$$p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\Sigma_{si}}(\mathbf{x} - \mathbf{x}_i) \tag{3.3.31}$$

where

$$\phi_{\Sigma}(\mathbf{x}-\mu) = (2\pi)^{-d/2}|\Sigma|^{-1/2}e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)} \tag{3.3.32}$$

is a Gaussian kernel with the center μ and covariance matrix Σ. The Gaussian mixture model is used the most widely for the the data come from one or more different distribution[60, 54], it can be well estimated if all observations come from a same distribution. The Kernel density estimation with a bandwidth $\mathbf{H}$:

$$\hat{p}_{KDE} = \phi_{\mathbf{H}} * p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\Sigma_{si}+H}(\mathbf{x}-\mathbf{x}_i) \tag{3.3.33}$$

In order to reduce the complexity of KDE as new data adding in, we need to compress the sample distribution $p_s(\mathbf{x})$ with time by replacing the clusters of the components. There is an additional model $q_i(\mathbf{x})$ for each component that is used for recovery from these early over-compressions (Figure 3.4). The combined model is:

$$S_{model} = \{p_s(\mathbf{x}), \{q_i(\mathbf{x})\}_{i=1:N}\} \tag{3.3.34}$$

3.3.1 Bandwidth Selection

The traditional measure of the difference between $\hat{p}_{KDE}$ and the unknown underlying Probability Density Function (PDF) is "Asymptotic Mean Integrated Squared Error (AMISE)", which is defined in Eq(3.3.40). The construction of AMISE is given in the following steps from Eq(3.3.35) to Eq(3.3.40). First, let us define the Multivariate Taylor's series as shown in [39]. Define f to be a d variate function, $\mathcal{D}_f(\mathbf{x})$ to be the vector of first-order partial derivatives, and $\mathcal{H}_f(\mathbf{x})$ to be the Hessian matrix of f. Assume all of the entries of $\mathcal{H}$ are continuous in a neighborhood of $\mathbf{x}$. We have:

$$f(\mathbf{x}+a) = f(\mathbf{x}) + a^T\mathcal{D}_f(\mathbf{x}) + \frac{1}{2}a^T\mathcal{H}_f(\mathbf{x})a + o(a^Ta) \tag{3.3.35}$$

The Mean Square Error of a function is defined as:

$$MISE(H) = E\Big[\int \Big((\hat{P}_{KDE}(\mathbf{x})) - P_{KDE}(\mathbf{x})\Big)^2 d\mathbf{x}\Big] \tag{3.3.36}$$

Under weak assumptions, $MISE(H) = AMISE(H) + o(tr(|H|^2) + n^{-1}|H|^{-1/2})$. Hence,

$$AMISE(H) = \int \Big[E\Big(\hat{P}_{KDE}(\mathbf{x})\Big) - P_{KDE}(\mathbf{x})\Big]^2 d\mathbf{x} + \int V(\hat{P}_{KDE}(\mathbf{x}))d\mathbf{x} \tag{3.3.37}$$

If we contain all the observations in the sample model without compression and each observation associated with one kernel function, variance $\Sigma_{si} = 0$ for all i

and $\hat{P}_{KDE}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_H(\mathbf{x} - \mathbf{x}_i)$. Here,

$$E\Big(\hat{P}_{KDE}(\mathbf{x})\Big)$$
$$= \int K_H(\mathbf{x}-\mathbf{y}) P_{KDE}(\mathbf{y}) d\mathbf{y}$$
$$= \int |H|^{-1/2} K(H^{-1/2}(\mathbf{x}-\mathbf{y})) P_{KDE}(\mathbf{y}) d\mathbf{y}$$
$$= \int K(\mathbf{z}) P_{KDE}(\mathbf{x} - H^{1/2}\mathbf{z}) d\mathbf{z}$$

(Multivariate Taylor's series)

$$= \int P_{KDE}(\mathbf{x}) K(\mathbf{z}) d\mathbf{z} - \int \mathbf{z}^T H^{1/2} \mathcal{D}_f(\mathbf{x}) K(\mathbf{z}) d\mathbf{z} + \frac{1}{2} \int \mathbf{z}^T H^{1/2} \mathcal{H}_f(\mathbf{x}) H^{1/2} \mathbf{z} K(\mathbf{z}) d\mathbf{z} + o(tr(H))$$
$$= P_{KDE}(\mathbf{x}) + \frac{1}{2} tr\Big\{ H^{1/2} \mathcal{H}_f(\mathbf{x}) H^{1/2} \int \mathbf{z}\mathbf{z}^T K(\mathbf{z}) d\mathbf{z} \Big\} + o(tr(H))$$
$$= P_{KDE}(\mathbf{x}) + \frac{1}{2} tr(H \mathcal{H}_f(\mathbf{x}) \mu_2(K)) + o(tr(H)) \tag{3.3.38}$$

Therefore,

$$E\Big(\hat{P}_{KDE}(\mathbf{x})\Big) - P_{KDE}(\mathbf{x}) = \frac{1}{2} \mu_2(K) tr(H \mathcal{H}_f(\mathbf{x}))$$

$$
\begin{aligned}
&V(\hat{P}_{KDE}(\mathbf{x})) \\
&= V(\sum_{i=1}^{N}\alpha_i\phi_H(\mathbf{x}-\mathbf{x}_i)) \\
&= \sum_{i=1}^{N}\alpha_i^2 V(K_H(\mathbf{x}-\mathbf{y})) \\
&= \sum_{i=1}^{N}\alpha_i^2\Big[E\Big(K_H^2(\mathbf{x}-\mathbf{y})\Big)-E^2\Big(K_H(\mathbf{x}-\mathbf{y})\Big)\Big] \\
&= \sum_{i=1}^{N}\alpha_i^2\Big[\int P_{KDE}(\mathbf{y})|H|^{-1}K^2(H^{1/2}(\mathbf{x}-\mathbf{y}))d\mathbf{y}-\Big(\int P_{KDE}(\mathbf{y})|H|^{-1/2}K(H^{1/2}(\mathbf{x}-\mathbf{y}))d\mathbf{y}\Big)^2\Big] \\
&= \sum_{i=1}^{N}\alpha_i^2\Big[\int P_{KDE}(\mathbf{x}-H^{1/2}\mathbf{z})|H|^{-1/2}K^2(\mathbf{z})d\mathbf{z}-\Big(\int P_{KDE}(\mathbf{x}-H^{1/2}\mathbf{z})K(\mathbf{z})d\mathbf{z}\Big)^2\Big] \\
&= \sum_{i=1}^{N}\alpha_i^2\Big[\int(P_{KDE}(\mathbf{x})+\mathbf{z}^TH^{1/2}\mathcal{D}_f(\mathbf{x})+o(|H|^2))|H|^{-1/2}K^2(\mathbf{z})d\mathbf{z} \\
&-\Big(\int(P_{KDE}(\mathbf{x})+\mathbf{z}^TH^{1/2}\mathcal{D}_f(\mathbf{x})+o(|H|^2))K(\mathbf{z})d\mathbf{z}\Big)^2\Big] \\
&= \sum_{i=1}^{N}\alpha_i^2\Big[P_{KDE}(\mathbf{x})|H|^{-1/2}R(K)-P_{KDE}^2(\mathbf{x})+o(|H|^2)\Big] \\
&= \sum_{i=1}^{N}\alpha_i^2P_{KDE}(\mathbf{x})|H|^{-1/2}R(K)+o(\sum_{i=1}^{N}\alpha_i^2) \qquad (3.3.39)
\end{aligned}
$$

For Gaussian kernel, $\mu_2(K)=1$ and $R(K)=(4\pi)^{d/2}$. Therefore,

$$
AMISE=(4\pi)^{-d/2}|H|^{-1/2}N_\alpha^{-1}+\frac{1}{4}d^2\int tr^2\{H\mathcal{H}_p(\mathbf{x})\}d\mathbf{x} \qquad (3.3.40)
$$

where $tr(\cdot)$ is the trace operator, $\mathcal{H}_p(\mathbf{x})$ is a Hessian matrix of $p(\mathbf{x})$ and $N_\alpha=(\sum_{i=1}^{N}\alpha_i^2)^{-1}$. If we rewrite the bandwidth matrix in terms of scale β and a known structure $\mathbf{F}$, $H=\beta^2\mathbf{F}$. The optimal β is derived by minimizing AMISE with respect to the scale and the result is shown as follows:

$$\beta_{opt} = [d(4\pi)^{d/2} N_\alpha R(p, \mathbf{F})]^{-\frac{1}{d+4}} \tag{3.3.41}$$

where

$$R(p, \mathbf{F}) = \int tr^2\{F\mathcal{H}_p(\mathbf{x})\} d\mathbf{x} \tag{3.3.42}$$

Usually, this function is estimated by the plug-in method [79]. However, this plug-in method requires all observed data points and it is offline method. First of all, $R(p, \mathbf{F})$ can be written as the expectation of the derivative $\psi_r = \int p^{(r)}(\mathbf{x})p(\mathbf{x})d\mathbf{x}$. We can use the $p_s(\mathbf{x})$ to obtain the approximation:

$$p(\mathbf{x}) \approx p_s(\mathbf{x}), p^{(r)}(\mathbf{x}) = p_{\mathbf{G}}^{(r)}(\mathbf{x}) \tag{3.3.43}$$

where we approximate $p_{\mathbf{G}}^{(r)}(\mathbf{x})$, the derivative of $p(\mathbf{x})$ through the kernel density estimation:

$$p_{\mathbf{G}}(\mathbf{x}) = \phi_{\mathbf{G}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi_{\Sigma_{sj}+\mathbf{G}}(\mathbf{x} - \mu_j) \tag{3.3.44}$$

The estimate $p_G(\mathbf{x})$ is called pilot distribution, $\mathbf{G}$ is pilot bandwidth. Combined with the approximation in Eq(3.3.43), the estimation of $R(p, \mathbf{F})$ is:

$$R(p, \hat{\mathbf{F}}, \mathbf{G}) = \int tr\big(\mathbf{F}\mathcal{H}_{p\mathbf{G}}(\mathbf{x})\big) tr\big(\mathbf{F}\mathcal{H}_{ps}(\mathbf{x})\big) \tag{3.3.45}$$

To get the functional result in Eq(3.3.45) using the matrix algebra,

$$R(p, \hat{\mathbf{F}}, \mathbf{G})$$
$$= \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j \phi_{\mathbf{A}_{ij}^{-1}}(\Delta_{ij}) \times [2tr(\mathbf{F}^2\mathbf{A}_{ij}^2)(1-2m_{ij}) + tr^2(\mathbf{F}\mathbf{A}_{ij})(1-m_{ij})^2] \quad (3.3.46)$$

where

$$\mathbf{A}_{ij} = (\Sigma_{gi} + \Sigma_{sj})^{-1},$$
$$\Delta_{ij} = \mu_i + \mu_j,$$
$$m_{ij} = \Delta_{ij}^T \mathbf{A}_{ij} \Delta_{ij} \quad (3.3.47)$$

We use the empirical covariance of the sample observations $\hat{\Sigma}_{smp}$ to estimate F, i.e $\mathbf{F} = \hat{\Sigma}_{smp}$. We estimate the pilot bandwidth $\mathbf{G}$ by a multivariate normal-scale rule:

$$\mathbf{G} = \hat{\Sigma}_{smp}\Big(\frac{4}{(d+2)N_\alpha}\Big)^{\frac{2}{d+4}} \quad (3.3.48)$$

3.3.2 Compression of the Sample Model

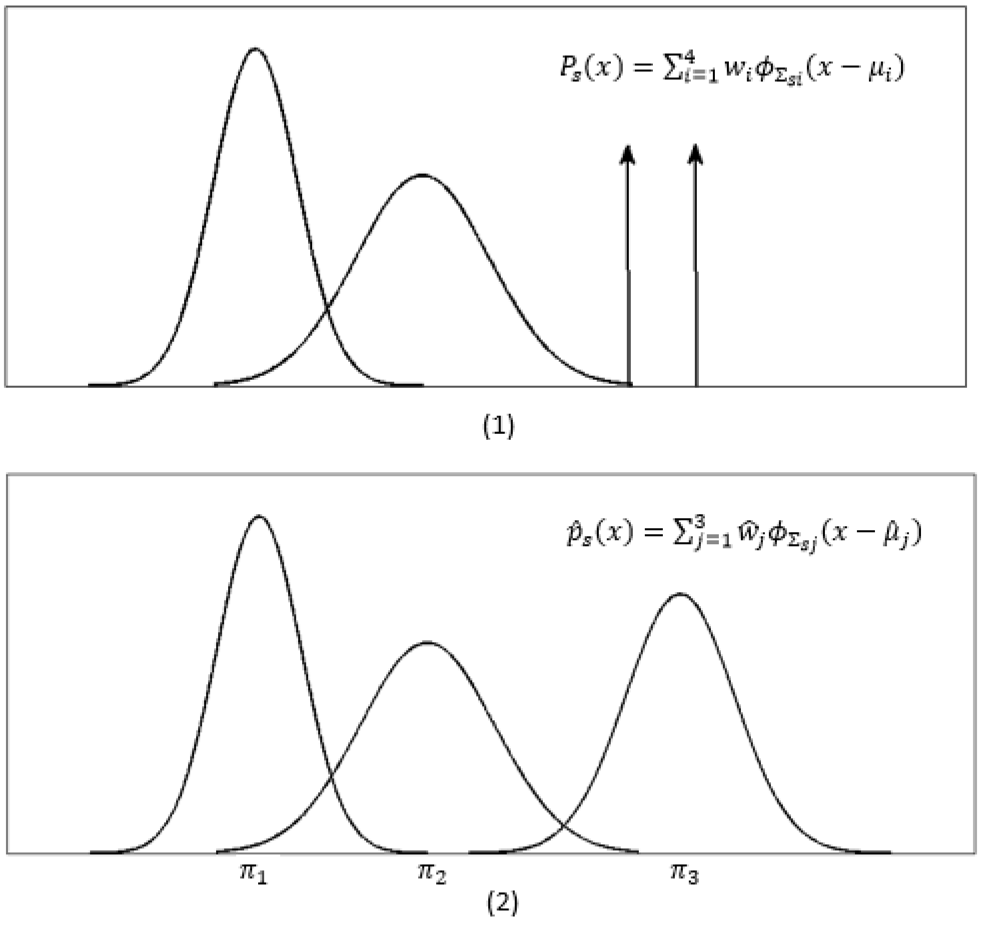


Figure 3.5: Compression of four-component sample distribution $p_s(\mathbf{x})$ (1) into three-component sample distribution (2)

This part introduces how to compress the sample distribution (Figure 3.5) and refine the original N-components sample distribution by a M-components model $\hat{p}_s(\mathbf{x})$, $M \leq N$:

$$\hat{p}_s(\mathbf{x}) = \sum_{j=1}^{M} \hat{w}_j \phi_{\Sigma_{sj}}(\mathbf{x} - \hat{\mu}_j) \tag{3.3.49}$$

Because of slow convergence for a finite number of dimensions, there is a clustering-based approach to speed up convergence, which is to identify the clusters of components in $p_s(\mathbf{x})$ and each cluster is associated with a single component. Let $\Xi(M) = \{\pi_j\}_{j=1:M}$ be a collection of disjointed sets of indexes. Therefore,

$$p_s(\mathbf{x}; \pi_j) = \sum_{i \in \pi_j} w_i \phi_{\Sigma_{si}}(\mathbf{x} - \mu_i) \tag{3.3.50}$$

The parameters of $j - th$ component are defined as:

$$\begin{aligned} \hat{w}_j &= \sum_{i \in \pi_j} w_i, \hat{\mu}_j = \hat{w}_j^{-1} \sum_{i \in \pi_j} w_i \hat{\mu}_i \\ \hat{\Sigma}_j &= \hat{w}_j \sum_{i \in \pi_j} w_i (\Sigma_i + \mu_i \mu_i^T) - \hat{\mu}_j \hat{\mu}_j^T \end{aligned} \tag{3.3.51}$$

Hence, the compression is to identify a minimal number of M and the clustering $\Xi(M)$, which constructs the lowest clustering error.

$$\hat{M} = argmin_M E(\Xi(M)), s.t. E(\Xi(\hat{M})) \leq D_{th} \tag{3.3.52}$$

where D_{th} is pre-defined threshold, $E(\Xi(\hat{M}))$ is the largest local clustering error. Here,

$$E(\Xi(\hat{M})) = max_{\pi_j \in \Xi(M)} \hat{E}(p_s(\mathbf{x}; \pi_j), H_{opt}) \tag{3.3.53}$$

Local Clustering Error

We want to approximate the component in Eq(3.3.50) with a single Gaussian function $p_0(\mathbf{x})$ using the method discussed in Eq(3.3.51). The local clustering error is defined as:

$$\hat{E}(p_1(\mathbf{x}), H_{opt}) = D(p_{1KDE}(\mathbf{x}), p_{0KDE}(\mathbf{x})) \tag{3.3.54}$$

where,

$$\begin{aligned}
&\mathbf{H}_{opt} \text{ is current estimated bandwidth}\\
&p_1(\mathbf{x}) = p_s(\mathbf{x}; \pi_j)\\
&p_{1KDE}(\mathbf{x}) = p_1(\mathbf{x}) * \phi_{\mathbf{H}_{opt}}(\mathbf{x})\\
&p_{0KDE}(\mathbf{x}) = p_0(\mathbf{x}) * \phi_{\mathbf{H}_{opt}}(\mathbf{x})
\end{aligned} \tag{3.3.55}$$

In [41], the Hellinger distance in Eq(3.3.56) is used to measure the distance between two probability distributions,

$$D_H(p_{1KDE}(\mathbf{x}), p_{0KDE}(\mathbf{x})) = \frac{1}{2}\int \left((p_{1KDE}(\mathbf{x})^{1/2} - p_{0KDE}(\mathbf{x}))^{1/2}\right)^2 d\mathbf{x} \tag{3.3.56}$$

Instead of taking Hellinger distance as a distance measurement of two probability distributions, the KullbackLeibler divergence has been applied in this study. The KullbackLeibler divergence [43] is used to measure how two probabilities diverge from each other from a second expected probability distribution, and it is defined in Eq(3.3.57).

$$D_{KL}(p_{1KDE}(\mathbf{x}), p_{0KDE}(\mathbf{x})) = \int p_{0KDE}(\mathbf{x}) log \frac{p_{1KDE}(\mathbf{x})}{p_{0KDE}(\mathbf{x})} dx \tag{3.3.57}$$

The relation between the Hellinger distance and the KullbackLeibler divergence is:

$$D_{KL}(p_{1KDE}(\mathbf{x}), p_{0KDE}(\mathbf{x})) \geq 2D_H^2(p_{1KDE}(\mathbf{x}), p_{0KDE}(\mathbf{x})) \tag{3.3.58}$$

As a sequence, $D_{KL}(p_{1KDE}(\mathbf{x}), p_{0KDE}(\mathbf{x})) \leq 1/n$ and we guarantee $D_H(p_{1KDE}(\mathbf{x}), p_{0KDE}(\mathbf{x})) \leq 1/\sqrt{n}$. However, the KullbackLeibler divergence cannot be calculated analytically for the mixture model, we use unscented transform to approximate it and check more details from Eq(3.3.59) to Eq(3.3.61). Because of the nonlinearity of Eq(3.3.57), the integral is calculated using unscented transformation that is discussed in section 2.1.3. We look for a minimal set of sample points (called sigma points) around the mean. Here, $p_{0KDE}(\mathbf{x})$ is a Gaussian mixture model in a form $p_{0KDE}(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_{\Sigma_i}(\mathbf{x} - \mathbf{x}_i)$.Therefore, the KullbackLeibler divergence is:

$$\begin{aligned} & D_{KL}(p_{1KDE}(\mathbf{x}), p_{0KDE}(\mathbf{x})) \\ & = \int p_{0KDE}(\mathbf{x}) log \frac{p_{1KDE}(\mathbf{x})}{p_{0KDE}(\mathbf{x})} dx \\ & = \sum_{i=1}^{N} w_i \int \phi_{\Sigma_i}(\mathbf{x} - \mathbf{x}_i) log \frac{p_{1KDE}(\mathbf{x})}{p_{0KDE}(\mathbf{x})} dx \end{aligned} \tag{3.3.59}$$

We let $f(\mathbf{x}) = log\frac{p_{1KDE}(\mathbf{x})}{p_{0KDE}(\mathbf{x})}$. Since distance is the only dimension of the augmented state, $L = 1$.

$$\begin{aligned}
W_i^0 &= \frac{\lambda}{1+\lambda} \\
W_i^j &= \frac{\lambda}{2(1+\lambda)} \\
\chi_i^0 &= \mathbf{x}_i \\
\chi_i^j &= \mathbf{x}_i + \sqrt{1+\lambda}(\sqrt{d\Sigma_i})_j,\ j = 1, ..., d \\
\chi_i^j &= \mathbf{x}_i - \sqrt{1+\lambda}(\sqrt{d\Sigma_i})_j,\ j = d+1, ..., 2d
\end{aligned} \tag{3.3.60}$$

We have the transformed distance in [41]:

$$D_{KL}(p_{1KDE}(\mathbf{x}), p_{0KDE}(\mathbf{x})) = \sum_{i=1}^{N} w_i \sum_{j=0}^{2d+1} W_i^j f(\chi_i^j)^j \tag{3.3.61}$$

Here, $\lambda = max(0, M - d)$ and $(\sqrt{d\Sigma_i})_j$ is the jth column of the square root of matrix Σ_i.

Compression by Hierarchical Error Minimization

A hierarchical approach can be applied to optimize Eq(3.3.52) with all possible clusters $\Xi(M)$ for the number of clusters M, which start by splitting the entire sample distribution into two sub-mixtures Eq(3.3.50) using Goldberger's K-means algorithm. Each sub-mixture is to estimate a single Gaussian $p_0(\mathbf{x})$. The hierarchical process recursively splits the tree until the largest local error is sufficiently small and satisfies $E(\Xi(M)) \leq D_{th}$.

3.3.3 Online Kernel Density Estimation

The first step of OKDE is to update the sample by combining it with the previous model and a new observation using weight $w_0 = N_t^{-1}$:

$$\tilde{p}_{s(t)}(\mathbf{x}) = (1 - w_0)p_{s(t-1)}(\mathbf{x}) + w_0\phi_0(\mathbf{x} - \mathbf{x}_t) \tag{3.3.62}$$

Let $\tilde{q}_{i(t)}(\mathbf{x}) = \phi_0(\mathbf{x} - \mathbf{x}_t)$, we have the updated sample model,

$$\begin{aligned} &\tilde{S}_{model(t)} = \{\tilde{p}_{s(t)}(\mathbf{x}), \{\tilde{q}_{i(t)}(\mathbf{x})\}_{i=1:\tilde{M}_t}\} \\ &\{\tilde{q}_{i(t)}(\mathbf{x})\}_{i=1:\tilde{M}_t} = \{q_i(\mathbf{x})\}_{i=1:M_t} \end{aligned} \tag{3.3.63}$$

Here, $\tilde{\cdot}$ denotes the update model before the compression.

The bandwidth in Eq(3.3.48) is updated in $N_{\alpha t} = \left(N_{\alpha(t-1)}^{-1}(1 - w_0)^2 + w_0^2\right)^{-1}$. Therefore,

$$\begin{aligned} &\mathbf{H}_t = \mathbf{F}[d(4\pi)^{d/2}N_{\alpha t}\hat{R}(p, \mathbf{F}, \mathbf{G})]^{-\frac{2}{d+4}} \\ &\mathbf{F} = \hat{\Sigma}_{smp} \\ &\mathbf{G} = \hat{\Sigma}_{smp}\Big(\frac{4}{(d+2)N_{\alpha t}}\Big)^{\frac{2}{d+4}} \end{aligned} \tag{3.3.64}$$

The Online Kernel Density Estimation is shown in the following Figure 3.6.

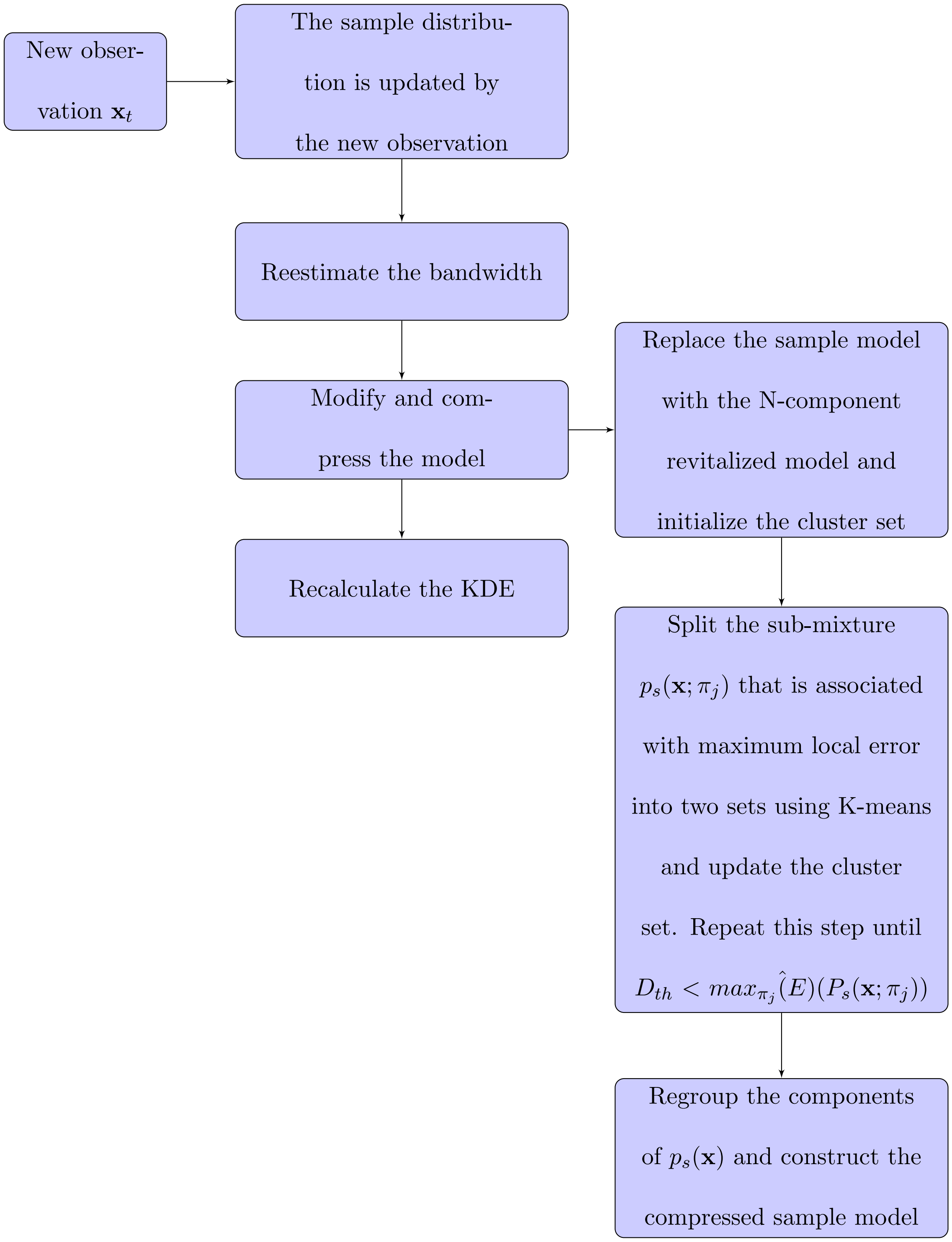

New observation $\mathbf{x}_t$
The sample distribution is updated by the new observation
Reestimate the bandwidth
Modify and compress the model
Recalculate the KDE
Replace the sample model with the N-component revitalized model and initialize the cluster set
Split the sub-mixture $p_s(\mathbf{x};\pi_j)$ that is associated with maximum local error into two sets using K-means and update the cluster set. Repeat this step until $D_{th} < max_{\pi_j}\hat(E)(P_s(\mathbf{x};\pi_j))$
Regroup the components of $p_s(\mathbf{x})$ and construct the compressed sample model

Figure 3.6: Online Multivariate Kernel Density Estimation

Algorithm 1 Update the sample model

```
1: procedure UPDATE THE SAMPLE MODEL
2:     At time $t$, the sample is defined as:
3: $S_{model(t)} = \{p_{st}(\mathbf{x}), \{q_{it}(\mathbf{x})\}_{i=1:M_t}\}$
4:     Update the effective number of observed samples:
5: $N_{t+1} = N_t + 1$ and $w_0 = 1/N_{t+1}$
6:     Update the sample distribution at time $t+1$:
7: $\tilde{p}_{s(t+1)}(\mathbf{x}) = (1 - w_0)p_{st}(\mathbf{x}) + w_0\phi_0(\mathbf{x} - \mathbf{x}_{t+1})$
8:     The sample model at time $t+1$ becomes:
9: $\tilde{S}_{model(t+1)} = \{\tilde{p}_{s(t+1)}(\mathbf{x}), \{\tilde{q}_{i(t+1)}(\mathbf{x})\}_{i=1:\tilde{M}_t}\}$
10:     where $\{\tilde{q}_{i(t+1)}(\mathbf{x})\}_{i=1:\tilde{M}_t} = \{\{q_{it}(\mathbf{x})\}_{i=1:M_t}, \tilde{q}_{\tilde{M}_t}(\mathbf{x}) = \phi_0(\mathbf{x} - \mathbf{x}_{t+1})\}$
11: end procedure
```

Algorithm 2 Update bandwidth

```
1: procedure BANDWIDTH ESTIMATION
2:     Update empirical covariance $\hat{\Sigma}_{smp}$ using Eq(3.3.50) to approximate the covariance
   from a single Gaussian
3:     Update $N_{\alpha(t+1)} = (N_{\alpha t}^{-1}(1 - w_0)^2 + w_0^2)^{-1}$
4:     Re-calculate $\hat{R}(p, \mathbf{F}, \mathbf{G})$ using Eq(3.3.46) and Eq(3.3.64)
5:     Estimate the optimal bandwidth at time $t+1$ by Eq(3.3.64)
6: end procedure
```

Algorithm 3 Compress the sample model

```
1: procedure COMPRESS THE SAMPLE MODEL
2:     According to Algorithm 1 and Algorithm 2, $\tilde{S}_{model(t+1)}$ and $\mathbf{H}_t$ is estimated
3:     Based on Section 2.3.2, re-calculate each i-th component in $\tilde{S}_{model(t+1)}$ when $\hat{E}(\tilde{q}_i(\mathbf{x}), \mathbf{H}_{t+1}) > D_{th}$
4:     Initialize the cluster set: $M = 1, \Xi(M) = \pi_1, \pi_1 = \{1, 2, ..., N\}$
5:     Do until $max_{\pi_j \in \Xi(M)} \hat{E}(p_s(\mathbf{x}; \pi_j)) < D_{th}$
6:     Select the cluster $j$ such that $\pi_j = argmax_{\pi_j \in \Xi(M)} \hat{E}(p_s(\mathbf{x}; \pi_j))$
7:     Split $\pi_j$ into two sub sets $\pi_{j1}$ and $\pi_{j2}$using the Goldberger's K-means
8:     M=M+1, $\Xi(M) = \{\{\Xi(M)\ \pi_j\}, \pi_{j1}, \pi_{j2}\}$
9:     End loop
10:    Construct each component in $\hat{p}_s(\mathbf{x})$ and its detailed model $\hat{q}_j(\mathbf{x})$ according to the clustering $\Xi(M)$
11: end procedure
```

3.4 Contribution

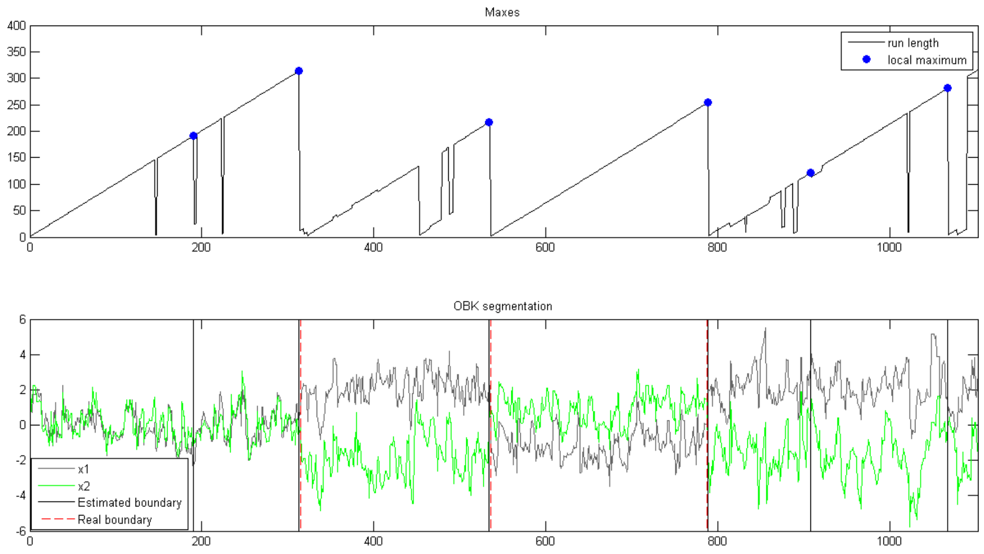


Figure 3.7: OBKS result with "run length" r_t

In this chapter, we have proposed a new online segmentation method, Online Empirical Bayesian Kernel Segmentation (OBKS), that defines the change points in a time series, which generates non-overlapped segments automatically on the timeline, as shown in Figure 3.7. Built on the Online Empirical Bayesian Segmentation algorithm, the Empirical Bayesian method requires the assumption that the observation of a segment comes from a particular multivariate distribution with a prior distribution and a likelihood function. Most of the time, the distribution is unknown. Using the technique of Online Kernel Density Estimation, we can avoid this assumption, resulting in a more flexible segmentation method. Furthermore, this new method improves the segmentation accuracy. The experimental analysis is discussed in Chapter

4. The time series segmentation is an important step for the feature selection of each segment in the next stage of this study.

4 Experiment Result of Online Empirical Bayesian Kernel Segmentation

4.1 Related Work

The Markov Chain Monte Carlo (MCMC) method is a technique for random sampling from a distribution that is established on a Markov chain, which is the target function and stationary distribution. The essential core of MCMC simulation is to generate a Markov process that has a particular transition distribution. If the simulation process is long enough, then the sample would represent the current stationary distribution.

4.1.1 Metropolis Hastings Algorithm

The Metropolis Hastings algorithm is one of Markov Chain Monte Carlo (MCMC) methods that obtains a sequence of random samples from a probability distribution, which is usually used for generating the random variables from a multi-dimensional distribution. These sample values are generated iteratively, where the distribution of the next sample would be only dependent on the current sample value. With the probability of acceptance, the candidate is used in the next iteration; however, if the candidate value is rejected, then the current value is reused in the next iteration. The probability of the candidate accepted is determined by comparing the current function value $f(\mathbf{x})$ and the candidate sample values with respect to the desired distribution $P(\mathbf{x})$.

1. Choose an arbitrary probability density $g(\mathbf{x}|\mathbf{y}) = g(\mathbf{y}|\mathbf{x})$, where $\mathbf{x}$ is a new observation and $\mathbf{y}$ is a current sample value. Usually, we take the Gaussian distribution with mean $\mathbf{y}$ as the function $g(\mathbf{x}|\mathbf{y})$.

2. To reach this, the algorithm uses a Markov process which asymptotically creates a unique stationary distribution $\pi(x)$ such that $\pi(x) = P(x)$ [65]. Each transition $x \to y$ is reversible: for every pair of states x and y, the probability of state x transitioning to state y must be equal to the probability of transitioning state y to state x,

$$\begin{aligned} &\pi(x)P(y|x) = \pi(y)P(x|y) \\ &\frac{P(y|x)}{P(x|y)} = \frac{P(y)}{P(x)} \end{aligned} \tag{4.1.1}$$

3. The acceptance distribution $A(y|x)$ is a conditional probability to accept the proposed state y. Therefore,

$$\begin{aligned} &P(y|x) = g(y|x)A(y|x) \\ &\frac{A(y|x)}{A(x|y)} = \frac{P(y)}{P(x)}\frac{g(x|y)}{g(y|x)} \end{aligned} \tag{4.1.2}$$

4. Choose an acceptance (Metropolis choice):

$$A(y|x) = \min\left(1, \frac{P(y)}{P(x)}\frac{g(x|y)}{g(y|x)}\right) \tag{4.1.3}$$

We always accept the acceptance when it is greater than 1. Assume at current time t, $X_t = x$, a random walk y_1 is create from g_1 and the acceptance probability is:

$$A_1(y_1|x) = \min\left(1, \frac{P(y_1)}{P(x)}\frac{g_1(x|y_1)}{g_1(y_1|x)}\right) \tag{4.1.4}$$

The second stage not only considers the current position of the chain but also considers what we just rejected: g_2. The acceptance probability of the second stage is:

$$A(y_2|x, y_1) = \min\left(1, \frac{P(y_2)g_1(y_1|y_2)g_2(x|y_2, y_1)[1 - A_1(y_1|y_2)]}{P(x)g_1(y_1|x)g_2(y_2|y_1, x)[1 - A_1(y_1|x)]}\right) \tag{4.1.5}$$

The acceptance probability of iterating the delying rejection (DR) process at i-th stage is [55]:

$$\begin{aligned} A_i(y_i|x, y_1, ..., y_{i_1}) = \min &\left(\frac{P(y_i)g_1(y_{i-1}|y_i)g_2(y_{i-2}|y_{i-1}, y_i)...g_i(x|y_1, ..., y_i)}{P(x)g_1(y_1|x)g_2(y_2|y_1, x)...g_i(y_i|y_{i-1}, ..., y_1, x)},\right. \\ &\left.\frac{[1 - A_1(y_{i-1}|y_i)][1 - A_2(y_{i-2}|y_{i-1}, y_i)]...[1 - A_{i-1}(y_1|y_2, ..., y_{i-1})]}{[1 - A_1(y_1|x)][1 - A_2(y_2|x, y_1)]...[1 - A_{i-1}(y_{i-1}|x, y_1, ..., y_{i-2})]}\right) \\ = \min &\left(1, \frac{N_i}{D_i}\right) \end{aligned} \tag{4.1.6}$$

If the i-th stage is reached, it means that $N_j < D_j$ for $j = 1, ..., i - 1$. The recursive formula is:

$$D_i = g_i(y_i|x, y_1, ..., y_{i-1})(D_{i-1} - N_{i-1}) \tag{4.1.7}$$

which leads to the following equation:

$$D_i = g_i(y_i|x, ..., y_{i-1})[g_{i-1}(y_{i-1}|x, ..., y_{i-2})[g_{i-2}(y_{i-2}|x, ..., y_{i-3})...$$
$$[g_2(y_2|x, y_1)[g_1(y_1|x)P(x) - N_1] - N_2] - N_3]... - N_{i-1}] \qquad (4.1.8)$$

4.1.2 Adaptive MCMC

The proposal distribution $g(\cdot|x_1, ..., x_t)$ is a Gaussian distribution with mean at the current point x_t and covariance $C_t = C_t(x_1, ..., x_t)$. We select an index $t_0 > 0$ for the length of an initial period and define it as:

$$C_t = \begin{cases} C_0, & t \leq t_0 \\ s_d cov(x_1, ..., x_{t-1}) + s_d \epsilon I_d, & t > t_0 \end{cases} \qquad (4.1.9)$$

Here, C_0 is a initial covariance based on the prior experience. s_d is a parameter only dependent on the dimension d, ϵ is a constant, and I_d is d dimensional identity matrix. As we know, the empirical covariance matrix is determined by

$$cov(x_1, ..., x_t) = \frac{1}{t}\left(\sum_{i=1}^{t} x_i x_i^T - (t+1)\bar{x}_t \bar{x}_t^T\right) \qquad (4.1.10)$$

where $\bar{x}_t = (1/t)\sum_{i=1}^{t} x_1$. Combining the above two functions, we get:

$$C_{t+1} = \frac{t-1}{t}C_t + \frac{s_d}{t}\left(t\bar{x}_{t-1}\bar{x}_{t-1}^T - (t+1)\bar{x}_t\bar{x}_t^T + x_t x_t^T + \epsilon I_d\right) \qquad (4.1.11)$$

As suggested in [25], the scaling parameter $s_d = (2.4)^2/d$. It shows, in a certain sense, that this choice optimizes the mixing properties of the Metropolis searched in the case of the Gaussian targets and the Gaussian proposals.

4.1.3 Combining Delying Rejection (DR) and Adaptive MCMC

The proposal of the first stage of DR is adapted just as in AM: the covariance for AM is computed from the points of the sampled chain, no matter at which stage these points in the sample path have been accepted. The proposal for the higher stages is always computed directly as a scaled version of the proposal of the first stage. The scale factor can be freely chosen: the proposals of the higher stages can have a smaller or larger variance than the proposal at the earlier stages. The simulation results in [25] suggest that it is more beneficial in terms of asymptotic variance reduction of the resulting MCMC estimators, which have a larger variance at the earlier stages and then reduce the variance upon rejection.

It sometimes may be difficult to start the adaptive MCMC adaptation. This process happens if the initial guess for the proposal distribution is far off from the true distribution. This situation occurs if the variance of the proposal distribution is too large, or the covariance for the proposal distribution is nearly singular. Now the DR framework provides a natural remedy for these situations: by scaling down the size of the proposals at the higher DR stages, we ensure that some points will be accepted.

4.2 Simulated Experiment Result

To compare with Online Empirical Bayesian Change Point Detection, instead of using the posterior predictive model, we apply the Online Multivariate Kernel Density Estimation to update the current model for prediction when the new data comes in. The purpose of choosing MCMC method to simulate the observations is the MCMC algorithm can sample from a probability density function within a Markov chain based desired distribution with time-related. These processes demonstrate the performance of Online Empirical Bayesian Kernel Segmentation (OBKS). Firstly, let us compare the segmentation accuracy on four different types of bi-normal variables: low covariance with low correlation, low covariance with high correlation, high covariance with low correlation and high covariance with high correlation. We want to test how covariance and correlation of the bivariate variables affect on the accuracy. Accuracy is the percentage of the correctly segmented data points. Here, OB represents the Online Empirical Bayesian detecting method, and OBKS represents the Online Empirical Bayesian Kernel Segmentation. Each combination includes three types of bi-normal variables Eq(3.2.29).

$$\text{Combination 1: low covariance with low correlation}$$
$$\mu_1 = (0,0), \Sigma_1 = \begin{bmatrix} 1.6 & -0.2 \\ -0.2 & 1 \end{bmatrix}$$
$$\mu_2 = (2,2), \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}$$
$$\mu_3 = (-1,1), \Sigma_2 = \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix} \tag{4.2.12}$$

$$\text{Combination 2: low covariance with high correlation}$$
$$\mu_1 = (0,0), \Sigma_1 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$
$$\mu_2 = (-1,1), \Sigma_2 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$
$$\mu_3 = (2,-2), \Sigma_2 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1.5 \end{bmatrix} \tag{4.2.13}$$

Combination 3: high covariance with low correlation

$$\begin{aligned}
\mu_1 &= (0,0), \Sigma_1 = \begin{bmatrix} 4 & -0.4 \\ -0.4 & 4 \end{bmatrix} \\
\mu_2 &= (-1,1), \Sigma_2 = \begin{bmatrix} 3 & 0.3 \\ 0.3 & 5 \end{bmatrix} \\
\mu_3 &= (2,-2), \Sigma_2 = \begin{bmatrix} 5 & -0.4 \\ -0.4 & 4 \end{bmatrix}
\end{aligned} \tag{4.2.14}$$

Combination 4: high covariance with low correlation

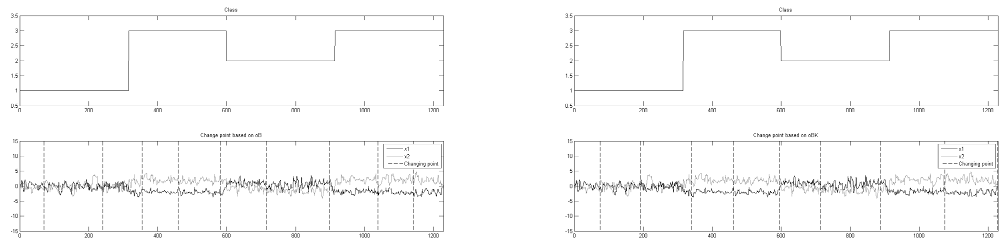
$$\begin{aligned}
\mu_1 &= (0,0), \Sigma_1 = \begin{bmatrix} 5 & 4.5 \\ 4.5 & 5 \end{bmatrix} \\
\mu_2 &= (-1,1), \Sigma_2 = \begin{bmatrix} 5 & 3.6 \\ 3.6 & 4 \end{bmatrix} \\
\mu_3 &= (2,-2), \Sigma_2 = \begin{bmatrix} 3 & 2.3 \\ 2.3 & 3 \end{bmatrix}
\end{aligned} \tag{4.2.15}$$

Based on the segmentation accuracy result shown in Table 4.1, OBKS has higher accuracy and OBKS is the better choice as an online segmentation algorithm. Even if the observation has a relatively large variance and the time series fluctuates heavily, OBKS can still adapt and fit the model properly. Especially when these bivariate

Table 4.1: Detection accuracy of OB and OBKS on four different simulated data

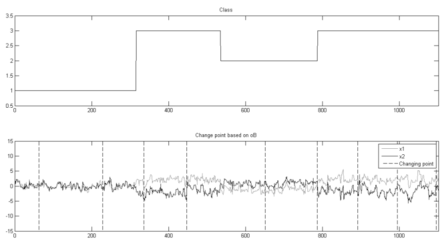
	low_cov low_corr	low_cov high_corr	high_cov low_corr	high_cov high_corr
OB	0.9414	0.9002	0.9147	0.8839
OBKS	0.9552	0.9995	0.9238	0.9861

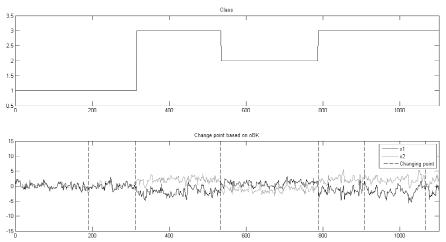
variables have a stronger correlation, OBKS still clearly generates a better-segmenting result. For example, by checking the binormal distribution with the lower covariance and higher correlation combination shown below in Figure 4.2, the OB method has more sensitivity for updating the "run length", and therefore the estimated "run length" using this method is more unstable. The OBKS method already includes the adaptive Online Multivariate Kernel approach, which results in a steadier and stronger capability for updating "run length".



,

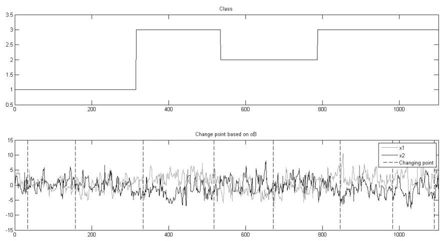
Figure 4.1: Detection accuracy of OB and OKDE on low covariance and low correlation bi-normal simulation data

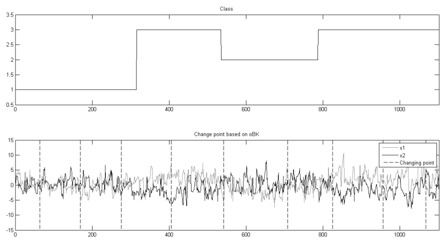




,

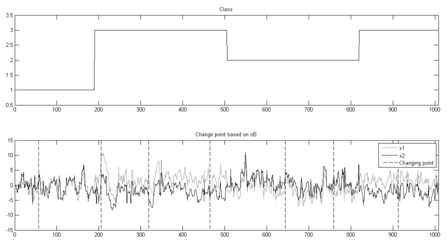
Figure 4.2: Detection accuracy of OB and OKDE on low covariance and low correlation bi-normal simulation data

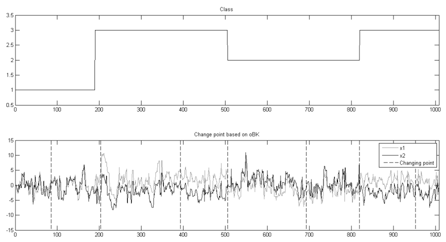




,

Figure 4.3: Detection accuracy of OB and OKDE on high covariance and low correlation bi-normal simulation data





,

Figure 4.4: Detection accuracy of OB and OKDE on high covariance and high correlation bi-normal simulation data

In addition, if we only consider the capability of estimating the probability density by using the OB method and the OBKE method, we train on four different types of observations: one-dimension skew, mix_gaussian, sine wave, and bi_norm simulation data regarding four different sizes of samples: N=100, N=500, N=1500 and N=3000. The testing criterion is the negative maximum likelihood estimation and mean square error (MSE).

Table 4.2: MLE and MSE of OB and OKDE on four different simulated data

		N=100		N=200		N=1000		N=3000	
		MLE	MSE	MLE	MSE	MLE	MSE	MLE	MSE
skew	oB	4.4435	-	4.4545	-	4.3456	-	4.384	-
	oKDE	2.6386	-	2.4764	-	2.3821	-	2.3818	-
sin wave	oB	2.8741	9.1706	2.9233	134.789	2.9406	76.2118	2.9139	43.7261
	oKDE	0.8989	8.6324	0.8676	133.5121	0.7056	74.2366	0.7917	42.5037
bi normal	oB	3.6527	0.0044	3.559	0.0045	3.665	0.0036	3.6781	0.0037
	oKDE	2.8446	0.0016	2.6854	0.0004	2.7959	0.0005	2.878	0.0005
mix gau	oB	3.9226	0.0018	4.0485	0.0022	4.063	0.002	4.0431	0.0018
	oKDE	3.0911	0.0004	3.2065	0.0003	3.2109	0.0002	3.1349	0.0002

In Figure 4.2, the OBKE algorithm has a smaller negative maximum likelihood estimation, and a smaller MSE on four different simulations, which implies the OBKE algorithm has more power to estimate a density than the OB method.

4.3 Empirical Observation Experiment Result

[52] and [46] provide an overview of the human activity recognition process and discussion of the segmenting methods. The data source is provided by [63], where the experiments were carried out with a group of 30 volunteers. They performed a pro-

tocol of six basic activities: three static patterns (standing, sitting, lying) and three dynamic patterns (walking, walking_downstairs, walking_upstairs). All of the participants were wearing a smartphone (Samsung Galaxy S II) on the waist while executing the experiment. The embedded accelerometer in a smartphone generated the captured 3-axial linear acceleration at a constant rate of 50Hz. Randomly chosen 200s worth of observations of a few volunteers show the tri-axial observations of the six activities in Figure 4.5. It is relatively easy to distinguish the dynamic activities and the static activities because the dynamic activities have a stronger fluctuation than the static activities. However, it is a challenge to identify walking, walking_upstairs and walking_downstairs in the dynamic group, as well as identifying sitting, standing and lying in the static group. Also, these volunteers generated their particular wave stream so that the time series observations are quite different with each other even when they performed the same action. For instance, in Figure 4.5_walking, the z-axial data around the first 80s is less fluctuating than the z-axial data from 80s to 120s. On the other hand, the observations of different activities exhibited from different individuals could be same. Therefore, the combination of the sensor data from all of these volunteers is difficult to detect due to the differences of the pattern. The reason of training on the combined data is that the prototype created by such a training set can be applied so that the user is not required to record the start and stop time for the particular activities.

First of all, let us check the confusion matrix result of automatically segmenting the observations using the Online Empirical Bayesian Kernel Segmentation algorithm.

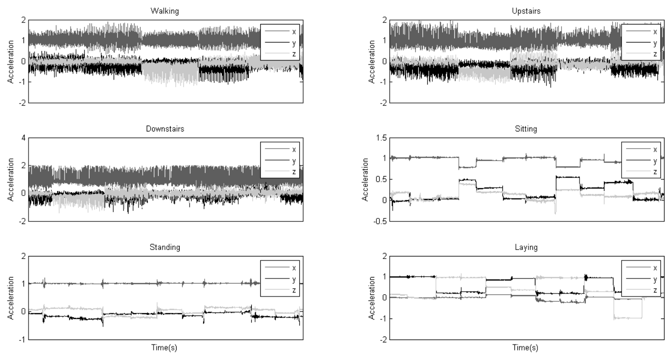


Figure 4.5: Acceleration plot of six activities

The result is shown in Table 4.3. Those errors are caused by the segmenting bias and delays with the real boundaries. Here, we use the majority vote to label the class of each segment.

Table 4.3: Online Empirical Bayesian Kernel Segmentation Confusion Matrix

	Walking	Walking Upstairs	Walking Downstairs	Sitting	Standing	lying
Walking	1	0	0	0	0	0
Walking Upstairs	0	0.9601	0.0399	0	0	0
Walking Downstairs	0	0.0451	0.9549	0	0	0
Sitting	0	0	0	0.9533	0.0265	0.0087
Standing	0	0	0	0.0477	0.9415	0
lying	0	0	0	0.0078	0.0058	0.9749

The average error rate of bias and delay in Table 4.3 is 3.55%, which equates to 1 minus the overall accuracy. This result shows that this algorithm can automatically and efficiently detect the change points and find activity in the time interval. To compare with the Online Empirical Bayesian Kernel Segmentation method, we use the Sliding Window and Bottom-up (SWAB) online segmentation algorithm [37] as an optional choice. The SWAB is to process the the Bottom-up algorithm during a large enough sliding window that includes a few segments. Unlike a Empirical Bayesian method, the SWAB requires defining original minimal segmentation length and a final merged number of segments. In Table 4.4, we display the overall accuracy based on the different scales of minimal length and the number of segments. The accuracy increases as long as we increase the number of segments and the minimum length. However, increasing both criteria will result in a meaningless segmentation. Therefore, it is uncertain which is the best option. Compared with that, OBKS has a less pre-defined requirement, and hence it is more desirable to fit the observations.

Table 4.4: SWAB overall accuracy

	min len = 2	min len = 5	min len = 10	min len = 20	min len = 30
num seg= 25	0.8417	0.8444	0.8494	0.8793	0.8991
num seg= 50	0.8854	0.8914	0.9022	0.9329	0.9534
num seg= 100	0.9227	0.9386	0.9514	0.9688	0.9931
num seg= 200	0.9658	0.9663	0.9739	0.9992	0.9981

Finally, we use the proposed OBKS method on the human pattern dataset to demonstrate the segmenting capability and Table 4.5 records the confusion matrix result.

The result of detection delay is shown in Table 4.6, which calculates the time gap (time difference) between the true change points and the experimental change

Table 4.5: Online Empirical Bayesian Kernel Segmentation Confusion Matrix

	Walking	Walking Upstairs	Walking Downstairs	Sitting	Standing	lying
Walking	1	0	0	0	0	0
Walking Upstairs	0	0.9713	0.0287	0	0	0
Walking Downstairs	0	0.0449	0.9551	0	0	0
Sitting	0	0	0	0.9645	0.0112	0.0243
Standing	0	0	0	0.0583	0.9417	0
lying	0	0	0	0.0146	0.0043	0.9811

point generated by the OBKS method. The average detection delay is computed as taking an average of all time gaps that are 65 observations, which means the detecting delay is 1.3s.

Table 4.6: Detection delay using OBKS method

	Average detection delay	Minimum detection delay	Maximum detection delay
Observations	65	0	685
Time	1.3s	0s	68.5s

4.4 Comparing OBKS with HMM

Recall the previous work discussed in Chapter 2, and the data source is provided by Dr.Tu's lab. It is a tri-dimensional sensor data with a 50Hz sampling rate. The sensor data include sitting, standing, walking_upstairs, walking_downstairs, walking and jogging six activities. However, the label of each activity is unknown. Comparing the change point detecting result by OBKS method in Figure 4.6 with HMM state estimation result mentioned in Chapter 2, the OBKS algorithm does an excellent performance.

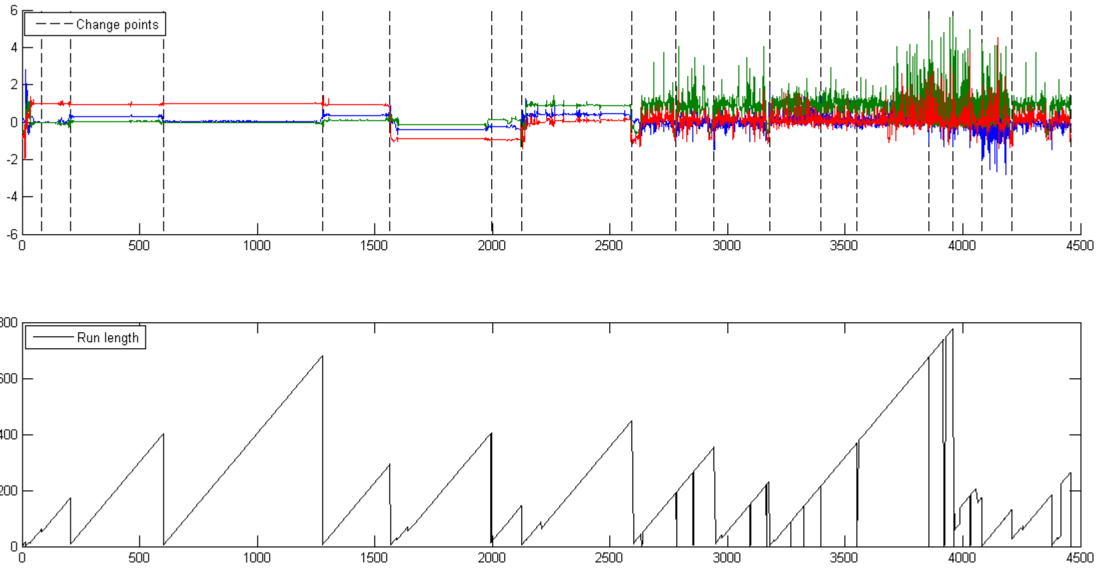


Figure 4.6: Acceleration plot of six activities

4.5 Contribution

In the previous chapter, we described a new online segmentation method, Online Empirical Bayesian Kernel Segmentation (OBKS). In this chapter, we tested the segmenting performance of OBKS on the MCMC simulated observations and the empirical data set, which showed a better result compared with the Bottom-Up Sliding

Window and Online Empirical Bayesian Detecting method. Accuracy is used as the criterion of testing the performance that is the proportion of the correctly segmented data point. In addition, we evaluated the performance of density estimation through comparing the Online Kernel Density Estimation (OKDE) and the Online Empirical Bayesian Estimation (OBE). The OKDE resulted in lower negative MLE values and lower MSE values, which implies that the OKDE method has a better performance of density estimation. Hence, these experiments verify our proposed method.

5 Two-layer Classification and Experiment Result

5.1 Classification Algorithm

5.1.1 k-Nearest Neighbor

K-nearest neighbor classification is a style of instance-based learning or lazy learning, which is one of the simplest marching learning algorithms. This method is used widely in pattern recognition as a non-parametric estimation approach. This process classifes an object by majority voting locally among the k most close training samples, so that learning is not implemented until classification occurs. The closeness is measured by a distance, such as Euclidean distance, Manhattan distance, and Minkowski distance. Here, $\mathbf{x}$ is d-dimensional observation, $i \neq j$.

$$\begin{aligned} &\text{Euclidean } \sqrt{\sum_{k=1}^{d}(\mathbf{x}_{ik} - \mathbf{x}_{jk})^2} \\ &\text{Manhattan } \sum_{k=1}^{d}|\mathbf{x}_{ik} - \mathbf{x}_{jk}| \\ &\text{Minkowski } \left(\sum_{k=1}^{d}(|\mathbf{x}_{ik} - \mathbf{x}_{jk}|)^q\right)^{1/q} \end{aligned} \tag{5.1.1}$$

k is a pre-defined constant, and different k values result in different classification targets. The cross-validation is used to identify the optimal k value.

5.1.2 Decision Tree

A decision tree displayed in Figure 5.1 builds a tree which recursively partitions the features into a set of rectangles by a particular splitting rule. In this structure, the leaves represent the class targets, and the branches represent the conjunctions of features that lead to those class labels. The splitting criteria that searches for the best tree construction minimizes the impurity-based criteria, such as the misclassification error rate, the Gini index, and the information gain. Let $\hat{P}_k$ represent the proportion of the observations belonging to a class $k, k = 1, ..., K$.

$$\begin{aligned} &\text{Classification error rate: } E = 1 - max_k(\hat{P}_k) \\ &\text{Gini index: } G = 1 - \sum_{k=1}^{K} \hat{P}_k^2 \\ &\text{Information gain } D = -\sum_{k=1}^{K} \hat{P}_k^2 log_2(\hat{P}_k^2) \end{aligned} \tag{5.1.2}$$

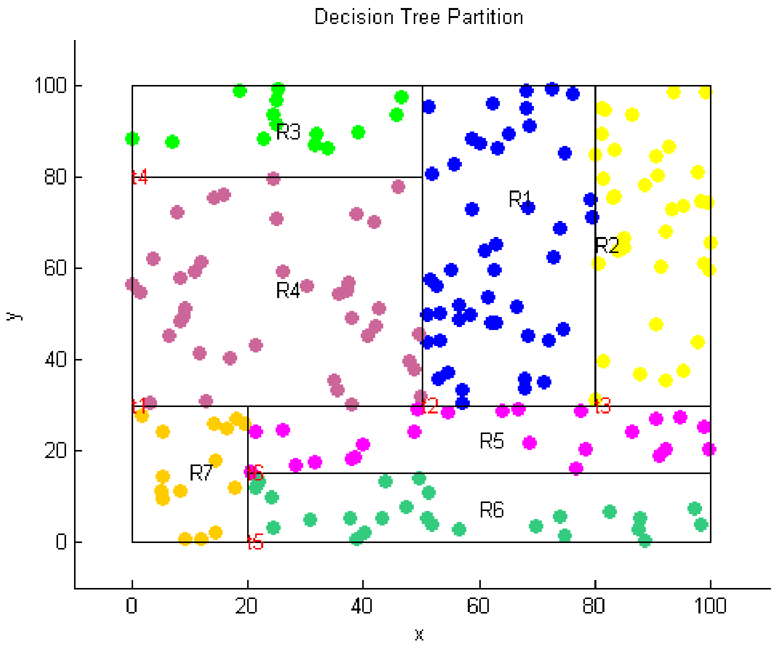


Figure 5.1: Decision tree partition

Many researchers discuss this algorithm, such as ID3 [61], C4.5 [67], and CART [50]. One single tree usually leads to a high variance based on different learning models; ensemble learning is motivated by the idea of including multiple decision trees. There are two directions to create the multiple trees: parallel tree (Bagging [11], Random Forrest [31, 12]) and serial tree (Boosting [68, 22]). Bagging is also known as a bootstrap aggregation, which was introduced by Breiman in 1994 in an early version of [11]. Parallel method displayed in Figure 5.2 generates n decision trees at the same time and generates n target values based on the different trees. The final class of an individual is defined by taking the majority vote over all trees in the forest. The difference between Bagging and Random Forest is the Bagging method trains on all of the features and Random Forest only considers the randomly selected sub-features, which avoids the correlations between these features. Directly training multiple trees on a single training set would give the most strongly correlated trees. Random Forest improved a decision tree with reduced over-fitting.

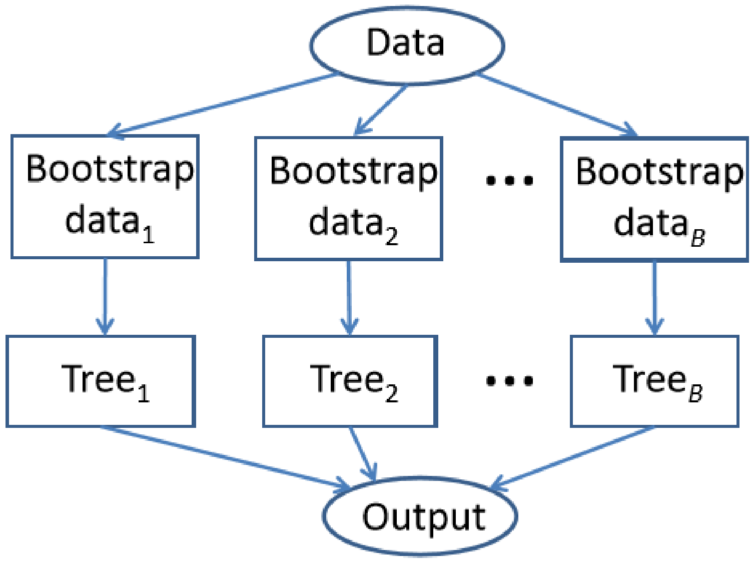


Figure 5.2: Structure of Bagging

Unlike the Bagging tree, the parallel tree is designed to improve the stability and accuracy of a decision tree. The Boosting method, as shown in Figure 5.3, is the serial tree. It also has reduced variance and avoids the over-fitting issue. The reason of reducing variance is it adaptively updates its weight instantly to correct the previous mis-classifier. The goal is to generate a set of weak learners that create a single strong learner.

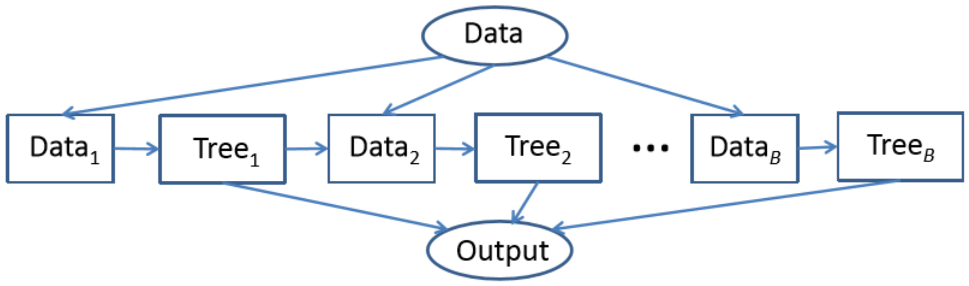


Figure 5.3: Structure of Boosting

5.1.3 Naive Bayes

The Naive Bayes classifier [66] is a probability model based on applying Bayes theorem with the naive assumption of independence between every pair of features. A naive Bayes classifier considers these features independently of each other as they contribute to probability. The advantage of the Naive Bayes classifier is that it only requires a small amount of training data to estimate the means and variances of the variables necessary for classification. Let training examples x with attributes $\{a_1, a_2, ..., a_p\}$, and assuming the attributes are conditionally independent, the Naives Bayes classifier is defined as:

$$\begin{aligned} y_{NB} &= argmax_{y_j \in Y} P(y_j | a_1, a_2 ...., a_p) \\ &= argmax_{y_j \in Y} \frac{P(a_1, a_2 ...., a_p | y_j) P(y_j)}{P(a_1, a_2 ...., a_p)} \text{ (Bayes rule)} \\ &= argmax_{y_j \in Y} P(a_1, a_2 ...., a_p | y_j) P(y_j) \text{ } (P(a_1, a_2 ...., a_p) \text{ is constant)} \\ &= argmax_{y_j \in Y} P(y_j) \prod_{i=1}^{p} P(a_i | y_j) \text{ Independent assumption} \end{aligned} \tag{5.1.3}$$

where the estimated target is given by the target associated with maximum probability.

5.1.4 Neural Network

Artificial Neural Network (ANN) is a computation process that depends on an interconnected group of nodes which calculates in a way similar to that of a biological brain. This approach is first introduced by Warren McCulloch and Walter Pitts in [53] and combines mathematics and threshold logic algorithm. Due to the complexity and flexibility of ANN, it is broadly applied to a variety of topics, such as pattern recognition and writing recognition.

The ANN system shown in Figure 5.4 includes at least three layers, one is for input node, at least one is for hidden node, and the last one is for output node. Different nodes correspond to different weights and associate with a transfer function. The task is to minimize the mean square error by searching for an optimal set of weights.

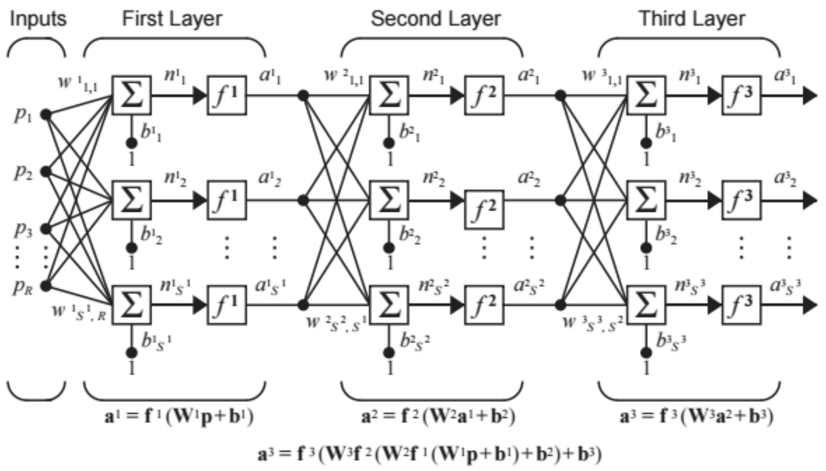

Inputs
First Layer
Second Layer
Third Layer
$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)$
$\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2)$
$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$
$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$

Figure 5.4: Structure of ANN

5.2 Feature Extraction and Feature Selection

In machine learning and statistics, the feature extraction starts from an initial set of measured data and builds derived values (features), which aim to extract more information and key features. The types of features widely used so far are categorized as following four types of features [7].

Time Domain Features: mean, variance, mode, maximum, minimum, etc.;

Frequency Domain Features: spectral energy, spectral entropy, Fourier transforms coefficients, etc.;

Time-Frequency Domain Features: Wavelet coefficients;

Heuristic Features: Signal Magnitude-Area, Signal Vector Magnitude, Inter-axis Correlation, etc.

When the input data is too large to be processed by an algorithm or some of the variables are highly related to each other, a process called feature selection is considered that transforms the entitled features (input data) into a reduced set of features for classifying. This procedure can reduce the over-fitting issue and the irrelevant results that produce more efficient classification training and testing.

5.3 Feature Selection for Human Pattern Observation

In this experiment, instead of extracting features from those segments with fixed length, such as 100 observations and 1000 observations, we consider extracting the features from the entire time series segment constructed automatically by the Online Empirical Bayesian Kernel Estimation (OBKE) algorithm. One of the reasons for considering the features on a whole segment is less computation time for each feature. Another reason is to classify the whole sequence rather than classifying the subsequence because the subsequences contain less information of a pattern. Recall the human pattern recognition experiment mentioned in Chapter 4, there is an important measurement for the feature selections of the three-dimensional data (x_i, y_i, z_i): the distance of a observation to the original point $(0, 0, 0)$ called magnitude, which is defined as $d_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$. The features we use here follow the eight criteria. We consider the time frame $t_i, t_{i+1}, ..., t_j$ of certain segment, the length is $j - i + 1$:

1. **Average** of (x_i, y_i, z_i, d_i), such as $\bar{x} = \frac{1}{j-i+1} \sum_{t=i}^{t=j} x_t$;

2. **Standard deviation** of (x_i, y_i, z_i, d_i), such as $\sigma_x = \sqrt{\sum_{t=i}^{t=j} \frac{(x_t - \bar{x})^2}{j-i+1}}$;

3. **Average of local maximum** of (x_i, y_i, z_i, d_i), such as, assume there are k local maximums for x-axis, $\bar{x}_{max} = \frac{1}{k} \sum x_{max}^k$;

4. **Average of time difference between two consecutive local maximums** of (x_i, y_i, z_i, d_i), such as diff$x_{max} = \frac{1}{k-1} \sum x_{max}^{k_i} - x_{max}^{k_j}$;

5. **Skewness** of (x_i, y_i, z_i, d_i), such as Skew$x = \frac{\frac{1}{j-i+1} \sum (x_t - \bar{x})^2}{[\frac{1}{j-i} \sum (x_t - \bar{x})^2]^{3/2}}$

6. **Kurtosis** of (x_i, y_i, z_i, d_i), such as Kurt$x = \frac{\frac{1}{j-i+1} \sum (x_t - \bar{x})^4}{[\frac{1}{j-i+1} \sum (x_t - \bar{x})^2]^2 - 3}$

7. **Pearson Correlation Coefficient** among x-axis, y-axis and z-axis, such as $\rho_{xy} = \frac{cov(x,y)}{\sigma_x \sigma_y}$

8. **Inter Quantile** Range of (x_i, y_i, z_i, d_i), such as $iqr_x = Q_{x3} - Q_{x1}$,Q_{x1} is the first quantile of x-axis and Q_{x3} is the third quantile of x-axis.

The observations can be approximated as a periodic wave, and the repetitive peak can be regarded as one of the characteristics that are applied to distinguish different axes of different activities [44]. Since the wave of the dynamic activities repeats more quickly than the static activities, the time gap between two consecutive peaks of the dynamic activities is shorter than the static activities. The time gap is also used as the index to distinguish the activities.

The criterion used to select the features is the variable importance. There are two types of variable importance measure methods: out-of-bag error rate and Gini index. In Table 5.1, it shows the variable contributions by the Gini index. A variable is not significant if the Gini index is close to 0; a variable gives remarkable contribution for classification if the Gini index is much higher compared to other variables, such as the importance of the average of the x-axis is 30.0708. That is the reason why we choose this feature.

5.4 Classification Comparison

Considering the experimental dataset, after processing the time series segmentation algorithm, and we train the two-layer classification. The first layer classifies the activities into two key categories: the dynamic activities (walking, walking_upstairs, walking_downstairs) and the static activity (standing, sitting, lying). The features used for the first layer classification only include the first four criteria of d_i: average, standard deviation, the average of local maximum and the average of the time gap between two consecutive local peaks. These criteria of d_i are the main factors to distinguish a dynamic pattern and a static pattern. For the second layer classification, there are two separated classification processes. One classifier trains and tests only on the dynamic activities resulting in three classes: walking, walking_upstairs, and walking_downstairs. Another classifier trains and tests on the static activities that also lead to three classes: sitting, standing, and lying. Here, we apply Random Forest as the classification method, which is the most widely used classifier and it is suitable for different types of data [70]. The two-layer classification algorithm is displayed in Figure 5.5.

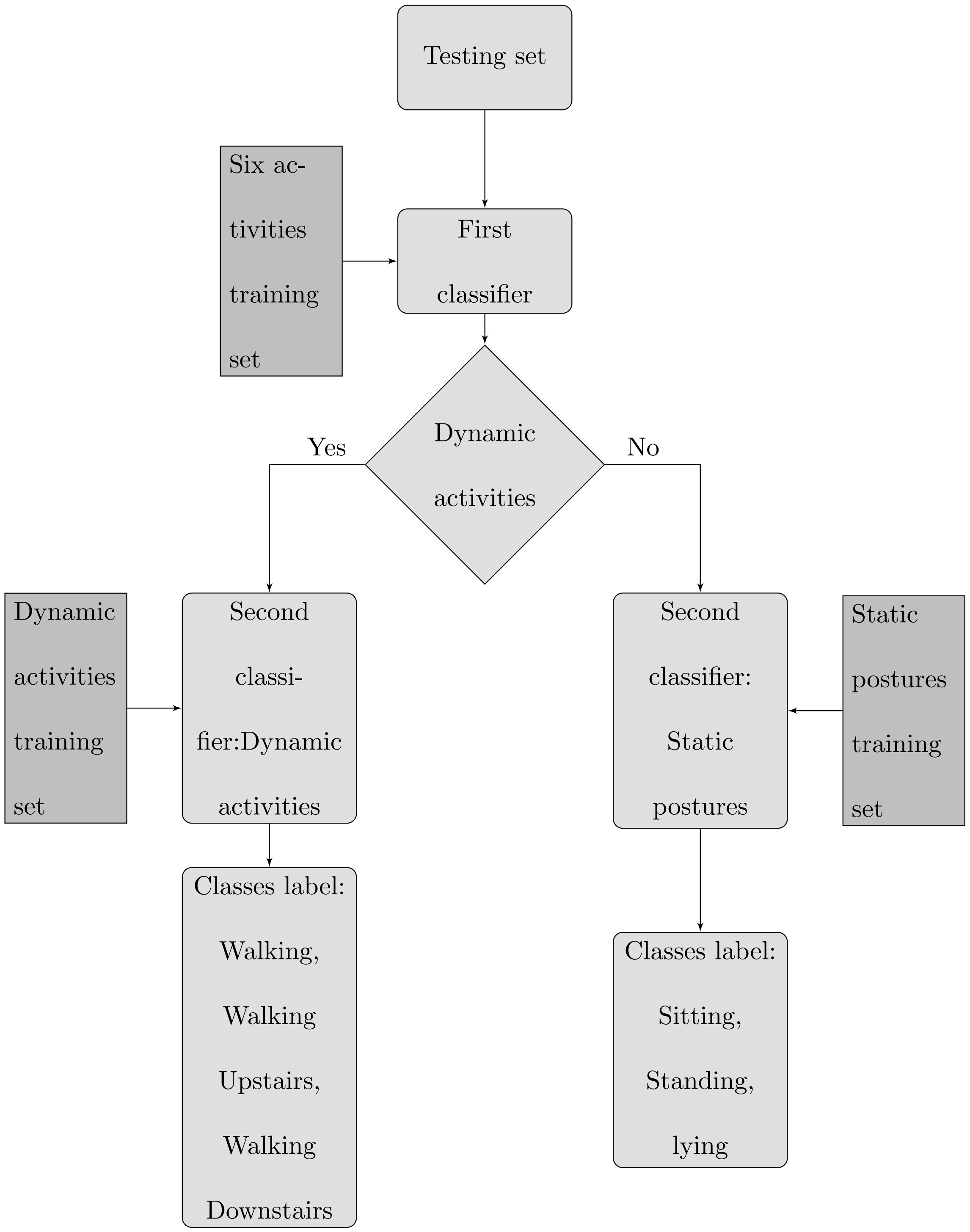

Testing set
Six activities training set
First classifier
Dynamic activities
Yes
No
Dynamic activities training set
Second classifier:Dynamic activities
Second classifier: Static postures
Static postures training set
Classes label: Walking, Walking Upstairs, Walking Downstairs
Classes label: Sitting, Standing, lying

Figure 5.5: Two-layer Classification

The human pattern dataset was randomly split into two sets, where 70% of the data was collected as the training data, and the other 30 % are the test data. The training set is applied to train the two-layer Random Forest to get the prototype model. Subsequently, the model is applied to classify the patterns (segments) generated by the Online Empirical Bayesian Kernel Segmentation discussed in Chapter 3, and produces a class label as the output. Comparing the one layer and two-layer classification method: SVM, kNN, Boosting, and Random Forest in Table 5.2, there is a significant difference between the one layer and two-layer associated with kNN and Random Forest on this experimental data. For example, the overall accuracy by the one layer Random Forest is 0.8484, and the overall accuracy using the two-layer Random Forest is 0.9239.

We compare the accuracy results of kNN, SVM, and Boosting with Random Forest, as shown in Table 5.2 and Figure 5.6. The overall accuracy of Boosting and Random Forest are very close and both are higher than the other two algorithms. Checking the accuracy for each activity in Figure 5.6, the results generated by Boosting and Random are better than the others, except for walking_upstairs. SVM performs the best classification result on the walking_upstairs pattern. Taken as a whole, Boosting and Random Forest are the best choices. Nonetheless, the time complexity of Boosting $O(ndKlog(n))$ is higher than the time complexity of Random Forest $O(ndlog(n))$. Here, n is the number of the observations, d is the number of the features, and K is the depth. We have displayed the actual running time for each classification method in Table 5.3 after extracting out the meaningful features. The SVM cost the least of process time, the time consumption of boosting is almost six times of the random forest's. Consider the task of human pattern detecting, not only accuracy is important, but less computation time is also a prominent criterion. Therefore, the Random Forest is the better way to handle our data.

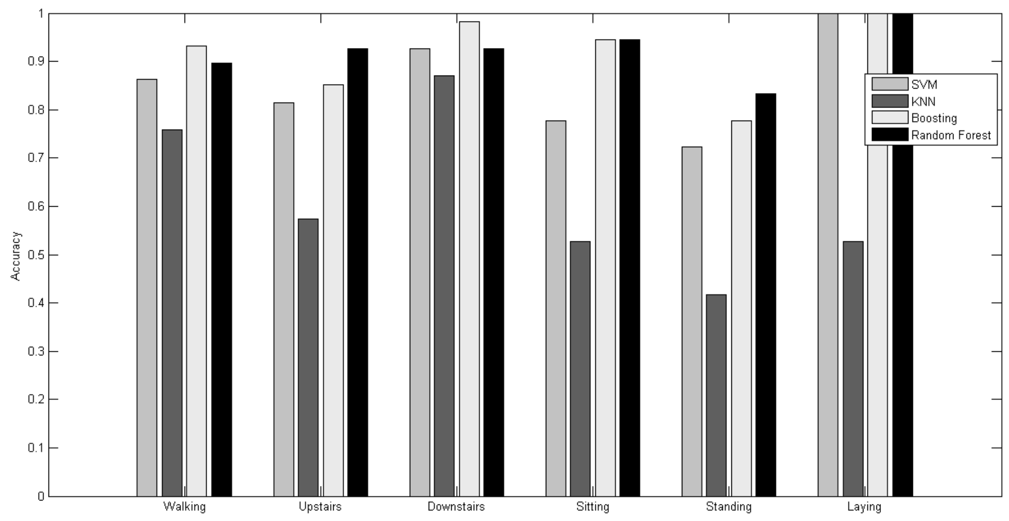


Figure 5.6: Accuracy of six activities using four classification methods

5.5 Contribution

In this section, a two-layer Random Forest classification method has been proposed, which considers using different features for different layers. After developing this new segmentation method discussed in Chapter 3, the procedure of feature selection is performed, followed by classification. In [63], Jorge-L Reyes-Ortiz and Luca Oneto proposed 561 features for the human pattern recognition application in the classification step. However, not all features are necessary; some of them are redundant. We use fewer features than other researchers to reduce time complexity in the classification step. In order to choose the most appropriate classification method for certain human activity sensor data, we have analyzed Random Forest, Boosting, SVM, and kNN experimental results for the case of defining the best fitting classification algorithm.

Table 5.1: Variable importance

Variable	Gini	Variable	Gini	Variable	Gini
Avgdiff_d	17.3045	Avgdiff_z	2.6665	Entropy_z	11.566012
avgmax_d	19.5482	avgmax_z	7.8080	Corr_xy	7.6513
avg_d	17.6678	skew_d	6.0167	Corr_xz	5.1707
std_d	28.2118	skew_x	5.4729	Corr_yz	4.5131
avg_x	30.0708	skew_y	5.8925	iqr_d	15.9902
avg_y	20.4652	skew_z	4.6519	iqr_x	7.4886
avg_z	8.0008	kurt_d	8.7959	iqr_y	5.8003
std_x	14.7185	kurt_x	4.3166	iqr_z	5.2102
std_y	2.6321	kurt_y	5.3447	energy_d	35.8213
std_z	2.3531	kurt_z	6.5595	energy_x	33.1778
Avgdiff_x	8.5406	Entropy_d	35.2387	energy_y	12.8090
avgmax_x	19.8906	Entropy_x	18.0300	energy_z	6.2494
Avgdiff_y	3.5887	Entropy_y	30.9861	-	-
avgmax_y	9.5369	-	-	-	-

Table 5.2: Overall accuracy of six activities using four classification within one layer and two-layer approach

	SVM	kNN	Boosting	Random Forest
One layer	0.8237	0.5922	0.8594	0.8484
Two-layer	0.8237	0.6116	0.8585	0.9238

Table 5.3: Experimental time consuming for each classification method

	SVM	kNN	Boosting	Random Forest
Time consuming	4.7s	6.3s	1317.31s	20.69s

6 Human Pattern Recognition System

6.1 Human Pattern Recognition System

In this section, combining the knowledge from Chapter 3 to Chapter 5, we present a Human Activity Recognition (HAR) system in Figure 6.1 using the sensor data generated from the tri-axial accelerometer that was built into the smartphone [64]. The six daily life activities conducted here are: walking, walking_upstairs, walking_downstairs, sitting, standing, and lying. To deal with the HAR problem, first of all, we must split the entire time series interval into disjointed segments because the raw data cannot be directly applied to the classification algorithm. Next, we extract the essential features (mean, standard deviation, peaks, etc.) from each segment, and use these features for the classification algorithm, such as decision tree. Finally, we train and test the identifying process. Because of the limitation of a cell phone's battery and CPU, we cannot process the training step on a mobile phone. Generally, the training process is either already learned on a computer or happening on cloud computing through the Internet. However, since technology is improving with time, more and more types of cell phones are allowed to run the training algorithm [70]. Therefore, the user can collect the data and identify the activities on their phone. Training classification on a cell phone is more convenient for a user when a user wants to adjust the activity training model to fit his/her activities or when the Internet is not available. In addition, it is also desirable to build more efficient and less complex learning algorithms so that the training procedure can be developed on a cell phone. For the classification methods, there are two types of the algorithms: online learning and offline learning ([33],[56]). Compared with the offline learning methods, the online learning algorithms can be applied to real-time training data because it is able to adapt the model as new data being collected.

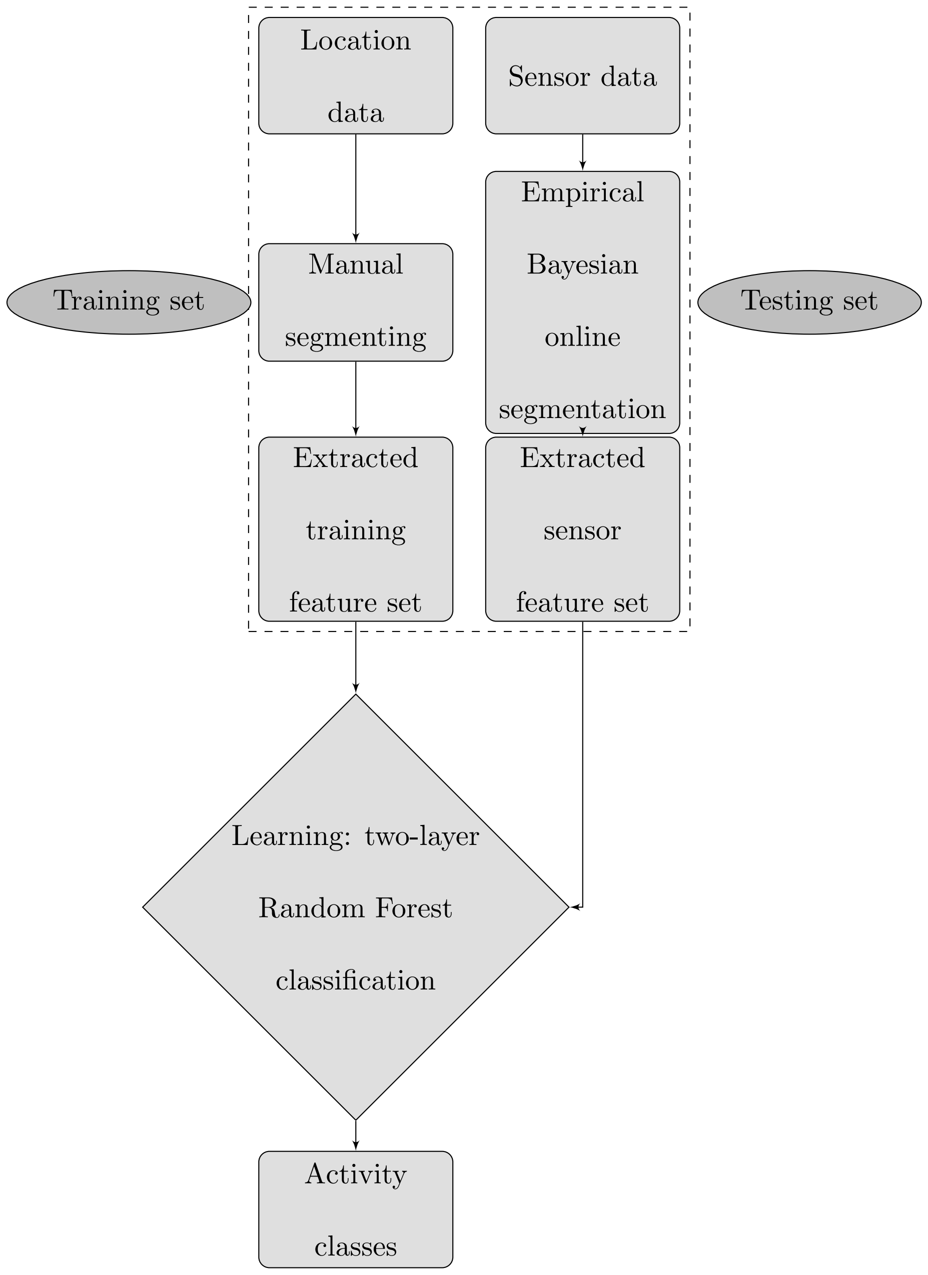

Location data
Sensor data
Empirical Bayesian online segmentation
Manual segmenting
Training set
Testing set
Extracted training feature set
Extracted sensor feature set
Learning: two-layer Random Forest classification
Activity classes

Figure 6.1: Online Activity Recognition Process

6.2 Empirical Experiment

Table 6.1 shows the testing accuracy of the two-layer Random Forest followed by Online Empirical Bayesian Kernel Segmentation. The overall accuracy is 91.4% for combining the six patterns. The accuracy of the static postures and the dynamic activities is 100 %. Based on the overall accuracy of the two groups, we can 100% detect whether a person moves or not. From the estimated change points, we can estimate the time length for a person being active or sedentary.

Table 6.1: Online two-layer Classification Confusion Matrix

	Walking	Walking Upstairs	Walking Downstairs	Sitting	Standing	lying
Walking	26	3	0	0	0	0
Walking Upstairs	3	50	1	0	0	0
Walking Downstairs	0	3	51	0	0	0
Sitting	0	0	0	34	2	0
Standing	0	0	0	5	30	1
lying	0	0	0	0	0	36

6.3 Contribution

In the presented research, we develop a human pattern recognition system using a cell phone accelerometer that generates the tri-dimensional time series observation, which combines Online Empirical Bayesian Kernel Segmentation, feature extraction, feature selection, and two-layer Random Forest. We create these class prototypes using training data. By considering the properties of cell phones, such as the battery limitation, we decide the training step would not happen on a smartphone. The performance is illustrated by testing on the smartphone accelerometer sensor data, which is generated from the smartphone with a built-in accelerometer when the 31 volunteers carried it on their waist to act the six states: sitting, standing, lying, walking, walking_upstairs, and walking_downstairs. This process shows the overall accuracy is 91.4%. In addition, it can detect lying 100%.

7 Conclusion and Future Work

We present a real-time human pattern activity recognition algorithm used on the smartphone platform for human activity identification. We introduce Online Empirical Bayesian Kernel Segmentation and two-layer Random Forest classifier. The multivariate normality assumption for the likelihood function of the variable is bit restrictive; we consider the Online Multivariate Kernel Density estimation to avoid this assumption. Furthermore, rather than extracting the features from each small and fixed length segment, this method computes the features based on the entire activity time interval. Different classification layers use different features in order to avoid using the meaningless features, to save the running time. The aim of the first layer is to distinguish the dynamic activities (walking, walking_upstairs, walking_downstairs) and the static activities (sitting, standing and lying) using the amplitude as the key feature. The process of the second layer is to distinguish the three sub-groups in different categories. However, it is still a big challenge to differentiate walking and walking_upstairs based only on the accelerometer sensor data. We could get better results with regarding the GPS data in later work. In this work, the performance has reached an overall accuracy of 91.4% for the six states, and the overall accuracy for the two categories dynamic activities and static activities is 100%. In the future, a real-time detecting activity system can be designed on a smartphone. Since it has been trained and it can automatically detect behavior change, it is easier to handle this process for a user without the user's knowledge of machine learning and the multiple inputs of a smartphone when the user changes his/her behaviors. Also, we can extend the set of activities with a low frequency and a high frequency, such as fast walking, slow walking, fast running, and jogging.

The main future work is to create an App based on this human pattern recognition system that is used on a cell phone, and any user can operate it directly after downloading it. First of all, we need to process the training step and generate the prototype. There are two approaches to developing this step. One of them is to collect the training sensor data from the multiple volunteers, then train the classification algorithm with these data on a computer, and store this prototype on the cell phone. This prototype is more flexible and suitable for different types of users. The other one is a user can collect data and train on a computer or Cloud computing. For example, push “start” and “end” option to store Walking data when a user performs walking on the first try. The prototype created this way is more suitable for this user and not affected by other users' time series data. Next, this App detects the pattern segmenting with the Online Empirical Bayesian Kernel Segmentation method. Finally, to classify each segment (pattern), compare it with each activity prototype using classification algorithm. By following all above steps, it would automatically trace a user's pattern once the user turns on the accelerometer signal.

For the statistical part, we need to compare the segmenting result of OBKS method by using different kernel functions mentioned in Table 3.1. It could define an optimal kernel function used for segmenting the human activities time series data. Also, the different types of time series data might require different types of kernel functions. Because properties of a computer and a smartphone are different, incremental updating of the Online Kernel Density function that is a part of the online segmentation method, which requires a more efficient algorithm when it is running on a cell phone compared with running on a computer. Looking for the best bandwidth matrix is the primary and challenging step in Online Kernel Estimation. In addition, we can consider the possible and meaningful features and select the relevant features. As an important step before classification, the appropriate features make the classification more efficient, since these features are given the characteristic of each class.

References

[1] Ericsson mobility report: 70 percent of world's population using smartphones by 2020. `https://www.ericsson.com/news/1925907`, June 2015.

[2] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[3] Charu C Aggarwal. A survey of stream classification algorithms., 2014.

[4] Dima Alberg and Avner Ben-Yair. Online hoeffding bound algorithm for segmenting time series stream data. *Journal of Applied Quantitative Methods*, 5(3):446–453, 2010.

[5] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International Workshop on Ambient Assisted Living*, pages 216–223. Springer, 2012.

[6] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. Energy efficient smartphone-based activity recognition using fixed-point arithmetic. *J. UCS*, 19(9):1295–1314, 2013.

[7] Akin Avci, Stephan Bosch, Mihai Marin-Perianu, Raluca Marin-Perianu, and Paul Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *Architecture of computing systems (ARCS), 2010 23rd international conference on*, pages 1–10. VDE, 2010.

[8] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *International Conference on Pervasive Computing*, pages 1–17. Springer, 2004.

[9] Eran Borenstein and Shimon Ullman. Class-specific, top-down segmentation. In *European conference on computer vision*, pages 109–122. Springer, 2002.

[10] Eran Borenstein and Shimon Ullman. Combined top-down/bottom-up segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 30(12):2109–2125, 2008.

[11] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[12] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[13] Pierluigi Casale, Oriol Pujol, and Petia Radeva. Human activity recognition from accelerometer data using a wearable device. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 289–296. Springer, 2011.

[14] Soumen Chakrabarti, Martin Ester, Usama Fayyad, Johannes Gehrke, Jiawei Han, Shinichi Morishita, Gregory Piatetsky-Shapiro, and Wei Wang. Data mining curriculum: A proposal (version 1.0). *Intensive Working Group of ACM SIGKDD Curriculum Committee*, page 140, 2006.

[15] Jingyuan Cheng, Oliver Amft, and Paul Lukowicz. Active capacitive sensing: Exploring a new wearable sensing modality for activity recognition. In *International Conference on Pervasive Computing*, pages 319–336. Springer, 2010.

[16] Nicolas Chopin. Dynamic detection of change points in long time series. *Annals of the Institute of Statistical Mathematics*, 59(2):349–366, 2007.

[17] László Dobos and János Abonyi. Fisher information matrix based time-series segmentation of process data. *Chemical Engineering Science*, 101:99–108, 2013.

[18] Tarn Duong and Martin L Hazelton. Cross-validation bandwidth matrices for multivariate kernel density estimation. *Scandinavian Journal of Statistics*, 32(3):485–506, 2005.

[19] Miikka Ermes, Juha Parkka, and Luc Cluitmans. Advancing from offline to online activity recognition with wearable sensors. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4451–4454. IEEE, 2008.

[20] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

[21] F Foerster, M Smeja, and J Fahrenberg. Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring. *Computers in Human Behavior*, 15(5):571–583, 1999.

[22] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156, 1996.

[23] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[24] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[25] Andrew Gelman, G Roberts, and W Gilks. Efficient metropolis jumping hules. *Bayesian statistics*, 5(599-608):42, 1996.

[26] Laurence Gillick and Stephen J Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 532–535. IEEE, 1989.

[27] James R Glass. A probabilistic framework for segment-based speech recognition. *Computer Speech & Language*, 17(2):137–152, 2003.

[28] Wolfgang Härdle, Helmut Lütkepohl, and Rong Chen. A review of nonparametric time series analysis. *International Statistical Review*, 65(1):49–72, 1997.

[29] Nils-Bastian Heidenreich, Anja Schindler, and Stefan Sperlich. Bandwidth selection for kernel density estimation: a review of fully automatic selectors. *AStA Advances in Statistical Analysis*, 97(4):403–433, 2013.

[30] Johan Himberg, Kalle Korpiaho, Heikki Mannila, Johanna Tikanmaki, and Hannu TT Toivonen. Time series segmentation for context recognition in mobile devices. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 203–210. IEEE, 2001.

[31] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[32] Andrey D Ignatov and Vadim V Strijov. Human activity recognition using quasiperiodic time series collected from a single tri-axial accelerometer. *Multimedia Tools and Applications*, pages 1–14, 2015.

[33] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning. 6, 2013.

[34] Simon J Julier and Jeffrey K Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, Technical report, Robotics Research Group, Department of Engineering Science, University of Oxford, 1996.

[35] Simon J Julier and Jeffrey K Uhlmann. A new extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics, 1997.

[36] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[37] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[38] Jens Kohlmorgen and Steven Lemm. A dynamic hmm for on-line segmentation of sequential data. In *Advances in neural information processing systems*, pages 793–800, 2001.

[39] Konrad Königsberger. *Analysis 2*. Springer-Verlag, 2013.

[40] Matej Kristan and Ales Leonardis. Online discriminative kernel density estimator with gaussian kernels. *IEEE transactions on cybernetics*, 44(3):355–365, 2014.

[41] Matej Kristan, Aleš Leonardis, and Danijel Skočaj. Multivariate online kernel density estimation with gaussian kernels. *Pattern Recognition*, 44(10):2630–2642, 2011.

[42] Dana Kulic and Yoshihiko Nakamura. Scaffolding on-line segmentation of full body human motion patterns. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2860–2866. IEEE, 2008.

[43] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[44] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[45] Christophe G Lambert, Scott E Harrington, Campbell R Harvey, and Arman Glodjo. Efficient on-line nonparametric kernel density estimation. *Algorithmica*, 25(1):37–57, 1999.

[46] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition

using wearable sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013.

[47] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[48] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[49] Xiaoyan Liu, Zhenjiang Lin, and Huaiqing Wang. Novel online methods for time series segmentation. *IEEE Transactions on Knowledge and Data Engineering*, 20(12):1616–1626, 2008.

[50] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[51] Fengjun Lv and Ramakant Nevatia. Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In *European conference on computer vision*, pages 359–372. Springer, 2006.

[52] Andrea Mannini and Angelo Maria Sabatini. Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2):1154–1175, 2010.

[53] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[54] Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[55] Antonietta Mira et al. On metropolis-hastings algorithms with delayed rejection. *Metron*, 59(3-4):231–241, 2001.

[56] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[57] Taketoshi Mori, Yu Nejigane, Masamichi Shimosaka, Yushi Segawa, Tatsuya Harada, and Tomomasa Sato. Online recognition and segmentation for time-series motion with hmm and conceptual relation of actions. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3864–3870. IEEE, 2005.

[58] Scott Niekum, Sarah Osentoski, Christopher G Atkeson, and Andrew G Barto. Champ: Changepoint detection using approximate model parameters. Technical report, DTIC Document, 2014.

[59] Suman H Pal and Jignasa N Patet. Time-series data mining: A review. *Binary Journal of Data Mining & Networking*, 5(1):01–04, 2015.

[60] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[61] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[62] Ananth Ranganathan. Pliss: Detecting and labeling places using online change-point detection. *Robotics: Science and Systems VI*, 2010.

[63] Jorge-L Reyes-Ortiz, Luca Oneto, Sama Albert, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, 2016.

[64] Jorge-Luis Reyes-Ortiz, Luca Oneto, Alessandro Ghio, Albert Sama, Davide Anguita, and Xavier Parra. Human activity recognition on smartphones with awareness of basic activities and postural transitions. In *International Conference on Artificial Neural Networks*, pages 177–184. Springer, 2014.

[65] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[66] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.

[67] Steven L Salzberg. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240, 1994.

[68] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[69] Hagit Shatkay and Stanley B Zdonik. Approximate queries and representations for large data sequences. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, pages 536–545. IEEE, 1996.

[70] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul JM Havinga. A survey of online activity recognition using mobile phones. *Sensors*, 15(1):2059–2085, 2015.

[71] Jeffrey S Simonoff. *Smoothing methods in statistics.* Springer Science & Business Media, 2012.

[72] Thad Starner and Alex Pentland. Real-time american sign language recognition from video using hidden markov models. In *Motion-Based Recognition*, pages 227–243. Springer, 1997.

[73] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[74] Wataru Takano and Yoshihiko Nakamura. Real-time unsupervised segmentation of human whole-body motion and its application to humanoid robot acquisition of motion symbols. *Robotics and Autonomous Systems*, 75:260–272, 2016.

[75] Paul A Tobias and David Trindade. *Applied reliability.* CRC Press, 2011.

[76] DK Vishwakarma, Prachi Rawat, and Rajiv Kapoor. Human activity recognition using gabor wavelet transform and ridgelet transform. *Procedia Computer Science*, 57:630–636, 2015.

[77] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications,*

and Control Symposium 2000. AS-SPCC. The IEEE 2000, pages 153–158. Ieee, 2000.

[78] Matt P Wand and M Chris Jones. *Kernel smoothing*. Crc Press, 1994.

[79] MP Wand and MC Jones. Multivariate plug-in bandwidth selection. *Computational Statistics*, 9(2):97–116, 1994.

[80] Qing Xie, Chaoyi Pang, Xiaofang Zhou, Xiangliang Zhang, and Ke Deng. Maximum error-bounded piecewise linear representation for online stream approximation. *The VLDB Journal*, 23(6):915–937, 2014.

[81] Zhenghua Xu, Rui Zhang, Ramamohanarao Kotagiri, and Udaya Parampalli. An adaptive algorithm for online time series segmentation with error bound guarantee. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 192–203. ACM, 2012.