Accurate and Efficient Human Activity Recognition

Weihao Cheng

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

June 2018

School of Computing and Information Systems
THE UNIVERSITY OF MELBOURNE

Abstract

Human Activity Recognition (HAR) is a promising technology which enables artificial intelligence systems to identify user's physical activities such as walking, running, and cycling. Recently, the demand for HAR is continuously increasing in pace with the rapid development of ubiquitous computing techniques. Major applications of HAR including fitness tracking, safety monitoring, and contextual recommendation have been widely applied in people's daily lives. For example, a music App on smartphones can use HAR to detect the current activity of the user and recommend activity-related songs.

State-of-the-art HAR methods are based on the machine learning technique, where a classification model is trained on a dataset to infer a number of predefined activities. The data for HAR is usually in the form of time series, which can be collected by sensors such as accelerometers, microphones, and cameras. In this thesis, we mainly focus on HAR using the data from inertial sensors, such as accelerations from accelerometers.

A large number of existing studies on HAR aim to obtain high recognition accuracy. However, efficiency is also an important aspect of HAR. In this thesis, we attempt to improve HAR methods for both accuracy and efficiency. Toward this goal, we first devise accurate HAR methods, and then improve the efficiency of HAR while maintaining the accuracy. More specifically, we tackle three problems. The first problem is to accurately recognize the current activity during activity transitions. Existing HAR methods train classification models based on tailored time series containing single activity. However, in practical scenarios, a piece of time series data could capture multiple interleaving activities causing activity transitions. Thus, recognition of the current activity, i.e., the most recent one, is a critical problem to investigate. The second problem is to accurately predict complex activities from ongoing observations. Many time-critical applications, such as safety monitoring, require early recognition of complex activities which are performed over a long period of time. However, without being fully observed, complex activities are hard to be recognized due to their complicated patterns. Therefore, predicting complex activities from ongoing observations is an important task to study. The third problem is to improve energy-efficiency of HAR on mobile devices while maintaining high accuracy. Many applications of HAR are based on mobile devices. However, due to the limited battery capacity, real-time HAR requires minimization of energy cost to extend the operating spans of the devices. Generally, the cost can be cut down by reducing algorithmic computations and sensing frequencies. Yet it is worth to find a maximal cost

reduction while preserving a high recognition accuracy.

In this thesis, we present a set of algorithms to address the proposed problems. The key contributions of the thesis can be summarized as follows:

1. We propose a method to accurately recognize the current activity in the presence of multiple activities with transitions. The method partitions a time series matching the occurring activities, where the maximum classification error of these activities is minimized.

2. We propose a method to accurately predict complex activities over time from ongoing multivariate time series. The method utilizes an action sequence model and a complex activity model, which make predictions alternately based on each other as the observed data increases.

3. We propose a method to minimize the computational cost of HAR while maintaining high recognition accuracy. The method uses a Markov Decision Process (MDP) to select an optimal subset of feature representations for ensemble classification that minimizes redundant computations.

4. We propose a method to minimize a combined measurement of sensing cost and classification error of HAR. The method uses MDP to select appropriate sensing rate to sample the incoming data points, where the sparsity of the outcome time series is ensured to preserve the recognition accuracy.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Weihao Cheng, 2 June 2018

Preface

The research of this thesis has been carried out in School of Computing and Information Systems, The University of Melbourne. The main contributions of the thesis are discussed in Chapters 3, 4, 5, 6, which are based on the following publications:

- **Weihao Cheng**, Sarah Erfani, Rui Zhang, Kotagiri Ramamohanarao, "Accurate Recognition of the Current Activity in the Presence of Multiple Activities", *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Jeju, South Korea, 2017.

- **Weihao Cheng**, Sarah Erfani, Rui Zhang, Kotagiri Ramamohanarao, "Predicting Complex Activities from Ongoing Multivariate Time Series", *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden, 2018.

- **Weihao Cheng**, Sarah Erfani, Rui Zhang, Kotagiri Ramamohanarao, "Markov Dynamic Subsequence Ensemble for Energy-Efficient Activity Recognition", *Proceedings of International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, Melbourne, Australia, 2017.

- **Weihao Cheng**, Sarah Erfani, Rui Zhang, Kotagiri Ramamohanarao, "Learning Datum-Wise Sampling Frequency for Energy-Efficient Activity Recognition", *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, USA, 2018.

I declare that I am the primary author and have contributed $> 50\%$ in the above papers.

Acknowledgements

I cherish the opportunity of pursuing my doctoral study under the supervision of Professor Kotagiri Ramamohanarao, Professor Rui Zhang, and Dr. Sara Erfani. I would like to express my sincere gratitude to them for continuous mentoring, monitoring, guidance, and technical advices during my PhD study.

I would like to thank the members of PhD committee: Professor Lars Kulik, for his constructive comments and suggestions on my work. I am also thankful to my colleagues in the department: Dr. Fang Meng, Yuan Li, Yitong Li, Zeyi Wen, Sun Yu, and Xiaojie Wang for their insightful discussion and guidance on the technical side.

I'm grateful for the scholarships from the University of Melbourne, which support me to pursue my doctoral degree.

My deepest gratitude goes to my parents who deserve the credit for whatever success that I have attained in my life. I thank them for their continuous support and dedication.

Weihao Cheng
Melbourne, Australia
June 2018

Contents

1 Introduction **1**
- 1.1 Introduction . . . 1
- 1.2 Thesis Structure . . . 5

2 Background **7**
- 2.1 Human Activity Recognition . . . 7
  - 2.1.1 Preliminary . . . 8
  - 2.1.2 Review of Representative Work on Human Activity Recognition . . . 9
- 2.2 Accurate Recognition of the Current Activity during Activity Transitions . . . 12
  - 2.2.1 Overview . . . 12
  - 2.2.2 Time Series Segmentation . . . 13
- 2.3 Accurate Predicting of Complex Activities from Ongoing Observations . . . 15
  - 2.3.1 Overview . . . 15
  - 2.3.2 Early Classification on Time Series . . . 16
  - 2.3.3 Deep Learning for Human Activity Recognition . . . 17
- 2.4 Energy-Efficient Human Activity Recognition . . . 20
  - 2.4.1 Overview . . . 20
  - 2.4.2 Ensemble Learning . . . 23
  - 2.4.3 Markov Decision Process . . . 25
- 2.5 Human Activity Recognition Datasets . . . 26
- 2.6 Conclusion . . . 28

3 Accurate Recognition of the Current Activity during Activity Transitions **29**
- 3.1 Introduction . . . 29
- 3.2 Methodology . . . 31
  - 3.2.1 Problem Statement . . . 32
  - 3.2.2 Min-Max Activity Recognition Model (MARM) . . . 32
  - 3.2.3 Weighted Min-Max Activity Recognition Model (WMARM) . . . 35
  - 3.2.4 Efficient Implementation of WMARM . . . 37
- 3.3 Empirical Evaluation . . . 38
  - 3.3.1 Measuring the Accuracy of Current Activity Recognition . . . 40
  - 3.3.2 Evaluating the Impact of μ on Accuracy . . . 41
  - 3.3.3 Measuring the Accuracy on Actual Transitions . . . 41
  - 3.3.4 Evaluating the Execution Time on Smartphone . . . 43
- 3.4 Conclusion . . . 43

4 Accurate Predicting of Complex Activities from Ongoing Observations **45**
4.1 Introduction . . . 45
4.2 Methodology . . . 47
4.2.1 Problem Statement . . . 48
4.2.2 Simultaneous Complex Activities Recognition and Action Sequence Discovering (SimRAD) . . . 48
4.3 Experiments . . . 54
4.3.1 Prediction of Complex Activities . . . 57
4.3.2 Recognition of Action Sequence . . . 59
4.4 Conclusion . . . 60

5 Efficient Human Activity Recognition by Reducing Computational Cost **61**
5.1 Introduction . . . 61
5.2 Proposed Method . . . 63
5.2.1 Markov Dynamic Subsequence Ensemble (MDSE) . . . 65
5.2.2 Theoretical Analysis of the Accuracy Constraints in MDSE . . . 68
5.2.3 Computational Efficiency of MDSE . . . 73
5.3 Empirical Evaluations . . . 75
5.3.1 Performances of Different Subsequences . . . 77
5.3.2 Markov Dynamic Subsequence Ensemble . . . 78
5.3.3 The Accuracy Constraints . . . 81
5.3.4 The Computational Efficiency of MDSE . . . 82
5.3.5 Evaluation on Smartphone . . . 82
5.4 Conclusion & Discussion . . . 83

6 Efficient Human Activity Recognition by Reducing Sensing Cost **85**
6.1 Introduction . . . 85
6.2 Methodology . . . 87
6.2.1 Problem Statement . . . 87
6.2.2 Datum-Wise Frequency Selection (DWFS) . . . 89
6.3 Empirical Evaluation . . . 96
6.3.1 Recognition Accuracy versus Energy Cost . . . 98
6.3.2 Evaluating the Performance of DWFS . . . 99
6.3.3 Insight Study of DWFS . . . 99
6.4 Conclusion & Discussion . . . 101

7 Conclusion & Future Research **105**
7.1 Summary of Contributions . . . 105
7.2 Future Research . . . 107

List of Figures

1.1 The figure shows a general work-flow of Human Activity Recognition (HAR), which is using sensor time series data to recognize an individual's daily activities, such as walking, running, and cycling. In step 1, sensors on portable devices are used to collect a piece of time series data (The red-blue-green signal curves are the 3-axis accelerations). In step 2, the time series data is processed and fed to a classification model. In step 3, the classification model infers the activity represented by the data. In step 4, the inferred activity can be used in various applications, such as fitness tracking. 2

2.1 The time series shown in the curves are 3-axis acceleration signals. There are 3 activities captured in the time series with transition points τ_0, τ_1, τ_2, τ_3, where τ_0 and τ_3 are two end points. Sitting and standing are the previous activities, walking is the current activity that we expect to recognize. 13

2.2 A general Deep Neural Network structure. 17

2.3 The time series shown in red, green and blue curves are 3-axis acceleration data. $\mathcal{W}_1, ..., \mathcal{W}_n$ are n overlapping windows, which observes different ranges of the time series from timestamp τ_1 to τ_2. $\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_i$ share one size with 50% overlaps. Similarly, $\mathcal{W}_{i+1}, ..., \mathcal{W}_j$ share another size, and there are several different sizes of windows. 22

3.1 The time series shown in the curves are 3-axis acceleration signals. There are 3 activities captured in the time series with transition points τ_0, τ_1, τ_2, τ_3, where τ_0 and τ_3 are two end points. Sitting and standing are the previous activities, walking is the current activity that we expect to recognize. 31

3.2 Accuracy of WMARM with respect to μ on datasets: HASC, HARSD, ACTR, and DSA. For $K = 0$, the accuracy slightly improves on HASC, HARSD, and DSA, and slightly drops on ACTR. For $K = 1, 2, 3$, the accuracy reaches maximum around $\mu = 0.7(\pm 0.1)$, and then slightly decreases by less than 1% or becomes stable. 42

4.1 The neural network structure of the feature learner G. The inputs 1, ..., d are the univariate time series on each channel of X. The output of G is the learned feature vector. 51

4.2 Prediction accuracies of CAs at different progress levels on OPP-BW dataset 57

4.3 Prediction accuracies of CAs at different progress levels on OPP-BH dataset 57
4.4 Prediction accuracies of CAs at different progress levels on OPP-BUA dataset 57
4.5 The accuracies of SimRAD with respect to λ_t 59

5.1 The time series shown in red, green and blue curves are 3-axis acceleration data. $\mathcal{W}_1, ..., \mathcal{W}_n$ are n overlapping windows, which observes different ranges of the time series from timestamp τ_1 to τ_2. $\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_i$ share one size with 50% overlaps. Similarly, $\mathcal{W}_{i+1}, ..., \mathcal{W}_j$ share another size, and there are several different sizes of windows. 63
5.2 Given $N = 20$ (subsequences), $m = 6$ (activities) and $\epsilon = 0.2$, the curves show the upper bound of Ω as function of β with different η. 75
5.3 Accuracies of Decision Trees (DTs) using the subsequences X_1, X_2, ..., X_{17} on different datasets. 77
5.4 Accuracy and Cost (%) of Markov Dynamic Subsequence Ensemble (MDSE) as functions of β with different η. 78
5.5 ΔAccuracy/ΔCost of MDSE as a function of β with different η. 78
5.6 Comparing accuracy of MDSE with CNN and RNN on different datasets. 78
5.7 The dashed curve shows the upper bound of Ω with respect to β, and the solid curve shows the empirical Ω. 82

6.1 The dynamics of the MDP. 91
6.2 The 4 relationships on HASC, HARSD, and DSA datasets. (a) Classification error with respect to λ. (b) Energy cost with respect to λ. (c) Classification error with respect to energy cost. (d) Frequency changing rate with respect to λ. 103

List of Tables

3.1 Results of accuracy on datasets: HASC, HARSD, ACTR, and DSA. K is the number of transitions in the time series. K is not known to the methods. . 41

4.1 The Wilcoxon test to compare the prediction accuracies regarding R^+, R^-, and p-values. 58
4.2 Recognition accuracies on action sequences. 59

5.1 Major expressions 64
5.2 Comparing MDSE with SBC and SWEM using base classifiers: DT, LR and KNN. 79
5.3 Comparing MDSE with Random K methods (DT as base classifier). Due to the page limitation, we only show the results of Random $K-2$, $K-1$, K, $K+1$ and All, where K is set as the smallest integer that greater than the obtained ensemble size of MDSE. 80
5.4 Performances on a Google Nexus 5X. 83

6.1 Recognition Accuracy versus Energy Cost (Accelerometer). The unit of the energy cost values is Joule per hour (J/h). 98
6.2 The Error-Cost Index with respect to λ on HASC dataset. 100
6.3 The Error-Cost Index with respect to λ on HARSD dataset. 101
6.4 The Error-Cost Index with respect to λ on DSA dataset. 102
6.5 The Wilcoxon test to compare the Error-Cost Indexes of DWFS, DWFS-CVL, DWFS-SL, RNN, MDP-DS, and Random regarding R^+, R^-, and p-values. 102

Chapter 1
Introduction

Human Activity Recognition (HAR) is a promising technique which is widely deployed on ubiquitous computing systems to identify individual's activities using contextual data. In this thesis, we aim to improve the performance of HAR on both accuracy and efficiency. Toward this goal, we study three problems. The first problem is accurate recognition of the current activity during activity transitions. We propose a method which divides a given time series into segments matching the appearing activities and recognizes the current activity simultaneously. The second problem is accurate predicting of complex activities from ongoing observations. We devise an algorithm which predicts a complex activity over time by mining a sequence of multivariate actions from the observed time series. The third problem is improving energy-efficiency of HAR on portable devices while maintaining high recognition accuracy. We devise two independent methods to address this problem. One uses a Markov Decision Process (MDP) to dynamically select ensemble base classifiers for reducing computational cost, and the other uses MDP to select appropriate sampling frequency for reducing sensing cost. Our solutions to the three HAR problems have achieved prominent results, and we hope that they can attract more attention to new paradigms of HAR.

1.1 Introduction

Human Activity Recognition (HAR) is one of the most promising research topics under the rapid development of ubiquitous technologies. The goal of HAR is to identify user's activities based on context information collected by sensors, such as accelerometer, gyroscope, and GPS. Generally, the context information is presented as a time series, where each data point of the time series is a sensor reading, such as a 3-axis acceleration or a GPS coordinate. HAR algorithms extract features from a window of the time

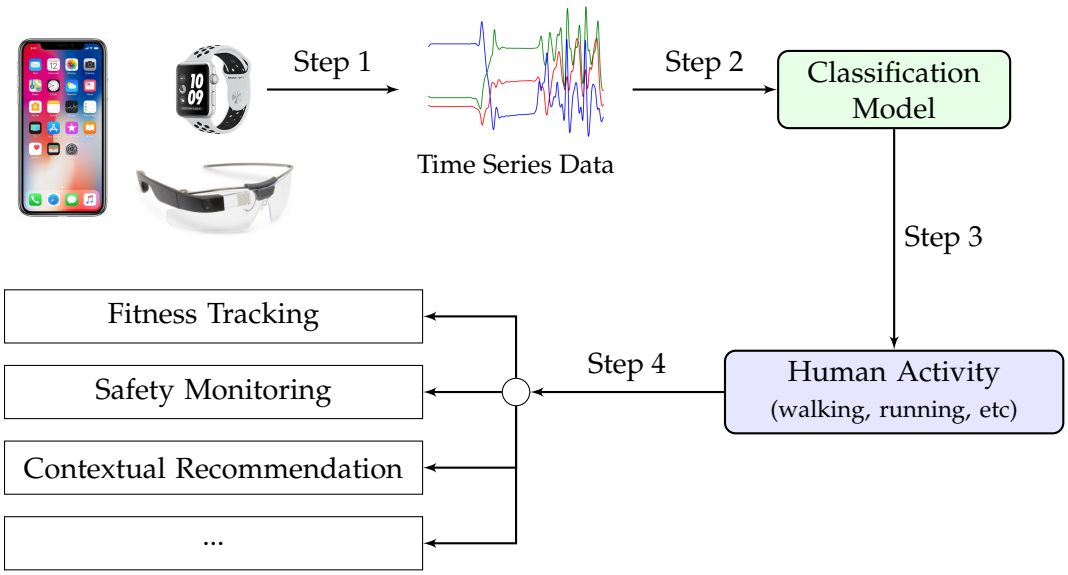


Figure 1.1: The figure shows a general work-flow of Human Activity Recognition (HAR), which is using sensor time series data to recognize an individual's daily activities, such as walking, running, and cycling. In step 1, sensors on portable devices are used to collect a piece of time series data (The red-blue-green signal curves are the 3-axis accelerations). In step 2, the time series data is processed and fed to a classification model. In step 3, the classification model infers the activity represented by the data. In step 4, the inferred activity can be used in various applications, such as fitness tracking.

series and learn a classification model based on the features for inferring the activity. HAR has been widely applied in many real-world scenarios. For example, virtual assistants on smartphones, such as Siri and Cortana, use HAR to detect a user's behaviors, and thus the systems can provide personal assistance regarding an individual's degree of functional ability and life style. Figure 1.1 summarizes a general work-flow of HAR. Although HAR has been studied for a decade, there are many challenging problems left behind, which attract significant attention to study. The most vital problems in HAR are: recognition of complex activities [62, 63, 106], energy-efficient HAR on mobile platforms [16, 28, 42, 46, 49, 65, 73, 94, 95, 98], abnormal activity recognition [68, 69, 101], optimal sensor placement [6, 48], ground truth data annotation [31, 90], and so on.

Most of the existing studies focus on the accuracy of HAR, however, efficiency is also an important aspect. Accurate and efficient recognition can greatly extend the applicability and scalability of HAR in the real world. Accordingly, in this thesis, **we aim to improve HAR on both accuracy and efficiency**. Toward this goal, we start by devising accurate HAR methods, and then we improve the efficiency of HAR while maintaining

the accuracy. More specifically, we solve three problems as follows:

- **Problem 1** - *Accurate recognition of the current activity during activity transitions*: Most existing HAR algorithms are designed to recognize the current activity based on a window of time series data assuming that only one activity is taking place. However, during activity transitions, a time series of multiple interleaving activities can be captured through the window. Simply feeding such time series to the classification model may cause incorrect recognition of the current activity. Therefore, finding the current activity in the presence of multiple activities is an important task for achieving accurate HAR.

- **Problem 2** - *Accurate predicting complex activities from ongoing observations*: Most existing work focuses on recognizing simple activities, also known as actions, such as 'lifting' and 'grabbing', which are performed in a short time period and can be efficiently captured. Compared with actions, complex activities comprised of a group of temporally related actions, are usually performed over a much longer time period. Many time-critical applications require recognition of complex activities without being fully observed, which incurs an emerging problem of predicting complex activities from ongoing observations.

- **Problem 3** - *Improving energy-efficiency of HAR while maintaining accuracy*: HAR algorithms have been widely deployed on smartphones and smartwatches due to their powerful sensing and computing capacity. However, real-time HAR requires continuous sensing and computing which cause a large amount of power consumption that significantly decreases the battery lifetime. Therefore, improving energy-efficiency of HAR is especially critical to mobile platforms. A naive way is to reduce computations of HAR algorithms or sampling frequencies of sensors, but this can affect the recognition accuracy as a trade-off. Thereafter, it is a challenging task to reduce energy expenditure of HAR while maintaining accuracy.

The problems proposed above are emergent yet challenging. We draw on advanced machine learning techniques to develop effective methods for improving HAR's accuracy and efficiency. In the following paragraphs, we briefly introduce our solutions regarding the proposed problems.

For the first problem: *accurate recognition of the current activity during activity transitions*, we propose Weighted Min-max Activity Recognition Model (WMARM), which recognizes the current activity by optimally partitioning the observed window of time series matching the activities presented. WMARM considers weights on the partitioned segments to obtain reliable recognition accuracy. WMARM can also effectively handle the time series containing an arbitrary number of transitions without any prior knowledge about the number of transitions. Instead of exhaustively searching the optimal solution of WMARM in exponential space, we propose an efficient dynamic programming algorithm that computes the model in $\mathcal{O}(n^2)$ time complexity, where n is the length of the window. Moreover, we present an efficient implementation of WMARM that the computation cost can be further reduced. Extensive experiments on 5 real-world HAR datasets have demonstrated the superior performance of WMARM on handling time series with one or more activity transitions. The results show about 10%-30% improvement on the accuracy of current activity recognition compared with state-of-the-art methods. The experiment on smartphones shows the significant computational efficiency of WMARM.

For the second problem: *accurate predicting of complex activities from ongoing observations*, we propose Simultaneous Complex Activities Recognition and Action Sequence Discovering (SimRAD) which predicts a complex activity over time by finding a sequence of multivariate actions from sensory times series data using a Deep Neural Network. SimRAD learns two probabilistic models for inferring complex activities and action sequences, where the estimations of the two models are conditionally dependent on each other. SimRAD alternately predicts the complex activity and the action sequence with the two models, thus the predictions can be mutually updated until the completion of the complex activity. We evaluate SimRAD on a real-world complex activity dataset of a rich amount of sensor data. The results demonstrate that SimRAD outperforms state-of-the-art methods by 7.2% on average prediction accuracy with very high confidence.

For the third problem: *improving energy-efficiency of HAR while maintaining accuracy*, we propose two independent methods which reduce energy cost from two different aspects. The first method focuses on reducing computational cost which is caused by performing computations on activity inference. This method improves the subsequence ensemble models which use multiple feature representations based on subsequences of a time se-

ries. The subsequence ensemble models deliver excellent performance on recognition accuracy, but they are very expensive due to a large amount of computation on multiple subsequences. We formalize a dynamic subsequence selection problem that minimizes the required computations while persevering a high recognition accuracy. To solve this problem, we propose Markov Dynamic Subsequence Ensemble (MDSE), an algorithm for the selection of the subsequences via a Markov Decision Process (MDP), where a policy is learned for choosing the best subsequence given the state of prediction. Regarding MDSE, we derive an upper bound of the expected ensemble size, so that the energy consumption caused by the computations of the proposed method is guaranteed. Extensive experiments are conducted on 6 real-world HAR datasets to evaluate the effectiveness of MDSE. Compared with the state-of-the-art methods, MDSE reduces 70.8% computational cost which is 3.42 times more energy efficient, and achieves a comparably high accuracy. The second method focuses on reducing sensing cost which is caused by sampling data with sensors. This method aims to dynamically select sampling frequencies by directly exploiting sensor data. We formalize a problem of minimizing an objective function regarding classification error and sensing cost, by finding an optimal classification model and dynamically appropriate sampling rates. To address this problem, we propose Datum-Wise Frequency Selection (DWFS) which utilizes a continuous state Markov Decision Process (MDP). The MDP learns a policy function that selects the best frequency for sampling an incoming data entity by utilizing previously sampled data. We propose an alternate learning scheme, where the parameters of the classification model and the policy function are mutually enhanced. We evaluate the performance of DWFS on 3 real-world HAR datasets, and the results show that DWFS statistically outperforms the state-of-the-art regarding a combined measurement of classification error and energy cost.

1.2 Thesis Structure

The rest of the thesis is organized as follows:

- Chapter 2 reviews related work of the thesis. We summarize existing studies on the three problems that will be addressed in the thesis. We also introduce the related

techniques used in our proposed methods.

- Chapter 3 studies the problem of accurate recognition of the current activity during activity transitions. We propose a method to infer the current activity from multiple interleaving activities. The idea of the method is to partition a given time series matching the occurred activities, where the maximum recognition error of those activities is minimized.

- Chapter 4 studies the problem of accurate predicting of complex activities recognition from ongoing observations. We propose a method to predict complex activities over time with the incrementing of the observed sensory data. The method utilizes an action sequence model and a complex model, which are alternately evaluated to make predictions based on each other.

- Chapter 5 studies the problem of improving HAR energy-efficiency by reducing computational cost. We propose a method to reduce the amount of HAR's computation while maintaining high recognition accuracy. The method uses Markov Decision Process (MDP) to select an optimal subset of feature representations for ensemble classification that minimizes the unnecessary computations.

- Chapter 6 studies the problem of improving HAR energy-efficiency by reducing sensing cost. We propose a method to minimize a combined measurement of sensing cost and classification error of HAR. The method uses MDP to select appropriate sensing rate to sample the incoming data points, where the sparsity of the sampled time series is ensured to maintain recognition accuracy.

- Chapter 7 concludes the key findings and highlights main research outcomes in this thesis. We also include the future directions regarding the study of accurate and efficient HAR.

Chapter 2
Background

In this chapter, we present the background knowledge of the thesis, where we mainly introduce our research problems and review the related literature. In section 2.1, we provide a brief introduction of Human Activity Recognition (HAR), including the preliminary knowledge and a number of representative work in previous HAR research. In section 2.2, we introduce the problem of accurate recognition of the current activity during activity transitions. For this problem, we mainly review the studies on time series segmentation, which can be used to precisely find the current activity in the presence of multiple activities. In section 2.3, we introduce the problem of accurate predicting of complex activities from ongoing observations. For this problem, we review related work on early time series classification. We also review the deep learning technique, which is a powerful tool for delivering accurate predictions of complex activities. In section 2.4, we discuss the problem of improving energy-efficiency of HAR while maintaining recognition accuracy. For this problem, we first survey the traditional work on reducing the energy cost of HAR. We then review Ensemble Learning and Markov Decision Process, which are promising techniques for obtaining optimal energy-efficiency. In section 2.5, we introduce the datasets used in the experiments of this thesis.

2.1 Human Activity Recognition

The task of Human Activity Recognition (HAR) is to recognize human physical activities, e.g., walking and running, using the data collected from sensors, e.g., accelerometers, microphones, and cameras. Due to the rapid development of ubiquitous computing technology in recent years, HAR is widely applied on smart systems, such as smartphones, to facilitate the understanding of a user's behaviors and provide assistance to the user.

The applications of HAR fall across a board range of disciplines [64]. A fitness tracking system uses HAR to analyze the daily activities performed by a user, and provide fitness recommendation to the user to keep healthy [53]. A safety monitoring system uses HAR to detect the abnormal behaviors of a user that an alert can be sent out to avert or minimize the consequences [35]. A context-aware system uses HAR to track the current activity of a user [26,75,88], and modifies the system's configuration to improve user experience, for example, a news App can enlarge font size to enhance reading experience when the user is walking.

2.1.1 Preliminary

State-of-the-art HAR methods are based on supervised machine learning technique, which includes a training stage and an inference stage. Let $\mathcal{Y} = \{y_1, y_2, ..., y_m\}$ be a label set of m human activities to identify. First of all, a set of sensors are used to collect a number of time series regarding the m activities. Then, these time series are split into several segments by a sliding time window $\mathcal{W}$ to generate a training dataset $\mathcal{D} = \{(X^{(i)}, y^{(i)})\}$, where $X^{(i)}$ is a time series segment, and $y^{(i)} \in \mathcal{Y}$ is the annotated activity label of $X^{(i)}$. At the training stage, we want to learn a probability model $p(y \mid X; \theta)$ to maximize an overall inference probability on the dataset $\mathcal{D}$ as:

$$\max_{\theta} \sum_{(X^{(i)}, y^{(i)}) \in \mathcal{D}} p(y = y^{(i)} \mid X = X^{(i)}; \theta), \tag{2.1}$$

where θ is the model's parameter. The probability model $p(y \mid X; \theta)$ can be derived from a classification model such as nearest neighbors [2], softmax regression [21], decision tree [74], support vector machine [89], naive Bayes [23], Hidden Markov Model [12], Deep Neural Network (DNN) [54], etc. For example, the probabilistic model of softmax regression is expressed as:

$$p(y = y_k \mid X; \theta = \{\theta_1, ..., \theta_m\}) = \frac{e^{-\theta_k^T \phi(X)}}{\sum_{j=1}^{m} e^{-\theta_j^T \phi(X)}}, \quad \text{for } k = 1, ..., m, \tag{2.2}$$

where θ consists of a set of weight parameters $\theta_1, ..., \theta_m$, and ϕ is a function that returns a vector of features representing the X. The chosen of the features are essential for the

learning of classification models. Some of the mostly used features in HAR can be categorized into: time domain features (e.g., mean, variance, auto-correlation, zero crossing) and frequency domain features (e.g., Fourier coefficients). Beside using predefined features, state-of-the-art classification models such as DNN can directly learn feature representations from raw data, which is more promising if there is sufficient training data. At the inference stage, a stream of data points are continuously sampled by sensors. A sliding time window is then used to collect a number of the most recent data points. These data points are formed as a testing time series X_t, and the model $p(y \mid X; \theta)$ is then used to infer the activity label of X_t as:

$$\hat{y} = \underset{y_k \in \mathcal{Y}}{\operatorname{argmax}}\, p(y = y_k \mid X = X_t; \theta), \tag{2.3}$$

where $\hat{y}$ is the inferred activity label. In experimental environments, a ground truth label y_{true} is provided along with the testing time series to evaluate the correctness of the inference. The error of the inference is measured as $1\{y_{true} \neq \hat{y}\}$, where the indicator function $1\{condition\}$ returns 1 if the *condition* holds true and 0 otherwise.

2.1.2 Review of Representative Work on Human Activity Recognition

In the last fifteen years, many HAR approaches have been proposed regarding a variety of sensors, classification models, application scenarios, etc [4, 10, 30, 33, 34, 52, 53, 55, 60, 76, 77, 86, 104, 108]. We review a number of representative work on general HAR.

Bao et al. [10] conduct a study of HAR using multiple body-worn accelerometers to recognize twenty daily activities including walking, sitting, running, etc. They examine four classification models: decision table, nearest neighbor, C4.5 decision tree, and naive Bayes classifier. They discover that using C4.5 decision tree obtains the best recognition accuracy which is 84.26%. Krumm et al. [52] study to use WiFi signal for recognizing two activities: move and still. They use the variance of WiFi signal strength as the feature, and they learn a Hidden Markov Model (HMM) which takes a sequence of the variances as observations and outputs the most likely sequence of activities represented by the hidden states. They report 87% accuracy achieved by their method. Ravi et al. [76] use a single triaxial accelerometer to recognize eight activities. They test five classifiers and

three ensemble schemes of combining the five classifiers. The five classifiers are decision table, C4.5 decision tree, K-nearest neighbor, support vector machine, and naive Bayes. The three ensemble schemes are voting, boosting, and bagging. They find that using the ensemble method of plurality voting obtains the best accuracy which is 99.82%. Lester et al. [55] propose a HAR method based on a set of sensors including accelerometer, ambient light, microphone, etc. They extract a number of candidate features from the data of all the sensors. For each activity, they use Boost Cascade [92] to select useful features, and they learn a two-stage model to calculate the likelihood of the activity. The first stage is an ensemble of decision stump classifiers based on the selected features to estimate the activity probability. The second stage is an HMM which takes a sequence of the probabilities by the ensemble classifier as observations and returns the activity likelihood. The final inferred activity is the one with the maximum likelihood. They test the method to recognize ten activities, and the method achieves an overall accuracy of 95%.

Liao et al. [60] develop a HAR model using location information from GPS. They construct a hierarchical relational Markov network where activities are simultaneously estimated with user's locations from local evidence. They test the method to recognize ten activities such as 'Work', 'Sleep', and 'Pickup', and the method achieves 85% accuracy. Anderson et al. [4] propose to use GSM Cellular of mobile phones to recognize three simple activities: walking, driving, and stationary. They apply an artificial neural network for classification using features based on the variance of signal strength and number of detected cellular towers. They report the results of accuracies for each activity: 90% for stationary, 79% for walking, and 36% for driving. Zheng et al. [108] study to use GPS logs to recognize user's transportation modes: walking, driving, bus, and bike. They adopt a decision tree for activity inference based on a set of geographical features such as heading change rate, velocity change rate, and stop rate. Then, they propose a graph-based post-processing algorithm to further improve the inference performance. The results show that their method achieves an inference accuracy of 76.2%. Reddy et al. [77] use accelerometer and GPS in combination to recognize user's transportation modes. They find that the information from GPS is the most important to make an accurate inference. They extract mean, variance, energy, 1-10Hz Fourier coefficients from accelerations

and GPS speed as features. They propose a two-stage classifier which combines decision tree with HMM. The experimental results show that their method achieves an overall accuracy of 93.6%.

Gu et al. [30] propose an Emerging Sequential Patterns based approach to recognize human activities using body sensor networks. For each activity, they select a set of sequential patterns which are frequently appeared and have large discriminative powers. Given a segment of testing time series, they calculate the aggregate probability of one activity by evaluating all the sequential patterns appearing in the time series, and the inferred activity is the one with highest aggregate probability. They further extend their method to recognize interleaved and concurrent activities by considering correlation probabilities. The experimental results show that the Emerging Sequential Pattern method obtains 91.89% accuracy. Kwapisz et al. [53] design a practical HAR system on Android devices. Their HAR method is trained on 10-second segments of acceleration data, where they extract mean, standard deviation, average absolute difference, average resultant acceleration, time between peak, and binned distribution as features. They test J48 decision tree [32], logistic regression, and multilayer perceptron [83] as classification models, and they found that multilayer perceptron achieves the best accuracy which is 91.7%. Sankaran et al. [86] propose a novel method which uses a barometer to recognize three locomotion activities: vehicle, walking, and idle. Since a barometer measures atmospheric pressure which contains vibration information, they calculate 'jump', peaks, and standard deviation from pressures data stream as the features. Then, they use a thresholding based method to detect the activities. The experimental results show that their method achieves an overall accuracy of 69%.

Zeng et al. [104] use a Convolutional Neural Network (CNN) for recognition of human activities. They incorporate the partial weight sharing strategy on the convolutional layer of the CNN, which learns temporal relevant patterns directly from raw time series data. They test their method on three public datasets: Skoda [103], Opportunity [81], and Actitracker [53], and the obtained accuracies are 88.19%, 76.83%, and 96.88%, respectively. Hammerla et al. [33] systematically investigate three deep learning models: deep feed-forward networks, CNN, and Recurrent Neural Network (RNN) for HAR. They evaluate the performances of the models with randomly sampled hyper-parameters on

three public datasets: Opportunity [81], PAMAP2 [79], and Daphnet Gait [7]. They measure f1-scores for the recognition of each activity. The results show that RNN performs the best on recognizing activities of short durations with natural ordering, and CNN performs the best on recognizing activities of long duration with repetitive patterns.

2.2 Accurate Recognition of the Current Activity during Activity Transitions

Most of the existing HAR methods [36, 53, 77] use segmented time series to train classifiers for activity inference, where each segment of time series represents a single activity. In practice, such HAR systems utilize a window to capture the data stream of sensors in a fixed time duration, and feed the captured time series data to a trained classifier to infer the current activity. However, a window of the data stream may contain more than one activity causing transitions at arbitrary time positions, see Figure 2.1 for example. Simply using a time series containing multiple activities for classification that expects input containing single activity can lead to a poor recognition accuracy. A trivial approach is to use a small window, so that there is a high probability to capture a clean time series of the current activity with a minimal chance of transition taking place. But the trade-off is that fewer data points will result in lower recognition performance. Therefore, accurate recognition of the current activity in the presence of multiple activities is a challenging task. In this section, we first survey the existing work on handling activity transitions. We then review the technique of time series segmentation, which can be used to accurately recognize the current activity from time series containing multiple activities.

2.2.1 Overview

To the best of our knowledge, none of the existing work has touched the problem of recognizing the current activity during transitions. There are a few related studies which aim to minimize the effects induced by activity transitions [78], or to recognize the transitions [45,80]. Rednic et al. [78] report that activity transitions can cause rapid fluctuations in classifier output. They utilize the Exponentially Weighted Voting filter to stabilize the inference, but the approach is unable to identify the current activity. Some HAR methods

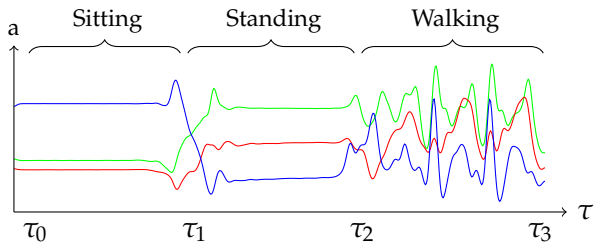


Figure 2.1: The time series shown in the curves are 3-axis acceleration signals. There are 3 activities captured in the time series with transition points τ_0, τ_1, τ_2, τ_3, where τ_0 and τ_3 are two end points. Sitting and standing are the previous activities, walking is the current activity that we expect to recognize.

[45, 80] learn a classifier to recognize the activity transitions in time series. As the transition can provide information of the activities sequence, this approach can be utilized to infer the current activity. However, it is unsuitable to handle time series containing several transitions, such as stand-walk-run, since there will be a factorial number of classes that should be trained. For example, if there are N different activities, and the system demands to handle at most m transitions, then the total number of required classes is $\sum_{r=1}^{m+1} P_r^N$, where P_r^N stands for the number of permutations selecting r ordered objects from N objects. As a consequence, learning transitions is not an efficient approach for current activity recognition. We consider to solve this problem by incorporating time series segmentation, where activities are divided matching the segments for easy recognition.

2.2.2 Time Series Segmentation

Time series segmentation aims to divide a sequence into several homogeneous segments. The existing methods can be summarized into four categories: Heuristic, LASSO, Clustering, and Dynamic Programming. Heuristic based methods [44] use top-down, bottom-up, sliding window, or hybrid ways for dividing time series. The results of heuristic methods are not stable since the optimal solution cannot be obtained. LASSO based methods [57] solve the segmentation problem via a least-square regression with a ℓ_1-penalty. However, LASSO based methods require the number of maximum transitions as input. Clustering based methods [91] divide the subsequences in a time series into K-clusters by using K-means approaches. However, Clustering based methods require

the number of time series patterns as input. Dynamic programming based methods [14,37,41,82] formulate an optimal substructure of the segmentation problem, and finds the optimal partition of a time series based on the optimal partitions of its subsequences. Dynamic programming based methods can be categorized into two types. The first type is for handling K-segmentation problem [14,37,82], which finds an optimal solution of K segments on time series, where the number of transitions K is required. The second type is for unconstrained segmentation problem [41,47] , which finds optimal solution over all possible partitions on time series, where the number of transitions is not required. The methods for unconstrained segmentation problem can be summarized as following. Let $X = \{x_1, x_2, ..., x_n\}$ be a time series of n data points, and $X_{i:j} = \{x_i, x_{i+1}, ..., x_{j-1}, x_j\}$ $(1 \leq i \leq j \leq n)$ be a subsequence of X containing data points from x_i to x_j. Let $\boldsymbol{\tau}_s = \{\tau_1, \tau_2, ..., \tau_{m_s}\}$ be a set of transition points on $X_{1:s}$, where $0 < \tau_1 < \tau_2 < ... < \tau_{m_s} < s$. For convenient discussion, we denote $\tau_0 = 0$ and $\tau_{m_s+1} = s$. Given a cost function $C(X_{i:j})$ which measures the heterogeneity of time series $X_{i:j}$, finding the transition points on $X \equiv X_{1:n}$ is to calculate:

$$F(X_{1:n}) = \min_{\boldsymbol{\tau}_n} \sum_{i=1}^{m_n+1} [C(X_{(\tau_{i-1}+1):\tau_i}) + \beta], \tag{2.4}$$

where β is a constant value to prevent over fitting. This problem can be solved by dynamic programming [41]. For any $s \in [1, n]$, we can derive $F(X_{1:s})$ as:

$$F(X_{1:s}) = \min_{\boldsymbol{\tau}_s} \sum_{i=1}^{m_s+1} [C(X_{\tau_{i-1}+1:\tau_i}) + \beta] \tag{2.5}$$

$$= \min_{0 \leq t < s} \{\min_{\boldsymbol{\tau}_t} \sum_{i=1}^{m_t+1} [C(X_{(\tau_{i-1}+1):\tau_i}) + \beta] + C(X_{t+1:s}) + \beta\} \tag{2.6}$$

$$= \min_{0 \leq t < s} \{F(X_{1:t}) + C(X_{t+1:s}) + \beta\}, \tag{2.7}$$

where $F(X_{1:0})$ is set to $-\beta$. This equation formulates a relationship which calculates the minimal cost $F(X_{1:s})$ in terms of $F(X_{1:t})$ for $t < s$. Thus, the problem of calculating $F(X_{1:n})$ can be solved in a recursive manner, and the optimal set of transition points $\boldsymbol{\tau}_n^*$ can be found as:

$$\boldsymbol{\tau}_n^* = \operatorname*{argmin}_{\boldsymbol{\tau}_n} F(X_{1:n}). \tag{2.8}$$

As the result, the unconstrained segmentation problem on time series is resolved via dynamic programming with a quadratic time complexity $\mathcal{O}(n^2)$.

2.3 Accurate Predicting of Complex Activities from Ongoing Observations

Sensor networks, which comprise groups of sensors, have been widely deployed to monitor environments such as shops, offices, and factories. Hence, using sensor multivariate time series to recognize human activities becomes an emerging problem for artificial intelligence systems to understand multiplex human behaviors. Classic learning models used in activity recognition are based on time series of fully observed activities. However, complex activities, such as 'cooking', generally have much longer durations compared with simple activities, which are also called actions, such as 'grabbing' and 'lifting'. Therefore, using the classic models for complex activities will result in late recognition. In the real world, many activity recognition applications are time-critical. For example in safety monitoring, a system need to predict dangerous complex activities with partial observations to avert or minimize their consequences. Accordingly, we require a method that can accurately recognize the complex activity given a multivariate time series of its early stage. In this section, we first introduce the existing work on complex activity recognition. We then briefly survey the early classification problem regarding time series data. Finally, we review the deep learning technique, which can deliver reliable performance in capturing ambiguity actions from multivariate time series for predicting high level complex activities.

2.3.1 Overview

Over the past decade, a large body of work has studied recognition of simple human activities [24, 33, 50, 99, 100]. Recently, the rapid development of sensor networks enables the recognition of complex activities from multivariate time series (MTS). Hierarchical recognitions with multi-level activity abstraction, which are originally studied in the field of computer vision [18, 67], have been widely applied for complex activity recognition [62, 63, 93, 106]. Wang et al. [93] propose a model which first detects actions and then

recognizes complex activities via an emerging pattern method. Later, a number of studies focus on the modeling of complex activities with temporal relations among actions [62,63,106]. Zhang et al. [106] propose a Bayesian network based approach for modeling temporal relations among action for complex activity recognition. Liu et al. [63] present an approach which extracts temporal patterns among actions for complex activity representation and uses multi-task learning framework for classification. Liu et al. [62] present a complex activity recognition model which uses Chinese Restaurant Process to capture the inherit structural varieties of complex activities. However, due to the demand for time-critical applications, predicting complex activity at early stages becomes an important challenge. Early recognition of human activities is first studied as a computer vision problem [58, 66, 84]. The proposed methods focus on video streams, where visual features/actions are computed from the video images and then used for early prediction. However, compared with videos, sensor MTS data contains much less information regarding each timestamp. For example, an image of a RGB video usually has more than hundreds of pixels, but a data point of an accelerometer signal has only 3 values (accelerations on 3 axises). Therefore, recognition of early human activities based on sensor MTS data incurs substantial challenges. Li et al. [58] propose to predict complex activities by finding the casual relations between actions and predictable characteristic of the activities. However, their method is designed based on pre-annotated actions, which need to be obtained from sensor MTS data with an additional step. Therefore, we need a predicting method which directly uses MTS of sensor data without annotating of actions.

2.3.2 Early Classification on Time Series

The task of time series classification is to build a classifier for inferring the class label of a given time series which has been fully observed. However, many time-sensitive applications require a quick decision without waiting until the end of the observation. Accordingly, early classification of time series becomes an emergent problem, toward which several models have been proposed [27, 59, 96, 97]. Xing et al. [96] develop a 1-nearest neighbor classification model for early prediction on univariate time series. Later, Xing et al. [97] and Ghalwash and Obradovic [27] propose to extract interpretable feature for early classification, which can provide the interpretation to the classifying results.

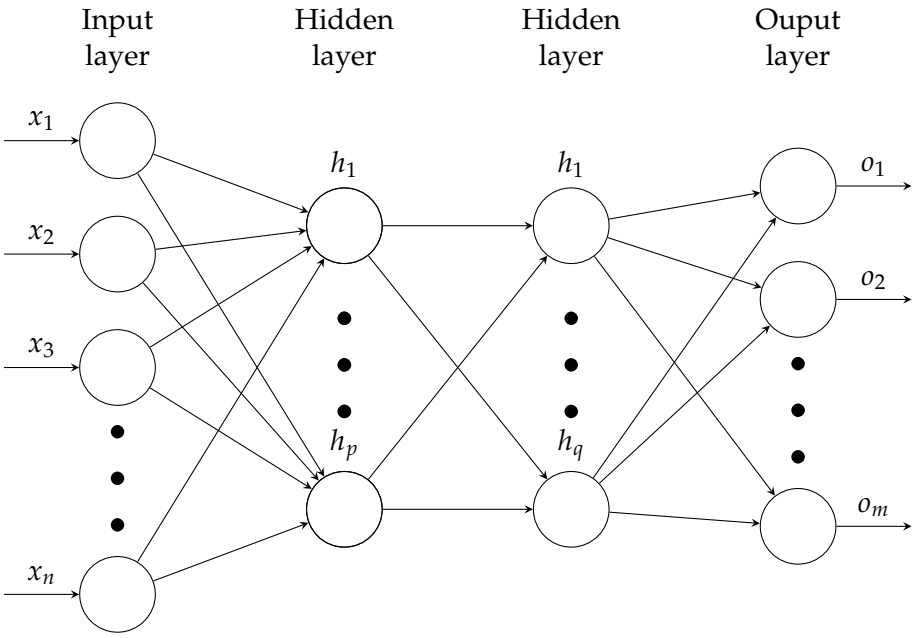


Figure 2.2: A general Deep Neural Network structure.

These methods intend to find one optimal early stage to classify a time series. Li et al. [59] propose a multivariate marked point-process based method that can classify time series at arbitrary early stages, where MTS is modeled into events for classifying based on temporal dynamics and sequential cue.

2.3.3 Deep Learning for Human Activity Recognition

Since the real world is highly complex, unpredictable, and constantly changing, finding reliable feature representation for identifying human activities from noisy sensor data remains a challenging problem. Deep learning is one of the most promising approaches that can automatically discover effective representations of data using a multi-level computation architecture akin to neuron connections in human brains, which is collectively referred to as **Deep Neural Network** (DNN) [54]. DNN has been widely applied in the fields of computer vision [51], speech recognition [38], natural language processing [8], and has dramatically improved the state-of-the-art. Typically, a DNN is a computational graph consisting of a number of layers, where each layer is composed of a collection of nodes. The nodes of one layer are connected with the nodes of its successor layer and predecessor layer. The raw data (e.g., accelerometer, audio, images) are fed to the

first layer, which is also called the **input layer**. The inference outcomes are presented by the last layer, which is also called the **output layer**, where each node of this layer captures the confidence of a class. Layers in between the input and the output layer are referred to **hidden layers**. The influence of nodes between layers varies on a pairwise basis determined by a weight value. Together with the synaptic connections and inherent non-linearity, the hidden layers transform raw data presented by the input layer into the inference probabilities of the classes, which are captured in the output layer. DNN-based inferencing follows a forward propagation algorithm. The algorithm starts to feed a data instance $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ to the input layer, and then sequentially updates the node activations of one layer based on its predecessor layers. The process finishes at the output layer, which computes a vector $\boldsymbol{o} = (o_1, o_2, ..., o_m)$, when all nodes have been updated. The final inferred class is identified as the one corresponding to the output vector $\boldsymbol{y}$ with the greatest softmax activation value. A common DNN structure is illustrated in Figure 2.2. Mathematically, a forward pass of DNN can be written as:

$$\boldsymbol{o} = f(\boldsymbol{x}) = f^{(n)}(f^{(n-1)}(\dots f^{(2)}(f^{(1)}(\boldsymbol{x})))), \tag{2.9}$$

where $f^{(i)}$ is a linear transformation function corresponding to the i-th layer. Suppose $\boldsymbol{h}^{(i)}$ is the input of $f^{(i)}$, then $f^{(i)}$ is expressed as:

$$\boldsymbol{h}^{(i+1)} = f^{(i)}(\boldsymbol{h}^{(i)}) = \sigma(W\boldsymbol{h}^{(i)}), \tag{2.10}$$

where W is the transformation matrix of $f^{(i)}$, and σ is an activation function such as sigmoid and hyperbolic tangent function. Note that W and σ can be different for each layer. The structure of DNN has many variants for different application scenarios. Convolutional neural network (CNN) is one variant that has brought about breakthrough in processing images, video, speech, and audio. CNN includes convolution layers which perform discrete convolution operations on the input. Assume that the input $\boldsymbol{h}^{(i)}$ is a 1-dimension sequence, the convolution layer can be expressed as:

$$\boldsymbol{h}^{(i+1)}(t) = (\boldsymbol{h}^{(i)} * w)(t) = \sum_{a=-\infty}^{\infty} \boldsymbol{h}^{(i)}(a)w(t-a), \tag{2.11}$$

where w is the kernel of the convolution layer. A significant advantage of CNN is that it can learn location invariant patterns from input. Beside the CNN architecture, recurrent neural network (RNN) is another variant of DNN which has shone light on sequential data. RNN uses recurrent computing structures to facilitate inference with history information of a sequence. A forward pass of a simple RNN can be written as:

$$\boldsymbol{h}(t) = \sigma_h(W_h\boldsymbol{x}(t) + U_h\boldsymbol{h}(t-1) + b_h), \tag{2.12}$$

$$\boldsymbol{o}(t) = \sigma_o(W_o\boldsymbol{h}(t) + b_o), \tag{2.13}$$

where W_h, U_h, W_o are transformation matrix, and b_h, b_o are bias vectors. For each step t, RNN encodes all the sighted data points $\boldsymbol{x}(1), \boldsymbol{x}(2), ..., \boldsymbol{x}(t)$ into a hidden vector $\boldsymbol{h}(t)$, which is then used to produce output $\boldsymbol{o}(t)$. Due to the recurrence, RNN is capable of learning to use history information. In practice, however, a simple RNN is unable to capture long-term dependency. Towards this problem, one of the most promising solution is Long Short-Term Memory (LSTM) network, which maintains a memory cell to store useful information. A forward pass of LSTM can be written as:

$$\boldsymbol{g}(t) = \sigma_g(W_g\boldsymbol{x}(t) + U_g\boldsymbol{h}(t-1) + b_g), \tag{2.14}$$

$$\boldsymbol{v}(t) = \sigma_v(W_v\boldsymbol{x}(t) + U_v\boldsymbol{h}(t-1) + b_v), \tag{2.15}$$

$$\boldsymbol{o}(t) = \sigma_o(W_o\boldsymbol{x}(t) + U_o\boldsymbol{h}(t-1) + b_o), \tag{2.16}$$

$$\boldsymbol{c}(t) = \boldsymbol{g}(t) \circ \boldsymbol{c}(t-1) + \boldsymbol{v}(t) \circ \sigma_c(W_c\boldsymbol{x}(t) + U_c\boldsymbol{h}(t-1) + b_c), \tag{2.17}$$

$$\boldsymbol{h}(t) = \boldsymbol{o}(t) \circ \sigma_h(\boldsymbol{c}(t)), \tag{2.18}$$

where W_g, U_g, W_v, U_v, W_o, U_o, W_h, U_h are transformation matrices, and b_g, b_v, b_o, b_c are bias vectors. Eq. 2.14 calculates a forget gate vector determining which to clear out from the memory cell. Eq. 2.15 calculates an input gate vector determining which to add into the memory cell. Eq. 2.17 updates the memory cell with the gate vectors. Accordingly, LSTM solves the long-term dependency problem and brings promising results.

Due to the great success of deep learning in computer vision, speech recognition, and natural language processing, a number of studies investigate DNN for solving HAR problems based on sensor time series data. Ploetz et al. [71] propose a feature learning

approach for HAR based on a deep belief network as an autoencoder, where the weights between each subsequent layers are generatively trained by Restricted Boltzmann Machines. The results of the deep model demonstrate the potential to address contemporary activity recognition tasks. Zeng et al. [104] tackle the HAR problem by adopting a CNN which captures local dependencies and scale invariants of sensor signals. The CNN uses the convolution layer which adopts a partial weight sharing strategy [1]. The results demonstrate that the CNN outperforms existing state-of-the-art methods. Yang et al. [99] propose a deeper CNN, which includes a number of convolution layers, max-pooling layers, normalization layers, and full-connected layers. Hammerla et al. [33] conduct an unbiased and systematic investigation of the performances of three deep learning models: deep feed-forward networks, CNN, and LSTM. The performances of the models are evaluated with randomly sampled hyper parameters. The results show that LSTM performs the best on recognizing activities of short durations with natural ordering, and CNN performs the best on recognizing activities of long duration with repetition.

2.4 Energy-Efficient Human Activity Recognition

Recent mobile devices, such as smartphones and smartwatches, are equipped with an increasing range of sensing and computing resources, which enable HAR to emerge across a wide variety of application areas. However, most HAR applications require continuously monitoring activities in real-time, so that the energy consumption of HAR becomes an important issue when HAR is performed on portable devices. In this section, we first review the existing studies of improving energy-efficiency of HAR. We then briefly introduce ensemble learning and markov decision process, which are two promising techniques for obtaining optimal energy-efficiency.

2.4.1 Overview

The problem of improving energy-efficiency of HAR on mobile platforms has been widely investigated in recent years [16, 28, 42, 46, 49, 65, 73, 94, 95, 98]. A number of HAR systems are designed to regulate the utilization of sensors to obtain energy-efficiency. Lu et al. [65] propose 'Jigsaw' system which controls the usage of high power GPS based

on an accelerometer. Bhargava et al. [15] propose 'SenseMe' system which uses linear accelerometer and rotation vector sensor as a means to suppress the usage of GPS. Bloch et al. [17] propose to use a low power cellular network information to detect user stationary/movement status which avoids battery-exhausting transportation mode detection. Some general sensor management methods have also been proposed. Kang et al. [42] propose 'SeeMon' system which reduces the energy cost by iteratively selecting cost-efficient sensors in a greedy manner. Wang et al. [95] propose 'EEMSS' which uses a hierarchical sensor management scheme for power control. Gordon et al. [28] choose the sensors to use based on the estimated future activity and quantified activity-sensor dependencies. However, those methods are designed based on a specific set of sensors, which cannot be generalized on universal devices. We need energy-efficient HAR algorithms which can be universally applicable to all kinds of mobile platforms. Towards this goal, we propose to improve energy-efficiency of HAR by reducing two types of energy cost: 1) computational cost, and 2) sensing cost. In the following part of this section, we briefly introduce the two types of cost, and review the related literatures inspired us to solve the problem.

Computational Cost

Running HAR algorithms in real-time generates computational cost, where the majority of the cost are from feature extraction and model inference. In the phase of feature extraction, HAR algorithms calculate a set of features from an input time series as a feature vector for inference. Recall that the commonly used features for HAR are categorized into two types: time domain features and frequency domain features. Time domain features require a linear time complexity $\mathcal{O}(n)$ to calculate, where n is the length of the input time series. Frequency domain features require a time complexity of $\mathcal{O}(n \log n)$ to calculate, which is more computationally expensive than time domain features. The computational cost of feature extraction increases with respect to the number of features to calculate and the time complexity of the features. To improve cost efficiency, a HAR method can use less number of features and avert using of frequency domain features. But inappropriate feature selection may bring down the recognition accuracy due to the loss of effective representation. In the phase of model inference, the classification model of HAR calculates

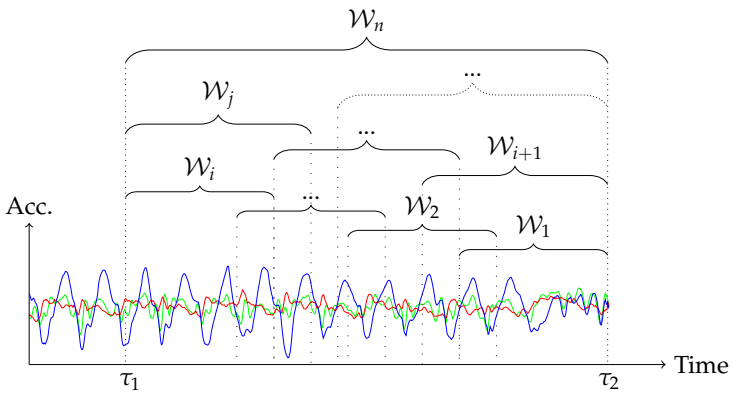


Figure 2.3: The time series shown in red, green and blue curves are 3-axis acceleration data. $\mathcal{W}_1, ..., \mathcal{W}_n$ are n overlapping windows, which observes different ranges of the time series from timestamp τ_1 to τ_2. $\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_i$ share one size with 50% overlaps. Similarly, $\mathcal{W}_{i+1}, ..., \mathcal{W}_j$ share another size, and there are several different sizes of windows.

activity inferences based on the extracted features. The computational cost of the model inference increases with respect to the complexity of the model. To improve cost efficiency, a HAR method can adopt less complexity classification models, such as decision tree and logistic regression. However, these models may not provide reliable recognition accuracy, as their performances highly depend on the selected features. State-of-the-art HAR algorithms use ensemble models [9,55,76,107], which deliver promised recognition accuracy. Ensemble models are considered as high complexity models, which make inferences based a group of linear classifiers, where each classifier is based on a unique set of features. *Subwindow Ensemble Model* (SWEM) is an ensemble model particularly designed for HAR. For a given time series, SWEM uses multiple windows to capture subsequences of different ranges, as shown in Figure 2.3, and extracts features from each subsequence. SWEM infers the activity with an ensemble of classifiers, where each classifier is based on one subsequence. Although SWEM achieves high recognition accuracy, SWEM is computationally very expensive, since the use of each subsequence incurs computations on feature extraction and inference calculation. Therefore, the total cost of SWEM equals to the sum of all these subsequence costs, which is several times more than conventional linear models. Due to such an overhead, running of SWEM on portal devices drains the battery quickly that affects the general daily use. To overcome this weakness, we need an algorithm which is capable of dynamically choosing an optimal set of subsequences that minimize the computational cost while maintaining the high recognition accuracy.

Sensing Cost

Sampling data from sensors, such as accelerometer and GPS, generates sensing cost. The sensing cost increases with respect to the number of times that sampling actions are performed. As real-time HAR requires continuous sampling data, it can cause excessive power consumption that greatly shortens the battery life of mobile devices. A simple solution to reduce sensing cost is setting a low sampling frequency to sensors, but this may result in a decreased recognition accuracy [49] due to the loss of data information. To address this problem, Khan et al. [46] propose to find a minimal sampling rate that perverse a certain accuracy, and use this precomputed sampling rate during testing. However, the methods of using fixed sampling rate are adaptive thus cause low performance of either accuracy or energy-efficiency. Yan et al. [98] and Qi et al. [73] focus on finding energy-efficient features and sampling rate for each activity during training, and they propose to adaptively change sensor sampling rate based on detected activity and predefined thresholds in testing. However, their methods are heuristic that cannot achieve a dynamic optimal. Wang et al. [94] propose a method that obtains Markov-optimal sensing policy for user state estimation. Yurur et al. [102] extend the work of Wang et al. and consider the trend of user preferences to regulate sensor sampling settings. However, these methods only utilize predefined user state to determine the next sensing policy, where the beneficial information from data samples is not exploited. Therefore, we need an algorithm which directly exploits the information from recently observed data to choose adaptive sampling frequency that achieves a global minimal regarding both accuracy and energy-efficiency.

2.4.2 Ensemble Learning

Ensemble learning utilizes multiple learning algorithms to obtain a better inference performance than could be obtained from any of the constituent algorithms alone [109]. An ensemble model for classification contains a number of classifiers, which are called **base classifiers**. A base classifier can be a decision tree, logistic regression, etc. Most of the ensemble models use base classifiers of the same type, and these models are called **homogeneous ensembles**. The ensemble models which use base classifiers of different types,

are called **heterogeneous ensembles**. The generalization ability of an ensemble model is often much stronger than a base classifier. Actually, ensemble models are appealing mainly because they are able to boost weak classifiers whose performances are even just slightly better than random guess to strong classifiers which can make very accurate inferences. At training phase, ensemble models generate a set of base classifiers from a dataset. Boosting [25, 87] and Bagging [19] are the two major algorithms for generating base classifiers. At inference phase, ensemble models combine the outputs of the base classifiers to make an ensemble inference, where voting is normally used as the combination method. Suppose we have a set of T base classifiers $\{h_1, h_2, ..., h_T\}$. The number T is called **ensemble size**. Our task is to infer the class label from a set of l possible class labels $\{1, 2, ..., l\}$. For an instance $\boldsymbol{x}$, the output of the classifier h_i is given as an l-dimensional voting vector $(y_{i,1}, y_{i,2}..., y_{i,l})$, where $y_{i,j} \in \{0, 1\}$ takes value one if h_i infers j as the class label and zero otherwise. The voting method takes the class label which receives the maximum number of votes. Thereby, the output class label of the ensemble is:

$$\underset{j}{\operatorname{argmax}} \sum_{i=1}^{T} y_{i,j}, \tag{2.19}$$

and ties are broken arbitrarily. Assume that the outputs of the classifiers are independent and each base classifier is better than random guess that makes a correct classification by probability $p > 1/2$. We denote $P_{correct}$ as the probability of the ensemble for making a correct inference. We can derive that:

$$P_{correct} \geq \sum_{k=\lceil T/2+1 \rceil}^{T} \binom{T}{k} p^k (1-p)^{T-k} \tag{2.20}$$

$$\geq 1 - \exp\{-\frac{1}{2} T(1-2p)^2\}. \tag{2.21}$$

The Inequality 2.20 is derived by the fact that $P_{correct}$ is greater than or equal to the probability of obtaining at least $\lfloor T/2 + 1 \rfloor$ correct classifiers out of T. The Inequality 2.21 is derived by using Hoeffding inequality [39]. It is obvious that $1 - \exp\{-\frac{1}{2} T(1-2p)^2\}$ is monotonically increasing with respect to ensemble size T. Therefore, the accuracy of ensemble inference can be theoretically improved by combing more base classifiers.

2.4.3 Markov Decision Process

Markov decision processes (MDPs) provide a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker [13, 72]. MDP has been applied in autonomous flight, robot legged locomotion, cell-phone network routing, marketing strategy selection, factory control, and efficient web-page indexing. An MDP is defined by a tuple $(S, A, \{P_{sa}\}, \gamma, R)$, where:

- S is a set of **states**.
- A is a set of **actions**.
- P_{sa} are the state **transition probabilities**. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space, which gives probability of the next state s' by taking action a at state s.
- $\gamma \in [0, 1)$ is the **discount factor**.
- $R : S \to \mathbb{R}$ is the **reward function**.

The dynamics of an MDP proceeds as follows: We start from an initial state s_0, and choose an action $a_0 \in A$. As a result, the state of the MDP randomly transits to a successor state s_1, drawn according to $s_1 \sim P_{s_0 a_0}$. Then, we choose another action a_1, and the state of MDP transits to $s_2 \sim P_{s_1 a_1}$. Accordingly, we continuously choose actions and visit a sequence of states s_0, s_1, s_2, ..., and then the total reward is:

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + ... \tag{2.22}$$

Solving the MDP is to choose actions over time to maximize the expected value of total reward:

$$\mathrm{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + ...] \tag{2.23}$$

A **policy** π is a function $\pi : S \to A$ mapping from states to actions. Given a state s, we obtain the action as $a = \pi(s)$. For a fixed policy π, a **value function** $V^\pi(s)$ is defined as:

$$V^\pi(s) = R(s) + \gamma \int_{s' \in S} P_{s\pi(s)}(s') V^\pi(s'), \tag{2.24}$$

which consists of the immediate reward $R(s)$ and the expected sum of future rewards. Therefore, the value function $V^{\pi}(s)$ returns the total sum of rewards starting from a state s. We define the **optimal value function** $V^{*}(s)$ as:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s), \tag{2.25}$$

and the **optimal policy** π^{*} as:

$$\pi^{*} = \operatorname*{argmax}_{a \in A} \int_{s' \in S} P_{sa}(s') V^{*}(s'). \tag{2.26}$$

Then, the solution of an MDP is to find the optimal policy π^{*}. There are several approaches to solve MDPs. If the MDP is on a finite state space that $|S| < \infty$, **value iteration** or **policy iteration** can be used to solve π. If the MDP is on an infinite state space that $|S| = \infty$, **fitted value iteration** can be used to solve π [29].

2.5 Human Activity Recognition Datasets

In Human Activity Recognition (HAR) community, there is a plenty of datasets for method evaluations. Most commonly used datasets are: Human Activity Sensing Consortium dataset (HASC) [43], Human Activity Recognition on Smartphones Dataset (HARSD) [5], Smartphone-Based Recognition of Human Activities and Postural Transitions dataset (HAPT) [80], Actitracker dataset (ACTR) [53], Daily Sport Activities dataset (DSA) [11], CHEST dataset[20], and Opportunity dataset (OPP) [81]. A brief description of these datasets is as follows:

- HASC: The dataset is collected from 7 subjects performing 6 activities: 'stay', 'walk', 'jog', 'skip', 'stair up', and 'stair down'. The major devices for data collection are iPhone and iPod Touch. The data of 3-axis acceleration is captured at 100 readings per second.

- HARSD: The dataset is collected from 30 subjects performing 6 activities: 'walking', 'walking_upstairs', 'walking_downstairs', 'stting', 'standing', and 'laying'. The device for data collection is a Samsung Galaxy S II placed at waist. The dataset in-

cludes time series of 3-axis linear acceleration and angular velocity, which are captured at 50 readings per second.

- HAPT: The dataset is extended from HARSD, where the data of 4 activity transitions is added: 'stand-to-sit', 'sit-to-stand', 'sit-to-lie', 'lie-to-sit', 'stand-to-lie', and 'lie-to-stand'.

- ACTR: The dataset is collected from 36 subjects performing 6 activities: 'walking', 'jogging', 'upstairs', 'downstairs', 'sitting', and 'standing'. The devices for data collection are Android phones (Nexus One, HTC Hero, Motorola Backflip). The dataset includes time series of 3-axis acceleration, which are captured at 20 readings per second.

- DSA: The dataset is collected from 8 subjects performing 19 activities: 'sitting', 'standing', 'standing in an elevator still', 'walking in a parking lot', etc. The devices for data collection are body-worn sensors placed on torso, right arm, left arm, right leg, and left leg. The dataset includes time series of 3-axis acceleration, angular speed, and magnetic field, which are captured at 25 readings per second.

- CHEST: The dataset is collected from 15 subjects performing 7 activities: 'working at computer', 'standing up, walking and going updown stairs', 'standing', 'walking', 'going updown stairs', 'walking and talking with someone', and 'talking while standing'. The device for data collection is a wearable accelerometer mounted on chest. The dataset includes time series of 3-axis acceleration, which are captured at 52 readings per second.

- OPP: The dataset is collected from 4 subjects performing 13 low-level actions, 17 mid-level gestures, and 5 high-level complex activities. The devices for data collection are body-worn sensors placed on 12 places: arm, wrist, back, etc. The dataset includes time series of 3-axis acceleration, which are captured at 30 readings per second.

In this thesis, we conduct experiments mainly on HASC, ACTR, and DSA datasets due to their simple data structures for setting up comparisons with state-of-the-art methods. For the problem regarding activity transitions, we additionally use HAPT dataset

which includes data of real activity transitions. For the problem regarding complex activity predicting, we use OPP dataset since it is the only dataset which includes data of complex activities and meets the experimental requirements.

2.6 Conclusion

In this chapter, we presented background knowledge of our research problems and reviewed related literature. Firstly, we provided a brief introduction of Human Activity Recognition (HAR), including the preliminary knowledge and a number of representative work on general HAR research. Then, we proposed our research problems and reviewed the related literature. In section 2.2, we introduced the problem of accurate recognition of the current activity during activity transitions. For this problem, we reviewed the methods of time series segmentation, which can be used to accurately find the current activity from time series containing multiple activities. In section 2.3, we introduced the problem of accurate predicting of complex activities from ongoing observations. For this problem, we reviewed related methods for early time series classification. We also reviewed the deep learning technique, which is a powerful tool for delivering accurate complex activity predictions. In section 2.4, we discussed the problem of improving energy-efficiency of HAR while maintaining the recognition accuracy. For this problem, we first surveyed the traditional methods for reducing the energy cost of HAR. We then reviewed Ensemble Learning and Markov Decision Process, which are the two promising techniques for obtaining optimal energy-efficiency with considerations of accuracy. In section 2.5, we introduced the datasets which are used in the experiments of this thesis.

Chapter 3

Accurate Recognition of the Current Activity during Activity Transitions

In this chapter, we conduct a study on accurate recognition of the current activity when a given window of time series data contains multiple interleaving activities causing activity transitions. Most of the traditional Human Activity Recognition (HAR) methods assume the entire window corresponds to a single activity, which may cause high error rate in activity recognition. To overcome this challenge, we propose Weighted Min-max Activity Recognition Model (WMARM), which reliably recognizes the current activity by finding an optimal partition of the time series matching the occurred activities. WMARM can handle the time series containing an arbitrary number of activities, without having any prior knowledge about the number of activities. We devise an efficient dynamic programming algorithm that solves WMARM in $\mathcal{O}(n^2)$ time complexity, where n is the length of the window. Extensive experiments conducted on 5 real datasets demonstrate about 10%-30% improvement on accuracy of WMARM compared with the state-of-the-art methods.

3.1 Introduction

Most of the existing Human Activity Recognition (HAR) methods [36, 53, 77] use segmented time series to train classifiers for activity inference, where each time series represents a single activity. In practice, such HAR systems utilize a window to capture the data stream of sensors in a fixed time duration, and feed the captured time series data to a trained classifier to infer the current activity. However, a window of the data stream

This chapter is dervied from: Weihao Cheng, Sarah Erfani, Rui Zhang, Kotagiri Ramamohanarao, "Accurate Recognition of the Current Activity in the Presence of Multiple Activities", *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Jeju, South Korea, 2017.

may contain more than one activity causing transitions at arbitrary time positions, see Figure 3.1 for an example. Simply using a time series containing multiple interleaving activities for classification that expects input containing single activity can lead to a poor recognition accuracy. A trivial approach is to use a small window, so that there is a high probability to capture the exact time series of the current activity with a minimal chance of transition taking place. But the trade-off is that the fewer data points will result in lower recognition performance. Therefore, accurate recognition of the current activity in the presence of multiple activities is a challenging task.

There are a few related studies, which aim to minimize the effects induced by activity transitions [78], or recognize the transitions [45, 80]. Rednic et al. [78] reported that activity transitions can cause rapid fluctuations in classifier output. They utilized filters to stabilize the inference, but the approach is unable to identify the current activity. Some AR systems [45, 80] learn a classifier to infer the activity transitions in time series. As the transitions can provide information of the activities sequence, this approach can be utilized to recognize the current activity. However, it is unsuitable to handle time series containing several transitions, such as stand-walk-run, since there will be a factorial number of classes that should be trained. For example, if there are N different activities, and the system demands to handle at most m transitions, then the total number of required classes is $\sum_{r=1}^{m+1} P_r^N$, where P_r^N stands for the number of permutations selecting r ordered objects from N objects. As a consequence, learning transitions is not an efficient approach for current activity recognition.

To address this difficult problem, an idea is to divide the observed window of time series into segments matching activities transitions. Thereby, the clean time series of the current activity, which is represented by the last segment, can be obtained for recognition. However, the existing time series segmentation methods [14, 37, 41, 44, 57, 82, 91] have at least one of the following drawbacks: (1) optimal solution is not guaranteed; (2) requiring the input of an exact or a maximum number of transitions; (3) only focusing on segmentation without considering activity recognition performance. A detailed discussion is later provided in the Related Works. To address these drawbacks, we propose *Weighted Min-max Activity Recognition Model (WMARM)*, which reliably recognizes the current activity by finding an optimal partition of the time series matching the occurred activities.

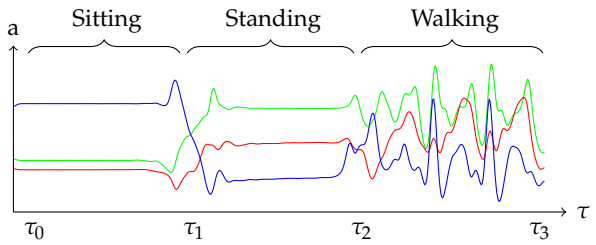


Figure 3.1: The time series shown in the curves are 3-axis acceleration signals. There are 3 activities captured in the time series with transition points τ_0, τ_1, τ_2, τ_3, where τ_0 and τ_3 are two end points. Sitting and standing are the previous activities, walking is the current activity that we expect to recognize.

WMARM calculates a set of segments that the maximum value of the recognition errors on those segments is minimized, and the current activity is recognized based on the last segment. WMARM can handle time series containing an arbitrary number of transitions without having any prior knowledge about the number of transitions. WMARM can also be extended by imposing weights on the segments to improve recognition accuracy. Since the search space size of WMARM is $\mathcal{O}(2^n)$, we provide an efficient algorithm using dynamic programming to solve the model in $\mathcal{O}(n^2)$ time complexity, where n is the length of the window. Moreover, we propose a computationally efficient implementation of WMARM that the time series is divided into frames for coarse-grained processing. We conduct extensive experiments on 5 real datasets. The results demonstrate the superior performance of WMARM compared with the state-of-the-art methods when handling time series that contains one or more activity transitions. We also measure the execution time of WMARM algorithm on a smartphone, and the results indicate that the model can be effectively used on such resource constrained devices.

3.2 Methodology

In this section, we first introduce the preliminary concepts of the methodology. We then propose Min-max Activity Recognition Model (MARM), which recognizes the current activity by optimally partitioning a given window of time series. We further improve the model by considering weights on the segments, and propose Weighted Min-max Activity Recognition Model (WMARM). Both of the models can be solved using dynamic programing in $\mathcal{O}(n^2)$ time complexity, where n is the length of the window. Finally, we

propose an efficient implementation of WMARM for obtaining high performance on resource constraint devices.

3.2.1 Problem Statement

Let $X = \{x_1, x_2, ..., x_n\}$ be a time series of n data points observed by a window. We define $X_{i:j} = \{x_i, x_{i+1}, ..., x_{j-1}, x_j\}$ $(1 \leq i \leq j \leq n)$ as a subsequence of X containing data points from x_i to x_j. Suppose there is a set of m transition points $\boldsymbol{\tau} = \{\tau_1, \tau_2, ..., \tau_m\}$ in the time series X, which are caused by $m+1$ interleaving activities. We define $\tau_0 = 0$, $\tau_{m+1} = n$ and $0 = \tau_0 < \tau_1 < \tau_2 < ... < \tau_m < \tau_{m+1} = n$. The transitions points divide the time series X into $m+1$ segments $\{X_{1:\tau_1}, X_{\tau_1+1:\tau_2}, ..., X_{\tau_m+1:n}\}$, where each segment $X_{\tau_i+1:\tau_{i+1}}$ represents a single activity that is different from its neighbors. Suppose we have a probability model $p(y \mid Z)$, which can estimate the probability of activity y given an arbitrary time series Z. Our goal is to infer the current activity y_c which is presented by the last segment $X_{\tau_m+1:n}$. Traditional methods assume that y_c is the only activity taken place in X. Therefore, they feed the entire X to the model and calculate the inference $\hat{y}_c$ as:

$$\hat{y}_c = \underset{y}{\operatorname{argmax}} p(y \mid X). \tag{3.1}$$

However, these methods usually cannot perform well as X contains interfering information of the other activities. To reliably infer the current activity, we attempt to locate the transition points in X, so that the observed time series can be well-divided into clean segments. As a consequence, the current activity can be exhibited by the last segment and is accurately inferred as:

$$\hat{y}_c = \underset{y}{\operatorname{argmax}} p(y \mid X_{\tau_m+1:n}). \tag{3.2}$$

3.2.2 Min-Max Activity Recognition Model (MARM)

Let Z be an arbitrary time series. The inference error of Z can be calculated by an error function $\mathcal{E}(Z)$ which is defined as:

$$\mathcal{E}(Z) = 1 - \max_{y} p(y \mid Z). \tag{3.3}$$

The error function $\mathcal{E}(Z)$ returns the probabilistic error of the inference $\hat{y} = \text{argmax}_y p(y \mid Z)$. Thus, given a time series segment $X_{\tau_i+1:\tau_{i+1}}$, we can obtain the probabilistic error $\mathcal{E}(X_{\tau_i+1:\tau_{i+1}})$ of the inferred activity $\hat{y} = \text{argmax}_y p(y \mid X_{\tau_i+1:\tau_{i+1}})$ on this segment. We propose a segmentation function $F(\boldsymbol{\tau})$ of the transition points $\boldsymbol{\tau}$ as follows:

$$F(\boldsymbol{\tau}) = \max_{\tau_i \in \boldsymbol{\tau} \bigcup \{\tau_0\}} \{\mathcal{E}(X_{\tau_i+1:\tau_{i+1}})\}. \tag{3.4}$$

The function $F(\boldsymbol{\tau})$ returns the maximum error of the segments corresponding to $\boldsymbol{\tau}$. Then, we propose Min-Max Activity Recognition Model (MARM) as:

$$\boldsymbol{\tau}^* = \underset{\boldsymbol{\tau}}{\text{argmin}}\{F(\boldsymbol{\tau})\}, \tag{3.5}$$

where we aim to find an optimal solution $\boldsymbol{\tau}^* = \{\tau_1^*, \tau_2^*, ..., \tau_m^*\}$ such that the maximum error of those segments is minimized. After obtaining $\boldsymbol{\tau}^*$, the current activity is considered to be represented by the last segment $X_{\tau_m^*+1:n}$ and is inferred as:

$$\hat{y}_c = \underset{y}{\text{argmax}} p(y \mid X_{\tau_m^*+1:n}). \tag{3.6}$$

The intuitive explanation of solving MARM is to properly place the transition points by forcing down the maximum of the recognition errors. Since the space size of valid $\boldsymbol{\tau}$ is $\mathcal{O}(2^n)$, exhaustive searching the solution is infeasible. However, we can employ dynamic programming to solve the problem in $\mathcal{O}(n^2)$ inspired from the work of [41]. We claim that the problem of optimizing our model exhibits optimal substructure, i.e., optimal solutions to a problem incorporate optimal solutions to related subproblems. Let X_k be the simplified notation of the time series $X_{1:k}$, and $X_0 = \emptyset$. Let $\boldsymbol{\tau}_k^*$ be an optimal solution on X_k, and $\boldsymbol{\tau}_0^* = \emptyset$. We propose the dynamic programming functional equation (DPFE) to solve MARM (Eq. 3.5) as follows:

$$F_{X_l}(\boldsymbol{\tau}_l^*) = \min_{0 \le k < l} \{\max\{F_{X_k}(\boldsymbol{\tau}_k^*), \mathcal{E}(X_{k+1:l})\}\} \ (0 < l \le n), \tag{3.7}$$

where $\boldsymbol{\tau}_1^*, \boldsymbol{\tau}_2^*, ..., \boldsymbol{\tau}_{l-1}^*$ are the previous optimal solutions that have already been obtained. Then, $\boldsymbol{\tau}_l^*$ is calculated as follows:

$$\boldsymbol{\tau}_l^* = \boldsymbol{\tau}_p^* \bigcup \{p\}, \tag{3.8}$$

where p is the last transition point in $\boldsymbol{\tau}_l^*$ obtained by:

$$p = \underset{0 \le k < l}{\operatorname{argmin}} \{\max \{F_{X_k}(\boldsymbol{\tau}_k^*), \mathcal{E}(X_{k+1:l})\}\}. \tag{3.9}$$

The DPFE in Eq. 3.7 indicates the optimal substructure that an optimal solution $\boldsymbol{\tau}_l^*$ to the problem regarding X_l is derived from the optimal solutions $\boldsymbol{\tau}_1^*, ..., \boldsymbol{\tau}_{l-1}^*$ to the sub-problems regarding $X_1, ..., X_{l-1}$, which are the prefixes of the time series X. We show the correctness of the DPFE in Theorem 1.

Theorem 3.1. *$\boldsymbol{\tau}_l^*$ obtained by Equations 3.8 and 3.9 is an optimal solution to $F_{X_l}(\boldsymbol{\tau})$.*

Proof. We assume $\boldsymbol{\tau}_l^*$ is not an optimal solution of $F_{X_l}(\boldsymbol{\tau})$, and claim that $\boldsymbol{\tau}_l^+$ is an optimal solution. Suppose p is the last transition point of $\boldsymbol{\tau}_l^*$, then

$$F_{X_l}(\boldsymbol{\tau}_l^*) = \max \{F_{X_p}(\boldsymbol{\tau}_p^*), \mathcal{E}(X_{p+1:n})\}, \tag{3.10}$$

where $\boldsymbol{\tau}_p^* = \boldsymbol{\tau}_l^* - \{p\}$. Let q be the last transition point of $\boldsymbol{\tau}_l^+$, then

$$F_{X_l}(\boldsymbol{\tau}_l^+) = \max \{F_{X_q}(\boldsymbol{\tau}_q^+), \mathcal{E}(X_{q+1:n})\}, \tag{3.11}$$

where $\boldsymbol{\tau}_q^+ = \boldsymbol{\tau}_l^+ - \{q\}$. Since $\boldsymbol{\tau}_l^+$ is an optimal solution and $\boldsymbol{\tau}_l^*$ is not, hence $F_{X_l}(\boldsymbol{\tau}_l^+) < F_{X_l}(\boldsymbol{\tau}_l^*)$. But we have:

$$\begin{aligned} F_{X_l}(\boldsymbol{\tau}_l^+) &= \max \{F_{X_q}(\boldsymbol{\tau}_q^+), \mathcal{E}(X_{q+1:l})\} \\ &\ge \max \{F_{X_q}(\boldsymbol{\tau}_q^*), \mathcal{E}(X_{q+1:l})\} \\ &\ge \max \{F_{X_p}(\boldsymbol{\tau}_p^*), \mathcal{E}(X_{p+1:l})\} = F_{X_l}(\boldsymbol{\tau}_l^*), \end{aligned} \tag{3.12}$$

which is a contradiction. Therefore, $\boldsymbol{\tau}_l^*$ is an optimal solution of $F_{X_l}(\boldsymbol{\tau})$ on the time series X_l. □

Algorithm 1 MARM Algorithm

Input: (1) A time series X of length n.
Output: (1) A set of transition points $\boldsymbol{\tau}^*$; (2) The inferred current activity $\hat{y}_c$.

1: $\boldsymbol{\tau}_0^* = \emptyset$
2: $\hat{y}_c = Unkown$
3: $F_{X_0}(\boldsymbol{\tau}_0^*) = 0$
4: **while** $l = 1, 2, ..., n$ **do**
5: $\quad p = \underset{0 \le k < l}{\text{argmin}}\{\max\{F_{X_k}(\boldsymbol{\tau}_k^*), \mathcal{E}(X_{k+1:l})\}\}$ ▷ Using Eq. 3.9.
6: $\quad \boldsymbol{\tau}_l^* = \boldsymbol{\tau}_p^* \bigcup \{p\}$ ▷ Using Eq. 3.8.
7: $\quad$ **if** $l == n$ **then**
8: $\quad\quad \hat{y}_c = \underset{y}{\text{argmax}}\, p(y \mid X_{p+1:l})$ ▷ Predicting the current activity.
9: $\quad$ **end if**
10: **end while**
11: $\boldsymbol{\tau}^* = \boldsymbol{\tau}_n^*$
12: **return** $\boldsymbol{\tau}^*, \hat{y}_c$

Based on the proposed DPFE, we can use dynamic programming to obtain an optimal solution $\boldsymbol{\tau}_n^* \equiv \boldsymbol{\tau}^*$ that minimizes $F_{X_n}(\boldsymbol{\tau}) \equiv F(\boldsymbol{\tau})$. We present the algorithm of solving MARM in Algorithm 1. We explain and analyze the algorithm in terms of time complexity: In lines 4-10, we iteratively calculate $\boldsymbol{\tau}_l^*$ from $l = 1$ to n, and each $\boldsymbol{\tau}_l^*$ is calculated in lines 5-6 with $\mathcal{O}(n)$ time complexity. In summary, the final solution $\boldsymbol{\tau}^*$ can be found in $\mathcal{O}(n^2)$ time complexity. When calculating $\boldsymbol{\tau}_n^*$, the last segment $X_{p+1:n}$ is exhibited, and the current activity is inferred as $\hat{y}_c$, which is shown in line 8.

3.2.3 Weighted Min-Max Activity Recognition Model (WMARM)

MARM finds a set of optimal segments on the observed time series X, and obtains the inference of the current activity based on the last segment. To further improve the inference accuracy for the current activity, we would like to emphasis on reducing the error of the last segment. We propose a new segmentation function $F_{LA}(\boldsymbol{\tau})$ which imposes weights on the last segment and the previous segments. Let $p \equiv \tau_m$ be the last transition point in $\boldsymbol{\tau}$, then $F_{LA}(\boldsymbol{\tau})$ is defined as follows:

$$F_{LA}(\boldsymbol{\tau}) = \max\{(1-\mu) \cdot F_{X_p}(\boldsymbol{\tau} - \{p\}),\ \mu \cdot \mathcal{E}(X_{p+1:n})\}, \tag{3.13}$$

in which a weight parameter $\mu \in [0,1]$ is multiplied to the error of the last segment $X_{p+1:n}$, and $1-\mu$ is multiplied to the maximum error of the previous $m-1$ segments obtained by $F_{X_p}(\boldsymbol{\tau}-\{p\})$. We propose WMARM based on $F_{LA}(\boldsymbol{\tau})$ as follows:

$$\boldsymbol{\tau}^* = \underset{\boldsymbol{\tau}}{\operatorname{argmin}}\{F_{LA}(\boldsymbol{\tau})\}. \tag{3.14}$$

The solution $\boldsymbol{\tau}^*$ of WMARM can be found with the following theorem:

Theorem 3.2. *Given $\boldsymbol{\tau}_1^*, \boldsymbol{\tau}_2^*, ..., \boldsymbol{\tau}_{n-1}^*$, which are the optimal solutions of $F_{X_1}(\boldsymbol{\tau})$, $F_{X_2}(\boldsymbol{\tau})$, ..., $F_{X_{n-1}}(\boldsymbol{\tau})$, respectively. An optimal solution $\boldsymbol{\tau}^*$ of $F_{LA}(\boldsymbol{\tau})$ can be calculated as:*

$$\boldsymbol{\tau}^* = \boldsymbol{\tau}_p^* \bigcup \{p\}, \tag{3.15}$$

where p is the last transition point of $\boldsymbol{\tau}^*$ obtained by:

$$p = \underset{0\le k<n}{\operatorname{argmin}}\{\max\{(1-\mu)\cdot F_{X_k}(\boldsymbol{\tau}_k^*),\ \mu\cdot\mathcal{E}(X_{k+1:n})\}\}. \tag{3.16}$$

Proof. We assume $\boldsymbol{\tau}^*$ is not an optimal solution, and claim that $\boldsymbol{\tau}^+$ is an optimal solution, then

$$F_{LA}(\boldsymbol{\tau}^*) = \max\{(1-\mu)\cdot F_{X_p}(\boldsymbol{\tau}_p^*),\ \mu\cdot\mathcal{E}(X_{p+1:n})\}, \tag{3.17}$$

where $\boldsymbol{\tau}_p^* = \boldsymbol{\tau}^* - \{p\}$. Let q be the last transition point of $\boldsymbol{\tau}_l^+$, then

$$F_{LA}(\boldsymbol{\tau}^+) = \max\{(1-\mu)\cdot F_{X_q}(\boldsymbol{\tau}_q^+),\ \mu\cdot\mathcal{E}(X_{q+1:n})\}, \tag{3.18}$$

where $\boldsymbol{\tau}_q^+ = \boldsymbol{\tau}^+ - \{q\}$. Since $\boldsymbol{\tau}^+$ is an optimal solution and $\boldsymbol{\tau}^+$ is not, then $F_{LA}(\boldsymbol{\tau}^+) < F_{LA}(\boldsymbol{\tau}^*)$. But we have:

$$\begin{aligned} F_{LA}(\boldsymbol{\tau}^+) &= \max\{(1-\mu)\cdot F_{X_q}(\boldsymbol{\tau}_q^+),\ \mu\cdot\mathcal{E}(X_{q+1:n})\} \\ &\ge \max\{(1-\mu)\cdot F_{X_q}(\boldsymbol{\tau}_q^*),\ \mu\cdot\mathcal{E}(X_{q+1:n})\} \\ &\ge \max\{(1-\mu)\cdot F_{X_p}(\boldsymbol{\tau}_p^*),\ \mu\cdot\mathcal{E}(X_{p+1:n})\} \\ &= F_{LA}(\boldsymbol{\tau}^*), \end{aligned} \tag{3.19}$$

which is a contradiction. Therefore, $\boldsymbol{\tau}^*$ is an optimal solution of $F_{LA}(\boldsymbol{\tau})$. □

Algorithm 2 WMARM Algorithm

Input: (1) A time series X of length n.
Output: (1) A set of transition points $\boldsymbol{\tau}^*$; (2) The inferred current activity $\hat{y}_c$.

1: $\boldsymbol{\tau}_0^* = \emptyset$
2: $\hat{y}_c = Unkonwn$
3: $F_{X_0}(\boldsymbol{\tau}_0^*) = 0$
4: **while** $l = 1, 2, ..., n-1$ **do**
5: $\quad p = \underset{0 \le k < l}{\operatorname{argmin}}\{\max\{F_{X_k}(\boldsymbol{\tau}_k^*), \mathcal{E}(X_{k+1:l})\}\}$ ▷ Using Eq. 3.9.
6: $\quad \boldsymbol{\tau}_l^* = \boldsymbol{\tau}_p^* \bigcup \{p\}$ ▷ Using Eq. 3.8.
7: **end while**
8: $p = \underset{0 \le k < n}{\operatorname{argmin}}\{\max\{(1-\mu) \cdot F_{X_k}(\boldsymbol{\tau}_k^*), \mu \cdot \mathcal{E}(X_{k+1:n})\}\}$ ▷ Using Eq. 3.16.
9: $\boldsymbol{\tau}^* = \boldsymbol{\tau}_p^* \bigcup \{p\}$ ▷ Using Eq. 3.15.
10: $\hat{y}_c = \underset{y}{\operatorname{argmax}}\, p(y \mid X_{p+1:n})$ ▷ Predicting the current activity.
11: **return** $\boldsymbol{\tau}^*, \hat{y}_c$

It is worth noting that, If μ is set to 0.5, the model is equivalent to the original MARM without weight. To calculate the solution of WMARM, we present a dynamic programming algorithm in Algorithm 2. Similar to Algorithm 1, Algorithm 2 computes $\boldsymbol{\tau}_1^*$, $\boldsymbol{\tau}_2^*$, ..., $\boldsymbol{\tau}_{n-1}^*$ in $\mathcal{O}(n^2)$, as shown in lines 4-7. Calculating the last transition point p needs to iteratively examine the optimal value of $F_{X_k}(\boldsymbol{\tau}_k^*)$ from $k = 0$ to $n-1$, as shown in line 8, which needs $\mathcal{O}(n)$ time complexity. Then, the final solution $\boldsymbol{\tau}^*$ is obtained by combining p into $\boldsymbol{\tau}_p^*$, shown in line 9. In summary, the total time complexity of Algorithm 2 is $\mathcal{O}(n^2)$. Finally, the last segment $X_{p+1:n}$ is exhibited when calculating $\boldsymbol{\tau}^*$, and the current activity is inferred as $\hat{y}_c$, which is shown in line 10.

3.2.4 Efficient Implementation of WMARM

WMARM partitions a time series on data point level, which results in $\mathcal{O}(n^2)$ time complexity where n is the length of a time series. However, the input time series for activity recognition usually contains hundreds of data points, which produces a large amount of computation regarding the quadratic complexity. To improve the time efficiency of WMARM, we divide the time series into several consecutive frames, and we treat those frames as atomic elements for computation instead of data points. In practice, we presume h lengths: $L_1, L_2, ..., L_h$, where $L_1 = n/h$, $L_i = i \times L_1$ and $L_h = n$. We train a number of h classifiers $C_1, C_2, ..., C_h$, where C_i is trained based on time series subsequences of

Algorithm 3 Efficient Implementation of WMARM

Input: (1) A time series X of length n; (2) A frame size h.
Output: (1) A set of transition points $\boldsymbol{\tau}^*$; (2) The inferred current activity $\hat{y}_c$.

$\boldsymbol{\tau}_0^* = \emptyset$
$\hat{y}_c = Unkonwn$
$F_{X_0}(\boldsymbol{\tau}_0^*) = 0$
while $l = 1, 2, ..., n/h - 1$ **do**
    $p = \underset{0 \le k < l}{\text{argmin}}\{\max\{F_{X_{L_k}}(\boldsymbol{\tau}_k^*),\ \mathcal{E}(X_{L_k+1:L_l})\}\}$
    $\boldsymbol{\tau}_l^* = \boldsymbol{\tau}_p^* \bigcup \{p\}$
end while
$p = \underset{0 \le k < n/h}{\text{argmin}}\{\max\{(1-\mu) \cdot F_{X_{L_k}}(\boldsymbol{\tau}_k^*),\ \mu \cdot \mathcal{E}(X_{L_k+1:n})\}\}$
$\boldsymbol{\tau}^* = \boldsymbol{\tau}_p^* \bigcup \{p\}$
$\hat{y}_c = \underset{y}{\text{argmax}}\, p(y \mid X_{L_p+1:n})$
return $\boldsymbol{\tau}^*, \hat{y}_c$

length L_i. During activity inference, transition points are only considered at every n/h data points of a testing time series X for approximation. Therefore, only subsequences of lengths $L_1, L_2, ..., L_h$ will be fed to the model $p(y \mid Z)$, which utilizes the classifier C_i to estimate the probabilities of subsequences of L_i. Based on this implementation, the number of candidature transition points is reduced by a factor of h^2, which makes WMARM more efficient at inference phase. The pseudocode of the efficient implementation is presented in Algorithm 3.

3.3 Empirical Evaluation

In this section, we evaluate the performance of Weighted Min-max Activity Recognition Model (WMARM) in terms of accuracy on a desktop platform. The experiment scripts are written in Python 2.7 on 64-bit Ubuntu 14.04 LTS operating system. We also evaluate the execution time of the proposed WMARM algorithm on an iPhone 6 (iOS 9.0 system). The source code is written in Objective-C and C++.

Datasets: The experiments are conducted on 5 datasets: (1) Human Activity Sensing Consortium dataset (HASC) [43]; (2) Human Activity Recognition on Smartphones Dataset (HARSD) [5]; (3) Actitracker dataset (ACTR) [53]; (4) Daily Sport Activities dataset (DSA) [11]; (5) Smartphone-Based Recognition of Human Activities and Postural Transitions

dataset (HAPT) [80]. We use the 3-axis acceleration data of those datasets in the experiments.

Experimental Settings: Let $\mathcal{D}$ be a dataset containing a number of time series, where each time series is collected independently from the others. Let $\mathcal{D}_y$ be the subset corresponding to activity y, and we have $\mathcal{D} = \bigcup_y \mathcal{D}_y$. For each subset $\mathcal{D}_y$, we randomly split $\mathcal{D}_y$ into 4 approximately equal sized groups: $\mathcal{D}_{y,1}$, $\mathcal{D}_{y,2}$, ..., $\mathcal{D}_{y,4}$, where the size of each group is between $\lfloor |\mathcal{D}_y|/4 \rfloor$ and $\lceil |\mathcal{D}_y|/4 \rceil$. We conduct experiments based on 4-fold cross-validation: e.g., for the first evaluation, the testing set is aggregated as $\bigcup_y \mathcal{D}_{y,1}$, and the training set is aggregated by all the remaining groups as $\bigcup_y (\mathcal{D}_{y,2} \bigcup \mathcal{D}_{y,3} \bigcup \mathcal{D}_{y,4})$[1]. We perform 5 times of this 4-fold cross-validation and report their average results. To prepare testing data, we generate 100 time series instances. Each instance is of 5 seconds and is randomly formed by $K+1$ segments of different activities with K transitions, where the length of each activity segment is randomly selected with no less than 0.5s. Since an activity is performed on average longer than 2 seconds, we assume that there are at most 3 activity transitions in a 5 seconds period. We thus test $K = 0, 1, 2, 3$ in the experiments.

Settings of WMARM: In the experiments, we use the efficient implementation of WMARM where $h = 10$. We thus train 10 different classifiers regarding time series of lengths from 0.5s to 5.0s (every 0.5s), respectively. The training time series are extracted from the training set by using sliding windows with 50% overlapping. We set each classifier as a random forest with 10 estimators, which uses the $1/3$ lowest frequency Fourier coefficients of an input time series as features. The final inference model $p(y \mid Z)$ consists of those 10 classifiers.

Settings of Baselines: We use 5 baseline methods in comparison with WMARM:

1 Naive: We simply use the entire time series for inference without segmentation.

2 GIR: We use Global Iterative Replacement (GIR) algorithm [37] to segment the time series, where the last segment is used for inference.

3 OPM: We use Optimal Partitioning Method (OPM) [41] to segment the time series.

4 RNN: We use Recurrent Neural Network (RNN) [85] with LSTM + Softmax layers

[1] This cross-validation setting avoids the issue reported in [34] that training and testing data could be highly similar by traditional data splitting ways.

to infer the current activity. We extract the Fourier coefficients on every 0.5s time series frames as stepwise input to the RNN.

5 MSG: We manually obtain the true segment of the current activity for inference, and we denote this method as 'ManualSegment' (MSG).

Note that Naive, GIR, OPM, and MSG methods use the same inference model with WMARM.

3.3.1 Measuring the Accuracy of Current Activity Recognition

To evaluate the accuracy of WMARM for current activity recognition, we compare WMARM with the baselines using datasets: HASC, HAR, ACTR, and DSA. In this experiment, we set the weight $\mu = 0.7$ for WMARM since we experimentally show later that this value of μ obtains the best accuracy among the cross-validation. According to the results shown in Table 3.1, WMARM outperforms Naive, GIR, OPM, and RNN in all cases, except for $K = 0$ on ACTR dataset, since that WMARM always find the optimal segments for recognizing the current activity. Generally, it should be expected that no algorithm can obtain a better accuracy than MSG. However, WMARM outperforms MSG when $K = 0$ on HASC, HAR, and DSA datasets. This is because that WMARM finds the best fitted segment for recognition instead of the true segment, thereby a part of noise can be excluded. In real applications, 0 or 1 transition is mostly happened in a 5-second time period, and it is rare to see more than 2 transitions. Therefore, WMARM can effectively handle most cases base on the experimental results.

To statistically compare the performance of WMARM with the baselines, we conduct the Wilcoxon signed-rank test on their results (80 pairs for each test). The returned p-values represent the lowest level of significance of a hypothesis that results in rejection. This value allows one to determine whether two methods have significantly different performance. We set the significance level $\alpha = 0.05$ for the comparisons. For $K = 1, 2, 3$, the returned p-values (7.747e-15 to 1.199e-13), reject the null hypothesis for the comparisons: WMARM vs. all the baselines except for MSG, which indicates superior performance of WMARM against those methods. For $K = 0$, only the p-value of WMARM vs. GIR (2.477e-06) rejects the null hypothesis, which indicates the similar performances of the

	HASC						HARSD					
K	Naive	GIR	OPM	RNN	WMARM	MSG	Naive	GIR	OPM	RNN	WMARM	MSG
0	88.00%	85.00%	88.55%	17.05%	**89.50%**	88.00%	81.30%	81.00%	82.85%	20.95%	**83.65%**	81.30%
1	35.90%	59.35%	46.25%	37.35%	**72.65%**	83.45%	32.35%	53.60%	45.25%	46.90%	**77.00%**	80.60%
2	22.45%	41.65%	28.45%	40.70%	**56.00%**	73.30%	20.40%	34.10%	29.20%	53.75%	**67.40%**	79.05%
3	16.05%	28.20%	19.85%	43.50%	**46.45%**	69.25%	15.90%	23.85%	22.85%	58.95%	**62.25%**	77.80%
	ACTR						DSA					
K	Naive	GIR	OPM	RNN	WMARM	MSG	Naive	GIR	OPM	RNN	WMARM	MSG
0	75.30%	73.25%	**75.50%**	18.65%	71.30%	75.30%	82.00%	76.00%	82.15%	7.00%	**84.60%**	82.00%
1	30.90%	51.60%	46.15%	43.55%	**63.70%**	72.85%	25.85%	63.75%	32.75%	19.25%	**65.25%**	77.30%
2	23.65%	37.55%	32.20%	50.15%	**61.15%**	70.45%	13.15%	47.80%	18.85%	22.30%	**52.30%**	73.90%
3	16.30%	25.65%	23.10%	50.00%	**54.95%**	67.90%	7.40%	31.85%	12.80%	28.50%	**45.10%**	70.05%

Table 3.1: Results of accuracy on datasets: HASC, HARSD, ACTR, and DSA. K is the number of transitions in the time series. K is not known to the methods.

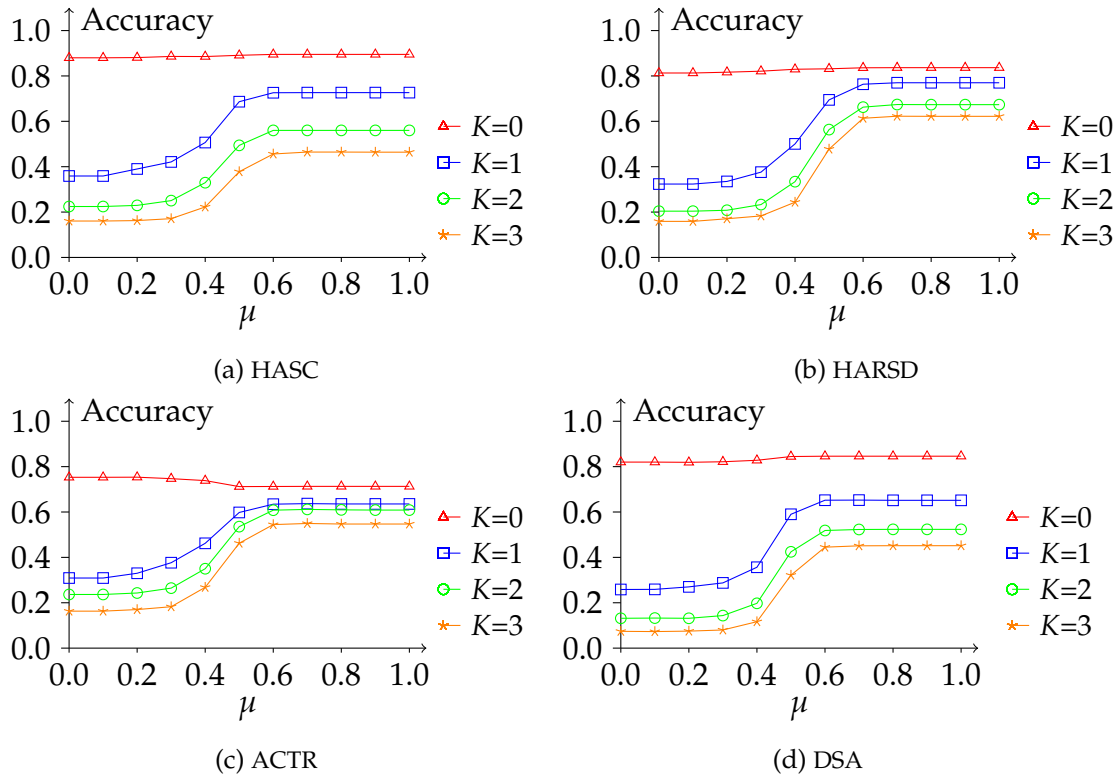
two methods.

3.3.2 Evaluating the Impact of μ on Accuracy

We conduct experiments to evaluate the performance of WMARM with different settings of μ. According to the results shown in Figure 3.2, when there is no transition in the data, i.e., $K = 0$, the accuracy is slightly affected by the weight μ since the optimal result should be only one segment. However, when there are transitions in the data, i.e., $K > 0$, the accuracy of WMARM significantly improves with respect to the increase of μ when $\mu < 0.7$, since the model emphasis more on the last segment, i.e., the current activity. The accuracy normally reaches the maximum around $\mu = 0.7(\pm 0.1)$, then slightly decreases by less than 1%, or becomes stable in a few cases. Since over emphasizing the weight on the last segment may impair the segmentation results on the previous segments, so that the inference accuracy on the last segment is affected by the previous segments.

3.3.3 Measuring the Accuracy on Actual Transitions

In the previous experiments, we used splicing testing instances to study the performance of WMARM. In this experiment, we evaluate WMARM on actual transitions resulting from user's changing activities, for example changing naturally from running to walk-



(a) HASC (b) HARSD

(c) ACTR (d) DSA

Figure 3.2: Accuracy of WMARM with respect to μ on datasets: HASC, HARSD, ACTR, and DSA. For $K = 0$, the accuracy slightly improves on HASC, HARSD, and DSA, and slightly drops on ACTR. For $K = 1, 2, 3$, the accuracy reaches maximum around $\mu = 0.7(\pm 0.1)$, and then slightly decreases by less than 1% or becomes stable.

ing. The instances are extracted from HAPT dataset [80] which provides several long time series containing a protocol of activities. We randomly select 70% of the data for training and the rest for testing, and repeat the experiment 10 times. The training instances are extracted during the activities, and the testing instances are extracted between transitions. WMARM obtains 75.41% accuracy, which outperforms the baselines: Naive (30.34%), GIR (49.44%), OPM (39.97%), and RNN (55.82%), except for MSG (76.69%). To explore the statistical significance of the performances of the methods on handling actual transitions, we conduct the Wilcoxon signed-rank test on their results (10 pairs). We set the significance level $\alpha = 0.05$ for the comparison. The p-values of WMARM vs. Naive/GIR/OPM/RNN (0.003346 for all), reject the null hypothesis for the accuracy measurements, implying a significant improvement of WMARM over those methods.

3.3.4 Evaluating the Execution Time on Smartphone

To evaluate the execution time of the WMARM algorithm, we develop an iOS App on iPhone 6 using Objective-C, and implement the WMARM algorithm as an internal function using C++. The App captures the 3-axis acceleration data with 100 data points per second, which is supplied to WMARM algorithm for processing. We observe a total execution time for 500 runs, and calculate the average time. WMARM algorithm only costs averagely 0.0153 seconds for one execution, which is not expensive for running AR systems on smartphones. Naive method costs averagely 0.0012 seconds for one execution, but its accuracy is much lower than WMARM.

3.4 Conclusion

In this chapter, we highlighted a problem normally presented in Human Activity Recognition (HAR) that traditional methods usually fail to recognize the current activity in the presence of multiple activities. To solve this problem, we devised Weighted Min-max Activity Recognition Model (WMARM), which recognizes the current activity by optimally partitioning the observed window of time series matching the activities presented. WMARM considers weights on the partitioned segments to obtain reliable recognition accuracy. WMARM can also effectively process the time series containing an arbitrary number of transitions without any prior knowledge about the number of transitions. Instead of exhaustively searching the optimal solution of WMARM in exponential space, we proposed an efficient dynamic programming algorithm that computes the model in $\mathcal{O}(n^2)$ time complexity, where n is the length of the window. Moreover, we presented an efficient implementation of WMARM that the computation cost can be further reduced. Extensive experiments on 5 real datasets have demonstrated the superior performance of WMARM on handling time series with one or more activity transitions. The results show about 10%-30% improvement on the accuracy of current activity recognition compared with state-of-the-art methods. The experiment on iPhone 6 shows the prominent computational efficiency of WMARM.

Chapter 4

Accurate Predicting of Complex Activities from Ongoing Observations

In this chapter, we conduct a study on accurate predicting of complex activities (CAs) from ongoing observations. The rapid development of sensor networks enables recognition of complex activities using multivariate time series. However, CAs are usually performed over long periods of time, which causes slow recognition by models based on fully observed data. Therefore, predicting CAs at early stages becomes an important problem. In this chapter, we propose Simultaneous Complex Activities Recognition and Action Sequence Discovering (SimRAD), an algorithm which predicts a CA over time by mining a sequence of multivariate actions from sensor data using a Deep Neural Network. SimRAD simultaneously learns two probabilistic models for inferring CAs and action sequences, where the estimations of the two models are conditionally dependent on each other. SimRAD continuously predicts the CA and the action sequence, thus the predictions are mutually updated until the end of the CA. We conduct evaluations on a real-world CA dataset consisting of a rich amount of sensor data, and the results show that SimRAD outperforms state-of-the-art methods by average 7.2% in prediction accuracy with high confidence.

4.1 Introduction

Due to the rapid development of sensor networks, recognition of complex human activities directly from multivariate time series (MTS) are becoming feasible for artificial intelligence systems to understand multiplex human behaviors. The classic models for activity recognition are based on time series of fully observed activities. However, com-

This chapter is derived from: Weihao Cheng, Sarah Erfani, Rui Zhang, Kotagiri Ramamohanarao, "Predicting Complex Activities from Ongoing Multivariate Time Series", *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden, 2018.

plex activities (CAs), such as 'cooking', generally have much longer durations compared with simple activities, a.k.a. actions, such as 'grabbing' and 'lifting'. Therefore, using the classic models for CAs will result in late recognition. For example in safety monitoring, a system is required to continuously predict dangerous CAs without full observations to avert or minimize their consequences. As a result, we need a method that can predict CAs given sensory MTS data at arbitrary early stages.

Early recognition of human activities is first studied in the computer vision field [58, 66, 84]. The proposed methods focus on video streams, where visual features/actions are computed from the video frames and then used for early prediction. However, compared with videos, sensor MTS data contains much less information regarding each timestamp. For example, an image of a RGB video usually has more than hundreds of pixels, but a data point of an accelerometer signal has only 3 values (accelerations on 3 axises). Therefore, early recognition of human activities based on sensor MTS data incurs substantial challenges. There are a few studies for early classification on time series data [3, 22, 27, 61, 96, 97]. However, the task of these methods is finding one optimal early stage to classify a time series, thus they are not suitable for predicting CAs at any early stages.

We consider that a stream of data points are sequentially received from sensors, and our goal is to devise a method that can continuously predict a CA from an ongoing MTS. A straightforward way is to build classifiers at several predefined stages of CAs, and use the corresponding classifier for prediction. However, during the inference procedure, it is impossible to determinate the exact stage of a CA given the observed MTS, since CAs are performed with different durations, for example, a 'cooking' activity may last from few minutes to hours. Therefore, it is a challenging task to devise a model that can predict CAs at arbitrary early stages. We propose Simultaneous Complex Activities Recognition and Action Sequence Discovering (SimRAD), an algorithm which predicts the CA over time from an ongoing MTS. As a CA can be characterized by a temporal composition of actions, SimRAD discovers and utilizes a sequence of multivariate actions from the observed MTS as primitives to infer the CA. SimRAD learns two conditional probabilistic models: **action sequence model** (ASM) and **complex activity model** (CAM). The ASM uses the MTS and an estimated CA to infer the action sequence. This model is designed as a Deep Neural Network (DNN) feature learner with category weights for recogniz-

ing actions based on different CAs. We jointly learn the DNN and the category weights by optimizing a novel objective function which augments the robustness of the prediction, thus the ASM can reliably infer the action sequence by means of a CA estimation. The CAM uses an estimated action sequence to infer the CA, where the feature of temporal patterns is extracted for CA classification. We learn the CAM based on the entire progressions of action sequences, thus the CA can be inferred at any progress level. SimRAD alternately updates the predictions of the two models based on each other, until the completion of the CA. More specifically, our main contributions are summarized as:

- We propose SimRAD which predicts a CA over time by discovering an action sequence from the observed MTS. SimRAD learns two conditional probabilistic models to infer action sequences and CAs. SimRAD predicts CAs by alternately using the two models where their predictions are mutually updated based on each other.

- We propose the action sequence model (ASM) which is constructed by a DNN feature learner with category weights for recognizing actions based on different CAs. The feature learner and category weights are jointly learned by optimizing a novel objective function which can enhance the prediction's robustness.

- We propose the complex activity model (CAM) which captures temporal patterns from the estimated action sequence for CA classification. The CAM is learned based on the entire progressions of action sequences so that the CA can be inferred regardless of its progress level.

We conduct experiments on a real-world CA dataset consisting of a rich amount of sensor data. The evaluation results show that SimRAD outperforms state-of-the-art methods by average 7.2% in prediction accuracy with high confidence.

4.2 Methodology

In this section, we first formalize our problem. Then, we propose Simultaneous Complex Activities Recognition and Action Sequence Discovering (SimRAD) for predicting complex activities (CAs) from ongoing multivariate time series (MTS).

4.2.1 Problem Statement

Let X be an MTS collected by a set of sensors. We present X as a sequence such that $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ where $n = |X|$ is the length of X and $\boldsymbol{x}_t \in \mathbb{R}^d$ is a d dimension data point at time $t \in [1, n]$. Let $\mathcal{Y}$ be a label set of CAs such that $\mathcal{Y} = \{1, 2, ..., |\mathcal{Y}|\}$. Let $y \in \mathcal{Y}$ be the category label of the CA represented by X. Let X_T be the MTS of a fully observed CA, where T is the length of X_T. Let X_t be a prefix of X_T that $X_t = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_t\}$ where $1 \leq t \leq T$. Suppose we have a data distribution $\mathcal{D} = \{(X^{(i)}, y^{(i)})\}$, where $X^{(i)} \equiv X^{(i)}_{T^{(i)}}$ is the MTS of a fully observed CA, and $y^{(i)}$ is the CA label. We formalize the problem of predicting CAs as finding a probability model $p(y \mid X)$ such that:

$$\max \mathbb{E}_{(X^{(i)}, y^{(i)}) \sim \mathcal{D}} \sum_{t=1}^{T^{(i)}} \log p(y = y^{(i)} \mid X = X^{(i)}_t). \tag{4.1}$$

The objective of Eq. 4.1 is to find a model that maximizes the expected probabilities of correct predictions over $\mathcal{D}$ at all time points. However, due to the non-trivial patterns of CAs, modeling $p(y \mid X)$ that achieves a promised accuracy is a difficult task in practice. To address this problem, we consider discovering the actions appearing in X for facilitating the prediction of y. Let $\mathcal{Z}$ be a label set of actions that $\mathcal{Z} = \{1, 2, ..., |\mathcal{Z}|\}$. Each $\boldsymbol{x}_t$ of X can be mapped into an action vector $\boldsymbol{a}_t \in \{0, 1\}^{|\mathcal{Z}|}$, of which the z-th element $a_{t,z} = 1$ indicates the presence of the action $z \in \mathcal{Z}$, and $a_{t,z} = 0$ indicates the absence of z. Note that multiple actions can appear at time t. We denote A as an action sequence such that $A = \{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n\}$, and we denote A_t as the action sequence corresponding to X_t. With the notations defined above, we present our method in the following sections.

4.2.2 Simultaneous Complex Activities Recognition and Action Sequence Discovering (SimRAD)

We propose Simultaneous Complex Activities Recognition and Action Sequence Discovering (SimRAD) for predicting CAs. Instead of directly inferring the CA label y from an MTS X, we estimate an action sequence A corresponding to X, and use A to infer y. Moreover, we consider that an estimation of y can also assist the estimation of A, since CAs of the same category have similar compositions of actions. As a result, we design a method where y and A are mutually updated based on each other during the inference.

Towards this goal, we propose two conditional probabilistic models. The first model is **action sequence model** (ASM):

$$p(A \mid y, X\,;\, \theta), \tag{4.2}$$

which estimates the probability of the action sequence A given y and X parameterized by θ. The second model is **complex activity model** (CAM):

$$p(y \mid A\,;\, \psi), \tag{4.3}$$

which estimates the probability of the CA label y given A parameterized by ψ. Suppose we are at time t. SimRAD first uses ASM to infer A_t by estimating $p(A = A_t \mid y = y_{t-1}, X = X_t; \theta)$ where y_{t-1} is the label inferred at time $t-1$, and uses CAM to infer y_t by estimating $p(y = y_t \mid A = A_t; \psi)$. Accordingly, SimRAD predicts CAs over time by utilizing the two models alternately. We provide the details of SimRAD in the rest of this section. First, we introduce ASM and CAM by describing the model structures and learning objectives. Then, we give the details of SimRAD in terms of training and inference algorithms.

Action Sequence Model (ASM)

The ASM $p(A \mid y, X\,;\, \theta)$ is used to infer A given y and X. It is worth noting that y can take an additional value of 0, which indicates the unknown of an estimated CA. This enables that the inference of SimRAD can be started at the initial time $t = 1$, where y is unknown. Designing the model $p(A \mid y, X\,;\, \theta)$ is non-trivial. First, the model should accurately recognize the actions from a given X of any progress level. Second, the model should collaboratively utilize y and X for inferring A. We rewrite the probability $p(A \mid y, X)$ as:

$$p(A \mid y, X) = p(\boldsymbol{a}_1, \boldsymbol{a}_2 ..., \boldsymbol{a}_n \mid y, X). \tag{4.4}$$

As a CA can be performed in many different ways, it is hard to recognize actions by capturing their temporal dependencies. Therefore, we make the Naive Bayes assumption on A that the actions $\boldsymbol{a}_1, ..., \boldsymbol{a}_n$ are conditionally independent given y and X. In addition, applying the Naive Bayes assumption brings two advantages: 1) the resulting model is

able to handle an observed X of arbitrary progress level, and 2) the modeling can be significantly simplified for incorporating y. We then have:

$$p(A \mid y, X) = \prod_{t=1}^{n} p(\boldsymbol{a}_t \mid y, X) = \prod_{t=1}^{n} \prod_{z=1}^{|\mathcal{Z}|} p(a_{t,z} \mid y, X). \tag{4.5}$$

Therefore, estimating the probability $p(A \mid y, X)$ of the whole sequence can be decoupled into estimating point-wise probability $p(a_{t,z} \mid y, X)$. Let $\boldsymbol{a}$ be a general action vector, and a_z be the z-th element of $\boldsymbol{a}$. We estimate $p(a_{t,z} \mid y, X)$ by probability $p(a_z \mid y, X, t)$ for any given t. Due to the fact that:

$$p(a_z = 1 \mid y, X, t) + p(a_z = 0 \mid y, X, t) = 1, \tag{4.6}$$

we only need to formulate the probability $p(a_z = 1 \mid y, X, t)$, and $p(a_z = 0 \mid y, X, t)$ is obtained by $1 - p(a_z = 1 \mid y, X, t)$. Let $p(a_z \mid y, X, t\,;\, W, G)$ be our prospective probability model parameterized by W and G. We model $p(a_z = 1 \mid y, X, t\,;\, W, G)$ as:

$$p(a_z = 1 \mid y, X, t\,;\, W, G) = \frac{1}{1 + \exp\{-W_{y,z}^T G(X, t)\}}, \tag{4.7}$$

where $W = \{W_{y,z} \mid 0 \leq y \leq |\mathcal{Y}|, 1 \leq z \leq |\mathcal{Z}|\}$ is the category weights, and $G(X, t)$ is the feature learner to extract features from X at t. The category weights W includes a set of vectors $W_{y,z}$, one for each pair of y and z. The feature learner $G(X, t)$ is a Deep Neural Network (DNN) which yields an effective representation of MTS data. Recall that X is a sequence of d-dim data point, so X can be represented by d channels of univariate time series. The first level of $G(X, t)$ is an input layer. This layer captures a subsequence $X_{t-w/2:t+w/2}$ where w is the window size, and decomposes $X_{t-w/2:t+w/2}$ into d inputs for each channel. The second level of $G(X, t)$ is a group of Fully Connected (FC) layers, where the κ-th FC layer takes the κ-th input and outputs a vector for a channel-independent representation. Then, we concatenate the outputs of those FC layers into one vector using a concatenation layer. Next, we use a Max-Pooling layer to reduce the dimension of the concatenated vector, and then we use a FC layer to extract a feature vector from the pooled vector. This feature vector is the final output of $G(X, t)$. The neural network structure of $G(X, t)$ is illustrated in Figure 4.1. Given a

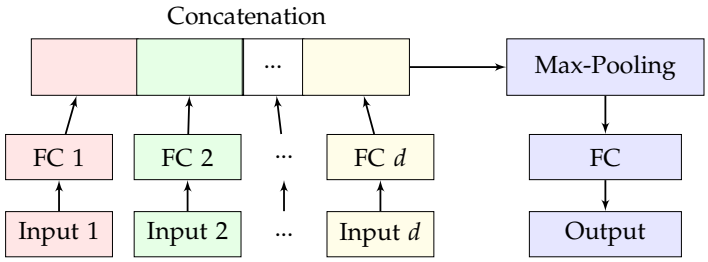


Figure 4.1: The neural network structure of the feature learner G. The inputs 1, ..., d are the univariate time series on each channel of X. The output of G is the learned feature vector.

dataset $\mathcal{D} = \{(X^{(i)}, y^{(i)}, A^{(i)})\}$, where $A^{(i)} = \{\boldsymbol{a}_1^{(i)}, \boldsymbol{a}_2^{(i)}, ..., \boldsymbol{a}_{T^{(i)}}^{(i)}\}$ is the ground-truth action sequence corresponding to $X^{(i)}$, we learn the model $p(a_z \mid y, X, t\,;\, W, G)$ in an end-to-end fashion as follows:

$$\max_{W,G} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{T^{(i)}} \sum_{z=1}^{|\mathcal{Z}|} [\log p(a_z = a_{t,z}^{(i)} \mid 0, X^{(i)}, t\,;\, W, G) + \tag{4.8}$$
$$\log p(a_z = a_{t,z}^{(i)} \mid y^{(i)}, X^{(i)}, t\,;\, W, G) +$$
$$\sum_{y' \in \mathcal{Y}/\{y^{(i)}\}} \log p(a_z = 0 \mid y', X^{(i)}, t\,;\, W, G)],$$

In the above equation, we jointly maximize three probabilities:

(1) $p(a_z = a_{t,z}^{(i)} \mid 0, X^{(i)}, t\,;\, W, G)$ is maximized to enhance the weight $W_{0,z}$ regarding the unknown of y;

(2) $p(a_z = a_{t,z}^{(i)} \mid y^{(i)}, X^{(i)}, t\,;\, W, G)$ is maximized to enhance the weight $W_{y^{(i)},z}$ regarding $y^{(i)}$;

(3) $p(a_z = 0 \mid y', X^{(i)}, t\,;\, W, G)$ for $y' \in \mathcal{Y}/\{y^{(i)}\}$ are maximized to improve the robustness of the model, since this will force $a_z = 0$ when the given y is mistakenly estimated by the CAM during inference.

After finding the optimal W^* and G^*, we obtain the model $p(A \mid y, X\,;\, \theta^*)$ for inferring A, where $\theta^* = (W^*, G^*)$.

Complex Activity Model (CAM)

The CAM $p(y \mid A\,;\,\psi)$ is used to infer y given A. Let $\phi(A)$ be a function that extracts features on the sequence A. We specifically extract the feature of Temporal Patterns [40, 63], because this feature can effectively capture the temporal relations among actions appearing in arbitrary stages of CAs. Moreover, the Temporal Patterns feature can be represented by a vector of fixed dimension, which enables the training and inference of CAM using action sequences of varied lengths. We propose $p(y \mid A\,;\,\psi)$ as follows:

$$p(y = h \mid A\,;\,\psi) = \frac{\exp\{-\psi_h^T \phi(A)\}}{\sum_{j=1}^{|\mathcal{Y}|} \exp\{-\psi_j^T \phi(A)\}}, \tag{4.9}$$

where $\psi = \{\psi_y | 1 \leq y \leq |\mathcal{Y}|\}$ is the weight parameter, and ψ includes a set of vectors ψ_y as the weights for each y. In the learning procedure of $p(y = h \mid A\,;\,\psi)$, we consider that the penalties for the predictions are monotonically increasing with respect to time. This is because that a model should be more certain on the correct prediction when CAs are performed at a later stage [66]. Therefore, the CAM can better capture the progressions of CAs with the increasing penalties. Given a dataset $\mathcal{D} = \{(A^{(i)}, y^{(i)})\}$, we learn the CAM $p(y \mid A\,;\,\psi)$ as follows:

$$\max_{\psi} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{T^{(i)}} \lambda_t \log p(y = y^{(i)} \mid A_t^{(i)}\,;\,\psi), \tag{4.10}$$

where λ_t is the penalty weight that $0 < \lambda_1 <, ..., < \lambda_{T^{(i)}} \leq 1$. After finding the optimal ψ^*, we obtain the model $p(y \mid A\,;\,\psi^*)$ for inferring y.

SimRAD Algorithms

Given the ASM $p(A \mid y, X\,;\,\theta)$ and the CAM $p(y \mid A\,;\,\psi)$, we introduce the processes of SimRAD to estimate A and y with the two models. SimRAD considers the prediction confidences of the CAs, which are presented by a group of probabilities p_0, p_1, ..., $p_{|\mathcal{Y}|}$ such that $\sum_{y=0}^{|\mathcal{Y}|} p_y = 1$. p_0 indicates the confidence of that CA is unknown, and p_y indicates the confidence on y, for $1 \leq y \leq |\mathcal{Y}|$. We initialize $p_0 = 1$, and $p_y = 0$ for $1 \leq y \leq |\mathcal{Y}|$. By taking into account the confidences, SimRAD uses the ASM to estimate

A by first calculating the probability $p(A \mid X)$ as:

$$p(A \mid X) = \sum_{y=0}^{|\mathcal{Y}|} p_y \cdot p(A \mid y, X; \theta), \tag{4.11}$$

and then obtain the estimation $\hat{A} = \{\hat{\boldsymbol{a}}_1, ..., \hat{\boldsymbol{a}}_n\}$ as:

$$\hat{A} = \underset{A}{\operatorname{argmax}}\, p(A \mid X). \tag{4.12}$$

We present $\hat{A}$ as a sequence of estimated action vectors $\hat{\boldsymbol{a}}_t$ such that $\hat{A} = \{\hat{\boldsymbol{a}}_1, ..., \hat{\boldsymbol{a}}_n\}$, where $\hat{\boldsymbol{a}}_t = \{\hat{a}_{t,1}, ..., \hat{a}_{t,|\mathcal{Z}|}\}$. Let $p_{t,z} = p(a_z = 1 \mid y, X, t)$. It can be derived that an $\hat{a}_{t,z}$ of $\hat{\boldsymbol{a}}_t$ is given by $\hat{a}_{t,z} = 1\{\sum_{y=0}^{|\mathcal{Y}|} p_y \cdot p_{t,z} > 0.5\}$. However, if there is a subset $\mathcal{S} \subseteq \mathcal{Z}$ such that the actions of $\mathcal{S}$ are mutually exclusive and one action must appear, then we obtain $\hat{A}$ by setting $\hat{a}_{t,z^*} = 1$ where $z^* = \text{argmax}_{z \in \mathcal{S}} \sum_{y=0}^{|\mathcal{Y}|} p_y \cdot p_{t,z}$, and setting $\hat{a}_{t,z} = 0$ for $z \in \mathcal{S}/\{z^*\}$. With the estimation $\hat{A}$, SimRAD can use the CAM to obtain the estimation $\hat{y}$ as:

$$\hat{y} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p(y \mid \hat{A}; \psi). \tag{4.13}$$

We then update $p_0 = 1/2$, and $p_y = 1/2 \cdot p(y \mid \hat{A}; \psi)$ for $1 \leq y \leq |\mathcal{Y}|$. The p_0 is updated to $1/2$ as we presume a half confidence on the predictions of the CA. Next, we provide the details of the SimRAD algorithms regarding both training and inferring procedures.

SimRAD Training: Given a training dataset $\mathcal{D}$, we first learn the ASM's parameter $\theta = (W, G)$ using $\mathcal{D}$ by Eq. 4.8. Then, we initialize the CAM's parameter ψ and iteratively learn ψ for L rounds. In each round, we use the estimated action sequences instead of ground truth for learning ψ. This makes ψ adapt to θ and achieves desired performance. For an instance $(X^{(i)}, y^{(i)}, A^{(i)}) \in \mathcal{D}$, we calculate the estimated action sequence $\hat{A}^{(i)}$ by Eq. 4.11 and Eq. 4.12. Then, the $\hat{A}^{(i)}$s together with $y^{(i)}$s form a new set of instances $\mathcal{D}'$, which is used to learn/update ψ. After learning/updating ψ in this round, we can update the $p_y^{(i)}$s for each instance of $\mathcal{D}$. The $p_y^{(i)}$s will be used in the next round for estimating $\hat{A}^{(i)}$. The pseudocode of SimRAD's training algorithm is provided in Algorithm 4.

SimRAD Inference: Suppose an ongoing MTS is given by a series of MTS prefixes $X_1, X_2, ..., X_T$, we predict A and y for T steps. At time t, we calculate the estimated action

Algorithm 4 SimRAD Training

Input: A dataset $\mathcal{D} = \{(X^{(i)}, y^{(i)}, A^{(i)})\}$
Output: The optimal parameters θ^* and ψ^*

Learn $\theta = (W, G)$ using $\mathcal{D}$ by Eq. 4.8
Initialize $p_0^{(i)} = 1$, $p_1^{(i)} = 0$, ..., $p_{|\mathcal{Y}|}^{(i)} = 0$ for $i \in \{1, ..., |\mathcal{D}|\}$
for $l = 1, 2, ..., L$ **do**
    Let $\mathcal{D}' = \emptyset$
    for $i = 1, 2, ..., |\mathcal{D}|$ **do**
        Estimate $\hat{A}^{(i)}$ using $p_y^{(i)}$s by Eq. 4.11 and Eq. 4.12
        $\mathcal{D}' = \mathcal{D}' \bigcup \{(\hat{A}^{(i)}, y^{(i)})\}$
    end for
    Update ψ using $\mathcal{D}'$ by Eq. 4.10
    for $i = 1, 2, ..., |\mathcal{D}|$ **do**
        Update $p_0^{(i)} = 1/2$, and $p_y^{(i)} = 1/2 \cdot p(y \mid \hat{A}^{(i)}; \psi)$ for $y \in \mathcal{Y}$
    end for
end for
return $\theta^* = \theta$, $\psi^* = \psi$

Algorithm 5 SimRAD Inference

Input: A series of MTS prefixes $X_1, X_2, ..., X_T$
Output: A series of CA inference outputs $\hat{y}_1, \hat{y}_2, ..., \hat{y}_T$

Initialize $p_0 = 1$, $p_1 = 0$, ..., $p_{|\mathcal{Y}|} = 0$
for $t = 1, 2, ..., T$ **do**
    Estimate $\hat{A}_t$ using p_ys by Eq. 4.11 and Eq. 4.12
    Update $p_0 = 1/2$, and $p_y = 1/2 \cdot p(y \mid \hat{A}_t; \psi)$ for $y \in \mathcal{Y}$
    $\hat{y}_t = \text{argmax}_{y \in \mathcal{Y}} p_y$
end for
return $\hat{y}_1, \hat{y}_2, ..., \hat{y}_T$

sequence $\hat{A}_t$ by Eq. 4.11 and Eq. 4.12. Then, we update the p_ys, and obtain the inferred CA by $\hat{y}_t = \text{argmax}_{y \in \mathcal{Y}} p_y$. The pseudocode of SimRAD's inference algorithm is provided in Algorithm 5.

4.3 Experiments

The experiments are conducted on a 64-bit Ubuntu 14.04 LTS operating system. The experimental scripts are written in Python 2.7 with the use of Scikit-learn [70] and Keras[1] packages.

[1] https://keras.io/

Dataset: The experiments are conducted on Opportunity (OPP) dataset [81]. OPP dataset is collected by 4 subjects with sensors placed on their bodies (back, hands, waists, arms, etc), where the sensors' sampling rate is 30 readings per seconds. Each subject is asked to perform 5 complex activities (CAs): *relaxing*, *early morning*, *coffee time*, *sandwich time*, and *cleanup*. The *relaxing* has 40 instances. For the other CAs, each category has 20 instances. The actions appearing in those CAs can be categorized into 3 types: 5 locomotion actions (LocA), 14 left-hand actions (LHA), and 14 right-hand actions (RHA). Each type includes an abnormal action which presents some undefined behaviors, and the other are defined actions, such as 'grabbing'. The actions of each type are mutually exclusive. At any time point, there will be presence of 3 concurrent actions one for each type.

Data Preparation: Although we have only one available CA dataset, the dataset contains a rich amount of data collected by sensors placed on various positions of subjects' body. Therefore, based on OPP dataset, we generate 3 sub-datasets: OPP-BH, OPP-BW, and OPP-BUA for empirical evaluation. Each sub-dataset contains 3 trails of 3-axis accelerations collected by sensors placed on 3 different locations of body. The details of the sensor placement locations of the 3 sub-datasets are: 1) OPP-BW: Back, Left Waist, Right Waist; 2) OPP-BH: Back, Left Hand, Right Hand; 3) OPP-BUA: Back, Left Upper Arm, Right Upper Arm. Let $\mathcal{D}$ be a sub-dataset, we describe a data instance $(X^{(i)}, y^{(i)}, A^{(i)})$ of $\mathcal{D}$ as: $X^{(i)}$ is the multivariate time series (MTS) of the 3 sensors' acceleration data, thereby, the dimension d of each data point $\boldsymbol{x}_t^{(i)}$ is 9 (9 channels = 3 sensors $\times$ 3 axises); $y^{(i)}$ is the CA label; $A^{(i)}$ is the action sequence, where the dimension of each $\boldsymbol{a}_t^{(i)}$ is 33 (33 = 5 LocA + 14 LHA + 14 RHA).

Experimental Settings: A sub-dataset $\mathcal{D}$ contains a number of data instances, where each instance is collected independently from the other. Let $\mathcal{D}_y$ be the set corresponding to y, and we have $\mathcal{D} = \bigcup_{y \in \mathcal{Y}} \mathcal{D}_y$. For each $\mathcal{D}_y$, we randomly split $\mathcal{D}_y$ into 4 approximately equal sized groups: $\mathcal{D}_{y,1}$, $\mathcal{D}_{y,2}$, ..., $\mathcal{D}_{y,4}$, where the size of each group is between $\lfloor |\mathcal{D}_y|/4 \rfloor$ and $\lceil |\mathcal{D}_y|/4 \rceil$. We conduct experiments based on 4-fold cross-validation: e.g., for the first evaluation, the training set is aggregated as $\bigcup_{y \in \mathcal{Y}} \mathcal{D}_{y,1}$, and the testing set is aggregated by all the remaining groups as $\bigcup_{y \in \mathcal{Y}} (\mathcal{D}_{y,2} \bigcup \mathcal{D}_{y,3} \bigcup \mathcal{D}_{y,4})$[2]. The reason that we use 1/4 for training and 3/4 for testing is that CAs can be performed various individual ways in the

[2]This cross-validation setting avoids the issue reported in [34] that training and testing data could be highly similar by traditional data splitting ways.

real world, so the amount of testing data should be greater than training data. The evaluation results are reported as the average results of the 4 folds.

Settings of SimRAD: For the action sequence model (ASM), we describe the detailed settings of feature learner G from bottom to top as follows: The input layer uses window size $w = 120$ for channels of locomotion actions, and uses the half size of w for channels of left/right-hand actions as hand actions are shorter than locomotion actions; The FC layers of each channel output 60-dim vectors; The Max-Pooling layer down-samples the concatenated vector by a scale of 2; The last FC layer outputs a 256-dim vector. We train the ASM with batch size of 10 and training epoch of 50. For the complex activity model (CAM), we set the feature $\phi(A)$ as Temporal Patterns of 1-pattern [63], and we use quadratic penalty weight $\lambda_t = (t/T)^2$ for $1 \leq t \leq T$. We train SimRAD with learning rounds $L = 10$. The settings of window size w, penalty weight λ_t, and learning round L will be studied in subsection 4.3.1.

Settings of Comparison Methods: We test 5 methods in comparison with SimRAD. The settings of the methods are as following:

1 1NN+DTW: We use 1-nearest neighbor (1NN) method to classify MTS based on Dynamic Time Wrapping (DTW) distance.

2 1NN+Fea: For each channel of MTS, we extract AR feature (mean, standard deviation, average absolute difference, average resultant acceleration, time between peak, and binned distribution) [53] and Fourier coefficients of 0Hz, 1Hz, and 2Hz. We combine all these features as a feature vector, and use 1NN to classify the feature vectors based on Euclidean distance.

3 DT+Fea: We extract the same features used in 1NN+Fea, and use decision tree (DT) to classify the feature vectors.

4 LSTM: We implement a Recurrent Neural Network (RNN) based method, where the architecture of the RNN consists of an LSTM layer with 256-dim output, a dense layer with K-dim output, and a softmax layer.

5 MD-MPP: We implement Multilevel-Discretized Marked Point-Process (MD-MPP) based method, where the piece and level parameters are both set to 10 [59].

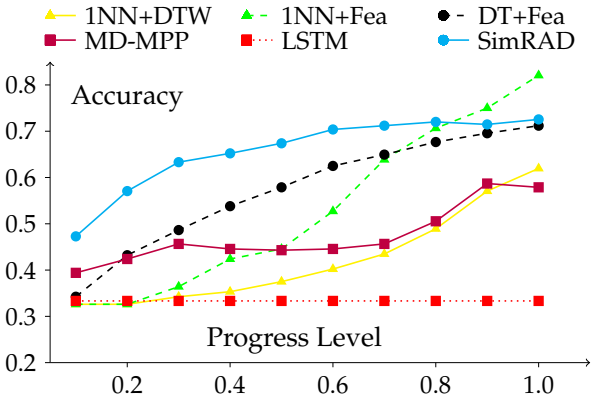


Figure 4.2: Prediction accuracies of CAs at different progress levels on OPP-BW dataset

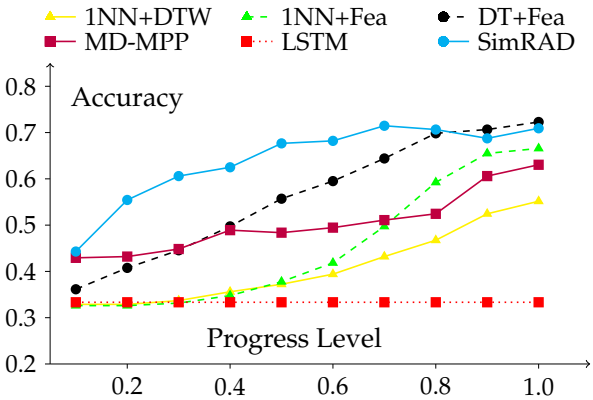


Figure 4.3: Prediction accuracies of CAs at different progress levels on OPP-BH dataset

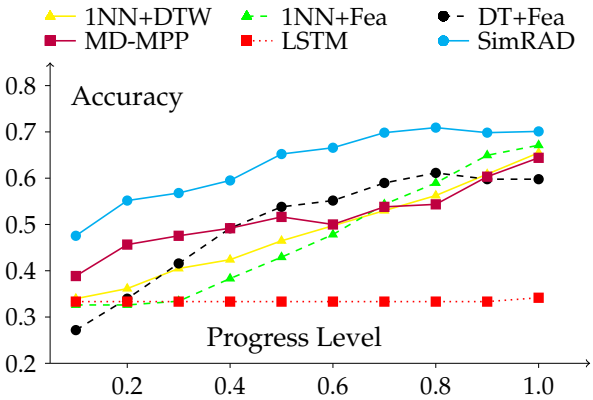


Figure 4.4: Prediction accuracies of CAs at different progress levels on OPP-BUA dataset

4.3.1 Prediction of Complex Activities

We evaluate the performances of SimRAD and the 5 comparison methods regarding prediction accuracies of CAs at progress levels 0.1, 0.2, ..., 1.0, where the progress level is

	SimRAD vs. 1NN+DTW			SimRAD vs. 1NN+Fea			SimRAD vs. DT+Fea		
Dataset	R^+	R^-	p-value	R^+	R^-	p-value	R^+	R^-	p-value
OPP-BW	**817.0**	3.0	0.0000	**720.0**	100.0	0.0000	**692.0**	128.0	0.0002
OPP-BH	**820.0**	0.0	0.0000	**809.0**	11.0	0.0000	**738.0**	82.0	0.0000
OPP-BUA	**810.0**	10.0	0.0000	**800.0**	20.0	0.0000	**769.0**	51.0	0.0000
	SimRAD vs. LSTM			SimRAD vs. MD-MPP					
Dataset	R^+	R^-	p-value	R^+	R^-	p-value			
OPP-BW	**820.0**	0.0	0.0000	**761.0**	59.0	0.0000			
OPP-BH	**820.0**	0.0	0.0000	**696.0**	124.0	0.0001			
OPP-BUA	**820.0**	0.0	0.0000	**701.0**	119.0	0.0001			

Table 4.1: The Wilcoxon test to compare the prediction accuracies regarding R^+, R^-, and p-values.

defined as t/T. According to the evaluation results shown in Figure 4.2, 4.3, and 4.4, SimRAD outperforms all the comparison methods in most of the times, except progress levels of 0.9 and 1.0 on OPP-BW and OPP-BH datasets. SimRAD uniformly surpasses the best accuracies of the comparison methods by average 7.2%. We find that the LSTM based model is not effectively learned in the experiments. This might due to insufficient number of data instances in the datasets. To statistically compare the performance of SimRAD with the 5 methods, we conduct the Wilcoxon signed-ranked test on the results (40 pairs) of the 3 datasets. The returned R^+ and R^- correspond to the sum of the ranks of the differences above and below zero, respectively. The returned p-value represents the lowest level of significance of a hypothesis that results in rejection. This value allows one to determine whether two methods have significantly different performances. According to the results shown in Table 4.1, the p-values in comparisons of SimRAD with all the 5 methods, reject the null hypotheses for accuracy with a level of significant $\alpha = 0.0005$. Therefore, SimRAD outperforms the 5 comparison methods at all progress levels with high confidence. We study the prediction accuracies of SimRAD with respect to penalty weight λ_t using the OPP-BW dataset. We test 4 different penalty weight functions regarding t : 1) Constant weight $\lambda_t = 1$; 2) Decreasing weight $\lambda_t = 1 - t/T$; 3) Linear weight $\lambda_t = t/T$; 4) Quadratic weight $\lambda_t = (t/T)^2$. According to Figure 4.5, all the 4 functions have not much differences in accuracy when progress level $t/T < 0.4$, while Linear and Quadratic surpass Constant and Decreasing when $t/T \geq 0.4$.

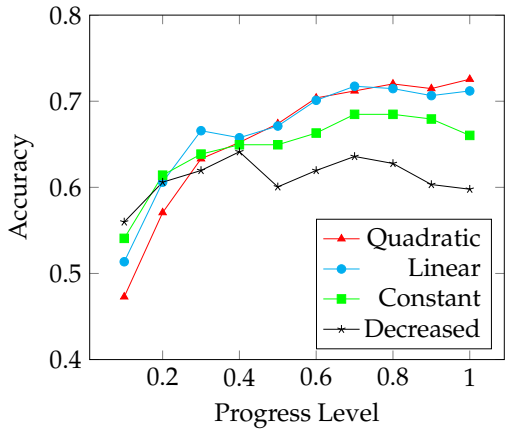


Figure 4.5: The accuracies of SimRAD with respect to λ_t

Dataset	ASM	SR	LDA	Lasso
OPP-BW	**75.06**($\pm$12.40)	35.92($\pm$13.96)	54.52($\pm$12.55)	56.86($\pm$13.70)
OPP-BH	**73.32**($\pm$12.76)	32.08($\pm$16.03)	56.23($\pm$11.51)	59.05($\pm$11.40)
OPP-BUA	**74.08**($\pm$13.40)	39.12($\pm$19.62)	47.12($\pm$18.85)	57.38($\pm$17.24)

Table 4.2: Recognition accuracies on action sequences.

4.3.2 Recognition of Action Sequence

Recall that SimRAD predicts CAs by discovering action sequences from sensor MTS data. We explore the detailed performance of SimRAD by studying the recognition accuracy of the actions. For each time point, we expect to recognize one of 5 LocAs, one of 14 LHAs, and one of 14 RHAs. Therefore, we evaluate the accuracy regarding one action sequence A, where the accuracy is calculated as:

$$\frac{1}{3n}\sum_{t=1}^{n}(1\{\text{LocA correct at } t\}+1\{\text{LHA correct at } t\}+1\{\text{RHA correct at } t\}). \quad (4.14)$$

We report the averaged accuracies on all the testing X. We compare our ASM with 3 methods: 1) Softmax Regression (SR) with regularization parameter $\gamma = 0.1$; 2) Linear Discriminant Analysis (LDA); 3) Lasso with $\gamma = 0.1$. According to the results shown in Table 4.2, the proposed ASM significantly outperforms the 3 comparison methods. As a result, SimRAD accurately finds the actions for predicting CAs.

4.4 Conclusion

In this chapter, we studied the problem of predicting complex activities with ongoing multivariate time series (MTS). We proposed Simultaneous Complex Activities Recognition and Action Sequence Discovering (SimRAD) which predicts a complex activity over time by finding a sequence of multivariate actions from sensor MTS data using a Deep Neural Network. SimRAD learns two probabilistic models for inferring complex activities and action sequences, where the estimations of the two models are conditionally dependent on each other. SimRAD alternately predicts the complex activity and the action sequence with the two models, thus the predictions can be mutually updated until the completion of the complex activity. We evaluated SimRAD on a real-world complex activity dataset of a rich amount of sensor data. The results demonstrate that SimRAD outperforms state-of-the-art methods by average 7.2% in prediction accuracy with very high confidence (p-values < 0.0005).

Chapter 5

Efficient Human Activity Recognition by Reducing Computational Cost

In this chapter, we conduct a study on improving energy-efficiency of Human Activity Recognition (HAR) by reducing computational cost. Recently, ensemble models using multiple feature representations based on time series subsequences have demonstrated excellent performance on recognition accuracy. However, these models can significantly increase the energy overhead and shorten battery life of the mobile devices. We formalize a dynamic subsequence selection problem that minimizes the computational cost while persevering a high recognition accuracy. To solve the problem, we propose Markov Dynamic Subsequence Ensemble (MDSE), an algorithm for the selection of the subsequences via a Markov Decision Process (MDP), where a policy is learned for choosing the best subsequence given the state of prediction. Regarding MDSE, we derive an upper bound of the expected ensemble size, so that the energy consumption caused by the computations of the proposed method is guaranteed. Extensive experiments are conducted on 6 real HAR datasets to evaluate the effectiveness of MDSE. Compared with the state-of-the-art methods, MDSE reduces 70.8% computational cost which is 3.42 times more energy-efficient, and achieves a comparably high accuracy.

5.1 Introduction

Human Activity Recognition (HAR) models of using multiple feature representations from time series subsequences have demonstrated excellent performance on accuracy

This chapter is dervied from: Weihao Cheng, Sarah Erfani, Rui Zhang, Kotagiri Ramamohanarao, "Markov Dynamic Subsequence Ensemble for Energy-Efficient Activity Recognition", *Proceedings of International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, Melbourne, Australia, 2017.

[9, 107]. For example, a Subwindow Ensemble Model (SWEM) [107] uses multiple sized windows to capture the time series for generating diverse feature representations, and infers activity with an ensemble of the predictions based on those features. Although SWEM delivers remarkable accuracy in practice, it is computationally expensive when compared to traditional methods. Both training and inference processes using multiple time series subsequences impose a high computational cost. For ordinary HAR applications, the training cost is insignificant as the model can be computed offline, but the inference cost is crucial as this process is performed on the portable device in real time. Considering the limited battery life of smartphones and the energy overhead of continuously running HAR applets in the background, the above state-of-the-art HAR models [9, 107] are unsuitable for resource constrained mobile platforms.

In this chapter, we aim to design a method that can reduce the computational cost while maintaining a competitive recognition accuracy. A drawback of SWEM is that it indiscriminately uses a fixed set of subsequences to process every incoming time series instance. However, the activities of many instances can be accurately inferred with fewer subsequences, which saves a huge amount of energy. And it is only required to use more subsequences when processes challenging instances, such as a fast walking instance, which can be confused with running. We formalize a problem of dynamically finding a set of subsequences that minimizes the computational cost with a desired accuracy of the ensemble prediction. We present two constraints for the problem which are proved to ensure a certain level of accuracy. To solve the minimization problem with the presented constraints, we propose **Markov Dynamic Subsequence Ensemble** (MDSE), an algorithm capable of dynamically choosing an optimal subsequence set for a given testing time series instance. The selection of the subsequences is modeled as a Markov Decision Process (MDP), where a policy is learned for choosing the best subsequence regarding the state of prediction. Due to the dynamic number of used subsequences (ensemble size), the computational cost can be extensively reduced. We further show the guaranteed computational efficiency of MDSE by deriving an upper bound of the expected ensemble size. Specifically, our main contributions are as follows:

- To improve energy-efficiency for accurate HAR, we formalize a problem that minimizes the computational cost with a desired accuracy of the ensemble prediction,

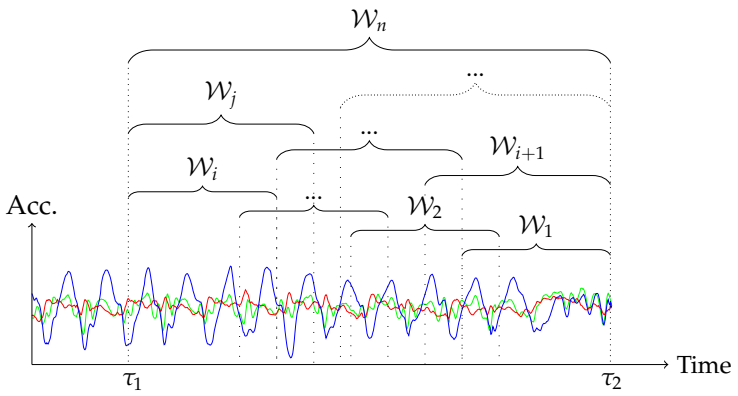


Figure 5.1: The time series shown in red, green and blue curves are 3-axis acceleration data. $\mathcal{W}_1, ..., \mathcal{W}_n$ are n overlapping windows, which observes different ranges of the time series from timestamp τ_1 to τ_2. $\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_i$ share one size with 50% overlaps. Similarly, $\mathcal{W}_{i+1}, ..., \mathcal{W}_j$ share another size, and there are several different sizes of windows.

by finding an optimal set of the subsequences.

- We present two constraints for the optimization problem and prove their effectiveness on ensuring a certain level of generalization accuracy.
- We propose MDSE to solve the constrained minimization problem. The selection of the subsequences is modeled as an MDP, where a policy is learned to choose the best subsequence during the dynamic process under the proposed constraints.
- We derive an upper bound of the expected ensemble size of MDSE, which indicates a guaranteed computational efficiency of MDSE.

We conduct extensive experiments on 6 real human activity datasets. The results of mathematical estimation show that MDSE can reduce average 74.41% of the original SWEM's computations on all datasets, and obtains similarly high accuracies which are only 1-2% less than SWEM. We also use a Google Nexus 5X to evaluate the energy expenditure of MDSE. The results show about 70.8% energy reduction, which demonstrates significant improvement in energy-efficiency by our proposed method.

5.2 Proposed Method

In this section, we propose an energy-efficient Human Activity Recognition (HAR) method with high recognition accuracy. We first formalize a dynamic subsequence selection prob-

lem that minimizes the computational cost with our presented constraints for ensuring the recognition accuracy. We then propose Markov Dynamic Subsequence Ensemble (MDSE) which uses a Markov Decision Process (MDP) to solve the problem. Regarding the presented constraints, we provide theoretical analysis to verify their effectiveness. Finally, we show that the computational efficiency of MDSE is guaranteed by deriving an upper bound of the expected ensemble size of MDSE. We summarize the frequently used symbols in Table 5.1.

Symbol	Explanation
X	Time series data.
y	Activity label of X.
m	Total number of activities.
N	Number of subsequences regarding X.
X_i	The i-th subsequence of X.
H	Subsequence set.
C	Computational cost.
$\boldsymbol{d}$	Voting vector by a classifier.
$\boldsymbol{v}$	Voting proportion vector.
$\hat{v}$	The highest voting proportion.
β, η	Parameters of the proposed constraints.
$\mathcal{S}$	State space of MDP.
s	A state $s \in \mathcal{S}$.
$\mathcal{A}$	Action space of MDP.
a	An action $a \in \mathcal{A}$.
π	Policy of MDP.
$R(a)$	Penalty function regarding action a.
$V(s)$	Value function regarding state s.
Ω	Expected ensemble size of MDSE.

Table 5.1: Major expressions used in this chapter

5.2.1 Markov Dynamic Subsequence Ensemble (MDSE)

Suppose there are m activities which are labeled from 1 to m. The task of HAR is to recognize the current activity y based on a segment of time series data X collected from one or more sensors (e.g. accelerometer). Suppose (X, y) is drawn from a distribution $\mathcal{D}$, i.e. $(X, y) \sim \mathcal{D}$, where the length of X is L. We define a set of subsequences: $H_X = \{X_1, X_2, ..., X_N\}$, where $X_i \equiv X_{\tau_{i,1}:\tau_{i,2}}$ is a subsequence of X, ranging from timestamp $\tau_{i,1}$ to $\tau_{i,2}$, and $1 \leq \tau_{i,1} < \tau_{i,2} \leq L$. For each subsequence X_i, we can generate a feature representation $\phi(X_i)$, and then a classifier based on $\phi(X_i)$ can output a voting vector $\boldsymbol{d}(X_i) \in \{0,1\}^m$:

$$\boldsymbol{d}(X_i) = (d_1, d_2, ..., d_m), \tag{5.1}$$

where $d_j = 1$ $(1 \leq j \leq m)$ implies that the activity j receives the vote from the classifier, and we have $d_1 + d_2 + ... + d_m = 1$. We denote the computational cost of utilizing X_i as $C(X_i)$. Given a subset $H \subseteq H_X$, the ensemble voting vector of using all the subsequences in H is defined as:

$$\boldsymbol{d}_H = \sum_{X_i \in H} \boldsymbol{d}(X_i), \tag{5.2}$$

and the ensemble voting proportion vector is defined as:

$$\boldsymbol{v}_H = \frac{1}{|H|} \boldsymbol{d}_H. \tag{5.3}$$

Thereby, we calculate the inferred activity as follows:

$$\hat{y}_H = \text{argmax}\{\boldsymbol{v}_H\}, \tag{5.4}$$

where $\hat{y}_H$ is the activity with the maximum number of votes. The total computational cost of using H is defined as:

$$C_H = \sum_{X_i \in H} C(X_i). \tag{5.5}$$

To improve computational efficiency for accurate HAR, we define a problem that minimizes the computational cost while maintaining the accuracy at a certain level. Let ρ be a selection function which can dynamically choose a subset $H_X^\rho = \rho(H_X)$ of H_X based on a latent policy. Then, our problem can be formalized as minimizing the expected compu-

tational cost with a constraint of generalization accuracy:

$$\min_{\rho} \mathrm{E}_{(X,y)\sim\mathcal{D}} C_{H_X^\rho}, \tag{5.6}$$

$$s.t.\ P_{(X,y)\sim\mathcal{D}}(\hat{y}_{H_X^\rho} = y) \geq \alpha, \tag{5.7}$$

where $\alpha \in (0.5, 1)$ is the parameter bounding the generalization accuracy $P_{(X,y)\sim\mathcal{D}}(\hat{y}_{H_X^\rho} = y)$, i.e., the probability of $\hat{y}_{H_X^\rho} = y$. Since numerically ensuring a generalization accuracy is non-trivial, we present two alternative constraints to replace (5.7), and formalize an alternative problem as follows:

$$\min_{\rho} \mathrm{E}_{(X,y)\sim\mathcal{D}} C_{H_X^\rho}, \tag{5.8}$$

$$s.t.\ \max\{\boldsymbol{v}_{H_X^\rho}\} \geq \beta, \tag{5.9}$$

$$|H_X^\rho| \geq \eta, \tag{5.10}$$

$$\text{for } (X, y) \in \mathcal{D}$$

where $\beta \in (0.5, 1)$ and $\eta \in (0, N]$ are predefined parameters. Later in section 5.2.2, we provide theoretical analysis to show that the constraints (5.9) and (5.10) can ensure a certain level of generalization accuracy regarding β and η.

To solve the above problem, we propose Markov Dynamic Subsequence Ensemble (MDSE) where the obtaining of H_X^ρ is modeled as a Markov Decision Process (MDP):

- $\mathcal{S}$ is a set of states. A state $s \in \mathcal{S}$ is defined as a tuple $s = (H,\ \max\{\boldsymbol{d}_H\})$.

- $\mathcal{A}$ is a set of actions, where $\mathcal{A} = \{1, 2, ..., N, N+1\}$. An action $a \in \mathcal{A}$ implies: (i) Using the subsequence X_a to make a prediction, if $1 \leq a \leq N$. (ii) Stopping the process, if $a = N + 1$.

- $R(a)$: $\mathcal{A} \mapsto \mathbb{R}$ is the penalty function. It represents a valuable penalty by taking an action a. We define the penalty function as:

$$R(a) = \begin{cases} C(X_a) & 1 \leq a \leq N \\ 0 & a = N+1 \end{cases}. \tag{5.11}$$

We design a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$, which returns the best action for each state $s \in \mathcal{S}$:

$$\pi(s) = \begin{cases} N+1 & \max\{\boldsymbol{v}_H\} \geq \beta \wedge |H| \geq \eta \\ M(s) & \text{otherwise} \end{cases}, \tag{5.12}$$

where $M(\boldsymbol{s})$ is a hash mapping from $\mathcal{S}$ to $\mathcal{A}$. Given a state $s = (H, \max\{\boldsymbol{d}_H\})$, we can obtain an action $a = \pi(s)$ for s, then the next state $s' = s'_a$ by taking the action a is obtained as:

$$s'_a = \begin{cases} (H \bigcap \{X_a\}, \max\{\boldsymbol{d}_{H \bigcap \{X_a\}}\}) & 1 \leq a \leq N \\ s & a = N+1 \text{ (stop action)} \end{cases}. \tag{5.13}$$

When the action $a = N+1$ is obtained from π, we stop the MDP and return the final state as the output of the MDP. The subset H of the final state is considered as H^ρ_X. Given the policy π, an initial state s_0, and an X, the MDP will visit a series of states $s_1, ..., s_K$ by taking actions $a_0, a_1, ..., a_{K-1}$, respectively. The total penalty of this process can be measured by a value function $V^\pi(s) : \mathcal{S} \mapsto \mathbb{R}$ as:

$$V^\pi(s) = R(a_0) + R(a_1) + ... + R(a_{K-1}) \mid s = s_0, \tag{5.14}$$

which can also be written as Bellman's equation:

$$V^\pi(s) = \begin{cases} R(\pi(s)) + V^\pi(s'_{\pi(s)}) & 1 \leq \pi(s) \leq N \\ R(\pi(s)) & \pi(s) = N+1 \end{cases}. \tag{5.15}$$

We can convert our problem into minimizing the value function regarding the policy π:

$$\min_\rho \mathrm{E}_{(X,y)\sim\mathcal{D}} C_{H^\rho_X},\ s.t.\ \max\{\boldsymbol{v}_{H^\rho_X}\} \geq \beta,\ |H^\rho_X| \geq \eta \tag{5.16}$$

$$= \min_\pi \mathrm{E}_{(X,y)\sim\mathcal{D}}[R(a_0) + R(a_1) + R(a_2) + ...] \tag{5.17}$$

$$= \min_\pi \mathrm{E}_{(X,y)\sim\mathcal{D}}[V^\pi(s_0) \mid s_0 = (\emptyset, 0)]. \tag{5.18}$$

Let $V^*_X(s)$ be the minimal value function for an X. Since $\mathcal{S}$ is a finite state space, $V^*_X(s)$

Algorithm 6 Markov Dynamic Subsequence Ensemble (MDSE)

Input: A time series X
Output: Predicted activity $\hat{y}$
$H \leftarrow \emptyset$, $\boldsymbol{d}_H \leftarrow \mathbf{0}$
while $|H| < N$ **do**
  $s \leftarrow (H, \max\{\boldsymbol{d}_H\})$
  $a \leftarrow \pi(s)$
  if $a = N + 1$ **then**
    break
  end if
  $H \leftarrow H \bigcap \{X_a\}$
  $\boldsymbol{d}_H \leftarrow \boldsymbol{d}_H + \boldsymbol{d}_{X_a}$
end while
return $\hat{y} = \text{argmax}\{\boldsymbol{d}_H\}$

can be represented as follows:

$$V_X^*(s) = \begin{cases} 0 & \max\{\boldsymbol{v}_H\} \geq \beta \wedge |H| \geq \eta \\ \min\limits_{a \in \mathcal{A}, X_a \notin H} \{R(a) + V_X^*(s'_a)\} & \text{otherwise} \end{cases}, \tag{5.19}$$

where the restriction $X_a \notin H$ is to avoid the re-selection of a used subsequence which brings no diversity effect to the ensemble performance. Then, we initialize $V_X^*(s) = +\infty$ for each $s \in \mathcal{S}$, and use the value iteration algorithm [72] to repeatedly calculate $V_X^*(s)$ using (5.19) until convergence. Finally, the optimal policy $\pi^*(s)$ on the distribution $\mathcal{D}$ is learned as follows:

$$\pi^*(s) = \begin{cases} N+1 & \max\{\boldsymbol{v}_H\} \geq \beta \wedge |H| \geq \eta \\ \underset{a \in \mathcal{A}, X_a \notin H}{\text{argmin}} \{R(a) + \mathrm{E}_{(X,y)\sim\mathcal{D}} V_X^*(s'_a)\} & \text{otherwise} \end{cases}. \tag{5.20}$$

After obtaining the policy $\pi = \pi^*$, MDSE can infer the activity of X by incorporating the MDP. The pseudocode of MDSE is presented in Algorithm 6.

5.2.2 Theoretical Analysis of the Accuracy Constraints in MDSE

We provide theoretical analysis to explain that the constraints Eq. 5.9 and Eq. 5.10 collaboratively ensure the solution of MDSE having a certain level of generalization accuracy regarding the parameters β and η. For the convenience of discussion, we use simplified

notations in this section. We denote $H = H_X^\rho$, $\hat{y} = \hat{y}_H$, and let $\hat{v} \in [0,1]$ be the highest voting proportion, which is defined as:

$$\hat{v} = \max\{v_H\}. \tag{5.21}$$

Thereby, the two constraints are expressed as: $\hat{v} \geq \beta$ and $|H| \geq \eta$.

Analysis of the 1st Constraint

We theoretically explain the effectiveness of Eq. 5.9 by showing that: given $\hat{v} \geq \beta$ and a fixed $|H|$, the generalized recognition accuracy increases monotonically with respect to $\beta \in (0.5, 1)$. Let $n = |H|$, then with the ensemble of n subsequences, the generalization accuracy of the ensemble prediction given $\hat{v} \geq \beta$ is written as the probability $P(\hat{y} = y \,|\, \hat{v} \geq \beta)$, which can be calculated based on the Bayes' rule:

$$\begin{aligned} P(\hat{y} = y \,|\, \hat{v} \geq \beta) &= \frac{P(\hat{v} \geq \beta\,,\, \hat{y} = y)}{\sum_{j=1}^{m} P(\hat{v} \geq \beta\,,\, \hat{y} = j)} \\ &= \frac{P(\hat{v} \geq \beta\,,\, \hat{y} = y)}{P(\hat{v} \geq \beta\,,\, \hat{y} = y) + \sum_{j \neq y} P(\hat{v} \geq \beta\,,\, \hat{y} = j)}. \end{aligned} \tag{5.22}$$

Suppose the predictions of the n subsequences are independent with the identical generalization error rate ϵ. The chance of seeing exact k incorrect votes among the n classifiers is $\binom{n}{k}\epsilon^k(1-\epsilon)^{n-k}$. Let $\hat{n}$ be the number of votes that the inferred activity $\hat{y}$ received, i.e., $\hat{n} = n\hat{v}$. If $\hat{y} = y$, then those $\hat{n}$ votes are considered to be correct, and the other $n - \hat{n}$ votes are considered to be incorrect. When $\hat{v} \geq \beta$, then $\hat{n} \geq \lceil n\beta \rceil$ and correspondingly $n - \hat{n} \leq n - \lceil n\beta \rceil$. Since $\beta > 0.5$, $\hat{y}$ is the only activity receives votes more than $n/2$, thus the probability $P(\hat{v} \geq \beta\,,\, \hat{y} = y)$ can be expressed as:

$$P(\hat{v} \geq \beta\,,\, \hat{y} = y) = \sum_{k=0}^{n - \lceil n\beta \rceil} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}. \tag{5.23}$$

If $\hat{y} = j$ for $j \neq y$, then those $\hat{n}$ votes are considered to be incorrect, and the correctnesses of the other $n - \hat{n}$ votes are not able to judge. For simplicity, we assume the error ϵ is equally shared on the other $m - 1$ activities, then the chance of a classifier voting for

activity j ($j \neq y$) is:

$$\tilde{\epsilon} = \frac{\epsilon}{m-1}, \tag{5.24}$$

and the chance of a classifier voting for any activity other than j is $1-\tilde{\epsilon}$. Similar to Eq. 5.23, the probability $P(\hat{v} \geq \beta\,, \hat{y} = j)$ for $j \neq y$ can be expressed as:

$$P(\hat{v} \geq \beta\,, \hat{y} = j) = \sum_{k=0}^{n-\lceil n\beta \rceil} \binom{n}{k} (1-\tilde{\epsilon})^k \tilde{\epsilon}^{n-k}. \tag{5.25}$$

We define $f(\beta) \equiv P(\hat{y} = y \,|\, \hat{v} \geq \beta)$ as a function of β, which can be written as:

$$f(\beta) = \frac{\sum_{k=0}^{n-\lceil n\beta \rceil} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}}{\sum_{k=0}^{n-\lceil n\beta \rceil} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k} + (m-1) \sum_{k=0}^{n-\lceil n\beta \rceil} \binom{n}{k} (1-\tilde{\epsilon})^k \tilde{\epsilon}^{n-k}}. \tag{5.26}$$

We assume $\epsilon < 0.5$ that the classifier outperforms random choice. The following theorem shows that $f(\beta)$ is a monotonically increasing function.

Theorem 5.1. *Given $\epsilon \in (0, 0.5)$, $f(\beta)$ is a monotonically increasing function.*

Proof. Let $t = n - \lceil n\beta \rceil$, and $0 \leq t \leq n$. We construct a new function $g(t)$ on $t \in \mathbb{Z}^+$ as:

$$g(t) = \frac{\sum_{k=0}^{t} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}}{\sum_{k=0}^{t} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k} + (m-1) \sum_{k=0}^{t} \binom{n}{k} (1-\tilde{\epsilon})^k \tilde{\epsilon}^{n-k}}. \tag{5.27}$$

Let $\gamma = \lceil n\beta \rceil - n\beta$. Since $t = n - \lceil n\beta \rceil \Rightarrow \beta = (n-t-\gamma)/n$, then $g(t) = f(\frac{n-t-\gamma}{n})$, and we have $f(\frac{n-t-\gamma'}{n}) = f(\frac{n-t}{n})$ for any $\gamma' \in [0, 1)$. Consequently, if $g(t)$ is a monotonically decreasing function, then $f(\beta)$ is a monotonically increasing function. We rewrite $g(t)$ as:

$$g(t) = \frac{1}{1 + (m-1)u(t)}, \tag{5.28}$$

where $u(t)$ is:

$$u(t) = \frac{\sum_{k=0}^{t} \binom{n}{k} (1-\tilde{\epsilon})^k \tilde{\epsilon}^{n-k}}{\sum_{k=0}^{t} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}}. \tag{5.29}$$

Since $m - 1 > 0$, if $u(t)$ is proved to be a monotonically increasing function, then $g(t)$ is a monotonically decreasing function. Let $\theta_1(k) = \binom{n}{k}(1-\tilde{\epsilon})^k \tilde{\epsilon}^{n-k}$, and $\theta_2(k) = \binom{n}{k}\epsilon^k (1-$

$\epsilon)^{n-k}$, then:

$$\frac{\theta_1(k)}{\theta_2(k)} = (\frac{\tilde{\epsilon}}{1-\epsilon})^n(\frac{1-\tilde{\epsilon}}{\tilde{\epsilon}})^k(\frac{1-\epsilon}{\epsilon})^k. \tag{5.30}$$

Since $0 < \tilde{\epsilon} \leq \epsilon < 0.5$, then $(1-\tilde{\epsilon})/\tilde{\epsilon} > 1$ and $(1-\epsilon)/\epsilon > 1$, thus we have:

$$\frac{\theta_1(k)}{\theta_2(k)} > \frac{\theta_1(k-1)}{\theta_2(k-1)}. \tag{5.31}$$

As a result, we can infer that:

$$u(t+1) - u(t) = \frac{\sum_{k=0}^{t}\theta_1(k) + \theta_1(t+1)}{\sum_{k=0}^{t}\theta_2(k) + \theta_2(t+1)} - \frac{\sum_{k=0}^{t}\theta_1(k)}{\sum_{k=0}^{t}\theta_2(k)} > 0 \tag{5.32}$$

According to the above derivation, we claim that $u(t)$ is a monotonically increasing function. Consequently, $f(\beta)$ is a monotonically increasing function. □

We prove the monotonic property of $P(\hat{y} = y \,|\, \hat{v} \geq \beta)$ on $\beta \in (0.5, 1)$. Therefore, we can increase $\beta \in (0.5, 1)$ to improve the recognition accuracy of MDSE.

Analysis of the 2nd Constraint

We theoretically explain the effectiveness of Eq. 5.10 by showing that: given $|H| \geq \eta$ and a fixed $\hat{v} \in (0.5, 1)$, the generalized recognition accuracy increases monotonically with respect to $\eta \in (0, N]$. Let $n = |H|$, the generalization accuracy of the ensemble prediction given $n \geq \eta$ is written as the probability $P(\hat{y} = y \,|\, n \geq \eta)$, which can be calculated based on the Bayes rule:

$$P(\hat{y} = y \,|\, n \geq \eta) = \frac{P(n \geq \eta\,,\, \hat{y} = y)}{P(n \geq \eta\,,\, \hat{y} = y) + \sum_{j \neq y} P(n \geq \eta\,,\, \hat{y} = j)}. \tag{5.33}$$

Similar to the analysis of the 1st constraint, we define $f(\eta) \equiv P(\hat{y} = y \,|\, n \geq \eta)$ as a function of η, which can be written as:

$$f(\eta) = \frac{\sum_{n=\eta}^{N}\binom{n}{\lceil n\hat{v}\rceil}\epsilon^{n-n\hat{v}}(1-\epsilon)^{n\hat{v}}}{\sum_{n=\eta}^{N}\binom{n}{\lceil n\hat{v}\rceil}\epsilon^{n-n\hat{v}}(1-\epsilon)^{n\hat{v}} + (m-1)\sum_{n=\eta}^{N}\binom{n}{\lceil n\hat{v}\rceil}(1-\tilde{\epsilon})^{n-n\hat{v}}\tilde{\epsilon}^{n\hat{v}}}. \tag{5.34}$$

Theorem 5.2. *Given $\epsilon \in (0, 0.5)$, $f(\eta)$ is a monotonically increasing function.*

Proof. Similar to the proof of Theorem 5.1, we rewrite $f(\eta)$ as:

$$f(\eta) = \frac{1}{1 + (m-1)u(\eta)}, \tag{5.35}$$

where $u(\eta)$ is:

$$u(\eta) = \frac{\sum_{n=\eta}^{N} \binom{n}{\lceil n\hat{v} \rceil} (1-\tilde{\epsilon})^{n-n\hat{v}} \tilde{\epsilon}^{n\hat{v}}}{\sum_{n=\eta}^{N} \binom{n}{\lceil n\hat{v} \rceil} \epsilon^{n-n\hat{v}} (1-\epsilon)^{n\hat{v}}} \tag{5.36}$$

If we can prove $u(\eta)$ is a monotonically decreasing function, then $f(\eta)$ is a monotonically increasing function. Let $\theta_1(n) = \binom{n}{\lceil n\hat{v} \rceil}(1-\tilde{\epsilon})^{n-n\hat{v}}\tilde{\epsilon}^{n\hat{v}}$, and $\theta_2(n) = \binom{n}{\lceil n\hat{v} \rceil}\epsilon^{n-n\hat{v}}(1-\epsilon)^{n\hat{v}}$, then:

$$\frac{\theta_1(n)}{\theta_2(n)} = \left(\frac{1-\tilde{\epsilon}}{\epsilon}\right)^{n-n\hat{v}} \left(\frac{\tilde{\epsilon}}{1-\epsilon}\right)^{n\hat{v}} \tag{5.37}$$

$$= \left[\left(\frac{1-\tilde{\epsilon}}{\epsilon}\right)^{1-2\hat{v}} \left(\frac{\tilde{\epsilon}(1-\tilde{\epsilon})}{\epsilon(1-\epsilon)}\right)^{\hat{v}}\right]^{n} \tag{5.38}$$

Since $0 < \tilde{\epsilon} \leq \epsilon < 0.5$ and $\hat{v} > 0.5$, then we have $[(1-\tilde{\epsilon})/\epsilon]^{1-2\hat{v}} < 1$, and $[(\tilde{\epsilon}(1-\tilde{\epsilon}))/(\epsilon(1-\epsilon))]^{\hat{v}} < 1$. Therefore:

$$\frac{\theta_1(n)}{\theta_2(n)} < \frac{\theta_1(n-1)}{\theta_2(n-1)}. \tag{5.39}$$

As a result, we can infer that:

$$u(\eta+1) - u(\eta) = \frac{\sum_{n=\eta+1}^{N} \theta_1(n)}{\sum_{n=\eta+1}^{N} \theta_2(n)} - \frac{\sum_{n=\eta+1}^{N} \theta_1(n) + \theta_1(\eta)}{\sum_{n=\eta+1}^{N} \theta_2(n) + \theta_2(\eta)} < 0. \tag{5.40}$$

According to the above derivation, we claim that $u(\eta)$ is a monotonically decreasing function. Consequently, $f(\eta)$ is a monotonically increasing function. □

We prove the monotonic property of $P(\hat{y} = y \,|\, n \geq \eta)$ on $\eta \in (0, N]$. Therefore, we can increase $\eta \in (0, N]$ to improve the recognition accuracy of MDSE. We generally set $1 < \eta < N$ for obtaining a reasonable performance of MDSE. If we set $\eta = 1$, MDSE will constantly make prediction based on only one subsequence. This is because that, when the first prediction is obtained, the algorithm finds $|H| = 1 \geq \eta$ as well as $\hat{v} = 1 \geq \beta$, and then stops the dynamic process immediately without taking other subsequences into consideration. If we set $\eta = N$, MDSE will constantly take all the subsequences into

ensemble that degenerate into SWEM, since the constraint $|H| \geq \eta$ can only be fulfilled when $|H| = N$.

5.2.3 Computational Efficiency of MDSE

A significant advantage of MDSE is that its computational efficiency is guaranteed, toward which we provide theoretical analysis in this section. Let l be an integer, and P_l be the probability that the ensemble size $|H|$ equals l when MDSE returns. The expected ensemble size Ω is calculated as:

$$\Omega = \sum_{l=\eta}^{N} l P_l. \tag{5.41}$$

Let $\hat{v}^{(l)}$ be the highest voting proportion by the first l chosen subsequences, then P_l can be expressed as:

$$P_l = \begin{cases} P(\hat{v}^{(\eta)} \geq \beta), & \text{if } l = \eta \\ P(\hat{v}^{(\eta)} < \beta, ..., \hat{v}^{(l-1)} < \beta, \hat{v}^{(l)} \geq \beta), & \text{if } \eta < l < N \\ P(\hat{v}^{(\eta)} < \beta, ..., \hat{v}^{(N-2)} < \beta, \hat{v}^{(N-1)} < \beta), & \text{if } l = N \end{cases} \tag{5.42}$$

Due to the non-triviality of P_l, the expected ensemble size Ω is hard to estimate. However, it is obvious that $\sum_{l=\eta}^{N} P_l = 1$, then we can derive an upper bound for Ω:

$$\begin{aligned} \Omega = \eta P_\eta + \sum_{l=\eta+1}^{N} l P_l \ &\leq \ \eta P_\eta + N \sum_{l=\eta+1}^{N} P_l \\ &= \eta P_\eta + N(1 - P_\eta) = \eta + (N - \eta)(1 - P_\eta) \\ &= \eta + (N - \eta) P(\hat{v}^{(\eta)} < \beta). \end{aligned} \tag{5.43}$$

We present the probability $P(\hat{v}^{(\eta)} < \beta)$ using the representation of multinomial cumulative distribution function proposed in [56]. Let $\boldsymbol{v}^{(\eta)} = \{v_1^{(\eta)}, v_2^{(\eta)}, ..., v_m^{(\eta)}\}$ be the voting proportion vector, and $\hat{v}^{(\eta)} = \max\{v_1^{(\eta)}, v_2^{(\eta)}, ..., v_m^{(\eta)}\}$. Let $\eta_j = \eta v_j^{(\eta)}$ be the votes of the activity j. Then, $P(\hat{v}^{(\eta)} < \beta)$ can be written as:

$$\begin{aligned} P(\hat{v}^{(\eta)} < \beta) &= P(v_1^{(\eta)} < \beta, v_2^{(\eta)} < \beta, ..., v_m^{(\eta)} < \beta) \\ &= P(\eta_1 \leq \lceil \eta\beta \rceil - 1, \eta_2 \leq \lceil \eta\beta \rceil - 1, ..., \eta_m \leq \lceil \eta\beta \rceil - 1). \end{aligned} \tag{5.44}$$

Suppose each of the classifiers votes to activities $1, 2, ..., m$ with probabilities $p_1, p_2, ..., p_m$, respectively. The multinomial cumulative distribution function can be presented as follows:

$$\begin{aligned} &P(\eta_1 \leq \lceil \eta\beta \rceil - 1, \eta_2 \leq \lceil \eta\beta \rceil - 1, ..., \eta_m \leq \lceil \eta\beta \rceil - 1) \\ &= \frac{\eta!}{\eta^\eta e^{-\eta}} \{\prod_{j=1}^{m} P(W_j \leq \lceil \eta\beta \rceil - 1)\} P(\sum_{j=1}^{m} Z_j = \eta), \end{aligned} \tag{5.45}$$

where $W_j \sim \mathbf{P}(\eta p_j)$ = Poisson distribution, and $Z_j \sim T\mathbf{P}(\eta p_j)$ = truncated Poisson distribution with range $0, 1, 2, ..., \lceil \eta\beta \rceil - 1$. Without loss of generality, we set p_1 as $1 - \epsilon$ and set $p_2, ..., p_m$ as $\tilde{\epsilon}$ to obtain those distributions. Since the probability $P(\sum_{j=1}^{m} Z_j = \eta)$ is hard to calculate, we approximate it using normal distribution:

$$P(\sum_{j=1}^{m} Z_j = \eta) = \frac{1}{\sqrt{2\pi \sum_{j=1}^{m} \sigma_j^2}} \exp\{-\frac{1}{2}(\frac{\eta - \sum_{j=1}^{m} \mu_j}{\sqrt{\sum_{j=1}^{m} \sigma_j^2}})^2\}, \tag{5.46}$$

where $\mu_j = \mathrm{E}(Z_j)$ and $\sigma_j^2 = \mathrm{Var}(Z_j)$. These two moments of Z_i can be calculated as:

$$\mu_j = \eta p_j (1 - \frac{P(W_j = \lceil \eta\beta \rceil - 1)}{P(W_j \leq \lceil \eta\beta \rceil - 1)}), \tag{5.47}$$

$$\sigma_j^2 = \mu_j - (\lceil \eta\beta \rceil - 1 - \mu_j)(\eta p_j - \mu_j). \tag{5.48}$$

A more precise estimation for $P(\sum_{j=1}^{m} Z_j = \eta)$ can be obtained by Edgeworth approximation [56], which additionally incorporates 2nd to 4th central moments. We omit the details here due to the page limitation. In summary, the expected ensemble size Ω of MDSE is bounded by:

$$\Omega \leq \eta + (N - \eta) \frac{\eta!}{\eta^\eta e^{-\eta}} \{\prod_{j=1}^{m} P(W_j \leq \lceil \eta\beta \rceil - 1)\} P(\sum_{j=1}^{m} Z_j = \eta). \tag{5.49}$$

We intuitively show the upper bound of Ω in Figure 5.2, where the curves of the upper bounds are plotted with various settings of β and η. Let Ω_{ub} be the bound which equals to the right part of Eq. 5.49. We can infer that the expected computational cost of MDSE is bounded by $\sum_{i=1}^{\lceil \Omega_{ub} \rceil} C(X_i)$ assuming $C(X_1) \geq C(X_2) \geq ... \geq C(X_N)$.

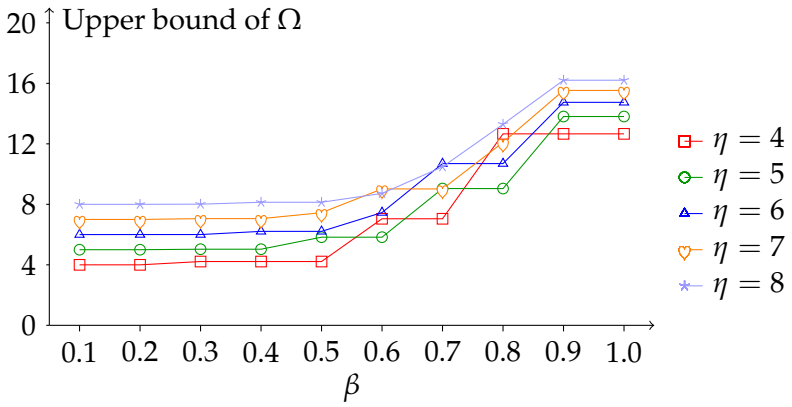


Figure 5.2: Given $N = 20$ (subsequences), $m = 6$ (activities) and $\epsilon = 0.2$, the curves show the upper bound of Ω as function of β with different η.

5.3 Empirical Evaluations

In this section, we evaluate the performance of Markov Dynamic Subsequence Ensemble (MDSE) with respect to accuracy and computational cost, and compare it with 4 baseline methods. The experiments are conducted on both desktop and mobile platforms. For the experiments on the desktop platform, we implement the methods with Python 2.7, and test the methods on a DELL PC with Intel (R) Core (TM) i7-4470 CPU 3.40 GHz, 16G RAM and 64-bit Ubuntu 14.04 LTS operating system. For the experiments on the mobile platform, we implement the methods with Java, and test the methods on a Google Nexus 5X with Android 6.0 system.

Datasets: The 3-axis acceleration data of 6 human activity datasets is used to evaluate the performance of MDSE. (1) Human Activity Sensing Consortium dataset (HASC) [43]: We use the data of all 6 activities, which is collected by iPhone and iPod Touch. (2) Human Activity Recognition on Smartphones Dataset (HARSD) [5]: We use the data of all 6 activities, which is collected by a Samsung Galaxy S II. (3) Actitracker dataset (ACTR) [53]: We use the data of all 6 activities, which is collected by Android phones. (4) Daily Sport Activities dataset (DSA) [11]: We use the data of all 19 activities, which is collected by an accelerometer placed on torso. (5) Opportunity dataset (OPP) [81]: We use the data of selected 5 arm activities ('close', 'reach', 'open', 'release', 'move'), which is collected by an accelerometer placed on left arm. (6) CHEST dataset [20]: We use the data of selected 4 activities ('working at computer', 'standing', 'working', 'up/down stairs'), which is col-

lected by an accelerometer placed on chest.

Experiment Settings: Let $\mathcal{D}$ be a dataset containing a number of time series data, where each one is collected independently. Let $\mathcal{D}_y$ be the subset corresponding to activity y, thus we have $\mathcal{D} = \bigcup_y \mathcal{D}_y$. For each subset $\mathcal{D}_y$, we randomly split it into three groups: $\mathcal{D}_{y,1}$, $\mathcal{D}_{y,2}$ and $\mathcal{D}_{y,3}$, where $|\mathcal{D}_{y,1}| = 0.5|\mathcal{D}_y|$, $|\mathcal{D}_{y,2}| = 0.25|\mathcal{D}_y|$ and $|\mathcal{D}_{y,3}| = 0.25|\mathcal{D}_y|$. We then generate $\mathcal{D}_1 = \bigcup_y \mathcal{D}_{y,1}$, $\mathcal{D}_2 = \bigcup_y \mathcal{D}_{y,2}$, $\mathcal{D}_3 = \bigcup_y \mathcal{D}_{y,3}$, which are aggregated crossing the activities y. For the time series in each aggregated group, we sequentially extract a number of 5 seconds segments as our instances, i.e, X. For $\mathcal{D}_1$ and $\mathcal{D}_3$, we perform the extraction for every 1.5 seconds. For $\mathcal{D}_2$, we perform the extraction for every 0.5 seconds. We use the instances of $\mathcal{D}_1$ to train the classifiers of MDSE and baselines. We use the $\mathcal{D}_2$ to learn the Markov Decision Process (MDP) for MDSE. We use the instances of $\mathcal{D}_3$ for testing.

Settings of MDSE: Let τ be the number of readings in one second, and $X_{k_1\tau:k_2\tau}$ be the subsequence of X containing readings from the k_1-th second to k_2-th second. We generate the subsequence set H_X as $\{X_{0:2\tau}, X_{1\tau:3\tau}, X_{2\tau:4\tau}, X_{3\tau:5\tau}, X_{0:2.5\tau}, X_{1.25\tau:3.75\tau}, X_{2.5\tau:5\tau}, X_{0:3\tau}, X_{\tau:4\tau}, X_{2\tau:5\tau}, X_{0:3.5\tau}, X_{1.5\tau:5\tau}, X_{0:4\tau}, X_{\tau:5\tau}, X_{0:4.5\tau}, X_{0.5\tau:5\tau}, X_{0:5\tau}\}$, where $|H_X| = 17$. For each $X_i \in H_X$, we perform Fast Fourier Transform (FFT) on each axis channel of the time series, and combine the Fourier coefficients of the 3 channels as the feature $\phi(X_i)$ [71]. The cost $C(X_i)$ of using X_i is set as $C(X_i) = L_i \log L_i$ where L_i is the length of X_i. Since the subsequences can be grouped by 6 different lengths: 2τ, 2.5τ, 3τ, 3.5τ, 4τ, 4.5τ, and 5τ, we build base classifiers one for each length. The choosing of classification model for the base classifiers is varied, where the details are described in the latter section.

Settings of Baselines: We test 4 baseline methods in comparison with MDSE: (i) Single Base Classifier (SBC): We classify X using a single base classifier. (ii) Subwindow Ensemble Model (SWEM): We classify X by combing all the base classifiers which are trained based on the subsequences of H_X in the same manner of MDSE. (iii) Convolutional Neural Network (CNN): We classify X by a CNN, whose structure is referenced from [104]. The CNN consists of a convolutional layer of 20 feature mappings with 1s length filter striding 0.5s, a max-pooling layer with 0.5s length filter, a hidden layer with 1024 output units, a hidden layer with 30 output units, and a softmax layer. We set learning rate to 0.001 and batch size to 10 for training the CNN. (iv) Recurrent Neural Network (RNN):

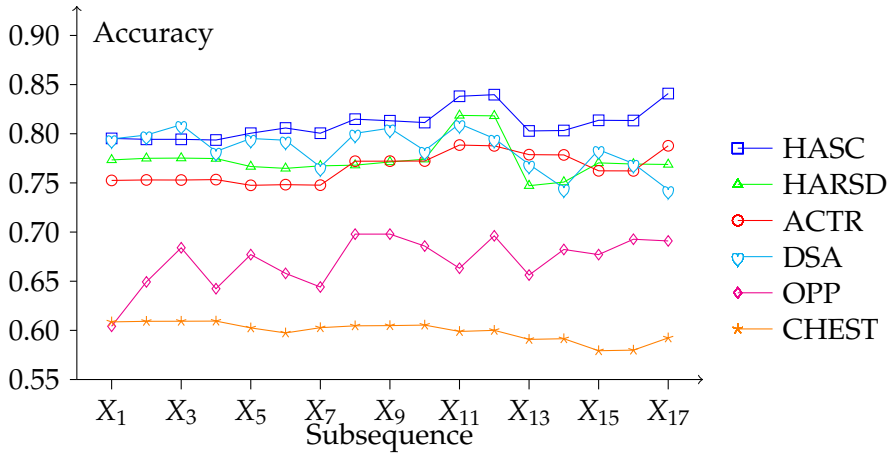
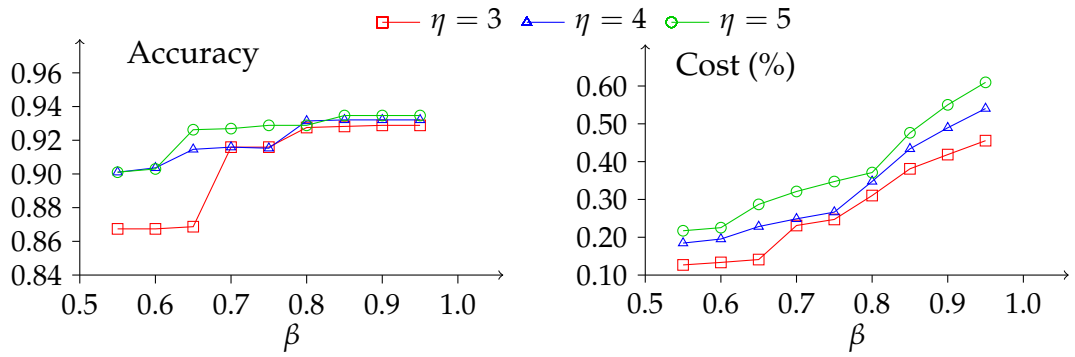


Figure 5.3: Accuracies of Decision Trees (DTs) using the subsequences X_1, X_2, ..., X_{17} on different datasets.

We classify X by a RNN which consists of a LSTM layer and a Softmax layer. We divide X into a sequence of 0.5 seconds chunks, which are used as input of the RNN. We set learning rate to 0.001 and batch size to 10 for training the RNN. We examine Decision Tree (DT), Logistic Regression (LR) and K-Nearest Neighbor (KNN) as the base classifier for SBC, SWEM and MDSE. The settings of DT, LR and KNN are: (i) DT: We use Classification and Regression Tree (CART) as the Decision Tree algorithm. (ii) LR: We set the regularization weight parameter to 1. (iii) KNN: We set the neighbor number to 1.

5.3.1 Performances of Different Subsequences

We conduct experiments to investigate the accuracy regarding each single subsequence $X_i \in H_X$ using DT. Figure 5.3 shows the accuracies of using the subsequences from X_1 to X_{17} on the 6 datasets. As shown in the figure, all the subsequences have different accuracies. A smaller subsequence usually result in low accuracy, since only a few data points are captured which many not be a good representative of the activity. A larger subsequences capture more data points which generally deliver a higher accuracy. However, this is not exactly correct, since a subsequence with more data points may not represents the activity properly. Therefore, we perform HAR using a number of feature representations based on multiple time series subsequences.



(a) Accuracy (b) Percentage of Cost

Figure 5.4: Accuracy and Cost (%) of Markov Dynamic Subsequence Ensemble (MDSE) as functions of β with different η.

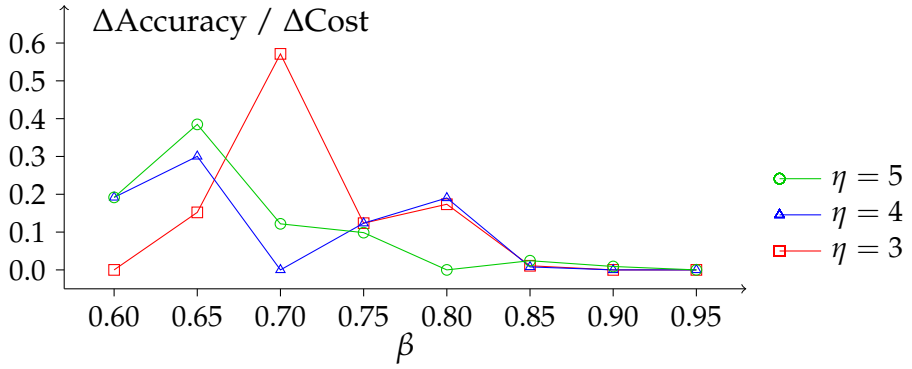


Figure 5.5: ΔAccuracy/ΔCost of MDSE as a function of β with different η.

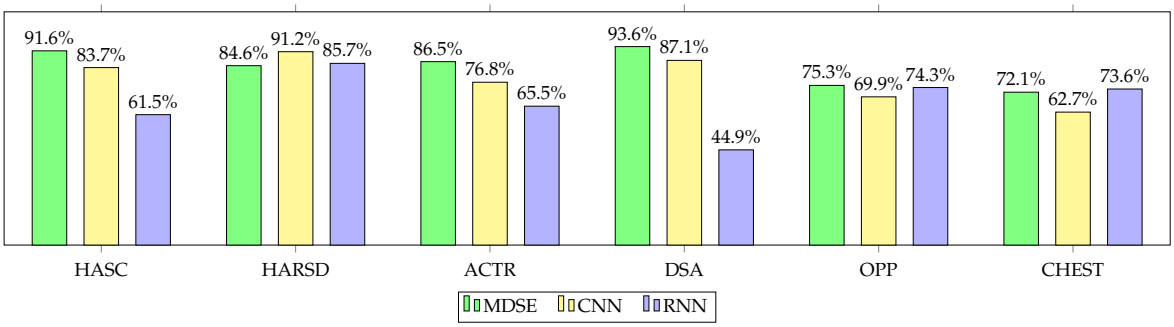


Figure 5.6: Comparing accuracy of MDSE with CNN and RNN on different datasets.

5.3.2 Markov Dynamic Subsequence Ensemble

We study the performance of the proposed MDSE regarding accuracy and computational efficiency. We compare the evaluation results of MDSE to SBC and SWEM, where DT, LR and KNN are used as base classifier separately. The parameters of MDSE are set as

	HASC					HARSD				
Method	Accuracy	#Sub	Cost	Cost (%)	Time	Accuracy	#Sub	Cost	Cost (%)	Time
SBC + DT	83.18%	1	4482.89	10.02%	0.224	80.60%	1	1991.45	10.11%	0.149
SWEM + DT	93.47%	17	44741.59	100%	2.780	85.86%	17	19695.79	100%	1.996
MDSE + DT	91.59%	5.17	10339.73	23.11%	0.769	84.62%	4.02	3142.80	15.96%	0.428
SBC + LR	90.04%	1	4482.89	10.02%	0.249	83.92%	1	1991.45	10.11%	0.155
SWEM + LR	94.83%	17	44741.59	100%	3.141	85.17%	17	19695.79	100%	2.047
MDSE + LR	93.66%	5.10	10335.02	23.10%	0.890	85.31%	6.30	6208.77	31.52%	0.760
SBC + KNN	88.94%	1	4482.89	10.02%	2.591	83.65%	1	1991.45	10.11%	1.013
SWEM + KNN	91.85%	17	44741.59	100%	34.144	84.55%	17	19695.79	100%	14.768
MDSE + KNN	92.43%	3.89	6872.99	15.36%	6.213	84.13%	3.54	2611.47	13.26%	2.861
	ACTR					DSA				
Method	Accuracy	#Sub	Cost	Cost (%)	Time	Accuracy	#Sub	Cost	Cost (%)	Time
SBC + DT	75.66%	1	664.39	10.28%	0.090	73.95%	1	870.72	10.20%	0.104
SWEM + DT	87.35%	17	6463.85	100%	1.346	94.12%	17	8537.68	100%	1.566
MDSE + DT	86.54%	5.32	1550.86	23.99%	0.420	93.60%	5.46	2096.86	24.56%	0.483
SBC + LR	89.38%	1	664.39	10.28%	0.102	85.04%	1	870.72	10.20%	0.109
SWEM + LR	90.93%	17	6463.85	100%	1.485	89.47%	17	8537.68	100%	1.607
MDSE + LR	90.89%	5.82	1804.69	27.92%	0.523	89.47%	8.80	4019.88	47.08%	0.886
SBC + KNN	83.25%	1	664.39	10.28%	2.787	94.34%	1	870.72	10.20%	0.952
SWEM + KNN	85.28%	17	6463.85	100%	35.983	95.88%	17	8537.68	100%	17.549
MDSE + KNN	85.39%	4.21	1087.78	16.83%	7.055	95.66%	3.66	1165.30	13.65%	3.327
	OPP					CHEST				
Method	Accuracy	#Sub	Cost	Cost (%)	Time	Accuracy	#Sub	Cost	Cost (%)	Time
SBC + DT	68.49%	1	1084.32	10.20%	0.124	59.90%	1	1991.45	10.11%	0.169
SWEM + DT	76.71%	17	10634.65	100%	1.648	73.31%	17	19695.79	100%	2.165
MDSE + DT	75.34%	6.94	3651.18	34.33%	0.704	72.13%	6.55	6198.73	31.47%	0.791
SBC + LR	76.71%	1	1084.32	10.20%	0.123	69.31%	1	1991.45	10.11%	0.177
SWEM + LR	77.05%	17	10634.65	100%	1.665	71.08%	17	19695.79	100%	2.386
MDSE + LR	77.40%	7.41	4055.15	38.13%	0.768	71.21%	10.68	11719.38	59.50%	1.492
SBC + KNN	72.95%	1	1084.32	10.20%	0.719	64.13%	1	1991.45	10.11%	1.536
SWEM + KNN	78.77%	17	10634.65	100%	11.687	67.15%	17	19695.79	100%	21.378
MDSE + KNN	79.45%	6.05	3000.77	28.22%	4.252	67.32%	5.23	4657.79	23.65%	6.166

Table 5.2: Comparing MDSE with SBC and SWEM using base classifiers: DT, LR and KNN.

$\beta = 0.7$, $\eta = 3$. We use 5 indicators for the comparison including: (1) 'Accuracy'; (2) '#Sub': the number of used subsequences; (3) 'Cost': the average cost for one instance; (4) 'Cost (%)': the percentage of 'Cost', where the 'Cost (%)' of SWEM is set to 100%, and the others are represented as proportional rates to the 'Cost' of SWEM; (5) 'Time': the average execution time for one instance (millisecond). The experimental results on 6 datasets

	HASC				HARSD			
Method	Accuracy	#Sub	Cost	Cost (%)	Accuracy	#Sub	Cost	Cost (%)
Random $K-2$	89.20%	4	10424.912	23.30%	83.99%	3	3433.619	17.43%
Random $K-1$	90.17%	5	13054.256	29.18%	84.48%	4	4593.724	23.32%
MDSE	**91.59%**	5.17	10339.734	**23.11%**	**84.62%**	4.02	3142.803	**15.96%**
Random K	90.88%	6	15732.360	35.16%	85.17%	5	5752.307	29.21%
Random $K+1$	91.72%	7	18366.606	41.05%	85.03%	6	6928.949	35.18%
Random All (SWEM)	93.47%	17	44741.588	100.00%	85.86%	17	19695.794	100.00%
	ACTR				DSA			
Method	Accuracy	#Sub	Cost	Cost (%)	Accuracy	#Sub	Cost	Cost (%)
Random $K-2$	84.18%	4	1518.817	23.50%	88.16%	4	1994.436	23.36%
Random $K-1$	84.37%	5	1896.888	29.35%	89.96%	5	2494.316	29.22%
MDSE	**86.54%**	5.32	1550.862	**23.99%**	**93.60%**	5.46	2096.855	**24.56%**
Random K	85.42%	6	2277.220	35.23%	90.61%	6	3001.907	35.16%
Random $K+1$	85.46%	7	2657.417	41.11%	91.80%	7	3505.834	41.06%
Random All (SWEM)	87.35%	17	6463.855	100.00%	94.12%	17	8537.676	100.00%
	OPP				CHEST			
Method	Accuracy	#Sub	Cost	Cost (%)	Accuracy	#Sub	Cost	Cost (%)
Random $K-2$	71.92%	5	3107.821	29.22%	69.42%	5	5774.697	29.32%
Random $K-1$	73.63%	6	3758.417	35.34%	70.12%	6	6937.692	35.22%
MDSE	**75.34%**	6.94	3651.182	**34.33%**	**72.13%**	6.55	6198.734	**31.47%**
Random K	75.34%	7	4371.330	41.10%	70.38%	7	8102.068	41.14%
Random $K+1$	77.05%	8	5000.964	47.03%	71.01%	8	9265.011	47.04%
Random All (SWEM)	76.71%	17	10634.647	100.00%	73.31%	17	19695.794	100.00%

Table 5.3: Comparing MDSE with Random K methods (DT as base classifier). Due to the page limitation, we only show the results of Random $K-2$, $K-1$, K, $K+1$ and All, where K is set as the smallest integer that greater than the obtained ensemble size of MDSE.

are shown in Table 5.2. For accuracy comparison, MDSE performs only approximately 1-2% less than SWEM. However, for computational cost comparison, MDSE reduces average 74.41% of the SWEM's computational cost. Specifically, MDSE reduces average 74.32% with DT, 67.41% with LR, and 81.51% with KNN. To explore the statistical significance of the performances of MDSE and SWEM, we conduct Wilcoxon signed-rank test on their results (18 pairs). The returned p-values represent the lowest level of significance of a hypothesis that results in rejection. This value allows one to determine whether two methods have significantly different performance. We set the significance level $\alpha = 0.05$ for the following statistical comparisons. For the accuracy comparison between MDSE and SWEM, the returned p-value $= 0.102434 > \alpha$ fails to reject the null

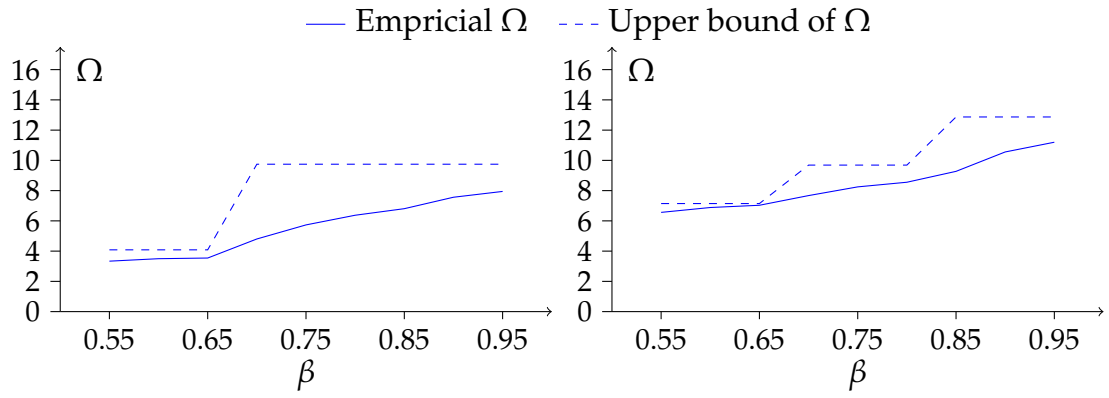
hypothesis of the comparison, implying a similar accuracy performance of the two methods. For the computational cost comparison between MDSE and SWEM, the returned p-value $= 0.000196 < \alpha$ rejects the null hypothesis, implying a significant cost efficiency of MDSE against SWEM. We also compare the accuracy of MDSE to the CNN and RNN. The results on 6 datasets are shown in Figure 5.6. MDSE obtains better performances than both the CNN and RNN on 4 out of 6 datasets. The CNN only shows a better result on HARSD dataset, and the RNN only shows better results on HARSD and CHEST datasets. Designing widely applicable deep neural nets for HAR still need more investigations which can be our future work.

We further evaluate the effectiveness of MDSE by comparing it with a method called Random K which randomly chooses K subsequences for ensemble prediction. We test Random K from $K = 1$ to $|H_X|$. According to the comparison results shown in Table 5.3, when MDSE uses a similar number of subsequences to the Random K, MDSE achieves a higher or similar accuracy, and spends less computational cost.

5.3.3 The Accuracy Constraints

The recognition accuracy can be ensured by the presented constraints Eq. 5.9 and Eq. 5.10, since we can prove that the accuracy is monotonically increasing with respect to the parameters β and η. Therefore, we conduct experiments to study the effects of increasing β and η to accuracy and computational cost. We test MDSE on HASC dataset with β varying from 0.55 to 0.95, and $\eta \in \{3, 4, 5\}$. The 'Accuracy' and 'Cost (%)' results are shown in Figure 5.4a and Figure 5.4b, respectively. The accuracy improves by increasing β and η, which verifies our analysis. The computational cost is also increased, since that the more restrictive constraints will cause MDSE take more subsequences into ensemble, which results in more computations.

A critical problem is how to choose β and η. Given a fixed η, we can use $\Delta\text{Accuracy}/\Delta\text{Cost}$ as an indicator to express the ratio of accuracy increment and cost increment by increasing β. We test $\Delta\text{Accuracy}/\Delta\text{Cost}$ with respect to β and η, where $\mathcal{D}_2$ is used as testing data. The curves of $\Delta\text{Accuracy}/\Delta\text{Cost}$ are plotted in Figure 5.5. We can observe a significant peak when $\beta = 0.7$ and $\eta = 3$. Therefore, using $\beta = 0.7$ and $\eta = 3$ is an empirically optimal choice for MDSE.



(a) $\eta = 3$ (b) $\eta = 6$

Figure 5.7: The dashed curve shows the upper bound of Ω with respect to β, and the solid curve shows the empirical Ω.

5.3.4 The Computational Efficiency of MDSE

The computational efficiency of MDSE is guaranteed, since we can estimate an upper bound of the expected ensemble size Ω. We evaluate the correctness of the upper bound which is presented in Eq. 5.49. We conduct experiments on HASC dataset to obtain the empirical Ω (average ensemble size) of MDSE. We estimate the upper bound of Ω using Eq. 5.49, where the generalization error ϵ is set as the largest empirical error of the base classifiers. According to the results shown in Figure 5.7, the curve of empirical Ω is consistently below the curve of the estimated upper bound, which verifies the correctness of our derived bound for Ω.

5.3.5 Evaluation on Smartphone

We study the performance of MDSE on smartphone regarding accuracy, energy cost and the size of classification model. We compare the evaluation results of MDSE to SBC and SWEM, where DT is used as base classifier. The methods are implemented using Java with Weka Machine Learning Library[1], and tested on a Google Nexus 5X. In order to measure the pure algorithm running cost excluding the impact from sensors, we import the HASC dataset to the smartphone where the data is prepared following our previous Data Preparation steps. We run the methods on the data of testing group (939 instances),

[1]http://www.cs.waikato.ac.nz/ml/weka/

and use PowerTutor [105] to measure a total power consumption. We run each method for 3 times to calculate the average results. According to Table 5.4, MDSE obtains an accuracy about 1.4% less than the SWEM, but reduces about 70.8% energy of the SWEM. Moreover, the model size of MDSE is not large, which demonstrates the feasibility of applying MDSE on mobile devices.

Method	Accuracy	Energy Cost	Model Size
SBC	82.22%	0.95 J	87 KB
SWEM	92.65%	10.03 J	1367 KB
MDSE	91.27%	2.93 J	2958 KB

Table 5.4: Performances on a Google Nexus 5X.

5.4 Conclusion & Discussion

State-of-the-art models using feature representations based on multiple time series subsequences have shown to be effective in Human Activity Recognition (HAR). However, they consume a large amount of energy on mobile devices. In this chapter, we addressed this major issue by formalizing a dynamic subsequence selection problem that minimizes the computational cost with constraints for ensuring the recognition accuracy. We theoretically showed that our presented constraints can guarantee the generalization accuracy at a certain level. To solve the problem, we proposed Markov Dynamic Subsequence Ensemble (MDSE) which learns a policy that chooses the best subsequence given a state of prediction. We then derived an upper bound of the expected ensemble size of MDSE, thus the computational efficiency is guaranteed. We conducted experiments on 6 real human activity datasets to evaluate the performance of MDSE. Comparing to the state-of-the-arts, MDSE demonstrates 70.8% energy reduction that is 3.42 times more energy-efficiency, and achieves a similarly high accuracy.

MDSE greatly enhances the practicality of subsequence ensemble models on mobile devices due to the significant reduction of computational cost with minor recognition accuracy sacrificing. Since MDSE focuses only on reducing model inference computations, it is capable of working together with many conventional energy saving methods which

focus on reducing sensor usages. Accordingly, a future study regarding joint optimizing inference computing and sensing scheduling can be conducted to further improve energy-efficiency of HAR.

Chapter 6

Efficient Human Activity Recognition by Reducing Sensing Cost

In this chapter, we conduct a study on improving energy-efficiency of Human Activity Recognition (HAR) by reducing sensing cost. HAR in real-time requires continuous sampling of data using built-in sensors (e.g., accelerometer), which significantly increases the energy cost and shortens the operating span. Reducing sampling rate can save energy but causes low recognition accuracy. Therefore, choosing adaptive sampling frequency that balances accuracy and energy-efficiency becomes a critical problem in HAR. We formalize the problem as minimizing both classification error and energy cost by choosing dynamically appropriate sampling rates. We propose Datum-Wise Frequency Selection (DWFS) to solve the problem via a continuous state Markov Decision Process (MDP). A policy function is learned from the MDP, which selects the best frequency for sampling an incoming data entity by exploiting a datum related state of the system. We propose a method for alternate learning the parameters of an activity classification model and the MDP that improves both the accuracy and the energy-efficiency. We evaluate DWFS with three real-world HAR datasets, and the results show that DWFS statistically outperforms the state-of-the-arts regarding a combined measurement of accuracy and energy-efficiency.

6.1 Introduction

Human Activity Recognition (HAR) in real-time requires continuous sampling data using built-in sensors (e.g., accelerometer, microphone, and camera), which causes excessive power consumption that greatly shortens the lifespan of the mobile devices [46].

This chapter is derived from: Weihao Cheng, Sarah Erfani, Rui Zhang, Kotagiri Ramamohanarao, "Learning Datum-Wise Sampling Frequency for Energy-Efficient Activity Recognition", *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, USA, 2018.

Simply using a low sampling frequency can save energy, but this comes at the cost of reduced recognition accuracy [49]. Therefore, dynamically selecting proper sampling frequency to balance accuracy and energy-efficiency becomes an emergent challenge for HAR on resource constrained mobile devices.

Most of the existing HAR methods use a fixed sampling frequency for data acquisition. A problem of these methods is that, they can either spend redundant energy for identifying highly discriminable instances, or obtain unsatisfactory accuracy for confusing instances. Several works have been proposed for seeking dynamic sampling frequency for HAR [94, 98, 102]. An A3R method [98] follows a table of heuristic rules for frequency selection, and it decides to switch frequency based on a predefined threshold of inference probability. Some other methods [94,102] solve the problem with a discrete state Markov Decision Process (MDP), which merely takes one of the predefined user states as input and returns actions from a limited space (increase/keep/decrease) for changing the frequency. These methods lack direct information extraction from data instance and cannot find a desired optimal performance on balancing accuracy and energy-efficiency.

In this chapter, we propose an effective method that dynamically chooses sampling frequencies for HAR. In contrast to the existing approaches, our method directly exploits the information from recently observed data instances. Suppose we have a sequence of data entities, where a data entity represents the full information of an activity in a period of time, and it can only be inferred by being sampled into data instance. Our goal is to dynamically choose frequencies to sample each data entity for inference, so that the recognition accuracy and the energy-efficiency are balanced in a desired way. We formalize the problem as finding an optimal classification model and dynamically appropriate sampling frequencies that minimize an objective function regarding overall classification error and total energy cost. We propose Datum-Wise Frequency Selection (DWFS) to address the minimization problem through an MDP, where a policy function is learned for choosing the best sampling frequency based on a continuous state. We assign the policy function with the datum-wise property, where the feature representation of sampled data is utilized to build a connection between MDP states and sampling frequencies. DWFS models a mutual relationship between the classification model and the MDP, thereby, we propose to learn their parameters via an alternate optimization approach. More specifi-

cally, our main contributions can be summarized as follows:

- We formalize a problem which finds an optimal classification model and dynamically appropriate sampling frequencies to minimize an objective function regarding a combined measurement of classification error and energy cost.
- We propose DWFS to solve the minimization problem. DWFS utilizes a continuous state MDP, where a datum-wise policy function is proposed to select the best sampling frequency by directly exploiting the information from recently sampled data.
- DWFS unifies the parameters of the classification model and the policy function in one model, thereafter, we propose an alternate optimization approach where the parameters are mutually enhanced.

We conduct extensive experiments on 3 real-world HAR datasets to evaluate DWFS. The results demonstrate that DWFS statistically outperforms the state-of-the-arts in terms of a combined measurement of accuracy and energy-efficiency.

6.2 Methodology

In this section, we first formalize the problem of balancing recognition accuracy and energy-efficiency as minimizing an objective function regarding recognition error and energy cost. Then, we propose Datum-Wise Frequency Selection (DWFS) based on a continuous state MDP, which can sequentially choose the optimal sensor sampling frequencies for incoming data entities.

6.2.1 Problem Statement

Let $F = \{f_1, f_2, ..., f_K\}$ be a set of K sampling frequencies supported by a sensor, where $f_1 < f_2 < ... < f_K$. The energy costs of using these frequencies are $c_{f_1}, c_{f_2}, ..., c_{f_K}$, respectively, and we have $c_{f_1} < c_{f_2} < ... < c_{f_K}$. Suppose we have m activities to recognize. Let $\mathcal{Y}$ be a set of activity labels such that $\mathcal{Y} = \{1, 2, ..., m\}$. Let y be an activity label such that $y \in \mathcal{Y}$. Suppose $\boldsymbol{x}$ is an data entity which represents the full information of an activity

in a period of time. As $\boldsymbol{x}$ is in a space of infinite dimension, it is required to be sampled by a sensor for processing. Let $g(\boldsymbol{x}, f)$ be a sampling function which returns an observed data instance $\tilde{\boldsymbol{x}} = g(\boldsymbol{x}, f)$ by sampling $\boldsymbol{x}$ with a frequency $f \in F$. It is worth noting that f is implicitly represented by $\tilde{\boldsymbol{x}}$ for simplicity of notation. Given an $\tilde{\boldsymbol{x}}$, we can use a classification model $p(y \mid \boldsymbol{x}\,; \theta)$ parameterized by θ to make a probabilistic inference, then the output of the model is a vector of probabilities $\boldsymbol{p} \in [0, 1]^m$:

$$\boldsymbol{p} = (p_1, p_2, ..., p_m), \tag{6.1}$$

where the y-th element p_y of $\boldsymbol{p}$ is the probability of the activity y:

$$p_y = p(y \mid \tilde{\boldsymbol{x}}\,; \theta). \tag{6.2}$$

Let $\hat{y}$ be the inferred activity label that $\hat{y} = \text{argmax}_y\, p(y \mid \tilde{\boldsymbol{x}}\,; \theta)$. Let $1\{\hat{y} \neq y\}$ be the error measurement regarding the inferred activity that $1\{\hat{y} \neq y\}$ equals 1 if $\hat{y} \neq y$ and 0 otherwise. Suppose there is a sequence of labeled data entities $Q = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), ..., (\boldsymbol{x}^{(N)}, y^{(N)})\}$. Our goal is twofold: 1) we want to find an optimal θ for the classification model, and 2) for each data entity $\boldsymbol{x}^{(t)}$, we want to select a proper frequency $f^{(t)} \in F$ to obtain $\tilde{\boldsymbol{x}}^{(t)} = g(\boldsymbol{x}^{(t)}, f^{(t)})$, that a combined measurement of the overall classification error and the total energy cost is minimized:

$$\min_{\theta, f^{(1)}, ..., f^{(N)}} \sum_{t=1}^{N} 1\{\hat{y}^{(t)} \neq y^{(t)}\} + \lambda \sum_{t=1}^{N} c_{f^{(t)}}, \tag{6.3}$$

where λ is a predefined weight parameter between the error and the cost. We denote this combined measurement as **Error-Cost Index**. For the convenience of using function optimization to reduce the proposed Error-Cost Index, we replace the classification error $1\{\hat{y}^{(t)} \neq y^{(t)}\}$ in Eq. 6.3 with cross-entropy loss, and formalize an objective function as:

$$\min_{\theta, f^{(1)}, ..., f^{(N)}} \sum_{t=1}^{N} -\log p(y^{(t)} \mid \tilde{\boldsymbol{x}}^{(t)}\,; \theta) + \lambda \sum_{t=1}^{N} c_{f^{(t)}}, \tag{6.4}$$

where λ is a predefined weight parameter between the cross-entropy loss and the energy cost. In an experimental environment, one can naively test all the possible sampling fre-

quencies for each data entity to obtain the optimum solution. However, this is infeasible for real-time inference, since a sensor can only sample an incoming entity once. Therefore, we need to find an effective way to solve the problem.

6.2.2 Datum-Wise Frequency Selection (DWFS)

We propose DWFS to solve the optimization problem of Eq. 6.4. DWFS effectively predicts the best sampling frequency for an incoming data entity based on recent context. A significant advantage of DWFS over the existing methods is that, it models a direct connection from data to sampling frequencies, thus the model is so called **Datum-Wise**. DWFS solves the minimization problem of Eq. 6.4 by incorporating a continuous state MDP, where a **policy** π is learned to select the next sampling frequency based on the current state of the MDP. The state of the MDP is defined as a pair of two components: 1) the feature representation of a previously sampled data instance, and 2) the inference probabilities of this instance. The detailed description of DWFS is provided in the rest of this section. We first define our continuous state MDP, and transform the original problem into the MDP problem. We then design the policy of the MDP. Next, we introduce an alternate learning approach to optimize the parameters of DWFS. Finally, we describe the algorithm of DWFS to select the sampling frequencies in real-time inference.

The Markov Decision Process of DWFS

Let $\phi(\tilde{\boldsymbol{x}}) \in \mathbb{R}^d$ be the feature representation of $\tilde{\boldsymbol{x}} = g(\boldsymbol{x}, f)$, where d is the dimension of $\phi(\tilde{\boldsymbol{x}})$. We introduce the MDP to solve our problem as follows:

- S is an infinite space of states. Each state $\boldsymbol{s} \in S$ is defined as a vector $\boldsymbol{s} = (\phi(\tilde{\boldsymbol{x}}), \boldsymbol{p}) \in \mathbb{R}^{d+m}$, which is concatenated by the feature representation $\phi(\tilde{\boldsymbol{x}})$ and the label probabilities $\boldsymbol{p}$ of $\tilde{\boldsymbol{x}}$. Specifically, we define $\boldsymbol{s}^{(t)} = (\phi(\tilde{\boldsymbol{x}}^{(t)}), \boldsymbol{p}^{(t)})$, where $\boldsymbol{p}^{(t)}$ is the label probabilities of $\tilde{\boldsymbol{x}}^{(t)}$.

- A is a set of actions that $A = \{a_1, a_2, ..., a_K\}$. The action $a_k \in A$ maps to the frequency $f_k \in F$, which indicates the choice of f_k. We can consider that $a_k \equiv f_k$.

- $P_{sa}(\boldsymbol{s}')$ is the transition probability function. Given a state $\boldsymbol{s} \in S$ and an action

$a \in A$, $P_{sa}(\boldsymbol{s}')$ returns the probability of the next state $\boldsymbol{s}'$ by taking the action a.

- γ is the discount factor. We set $\gamma = 1$ as for no discount involved, thereby every incoming data entity can be considered equally.

- $R(\boldsymbol{s})$: $S \mapsto \mathbb{R}$ is the reward function. It represents a valuable rewarded by visiting a state $\boldsymbol{s} = (\phi(\tilde{\boldsymbol{x}}), \boldsymbol{p})$. We define the reward function for our problem as:

$$R(\boldsymbol{s}) = \log p(y \mid g(\boldsymbol{x}, f)\,;\,\theta) - \lambda c_f, \tag{6.5}$$

  where $f \in F$ is the frequency used to sample $\boldsymbol{x}$. Particularly, we define $R(\mathbf{0}) = 0$, where $\mathbf{0}$ is a zero vector.

The dynamics of the MDP proceeds as follows: We start at an initial state $\boldsymbol{s}^{(0)} = \mathbf{0}$. We then choose the first action $a^{(0)}$, and use the corresponding sampling frequency $f^{(1)}$ to sample the first data entity $\boldsymbol{x}^{(1)}$. As the result of $f^{(1)}$, we step into the next state $\boldsymbol{s}^{(1)} = (\phi(\tilde{\boldsymbol{x}}^{(1)}), \boldsymbol{p}^{(1)})$. We then choose the second action $a^{(1)}$ based on $\boldsymbol{s}^{(1)}$ and obtain a new state. We repeat this procedure until the last data entity $\boldsymbol{x}^{(N)}$ is sampled. An intuitive representation of the MDP dynamics is shown in Figure 6.1, and the total reward of visiting the state sequence $\boldsymbol{s}^{(0)}$, $\boldsymbol{s}^{(1)}$, ..., $\boldsymbol{s}^{(N)}$ is calculated as:

$$\begin{aligned} &R(\boldsymbol{s}^{(0)}) + \gamma R(\boldsymbol{s}^{(1)}) + \gamma^2 R(\boldsymbol{s}^{(2)}) + ... + \gamma^N R(\boldsymbol{s}^{(N)}) \\ =&R(\boldsymbol{s}^{(0)}) + \sum_{t=1}^{N} R(\boldsymbol{s}^{(t)}). \end{aligned} \tag{6.6}$$

The essential problem of MDP is to find a policy function $\pi : S \mapsto A$, which chooses the best action for a given state to maximize a value function defined as:

$$V(\boldsymbol{s}, t) = \begin{cases} R(\boldsymbol{s}) + \int_{\boldsymbol{s}' \in S} P_{sa}(\boldsymbol{s}') V(\boldsymbol{s}', t+1) & t < N \\ R(\boldsymbol{s}) & t = N \end{cases}. \tag{6.7}$$

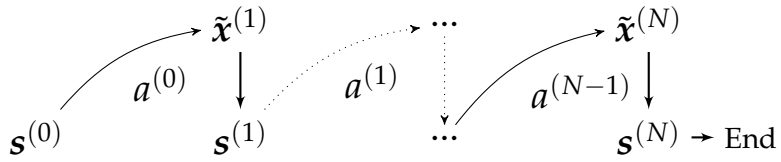


Figure 6.1: The dynamics of the MDP.

The value function $V(\boldsymbol{s}, t)$ returns the total rewards of the MDP starting from $\boldsymbol{s}$ and t. By incorporating the MDP, we can rewrite the problem of Eq. 6.4 as:

$$\begin{aligned}
&\min_{\theta, f^{(1)}, \ldots, f^{(N)}} \sum_{t=1}^{N} -\log p(y^{(t)} \mid \tilde{\boldsymbol{x}}^{(t)}\,; \theta) + \lambda \sum_{t=1}^{N} c_{f^{(t)}} \\
&= \min_{\theta, f^{(1)}, \ldots, f^{(N)}} \sum_{t=1}^{N} -\log p(y^{(t)} \mid \tilde{\boldsymbol{x}}^{(t)}\,; \theta) + \lambda c_{f^{(t)}} \\
&= \max_{\theta, f^{(1)}, \ldots, f^{(N)}} \sum_{t=1}^{N} \log p(y^{(t)} \mid \tilde{\boldsymbol{x}}^{(t)}\,; \theta) - \lambda c_{f^{(t)}} \\
&= \max_{\theta, \pi} R(\boldsymbol{s}^{(0)}) + \sum_{t=1}^{N} R(\boldsymbol{s}^{(t)}) \\
&= \max_{\theta, \pi} V(\boldsymbol{s}^{(0)}, 0).
\end{aligned} \tag{6.8}$$

Therefore, our goal is to find optimal classification parameter θ^* and policy π^* that maximize the value function $V(\boldsymbol{s}^{(0)}, 0)$.

Modeling the Policy

Suppose we have a fixed θ. Let $V^*(\boldsymbol{s}, t)$ be the optimal value function regarding π^*. For $t = N$, we have $V^*(\boldsymbol{s}, t) = R(\boldsymbol{s})$. For $t < N$, we can use Bellman's equation to recursively calculate $V^*(\boldsymbol{s}, t)$ as:

$$V^*(\boldsymbol{s}, t) = R(\boldsymbol{s}) + \max_{a \in A} \int_{\boldsymbol{s}' \in S} P_{\boldsymbol{s}a}(\boldsymbol{s}') V^*(\boldsymbol{s}', t+1), \tag{6.9}$$

where the optimal values of V^* for each state are obtained in a dynamic programming manner. Correspond to V^*, the optimal policy function π^* regarding $\boldsymbol{s}$ is calculated as:

$$\pi^*(\boldsymbol{s}) = \operatorname*{argmax}_{a \in A} \mathbb{E}_{t \sim \mathcal{T}} \int_{\boldsymbol{s}' \in S} P_{\boldsymbol{s}a}(\boldsymbol{s}') V^*(\boldsymbol{s}', t), \tag{6.10}$$

where $\mathcal{T}$ is a distribution of t. Since $\pi^*(\boldsymbol{s})$ is independent of t, the output of $\pi^*(\boldsymbol{s})$ given $\boldsymbol{s}$ should be maximized on $\mathcal{T}$. As the state $\boldsymbol{s} = (\phi(\tilde{\boldsymbol{x}}), \boldsymbol{p})$ is in a continuous space, we are unable to use the traditional tabular approach to obtain $\pi(\boldsymbol{s})$. Instead, we stimulate $\pi(\boldsymbol{s})$ with a probabilistic model $p(a \mid \boldsymbol{s}\,;\, \psi)$ parameterized by ψ:

$$\pi(\boldsymbol{s}) = \underset{a \in A}{\operatorname{argmax}}\, p(a \mid \boldsymbol{s}\,;\, \psi). \tag{6.11}$$

We consider the parameter $\psi = \psi(\boldsymbol{s})$ as a function of $\boldsymbol{s}$. The probability $p(a \mid \boldsymbol{s}\,;\, \psi)$ is then given through a softmax assignment based on $\psi(\boldsymbol{s})$:

$$p(a = a_k \mid \boldsymbol{s}\,;\, \psi) = \frac{e^{\psi_k(\boldsymbol{s})}}{\sum_{i=1}^{K} e^{\psi_i(\boldsymbol{s})}}. \tag{6.12}$$

Therefore, we can learn the parameter ψ to obtain the policy $\pi(\boldsymbol{s})$ of the MDP, and our problem of Eq. 6.8 is then rewritten as:

$$\max_{\theta, \psi} V(\boldsymbol{s}^{(0)}, 0). \tag{6.13}$$

Learning DWFS

Learning DWFS consists of two tasks: optimizing the parameter θ for the classification model and optimizing the parameter ψ for the MDP. We consider that θ and ψ are affected by each other, and can be mutually enhanced. It is clear that ψ can be refined based on θ, as the MDP reward $R(\boldsymbol{s})$ is a function of θ. The problem is how to refine θ based on ψ. We propose to use the results of the MDP to regulate the sample weights for learning θ. Given a ψ and a training sequence Q, we can use the MDP to predict a sequence of frequencies $\{f^{(1)}, ..., f^{(N)}\}$, one for each data entity $\boldsymbol{x}^{(t)}$. We assign a high weight to $g(\boldsymbol{x}^{(t)}, f^{(t)})$ for learning θ, so that θ can put more emphasis on the instances sampled by the chosen frequencies, which improves the classification accuracy. Thereafter, θ can be refined based on ψ. We propose an alternate learning approach to optimize θ and ψ. Suppose we have a set of training sequences $\{Q_0, Q_1, ..., Q_n\}$. At the beginning, we use

Q_0 to learn an initial θ by:

$$\min_{\theta} \sum_{(\boldsymbol{x},y)\in Q_0} \sum_{f\in F} -\log p(y \mid g(\boldsymbol{x}, f)\,;\,\theta), \tag{6.14}$$

where all the sampled data instances are considered equally. Given a sequence Q iteratively picked from $\{Q_1, ..., Q_n\}$, we fix θ, and extract a set of state-action pairs H from Q based on Eq. 6.10. Each pair $(\boldsymbol{s}, a) \in H$ is used as an instance for learning the MDP, thereby, the parameter ψ is optimized as:

$$\min_{\psi} \sum_{(\boldsymbol{s},a)\in H} -\log p(a \mid \boldsymbol{s}\,;\,\psi). \tag{6.15}$$

We then fix ψ, and use the MDP to predict Q, that a sequence of chosen frequencies $\{f^{(1)}, ..., f^{(N)}\}$ is obtained. Let $\mu_{\boldsymbol{x},f}$ be the weight of the instance $g(\boldsymbol{x}, f)$, and we optimize θ as:

$$\begin{aligned} &\min_{\theta} \sum_{(\boldsymbol{x},y)\in Q} \sum_{f\in F} -\mu_{\boldsymbol{x},f} \log p(y \mid g(\boldsymbol{x}, f)\,;\,\theta), \\ &\text{with}\ \ \mu_{\boldsymbol{x},f} = \beta, \ \ \text{if}\ \boldsymbol{x} = \boldsymbol{x}^{(t)}\ \text{and}\ f = f^{(t)}, \\ &\qquad\ \ \mu_{\boldsymbol{x},f} = 1, \ \ \text{otherwise}, \end{aligned} \tag{6.16}$$

where $\beta > 1$ is a predefined parameter for imposing the weight. Accordingly, we alternately optimize θ and ψ until the training set has been iterated for L rounds. Before concluding the entire DWFS learning algorithm, we first provide the algorithm for obtaining the state-action pairs set H. Based on the general framework of Eq. 6.10, we iteratively calculate the optimal $V^*(\boldsymbol{s}, t)$ from $t = N$ to 1. Then, for each t from $t = 1$ to $N - 1$, we collect K state-action pairs, where one pair is for one action and there are K actions in total. Let $\tilde{\boldsymbol{x}}_k^{(t)}$ be the t-th instance $\tilde{\boldsymbol{x}}^{(t)}$ sampled by the frequency f_k that $\tilde{\boldsymbol{x}}_k^{(t)} = g(\boldsymbol{x}^{(t)}, f_k)$. Let $\boldsymbol{p}_k^{(t)}$ be the probabilities of $\tilde{\boldsymbol{x}}_k^{(t)}$, and $\boldsymbol{s}_k^{(t)} = (\phi(\tilde{\boldsymbol{x}}_k^{(t)}), \boldsymbol{p}_k^{(t)})$. We set the transition probability function $P_{\boldsymbol{s}a}(\boldsymbol{s}')$ as:

$$P_{\boldsymbol{s}a}(\boldsymbol{s}') = \begin{cases} 1, & \boldsymbol{s} = \boldsymbol{s}_k^{(t)}, a = a_{k'}, \boldsymbol{s}' = \boldsymbol{s}_{k'}^{(t+1)}, \\ 0, & \text{otherwise}. \end{cases} \tag{6.17}$$

Algorithm 7 DWFS Generate H

Input: A sequence of training entities $Q = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), ..., (\boldsymbol{x}^{(N)}, y^{(N)})\}$
Output: A set of state-action pairs H

while $t = N, N-1, ..., 1$ **do**
  for $k = 1, 2, ..., K$ **do**
    $\tilde{\boldsymbol{x}}_k^{(t)} = g(\boldsymbol{x}^{(t)}, f_k)$
    $\boldsymbol{s}_k^{(t)} = (\phi(\tilde{\boldsymbol{x}}_k^{(t)}), \boldsymbol{p}_k^{(t)})$
    if $t == N$ **then**
      $V(\boldsymbol{s}_k^{(t)}) = R(\boldsymbol{s}_k^{(t)})$
    else
      $V(\boldsymbol{s}_k^{(t)}) = R(\boldsymbol{s}_k^{(t)}) + \max_{1 \le k' \le K} V(\boldsymbol{s}_{k'}^{(t+1)})$
    end if
  end for
end while
$H = \emptyset$
while $t = 1, 2, ..., N-1$ **do**
  for $k = 1, 2, ..., K$ **do**
    $k' = \text{argmax}_{1 \le k'' \le K} V(\boldsymbol{s}_{k''}^{(t+1)})$
    $H = H \bigcup \{(\boldsymbol{s}_k^{(t)}, a_{k'})\}$
  end for
end while
return H

Algorithm 8 DWFS Learning

Input: A set of training sequences $\{Q_0, Q_1, ..., Q_n\}$
Output: Parameters θ and ψ

Initialize θ, ψ
Learn θ by Eq. 6.14 using Q_0
for $l = 1, 2, ..., L$ **do**
  for $Q \in \{Q_1, ..., Q_n\}$ **do**
    Fix θ, update ψ by Eq. 6.15 using Q
    Fix ψ, update θ by Eq. 6.16 using Q
  end for
end for
return θ, ψ

Therefore, the state $\boldsymbol{s}_k^{(t)}$ only transits to $\boldsymbol{s}_{k'}^{(t+1)}$ by taking the action $a_{k'}$, which indicates to use $f_{k'}$ for sampling the next entity $\boldsymbol{x}^{(t+1)}$. The pseudocode of H generating algorithm is provided in Algorithm 7. Up to now, we can conclude the DWFS learning algorithm, where the pseudocode is provided in Algorithm 8.

Algorithm 9 DWFS Inference

Input: A sequence of data entities $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(N)}$ ▷ N can be an arbitrary number or infinity.
Output: A sequence of inferred labels $\hat{y}^{(1)}, \hat{y}^{(2)}, ..., \hat{y}^{(N)}$
1: $\boldsymbol{s}^{(0)} = \mathbf{0}$
2: **while** $t = 1, 2, ..., N$ **do**
3: $a^{(t-1)} = \pi^*(\boldsymbol{s}^{(t-1)})$
4: Choose $f^{(t)}$ corresponding to $a^{(t-1)}$
5: $\tilde{\boldsymbol{x}}^{(t)} = g(\boldsymbol{x}^{(t)}, f^{(t)})$
6: $\hat{y}^{(t)} = \text{argmax}_y \, p(y \mid \tilde{\boldsymbol{x}}^{(t)}; \theta^*)$
7: $\boldsymbol{s}^{(t)} = (\phi(\tilde{\boldsymbol{x}}^{(t)}), \boldsymbol{p}^{(t)})$
8: **end while**
9: **return** $\hat{y}^{(1)}, \hat{y}^{(2)}, ..., \hat{y}^{(N)}$

DWFS Inference Algorithm

After learning the parameter θ^* of the classification model and the parameter ψ^* of the policy function $\pi^*(\boldsymbol{s})$, we can use the DWFS inference algorithm to select frequencies on a testing sequence. We start at an initial state $\boldsymbol{s}^{(0)} = \mathbf{0}$. For an incoming data entity $\boldsymbol{x}^{(t)}$, we obtain the frequency via the policy as:

$$f^{(t)} = a^{(t-1)} = \pi^*(\boldsymbol{s}^{(t-1)}). \tag{6.18}$$

We then change the sampling frequency of the sensor to $f^{(t)}$, and the sensor will sample out an observed data instance $\tilde{\boldsymbol{x}}^{(t)} = g(\boldsymbol{x}^{(t)}; f^{(t)})$. We use the classification model with θ^* to infer the activity label $\hat{y}^{(t)}$ for $\tilde{\boldsymbol{x}}^{(t)}$:

$$\hat{y}^{(t)} = \underset{y}{\text{argmax}} \, p(y \mid \tilde{\boldsymbol{x}}^{(t)}; \theta^*). \tag{6.19}$$

We obtain the current state as $\boldsymbol{s}^{(t)} = (\phi(\tilde{\boldsymbol{x}}^{(t)}), \boldsymbol{p}^{(t)})$, which will be used to predict the frequency for the next data entity $\boldsymbol{x}^{(t+1)}$. We provide the pseudocode of the DWFS inference algorithm in Algorithm 9. It is worth noting that the length N of the testing sequence can be an arbitrary number or infinity, therefore, DWFS is capable to be applied in real-world scenarios.

6.3 Empirical Evaluation

In this section, we evaluate the performance of DWFS in terms of recognition accuracy, energy cost, and the Error-Cost Index as shown in Eq. 6.3. The experimental scripts are written in Python 2.7 on a 64-bit Ubuntu 14.04 LTS operating system.

Datasets: We use 3-axis acceleration data of 3 real-world HAR datasets to evaluate the performance of DWFS. 1) *Human Activity Sensing Consortium dataset (HASC)* [43]: The data of 6 activities is collected at a sensing rate of 100Hz using iPhone and iPod Touch. 2) *Human Activity Recognition on Smartphones Dataset (HARSD)* [5]: The data of 6 activities is collected at a sensing rate of 50Hz using a Samsung Galaxy S II. 3) *Daily Sport Activities dataset (DSA)* [11]: The data of 19 activities is collected at a sensing rate of 25Hz using body-worn sensors (we only use the data collected by the sensor placed on a subject's torso).

Experiment Settings: Let $\mathcal{D}$ be a dataset that contains a number of time series, where each one is collected independently from the other. Let $\mathcal{D}_y$ be the subset corresponding to activity y, and we have $\mathcal{D} = \bigcup_{y \in \mathcal{Y}} \mathcal{D}_y$. For each subset $\mathcal{D}_y$, we randomly split $\mathcal{D}_y$ into 5 approximately equal sized groups: $\mathcal{D}_{y,1}$, $\mathcal{D}_{y,2}$, ..., $\mathcal{D}_{y,5}$, where the size of each group is between $\lfloor |\mathcal{D}_y|/5 \rfloor$ and $\lceil |\mathcal{D}_y|/5 \rceil$. We conduct experiments based on 5-fold cross-validation: e.g., for the first evaluation, the testing set is aggregated as $\bigcup_{y \in \mathcal{Y}} \mathcal{D}_{y,1}$, and the training set is aggregated by all the rest groups as $\bigcup_{y \in \mathcal{Y}} (\mathcal{D}_{y,2} \bigcup \mathcal{D}_{y,3} \bigcup \mathcal{D}_{y,4} \bigcup \mathcal{D}_{y,5})$[1]. The experimental results are reported as the average results of the 5 folds. For each time series data in the training set, we divide it into several 5 seconds time series segments without overlapping, that each segment is considered as a data entity $\boldsymbol{x}$ paired with a label y. We preserve the order of the segments to generate a sequence $Q = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), ..., (\boldsymbol{x}^{(N)}, y^{(N)})\}$. Accordingly, we obtain a set of training sequences. We randomly concatenate all the testing time series into one time series, and transform it into a testing sequence following the same way of processing training data. For convenient descriptions, we denote N as the length of the testing sequence in the rest of this section.

The frequencies F used for each dataset are set as follows: 1) *HASC*: 5Hz, 16Hz, 50Hz, 100Hz; 2) *HARSD*: 2Hz, 5Hz, 16Hz, 50Hz; 3) *DSA*: 2Hz, 5Hz, 16Hz, 25Hz. The highest

[1]This cross-validation setting avoids the issue reported in [34] that training and testing data could be highly similar by traditional data splitting ways.

frequency in F is the frequency used to collect the dataset. Given $F = \{f_1, f_2, ..., f_K\}$, for each frequency $f_k \in F$, the corresponding energy cost is set as $c_{f_k} = f_k / f_K$, so that c_{f_k} is normalized in the range of $(0, 1]$. The normalization of c_{f_k} ensures that a misclassification or a usage of the highest frequency for a data entity incur the same loss when $\lambda = 1.0$. The feature representation $\phi(\tilde{\boldsymbol{x}})$ is set as Fourier coefficients extracted from $\tilde{\boldsymbol{x}}$, where the coefficients are the intensities of the frequencies from 0Hz to 2Hz (step-size of 0.1Hz). The activity classification model takes $\phi(\tilde{\boldsymbol{x}})$ as input. The iteration round L for training is set to 5.

Settings of DWFS: We use softmax regression as the classification model of DWFS, thus the parameter θ is an $m \times (d + 1)$ matrix including intercepts, and we optimize θ iteratively by Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. We model the parameter ψ as a $K \times (d + m)$ matrix, that $\psi_k(\boldsymbol{s}) = \psi_k \cdot \boldsymbol{s}$, and we optimize ψ iteratively via BFGS as well. We set the parameter $\beta = 1.2$, which is used to impose instance weight.

Settings of Baselines: We test 6 baseline methods in comparison with DWFS. The last two baselines are the variants of DWFS, where we test two different learning approaches instead of alternate learning. The settings of the baselines are as follows:

1 Constant sampling frequency: We use constant frequency to sample data. For each frequency $f_k \in F$, we train and test a classification model with the data sampled by f_k. Therefore, we report K results for this method.

2 Random: We randomly choose sampling frequencies for this method.

3 MDP-DS: We implement a discrete state MDP which takes the inferred activity label of the most recent instance as the state of MDP. The actions of the MDP is the next frequency to choose.

4 RNN: We implement a Recurrent Neural Network (RNN) based method to choose sampling frequency, where the architecture of the RNN consists of an LSTM layer with 32-dim output, a dense layer with K-dim output, and a softmax layer.

5 DWFS-SL (Separate Learning): We first learn θ on the entire training data, and learn ψ based on θ, that there is no dependence of θ on ψ.

6 DWFS-CVL (Cross-Validation Learning): We learn θ on the entire training data,

HASC				
f	100Hz	50Hz	16Hz	5Hz
Accuracy	88.79 ($\pm$ 1.84)	88.20 ($\pm$ 2.10)	84.92 ($\pm$ 2.93)	71.46 ($\pm$ 2.78)
Energy	327.42 J/h	81.20 J/h	51.16 J/h	10.62 J/h
HARSD				
f	50Hz	16Hz	5Hz	2Hz
Accuracy	81.59 ($\pm$ 2.21)	81.65 ($\pm$ 2.32)	78.33 ($\pm$ 2.56)	73.70 ($\pm$ 2.82)
Energy	81.20 J/h	51.16 J/h	10.62 J/h	3.01 J/h
DSA				
f	25Hz	16Hz	5Hz	2Hz
Accuracy	77.33 ($\pm$ 3.63)	76.18 ($\pm$ 4.33)	72.49 ($\pm$ 3.39)	63.57 ($\pm$ 3.79)
Energy	55.45 J/h	51.16 J/h	10.62 J/h	3.01 J/h

Table 6.1: Recognition Accuracy versus Energy Cost (Accelerometer). The unit of the energy cost values is Joule per hour (J/h).

and learn ψ via 4-fold cross-validation on the training data, where 3/4 of the data is used to learn a temporal $\tilde{\theta}$, and ψ is iteratively learned based on $\tilde{\theta}$ of each fold.

6.3.1 Recognition Accuracy versus Energy Cost

We study the recognition accuracy and the energy cost using different sampling frequencies, where an accelerometer is utilized as the sensor. Given a dataset, we train K classifiers, one for each frequency in F, where the k-th classifier is trained with the data instances sampled using the k-th frequency. The recognition accuracy of each classifier is calculated as:

$$\frac{\#\{\text{correct inference}\}}{N}, \tag{6.20}$$

where $\#\{\text{correct inference}\}$ represents the number of correct inferences. The energy cost of each frequency is derived from the work [73], where the authors provided a comprehensive list of energy consumptions regarding accelerometer sampling rates. According to the results shown in Table 6.1, one can generally obtain a better recognition accuracy with a higher sampling frequency, however, this will also bring more energy expenditures. Therefore, learning dynamic sampling frequency that finds a balanced performance between accuracy and cost is important for HAR on resource constrained mobile platforms.

6.3.2 Evaluating the Performance of DWFS

We compare the performance of DWFS to the 6 baselines regarding the Error-Cost Index shown in Eq. 6.3. We test the methods with various settings of λ from 0.0 to 1.0. For easy comparison, the final testing results of the Error-Cost Indexes are multiplied by $\frac{100}{N}$. The results on datasets: HASC, HARSD, and DSA, are shown in Table 6.2, 6.3, and 6.4, respectively. The proposed DWFS outperforms than other methods for most of the settings of λ from 0.1 to 1.0 (step of 0.1). For $\lambda = 0$, DWFS fails to outperform the method of using the highest frequency. Since the energy cost is not considered when $\lambda = 0$, always choosing the highest frequency will obtain the best expected performance. Due to the empirical uncertainty, DWFS cannot always choose the best frequency in testing, therefore, it is defeated by the constant sampling scheme for the case of $\lambda = 0$. To statistically compare the performance of DWFS with the baselines, we conduct the Wilcoxon signed-ranked test on the results of the 3 datasets. The returned R^+ and R^- correspond to the sum of the ranks of the differences above and below zero, respectively. The returned p-value represents the lowest level of significance of a hypothesis that results in rejection. This value allows one to determine whether two methods have significantly different performances. According to the results shown in Table 6.5, the p-values in comparisons of DWFS with Random, MDP-DS, RNN, and DWFS-CVL, reject the null hypotheses for the Error-Cost Index with a level of significance of $\alpha = 0.05$ on all the datasets. The p-values in comparisons of DWFS with DWFS-SL, reject the null hypotheses for the Error-Cost Index with a level of significance of $\alpha = 0.05$ on HARSD and DSA datasets.

6.3.3 Insight Study of DWFS

We further investigate DWFS regarding 4 relationships: 1) classification error and λ; 2) energy cost and λ; 3) classification error and energy cost; 4) frequency changing rate and λ. The frequency changing rate is defined as:

$$\frac{\#\{\text{frequency changes}\}}{N}, \tag{6.21}$$

which can be used to reveal the effectiveness of DWFS. According to the results shown in Figure 6.2a, 6.2b, and 6.2c, the classification error continuously increases, and the en-

ergy cost continuously decreases with respect to λ, due to the imposed penalty on the energy cost causing sampling frequencies are dynamically reduced. According to the results shown in Figure 6.2d, there is no significant pattern between frequency changing rate and λ, but we can observe that DWFS maintains an approximately stable changing rate between 0.1 and 0.3, which demonstrates the robustness of DWFS in terms of λ. In real-life scenarios, the chosen of λ depends on the requirement of energy-efficiency. In the case that accuracy is critical but energy reservation is substantial, a lower λ is recommended. In the case that accuracy is not critical but energy consumption is highly constrained, then a higher λ is recommended. For different applications, the value of λ should be determined differently. DWFS mainly provides a general framework of using λ to balance accuracy and energy-efficiency, while the practical usage of the model requires further studies.

HASC Dataset						
Method	$\lambda = 0.0$	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.3$	$\lambda = 0.4$	$\lambda = 0.5$
100Hz	**11.21**($\pm$1.84)	21.21($\pm$1.84)	31.21($\pm$1.84)	41.21($\pm$1.84)	51.21($\pm$1.84)	61.21($\pm$1.84)
50Hz	11.80($\pm$2.10)	16.80($\pm$2.10)	21.80($\pm$2.10)	26.80($\pm$2.10)	31.80($\pm$2.10)	36.80($\pm$2.10)
16Hz	15.08($\pm$2.93)	16.68($\pm$2.93)	18.28($\pm$2.93)	19.88($\pm$2.93)	21.48($\pm$2.93)	23.08($\pm$2.93)
5Hz	28.54($\pm$2.78)	29.04($\pm$2.78)	29.54($\pm$2.78)	30.04($\pm$2.78)	30.54($\pm$2.78)	31.04($\pm$2.78)
Random	16.54($\pm$2.15)	21.14($\pm$2.14)	25.74($\pm$2.12)	30.34($\pm$2.11)	34.94($\pm$2.09)	39.54($\pm$2.08)
MDP-DS	13.88($\pm$2.26)	18.85($\pm$3.25)	21.83($\pm$4.07)	23.52($\pm$3.66)	25.14($\pm$3.84)	26.40($\pm$3.49)
RNN	12.01($\pm$2.32)	16.74($\pm$3.42)	18.39($\pm$2.96)	20.01($\pm$2.47)	21.32($\pm$3.07)	23.12($\pm$3.42)
DWFS-CVL	12.22($\pm$1.71)	18.12($\pm$3.04)	19.88($\pm$3.31)	21.16($\pm$3.07)	22.73($\pm$2.87)	23.72($\pm$2.66)
DWFS-SL	11.54($\pm$2.21)	16.30($\pm$2.58)	18.31($\pm$3.03)	**19.29**($\pm$2.94)	21.08($\pm$3.45)	22.51($\pm$3.63)
DWFS	12.07($\pm$2.91)	**16.24**($\pm$2.67)	**17.83**($\pm$3.01)	19.54($\pm$2.67)	**20.87**($\pm$3.69)	**22.27**($\pm$3.44)
	$\lambda = 0.6$	$\lambda = 0.7$	$\lambda = 0.8$	$\lambda = 0.9$	$\lambda = 1.0$	
100Hz	71.21($\pm$1.84)	81.21($\pm$1.84)	91.21($\pm$1.84)	101.21($\pm$1.84)	111.21($\pm$1.84)	
50Hz	41.80($\pm$2.10)	46.80($\pm$2.10)	51.80($\pm$2.10)	56.80($\pm$2.10)	61.80($\pm$2.10)	
16Hz	24.68($\pm$2.93)	26.28($\pm$2.93)	27.88($\pm$2.93)	29.48($\pm$2.93)	31.08($\pm$2.93)	
5Hz	31.54($\pm$2.78)	32.04($\pm$2.78)	32.54($\pm$2.78)	33.04($\pm$2.78)	33.54($\pm$2.78)	
Random	44.14($\pm$2.06)	48.74($\pm$2.05)	53.34($\pm$2.04)	57.94($\pm$2.03)	62.54($\pm$2.02)	
MDP-DS	27.91($\pm$3.92)	29.39($\pm$3.90)	30.81($\pm$3.92)	32.09($\pm$3.86)	33.23($\pm$3.88)	
RNN	24.81($\pm$3.62)	26.38($\pm$3.08)	27.47($\pm$3.35)	28.32($\pm$3.45)	29.87($\pm$2.78)	
DWFS-CVL	25.61($\pm$2.34)	27.01($\pm$2.44)	28.33($\pm$1.84)	29.00($\pm$2.20)	30.45($\pm$2.03)	
DWFS-SL	**23.56**($\pm$3.62)	**25.56**($\pm$3.81)	26.57($\pm$4.06)	27.74($\pm$3.39)	29.01($\pm$3.41)	
DWFS	24.17($\pm$3.70)	25.67($\pm$3.10)	**26.10**($\pm$3.26)	**27.38**($\pm$3.35)	**28.87**($\pm$3.12)	

Table 6.2: The Error-Cost Index with respect to λ on HASC dataset.

HARSD Dataset						
Method	$\lambda = 0.0$	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.3$	$\lambda = 0.4$	$\lambda = 0.5$
50Hz	18.41(±2.21)	28.41(±2.21)	38.41(±2.21)	48.41(±2.21)	58.41(±2.21)	68.41(±2.21)
16Hz	**18.35**(±2.32)	21.55(±2.32)	24.75(±2.32)	27.95(±2.32)	31.15(±2.32)	34.35(±2.32)
5Hz	21.67(±2.56)	22.67(±2.56)	23.67(±2.56)	24.67(±2.56)	25.67(±2.56)	26.67(±2.56)
2Hz	26.30(±2.82)	26.70(±2.82)	27.10(±2.82)	27.50(±2.82)	27.90(±2.82)	28.30(±2.82)
Random	20.16(±2.93)	24.12(±2.93)	28.09(±2.92)	32.05(±2.92)	36.01(±2.92)	39.97(±2.92)
MDP-DS	19.91(±2.16)	22.85(±2.10)	24.45(±2.06)	25.17(±2.18)	26.06(±2.52)	26.87(±2.48)
RNN	18.97(±2.41)	22.77(±2.22)	24.45(±2.51)	25.23(±2.71)	25.48(±3.06)	25.99(±2.94)
DWFS-CVL	18.86(±2.49)	**21.50**(±1.91)	23.34(±2.09)	25.15(±2.85)	25.66(±3.13)	26.56(±2.99)
DWFS-SL	19.33(±2.47)	22.60(±2.71)	23.44(±2.40)	24.99(±2.66)	25.86(±2.58)	26.44(±2.49)
DWFS	18.52(±1.76)	22.19(±2.45)	**23.29**(±1.93)	**23.41**(±2.10)	**24.33**(±2.95)	**25.37**(±2.26)
	$\lambda = 0.6$	$\lambda = 0.7$	$\lambda = 0.8$	$\lambda = 0.9$	$\lambda = 1.0$	
50Hz	78.41(±2.21)	88.41(±2.21)	98.41(±2.21)	108.41(±2.21)	118.41(±2.21)	
16Hz	37.55(±2.32)	40.75(±2.32)	43.95(±2.32)	47.15(±2.32)	50.35(±2.32)	
5Hz	27.67(±2.56)	28.67(±2.56)	29.67(±2.56)	30.67(±2.56)	31.67(±2.56)	
2Hz	28.70(±2.82)	29.10(±2.82)	29.50(±2.82)	29.90(±2.82)	30.30(±2.82)	
Random	43.93(±2.92)	47.90(±2.92)	51.86(±2.92)	55.82(±2.92)	59.78(±2.92)	
MDP-DS	27.57(±2.60)	28.44(±2.78)	29.27(±2.86)	30.07(±2.87)	31.19(±2.80)	
RNN	26.70(±2.76)	27.68(±2.59)	28.73(±3.37)	29.58(±2.61)	30.57(±2.75)	
DWFS-CVL	27.71(±3.09)	28.43(±3.07)	28.67(±2.75)	29.15(±3.05)	30.57(±3.01)	
DWFS-SL	27.01(±2.66)	27.85(±2.61)	28.64(±2.49)	29.05(±2.31)	30.00(±3.03)	
DWFS	**26.67**(±3.11)	**27.20**(±2.94)	**27.19**(±2.37)	**27.95**(±2.58)	**29.53**(±2.49)	

Table 6.3: The Error-Cost Index with respect to λ on HARSD dataset.

6.4 Conclusion & Discussion

In this chapter, we addressed an emergent problem of Human Activity Recognition (HAR) on mobile/wearable platforms, which is adaptively determining sampling frequencies to balance recognition accuracy and energy-efficiency. We formalized the problem as minimizing an objective function regarding classification error and energy cost, by finding an optimal classification model and dynamically appropriate sampling rates. We proposed Datum-Wise Frequency Selection (DWFS) to solve the problem via a continuous state Markov Decision Process (MDP). The MDP learns a policy function that selects the best frequency for sampling an incoming data entity by exploiting the information of previously sampled instances. We proposed an alternate learning method, where the parameters of the classification model and the policy function are mutually enhanced. We

DSA Dataset						
Method	$\lambda = 0.0$	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.3$	$\lambda = 0.4$	$\lambda = 0.5$
25Hz	**22.67**($\pm$3.63)	32.67($\pm$3.63)	42.67($\pm$3.63)	52.67($\pm$3.63)	62.67($\pm$3.63)	72.67($\pm$3.63)
10Hz	23.82($\pm$4.33)	27.82($\pm$4.33)	31.82($\pm$4.33)	35.82($\pm$4.33)	39.82($\pm$4.33)	43.82($\pm$4.33)
5Hz	27.51($\pm$3.39)	29.51($\pm$3.39)	31.51($\pm$3.39)	33.51($\pm$3.39)	35.51($\pm$3.39)	37.51($\pm$3.39)
2Hz	36.43($\pm$3.79)	37.23($\pm$3.79)	38.03($\pm$3.79)	38.83($\pm$3.79)	39.63($\pm$3.79)	40.43($\pm$3.79)
Random	30.26($\pm$3.46)	34.55($\pm$3.46)	38.85($\pm$3.46)	43.14($\pm$3.46)	47.43($\pm$3.46)	51.72($\pm$3.46)
MDP-DS	24.78($\pm$3.85)	29.19($\pm$3.86)	31.72($\pm$3.94)	33.97($\pm$3.72)	35.98($\pm$3.62)	37.91($\pm$3.56)
RNN	23.26($\pm$3.60)	29.04($\pm$3.81)	31.15($\pm$3.61)	32.53($\pm$3.97)	33.93($\pm$3.63)	35.29($\pm$3.89)
DWFS-CVL	26.04($\pm$3.84)	29.94($\pm$3.21)	32.16($\pm$2.97)	34.05($\pm$3.06)	35.89($\pm$3.15)	37.35($\pm$3.42)
DWFS-SL	23.53($\pm$3.88)	27.66($\pm$3.69)	29.78($\pm$3.63)	31.74($\pm$3.38)	33.45($\pm$3.52)	34.89($\pm$3.39)
DWFS	23.50($\pm$3.77)	**27.30**($\pm$3.25)	**29.58**($\pm$3.06)	**31.15**($\pm$3.29)	**33.29**($\pm$2.78)	**34.87**($\pm$2.90)
	$\lambda = 0.6$	$\lambda = 0.7$	$\lambda = 0.8$	$\lambda = 0.9$	$\lambda = 1.0$	
25Hz	82.67($\pm$3.63)	92.67($\pm$3.63)	102.67($\pm$3.63)	112.67($\pm$3.63)	122.67($\pm$3.63)	
10Hz	47.82($\pm$4.33)	51.82($\pm$4.33)	55.82($\pm$4.33)	59.82($\pm$4.33)	63.82($\pm$4.33)	
5Hz	39.51($\pm$3.39)	41.51($\pm$3.39)	43.51($\pm$3.39)	45.51($\pm$3.39)	47.51($\pm$3.39)	
2Hz	41.23($\pm$3.79)	42.03($\pm$3.79)	42.83($\pm$3.79)	43.63($\pm$3.79)	44.43($\pm$3.79)	
Random	56.01($\pm$3.46)	60.30($\pm$3.46)	64.60($\pm$3.46)	68.89($\pm$3.46)	73.18($\pm$3.46)	
MDP-DS	39.79($\pm$3.56)	41.46($\pm$3.51)	43.00($\pm$3.51)	44.54($\pm$3.45)	45.86($\pm$3.48)	
RNN	36.58($\pm$3.45)	38.27($\pm$3.99)	39.40($\pm$3.40)	40.53($\pm$3.37)	42.19($\pm$4.00)	
DWFS-CVL	38.79($\pm$3.29)	40.25($\pm$3.50)	41.82($\pm$3.37)	42.70($\pm$3.29)	44.21($\pm$3.49)	
DWFS-SL	36.54($\pm$3.42)	37.84($\pm$3.59)	39.04($\pm$3.46)	40.18($\pm$3.50)	41.64($\pm$3.45)	
DWFS	**36.14**($\pm$3.23)	**37.40**($\pm$3.28)	**38.96**($\pm$3.26)	**39.92**($\pm$3.44)	**41.41**($\pm$3.30)	

Table 6.4: The Error-Cost Index with respect to λ on DSA dataset.

	DWFS vs. Random			DWFS vs. MDP-DS			DWFS vs. RNN		
Dataset	R^+	R^-	p-value	R^+	R^-	p-value	R^+	R^-	p-value
HASC	**0.0**	1540.0	0.000000	**2.0**	1538.0	0.000000	**288.0**	1252.0	0.000054
HARSD	**9.0**	1531.0	0.000000	**110.0**	1430.0	0.000000	**398.0**	1142.0	0.001828
DSA	**0.0**	1540.0	0.000000	**1.0**	1539.0	0.000000	**166.0**	1374.0	0.000000
	DWFS vs. MDP-CVL			DWFS vs. DWFS-SL					
Dataset	R^+	R^-	p-value	R^+	R^-	p-value			
HASC	**112.0**	1428.0	0.000000	744.5	**740.5**	0.986261			
HARSD	**323.0**	1162.0	0.000304	**212.0**	1273.0	0.000005			
DSA	**0.0**	1540.0	0.000000	**437.0**	1103.0	0.005270			

Table 6.5: The Wilcoxon test to compare the Error-Cost Indexes of DWFS, DWFS-CVL, DWFS-SL, RNN, MDP-DS, and Random regarding R^+, R^-, and p-values.

evaluated the performance of DWFS on 3 real-world HAR datasets, and the results show that DWFS statistically outperforms the state-of-the-arts regarding a combined measure-

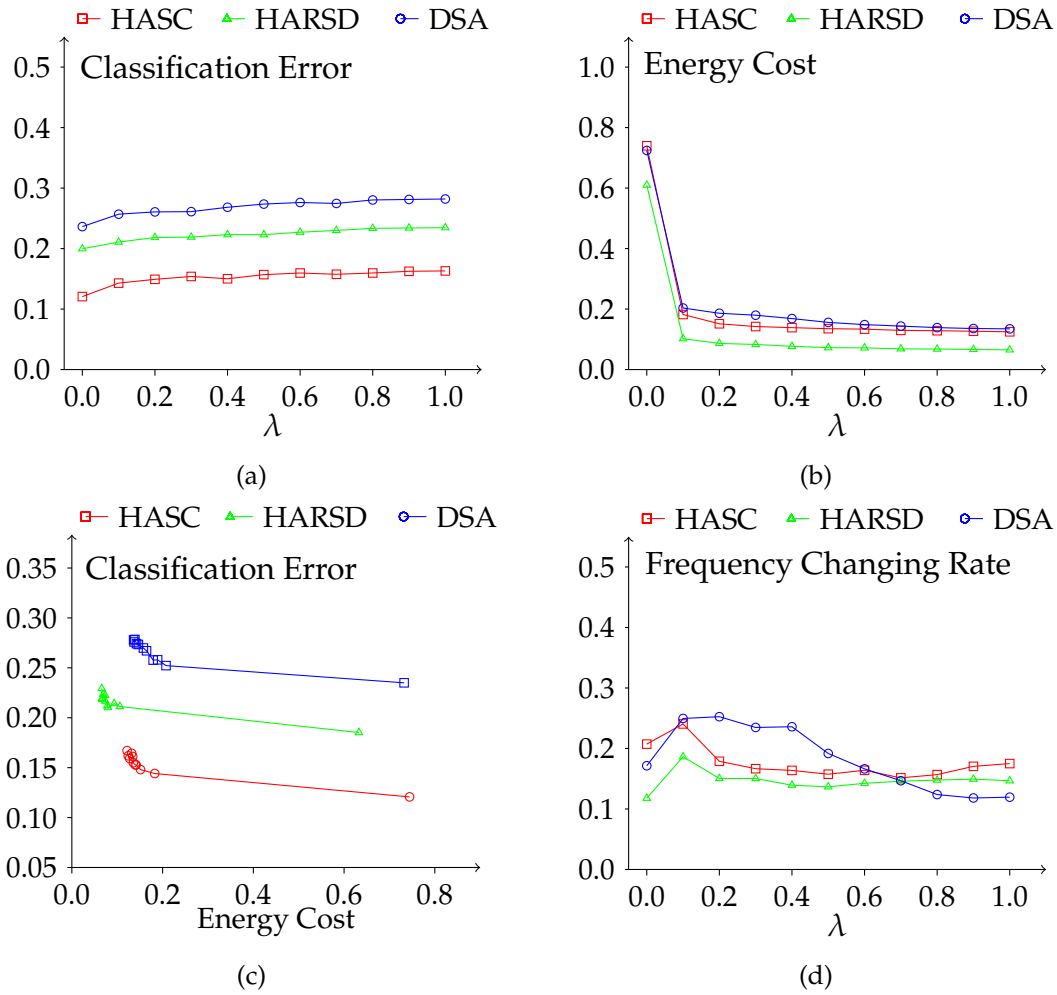


Figure 6.2: The 4 relationships on HASC, HARSD, and DSA datasets. (a) Classification error with respect to λ. (b) Energy cost with respect to λ. (c) Classification error with respect to energy cost. (d) Frequency changing rate with respect to λ.

ment of classification error and energy cost.

Compared with practical methods which use heuristically designed strategies for saving energy, DWFS learns the strategies from data. Before deployment on real devices, the model of DWFS should be trained with proper settings of two important parameters. The first parameter is the energy costs c_f regarding frequencies, which are dependent on devices. Given a device, real costs should be examined with respect to the supported frequencies. Then, c_f can be configured based on these real costs. The second parameter is λ, i.e., the weight balancing classification error and cost. λ should be predefined based on efficiency requirements. Tunning λ is a tedious task, since it need repeated training

and testing until the desired trade-off performance is achieved. Accordingly, a future study of automatic parameter determining can be conducted to enhance the practicality of DWFS on real devices.

Chapter 7
Conclusion & Future Research

In this chapter, we conclude our key findings and highlight our main research outcomes. We also include the future directions regarding the topic of accurate and efficient Human Activity Recognition.

7.1 Summary of Contributions

In this thesis, we focused on improving the performance of Human Activity Recognition (HAR) on both accuracy and efficiency. We first devised accurate HAR methods, then we improved the energy-efficiency of HAR while preserving high recognition accuracy. In particular, we studied 3 problems: 1) improving the accuracy of recognizing the current activity during activity transitions; 2) improving the accuracy of predicting complex activities from ongoing observations; 3) improving energy-efficiency of HAR while preserving prominent accuracy.

Chapter 3 studies the problem of accurately recognizing the current activity during activity transitions. We proposed Weighted Min-max Activity Recognition Model (WMARM), which identifies the current activity by optimally partitioning an observed window of time series matching the activities taking place. WMARM considers weights on the partitioned segments to obtain reliable recognition accuracy. WMARM can also effectively handle the time series containing an arbitrary number of transitions without any prior knowledge about the number of transitions. Instead of exhaustively searching the optimal solution of WMARM in exponential space, we proposed an efficient dynamic programming algorithm that computes the model in $\mathcal{O}(n^2)$ time complexity, where n is the length of the time series. Moreover, we presented an efficient implementation of

WMARM that the computational cost can be further reduced. Extensive experiments on 5 real HAR datasets have demonstrated the superior performance of WMARM on handling time series with one or more activity transitions. The results show about 10%-30% improvement on the accuracy of current activity recognition compared with state-of-the-art methods. The experiment on smartphones shows the prominent computational efficiency of WMARM.

Chapter 4 studies the problem of accurately predicting complex activities from ongoing multivariate time series (MTS). We proposed Simultaneous Complex Activities Recognition and Action Sequence Discovering (SimRAD) which predicts the complex activity over time by finding a sequence of multivariate actions from sensor MTS data using a Deep Neural Network. SimRAD learns two probabilistic models for inferring complex activities and action sequences, where the estimations of the two models are conditionally dependent on each other. SimRAD alternately predicts the complex activity and the action sequence with the two models, thus the predictions can be mutually updated until the completion of the complex activity. We evaluated SimRAD on a real-world complex activity dataset of a rich amount of sensor data. The results demonstrate that SimRAD outperforms state-of-the-art methods by average 7.2% in prediction accuracy with very high confidence.

Chapter 5 studies the problem of improving energy-efficiency of HAR by reducing computational cost on activity inference. We attempted to improve the subsequence ensemble models which use multiple feature representations based on subsequences of time series that cause heavy computations. We formalized a dynamic subsequence selection problem that minimizes the expected computations while persevering a high recognition accuracy. To solve this problem, we proposed Markov Dynamic Subsequence Ensemble (MDSE), an algorithm for the selection of the subsequences with a Markov Decision Process (MDP). The MDP learns a policy for choosing the best subsequence given the state of prediction. Regarding the expected ensemble size of MDSE, we derived an upper bound so that the energy consumption caused by the computations of the proposed method is guaranteed. We conducted extensive experiments on 6 real HAR datasets to evaluate the effectiveness of MDSE. Compared with the state-of-the-art methods, MDSE reduces 70.8% computational cost which is 3.42 times more energy efficient, and achieves

a comparably high accuracy.

Chapter 6 studies the problem of improving energy-efficiency of HAR by reducing sensing cost on sampling incoming data points. We attempted to dynamically select sampling frequencies by directly exploiting sensor data. We formalized a problem of minimizing an objective function regarding classification error and sensing cost, by finding an optimal classification model and dynamically appropriate sampling rates. To address this problem, we proposed Datum-Wise Frequency Selection (DWFS) which utilizes a continuous state MDP. The MDP learns a policy function that selects the best frequency for sampling an incoming data entity by utilizing previously sampled data. We proposed an alternate learning scheme, where the parameters of the classification model and the policy function are mutually enhanced. We evaluated the performance of DWFS on 3 real-world HAR datasets, and the results show that DWFS statistically outperforms the state-of-the-art methods regarding a combined measurement of classification error and energy cost.

7.2 Future Research

There are many future research directions that can be carried out regarding the topic of accurate and efficient HAR. We briefly discuss three attractive directions as follows:

- The new generation of smartphones is equipped with a variety of advanced sensors, such as accelerometer, gyroscope, barometer, etc. Existing studies have shown that many sensors can be used for HAR, and each of the sensors is advanced on recognizing a number of activities with a level of energy consumption. For example, an accelerometer can be used to accurately recognize running, walking, stationary with medium energy consumption, a barometer can be used to accurately recognize 'up climbing' and 'down climbing' with low energy consumption, and a GPS can be used to accurately recognize driving, running, and stationary with high energy consumption. Suppose there is a set of activities to recognize, a future study is to find a model that can manage the usage of the available sensors to maximize the accuracy on all the activities and minimize the total energy consumption, or to maximize the accuracy given a certain energy budget.

- Deep Neural Network (DNN) has delivered prominent performance in recognition of natural images, natural speech, natural language corpora, etc, and also has demonstrated great potential in HAR. However, the complex structure of DNN limits its applications in real life. As one forward computing of DNN involves several matrix multiplications regarding its depth, the inference computations of DNN can be a lot greater than the linear models. Real-time HAR requires computational-efficient models which can perform fast inference, and also save energy cost on portable devices. Therefore, a future problem is to improve the computational-efficiency of DNN. This problem can be potentially addressed by reducing the depth and seeking sparsity of DNN layers. However, a further investigation is required for maximizing the DNN accuracy while minimizing the number of computations.

- Note that state-of-the-art HAR methods are based on supervised learning, thus data annotation is an essential step for preparing a training dataset. However, data annotation requires significant human effort, especially for the data of inertial sensor where the time series patterns are hard to be manually located and distinguished. Therefore, a future problem is to improve training efficiency of HAR methods by reducing the annotation effort. This problem can be potentially addressed by semi-supervised or weakly supervised learning schemes, where the training data is partially annotated. However, a further investigation is required for maximizing the model accuracy while minimizing the required number of annotated data.

Bibliography

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4277–4280, 2012.

[2] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[3] Hyrum S Anderson, Nathan Parrish, Kristi Tsukida, and Maya R Gupta. Reliable early classification of time series. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2073–2076, 2012.

[4] Ian Anderson, Julie Maitland, Scott Sherwood, Louise Barkhuus, Matthew Chalmers, Malcolm Hall, Barry Brown, and Henk Muller. Shakra: tracking and sharing daily activity levels with unaugmented mobile phones. *Mobile Networks and Applications*, 12(2-3):185–199, 2007.

[5] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2013.

[6] Louis Atallah, Benny Lo, Rachel King, and Guang-Zhong Yang. Sensor positioning for activity recognition using wearable accelerometers. *IEEE Transactions on Biomedical Circuits and Systems*, 5(4):320–329, 2011.

[7] Marc Bachlin, Daniel Roggen, Gerhard Troster, Meir Plotnik, Noit Inbar, Inbal Meidan, Talia Herman, Marina Brozgol, Eliya Shaviv, Nir Giladi, et al. Potentials of en-

hanced context awareness in wearable assistants for parkinson's disease patients with the freezing of gait syndrome. In *Proceedings of International Symposium on Wearable Computers (ISWC)*, pages 123–130, 2009.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Oresti Banos, Juan-Manuel Galvez, Miguel Damas, Alberto Guillen, Luis-Javier Herrera, Hector Pomares, Ignacio Rojas, Claudia Villalonga, Choong Seon Hong, and Sungyoung Lee. *Multiwindow Fusion for Wearable Activity Recognition*, pages 290–297. Springer, 2015.

[10] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In *Proceedings of International Conference on Pervasive Computing (PerCom)*, pages 1–17, 2004.

[11] Billur Barshan and Murat Cihan Yüksek. Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. *The Computer Journal*, 28:bxt075, 2014.

[12] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[13] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[14] Richard Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, 1961.

[15] Preeti Bhargava, Nick Gramsky, and Ashok Agrawala. Senseme: a system for continuous, on-device, and multi-dimensional context and activity recognition. In *Proceedings of International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, pages 40–49, 2014.

[16] Sourav Bhattacharya and Nicholas D. Lane. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *Proceedings of Conference on Embedded Networked Sensor Systems (SenSys)*, pages 176–189, 2016.

[17] Andreas Bloch, Robert Erdin, Sonja Meyer, Thomas Keller, and Alexandre de Spindler. Battery-efficient transportation mode detection on mobile devices. In *Proceedings of International Conference on Mobile Data Management (MDM)*, volume 1, pages 185–190, 2015.

[18] Aaron F Bobick. Movement, activity and action: the role of knowledge in the perception of motion. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 352(1358):1257–1265, 1997.

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

[20] Pierluigi Casale, Oriol Pujol, and Petia Radeva. Personalization and user verification in wearable systems using biometric walking patterns. *Personal Ubiquitous Comput.*, 16(5):563–580, 2012.

[21] M Bishop Christopher. *Pattern Recognition and Machine Learning*. Springer, 2006.

[22] Asma Dachraoui, Alexis Bondu, and Antoine Cornuéjols. Early classification of time series as a non myopic sequential decision making problem. In *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 433–447, 2015.

[23] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[24] Xiuyi Fan, Huiguo Zhang, Cyril Leung, and Chunyan Miao. Comparative study of machine learning algorithms for activity recognition with data sequence in home-like environment. In *Proceedings of International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 168–173, 2016.

[25] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337–407, 2000.

[26] Hans W Gellersen, Albercht Schmidt, and Michael Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications*, 7(5):341–351, 2002.

[27] Mohamed F Ghalwash and Zoran Obradovic. Early classification of multivariate temporal observations by extraction of interpretable shapelets. *BMC bioinformatics*, 13(1):195, 2012.

[28] Dawud Gordon, Jurgen Czerny, Takashi Miyaki, and Michael Beigl. Energy-efficient activity recognition using prediction. In *Proceedings of International Symposium on Wearable Computers (ISWC)*, pages 29–36, 2012.

[29] Geoffrey J Gordon. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*, pages 261–268. Elsevier, 1995.

[30] Tao Gu, Liang Wang, Zhanqing Wu, Xianping Tao, and Jian Lu. A pattern mining approach to sensor-based human activity recognition. *Transactions on Knowledge and Data Engineering (TKDE)*, 23(9):1359–1372, 2011.

[31] Xinze Guan, Raviv Raich, and Weng-Keen Wong. Efficient multi-instance learning for activity recognition from time series data using an auto-regressive hidden markov model. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 2330–2339, 2016.

[32] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *SigKDD Explorations Newsletter*, 11(1):10–18, 2009.

[33] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of International Joint Conference on Aritficial Intelligence (IJCAI)*, pages 1533–1540, 2016.

[34] Nils Y. Hammerla and Thomas Plötz. Let's (not) stick together: Pairwise similarity biases cross-validation in activity recognition. In *Proceedings of International Conference on Ubiquitous Computing (UbiComp)*, pages 1041–1051, 2015.

[35] Yi He, Ye Li, and Shu-Di Bao. Fall detection by built-in tri-accelerometer of smartphone. In *Proceedings of International Conference on Biomedical and Health Informatics (BHI)*, pages 184–187, 2012.

[36] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of Conference on Embedded Networked Sensor Systems (SenSys)*, page 13, 2013.

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

[38] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine*, 29(6):82–97, 2012.

[39] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[40] Frank Höppner. Discovery of temporal patterns. In *Proceedings of Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 192–203, 2001.

[41] Brad Jackson, Jeffrey D Scargle, David Barnes, Sundararajan Arabhi, Alina Alt, Peter Gioumousis, Elyus Gwin, Paungkaew Sangtrakulcharoen, Linda Tan, and Tun Tao Tsai. An algorithm for optimal partitioning of data on an interval. *Signal Processing Letters*, 12(2):105–108, 2005.

[42] Seungwoo Kang, Jinwon Lee, Hyukjae Jang, Hyonik Lee, Youngki Lee, Souneil Park, Taiwoo Park, and Junehwa Song. Seemon: Scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *Proceedings of International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 267–280, 2008.

[43] Nobuo Kawaguchi, Ying Yang, Tianhui Yang, Nobuhiro Ogawa, Yohei Iwasaki, Katsuhiko Kaji, Tsutomu Terada, Kazuya Murao, Sozo Inoue, Yoshihiro Kawahara, Yasuyuki Sumi, and Nobuhiko Nishio. Hasc2011corpus: Towards the common ground of human activity recognition. In *Proceedings of International Conference on Ubiquitous Computing (UbiComp)*, pages 571–572, 2011.

[44] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting time series: A survey and novel approach. *Data mining in Time Series Databases*, 57:1–22, 2004.

[45] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Y Lee, and Tae-Seong Kim. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *Transactions on Information Technology in Biomedicine*, 14(5):1166–1172, 2010.

[46] Aftab Khan, Nils Hammerla, Sebastian Mellor, and Thomas Plötz. Optimising sampling rates for accelerometer-based human activity recognition. *Pattern Recognition Letters*, 73:33–40, 2016.

[47] Rebecca Killick, Paul Fearnhead, and IA Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[48] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of Information Processing in Sensor Networks (IPSN)*, 2006.

[49] Andreas Krause, Matthias Ihmig, Edward Rankin, Derek Leong, Smriti Gupta, Daniel Siewiorek, Asim Smailagic, Michael Deisher, and Uttam Sengupta. Trading off prediction accuracy and power consumption for context-aware wearable computing. In *Proceedings of International Symposium on Wearable Computers (ISWC)*, pages 20–26, 2005.

[50] Narayanan C Krishnan and Diane J Cook. Activity recognition on streaming sensor data. *Pervasive and Mobile Computing*, 10:138–154, 2014.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[52] John Krumm and Eric Horvitz. Locadio: Inferring motion and location from wi-fi signal strengths. In *Proceedings of International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, pages 4–13, 2004.

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

[54] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[55] Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, volume 5, pages 766–772, 2005.

[56] Bruce Levin. A representation for multinomial cumulative distribution functions. *The Annals of Statistics*, 9(5):1123–1126, 1981.

[57] Céline Levy-leduc and Zaïd Harchaoui. Catching change-points with lasso. In *Proceedings of Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 617–624, 2008.

[58] Kang Li and Yu Fu. Prediction of human activity by discovering temporal sequence patterns. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(8):1644–1657, 2014.

[59] Kang Li, Sheng Li, and Yun Fu. Early classification of ongoing observation. In *Proceedings of International Conference on Data Mining (ICDM)*, pages 310–319, 2014.

[60] Lin Liao, Dieter Fox, and Henry Kautz. Location-based activity recognition. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Proceedings of Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 787–794, 2005.

[61] Yu-Feng Lin, Hsuan-Hsu Chen, Vincent S Tseng, and Jian Pei. Reliable early classification on multivariate time series with numerical and categorical attributes. In *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 199–211, 2015.

[62] Li Liu, Li Cheng, Ye Liu, Yongpo Jia, and David S Rosenblum. Recognizing complex activities by a probabilistic interval-based model. In *Proceedings of AAAI Conference on Artifical Intelligence (AAAI)*, volume 30, pages 1266–1272, 2016.

[63] Ye Liu, Liqiang Nie, Lei Han, Luming Zhang, and David S Rosenblum. Action2activity: Recognizing complex activities from sensor data. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1617–1623, 2015.

[64] Jeffrey W Lockhart, Tony Pulickal, and Gary M Weiss. Applications of mobile activity recognition. In *Proceedings of International Conference on Ubiquitous Computing (UbiComp)*, pages 1054–1058, 2012.

[65] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of Conference on Embedded Networked Sensor Systems (SenSys)*, pages 71–84, 2010.

[66] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1942–1950, 2016.

[67] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126, 2006.

[68] Le T. Nguyen, Ming Zeng, Patrick Tague, and Joy Zhang. I did not smoke 100 cigarettes today!: Avoiding false positives in real-world activity recognition. In *Proceedings of International Conference on Ubiquitous Computing (UbiComp)*, pages 1053–1063, 2015.

[69] Le T Nguyen, Ming Zeng, Patrick Tague, and Joy Zhang. Recognizing new activities with limited training data. In *Proceedings of International Symposium on Wearable Computers (ISWC)*, pages 67–74, 2015.

[70] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Passos Alexandre, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[71] Thomas Plotz, Nils Y Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 1729–1734, 2011.

[72] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[73] Xin Qi, Matthew Keally, Gang Zhou, Yantao Li, and Zhen Ren. Adasense: Adapting sampling rates for activity recognition in body sensor networks. In *Proceedings of Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 163–172, 2013.

[74] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[75] Cliff Randell and Henk Muller. Context awareness by analysing accelerometer data. In *Proceedings of International Symposium on Wearable Computers (ISWC)*, pages 175–176, 2000.

[76] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity recognition from accelerometer data. In *Proceedings of Conference on Innovative Applications of Artificial Intelligence (IAAI)*, volume 5, pages 1541–1546, 2005.

[77] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *Transactions on Sensor Networks (TOSN)*, 6(2):13, 2010.

[78] Ramona Rednic, Elena Gaura, John Kemp, and James Brusey. Fielded autonomous posture classification systems: design and realistic evaluation. In *Proceedings of ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 635–640, 2013.

[79] Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *Proceedings of International Symposium on Wearable Computers (ISWC)*, pages 108–109, 2012.

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

[81] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriage, Hesam Sagha, Hamidreza Bayati, Marco Creatura, and Jose del R. Millan. Collecting complex activity datasets in highly rich networked sensor environments. In *Proceedings of International Conference on Networked Sensing Systems (INSS)*, pages 233–240, 2010.

[82] Guy Rosman, Mikhail Volkov, Dan Feldman, John W Fisher III, and Daniela Rus. Coresets for k-segmentation of streaming data. In *Proceedings of Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 559–567, 2014.

[83] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[84] Michael S Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 1036–1043, 2011.

[85] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proceedings of Interspeech*, pages 338–342, 2014.

[86] Kartik Sankaran, Minhui Zhu, Xiang Fa Guo, Akkihebbal L Ananda, Mun Choon Chan, and Li-Shiuan Peh. Using mobile phone barometer for low-power transportation context detection. In *Proceedings of Conference on Embedded Networked Sensor Systems (SenSys)*, pages 191–205, 2014.

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

[88] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

[89] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[90] Maja Stikic, Diane Larlus, Sandra Ebert, and Bernt Schiele. Weakly supervised recognition of daily life activities with wearable sensors. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(12):2521–2537, 2011.

[91] Vincent S Tseng, Chun-Hao Chen, Pai-Chieh Huang, and Tzung-Pei Hong. Cluster-based genetic segmentation of time series with dwt. *Pattern Recognition Letters*, 30(13):1190–1197, 2009.

[92] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2001.

[93] Liang Wang, Tao Gu, Xianping Tao, and Jian Lu. A hierarchical approach to real-time activity recognition in body sensor networks. *Pervasive and Mobile Computing*, 2012.

[94] Yi Wang, Bhaskar Krishnamachari, Qing Zhao, and Murali Annavaram. Markov-optimal sensing policy for user state estimation in mobile devices. In *Proceedings of International Conference on Information Processing in Sensor Networks (IPSN)*, pages 268–278, 2010.

[95] Yi Wang, Jialiu Lin, Murali Annavaram, Quinn A Jacobson, Jason Hong, Bhaskar Krishnamachari, and Norman Sadeh. A framework of energy efficient mobile sens-

ing for automatic user state recognition. In *Proceedings of International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 179–192, 2009.

[96] Zhengzheng Xing, Jian Pei, and S Yu Philip. Early prediction on time series: A nearest neighbor approach. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1297–1302, 2009.

[97] Zhengzheng Xing, Jian Pei, Philip S Yu, and Ke Wang. Extracting interpretable features for early classification on time series. In *Proceedings of SIAM International Conference on Data Mining (SDM)*, pages 247–258, 2011.

[98] Zhixian Yan, Vigneshwaran Subbaraju, Dipanjan Chakraborty, Archan Misra, and Karl Aberer. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *Proceedings of International Symposium on Wearable Computers (ISWC)*, pages 17–24, 2012.

[99] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3995–4001, 2015.

[100] Qiang Yang. Activity recognition: Linking low-level sensors to high-level intelligence. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 20–25, 2009.

[101] Jie Yin, Qiang Yang, and Jeffrey Junfeng Pan. Sensor-based abnormal human-activity detection. *Transactions on Knowledge and Data Engineering (TKDE)*, 20(8), 2008.

[102] Özgür Yürür, Chi Harold Liu, Charith Perera, Min Chen, Xue Liu, and Wilfrido Moreno. Energy-efficient and context-aware smartphone sensor employment. *Transactions on Vehicular Technology*, 64(9):4230–4244, 2015.

[103] Piero Zappi, Clemens Lombriser, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Tröster. Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. In *Wireless Sensor Networks*, pages 17–33. Springer, 2008.

[104] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Juyong Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *Proceedings of International Conference on Mobile Computing, Applications and Services (MobiCASE)*, pages 197–205, 2014.

[105] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis*, pages 105–114, 2010.

[106] Yongmian Zhang, Yifan Zhang, Eran Swears, Natalia Larios, Ziheng Wang, and Qiang Ji. Modeling temporal interactions with interval temporal bayesian networks for complex activity recognition. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(10):2468–2483, 2013.

[107] Yonglei Zheng, Weng-Keen Wong, Xinze Guan, and Stewart Trost. Physical activity recognition from accelerometer data using a multi-scale ensemble method. In *Proceedings of Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pages 1575–1581, 2013.

[108] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *Proceedings of International Conference on Ubiquitous Computing (UbiComp)*, pages 312–321, 2008.

[109] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.