AN ABSTRACT OF THE THESIS OF

Michael M. Anderson for the degree of Master of Science in Computer Science presented on June 13, 2013.

Title: Physical Activity Recognition of Free-Living Data Using Change-Point Detection Algorithms and Hidden Markov Models

Abstract approved: _____

Weng-Keen Wong

Physical activity recognition using accelerometer data is a rapidly emerging field with many real-world applications. Much of the previous work in this area has assumed that the accelerometer data has already been segmented into pure activities, and the activity recognition task has been to classify these segments. In reality, activity recognition would need to be applied to "free-living" data, which is collected over a long, continuous time period and would consist of a mixture of activities. In this thesis, we explore two approaches for segmenting realistic free-living time series data. In the first approach, we apply a top-down strategy in which we segment free-living data using change-point detection algorithms and then classify the resulting segments using supervised learning techniques. In the second approach, we employ a bottom-up strategy in which we split the time series into small fixed-length windows, classify these windows, and then smooth the predictions using an HMM. Our results clearly show that the bottom-up approach is far superior to the top-down approach in both accuracy and timeliness of detection.

Physical Activity Recognition of Free-Living Data Using Change-Point Detection Algorithms and Hidden Markov Models

by

Michael M. Anderson

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 13, 2013
Commencement June 2013

Master of Science thesis of Michael M. Anderson presented on June 13, 2013.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Michael M. Anderson, Author

ACKNOWLEDGEMENTS

I would like to acknowledge and thank my advisor, Dr. Weng-Keen Wong, for providing me with a clear and overarching vision of this project at all stages of its development, as well as for his experience and helpfulness with all of the rough patches, sticking points, and unexpected problems that invariably accompany an endeavor of this magnitude. I would like to thank Dr. Stewart Trost for providing his expertise and for his generous assistance in obtaining the datasets that we used for our experiments. I would also like to thank the members of my committee: Dr. Prasad Tadepalli, Dr. Raviv Raich, and Dr. Hector Vergara, for giving their time to help supervise the final stages of this project. Finally, thanks go to all of my family and friends who encouraged and stood by me along the way. They are too numerous to name but they know who they are.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

Chapter 1: Introduction

One of the general goals of artificial intelligence is to build computing devices that are "context-aware", that act as more than just passive number-crunching machines that receive input data through very restrictive and wholly human-operated channels such as a keyboard or mouse. Context-aware devices are capable of using sensor data to understand the environment that they are situated in, such as the locations of nearby objects and how the objects are moving [1]. One subfield of context-aware computing that has been receiving considerable attention in recent years is activity detection. The goal of activity detection is to build computer plus sensor systems that are able to determine what activity a human subject is performing at any given moment.

Such systems have a variety of real-world applications. Research is exploring the feasibility of using both wearable and non-wearable sensor systems to monitor the health of elderly patients that have or are at risk of developing degenerative physical and mental diseases [10]. The goal is to build sensor-based monitoring systems that can aid doctors and family members in tracking the decline of patients over time. Also, detection of an abnormal activity may indicate that a senior is undergoing a serious medical event such as a heart attack or slip-and-fall [39]. Another application of activity detection is to track the energy expenditure of subjects as they go through the course of their day. The traditional method of performing such tracking is with self-reporting by the subject of their activities. Wearable sensors offer an alternative approach that is not susceptible to misreporting due to bias, poor memory, or other confounding factors that a human reporter introduces into the system. One approach is to use sensor data to estimate the vigorousness or metabolic equivalent (MET) of an activity and calculate energy expenditure directly [31], while another is to attempt to predict the type of activity performed, and calculate energy expenditure using knowledge of how vigorous that activity is generally [37].

Activity detection generally assumes that sensor data will be represented as a time series, and that at any given moment in the time series the subject is performing one and only one type of activity. Thus the time series is thought of as being partitioned

into a number of non-overlapping intervals (*windows*), which are delimited by moments in time when the subject stopped performing one activity and started performing another. Previous work has treated activity detection as an offline problem, and has rarely considered performance metrics other than accuracy. In this work we are interested in the feasibility of partitioning and classifying a time series on free-living data in real time. In addition to accuracy, we will also evaluate our algorithms in terms of the amount of time required to detect that an activity change has occurred.

We used change-point detection to partition time series data into activity windows for classification. Change-point detection is a field of statistics popular in control theory and other similar applications. We call this our top-down approach, because this method takes as input an initial time series and partitions it into smaller pieces using change-point detection. As an alternate approach we used the well-known technique of partitioning the time series into small fixed-length, non-overlapping windows, predicting the activity type of each window using a base classifier, treating that prediction as the observable state of an HMM, and then finally solving the HMM for its hidden states. We call this our bottom-up approach, because we begin with small windows of fixed-length, and use an HMM to smooth windows together into larger activity intervals.

We found that the bottom-up approach outperformed the top-down approach in terms of both accuracy and detection time, for all of the datasets and base classifiers that we tried.

Chapter 2: Related Work

As mentioned in the previous chapter, sensor systems may consist of environmental or wearable devices. Some examples of environmental sensors that activity detection researchers have used to gather data are microphones [10], weight detection panels [25], cameras [9], and water usage detectors [10]. Researchers will generally place environmental sensors inside of a house, have subjects live in the house for a period of time, and attempt to predict for activity types such as cooking and watching TV.

Various wearable devices have been tried as well, such as RFID gloves [12], [25], but the most popular wearable for activity detection purposes is the accelerometer. Besides being inexpensive, accelerometers tend to be small and lightweight, and so are fairly unobtrusive and user-friendly. Accelerometers also gather data at a high frequency, and as such may be used to collect a sizeable amount of data in a relatively short amount of time.

Whether or not an accelerometer will yield data that is discriminative for a set of activity types depends partially on where the accelerometer is worn on a subject's body. For example, an accelerometer worn on the ankle will be more discriminative for the activity of cycling than it would be if it was worn on the hip, and different types of arm movements will likely be discriminated only by an accelerometer worn on the arm. For this reason some researchers have opted to use multiple accelerometer systems to capture movement information from different parts of the body [4], [8]. However, this approach can be cumbersome for the wearer, so a single accelerometer is preferred when it is reasonable to assume that it will be discriminative for the relevant set of activities. Recent research has noticed that real subjects are likely to carry smartphones with built-in accelerometers, and has explored the possibility of collecting data from those accelerometers for activity detection purposes [4], [7], [14], [21].

Activity sensor data tends to be noisy and not amenable to a deterministic or rule-based analysis, so activity types are typically modeled probabilistically, and activity detection is usually formulated as a supervised learning problem. The various common supervised learning algorithms are all familiar to the activity detection literature, though

neural networks are especially popular, such as in [2], [29], [31]. More complicated modeling approaches have also been tried, such as plurality voting with bagged, boosted, and stacked classifiers [22]; conditional random fields [6], [12], [38], [40]; and HMMs [12], [15], [20], [40].

In the past few years researchers have begun to recognize the need to test on realistic free-living data [12], [14], [32], [40]. The time required to detect that a change in activity has occurred was considered in [11], [28], but in the context of the very different problem of video activity recognition. Our work is one of the first to consider the feasibility of performing accelerometer activity recognition in real time by using both accuracy and detection time as performance metrics.

Chapter 3: Methodology

3.1 Datasets

For our experiments we were interested in testing our algorithms on real-world free-living data. In the past researchers have gathered activity data under unrealistic laboratory conditions, and unfortunately there are not many labeled free-living datasets publicly available. We tested our algorithms on two datasets that were available to us. The first was synthetically generated by concatenating lab visit data together, while the second was real-world free-living data.

3.1.1 Synthetic Data (OSU Hip)

We generated a synthetic, free-living dataset using physical activity data collected by Stewart Trost of the Nutrition and Exercise Sciences department of Oregon State University [37], [41]. This dataset consisted of 91 time series collected over a 2-week period in a laboratory environment, from 50 children between the ages of 5 and 15. Subjects performed 12 different types of activities over two separate lab visits, with a hip-worn ActiGraph GT1M accelerometer that collected triaxial acceleration data at a frequency of 30Hz (see top graph of Figure 3.1).

Data was collected from two separate visits to the lab, where the subjects performed 6 activities per visit. Each subject performed the 12 activities in the same order. In the version of the dataset available to us, we had all 12 activities of 41 of the subjects, only the first 6 activities of an additional 5 subjects, and the last 6 activities of the remaining 4 subjects. Subjects were given breaks in between each activity and activities lasted 5-10 minutes, however, these unlabelled breaks were removed from the version that we used. Additionally, only two minutes of data were available for each subject, so time series consisted of six 120 second long activities. We concatenated the data from these six activities together to create a synthetic, free-living dataset. Each of the 91 time series contained a total of $6 * 120 * 30 = 21600$ data ticks.

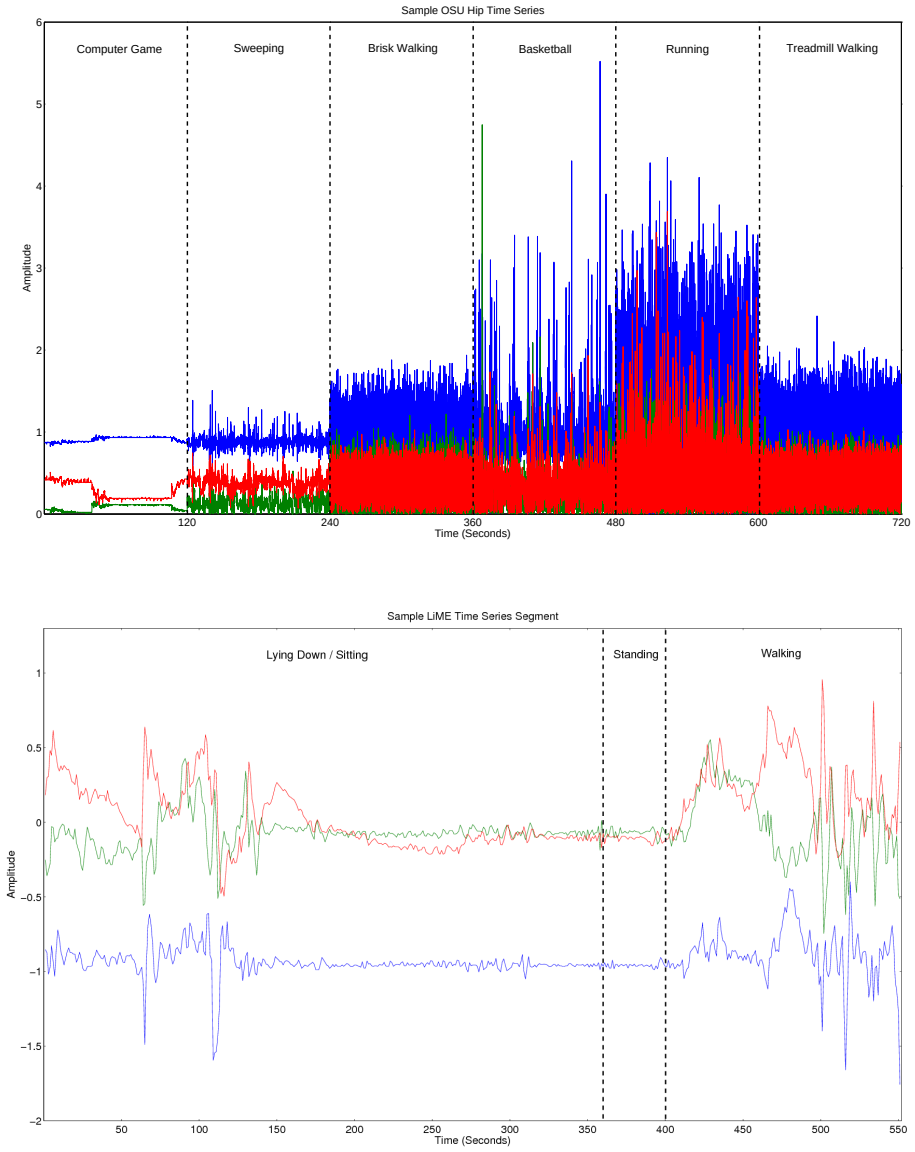

Figure 3.1: Sample data from the OSU Hip and LiME datasets. Both datasets are triaxial and each axis is shown in a different color.

We determined that several of the activities were very similar and that it would be difficult to discriminate between them. As a result, we combined some of them together to create a 7-class version of the data. The classes were lying down, sitting (hand-writing, computer game), standing (laundry, sweeping, and catch), walking (comfortable, brisk and treadmill walking), dancing, running, and basketball.

3.1.2 LiME

This dataset consisted of 23 time series, each containing roughly 10 continuous days worth of data from an individual subject. It was collected by Helen Brown from the Univerity of Cambridge, and Gemma Ryde from the University of Stirling, Scotland. Subjects wore an ActiGraph GT3X+ accelerometer during the entire period, which collected triaxial acceleration data at a frequency of 30Hz, as well as an activPal inclinometer on their thighs. The inclinometer provided what we considered the ground truth labels of the data by automatically delimitting and classifying intervals using the orientation of the subject's thigh at any given moment. It classified a horizontal orientation as lying down/sitting, a vertical orientation as standing, and a combination of the two as walking. The bottom graph of Figure 3.1 shows a time series segment from the dataset that contains all 3 activities.

This dataset was challenging to work with because of its size, as each individual time series contained roughly 25 million ticks of data. To help alleviate this problem, we split each time series into individual days. We then treated the first full 24 hour period of data that began at midnight, from each subject, as one whole dataset (LiME Day 1), and the second such period as a separate dataset (LiME Day 2). We did not use any data from the remaining days.

In contrast to the OSU Hip dataset, LiME was not synthetic and activity lengths were variable. For LiME Day 1, the average activity length was 100 seconds with a standard deviation of 881 seconds, and the median length was 12 seconds. The average number of activities per time series was 871. As would be expected, statistics for LiME Day 2 were comparable: the average activity length was 104 seconds with a standard deviation of 803 seconds, and the median length was 12 seconds. The average number of activities per time series was 834. The medians were relatively small because many of the activities were short, while the mean and standard deviations were larger because a

few of the activities were extremely long (e.g. when subjects were sleeping).

3.2 Featurization

To formulate our experiments as classification problems, we split each time series into a set of non-overlapping windows and represented each window as a feature vector. How we decided where one window ended (and where the next began) varied between experiments, and is described in sections 3.4 and 3.5. Our feature set was a large collection of statistics that have been shown to be discriminative for activity classification in previous research [16], [24], [31], [41]. In all we used 18 statistics that were uniaxial, i.e. were only a function of the data from a single axis of a given window, as well as one biaxial statistic. The uniaxial statistics were applied to data from each axis separately, and the biaxial statistic was applied to data from each of the $C_2^3 = 3$ possible pairs of axes, for a total of $18 * 3 + 3 = 57$ features.

F1. Sum of values of a period of time: $\sum_{i=1}^{T} x(i)$.

F2. Mean: $\mu_x = \frac{1}{T}\sum_{i=1}^{T} x(i)$.

F3. Standard deviation: $\sigma_x = \sqrt{\frac{1}{T}\sum_{i=1}^{T}[x(i)-\mu_x]^2}$.

F4. Coefficients of variation: $\frac{\sigma_x}{\mu_x}$.

F5. Peak-to-peak amplitude: $max\{x(1),...,x(T)\} - min\{x(1),..,x(T)\}$.

F6-10. Percentiles: $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$.

F11. Interquartile range: difference between the 75^{th} and 25^{th} percentiles.

F12. Lag-one-autocorrelation: $\frac{\sum_{i=1}^{T-1}[x(i)-\mu_x][x(i+1)-\mu_x]}{\sum_{i=1}^{T}[x(i)-\mu_x]^2}$.

F13. Skewness: $\frac{\frac{1}{T}\sum_{i=1}^{T}[x(i)-\mu_x]^3}{(\frac{1}{T}\sum_{i=1}^{T}[x(i)-\mu_x]^2)^{\frac{3}{2}}}$, asymmetry of the signal probability distribution.

F14. Kurtosis: $\frac{\frac{1}{T}\sum_{i=1}^{T}[x(i)-\mu_x]^4}{(\frac{1}{T}\sum_{i=1}^{T}[x(i)-\mu_x]^2)^3} - 3$, peakedness of the signal probability distribution.

F15. Signal power: $\sum_{i=1}^{T} x(i)^2$.

F16. Log-energy: $\sum_{i=1}^{T} \log[x(i)^2]$.

F17. Peak intensity: number of signal peak appearances.

F18. Zero crossings: number of times the signal crosses its median.

F19. Correlation between each pair of axes: $\frac{\sum_{i=1}^{T}[x(i)-\mu_x][v(i)-\mu_v]}{\sqrt{\sum_{i=1}^{T}[x(i)-\mu_x]\sum_{j=1}^{T}[v(j)-\mu_v]}}$.

Table 3.1: Statistics used to convert time series windows into feature vectors (taken from [42]). T is the number of ticks in the given time series, x and v are individual axes of the time series data, and $x(i)$ is the ith data tick of x.

One discriminative characteristic of an activity is its overall vigorousness. The sum [F1] and the sample mean [F2] both act as simple and obvious ways of measuring vigorousness, as more intense activities will tend to involve higher rates of acceleration during movement. We also used the 10th [F6], 25th [F7], 50th [F8], 75th [F9], and 90th [F10]

percentiles of the data, as well as signal power [F15] and log energy [F16] as supplemental measures of overall activity intensity.

Another characteristic of an activity is how much it varies in intensity. The sample standard deviation [F3], coefficient of variation [F4], peak-to-peak amplitude [F5], number of zero crossings [F18], as well as the interquartile range [F11] were useful for discriminating between activities with a consistent level of intensity (low variance, etc.) and activities that were more rhythmic or staccato in intensity (high variance, etc.).

Lag-one-autocorrelation [F12], skewness [F13], kurtosis [F14], and peak intensity [F17] were useful for discriminating between activities that tend to be similar in their overall intensity and variation in intensity, but that yielded data with other types of difference in shape. Skewness indicates whether the data is more concentrated above or below its mean. Kurtosis indicates that the data is concentrated near its mean or conversely that it is fat-tailed. Lag-one-autocorrelation is a measure of the general relationship between data ticks and their immediate neighbors in time. Peak intensity is the number of times that the data repeatedly reached its maximum value.

Finally we looked at a single bimodal statistic across each pair of axes, the correlation coefficient [F19], which discriminates between activities where acceleration values in one axis are predictive of acceleration values in another axis, versus activities where that is not the case.

3.3 Base Classifiers

We tested three classification models on the featurized versions of our data: decision trees, support vector machines, and neural networks. We used R for our experiments, and used the R libraries 'rpart' [36], 'e1071' [18], and 'nnet' [23] to build our decision tree, SVM, and neural network models, respectively. Default rpart values were used for the decision tree experiments. The rpart package implements a procedure for automatically tuning how aggressively it prunes its decision tree models, and hence our validation set was not used to tune this parameter. For the neural net experiments, the maximum number of iterations was set to 100000, and the maximum number of weights was set to 1000000.

For the OSU Hip experiments we tuned the cost parameter C of the svm on the validation set with 6 values: $\{0.01, 0.1, 1, 10, 100, 1000\}$. The single-layer feed-forward

neural network took two tuning parameters, and we tuned with each 2-tuple from the set $N \times W$, where $N = \{1, 2, \ldots, 30\}$ was the numbers of nodes in the hidden layer, and $W = \{0, 0.5, 1\}$ was the weight decay parameter.

Since the LiME datasets were an order of magnitude larger, we tuned them slightly differently because of time constraints. Setting the C parameter to 1000 proved to be very computationally expensive for the SVM model; instead we tuned C from the values $\{0.01, 0.1, 1, 10, 100\}$. Running $30 * 3 = 90$ tuning experiments for the neural networks was also prohibitively expensive; instead we tuned using values from $N \times W = \{5, 10, 15\} \times \{0, 0.5, 1\}$.

3.4 Top-Down Approach

3.4.1 Change-Point Detection

For this approach, the data was split into non-overlapping segments using techniques from the statistical field of change-point detection. Change-point detection has found application in many problem domains that require analysis of time series data from dynamic systems, including failure detection [3], quick detection of attacks on computer networks [34], and monitoring of heartbeat fluctuations during sleep [30]. Change-point detection techniques assume that each tick of a time series is a draw from some probability distribution, but that the distribution may suddenly change as time passes. The goal is to predict when these changes have occurred. A *score* is generated for each time tick, and if the score is above a given threshold a change is predicted to have occurred between that tick and its immediate predecessor. To generate a score at a time tick, a window of data that immediately preceeds it (the *reference data*) is compared to it along with a window of data that immediately follows it (the *test data*), as shown in Figure 3.2.

Model-based approaches to change-point detection assume that each tick in a time series is a draw from some underlying probability distribution. Scores are generated by estimating the distribution of the reference data and the test data, and then by calculating the likelihood that the two distributions are different. Parametric estimation methods have been employed where is it reasonable to assume that the given data belongs to a particular family of distributions [35]. Non-parametric methods have also been found to be viable where no such modeling assumptions are reasonable [17]. Distance-based

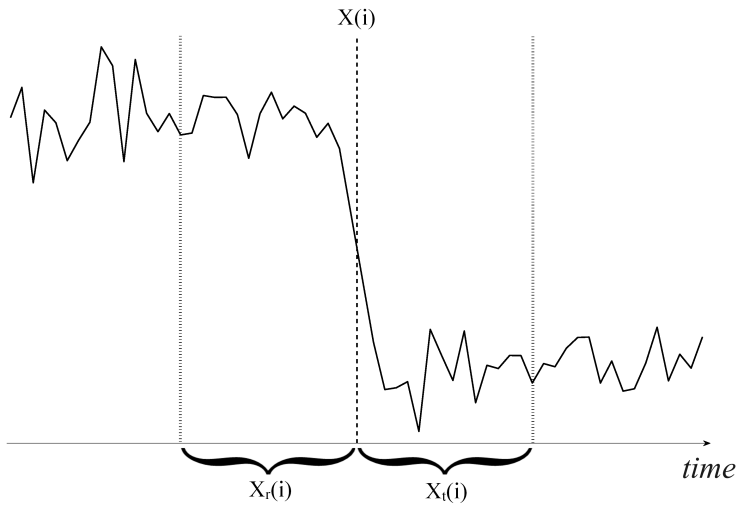

Figure 3.2: Reference and Test Data

approaches such as Singular Spectrum Analysis generate scores through other metrics of dissimilarity or difference between the reference data and the test data [19]. Notationally, we say that for each tick i in a time series:

$$s(i) = D(X_r(i), X_t(i))$$

Where $s(i)$ is the score of the ith tick, $X_r(i)$ is the reference data associated with the ith tick, $X_t(i)$ is the test data associated with the ith tick, and $D(A, B)$ is a function that computes the dissimiliarity between a matrix of data A and matrix of data B. $D(A, B)$ varies between change-point algorithms. Note that for a given algorithm it may not be possible to generate scores right at the very beginning of the time series (insufficient reference data) or at the very end of a time series (insufficient test data).

3.4.2 Experimental Setup

Each dataset that we tested consisted of multiple time series gathered from a number of different subjects. To perform an experiment on a dataset we began by partitioning

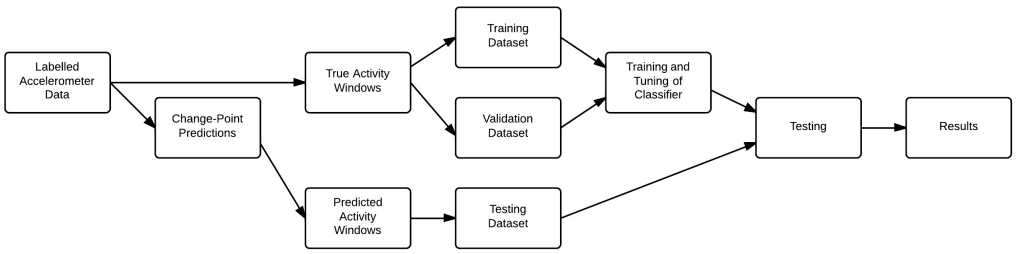

Figure 3.3: Data Lifecycle of Change-Point Detection Experiments

the set of time series into disjoint subsets of training, validation, and testing data. Each individual time series was then partitioned into a set of non-overlapping windows, and each window was converted into its own feature vector. Once the dataset was featurized, the experiment could be treated as a typical classification problem. Classifiers were built with the training set, and tuned (when necessary) on the validation set. Finally, the tuned model was evaluated by prediction on the testing set. A visualization of the data lifecycle is shown in Figure 3.3.

There are many different modeling assumptions and associated algorithms for generating change-point detection scores, and one simple baseline approach that we wanted to test was the Shewhart Control Chart [27]. This approach assumes that the reference data is drawn from a multivariate normal distribution, and that scores are calculated by the Mahalanobis distance of the target time tick from the estimated multivariate normal:

$$s(i) = \sqrt{[\bar{X}_r(i) - X(i)]^T \ \frac{1}{S_r(i)} \ [\bar{X}_r(i) - X(i)]}$$

where $\bar{X}_r(i)$ is the sample mean of the reference data, $S_r(i)$ is the sample covariance matrix of the reference data, and $X(i) = X_t(i)$ is the ith data point.

We were also interested in testing the performance of a newer and more sophisticated change-point detection algorithm: the Kullback-Leibler Importance Estimation Procedure (KLIEP) [13], [33]. This approach generates scores using the Kullback-Leibler (KL) divergence between the reference data and the test data. One method of doing this is

to estimate the density of the reference distribution and test distribution separately, and then compare them using a likelihood ratio (known in the change-point detection literature as *importance*). Instead, KLIEP estimates the importance directly using a non-parametric model.

Let the estimate of the importance $\hat{R}$ be represented by this model:

$$\hat{R} = \frac{p_t}{\hat{p}_r} = \sum_{i=1}^{T_t} \alpha_j K_G(X_{rt}, X_t(j))$$

Where p_r and is the probability density of the reference data, $hatp_r$ is the estimate of p_r, p_t is the probability density of the test data, T_t is the number of ticks in the test window, α is a vector of model parameters to solve for, X_{rt} is the concatenation of the reference and the test data, $X_t(i)$ is the ith element of the test data, and $K_G[A, B]$ is the Gaussian kernel with width σ:

$$K_G[A, B] = \exp\left(-\frac{||A - B||^2}{2\sigma^2}\right)$$

Now solve for α so that the empirical KL divergence between $\hat{p}_t$ and $p_t = \hat{R}p_r$ is minimized, which is equivalent to the following convex optimization problem:

$$\begin{cases} \max\limits_{\alpha} & \sum\limits_{j=1}^{T_t} \log\left(\sum\limits_{k=1}^{T_t} \alpha_k K_G[X_t(j), X_t(k)]\right) \\ \text{s.t.} & \frac{1}{T_r}\sum\limits_{j=1}^{T_r}\sum\limits_{k=1}^{T_t} \alpha_k K_G[X_r(j), X_t(k)] = 1 \\ & \text{and } \alpha_1 \ldots \alpha_{T_t} \geq 1 \end{cases}$$

Finally, the scores that we wish to generate are just the estimate of the importance given by the solution to the complex optimization problem, i.e. $s(i) = \hat{R}(i)$.

Since this approach uses a Gaussian kernel, it requires the selection of a kernel width σ for each time tick. We used an implementation of KLIEP that is available at Sugiyama's website (http://sugiyama-www.cs.titech.ac.jp/ sugi/software/KLIEP/), which included a cross-validation procedure for the value of σ. The CV procedure chooses a number of disjoint splits of the test data along with a number of different candidate σ's, and runs KLIEP with each combination of split and candidate σ. Then it chooses the candidate

σ that, on the average across all of the splits, maximizes the KL divergence (the $\max_\alpha$ equation above) the most.

For the OSU Hip dataset, we used this CV procedure to choose the kernel width. This computationally intensive approach was impractical for the UQ dataset because it is orders of magnitude larger, so instead of running it on every tick of that data, we ran the CV procedure on a small sample of data. From this smaller sample we were able to empirically identify 0.01 as a plausible σ, and so fixed σ at that value for our experiments on that dataset.

Our selection of reference and test window sizes were informed by two considerations: first that the window sizes contain enough data to accurately model the reference and test distributions, and second that the window sizes were small enough to detect activity changes at real time speeds. Previous research [41] found that a reference window size of 10 seconds contained just enough information to discriminate well between OSU Hip activities. Since this window size worked well in previous experiments, and since the activities of the UQ dataset were comparable in average length, we decided to fix our reference window size at 10 seconds for both datasets. Because we were interested in minimizing detection time, and because 1 second was the smallest window that we felt could provide some information about an activity, we fixed our test window size at 1 second for both datasets.

Once change-point detection scores were generated, we tested a number of threshold values that determined which scores were high enough to be considered a predicted change-point. Threshold values were chosen by considering the false positive rates of change prediction for the change-point detection algorithms. A smaller false positive rate corresponded to a higher and more conservative threshold, which split the time series into fewer segments. A larger false positive rate corresponded to a lower threshold, which split the time series into more segments.

3.5 Bottom-Up Approach

3.5.1 HMMs

An alternate technique to our change-point detection approach was based around the Hidden Markov Model (HMM). An HMM is a temporal graphical model that contains

a set of hidden states $H = \{H_0, H_1, \ldots, H_w\}$ as well as a set of observed states $O = \{O_1, O_2, \ldots, O_w\}$ (see Figure 3.4). The index of either type of state represents a point in time, such that if there exists two indices i and j where $i < j$, i is thought of as having happened before j. Each hidden state in the model contains a value from the *state space* $U = \{U_1, U_2, \ldots, U_\ell\}$, and each observed state contains a value from the *observation space* $V = \{V_1, V_2, \ldots, V_m\}$. The values of the hidden states are unknown, and the values of the observable states are known. It is also assumed that, as indicated by Figure 3.4, that the value of an observable state O_i is dependent only the value of the corresponding hidden state H_i, and that the value of any hidden state H_i is dependent only on the value of its immediate predecessor H_{i-1}.

Furthermore, the dependencies between hidden states and their followers are assumed to be described by a stationary stochastic process known as the *transition model*, $T : U^2 \to [0, 1]$. The dependencies between a hidden state and its adjacent observable state are assumed to be part of a separate but also stationary stochastic process known as the *observation model*, $S : U \times V \to [0, 1]$. In other words, both models can be thought of as a function of two values, that output the probability of a change from the first value to the second via a dependency arc in the HMM. The usual approach to estimating T and S given only O is an application of expectation maximation known as the Baum-Welch algorithm [5], which is useful for finding $\hat{T}$ and $\hat{S}$ that are locally maximally likely. However, suppose a *training HMM* $\langle H_{tr}, O_{tr} \rangle$ with the same model parameters is given, and the values of all of its hidden states as well as its observable states are known. T and S are then approximated by the following global maximum likelihood estimators:

$$\hat{T}(U_i, U_j) = \frac{|\{H_k \in H_{tr} \mid H_k = U_i \text{ and } H_{k+1} = U_j\}|}{|\{H_k \in H_{tr} \mid H_k = U_i\}|}$$

$$\hat{S}(U_i, V_j) = \frac{|\{H_k \in H \mid H_k = U_i \text{ and } O_k = V_j\}|}{|\{H_k \in H \mid H_k = U_i\}|}$$

Finally, if all of the values of O are known, and we are given a $\hat{T}$ and an $\hat{S}$ estimated from a training HMM, then the goal we are interested in is to use that information (along with the model assumptions) to find the most likely values for each state in H. There exists a polynomial-time dynamic programming solution to this problem known as the

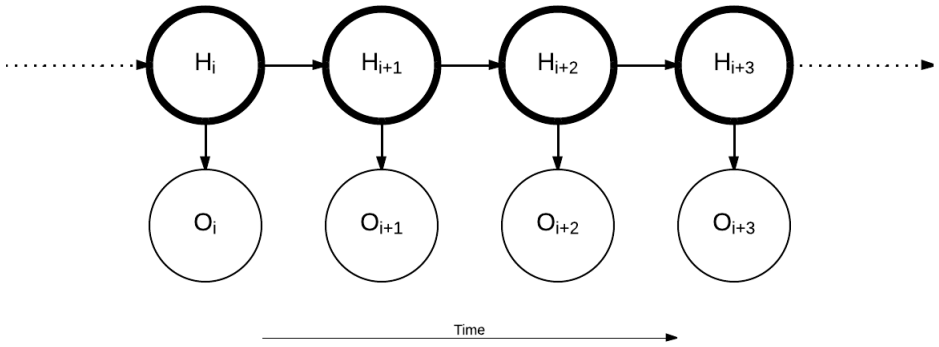

Figure 3.4: Visual Interpretation of an HMM

Viterbi algorithm. [26]

3.5.2 Experimental Setup

The data lifecycle for the HMM experiments is shown in Figure 3.5. We began by partitioning each time series into small non-overlapping windows, where each window corresponded to a discrete time index in the HMM. Within a given experiment the window size was fixed, but across different experiments we tested window sizes of length $\{10, 12, 14, 16, 18, 20\}$. After the time series were partitioned they were then featurized. Classification models were built with training data, and tuned (in the case of the SVM and neural net models) using validation data, in the same way that has already been described previously in this chapter.

Unlike the change-point detection experiments, this experiment required that the data be split into 4 equal parts: training (classifier), validation, training (HMM), and testing rather than 3. Here we formulated the problem of making predictions on the testing set in terms of an HMM by treating the second training set as a training HMM. In our datasets we let H be the ground truth activity classes of the windows, and O be the predicted activity classes of the windows, where the predictions were made by the classifiers trained on the first training set. We used the procedure above to calculate $\hat{T}$ and $\hat{S}$, and assumed that these estimates held for the testing set as well as the second

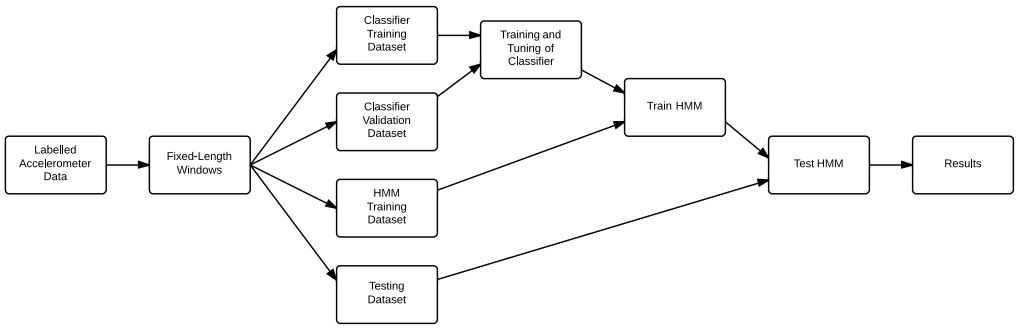

Figure 3.5: Data Lifecycle of HMM Experiments

training set. We then used the tuned base classifier to predict on the testing set, giving us O. Finally, we used O, $\hat{T}$, and $\hat{S}$ to run the Viterbi algorithm on the testing set and predict the ground truth activity classes H.

3.6 Performance Metrics

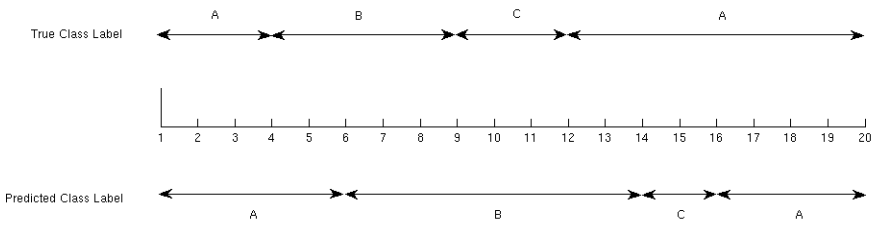

Figure 3.6: An example time series with 20 ticks of data, from a dataset with three classes: A, B, and C. Four true class labels and their associated windows are shown above the axis; four predicted class labels and their associated windows are shown below the axis.

To measure the performance of our classification algorithms we used two metrics. Accuracy is defined as the number of ticks that an algorithm correctly classifies in a time series, over the total number of ticks in the time series. Accuracy is computed

by counting the number of correctly predicted ticks for each true window separately, summing the counts, and dividing by the total number of ticks. An example section of a time series is shown in Figure 3.6, and the accuracy of the shown predictions is given as follows:

$$\text{Accuracy} = \frac{\text{CPT}(\text{A}_1) + \text{CPT}(\text{B}) + \text{CPT}(\text{C}) + \text{CPT}(\text{A}_2)}{(\text{Total number of ticks})} =$$

$$\frac{3+3+0+4}{20} = 50\%$$

where CPT(.) is the number of correctly predicted ticks in an interval, and A_1 and A_2 are the true class "A" windows.

Since we were also interested in our algorithms' feasibility for activity classification in real time, we used detection time as a second metric. Detection time is computed by counting how many ticks are required for a prediction algorithm to start correctly predicting the class, after a true window begins. These counts are summed, and then divided by the number of true activities.

Consider again the time series segment in Figure 3.6. Over the true window beginning at tick 1 (class A), the algorithm predicts A immediately, so the detection time for that window is 0. Over the second true window beginning at tick 4 (class B), the algorithm does not start predicting B until tick 6 so the detection time for that window is $6-4=2$. Over the third true window starting at tick 9 (class C), the algorithm never predicts C, so the detection time for that window is 3, the full length of the window. Over the fourth true window starting at tick 12 (class A), the algorithm starts to predict C at tick 14, but does not predict A until tick 16, so the detection time for that window is $16-12=4$. Thus the average detection time over the time series segment is:

$$\frac{(0+2+3+4)\text{ ticks}}{4\text{ windows}} = \frac{9}{4}\text{ ticks per window}$$

In the change-point detection experiments accuracy and detection time were averaged over 30 random splits of the given dataset into training, validation, and testing sets. Because the HMM experiments were more computationally expensive, accuracy and detection time were averaged over 10 random splits of the given dataset into training (base classifier), validation, training (HMM), and testing sets.

Chapter 4: Results

4.1 Top-Down

Results for our change-point detection experiments are given in Figures 4.1-4.3. The performance of the change-point detection algorithms depended heavily on the threshold level for change prediction. In our experiment we varied the threshold level, thus changing the false positive rate. A large number of false positives per second were tested, but for the sake of brevity only a representative sample of $\{0.005, 0.01, 0.05, 0.1\}$ are shown here.

In the OSU Hip experiments, control charts outperformed KLIEP in terms of detection time (Figures 4.1.2, 4.1.4, 4.1.6), while the accuracy results (Figures 4.1.1, 4.1.3, 4.1.5) were mixed. Except when predicted windows are large enough to span across multiple true activities, it is generally expected that accuracy will decrease as false positive rate increases because small windows contain less information and are less discriminative than larger windows. This behavior is seen in the control chart accuracy results (grey bars in Figures 4.1.1, 4.1.3, 4.1.5), but not in the KLIEP accuracy results (white bars in Figures 4.1.1, 4.1.3, 4.1.5). Follow-up experiments showed that KLIEP peaks in accuracy for false positives per second between 0.2 and 0.3 for all three classifiers. KlIEP seemed to perform best on this dataset when it was given many opportunities to predict changes.

Further investigation indicated that across the OSU Hip dataset the KLIEP algorithm was unable to detect many of the different activity changes without a very low score threshold value (and a very high false positive rates). Some qualitative plotting of the OSU Hip data showed that most of its activities have accelerometer amplitude values that strongly resemble draws from a multivariate normal distribution. Since control charts assume that the data is drawn from a distribution that is a member of that family, it is logical that control charts would outperform algorithms with different modeling assumptions on OSU Hip.

In the LiME experiments, KLIEP outperformed control charts in terms of accuracy

across the board, and control charts outperformed KLIEP in terms of detection time across the board. This suggests that in general control charts correctly detected true changes more quickly, but that after a correct change prediction it was more likely to make an incorrect change prediction.

In a few cases (Figures 4.1.2, 4.1.6, 4.2.6) the detection time did not decrease as the false positive rate increased. On the face of it this would seem to be a non-sequitur, but this only happened in cases when accuracy also decreased (Figures 4.1.1, 4.1.5, 4.2.5). Smaller window sizes tend to be correlated with decreased detection times, but it is possible that predicting with smaller windows, if they happen to contain an insufficient amount of discriminative data, can actually increase the time required for the classifier to start correctly predicting the ground-truth activity. Additionally, the given increases in detection time were small and within confidence bounds.

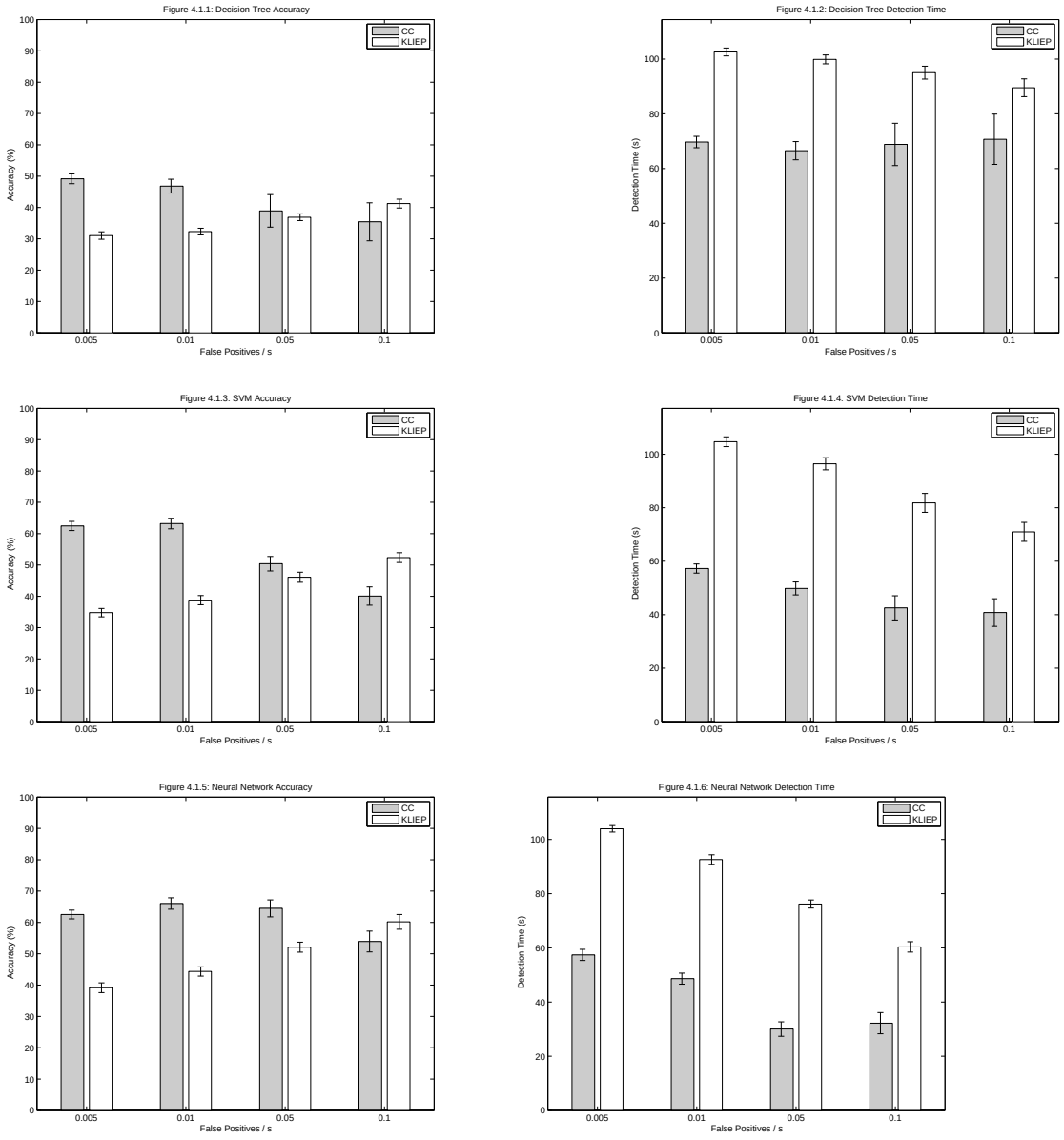

Figure 4.1: OSU Hip Results. Graphs are organized into rows by base classifier, and columns by evaluation metric. Change-point detection results were averaged over 30 splits into training, testing, and validation datasets. Error bars show a 95% confidence interval around the average.

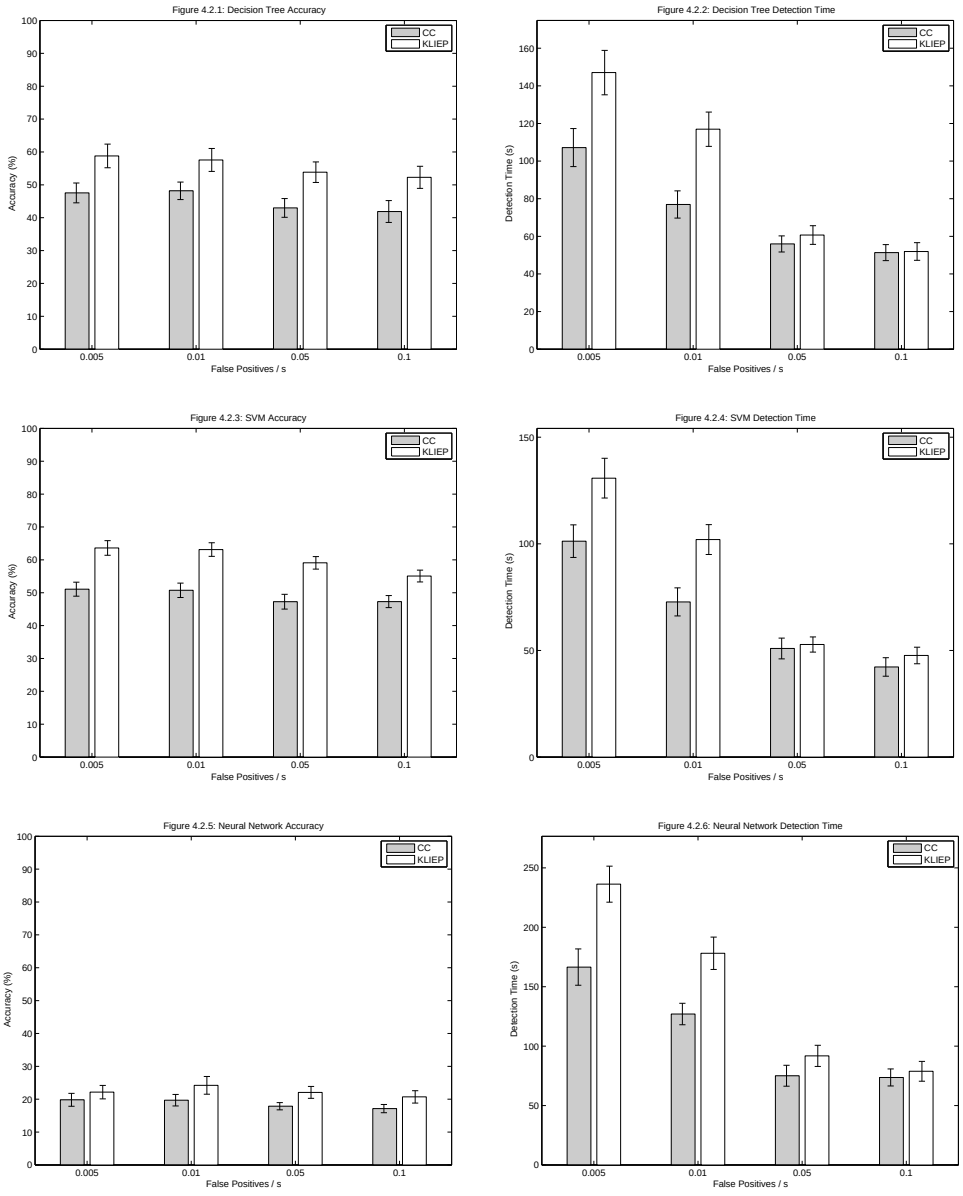

Figure 4.2: LiME Day 1 Results. Graphs are organized into rows by base classifier, and columns by evaluation metric. Change-point detection results were averaged over 30 splits into training, testing, and validation datasets. Error bars show a 95% confidence interval around the average.

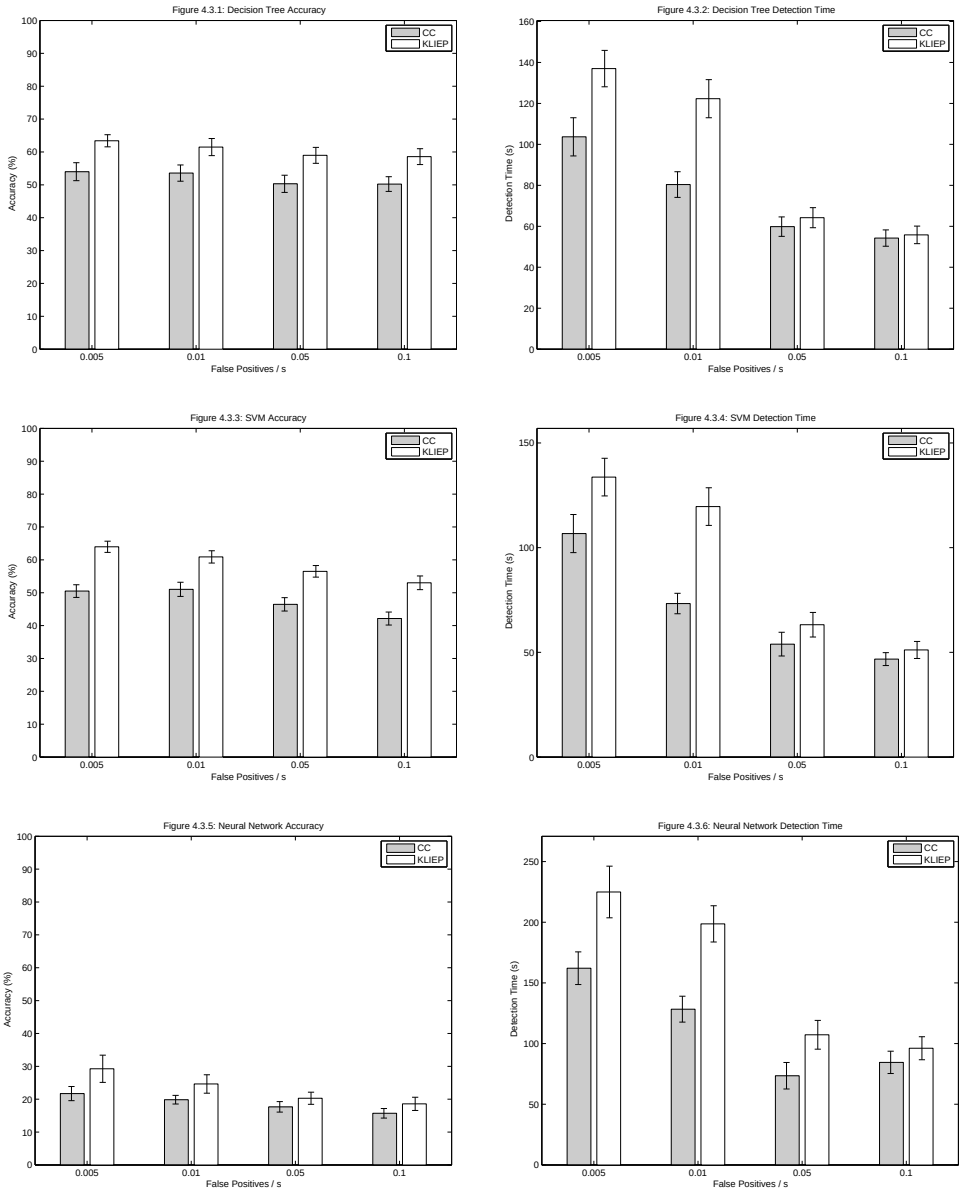

Figure 4.3: LiME Day 2 Results. Graphs are organized into rows by base classifier, and columns by evaluation metric. Change-point detection results were averaged over 30 splits into training, testing, and validation datasets. Error bars show a 95% confidence interval around the average.

4.2 Bottom-Up

Results for our bottom-up experiments are given in Figures 4.4-4.6. Each experiment was performed by splitting each time series into windows of fixed length corresponding to discrete time "ticks" in an HMM, and results for windows of length $\{10, 12, 14, 16, 18, 20\}$ seconds are shown.

For all three base classifiers, accuracy was high and stable with respect to window size, over all three datasets. Detection time was also fairly stable in the OSU Hip experiments, though as would generally be expected it increased somewhat with window size in the LiME experiments. Further experiments [results not shown] on the OSU Hip dataset showed that the accuracy and detection time of our bottom-up approach tends to be poor for very small window sizes, but that it stabilizes with window sizes that are greater than roughly 5 seconds. This gives a strong indication that 5 seconds is the amount of information necessary for the classifiers to become as discriminative as they can be on the datasets in our work.

Using the HMM tended to give a slight boost in accuracy to the base classifiers, but at the expense of increased detection time. It is likely that the increase in accuracy was due to the smoothing of false base classifier predictions that were sandwiched in time between true base classifier predictions, and that the increase in detection time was due to the general stickiness of activities. Once the subject started performing an activity the probability of the subject stopping that activity to start a new one was recognized to be low, which meant that a true change in activity would be less likely to be confirmed by the smoothing algorithm immediately.

The experiments also show the reason for the difference in performance between the top-down and bottom-up approaches. The difference could have been caused by the use of change-point detection algorithms to segment the data as opposed to using fixed length windows, or the ability of the HMM to smooth predictions made on temporal data. The side-by-side comparison of the same fixed window length classification experiments performed with and without HMM smoothing suggests that the smoothing effect of the HMM does not explain most of the performance difference, and that the performance difference between the top-down and bottom-up approaches was mostly due to noisy segmentation by the change-point detection algorithms.

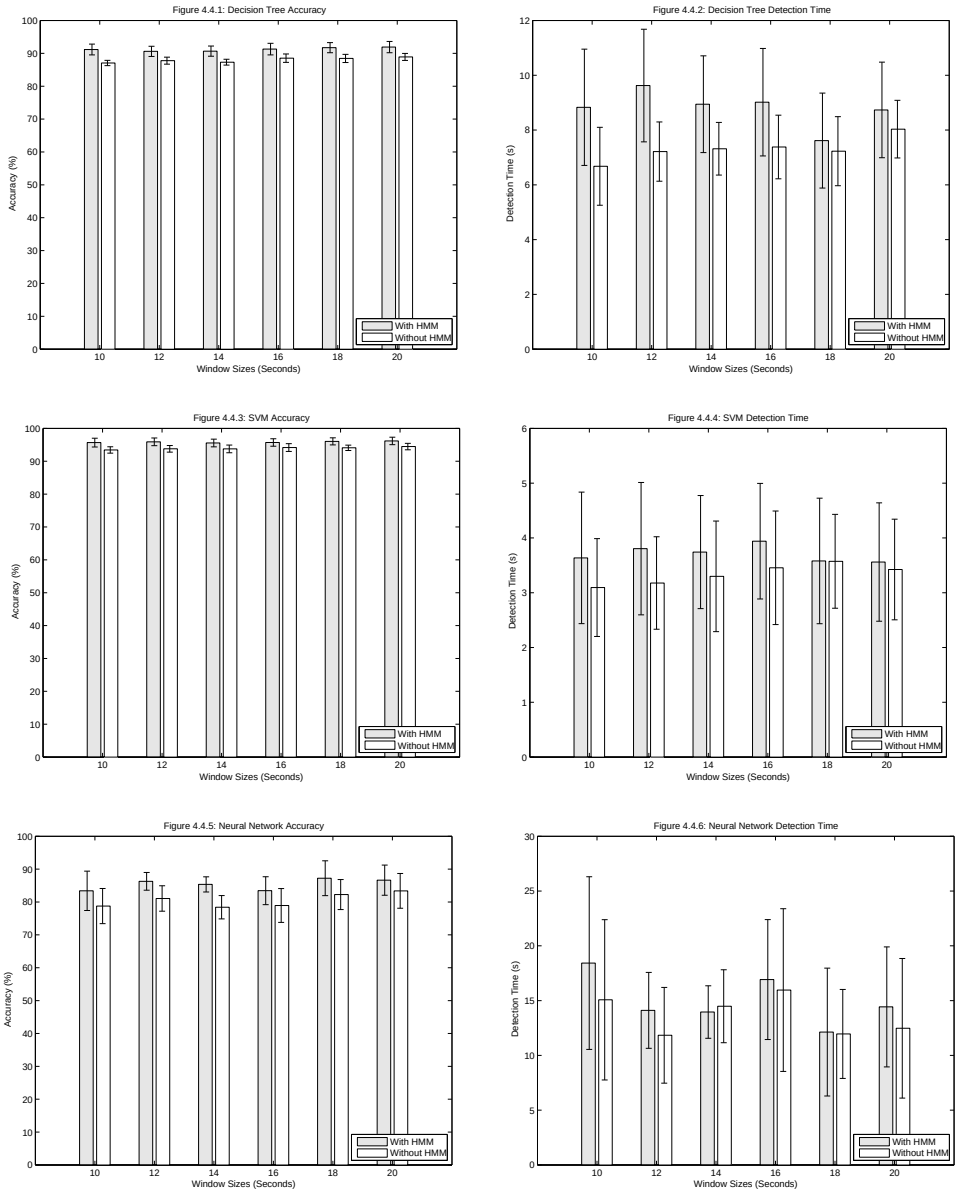

Figure 4.4: OSU Hip Results. Comparison of base classifier performance with and without the HMM smoothing layer, with test data split into fixed window sizes. Graphs are organized into rows by base classifier, and columns by evaluation metric. Error bars show a 95% confidence interval.

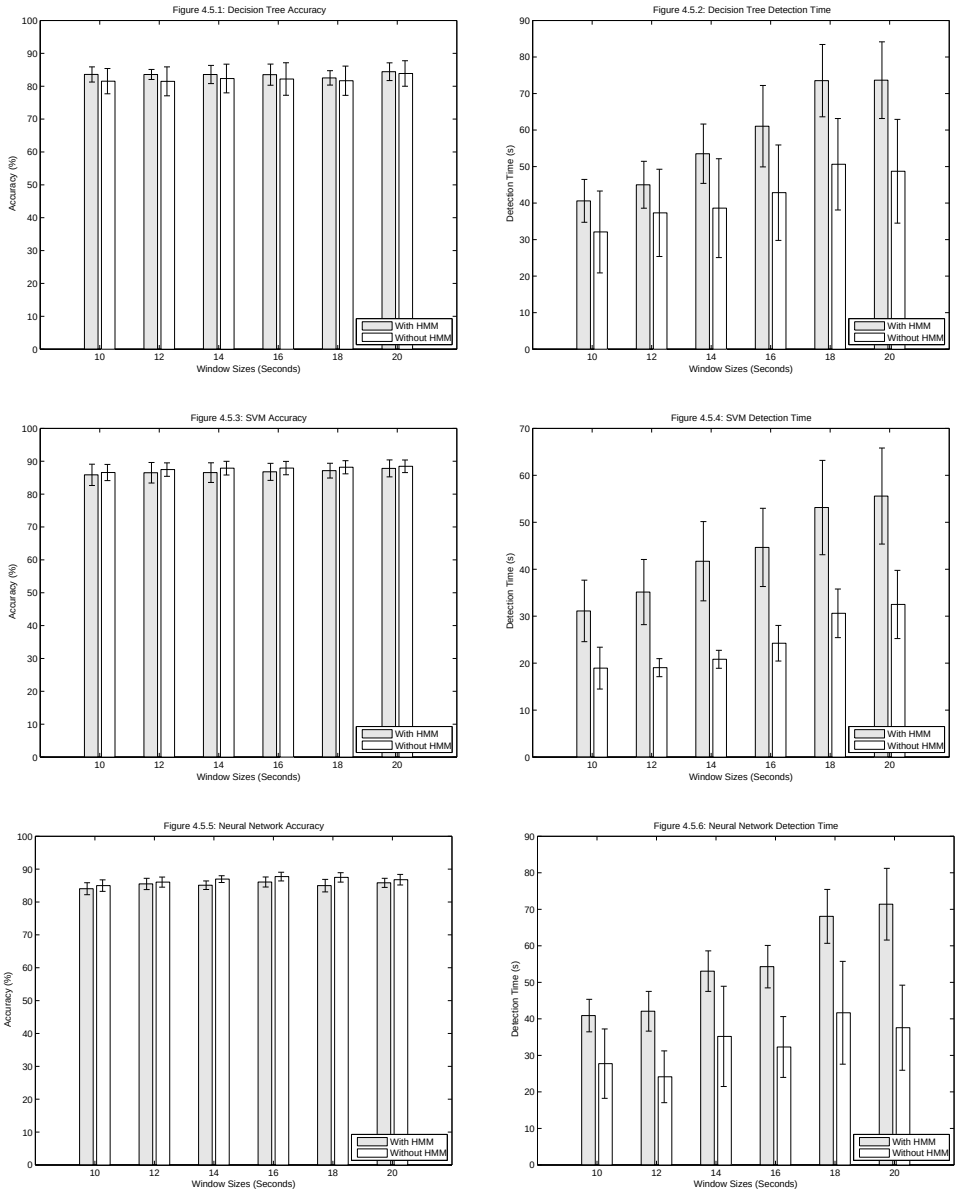

Figure 4.5: LiME Day 1 Results. Comparison of base classifier performance with and without the HMM smoothing layer, with test data split into fixed window sizes. Graphs are organized into rows by base classifier, and columns by evaluation metric. Error bars show a 95% confidence interval.

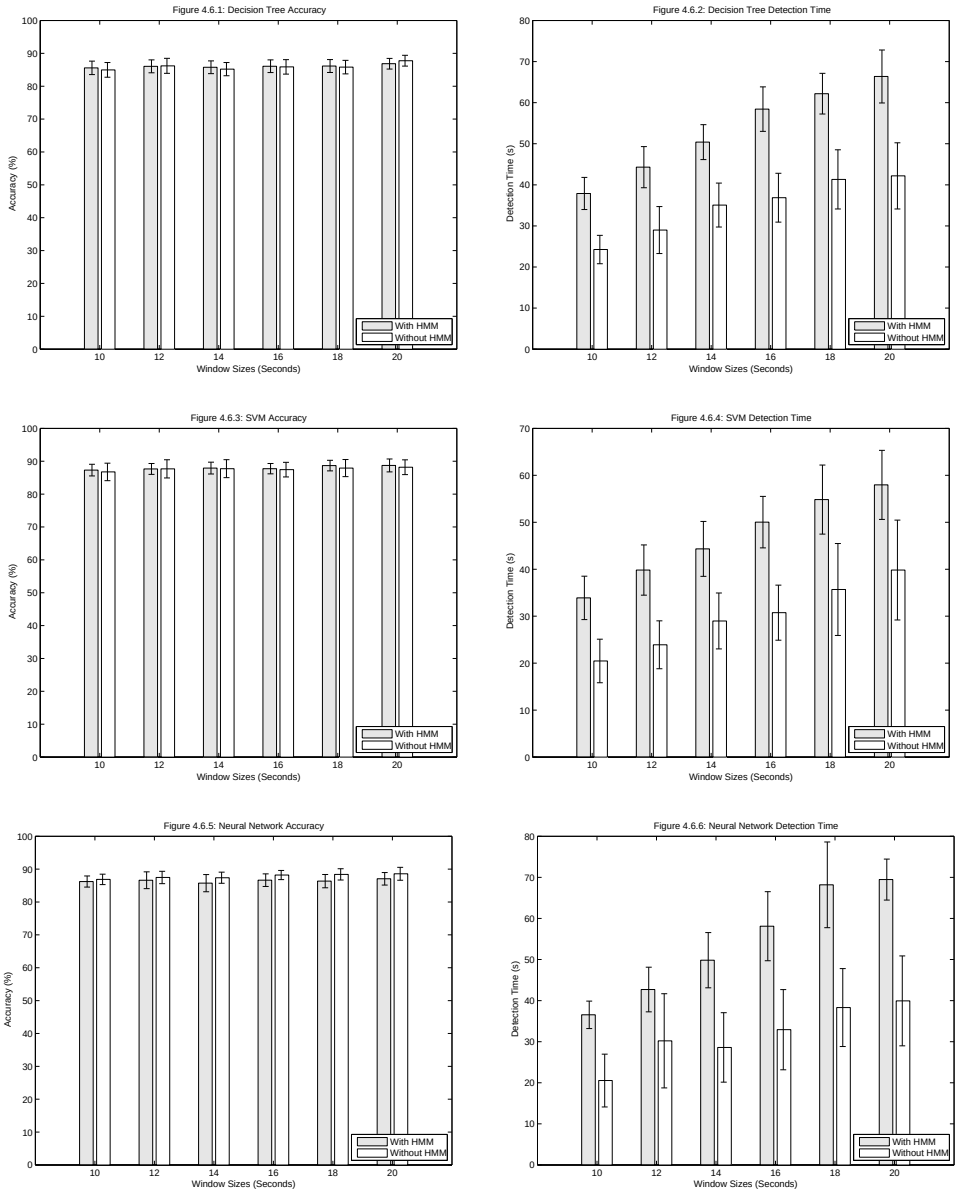

Figure 4.6: LiME Day 2 Results. Comparison of base classifier performance with and without the HMM smoothing layer, with test data split into fixed window sizes. Graphs are organized into rows by base classifier, and columns by evaluation metric. Error bars show a 95% confidence interval.

4.3 Discussion

We found it difficult to directly compare performance between the top-down and bottom-up approaches, because we varied false positives per second in the top-down experiments, and varied the fixed-length window size in the bottom-up experiments. However, for a given false positive rate change-point detection algorithms split a time series into windows of a certain average size, which generally decreases as the false positive rate increases. From this it is possible to relate false positive rates from the top-down experiments to the window sizes of the bottom-up experiments. HMM false positive rates per second of {0.033, 0.028, 0.024, 0.021, 0.019, 0.017} correspond to average window sizes of {10, 12, 14, 16, 18, 20}. Figures 4.7, 4.8, and 4.9 show a side-by-side comparison of the top-down and bottom-up approaches, using this conversion. As seen in Section 4.1, two algorithms were tested for each of the change-point detection experiments, but only the best performing algorithm (highest in accuracy or lowest in detection time) is shown here, for each individual experiment. Our results show that the bottom-up approach outperformed the top-down approach, both in terms of accuracy and detection time, regardless of the dataset and base classifier.

A contributing factor to the particularly high accuracy and low detection time results generally attained for the OSU Hip experiments was that the data consisted of activities that were synthetically glued together. The same group of activities were performed in the same order by each of the 50 subjects in this dataset, making transitions from one activity to the other very predictable for a temporal model. By contrast, the LiME datasets consisted of unsynthetic data gathered from a large set of unstructured and variable-length activities, so the activity transitions were not as predictable and are more indicative of an application of our techniques in the real world.

A final point of interest was that SVM (Figures 4.7.3, 4.7.4, 4.8.3, 4.8.4, 4.9.3, 4.9.4) clearly outperformed the other two base classifiers, and that the faster and simpler decision tree model (Figures 4.7.1, 4.7.2, 4.8.1, 4.8.2, 4.9.1, 4.9.2) matched up well against neural networks (Figures 4.7.5, 4.7.6, 4.8.5, 4.8.6, 4.9.5, 4.9.6). This result is significant because much of the previous research that has formulated activity detection as a supervised learning problem has used neural networks exclusively.

4.4 Timing

Since we are interested in the feasibility of performing activity detection on accelerometer data in real time, we tested how long it would require our approaches to segment and classify a time series in a streaming, online fashion. Experiments were performed using an Intel(R) Core(TM) 2 Quad Processor Q9550.

There are three different computations that a device must perform to do activity recognition in real time. The first (when using the top-down approach) is to calculate a change-point detection score for an individual time tick. When the device obtains a new tick of accelerometer data, it must generate a change-point detection score and compare it to the score threshold to decide whether or not to predict an activity change. We estimated the amount of time required to determine whether an activity change should be predicted by choosing 1000 random ticks from our data, and running our two change-point detection algorithms on the reference and test data corresponding to each tick. We found that on average the control chart algorithm took 0.050ms per tick, and that the KLIEP algorithm took 31.4ms per tick. If no change was predicted, then the device needs to take no further action for this tick.

If a change was predicted in the first step, then the second step is to featurize the window of data corresponding to the activity that just ended. The amount of time required to featurize an activity window increases with the length of the window. The OSU Hip dataset contained 120 second long activities, and the median length of the LiME activities was considerably smaller than that, therefore we fixed the window length at 120 seconds for this experiment. The amount of time required to featurize these windows, again averaged over 1000 runs, was 98.5ms.

Once the data from an activity window is featurized, the activity must be predicted using a base classifier. We assume that the classifier has already been trained and validated previously. We averaged the activity prediction time over 60 runs on each of our base classifiers, with featurized windows and models built during our change-point detection experiments. We found that the average time required to predict an activity with decision trees was 372ms, with SVMs was 368ms, and with neural networks was 369ms.

Alternatively, the bottom-up approach does not include the computation of a change-point detection score, but does include as a final step the computation (via the Viterbi

algorithm) of the set of hidden states in an HMM that are most likely to correspond with classifier predictions on the windows in the test data. The Viterbi algorithm is an offline algorithm that considers all of the windows in the time series rather than just the latest window, but it nonetheless runs quickly. The amount of time required to perform this calculation on an OSU Hip time series, averaged over 1000 runs, was 928ms. An online implementation of this algorithm would be faster still, running in constant time with respect to the number of windows rather than linear time.

These experiments show that the amount of time required to process an activity tick is in the worst case only 98.5ms + 372ms + 928ms = 1.40s, which is a very reasonable amount of delay for online applications.

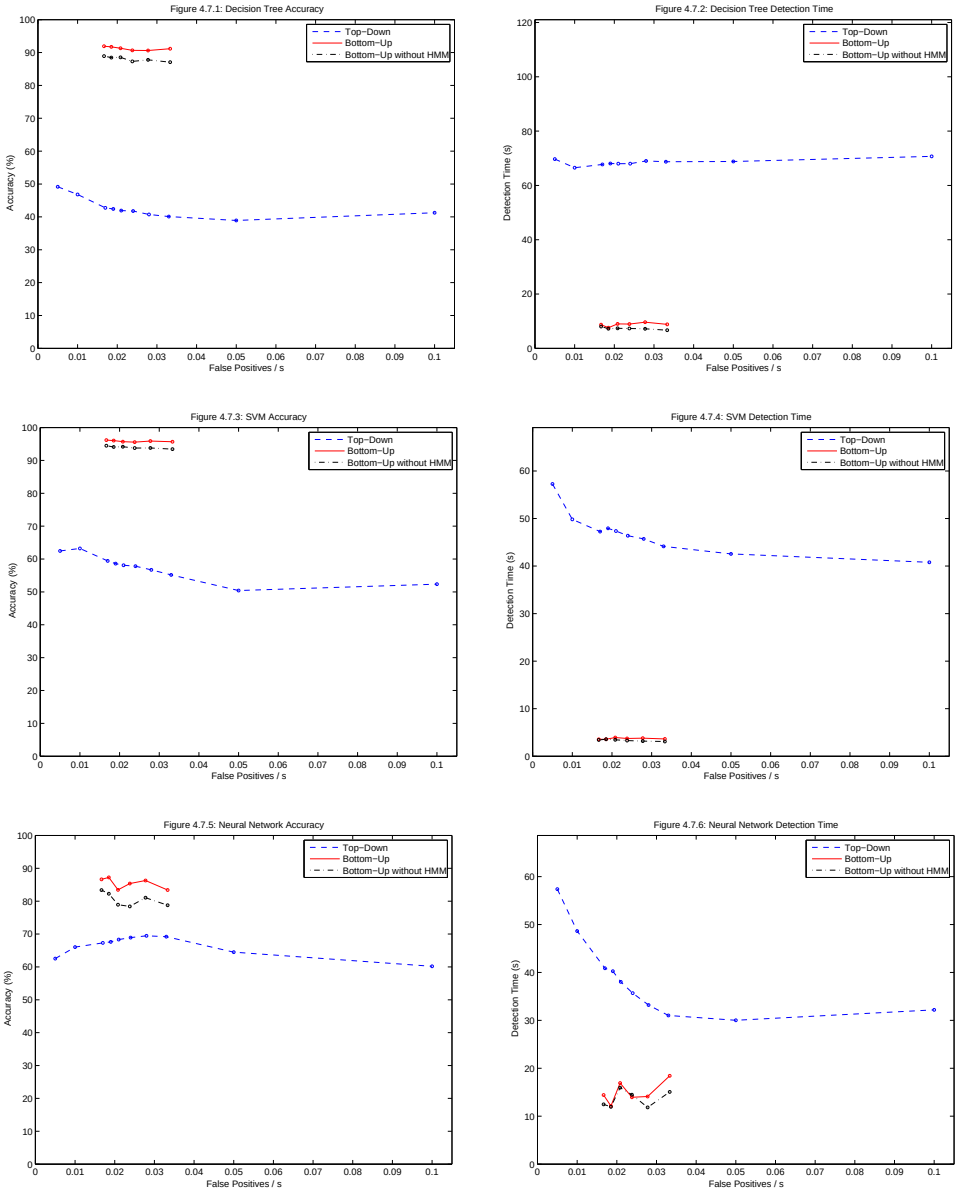

Figure 4.7: Comparison of top-down and bottom-up results for OSU Hip. Graphs are organized into rows by base classifier, and columns by evaluation metric. Results for false positives per second of {0.005, 0.01, 0.017, 0.019, 0.021, 0.024, 0.028, 0.033, 0.05, 0.1} are top-down experiments, while results for false positives per second of {0.017, 0.019, 0.021, 0.024, 0.028, 0.033} are bottom-up experiments.

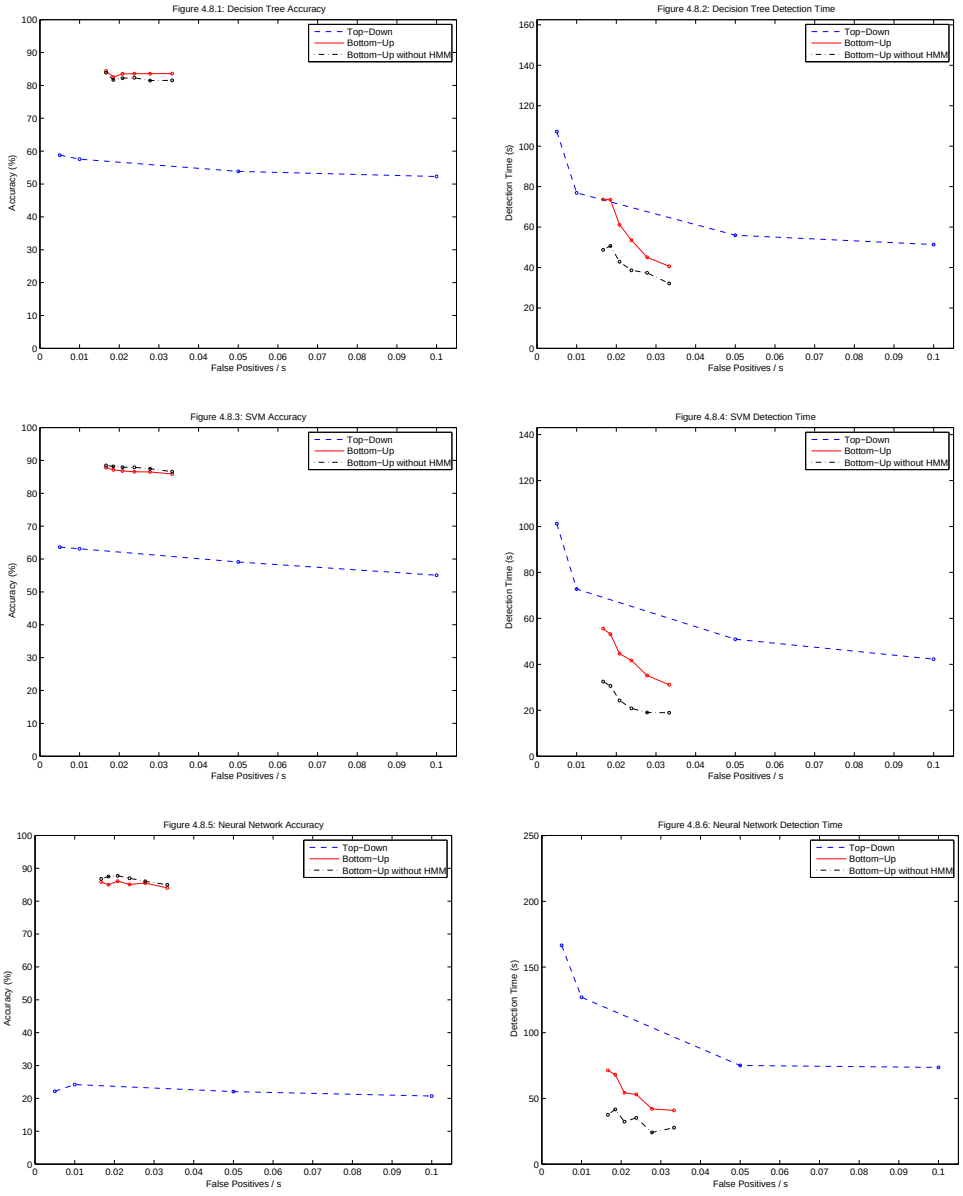

Figure 4.8: Comparison of top-down and bottom-up results for LiME Day 1. Graphs are organized into rows by base classifier, and columns by evaluation metric. Results for false positives per second of {0.005, 0.01, 0.05, 0.1} are top-down experiments, while results for false positives per second of {0.017, 0.019, 0.021, 0.024, 0.028, 0.033} are bottom-up experiments.

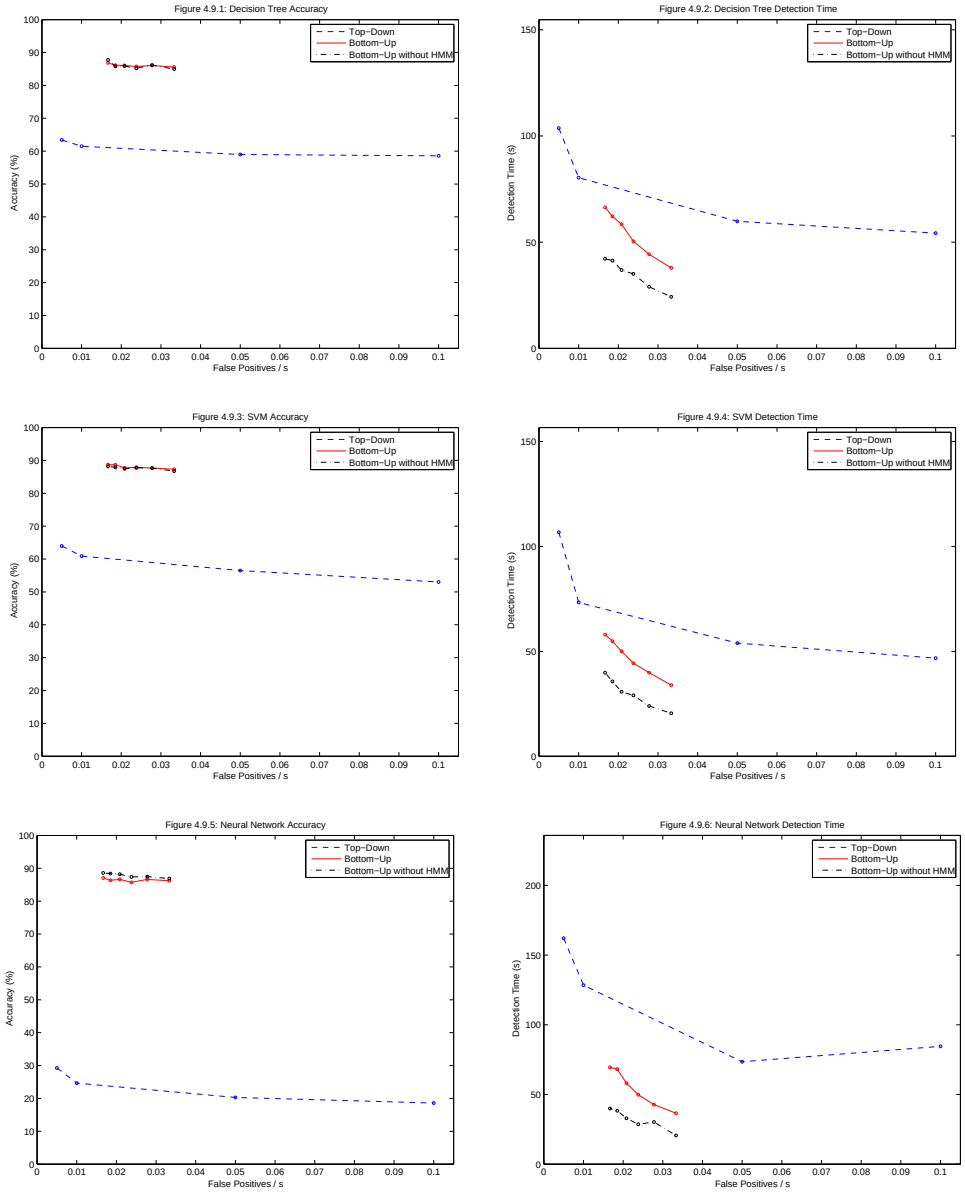

Figure 4.9: Comparison of top-down and bottom-up results for LiME Day 2. Graphs are organized into rows by base classifier, and columns by evaluation metric. Results for false positives per second of {0.005, 0.01, 0.05, 0.1} are top-down experiments, while results for false positives per second of {0.017, 0.019, 0.021, 0.024, 0.028, 0.033} are bottom-up experiments.

Chapter 5: Conclusion

The purpose of this work was to test the feasibility of classifying accelerometer data in real time. We were also interested in using change-point detection techniques for deciding when one activity ended within a time series and the next began, and to contrast this methodology with an HMM approach. The bottom-up approach clearly outperformed the top-down approach in terms of both accuracy and detection time, because our change-point detection algorithms did a poor job of correctly classifying the data. Additionally, we showed that both the detection time and the computation time required by our best performing approaches were low enough for online activity recognition to be feasible.

Bibliography

[1] G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, UK, 1999. Springer-Verlag.

[2] K. Aminian, P. Robert, E. Jequier, and Y. Schutz. Estimation of speed and incline of walking using neural network. *Instrumentation and Measurement, IEEE Transactions on*, 44(3):743–746, 1995.

[3] S.J. Bae, B.M. Mun, and K.Y. Kim. Change-point detection in failure intensity: A case study with repairable artillery systems. *Computers and Industrial Engineering*, 64:11–18, January 2013.

[4] L. Bao and S.S. Intille. Activity recognition from user-annotated acceleration data. pages 1–17. Springer, 2004.

[5] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[6] U. Blanke and B. Schiele. Remember and transfer what you have learned-recognizing composite activities based on activity spotting. In *Wearable Computers (ISWC), 2010 International Symposium on*, pages 1–8. IEEE, 2010.

[7] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. Legrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway, P. Klasnja, K. Koscher, J.A. Landay, J. Lester, D. Wyatt, and D. Haehnel. The mobile sensing platform: An embedded activity recognition system. *Pervasive Computing, IEEE*, 7(2):32–41, 2008.

[8] S.I. de Vries, F.G. Garre, L.H. Engbers, V.H. Hildebrandt, and S. van Buuren. Evaluation of neural networks to identify types of activity using accelerometers. *Medicine & Science in Sports & Exercise*, 43(1):101, 2011.

[9] T.V. Duong, H.H. Bui, D.Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 838–845, June 2005.

[10] J. Fogarty, C. Au, and S.E. Hudson. Sensing from the basement: a feasibility study of unobtrusive and low-cost home activity recognition. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 91–100, New York, NY, USA, 2006. ACM.

[11] K. Grauman. Efficient activity detection with max-subgraph search. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 1274–1281. IEEE Computer Society, 2012.

[12] T. Gu, Z. Wu, X. Tao, H.K. Pung, and J. Lu. Epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In *Pervasive Computing and Communications, IEEE International Conference on*, pages 1–9, 2009.

[13] Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. *Proceedings of the SIAM International Conference on Data Mining*, pages 389–300, 2009.

[14] J.R. Kwapitz, G.M. Weiss, and S. Moore. Activity recognition using cell phone accelerometers. *SIGKDD*, 12(2):74–82, 2010.

[15] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI*, pages 766–772, 2005.

[16] Z. Li. Exercises intensity estimation based on the physical activities healthcare system. In *Communications and Mobile Computing, 2009. CMC'09. WRI International Conference on*, volume 3, pages 132–136. IEEE, 2009.

[17] D. Matteson and N. James. A nonparametric approach for multiple change point analysis of multivariate data. 2012.

[18] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2012. R package version 1.6-1.

[19] V. Moskvina and A.A. Zhigjavsky. An algorithm based on singular-spectrum analysis for change-point detection. *Communication in Statistics. Statistics and Simulations*, 32:319–352, 2003.

[20] D.M. Pober, J. Staudenmayer, C. Raphael, and P.S. Freedson. Development of novel techniques to classify physical activity mode using accelerometers. *Medicine & Science in Sports & Exercise*, 38:1626, 2006.

[21] A. Rai, Z. Yan, D. Chakraborty, T. Wijaya, and K. Aberer. Mining complex activities in the wild via a single smartphone accelerometer. In *Proceedings of the Sixth International Workshop on Knowledge Discovery from Sensor Data*, pages 43–51. ACM, 2012.

[22] N. Ravi, N. Dandekar, P. Mysore, and M. Littman. Activity recognition from accelerometer data. In *In Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence(IAAI)*, pages 1541–1546. AAAI Press, 2005.

[23] B. Ripley. *Feed-forward Neural Networks and Multinomial Log-Linear Models*, 2013. R package version 7.3-6.

[24] M.P. Rothney, M. Neumann, A. Béziat, and K.Y. Chen. An artificial neural network model of energy expenditure using nonintegrated acceleration signals. *Journal of Applied Physiology*, 103(4):1419–1427, 2007.

[25] J. Rowan and E.D. Mynatt. Digital family portrait field trial: Support for aging in place. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 521–530, New York, NY, USA, 2005. ACM.

[26] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pages 566–583. Prentice Hall, third edition, 2010.

[27] W. Shewhart. Quality control charts. *Bell System Technical Journal*, pages 593–603, 1926.

[28] B. Song, N. Vaswani, and A.K. Roy-Chowdhury. Summarization and indexing of human activity sequences. In *IEEE International Conference on Image Processing*, pages 2925–2928, 2006.

[29] Y. Song, S. Shin, S. Kim, D. Lee, and K. Lee. Speed estimation from a tri-axial accelerometer. In *Proceedings of the 29th Annual International Conference of IEEE EMBS*, August 2007.

[30] M. Staudacher, S. Telserb, A. Amannc, H. Hinterhuberb, and M. Ritsch-Marte. A new method for change-point detection developed for on-line analysis of the heart beat variability during sleep. *Statistical Mechanics and its Applications*, 349:582–596, April 2005.

[31] J. Staudenmeyer, D. Pober, S. Crouter, D. Bassett, and P. Freedson. An artificial neural network to estimate physical activity energy expenditure and identify physical activity type from an accelerometer. *Journal of Applied Physiology*, pages 1300–1307, 2009.

[32] C. Strohrmann, H. Harms, G. Tröster, S. Hensler, and R. Müller. Out of the lab and into the woods: kinematic analysis in running using wearable sensors. In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 119–122. ACM, 2011.

[33] M. Sugiyama, S. Nakajima, H. Kashima, P. von Bnau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60:699–746, 2008.

[34] A.G. Tartakovsky, B.L. Rozovskii, R.B. Blazek, and H. Kim. A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. *IEEE Transactions on Signal Processing*, 54:3372–3382, September 2006.

[35] G. Thatte, U. Mitra, and J. Heidemann. Parametric methods for anomaly detection in aggregate traffic. *IEEE/ACM Transactions on Networking*, 19:512–519, April 2011.

[36] T. Therneau, B. Atkinson, and B. Ripley. *Recursive Partitioning*, 2013. R package version 4.1-1.

[37] S.G. Trost, W.K. Wong, K.A. Pfeiffer, and Y. Zheng. Artificial neural networks to predict activity type and energy expenditure in youth. *Medicine and Science in Sports and Exercise*, pages 1801–1809, September 2012.

[38] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, pages 1–9, New York, NY, USA, 2008. ACM.

[39] J. Wang, Z. Cheng, M. Zhang, Y. Zhou, and L. Jing. Design of a situation-aware system for abnormal activity detection of elderly people. In *Proceedings of the 8th international conference on Active Media Technology*, AMT'12, pages 561–571, Berlin, Heidelberg, 2012. Springer-Verlag.

[40] T. Wu, Y.T. Chiang, and J.Y. Hsu. Continuous recognition of daily activities from multiple heterogeneous sensors. In *Proceedings of the 2009 AAAI Spring Symposium on Human Behavior Modeling*, pages 81–85, March 2009.

[41] Y. Zheng. Predicting activity type from accelerometer data. Master's thesis, Oregon State University, August 2012.

[42] Y. Zheng, W.K. Wong, X. Guan, and S. Trost. Physical activity recognition from accelerometer data using a multi-scale ensemble method. In *Twenty-Fifth Annual Conference on Innovative Applications of Artificial Intelligence*. IAAI, 2013.