AN ABSTRACT OF THE THESIS OF

Michael M. Anderson for the degree of Master of Science in Computer Science presented on June 7, 2013.

Title: Activity Detection on Free-Living Data Using Change Point Detection

Abstract approved: ______________________________

Weng-Keen Wong

(Abstract text)

Activity Detection on Free-Living Data Using Change Point Detection

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 7, 2013
Commencement June 2013

Master of Science thesis of Michael M. Anderson presented on June 7, 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

(Acknowledgement text)

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Chapter 1: Introduction

1.1 Motivation

1.2 Previous Research

Chapter 2: Methodology

2.1 Overview

Each dataset that we tested consisted of multiple time series gathered from a number of different subjects, so to perform an experiment on a dataset we began by partitioning the set of time series into disjoint subsets of training, validation, and test data. Each individual time series was then partitioned into a set of disjoint 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. Base classifiers were built with the training set, and tuned (when necessary) on the validation set. To complete the change-point detection experiments (2.1), the tuned model was evaluated with the testing data. For the HMM experiments (2.2), the tuned model made predictions on a second training set, which were used to build an HMM metamodel. Finally, that metamodel was evaluated with the testing data. Following sections describe these processes in detail.

2.2 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 by performing activities themselves instead of using independent subjects. Unfortunately there are not many such labeled free-living datasets available, so we only tested our algorithms on two datasets.

2.2.1 OSU Hip

Our first dataset was collected by the Nutrition and Exercise Sciences department of Oregon State University, and has been used for previous activity detection research with the goal of designing a system to calculate and monitor subjects' energy expenditure [17] [19]. This dataset consisted of 91 time series collected over a 2-week period in a laboratory environment, from 50 subjects who were children between the ages of 5 and

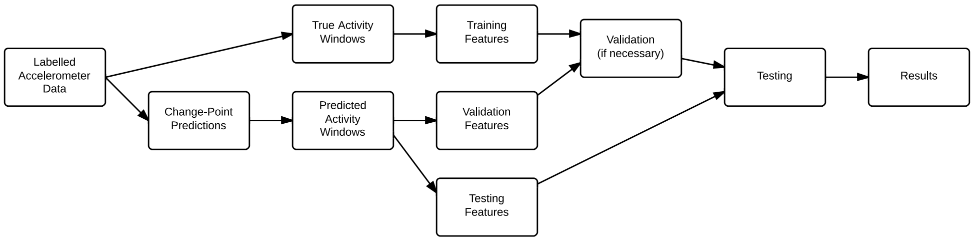


Figure 2.1: Data Lifecycle for Change-Point Detection Experiments

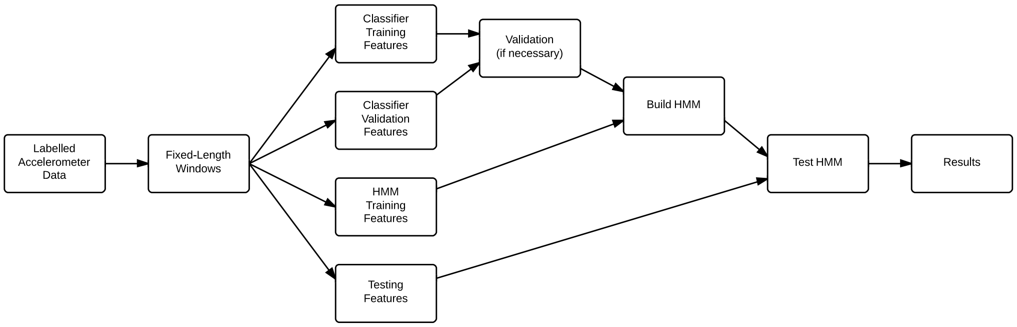

Classifier Training Features
Validation (if necessary)
Classifier Validation Features
Build HMM
Labelled Accelerometer Data
Fixed-Length Windows
Test HMM
Results
HMM Training Features
Testing Features

Figure 2.2: Data Lifecycle for HMM Experiments

15. Subjects performed 12 different types of activities (as shown in Figure 2.3) over two separate visits, while an ActiGraph GT1M accelerometer worn on their hip collected triaxial acceleration data at a frequency of 30Hz.

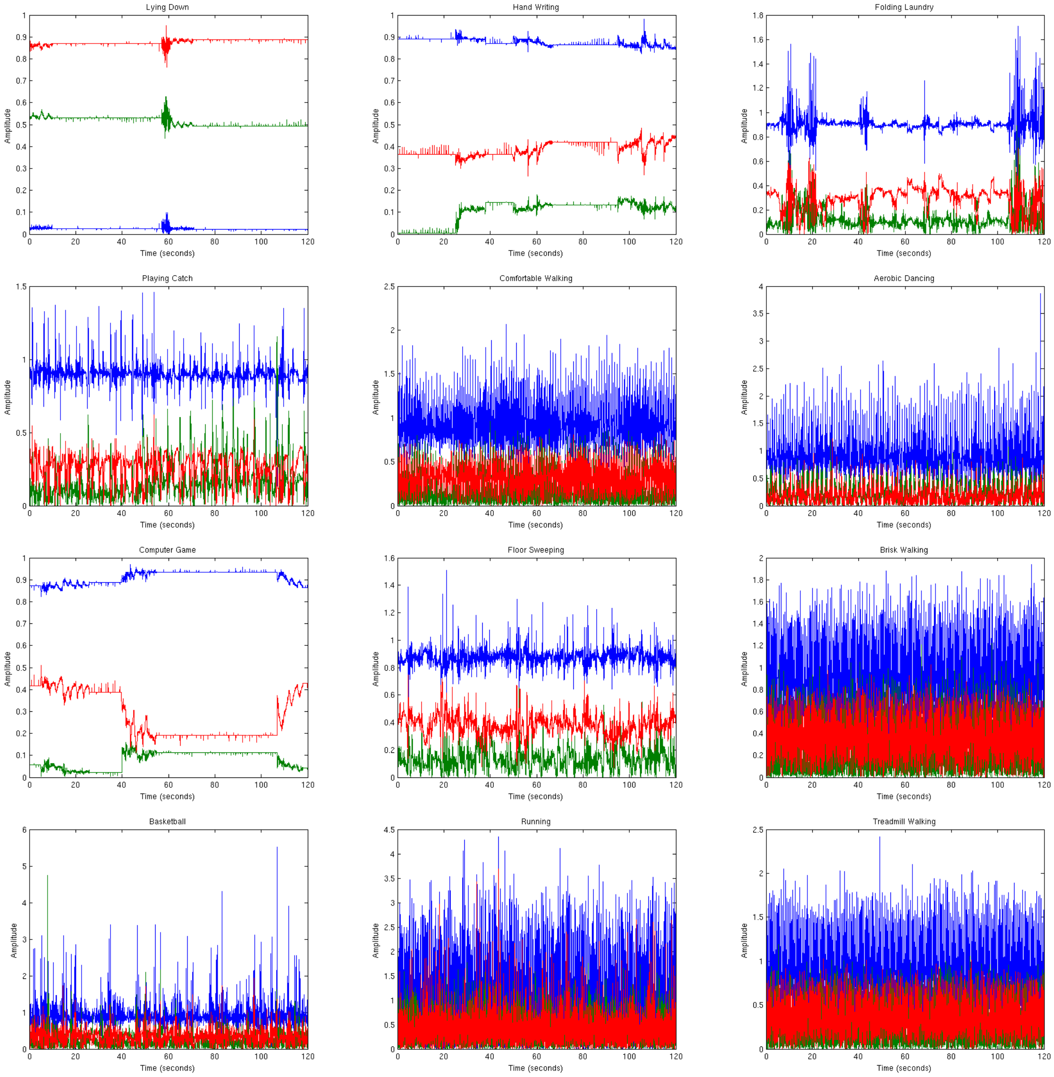
Data was collected from two separate visits to the lab, where the subjects performed 6 activities per visit. The 12 activities were performed in the same order for every other subject. In the version of the dataset available to us, we had all 12 activities of 41 of the subjects, only the first 6 activities 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 that were concatenated together, qualifying the data as synthetic. Each of the 91 time series contained a total of $6 * 120 * 30 = 21600$ data ticks.

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

2.2.2 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. Figure **??** shows samples of accelerometer data from the 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


Lying Down

Hand Writing

Folding Laundry

Playing Catch

Comfortable Walking

Aerobic Dancing

Computer Game

Floor Sweeping

Brisk Walking

Basketball

Running

Treadmill Walking

Figure 2.3: OSU Hip Activity Samples

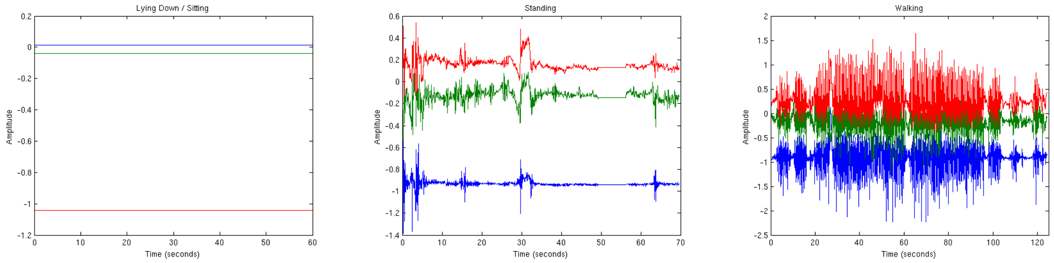


Figure 2.4: LiME Day 1 Activity Samples

each time series into individual days. We then treated the first 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 2977 with a standard deviation of 26428, and the median length was 348. 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 3108 with a standard deviation of 24099, and the median length was 348. 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 (when subjects were sleeping, for example).

2.3 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 2.5 and 2.6. Our feature set was a large collection of statistics that have been shown to be discriminative for activity classification in previous research [3] [8] [12] [19]. 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, and 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.

Features (from [18])

1. Sum of values of a period of time: $\sum_{i=1}^{T} x_i$.

2. Mean: $\mu_s = \frac{1}{T} \sum_{i=1}^{T} x_i$.

3. Standard deviation: $\sigma_x = \sqrt{\frac{1}{T} \sum_{i=1}^{T} (x_i - \mu_x)}$.

4. Coefficients of variation: $c_v = \frac{\sigma_s}{\mu_x}$.

5. Peak-to-peak amplitude: $max\{x_1, ..., x_T\} - min\{x_1, .., x_T\}$.

6-10. Percentiles: $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$.

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

12. Lag-one-autocorrelation: $\frac{\sum_{i=1}^{T-1} (x_i - \mu_x)(x_{i+1} - \mu_x)}{\sum_{i=1}^{T} (x_i - \mu_x)^2}$.

13. 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.

14. 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.

15. Signal power: $\sum_{i=1}^{T} x_i^2$.

16. Log-energy: $\sum_{i=1}^{T} \log(x_i^2)$.

17. Peak intensity: number of signal peak appearances.

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

19. Correlation between each pair of axes: $\frac{\sum_{i=1}^{T} (x_i - \mu_x)(v_i - \mu_v)}{\sqrt{\sum_{i=1}^{T} (x_i - \mu_x) \sum_{j=1}^{T} (v_j - \mu_v)}}$.

One discriminative characteristic of an activity is its overall vigorousness, and the sum and the sample mean both act as simple and obvious ways of measuring this, as more intense activities will tend to involve higher rates of acceleration during movement. We also used the 10th, 25th (quartile 1), 50th (median), 75th (quartile 3), and 90th percentiles of the data, as well as signal power and log energy as supplemental measures of overall activity intensity.

Another characteristic of an activity is how much it varies in intensity. The sample standard deviation, coefficient of variation, peak-to-peak amplitude (max-min), zero crossings (the number of times the data crosses its median), as well as the interquartile range (75th% - 25th%) 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.).

Skewness, kurtosis, lag-one-autocorrelation, and peak intensity were useful for discriminating between activities that tend to be similar in their overall intensity and variation in intensity, but that showed 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 reached its maximum value.

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

2.4 Base Classifiers

We tested 3 types of 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’ [16], ‘e1071’ [5], and ‘nnet’ [7] to build our decision tree, svm, and neural net models, respectively. We treated the decision tree as a simple and quick baseline algorithm, and did not tune it in any way, ignoring the validation set. For all of 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 element of 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 parameters.

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, so 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, so we drew from $N \times W = \{5, 10, 15\} \times \{0, 0.5, 1\}$.

2.5 Top-Down Approach

2.5.1 Change Point Detection

For this approach, the data was split into non-overlapping segments for featurization 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 [1], quick detection of attacks on computer networks [14], and monitoring of heartbeat fluctuations during sleep [11]. Change point detection assumes that each tick of a time series is a draw from some process, but that the process 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 2.5.

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. Where it is reasonable to assume that the data belongs to a particular family of distributions then parametric estimation methods have been employed [15]. If no such modeling assumptions are reasonable then non-parametric methods have also been found to be viable [4]. Distance-based approaches such as Singular Spectrum Analysis generate scores through other metrics of dissimilarity or difference between the reference data and the test data [6]. Notationally, we say that for each tick i in a time series:

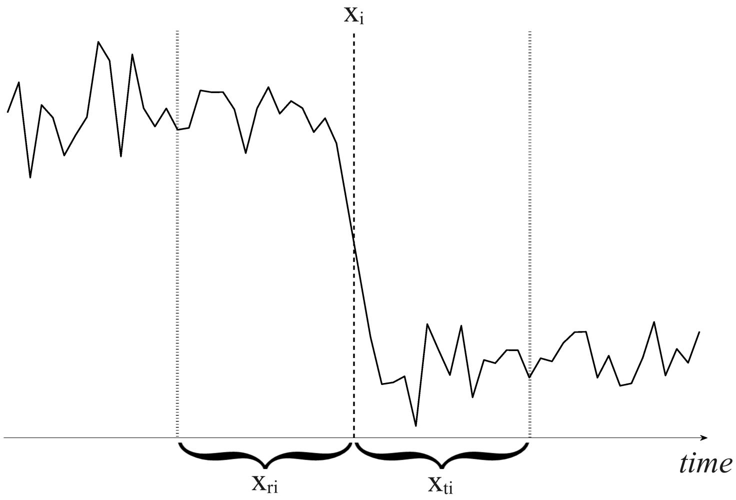


Figure 2.5: Reference and Test Data

$$s_i = D(x_{ri}, x_{ti})$$

Where s_i is the score of the ith tick, x_{ri} is the reference data associated with the ith tick, x_{ti} 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, and 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).

2.5.2 Experimental Setup

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. 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:

$$s_i = \sqrt{(\bar{x}_{ri} - x_i)^T \, S_{ri}^{-1} \, (\bar{x}_{ri} - x_i)}$$

where $\bar{x}_{ri}$ is the sample mean of the reference data, S_{ri} is the sample covariance matrix of the reference data, and $x_i = x_{ti}$ is the ith data point [10].

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), introduced by Kawahara and Sugiyama [2] [13]. 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:

$$\hat{R} = \frac{p_t}{\hat{p}_r} = \sum_{j=1}^{n_t} \alpha_j K_G(x, x_{tj})$$

Where p_r and p_t are the probability densities of the reference data and the test data, n_t is the number of ticks in the test window, α is a vector of model parameters to solve for, x is the concatenation of the reference and the test data, x_{tj} is the jth 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 $p_t = p_r\hat{R}$ is minimized, which is equivalent to the following convex optimization problem:

$$\begin{cases} \max\limits_{\alpha} & \sum\limits_{j=1}^{n_t} \log\left(\sum\limits_{k=1}^{n_t} \alpha_k K_G(x_{tj}, x_{tk})\right) \\ \text{s.t.} & \frac{1}{n_r} \sum\limits_{j=1}^{n_r} \sum\limits_{k=1}^{n_t} \alpha_k K_G(x_{rj}, x_{tk}) = 1 \\ & \text{and } \alpha_1 \ldots \alpha_{n_t} \geq 1 \end{cases}$$

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, 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 io choose the the kernel width at each individual time tick. 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 number of random ticks drawn from the data. From this 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 realtime speeds. Previous research [19] found that a refernce 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 for featurization. A larger false positive rate corresponded to a lower threshold, which split the time series into more segments.

2.6 Bottom-Up Approach

2.6.1 HMMs

Once we had created our methodology for splitting up time series using change- point detection, we decided to test it against a more standard, baseline technique. For this approach, we used the Hidden Markov Model to take advantage of the sequential nature of our data. 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 $O = \{O_1, O_2, \ldots, O_w\}$ (Figure 2.6). An 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 has one of a discrete set of values associated with it drawn from $X = \{X_1, X_2, \ldots, X_\ell\}$, and each observed state has one of a discrete set of values associated with it drawn from $Y = \{Y_1, Y_2, \ldots, Y_m\}$. 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 2.6, the value of an observable state O_i is dependent on the corresponding hidden state H_i, and that the value of any hidden state H_i is dependent only on 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*, $T : X^2 \rightarrow [0, 1]$. 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*, $S : X \times Y \rightarrow [0, 1]$. In other words, both models can be thought of as a function of two values, that outputs the probability of a change from the first value to the second via a dependency arc in the HMM. Now suppose that we are given a *training HMM* $\langle H, O \rangle$, where the values of all of the hidden states as well as the observable states are known. We would estimate T and S by the following:

$$\hat{T}(X_i, X_j) = \frac{|\{H_k \in H \mid H_k = X_i \text{ and } H_{k+1} = X_j\}|}{|\{H_k \in H \mid H_k = X_i\}|}$$

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

Finally, if all of the values of O are known, and we are given a $\hat{T}$ and an $\hat{S}$ estimated

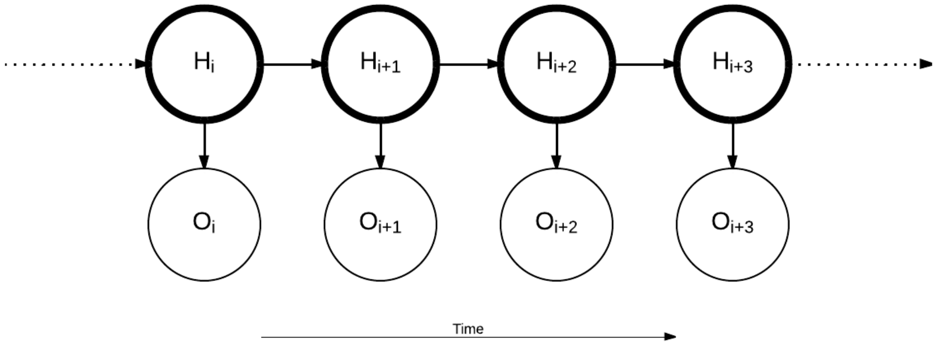


Figure 2.6: Visual Interpretation of an HMM

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 dynamic programming solution to this problem known as the Viterbi algorithm, which has a runtime that is linear in w. [9]

2.6.2 Experimental Setup

For our experiments we began by splitting each time series into small non-overlapping windows. Within a given experiment the window size was fixed, but across different experiments we tested window sizes of length $\{1, 2, \ldots, 20\}$. Once the time series were split they were 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 (training1, validation, training2, and testing) rather than 3. Here we formulated the problem of making predictions on the testing set in terms of an HMM, first by treating the second training set as a training HMM. Each window of the second training set was treated as a time index $1, 2, \ldots, n$. 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. We used the precedure above to calculate $\hat{T}$ and $\hat{S}$, and assumed that these estimates held for the testing set as well as the second 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.

2.7 Performance Metrics

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. 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 defined as the average amount of time required for an algorithm to begin correctly classifying data after a true activity change has occurred.

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 3: Results

3.1 Discussion

Chapter 4: Conclusion

4.1 Summary

4.2 Directions for Further Research

Bibliography

[1] Suk Joo Bae, Byeong Min Mun, and Kyung Yong Kim. Change-point detection in failure intensity: A case study with repairable artillery systems. *Computers and Industrial Engineering*, 64:11–18, January 2013.

[2] Yoshinobu Kawahara and Masashi 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.

[3] 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.

[4] David S. Matteson and Nicholas A. James. A nonparametric approach for multiple change point analysis of multivariate data. 2012.

[5] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2012. R package version 1.6-1.

[6] 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.

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

[8] 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.

[9] Stewart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, pages 566–583. Prentice Hall, third edition, 2010.

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

[11] M. Staudacher and all. 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.

[12] 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.

[13] Masashi Sugiyama and all. Direct importance estimation with model selection and its application to covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60:699–746, 2008.

[14] A.G Tartakovsky and all. 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.

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

[16] Terry Therneau, Beth Atkinson, and Brian Ripley. *Recursive Partitioning*, 2013. R package version 4.1-1.

[17] 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.

[18] 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*. IAAA, 2013.

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