UNSUPERVISED DISCOVERY OF ACTIVITY PRIMITIVES FROM MULTIVARIATE SENSOR DATA

A Thesis
Presented to
The Academic Faculty

by

David C. Minnen

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing, College of Computing

Georgia Institute of Technology
August 2008

UNSUPERVISED DISCOVERY OF ACTIVITY PRIMITIVES FROM MULTIVARIATE SENSOR DATA

ACKNOWLEDGEMENTS

The research presented in this dissertation is the result of a long process of intellectual exploration, development, and refinement that would not have been possible without the support of others. In particular, I would like to thank my advisor, Thad Starner, who introduced me to the world of computer science research when I was an undergraduate and then helped me form my own research direction as a Ph.D. student.

The other members of my thesis committee have also been invaluable assets during my graduate studies. Irfan Essa challenged me to consider how my work connects to related subfields and helped me keep track of my broader goals despite my propensity to focus on algorithmic details. Charles Isbell encouraged me to explore the base assumptions of my work and to consider how the algorithms implied different claims about the world. Aaron Bobick seemed to always ask the deep questions, helping me to understand my research from many perspectives and ensuring that I could not take any intellectual short-cuts. Finally, Bernt Schiele helped push me to be clear about the applicability and limitations of my methods, ultimately leading to a much improved approach.

While my committee provided important guidance for my research, I could not have reached this point without the day-to-day discussions and support of the other students in my lab. I would like to thank Tracy Westeyn, Dan Ashbrook, Kent Lyons, Helene Brashear, and Brad Singletary from Thad's research group as well as Yifan Shi, Adam Feldman, and Raffay Hamid for focused discussions as we each developed our research within the area of activity recognition and discovery. I would also like to thank Chip Mappus, Michael Holmes, Peng Zang, David Roberts, Arya Irani, Liam

MacDermed, and Sooraj Bhat from Charles' lab for broader perspective and machine learning discussions.

Finally, I would like to thank my family for their long-term and unconditional support as I adapted my life, academic, and career goals. My accomplishments would not be possible without the encouragement and facilitation that I have received from my mother and father from my early childhood through today. My sister has long been an important ally in my endeavours by providing perspective and inspiration. Finally, I would like to acknowledge the vital role that Alexis plays in my life. She is a source of joy and passion that drives and supports me.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

This dissertation addresses the problem of discovering unknown activity primitives through the unsupervised analysis of low-level sensor data. These primitives correspond to recurring temporal patterns that support higher-level prediction, reasoning, and modeling. Thus, the goal is to discover semantically meaningful constructs from low-level sensor data using minimal top-down knowledge and few domain-specific constraints.

Existing techniques for detecting unknown recurring patterns, often called "motifs" in the bioinformatics and data mining communities, make restrictive assumptions or have poor computational complexity characteristics that make them impractical for analyzing large data sets. For instance, many algorithms are only applicable to symbolic sequences, do not scale well to multidimensional data, require extensive manual tuning of data-dependent parameters, or require patterns to span all of the input dimensions even for high-dimensional data sets. The contribution of my dissertation is the development of an improved discovery algorithm as measured by overall computational complexity, real-world run time, motif quality, and sensitivity to locally irrelevant input features.

I have developed and evaluated several new approaches to pattern discovery that use different data representations to achieve efficient operation. The first approach globally discretizes the sensor data and builds a suffix tree using Ukkonen's linear-time algorithm to expose recurring patterns [90, 57]. The algorithm works well in some low-dimensional data sets, but sensitivity to the precise quantization boundaries and number of symbols leads to relatively poor performance in more complex domains.

To address these shortcomings, I adopted the method of symbolic aggregate approximation to perform local, context-dependent discretization [50]. I used a random projection method to quickly find similar subsequences within the locally discretized data and developed a method for automatically estimating a similarity threshold for accurately locating additional occurrences [8, 10]. This enhancement increased the modeling flexibility of the approach and reduced the required number of user-specified parameters [59].

To investigate the possibility of efficient pattern discovery without the use of a discrete representation for filtering, I developed an approach based on subsequence density estimation [58]. Consider the space of all subsequences of a time series data set. In this space, the discovery algorithm identifies candidate patterns as those subsequences that lie at local density maxima. To maintain efficiency, density is estimated directly from k-nearest-neighbor distances, which can be quickly determined for large data sets using dual-tree methods [23].

In order to select accurate patterns from the candidates detected at density maxima, I represented each pattern with a probabilistic temporal model and used a competitive framework to iteratively learn a mixture model whose components jointly explain the original data [5]. At each iteration, the pattern is selected which leads to a maximal increase in data log-likelihood when the corresponding model is added to the mixture. The strength of this approach is that it is principled, efficient, and more robust than the previous approaches.

The observed performance boost is mainly due to the probabilistic modeling and competitive selection framework rather than the density estimation procedure used to locate candidates. Therefore, I developed an approach that uses the faster random projection technique to locate candidates and then combines it with probabilistic

modeling and greedy mixture learning to perform the final pattern detection. The result is an approach with run-time characteristics closer to the fastest discovery methods, accuracy rates comparable to algorithms that require far more computational resources, and a reduced reliance on and sensitivity to user-specified parameters.

Finally, I also introduce a variation of the multivariate pattern discovery problem in which each pattern only spans a subset of the dimensions. This formulation is more appropriate for systems such as distributed sensor networks and high-dimensional data sets in which not all of the sensor readings are relevant to every pattern. To address this variation, I developed an algorithm that automatically determines the relevancy of each input dimension on a per-pattern basis, which allows it to ignore spurious or distracting features.

In order to evaluate the empirical performance of my approach, I use it to analyze several data sets spanning different domains and different sensor modalities. The experiments show how the algorithm can discover recurring words from lectures, music, and other speech data sets. When applied to raw accelerometer and gyroscope readings collected by on-body sensors, the algorithm recovers motion primitives, while application to low-level vision features extracted from video of American Sign Language leads to the recovery of repeated signs.

CHAPTER I

INTRODUCTION

Intelligent systems can be categorized along a fundamental dimension that measures the amount of *a priori* knowledge available to the system as it learns to understanding its domain. At one end of this spectrum, a knowledge engineer must fully specify the structure of the domain and all of the relevant information, while at the other end, the designer only provides a minimal bias or representational restriction to allow meaningful learning. My thesis focuses on the design of computational systems that lie near the latter end of this spectrum. That is, they perceive their environment and learn to understand what is happening. My research investigates methods to automatically discover characteristic components of an activity by observing many examples. Through the analysis of activity data, intelligent systems can learn perceptual patterns that are useful for detection, recognition, and prediction of typical behavior within an activity. This analysis allows them to provide useful services to users including helping to form a better understanding of an activity, learning patterns for automatic surveillance systems, providing sensory apparatus for aware environments, and investigating early conceptual development through computational modeling.

Activity data is data that represent information about the behavior of an entity over time. My research focuses on the analysis of human activities, though the ideas presented here are equally applicable to a wider range of time series such as those representing data collected from other creatures (*e.g.,* a bee, ant, dog, or robot) or another system (*e.g.,* neuronal response to a particular stimulus, power usage for a city, automobile traffic density at key intersections, network packet identifiers observed by a router, or stock price fluctuations).

1.1 Background

The behavior of humans can be captured in a variety of ways using sensors either worn by the subject or embedded in the environment. For example, cameras mounted in the ceiling can capture the movement and interactions of people in a home or office, simple circuits built in to doors, cabinets, or drawers can detect when an inhabitant uses these fixtures, and microphones can detect a subject's speech and even their location if multiple microphones are arranged as a phased array.

Users can also wear sensors that may provide more precise or personal information. Examples of wearable sensors include accelerometers that directly measure the acceleration of the wearer's body or limbs, gyroscopes that record rotational motion, galvanic skin response (GSR) sensors, and body-mounted cameras or microphones that record the user's first-person view of the world.

The idea of activity discovery arose out of existing research in activity recognition, monitoring, and prediction. In activity recognition, as in the general case of temporal sequence recognition and time series prediction, a learner is provided with annotated examples of an activity and seeks to learn a model. This model may take many forms, for example, state machines, grammars, or rule sets, as well as probabilistic generalizations therein. Success is measured by applying the learned model to previously unseen test sequences and comparing the detected activity (and in some cases, the detected sub-components) with ground truth labels. A more detailed examination of previous research in activity recognition will be given in Section 2.1, but the key difference between activity recognition and my research is that my goal is to *discover* embedded activity primitives and higher-level structure via *unsupervised* analysis. In other words, I seek to learn models without the help of previously segmented and labeled activity components, while research in activity recognition depends on such labeled training data for learning.

For the purposes of this dissertation, an *activity* is a general term for the extended behavior of a person or system. The behavior could be temporally or spatially extended. For example, compare COOKING_AN_OMELETTE or PLAYING_BASEBALL with the nucleotide sequence in DNA or with the density of a pollutant at various points along a river. In the latter two cases, the sequence is defined through space rather than through time by either detecting the nucleotides along the DNA or by inspecting the pollution level at successive locations along a river. Despite the incongruous description when applied to some data sets, I will continue to use the term *activity* for brevity and due to my focus on analyzing human activity data.

Naturally, my research only pertains to directly observable activities, though it should be clear that observability is relative to the sensors at hand. For example, RUNNING, WALKING, and SITTING are, barring occlusion and insufficient acuity, observable by a video camera, whereas THINKING and FEELING are not. However, with other sensor, THINKING, or at least some kinds of brain activity, become directly observable as with near-infrared (FNIR), electroencephalography (EEG), and functional magnetic resonance imaging (fMRI) machines.

Given an observable activity, *activity data* is a characterization of the activity through time or space as perceived by a sensor or set of sensors. Thus, activity data may be symbolic or continuous and may be univariate or multivariate. The approach proposed here can operate over data of any of these types, though I will focus on the multivariate, continuous case due to its generality. I do, however, make the simplifying assumption that the input data has already been (pre-)processed to include appropriate features, and, in the case of multivariate data sets, that the various features have already been temporally aligned. The issue of feature extraction, transformation, and synchronization is a central issue for all machine learning methods but is not addressed explicitly by my research. Therefore, for each experimental domain that will be discussed, I will explain any feature extraction processing used, and I have made a great effort to use either raw sensor values or standard features for the domain (*e.g.,* frequency spectrum amplitudes for audio or tracked "blob" characteristics for gesture-based vision applications).

The concept of a *motif* is important to my formulation of activity discovery. A motif is a set of recurring subsequences embedded in a larger sequence for which the occurrences exhibit high similarity. When the data represents an activity (in the narrow sense), I will refer to such motifs as *activity components* or *primitive actions* depending on whether the motif can reasonably be subdivided given the structure of the activity and the precision and temporal resolution of the activity data. More generally, the motifs are simply *recurring patterns* in the data set, though this phrase refers to a broader set of patterns and problems than just motif detection.

My research involves two kinds of multivariate motifs. The first kind corresponds to a set of recurring patterns that extend across all of the dimensions of the given time series data. These motifs are sensible when the multivariate signal comprises many features extracted from a single sensor channel (*e.g.,* MFCCs extracted from waveform data) or when the multivariate sensor readings are strongly coupled (*e.g.,* readings from a three-axis accelerometer). The second kind of motif, called a "subdimensional motif," corresponds to a recurring pattern which only includes a subset of the sensor channels. Such patterns may arise from distributed or uncorrelated sensor systems. For instance, an on-body, multi-sensor system may capture different primitive actions from sensors placed near the ankle and on the wrist. Similarly, an EEG recording may contain different patterns across different subsets of the electrode signals. Finally, a geographically distributed sensor system may contain sensors that are too distant to give rise to a meaningful motif, while spatially proximal sensors do record joint motifs.

For the purposes of this work, I define *activity discovery* as the discovery of motifs in activity data (see Figure 1) as well as learning higher-level structure over the discovered motifs, if such structure exists. As a practical matter, this dissertation

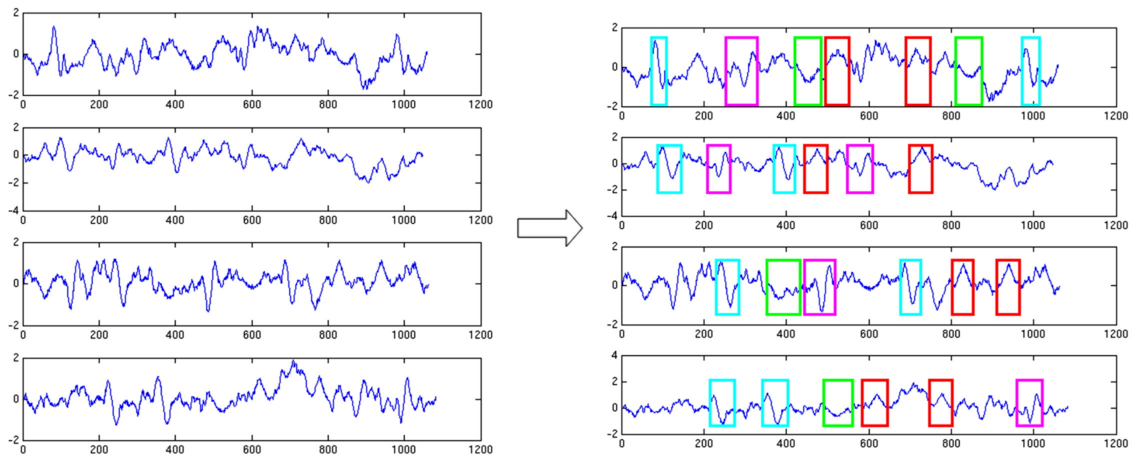

Figure 1: An illustration of the sparse motif discovery problem for four 1D time series. Rectangles correspond to motif occurrences and each color represents a different motif.

and the included empirical evaluation only covers specific measures of similarity and certain kinds of structure, but it should be clear that the approach allows a wide range of models and metrics.

Note that my formulation is only one possible definition for activity discovery. Specifically, it relies on the "signature" of the motif, that is, the space in the relevant feature space carved out (perhaps with appropriate time-warping) by the occurrences of the motif. Contrast this formulation with activities that are defined by a goal in which the particular signature is not important (see Section 2.5) or with patterns characterized by event occurrences and the time between them as with neurological spike trains.

Since a motif is a collection of occurrences, the properties of these occurrences will have a huge impact on how easy or difficult they are to discover. Previous research has assumed fixed-length occurrences, used distance metrics that assume a uniform temporal progression to measure the similarity between time series, and specified a single distance threshold to define the neighborhood size of all motifs (see Section 2.4 for details). Such assumptions are too restrictive for many activity data sets, and so they are relaxed in this work to allow variable-length occurrences and similarity based on hidden Markov models (HMMs), which allow time-warping for matching and modeling.

Finally, I am interested in discovering motifs in a wide range of data sets which may contain different numbers of motifs and different numbers of occurrences. In some data sets, such as in typical speech or American Sign Language (ASL) data where motifs correspond to words or signs, the motifs are *densely* distributed within each time series. Here, a dense distribution means that the majority of the data frames fall within an occurrence of one of the motifs. On the other hand, as in data representing activities of daily living and many sports, the motifs may occur relatively infrequently and thus be *sparsely* distributed within the time series. In this case, more efficient discovery algorithm is needed since most of the data may occur outside of any of the motifs, and therefore it will be more difficult to find any valid motif occurrences. Since I am interested in motif discovery that covers both cases, the approach presented here is largely motivated by quickly locating candidate patterns so that it performs efficiently for both sparse and dense data sets.

1.2 Motivation

The unsupervised analysis of activity data can be motivated on both practical and theoretical grounds. On the practical side, an activity discovery system can help verify an expert's intuition concerning the components of an activity. It can also find new components or provide a decomposition when an expert is not available either due to practical complications (*e.g.,* scarcity, scheduling conflicts, low funds, *etc.*) or simply because the data comes from a new field that is not yet well understood (*e.g.,* DNA sequences or spike responses in networks of neurons).

For users interested in building automated surveillance systems or robust perceptual subsystems for intelligent spaces, a more important motivation comes from the fact that discovered activity components are necessarily detectable. In contrast, human specified components, which are currently used in most approaches in the literature, may or may not be reliably detectable by a particular system and set of sensors. Discovered motifs are necessarily detectable due to the simple fact that a motif that can not be detected can not be discovered, and any motif that was discovered must have be detected as either a candidate seed or as a subsequent motif occurrence. The importance of this difference is highlighted by ideas from embodied cognition in which it is theorized that entities conceptualize their environment relative to their ability to sense and act in the world. Thus, a human who interacts with the world through their hands and feet and who senses primarily through sight, sound, and touch, may identify certain activity components as critical, whereas an artificial system that uses sonar or accelerometers may find that an entirely different set of primitive actions more robustly characterize the same activity.

The final motivation for activity discovery is philosophical and builds on the position put forth by Cohen *et al.* [13]. In that work, the authors argue that "most of the intellectual work in AI is done not by programs but by their creators, and virtually all the work involved in specifying the meanings of representations is done by people, not programs." They adopt Fred Dretske's theory of meaningful representation, which posits that an internal state must both indicate a condition and have the *function of indicating* that condition. Applying this theory to robotics, Cohen *et al.* conclude that learning meaningful representations requires a robot to analyze its sensor readings to learn abstractions that inform action. Similarly, I seek to design an activity discovery system that learns endogenously meaningful representations via unsupervised analysis of sensor data. Since I am primarily interested in wearable systems and intelligent spaces, however, the function of indicating (*i.e.,* the action triggered by a learned state) is not driving robotic actuators. Instead, the function of indicating is to provide useful information and robust services to the user. Importantly, the final output must be meaningful to the user (for how else could it be understood or evaluated?), but the internal states that serve to robustly characterize the data and allow accurate detection of the desired state are learned and need only be meaningful to the system. Thus, I see activity discovery as a method for acquiring endogenously meaningful perceptual concepts, a long-standing goal in artificial intelligence.

1.3 Example Domains

Although the motifs found by an activity discovery system need not correspond to those expected by a human analyst, it is still useful to motivate activity discovery with examples from domains with well-known primitives. For instance, in speech data a discovery system can be reasonably expected to locate either words or phonemes depending on the time-scale specified by the user. Similarly, sign language is comprised of many elemental signs, exercise routines consist of several repetitions of a basic, strenuous movement, and Kung Fu forms are typically built by combining different punches and kicks. In assembly tasks, we expect that the characteristic motions necessary to effectively use tools such as hammers, screwdrivers, hand saws, drills, and sanders will correspond to discoverable primitives. Similarly, economists have identified a range of so-called "chart patterns" including *flags*, *pennants*, *ascending triangles*, and the *head and shoulders pattern*. An effective discovery system should be able to rediscover these patterns, although this may require simultaneous analysis at several time-scales. Finally, some spatial data sets can be viewed as time series data and thus analyzed by an activity discovery system. For example, lines of handwritten text could be converted by scanning each line and extracting features from the vertical columns. Analysis of the resulting sequence would likely lead to primitives corresponding to either characters or words depending on the scale.

1.4 Thesis Statement and Contributions

The thesis of this dissertation is:

Semantically interpretable activity primitives can be automatically and efficiently discovered in multivariate activity data by identifying recurring motifs with high intra-motif similarity.

There are three main contributions of my research. First, I will demonstrate that discovering recurring patterns is feasible in interesting domains and realistic data sets. For instance, I have evaluated the performance of my approach on speech, American Sign Language, music, GPS tracks, rendered text, and on-body inertial sensor data (details can be found in Chapter 4). These domains are particularly interesting because they have been the subject of study for supervised learning methods within the pattern recognition community and because they all have higher-level structure that influences the temporal relationship between the primitive actions. Broad experiments across a range of different domains are important to demonstrate and evaluate the generality of an approach and to help avoid including unintended, domain-specific biases into the algorithm.

The second contribution is the development of a new approach to activity discovery that improves on existing methods from the bioinformatics and temporal data mining literature. For instance, many existing algorithms are only applicable to symbolic sequences, do not scale well to multidimensional data, require manual tuning of data-dependent parameters, or have poor run time characteristics. Correspondingly, the improvements of my approach relate to the algorithm's computational complexity, expected real-world run time, and range of applicability, as well as to the quality of the discovered motifs. More detailed information concerning the evaluation of activity discovery systems will be discussed in Section 4.1.

The third main contribution of my research is the development of an efficient algorithm for discovering subdimensional motifs. To my knowledge, this problem has not been directly addressed before in the literature, although existing methods could be adapted through a (generally inefficient) meta-search for applicable dimensions or by thresholding weights learned by additive factorization methods (described in Section 2.6). My approach adopts the same random projection framework used by the "all-dimensional" algorithm but modifies it to be robust to irrelevant dimensions without requiring an explicit search over the possible feature combinations.

CHAPTER II

RELATED WORK

2.1 Activity Recognition

As discussed previously, activity recognition is a supervised analogue of activity discovery. In typical approaches, an agent is first given labeled observations of a temporally extended activity and must learn a model from these training examples. The agent is then tested on previously unseen observation sequences and must detect the occurrence of a known activity and must often also segment and identify subcomponents.

Much research in this area has focused on learning a probabilistic model to represent the activity and then sought to match the model to new sequences. For example, HMMs have been used to model and recognize gestures [6, 82], American Sign Language [87, 92], and other actions [7, 63, 97]. Other researchers have shown how such state-based methods fail to capture the extended semantics of certain activities (at least without introducing an intractable number of states) and have turned to more complex models. Stochastic context-free grammars (SCFGs) offer one such model that extends the efficient representational power of HMMs [64, 33, 56]. To address the problem of concurrent, interleaved streams of activities Shi *et al.* developed Propagation Networks (P-Nets), which generalize HMMs and SCFGs with a network structure and allow multiple nodes to be "active" simultaneously [84].

A different form of activity recognition arose out of tracking research in which the state dynamics includes a discrete hidden variable which represents the current activity or behavior regime. Typically, the recognized activities are distinguishable by their dynamics, which are conditioned on the activity itself. Thus, the techniques adopt a switching-state algorithm in which the hidden activity variable is chosen to best account for the currently observed dynamics, perhaps also subject to direct filtering to promote temporal cohesion and known transition constraints. As an example of tracking-based activity recognition, Black and Jepson developed a vision-based, interactive white board system that tracked the position of a hand-held pen or eraser [4]. The object dynamics were conditioned on the gesture class, and particle filtering [31, 19] was used to simultaneously explore multiple class and location hypotheses. Similarly, Patterson *et al.* tracked people carrying a GPS sensor around a city and recognized their activity (WALKING, RIDING-THE-BUS, or DRIVING) using a similarly augmented particle filter [76].

Other researchers have used a variety of methods to recognize the current activity of the user. Typically, this assumes that only one activity can take place at each time step and little higher-level structure is utilized (though in some cases, a higher-level model is used to smooth the results of the frame-based classifier). For instance, Lukowicz *et al.*[55] recognize wood workshop activities using a variety of on-body sensors including microphones and accelerometers. They classified the incoming sensor data using linear discriminant analysis (LDA) for dimensionality reduction and HMMs. Wyatt *et al.* extracted hundreds of features from an on-body sensor package and used a boosting framework to select the features that were most discriminative for a set of standard daily activities [47]. Liao *et al.* improved upon these results by using a conditional random field (CRF) [43] along with a similar boosting framework to learn discriminative features while considering temporal context [48].

2.2 Time Series Clustering

Time series clustering is related to activity discovery since the solutions to both problems lead to sets of similar time series. While both problems require unsupervised learning, the key difference is that time series clustering assumes that the time series are segmented (or nearly segmented) and thus each sequence belongs to exactly one cluster. In motif discovery, however, each time series may contain multiple subsequences that are occurrences of different motifs. There is a huge literature on time series clustering and the references given here are just a small sampling of recent work on the problem (see [49, 1, 22, 61, 70, 51]).

2.3 Scene Detection

Research in the computer vision and multi-modal pattern recognition communities has focused on dense discovery in work that can broadly be called shot detection or scene boundary detection. A particularly nice example is the work of Xie *et al.* in which the parameters of a hierarchical hidden Markov model (HHMM) are estimated in an unsupervised manner to learn different segments of soccer and baseball games [96]. The games are analyzed based on both video and audio features, and the hierarchical model naturally accounts for both low-level and high-level structure. This research is specifically targeted at recovering dense structure, however, although it may be possible to identify one of the higher-level states as "background" and then separate valid motifs from "background frames." Further theoretical and empirical investigation is needed to determine if the HHMM successfully recovers motifs and if an unsupervised procedure can distinguish between motifs and background data.

Earlier work by Clarkson and Pentland explicitly used a two-layer HMM to model audio and video data captured using on-body sensors [11]. Low-level transitions between the states of each HMM captured local information, while transitions between HMMs captured scene boundaries. Although this was important early work, the model is subsumed by the more general hierarchical HMM formulation.

Similarly, research by Naphade and Huang [68] segmented video by learning a mixture of HMMs. The high-level model used in their research is also a Markov model, essentially leading to a simple HHMM structure overall. The parameters of the model were learned via EM after random initialization. This process leads to reasonable segmentation but requires a relatively long training period and makes it difficult for the component models to isolate short recurring scenes.

Huynh and Schiele take a different approach to activity discovery which tries to explain the time series data in terms of multiple low-dimensional eigenspaces [30]. Their approach uses a multiple eigenspaces algorithm [46] to automatically learn the number of subspaces, the projection vectors for each eigenspace, and the boundaries between different regions. The result is that the time series data is clustered based on the discovered boundaries and the corresponding subspace that best explains each region.

2.4 Motif Discovery

In bioinformatics, systems such as MEME were developed to discover motifs in DNA and protein sequences [3]. Many other specialized systems have been developed since then, though few are applicable to real-valued time series analysis since they were designed to work with categorical sequences (see [34] for a brief review). Recently, Jensen *et al.* generalized motif discovery over both categorical and continuous data and across arbitrary similarity metrics [34]. This research represents an important improvement, but their approach has two significant issues. First, it requires a pairwise comparison of all subsequences of a particular length, but this is a quadratic operation which precludes scaling to large data sets. Second, although they do support variable length motif occurrences, this is only achieved via a post-process. In sequence data this is probably not a major problem, but in time series, two very similar subsequences may appear quite different if only fixed-length segments are compared (*e.g.,* consider comparing `ABCD` and `AABBCCDD` with a window length of four).

Within the data mining community, Chiu *et al.* developed an efficient, probabilistic algorithm for motif discovery using a form of locality-sensitive hashing [10]. This approach only discovers fixed-length motifs, however. Their approach gains its efficiency by first searching for potential motifs amongst a discrete representation of the subsequences. Quantization is performed by symbolic aggregate approximation [50] (SAX, see Section 3.3.1 for details). Although the original presentation only worked with univariate data, SAX can also be applied to multivariate time series by concatenating the "words" representing each dimension [38]. This method, along with my improvement that allows automatic estimation of a motif-specific neighborhood size, is presented in more detail in Section 3.3.

Tanaka and Uehara generalized the approach of Chiu *et al.* to work with multivariate time series and to allow variable length motifs [88]. Their solution is to apply a univariate algorithm to the first principal component of the time series. Unfortunately, the first principal component will often not retain enough information about the original multivariate sequence to allow differentiation between different actions. Their approach also allows variable-length motifs (called *Different Length* or DL patterns in the paper). This is achieved by first using SAX to discretize short windows and then assigning a unique symbol to each SAX string. Although this appears to work well for the data sets presented in the paper, the number of unique symbols could grow exponentially in the length of the SAX string. Therefore, further empirical evaluation is needed to ensure that the number of unique symbols remains manageable for a wider range of data sets and time scales.

More recently, Yankov *et al.* extend Chiu's method to handle small variations in the duration of motif occurrences (though not variable-length motifs *per se*) by using a *uniform scaling* distance metric instead of Euclidean distance [98]. Their approach uses the same random projection algorithm to identify candidate motifs in univariate time series but then searches over a small range of similar durations to identify other motif occurrences. Their method also seeks to reduce the number of user-specified parameters by estimating some of the parameter values based on a separate, unlabeled training set. Inference from the unlabeled training set is possible within the context of the nearest-neighbor motif discovery problem, which only seeks to find the k most similar occurrences, instead of all of the occurrences.

The PERUSE algorithm discovers motifs directly in multivariate time series and allows both non-linear time warping and variable-length motifs [69]. It uses a model similar to a left-right HMM with an explicit, Gaussian distribution over the time between each sample. This additional parameter allows the model to map on to data that was not uniformly sampled and to skip over shot noise when necessary. PERUSE discovers motifs by positing a hidden multinomial distribution over all data frames that specifies the probability that an occurrence of the current motif ends at that frame. Expectation-maximization (EM) is used to iteratively estimate this multinomial along with the motif's model parameters. Impressively, PERUSE was able to learn many of the repeated words in a speech data set and was also able to learn "episodes" from robot sensor data. The algorithm was developed for data with a dense motif structure, an assumption that holds for both the speech and robot sensor series. However, although the algorithm is potentially applicable to sparse data, it scales quite poorly due to the initialization procedure and the need to re-estimate the hidden multinomial at all frames during each iteration. Also, in my experiments, the EM iterations often found trivial local maxima in which only a single, very accurate occurrence was found, rather than finding the appropriate set of approximate matches.

Catalano, Armstrong, and Oates focused on a different aspect of the motif discovery problem by developing a streaming discovery algorithm [9]. Their method runs in linear time relative to the number of sensor readings and only requires a fixed amount of memory since it does not analyze all of the data at once. To achieve these run-time properties, the algorithm essentially assumes that the motifs are densely distributed. Otherwise, a very large memory would be required to detect a motif that only occurs rarely and after relatively long intervals.

Hamid *et al.* proposed an approach for activity discovery that used variable memory Markov chains (VMMCs) to detect motifs in event streams [27]. My approach to finding seed motifs using a globally discretize representation of the time series data (see Section 3.2.1) shares much in common with this method but also allows time-warping, though it operates over data at a much lower-level of abstraction compared to the semantically meaningful events used in the work of Hamid *et al.* In addition, Hamid *et al.* used histograms of event n-grams as a representation to compute event sequence similarity, whereas my research focuses on multivariate sensor data (or extracted features) and thus uses direct, numerical measures for similarity. In more recent work, Hamid *et al.* improve this representation by using suffix trees to extract common, variable-length event sequences from which to build a histogram representation for each activity sequence [25]. This approach leads to more accurate models compared to the fixed n-grams used previously.

Park and Glass focused on the problem of pattern discovery in the speech domain [72, 73, 71]. Their research takes a different approach compared to much of the other motif discovery work due to their focus on human speech and the subsequent ability to take advantage of the specific properties of speech signals. In particular, the researchers pre-segment audio data by removing silent regions detected by analyzing the log-energy of the audio spectrum. Any segment below a given threshold is considered silence. The authors show that for typical speech data sets, the result after silence removal is a large set of relatively short speech segments, generally lasting between one and 10 seconds. Their algorithm then computes a "segmental dynamic time warp" between each pair of segments and searches for low distortion regions that correspond to similar utterances. Each segment that is part of such a pair with low distortion becomes a node in a graph with weighted edges representing the amount of distortion. Finally, to discover words and short phrases, a graph clustering algorithm is applied to the segment graph.

The main drawback of Park's approach is that it relies on the ability to pre-segment the data. While silence removal is quite reliable for human speech recorded in a quiet environment, similar heuristics may not exist in other domains. Trying to develop a domain-general segmentation rule is even more difficult. Even within the audio domain, other data sets such as music or recordings made in noisy environments can not be easily split into segments amenable to pair-wise comparisons. The computational complexity of the algorithm is also a concern since there are either $O(T^2)$ segments or, if there are fewer segments, the segmental DTW procedure scales as $O(N^2)$ where N is the length of the longest segment.

2.5 Other Forms of Activity Discovery

Other research is able to discover information about a subject's activity by directly analyzing descriptive information about the activity or by using other specialized data. For example, by clustering entries on a calendar or "To Do" list, especially when temporal regularities are detected from the clusters, a person's activities can be easily inferred.

Ashbrook and Starner collect data from GPS devices carried by subjects as they move through their daily lives [2]. Points are selected based on GPS signal loss, indicating that the person went inside of a building or some other structure. These points are clustered to learn "significant locations" using an algorithm similar to mean shift [14] with a uniform, circular kernel. The significant locations are then used to learn a Markov model that can be used to predict the user's future behavior. The predictive user model allows them to discover activities such as going to the grocery and to learn that the person will likely return home after finishing at the store. What differentiates my work from this kind of activity discovery is a relaxation of required meta-information. Discovery in Ashbrook and Starner's work is 2D spatial clustering. What makes it activity discovery is the observation that GPS signal loss indicates entering a building and the inference, made by the researcher rather than by the system, that activities such as `grocery_shopping` are robustly predicted by GPS signal loss in the vicinity of a grocery store.

2.6 Additive Component Models

The motifs of a time series can be viewed as components or as a basis set that combine to generate the time series data. I have assumed that activity primitives are temporally distinct components, and so a simple but sufficient generating process (from Oates [69]) is to randomly select a motif (or the background / non-motif model), sample from it to get an occurrence, append the occurrence to the data generated so far, and then repeat. Researchers focused on other domains have made different assumptions. For example audio signals are typically additive rather than temporally distinct, leading to an additive component model. In this domain, Smaragdis and Raj adopted an additive model for single-channel source discovery and separation [83, 86]. Similarly, Parry and Essa use repetitive structure in audio signals for blind source separation [74]. Within the field of neuromotor physiology, researchers have also adopted an additive component model. For instance, d'Avella and Bizzi discovered muscle activation components, which they call *muscle synergies*, by analyzing activation levels across several electrodes implanted in frogs [17]. They found that some of these *synergies* were stable across activities (*e.g.,* walking, hopping, and swimming) and across different frogs.

2.7 Anomaly Detection

A related form of unsupervised learning from activity data deals with discovering anomalous behavior. Whereas my goal is to discover those motifs that correspond to semantically interpretable or otherwise useful recurring components of an activity, in anomaly detection the goal is to find those events or subsequences that are most dissimilar, unexpected, or surprising. Typically, this dissimilarity is measured relative to a model built from the current data set or to another data set representing the same activity but without the anomalies. Note, however, that recurring patterns can be considered a kind of anomaly in that the similarity between and frequency of the occurrences are anomalously high [28].

In a representative research paper of anomaly detection within the computer vision community, Zhong *et al.* analyze video and automatically detect unusual intervals [100]. They divide the video into equal length intervals and divide each frame into a grid over which color and motion histograms are computed. Prototypes are learned over the frames using k-means clustering and then co-occurrence statistics are calculated between intervals and prototypes. Finally, a similarity metric between video segments is inferred by embedding the prototypes and segments in a low-dimensional space that preserves the co-occurrence information. Anomalies are then identified as isolated intervals in this induced space.

Using a different approach, Hamid *et al.* treat a video sequence as a string of manually defined and labeled events that are relevant to the activity [26]. An activity is then modeled as a histogram of n-grams of such events, which would encode global properties of the local temporal structure. For example, a histogram of trigrams would capture the relative frequency of all event triplets. Activity instances are then clustered to find sub-class structure and anomalies are defined as outliers of the closest sub-class model.

CHAPTER III

DISCOVERING ACTIVITY PRIMITIVES

I have investigated multiple approaches to activity discovery in order to compare the benefits of different methods and to explore various trade-offs between accuracy, efficiency, and generality. The purpose of this exploration was to use the empirical evaluations to guide the development of a single algorithm that incorporates and improves upon the best techniques from existing approaches. In this chapter, I will describe my approach to motif discovery and discuss several implementations that fit within this framework.

3.1 Discovery Framework

The central design characteristic of my approach to motif discovery is a split of the algorithm into two phases. The goal of the first phase is to quickly locate *candidate motif seeds* and, equivalently, to efficiently filter out the potentially large number of subsequences that are not part of any pattern. The second phase serves a complimentary role by performing a detailed analysis of the candidate motif seeds to determine which are valid and where additional occurrences are located in the data set.

Many discovery algorithms can be understood in terms of this two-part framework, including several algorithms that were not originally presented in this way. For instance, Oates' PERUSE algorithm [69] does not include an explicit filtering phase, although in practice one typically samples the potential motif locations to find the next pattern. This sampling serves the same purpose as a filter, but it does not depend on the data and so is unlikely to be particularly efficient.

Viewing existing algorithms in terms of this two-phase framework can make it easier to combine algorithmic components from different approaches to form improved algorithms. In this chapter, I discuss four different complete, published algorithms [57, 10, 59, 58] and then present an improved algorithm that combines components from the first four. An empirical evaluation of all of the algorithms on several different data sets is presented in the following chapter.

3.1.1 Phase 1: Efficient Pattern Filtering

In the initial stage of pattern discovery, the goal is to filter out regions of the time series data that are not part of any pattern. In particular, the algorithm must be very efficient since it is analyzing the entire data set, but it need not be extremely accurate. In particular, to be successful the algorithm only needs to retain two occurrences of each pattern whereas maintaining non-pattern regions reduces efficiency but will not cause the overall algorithm to fail. That is, false negatives are acceptable as long as enough occurrences are kept to allow model learning in the second phase, and false positives only decrease efficiency not accuracy.

I have investigated three different methods for pattern filtering, which are briefly described here and explained and evaluated in detail within the context of the full algorithm.

1. **Global Discretization & Suffix Trees** – In this approach, the real-valued, multivariate time series data is discretized so that each sequence becomes a string composed of symbols from a small alphabet. This representation provides dimensionality reduction by converting a multivariate sequence to a univariate sequence. More importantly, the discretization provides a simple, unequivocal measure of similarity. The suffix tree [24] data structure, along with a linear-time construction algorithm [90], exploits this direct measure of equivalence. Variable-length recurring patterns can then be detected in the suffix tree with a simple, linear traversal.

2. **Local Discretization & Random Projection** – Local discretization methods take into account the surrounding values in the real-valued signal when converting a particular reading into a discrete symbol. Instead of transforming each time series into a single string, these methods discretize overlapping subsequences extracted by a sliding window and thus create it words that can be arranged in a matrix of symbols that represents the original data (see Figure 10).

   Global discretization leads to a representation that allows very efficient, direct detection of variable-length patterns using suffix trees. Local methods are restricted due to the fixed length window used to construct each word but have the advantage of more sensitive symbols due to the contextual interpretation. In addition, random projection algorithms can be used to detect approximate patterns in linear time whereas the suffix tree approach is restricted to exact matches or simple warpings.

3. **Density-based Detection** – The third approach for pattern filtering that I have investigated does not require any discretization of the time series data to achieve efficient performance. It is based on the observation that since motifs are sets of similar subsequences, well-supported motifs in a particular data set will have many occurrences that are all very close together. The combined effect of set size and compactness is captured directly by the concept of subsequence density. Thus, the density-based detection method estimates the density around each subsequence using a k-nearest neighbor approximation and provides candidate patterns by selecting those subsequence that lie at local density maxima.

3.1.2 Phase 2: Pattern Modeling and Occurrence Detection

Along with the different approaches to pattern filtering, I investigated two pattern models and corresponding detection methods. The requirements for successful algorithms in this phase are complimentary to those of the earlier filter. Here, the system must learn a model for each candidate pattern, select the valid patterns, and detect all of the occurrences in the data set. Because the output of this phase constitutes the final decision of the overall algorithm, the results must be as accurate as possible. Thus, emphasis is placed on minimizing errors rather than minimizing execution time, although the system is still bound by our design goal of sub-quadratic complexity.

1. **Hypersphere Model** – The hypersphere model is a prototype-based model for motifs. It provides a measure of confidence that a specific subsequence is a member of the pattern based on the distance between the subsequence and the central prototype (or to a small set of prototypes). Subsequences that are farther from the protoype(s) are less likely to be valid members, and typically a hard decision is made based on a distance threshold [10, 89, 59].

2. **Probabilistic Temporal Model** – An alternative approach for modeling patterns is to use a probabilistic temporal model. In the case of generative models like an HMM, the model induces a probability distribution over all possible subsequences. To detect occurrences, one can threshold this probability or can estimate a model for each potential pattern and simultaneously fit all of the models to the data according to some objective such as maximum likelihood. The primary benefit of the latter approach is that the precise temporal boundaries of the occurrences are determined by the models within a competitive framework rather than being pre-specified.

   An alternative to learning a generative model for each pattern is to estimate a discriminative model or some other time series classification system. In general, discriminative approaches can be more flexible than their generative counterparts, and thus are likely to lead to more accurate decision boundaries for valid occurrences. However, the severe lack of training data and difficulty of applying such classifiers en masse limits their practical applicability for discovery.

3.2 Discovery Using Global Discretization

This approach to activity discovery seeks to combine the efficiency benefits of searching globally quantized time series with the modeling abilities of HMMs learned over the raw, real-valued data [57]. The algorithm follows the two-phased architecture described previously but also adds intermediate processing that refines the candidate motifs before the HMMs are learned (see Figure 2 for a diagrammatic overview). In the first phase, it generates a discrete representation of the data, builds a compact tree structure that allows efficient searches, and then identifies a set of seed motifs. The seed motifs are selected in a greedy fashion by first removing the occurrences of the best motif and then iterating until the next best motif fails to meet an information-theoretic criterion described in Section 3.2.1. The seed motifs are then refined using information from the continuous time series (Section 3.2.2), and, finally, a HMM is trained for each seed motif and used to detect the corresponding occurrences via a modified Viterbi alignment procedure (Section 3.2.3).

3.2.1 Identifying Seed Motifs

In the initial phase of the approach, the multivariate data is discretized such that each time series is transformed into a string. In all of the experiments presented here, the discretization is performed by fitting a mixture of Gaussian distributions to the data set comprised of all of the time series frames. The mixture model is learned using standard methods: first the k-means algorithm is used to find reasonable initial parameters, and then EM is used to refine the model parameters in order to maximize the data likelihood. The data used to learn the mixture model is simply the set of samples that make up the time series data. Thus, if the original data set consists of eight time series with an average of 1,000 frames per sequence, then the mixture model is fit to a data set of 8,000 points.

After estimating the mixture model parameters, each component is assigned a

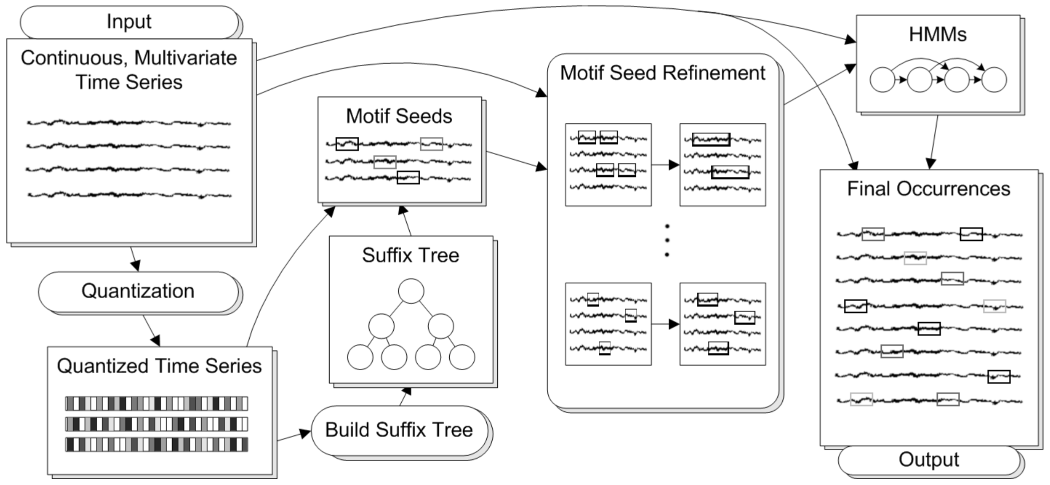

Figure 2: Overview of the global discretization discovery algorithm

unique symbol, and each frame of the time series is replaced with the symbol corresponding to the closest component. In the previous example, the result would be eight strings with an average of 1,000 symbols per string. Note that discretization methods other than fitting a Gaussian mixture model can also be used.

Once each time series is discretized, the resulting strings are used to build a generalized suffix tree. A suffix tree is a tree structure that holds all of the suffixes of a string in linear space [24] (see Figure 3). As an example, if the string `ABABC` is used as input, then the suffix tree will hold the strings `ABABC`, `BABC`, `ABC`, `BC`, and `C`. Importantly, every subsequence of the original string is the *prefix* of a *suffix*, and so every subsequence is stored in the suffix tree starting at the root node. For instance, in the `ABABC` example, the subsequence `BAB` is the prefix of the suffix `BABC`.

Ukkonen devised a conceptually simple algorithm to build a suffix tree for a single string in linear time [90]. His work was later extended to include linear time construction of *generalized* suffix trees which store multiple strings within a single tree [24]. Generalized suffix trees and the linear time construction algorithm provide a method to efficiently represent all of the quantized data and then to rapidly search it for common subsequences.

To identify potential motifs, each unique subsequence with a user-specified length

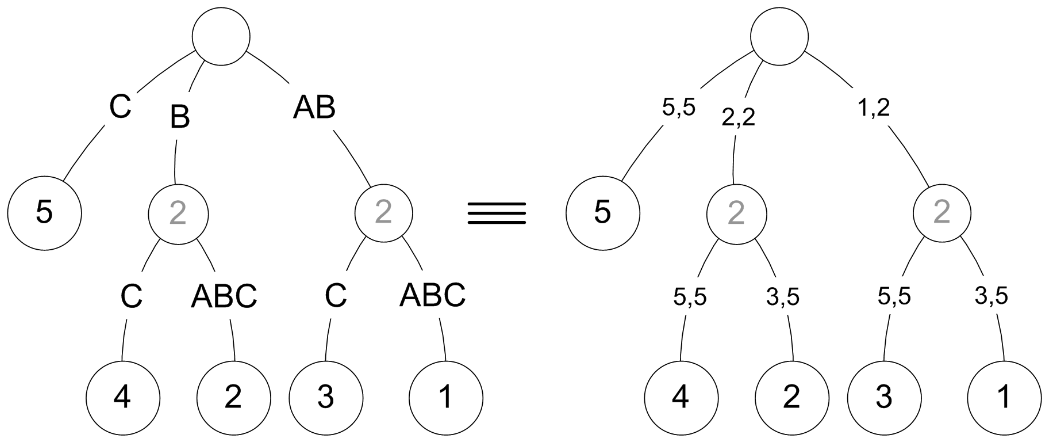

Figure 3: Suffix tree for the string `ABABC`. The left diagram shows explicitly labeled edges for clarity, while the diagram on the right shows how string indices are used to ensure linear space complexity. In the tree, every subsequence of the input strings is represented as a (possibly partial) path from root to node. The leaf nodes are annotated with the index in the original string of the corresponding suffix. Internal nodes hold a count of their children to allow efficient subsequence tallies.

is used as a query to find all other subsequences, of any length, that are equivalent after appropriate dynamic time warping. In the original implementation [57], this search was efficiently performed by a depth-limited traversal of the suffix tree. This search can be performed more efficiently, however, by pre-scaling each string to combine contiguous, repeated symbols. Then, once the suffix tree is built from the pre-scaled strings, the algorithm only needs to search for exact matches, which is much faster. In both cases, the result of the suffix tree search is a set of potential motifs along with a list of occurrences. Note that the algorithm can detect pattern occurrences with durations that differ from the precise value specified for the query length, but this value does restrict the temporal scale of the motifs that are likely to be found. This flexibility is due to the time-warping and temporal extension refinement steps (described in Section 3.2.2), but it is still highly unlikely that the algorithm will find a pattern with a drastically different duration (*e.g.,* a five minute long motif when the query length is only three seconds).

In order to rank the potential motifs and determine when to stop searching, each motif is scored according to an information-theoretic criterion. The criterion computes the change in description length of the original data sequence if the motif were encoded separately and every occurrence were replaced with a new symbol. This criterion serves to balance the number of occurrences of a particular motif with the complexity of each occurrence, which is necessary for two reasons. First, the motifs found should be *maximal*, which means that they should be as long as possible while maintaining intra-motif similarity. Second, in general, neither very long, complex, but rare motifs nor short, simple, and frequent motifs are particularly interesting. The criterion balances these two extremes. It arises directly from the work of Tanaka and Uehara [88] in this context, though using such minimum description length (MDL) criteria for knowledge discovery tasks has been used in many contexts before (*e.g.,* [27, 41, 1]).

The MDL criterion balances occurrence frequency with motif complexity and represents the number of bits needed to encode the motif plus the number of bits needed to mark each occurrence. The expression for the criterion is:

$$M \cdot log_2(q_m) + n \cdot log_2(q_s + 1)$$

where M is the total number of frames in all occurrences (*i.e.,* the total length of all occurrences), q_m is the number of unique symbols in the motif, n is the number of occurrences, and q_s is the total number of unique symbols in the quantized data. As an example, consider a string over the alphabet {`ABCDE`} and the motif `ABC` with the four occurrences `ABBC`, `AABC`, `ABBCC`, and `ABCCC`. In this case, $M = 18$, $q_m = 3$, $n = 4$, and $q_s = 5$. This leads to a description length of $18 \cdot log_2(3) + 4 \cdot log_2(5+1) = 38.8692$.

The motif with the largest score (implying the largest reduction in total description length) is selected as the most reliable and is added to the list of seed motifs. If this score is too small, however, seed motif discovery terminates. Otherwise, the process iterates in a greedy fashion to find the next best motif. The description length threshold used to decide when to terminate is a user-specified parameter.

3.2.2 Seed Motif Refinement

Once the full set of seed motifs are identified, they are refined to account for errors introduced by the quantization and by the user-specified query length. Four different kinds of refinement are performed: *splitting*, *merging*, *affix detection*, and *temporal extension*.

The split refinement step accounts for motifs that appear similar in the quantized data, but are clearly different when viewed in the continuous domain. For each motif, the occurrences are analyzed with agglomerative clustering using the farthest-neighbor rule. This linking rule compares the similarity of two sub-clusters and computes the distance between the farthest pair of members [20]. Then, the two sub-clusters that have the smallest farthest-neighbors are merged. This rule tends to generate more compact clusters compared to the nearest-neighbor or mean-distance rules. The result of agglomerative clustering is a binary tree of merges where each node represents a (sub)set of members. Leaf nodes represent individual members, while the root node corresponds to the entire set. The full merge tree can be visualized with a dendrogram (*e.g.,* see Figures 4 and 5).

For split refinement, the dendrogram is tested to see if it supports splitting the set into two new clusters. The test is performed by comparing the last merge (which takes the system from two down to one cluster, call it d_m) and the second to last merge (from three down to two clusters, d_{m-1}). If the corresponding distances are sufficiently large, then it means that the two clusters are very different, and so a split is performed (see Figures 4 and 5). This determination is made by comparing the proportion of the last two merges, d_{m-1}/d_m, to a user-specified ratio. If this proportion is larger than the threshold, then the split proceeds.

After all motifs have been tested for splitting, the resulting set then undergoes

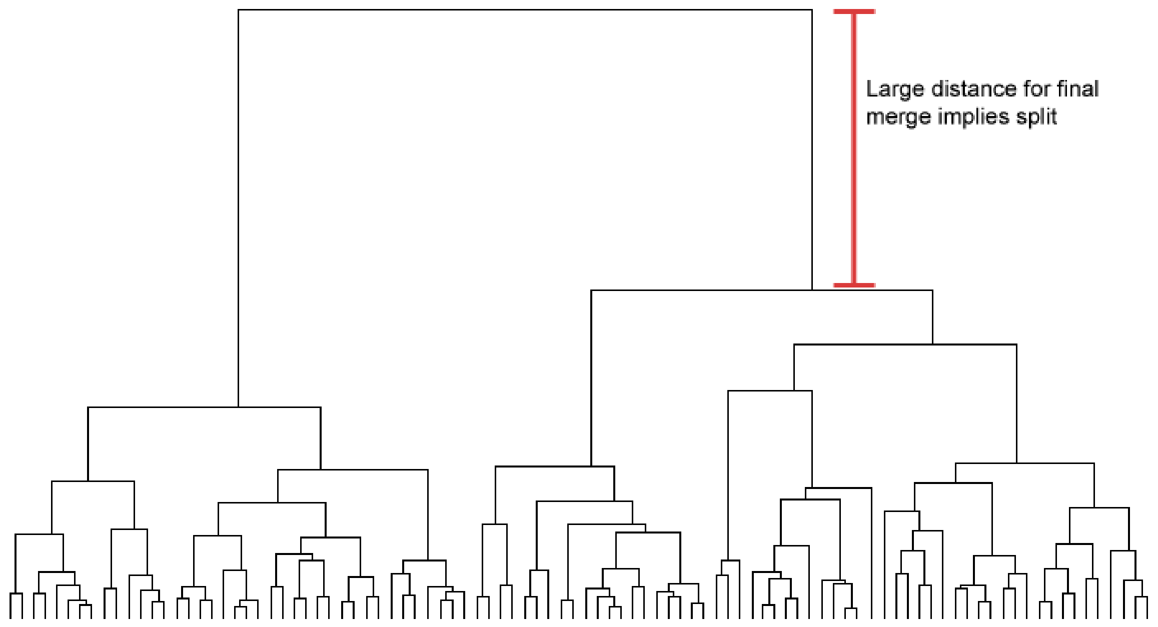

Figure 4: A dendrogram showing a motif that should be split (compare to Figure 5). A dendrogram represents the results of clustering by showing each item along the bottom and drawing a horizontal line for each merge (equivalently, for each split if the clustering is performed top-down). The distance between clusters increases as you move up the diagram, so merges near the bottom are typically more confident. Similarly, large vertical gaps between successive merges tend to indicate a natural partitioning.

merge refinement. This step addresses a common problem with quantized data that occurs when data exists very close to a quantization boundary. In this case, two values that are similar in the continuous domain may be assigned different symbols. As a simple, one dimensional example, consider the values 0.99 and 1.01 with a quantization boundary at 1.0. The two values will be assigned different symbols, even though they are quite close in the continuous space. Merging resolves such problems by combining clusters based on their similarity in the continuous domain. Like splitting, merging proceeds via agglomerative clustering, but this time the clustering is performed over the seed motifs rather than within their occurrences. As in the splitting case, two seed motifs are merged if the are very similar relative to a user-specified threshold.

Next, affix detection looks for pairs of seed motifs in which the occurrences of one

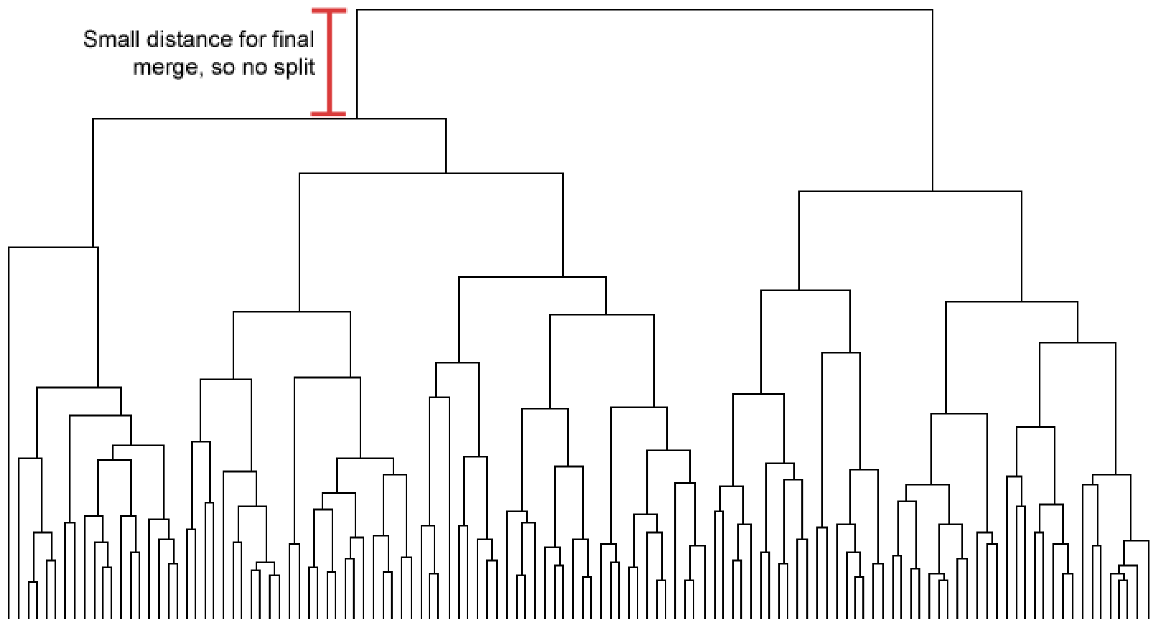

Figure 5: A dendrogram showing a motif that should **not** be split. Note how the final merge accounts for a relatively small amount of the total distance (compare to Figure 4).

motif always closely follow the occurrence of another. This can occur if the user-specified query length is short relative to the actual motif length. In such cases, a single motif can be identified multiple times, each time corresponding to a different portion of the full motif. For instance if the system is analyzing audio data and a duration of 250ms is used, the algorithm may detect the first syllable of “seven” (*i.e.,* “sev”) separately from the second (“en”). These multiple detections are considered incorrect since maximal motifs are desired. The affix detection refinement step seeks to locate these cases and then fixes them by extending one motif to include the other.

The final refinement step deals with temporal extension. This is the primary method used by the algorithm to adapt the motif length to fit the data when the user-specified query length is too small. For each seed motif, the parameters of a left-right HMM are estimated from the motif occurrences, and the set of variances in the observation distributions are extracted. The variance of the frame that precedes each occurrence is then compared to the mean of this set, and if it is comparable (or

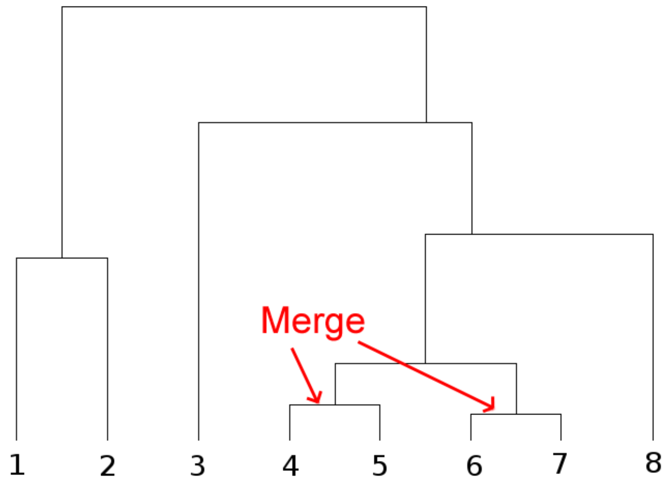

Figure 6: A dendrogram showing the results of motif clustering. Two motif pairs should be merged (Motifs 4 & 5 and 6 & 7).

smaller), the motif is extended to include the preceding frame. An identical procedure attempts to extend the motif forward in time, and in both cases, temporal extension continues until the variance of the next frame is too large or until the next frame is part of another motif.

3.2.3 Motif Modeling and Occurrence Detection

The final phase of the algorithm builds a probabilistic temporal model for each of the refined seed motifs, and then uses the models to detect all of the occurrences in the original time series. Left-right HMMs are used due to their history of good performance for speech and gesture recognition (*e.g.,* [87, 78]) and because a simple modification of the Viterbi alignment algorithm provides an efficient motif detection method.

The approach taken in this phase is to iteratively find the as yet undetected subsequence with the highest probability given the set of motif models. There are $O(T^2)$ subsequences, where T is the length of the longest time series, which means that the naïve approach of scoring each subsequence relative to each model is impractical even for medium-length data sets. We can adapt the Viterbi alignment algorithm,

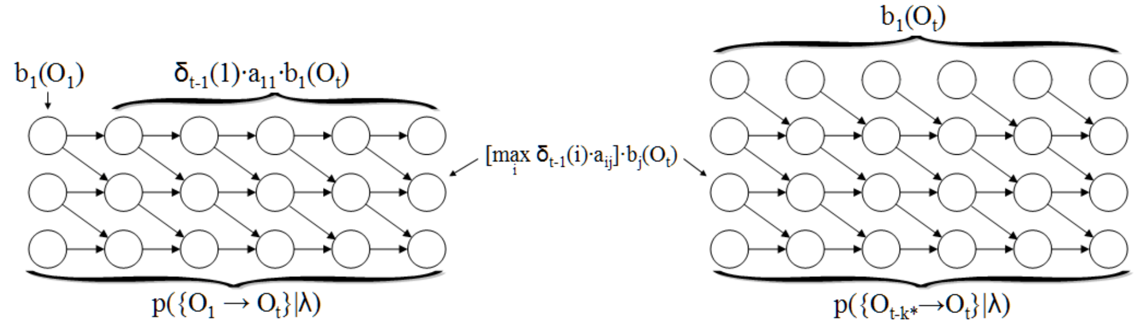
$b_1(O_1)$
$\delta_{t-1}(1) \cdot a_{11} \cdot b_1(O_t)$
$[\max_i \delta_{t-1}(i) \cdot a_{ij}] \cdot b_j(O_t)$
$b_1(O_t)$
$p(\{O_1 \rightarrow O_t\}|\lambda)$
$p(\{O_{t-k*} \rightarrow O_t\}|\lambda)$

Figure 7: By modifying the standard Viterbi alignment algorithm, we can find the best subsequence in linear time. k^* is the optimal subsequence length, which is found by tracing backward along "parent pointers" exactly as in the standard Viterbi algorithm.

however, to compute all of the needed probabilities with a single pass over the data, rather than with a single pass over each of the $O(T^2)$ subsequences.

The Viterbi alignment algorithm builds a trellis that stores the probability that each frame of data was generated in a particular state of the model assuming that the model explains all of the data starting with the first frame. In a trellis build for a left-right model, the top row gives the probability for the first state, while the bottom row gives the probability of the last state. Each column represents one frame of data. Typically, only the first column is initialized and then a dynamic programming algorithm is used to compute the probabilities in the rest of the trellis. In my approach, however, we fill in the first row as well as the first column, initializing each of the nodes along the top row as if the model started in the current frame (*i.e.,* $b_1(O_t)$ rather than $\delta_{t-1}(1) \cdot a_{11} \cdot b_1(O_t)$)[1]. The rest of the trellis is then computed using the standard method, but now the bottom row corresponds to the probability of being in the end state given that we started at *any* previous frame, rather than the first frame specifically. By scanning the bottom for a maximum, the end of the high probability subsequence can be detected in linear time.

[1] I adopt the notation of [79]: $b_i(O_t)$ is the probability of the t^{th} observation in the i^{th} state. $\delta_t(i)$ is the probability of being in the i^{th} state at time t given the observations up to that time. Finally, a_{ij} is the transition probability from state i to state j.

To find the first frame of the detected subsequence, one must simply trace backward from the last frame by following the "parent pointers" just as in the standard Viterbi algorithm [79]. This procedure is possible because the trellis is annotated by the dynamic programming algorithm that computes the internal probabilities of the trellis after the first column, as well as the top row in the modified version, has been initialized. For each trellis node corresponding to state j and observation frame t, the recursive equation $\delta_t(j) = [\max_i \delta_{t-1} a_{ij}] \cdot b_j(O_t)$ is computed. The value of i that maximizes $\delta_{t-1} a_{ij}$ for each node gives the index of the parent node which precedes the current node on the optimal path. By recording this value, it is possible to trace backward from the most likely end state to recover the optimal path.

This optimization works for two reasons. First, the use of a left-right topology ensures that any match starts with the first state (the top row of the trellis) and ends in the last state (the bottom row). Second, each model is constructed such that the transition probability from the first to the second state is equal to one ($a_{12} = 1.0$), while all other transitions from the first state are zero ($a_{1i} = 0.0,\ \forall i : i \neq 2$). The procedure of initializing each trellis node along the top row to $b_1(O_t)$ makes sense given this transition constraint, since it ensures that the probability of being in state 1 at time t is based solely on the observation distribution for that state, and not on any self-transitions from the first state to itself. Similarly, the first state can not be reached from any other state since the left-right topology requires that $a_{ij} = 0,\ \forall i : i < j$.

Given the above procedure, the motif occurrences can be enumerated in descending order of likelihood. At each iteration, the modified Viterbi algorithm is executed for each model, and the maximum end state probability is stored. The full occurrence location can be recovered given the end location and parent node annotations. Then, that segment is removed from the search region and the modified Viterbi algorithm is used to recompute the state probabilities for the regions that remain after the detected occurrence is removed (note that the values do not need to be recomputed for the other time series since those sequences will not be affected by the detected segment).

All that remains is to determine when the next best occurrence is not a valid occurrence. Two criteria are used to make this determination. The first is to simply check the length of the next best occurrence and compare it to the user-specified query length. If it is too small relative to a user-specified minimum, then it is assumed to be invalid and the search for valid occurrences is ended. The second criterion compares the likelihood of the next best occurrence to the distribution of likelihoods of the seed occurrences used for training. Specifically, each occurrence is given a score equal to $(1+e^{-(\ell-\mu)/\sigma})^{-1}$, where ℓ is the likelihood of the occurrence and μ and σ are the mean and standard deviation of the likelihoods of the seed occurrences. Thus, the scoring function is a sigmoid normalized by the standard deviation of the seed likelihoods, which allows a single stopping threshold to be applied to all of the motifs.

3.3 Discovery Using Local Discretization

Whereas global discretization methods convert each real-valued time series into a single string, local methods operate over relatively short windows. These windows may overlap and are individually normalized, which typically leads to the assignment of different symbols for the same frame when it is analyzed in the context of different windows. The benefit of local methods is that they can be more robust to noisy data, are typically more efficient to compute, and are more easily applied to streaming data since they do not require that all of the data be processed in batch. The remainder of this section describes an existing method that uses local discretization and then introduces my enhancement that improves modeling flexibility and often improves overall motif accuracy by automatically estimating a difficult to specify parameter that governs motif occurrence detection.

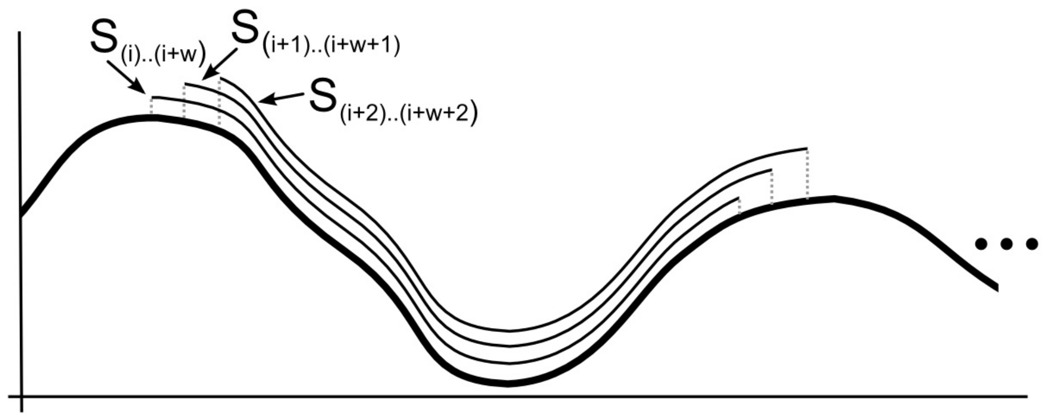

Figure 8: Three consecutive subsequences of length w from a 1D time series. The three subsequences are all very similar but are considered *trivial matches* since they overlap.

3.3.1 Local Discretization and Random Projection

This approach to activity discovery builds on the work of Chiu *et al.* [10]. Chiu's algorithm works as follows:

1. Extract all subsequences of a given length n from the time series using a sliding window (see Figure 8).
2. Convert each subsequence to a symbolic *word* using SAX with length w and with a unique symbols.
3. Build a *collision matrix* by comparing the strings via several iterations of random projection.
4. Select motif seeds as the two subsequences corresponding to the largest entry in the collision matrix.
5. Extract motifs by detecting other windows in the neighborhood of each seed. The neighborhood is defined as the space within a fixed distance R of either seed.

Subsequence Discretization: Each subsequence of the data is discretized using the SAX method developed by Lin *et al.* [50]. SAX is a local quantization method

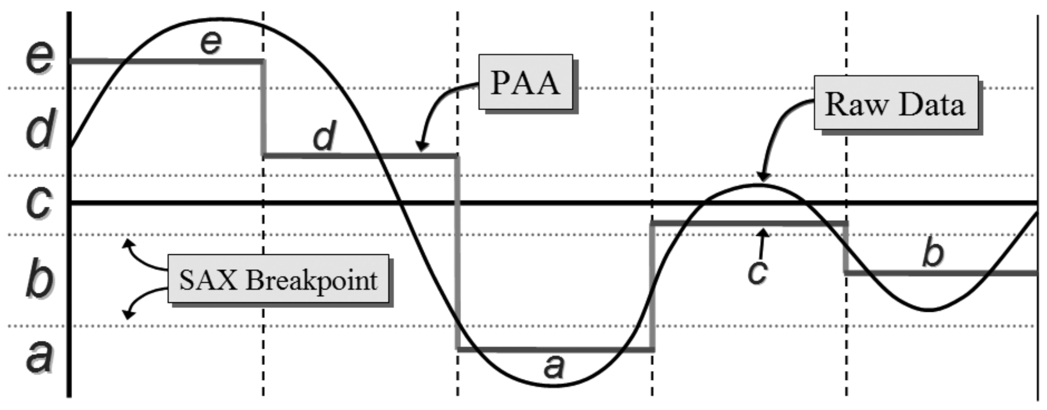

Figure 9: A real-valued, 1D signal is converted into a SAX string (`edacb`) using five PAA segments (bounded by the vertical dotted lines) and five SAX symbols.

that first computes a piecewise aggregate approximation (PAA) [35] of the normalized window data and then converts each PAA segment into a symbol. Normalization ensures that the subsequence has zero mean and unit variance. The PAA algorithm represents a sequence by dividing it into equal length temporal segments and then storing the mean within each segment. The final SAX quantization is based on pre-computed breakpoints that divide the data range into equiprobable regions assuming an underlying standard normal distribution. Each PAA segment is represented by the symbol of the corresponding bin (see Figure 9).

Building the Collision Matrix: Once each subsequence has been converted to a SAX string, the similarity between each pair is efficiently estimated using the random projection method introduced by Buhler and Tompa [8]. The key insight is that while direct comparison is infeasible since there are $O(T^2)$ pairs of fixed-length subsequences given T strings, similar sequences can be identified in $O(I{\cdot}T)$ time using I iterations of random projection (see Figure 11). Each iteration involves selecting a random subset of the string positions and building a hash table with the corresponding characters. After all strings are hashed, collisions (equivalent projections from different strings) are taken as evidence of similarity and the corresponding positions in a collision matrix are incremented.

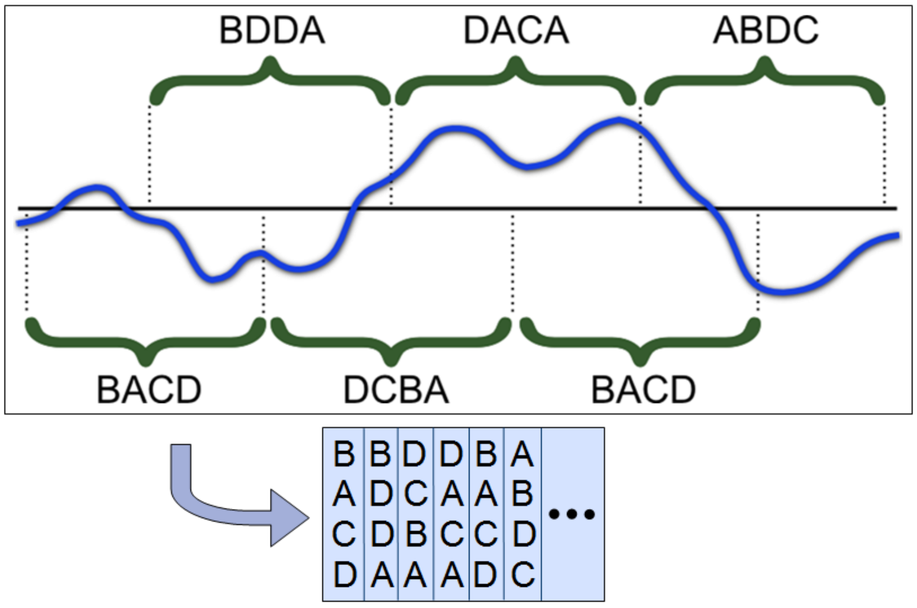

Figure 10: Depiction of the results of local discretization using four symbols when applied to a univariate signal. In general, the sliding window can have more or less overlap and can be applied to multivariate time series.

Using random projection to estimate similarity has several important properties. First, it is effective in the presence of shot noise. Even if some of the symbols are drastically different, when random projection does not select these positions, the string will still appear similar. Second, although the collision matrix could require $O(T^2)$ time and space in the worst case, realistic data sets typically produce a linear number of non-zero entries leading to linear time and space requirements when a sparse matrix is used. Furthermore, the complexity can be controlled online by dynamically adapting the parameters used for the discretization (the alphabet size and number of PAA bins) and for random projection (dimensionality of the subspace) [38].

Selecting Seed Motif Occurrences: Once the collision matrix is built, seed motif occurrences are extracted by locating the maximal entries in the matrix. Each entry corresponds to a pair of subsequences that can be used as seeds to find the other occurrences of the motif. Some restrictions are placed on the seed pair: (1) the distance between the seeds must be less than R, (2) they must not temporally overlap

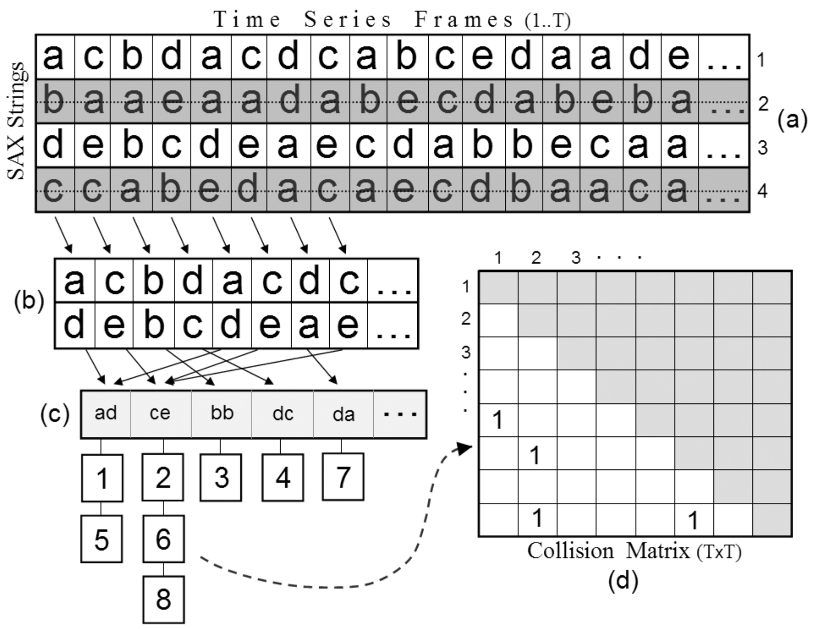
Time Series Frames (1..T)
SAX Strings
a c b d a c d c a b c e d a a d e ...
b a a e a a d a b e c d a b e b a ...
d e b c d e a e c d a b b e c a a ...
c c a b e d a c a e c d b a a c a ...
1
2
3
4
(a)
(b)
a c b d a c d c ...
d e b c d e a e ...
(c)
ad
ce
bb
dc
da
1
2
3
4
7
5
6
8
Collision Matrix (TxT)
(d)

Figure 11: (a) For each iteration of random projection, a subset of string positions are selected (here, positions one and three). (b,c) The remaining symbols are hashed, and (d) equivalent projections are tallied in a collision matrix.

previously identified occurrences, and (3) they must not overlap each other, which can lead to a *trivial match* and potentially to meaningless patterns [37] (see Figure 8).

Locating Additional Motif Occurrences: The final step of the discovery algorithm is to locate all other occurrences of the motif identified by the seed locations. This is achieved by scanning the original subsequences and selecting those within the neighborhood of the seeds (*i.e.*, $min(d(w_i, seed_1), d(w_i, seed_2)) \leq R$, where $w_i : i \in \{1..T\}$ represents each of the original subsequences).

Table 1: User-Specified Parameters for Chiu's Algorithm

Symbol	Description
n	window length
w	number of PAA segments (string length)
a	SAX alphabet size
R	radius of motif neighborhood

Discussion: Chiu's approach works very well despite the assumption that all motifs and all motif occurrences have the same length. It is, however, quite sensitive to the various user-specified parameters summarized in Table 1. In particular, the neighborhood radius, R, is vital for accurate motif discovery, yet it is very difficult to specify since it depends on the domain, the distance metric, the motif length, and the features selected for processing. However, if good parameter values can be specified, the algorithm is extremely fast due to the simplicity of each step. For example, during experimentation, this algorithm would often finish detecting all of the motifs and the corresponding occurrences before the global discretization approach had finished estimating the parameters of the Gaussian mixture model used for quantization.

Occurrence detection using the hypersphere model as is used by this algorithm is relatively straightforward because it is only a linear time operation once the models are specified. However, the problem of multiple detections must still be addressed. This problem arises because the subsequences that overlap a valid occurrence are typically quite similar and thus are likely to also be within the neighborhood of the corresponding motif. The issue is similar to that of trivial matches for unsupervised analysis but is much more well known due to the long history of research into supervised modeling and detection in temporal and spatial domains. A standard solution is to perform non-maxima suppression, which applies a post-process to the detected occurrences and removes any that are not considered to be the best matches within a local area. In the context of pattern discovery using a hypersphere model, non-maxima suppression requires the detection of overlapping occurrences and the removal of the one that is farther from the pattern's seeds.

This approach to non-maxima suppression can lead to issues when there are multiple, overlapping occurrences. The problem arises because the algorithm is greedy and local, whereas the desired objective function is a more complicated combination of occurrence quality as well as frequency. The MDL criterion used by the suffix tree-based algorithm presented previously, for instance, is an attempt to capture this more complex objective function, as is the use of density in the discretization-free approach described in the next section. Furthermore, the more complete analysis required to avoid the problems associated with a greedy selection process is intractable. In light of this problem, and due to some experimentation that demonstrated that the overall effect on accuracy was minimal, the implementation used to generate the results in Chapter 4 uses a simple linear scan to detect occurrences. Whenever a subsequence is encountered that is within the neighborhood of the current pattern, that subsequence is marked as an occurrence and the search continues after the end of the subsequence.

Chiu *et al.*'s motif discovery algorithm was developed in part as a response to his co-authors' research concerning trivial matches and the issues that they cause for subsequence clustering [37]. Their definition of a trivial match is any subsequence that is less than R units from the base subsequence and for which there is no intervening subsequence that is more than R units away. More formally, a subsequence w_j is a trivial match of a base subsequence w_i iff: $d(w_i, w_j) \leq R \; \& \; \nexists k : |i - k| < |j - k| \; \& \; d(w_i, w_k) > R$. The requirement adopted in this work, namely that motif occurrences can not temporally overlap, effectively addresses the issue of trivial matches but is somewhat more restrictive than the original definition.

3.3.2 Automatic Neighborhood Estimation

The primary restriction of the pattern discovery algorithm developed by Chiu *et al.* is the assumption of a fixed, user-specified neighborhood radius. The fact that the radius is user-specified adds to the knowledge burden on the user, but it is the assumption that all motifs have the same neighborhood size that is most restrictive (see Figure 12). In order to address this issue, I developed an algorithm for automatically estimating the neighborhood radius for each pattern independently by analyzing the potential

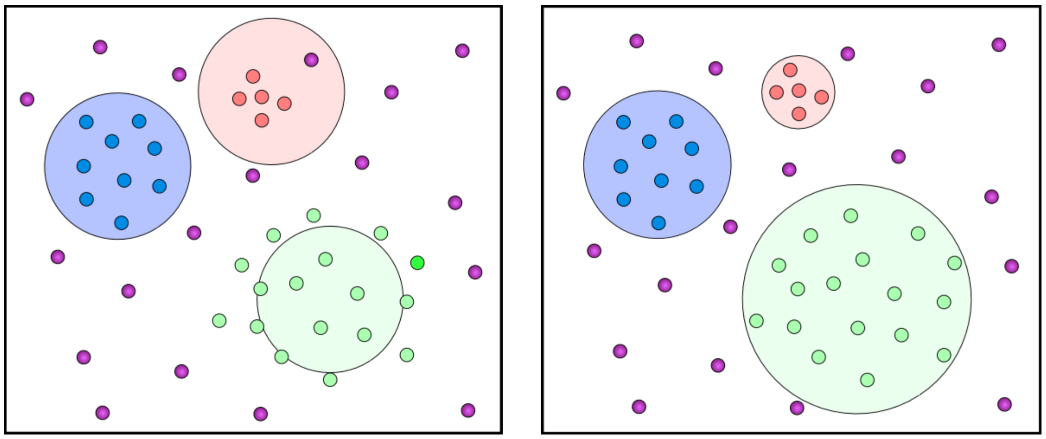

Figure 12: A single neighborhood radius may not be able to capture the extent of different patterns (*left*). When the radius is allowed to vary, however, each pattern can adapt to more accurately separate valid occurrences from other subsequence (*right*).

occurrences given the seed instances of each motif.

Because the radius is not used during the local discretization and random projection phase, that part of the algorithm proceeds as usual. Once a pair of motif seeds has been selected, the distance from each subsequence to the closer of the two seeds is computed using the original, real-valued data. Next, the smallest portion of the distances are selected for further analysis (note that such order statistics can be found in linear time using partitioning methods [16]). In the experiments presented in Chapter 4, the smallest 10% are used based on the assumption that 10% is a very loose upper bound on the number of motif occurrences. The empirical results show that the precise size of the portion only has a minor impact on the final estimate (see Figure 32).

Finally, to determine the appropriate neighborhood radius, the algorithm sorts the distances and seeks a "knee in the curve." The knee corresponds to the change from distances within the motif to those outside, which corresponds to an inflection point of the curve. Inflection points can only occur at a zero crossing of the second derivative, which is also an optimum of the first derivative. Thus, estimating the radius is equivalent to finding the maximum of the derivative (see Figure 13), and

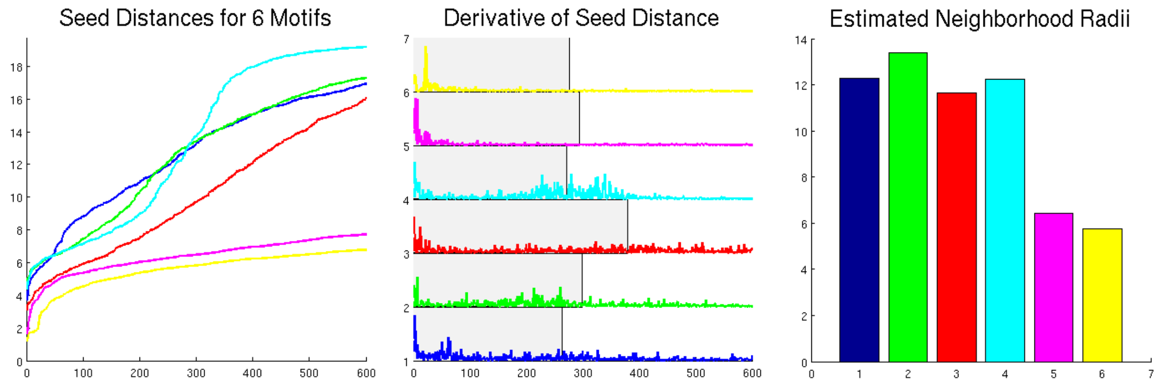

Figure 13: The radius of each motif neighborhood is estimated from the distance between each potential member and the motif seeds (*left*). It is equated with the inflection point, which is estimated by the weighted mean of the derivative of the distances (*center*). The result is automatic motif-specific neighborhood sizes (*right*).

because we can directly compute the discrete first derivative, no assumptions about the parametric form of the distance curve are necessary.

The inherent undersampling of the true distribution of motif occurrences causes the observed distances to be quite noisy. Therefore, the maximum is estimated as the weighted mean of the derivative. This procedure is summarized below:

1. Calculate the minimum distance from each subsequence, $w_i : i \in \{1..T\}$, to a motif seed:

   $d_i = min(d(w_i, seed_1), d(w_i, seed_2))$

2. Compute, v_k, the k^{th} order statistic for $k = \frac{T}{10}$

3. Select the k smallest distances: $d_i : d_i \leq v_k$

4. Sort the distances in ascending order

5. Calculate the discrete first derivative: $d'_i = d_{i+1} - d_i$

6. Treat the derivative as weighted votes for the best radius and compute the expected value:

   $E(i) = (\sum_i d'_i \cdot (i + \frac{1}{2}))/(\sum_i d'_i)$

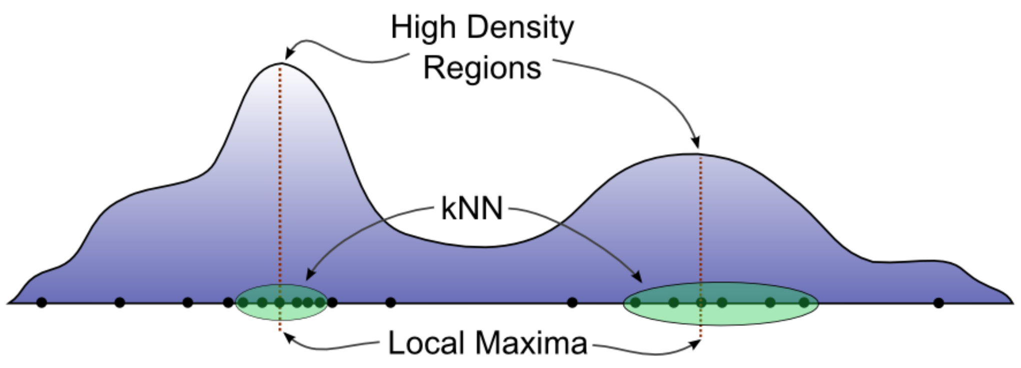

Figure 14: The density-based algorithm tries to locate high density regions by finding those data points with the nearest k^{th}-nearest neighbor.

The inclusion of this algorithm to automatically estimate the neighborhood radius for each motif is the primary difference relative to the original method of Chiu *et al.*

3.4 Density-based Discovery without Discretization

The third approach to motif discovery that I have designed and evaluated avoids discretizing the time series data yet maintains an expected run time that is sub-quadratic in the number of data points [58]. This method frames the motif discovery problem as one of locating high-density regions in the space of all subsequences. Since motifs are sets of subsequences with high intra-motif similarity, the specific meaning of "high intra-motif similarity" can be contextually defined by equating it with regions of high density relative to a particular data set. Density captures two key aspects of a motif: high similarity (*i.e.,* multiple subsequences within a small volume of space) and high frequency (*i.e.,* many occurrences within that region).

3.4.1 Efficiently Locating Density Modes

One of the most efficient and robust methods for locating density modes (local maxima of the density surface) is the mean shift procedure, especially when the approach is accelerated using dual-tree methods [14, 93]. This non-parametric algorithm iteratively locates modes by computing a vector aligned with the local gradient and then updating the mode estimate. This update is performed efficiently by noting that the mean of local data points, weighted by an appropriate kernel, points in the direction of the local gradient. The mean shift procedure was used by Denton to find density modes in fixed-length, univariate time series subsequences [18]. One drawback of this method, however, is that the bandwidth of the weight kernel must be estimated from the data to find a single satisfactory bandwidth or to estimate a data-adaptive bandwidth that varies over the data points [81, 15]. In either case, this is a computationally intensive task.

In our discovery algorithm, we use a computationally simpler method that approximates the local density by using the distance to the k^{th}-nearest neighbor [54, 85]. For a given value of k, a smaller distance implies a higher density since the same number of points lie within a smaller volume of space (see Figure 14). After estimating the density at each point, we keep only the local maxima, which we define as those subsequences that have higher density than all of their k-nearest neighbors (see Figure 15). Although a naïve implementation of all-points k-nearest neighbor search requires $O(n^2)$ distance calculations, numerous methods have been proposed for building spatial indexes that reduce this to $O(n \log n)$ and dual-tree and approximate methods typically provide even better performance [39, 53, 23].

3.4.2 Greedy Mixture Learning for Motif Selection

While the density-based procedure will identify candidate motifs seeds, it will neither detect all of the occurrences of each motif nor will it reliably select an accurate set of motifs. Incomplete motifs arise because there may be motif occurrences beyond the local maxima and its k-nearest neighbors. Our approach addresses these problems by learning a hidden Markov model from each candidate motif (*i.e.,* a local density maxima and its k-nearest neighbors) and letting the motif models compete to explain the time series data.

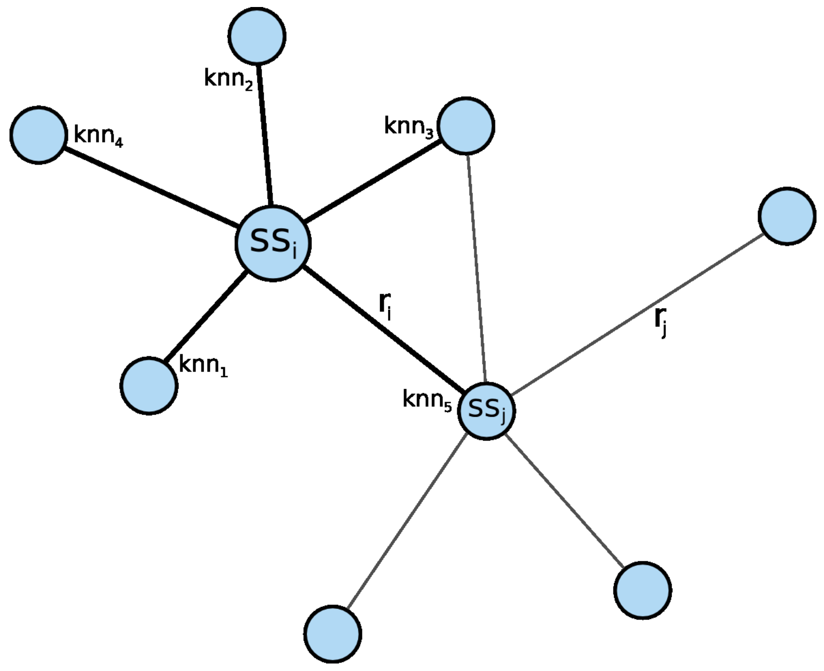

Figure 15: After computing the k-nearest neighbors for each subsequence and estimating the surrounding density, only those subsequence that are local maxima are kept as candidate motif seeds.

Although traditional parameter estimation methods for HMMs, such as the Baum-Welch algorithm, typically fail when applied to so few training examples, a simple construction algorithm is sufficient to capture the characteristics of each motif. Deficiencies in the resulting model are countered by the competitive continuous recognition framework and by full parameter re-estimation using all identified motif occurrences after all motifs have been identified.

The HMM construction algorithm builds a model with a left-right state topology. The number of states is set to half of the number of frames in each subsequence (this value is specified by the user as the subsequence length), and each training sequence is divided into equal-length, overlapping segments which are used to estimate the parameters for the Gaussian observation distribution for each state (see Figure 18). This procedure avoids segment boundary aliasing through redundant, overlapping states, and allows the model to adapt to motif occurrences with different lengths due to the state self-transitions and "skip" transitions.

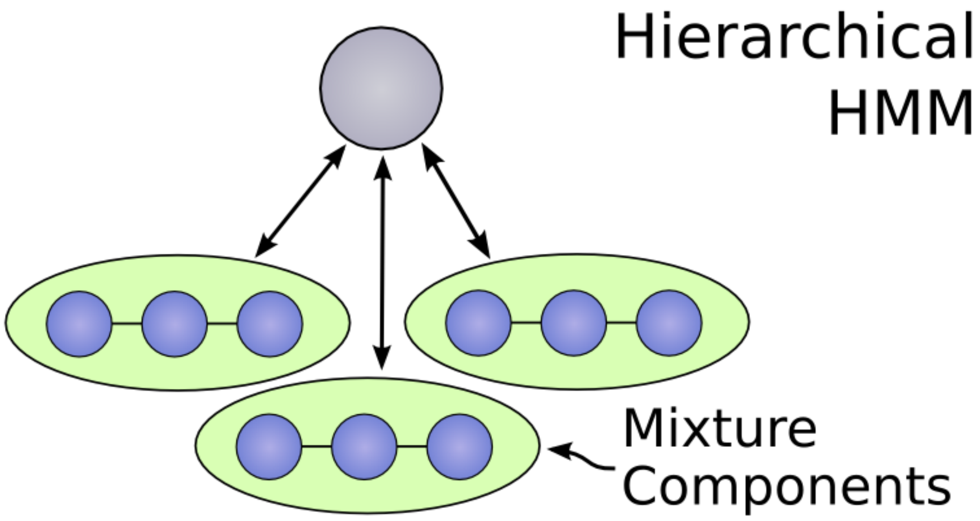

Figure 16: A hierarchical HMM can be used to learn the parameters of the motif models [68] but leads to a difficult optimization problem with many local minima.

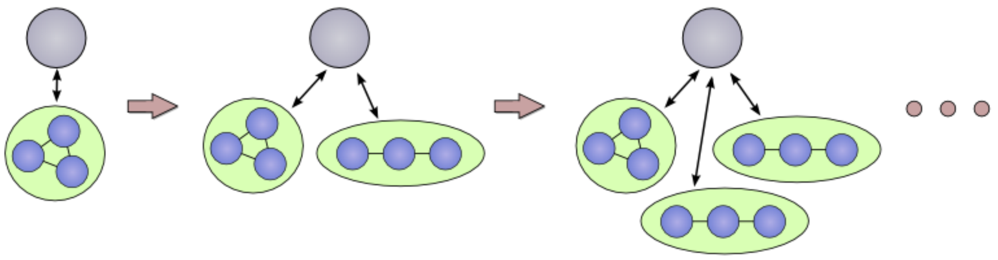

Figure 17: In a greedy mixture learning framework, models are added incrementally. While this approach leads to multiple learning problems, each problem is much simpler than in the full hierarchical model case. [5, 58]

In traditional applications, mixture components jointly explain the data by sharing responsibility for each data point (*i.e.*, $p(x) = \sum_{i=1}^{N} w_i \, p(x|\theta_i)$, where θ_i represents the parameters of the i^{th} component and w_i is the weight given to that component constrained by $\sum_{i=1}^{N} w_i = 1$). The learning problem is to estimate the parameters, θ_i, and the weights, w_i that maximize the total data likelihood. In the general case, the learning problem also includes estimation of the number of components in the mixture, typically using a Bayesian framework or some other model complexity penalty to prevent the trivial maximum-likelihood solution of having one mixture component

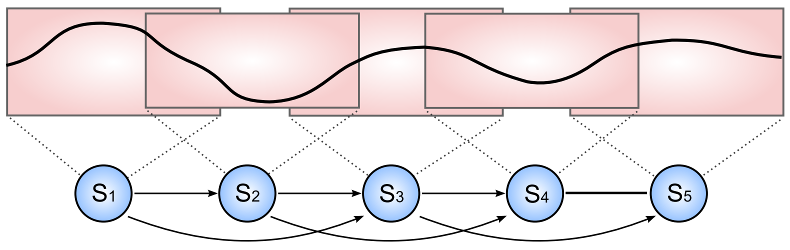

Figure 18: Illustration of HMM construction. Each sequence in the training set is divided into equal-sized, overlapping segments from which the Gaussian observation distribution of each state is estimated. Self-transitions for each state (not shown) allow the model to adapt to long occurrences, while the skip states allow it to map to shorter occurrences.

per data point.

In the context of pattern discovery, as posed here, motif occurrences do not overlap temporally, and so only one mixture component is used to explain each frame of data. The set of components must then jointly explain the full data set by accounting for different temporal data segments. The pattern recognition community has developed efficient algorithms for solving such problems, principally for the purpose of continuous speech recognition [99]. In typical speech systems, each word is modeled by a HMM (commonly this is a composite model built from HMMs representing the relevant phonemes) and then each utterance is recognized by finding the mapping from word HMMs to the speech signal that maximizes the log-likelihood of the utterance given the set of models [29, 78]. In our algorithm, HMMs constructed from motif seeds (*i.e.,* a local maximum in density space along with its k-nearest neighbors) take the place of the word models.

In this approach, the motif candidates compete within a greedy mixture learning framework [5, 91]. The mixture is comprised of different motif models that jointly explain the entire time series data set, and which iteratively grows as additional motifs are discovered. The mixture is initialized with a single HMM that represents the "background" model by capturing global statistics of the data. The background model used in all of the experiments presented here is a one state HMM with a three component Gaussian mixture model as its observation distribution. The parameters of the Gaussian mixture model are estimated from all of the frames from the time series data and then the variance is artificially inflated by a factor of three to avoid interfering with the motif models. This model is similar to other robust maximum-likelihood approaches in that the high variance background model assigns a small but non-negligible probability to all data points, thus allowing the real model to ignore outliers [12, 62].

Candidate motifs are greedily selected by measuring the information gain corresponding to adding each candidate motif model to the mixture. The information gain is calculated as the difference between the conditional log-likelihood of the time series data when the candidate model is included and when it is left out of the mixture. After each iteration, the motif that provides the largest information gain is selected and permanently added to the mixture.

The benefits of using a HMM-based continuous recognition system for motif occurrence detection are numerous. First we do not need to directly estimate the motif neighborhood size as was required by previous methods (*e.g.,* [10, 59, 89]). Second, the motif models are free to shrink or stretch to account for non-linear time warping and variable length motif occurrences in the data, whereas the existing approaches are restricted to fixed-length subsequences. Third, the fitting procedure locates all of the motif occurrences given the models and also simplifies the process of pruning redundant or spurious motif seeds. The pruning is possible because redundant motifs will lead to low information gain due to a previous motif already providing a good model for the relevant data segments, while spurious motifs simply will not map to many segments. Finally, because all of the frames of the time series are explained by either a motif model or the background model, there is no bias toward short occurrences. This bias often arises in systems which identify motif occurrences using maximum likelihood and do not model the non-motif data frames as was the case in

Algorithm 1 Density-based Motif Discovery

```
Input: Time series data ($S$), subsequence length ($w$), number of nearest neighbors to use ($k$), distance measure

Output: Set of discovered motifs including motif models and occurrence locations

1. Collect all subsequences, $S_i$ of length $w$ from the input data $S$
2. Locate the $k$-nearest neighbors for each subsequence: $knn(S_i) = S_{i,1..k}$
3. Estimate the density for each subsequence: $den(S_i) \propto 1/dist(S_i, S_{i,k})$
4. Identify local maxima according to density: $maxima(S_i) = S_i : \forall S_{i,j}\ den(S_i) > den(S_{i,j})$
5. Initialize the set of motif HMMs with a single background model: $H = \{bg\}$
6. For each motif seed in $maxima(S_i)$:
    (a) Construct $seed_i$, a HMM learned from the $i^{th}$ density maxima and its $k$-nearest neighbors
    (b) Fit the existing models plus the seed model ($H \cup seed_i$) to the time series data
7. Greedily select the best motif seed:
   $m = \underset{i}{arg\,max}\ log\ p(S|H \cup seed_i)$
8. Test for stopping criteria for $H \cup seed_m$; if test fails set $H = H \cup seed_m$ and goto Step 6
9. Re-estimate motif models in $H$ and return $H/\{bg\}$
```

the global discretization algorithm presented in Section 3.2.

Algorithm 1 gives an overview of the density-based approach to motif discovery. Steps 1 through 4 locate potential pattern occurrences that together form an over-complete set of candidate motifs corresponding to subsequences located near high density regions. Selection of motifs via temporal greedy mixture learning is performed in steps 5 through 8.

The final component of the greedy mixture learning approach is responsible for detecting when to stop adding motifs to the mixture model (Step 8). The stopping criteria is often one of the most difficult aspects of an unsupervised algorithm to design. Many methods use criteria that rely on user-specified thresholds or assumptions about the data generation process. For instance, the methods of Chiu *et al.* [10], Minnen *et al.* [59], and Tanaka and Uehara [89] search for additional motifs until no pair of subsequences lie within the same neighborhood. Denton's method [18], on the other hand, continues searching until no subsequence lies in a region with density above a threshold estimated from the assumed random-walk noise model.

In contrast, the approach presented here uses the density estimate of each motif, computed after all occurrences have been detected, to determine when to stop. The algorithm is similar to the process used to estimate the neighborhood radius in the algorithm presented in the previous section. It searches for a local minimum in the smoothed derivative of the motif densities, which corresponds to an inflection point in the curve representing the motif densities. This inflection point marks the transition from well-supported motifs (high density) to spurious motifs (low density). Despite its heuristic origin, the empirical results validate the usefulness of this metric and show that the approach provides stable results even if the number of motifs is not estimated perfectly.

Once the algorithm estimates the total number of motifs, the model parameters for all valid motifs are iteratively re-estimated (step 9 of algorithm 1). The re-estimation process uses the familiar Baum-Welch algorithm and uses all of the segments matched by the model for training data.

3.5 RP-GML: Random Projection with Greedy Mixture Learning

As the results presented in Chapter 4 demonstrate, the density-based algorithm discovers much more accurate patterns than the global or local discretization approaches. The filter stage of that algorithm, however, runs more slowly than that of the local discretization even though both have sub-quadratic asymptotic complexity. In order to investigate whether the improved performance is due to the filtering algorithm (*i.e.,* local discretization vs. density estimation) or due to the pattern model (*i.e.,* hypersphere model vs. greedy mixture learning with HMMs), I developed a hybrid method that combines the local discretization filter with the HMM motif model.

The filtering phase in the hybrid algorithm is the same as in the approach presented in Section 3.3, except that the final motif selection during each iteration is determined by a maximum-likelihood score. In other words, instead of simply choosing the largest entry in the collision matrix and using seed pair distance to break ties, the top N entries (ranked by collision value and then by seed pair distance) are passed on to the second phase. For each seed, an HMM is initialized and its effect on the overall data likelihood is computed as in the density-based approach. The model that leads to the largest increase in data likelihood is selected as the most reliable pattern and is permanently added to the mixture model.

The stopping criterion used in this hybrid algorithm is similar to that for the local discretization algorithm despite the change in motif model. The discovery process continues to search for additional patterns until either there are no more entries in the collision matrix with values higher than what is expected for random data or until there are no more unexplained frames in the time series data.

CHAPTER IV

EMPIRICAL RESULTS

In order to evaluate the performance of the motif discovery algorithms presented in the previous chapter, as well as to compare them to existing methods, each algorithm was executed on a variety of data sets drawn from different domains and using different sensing modalities. Because this research is not focused on discovery within a single domain (*e.g.,* biological sequences or human speech), the goal was to evaluate the performance across many domains to get a better picture of each algorithm's performance. This broader evaluation should help predict how the algorithms will generalize to new data sets.

Currently, there are no standard data sets used within the data mining or machine learning communities to evaluate motif discovery algorithms. In fact, there is little agreement on the precise definition of the problem, and the wide variation in assumptions used by different researchers to help them focus on specific aspects of the overall discovery problem make comparisons very difficult. In the existing literature, new algorithms are typically evaluated by presenting example motifs discovered in a small number of data sets. Occasionally, a previous algorithm is shown to fail to detect a recurring pattern that would likely be identified by a human analyst, or algorithms are compared by their run time characteristics with less attention paid to the specific patterns that are detected.

4.1 Evaluation Methods

In this research, algorithms are evaluated in terms of both the accuracy of the detected patterns and relative to their run time characteristics. Measuring the accuracy of the detected patterns is difficult because the discovery process is fundamentally unsupervised and so in general there is no labeled data that can be used to test the learned models. Broadly, there are three approaches that can be adopted to evaluate a discovery algorithm: expert analysis, evaluation on a primary task, and correspondence with expected/known patterns.

4.1.1 Expert Analysis of Discovered Patterns

This approach to evaluation relies on an analysis of the discovered patterns by an expert who can reliably assess the quality of each pattern. The exact nature of the evaluation can vary from objective measurements of internal pattern agreement to a qualitative assessment based on the expert's understanding of the domain. For instance, when analyzing speech data, a fluent speaker can listen to the utterances collected within a particular pattern and determine which word or phrase, if any, dominates the set. The motif can then be scored in terms of its precision, which corresponds to the homogeneity of the utterances that were grouped together. While a more homogeneous motif is preferable, this kind of analysis does not account for false negatives (*i.e.,* a measure of the recall rate or number of occurrences that should have been included but were incorrectly left out) and thus gives an incomplete indication of performance.

In other domains, it may not be as straightforward to provide a well-established label for each motif occurrence. Instead, a domain expert may need to interpret each pattern and provide either subjective labels for each occurrence or estimate the overall motif quality based. This assessment may be based on how well the motif can be explained using the established models used in the domain, or it could be based on the expert's personal experiences working with other, related data sets. This kind of evaluation may even include additional investigation of the underlying phenomena through other means such as direct statistical analysis or even further laboratory experimentation.

4.1.2 Effect on Primary Task

In many situations, the motifs detected by the unsupervised pattern discovery algorithm are intended to be used as the basis for further analysis. The motifs thus serve as primitives or temporal features that are the input to a system that reasons, plans, or classifies at a higher level of abstraction. In these cases, the accuracy of the motifs relative to some external set of clusters, patterns, or perceptual concepts may not matter per se, but rather the patterns are important only insofar as they support the goals of the higher-level reasoning system, which can be considered the primary task to which the motif discovery is subordinate.

To evaluate the discovery (sub)system when used in a hierarchical framework, one should measure the effect on the performance of the primary task rather than impose a metric on the motifs directly. One potential downside of this approach is that the evaluation becomes dependent on both the data set, which is always the case, and also the goal of the primary task. So whereas before one might conclude that a particular algorithm performs well for speech analysis but not for discovery in GPS traces, the indirect evaluation proposed here may only support the more restricted conclusion that the algorithm is well-suited for pattern discovery that supports continuous speech recognition but not word spotting.

4.1.3 Correspondence to Expected Patterns

A third approach for evaluating motif discovery systems, and the one that is used in this research, is to compare the agreement between the discovered patterns and a set of expected patterns. The underlying assumption of this evaluation method is that the user or some other domain expert is sufficiently confident about a set of patterns in the data set that it is considered an error if the discovery system fails to detect those patterns. Thus the system is scored based on how well the discovered patterns match the expected patterns, which is a similar procedure to supervised evaluation. There are, however, two key differences. First, in the supervised case, the automated system knows the labels associated with each class while the unsupervised system does not, and second, the discovery system may detect additional patterns which were not expected but which are nonetheless valid patterns.

To understand the first issue, consider an audio data set consisting of English speech in which the user can hear repetitions of the words "one," "two," and "three." When the discovery system analyzes the data, however, it will output sets of occurrences labeled "Motif 1," "Motif 2," *etc.*. The evaluation subsystem must figure out which, if any, of the motifs correspond to each of the expected utterances. Given a measure of similarity between each motif and each word, this correspondence calculation can be efficiently solved with the the Hungarian algorithm, also called the Kuhn-Munkres algorithm, which solves the assignment problem in $O(n^3)$ time [42, 67].

All that is left is to specify the similarity measure between a motif and an expected pattern. This measure has two components, one that measures the similarity between a particular motif occurrence and a single pattern instance, while the second uses that information to measure the similarity between the two sets. Many options are available for each component. In this research, a relatively simple approach is adopted, which nonetheless captures a notion of similarity that makes sense given the common application of a discovery system as a tool to help researchers locate and focus on patterns in their data. Specifically, a discovered occurrence and expected instance are considered to match if there is any temporal overlap between them. The similarity between a motif (a set of occurrences) and an expected pattern (a set of instances) is then computed as the maximum number of matches provided that no motif occurrence can match multiple instances or vice-versa.

Detecting overlap between two segments is straightforward. Given that each segment is parameterized by the tuple ($s = \langle\text{series index}\rangle, a = \langle\text{first frame}\rangle, b = \langle\text{last}$

frame⟩), two segments, seg_i and seg_j overlap when:

$$(s_i = s_j) \wedge (a_i \leq b_j) \wedge (a_j \leq b_i)$$

The best possible mapping, that is the assignment of motif occurrences to pattern instances that leads to the maximum number of valid matches, can then be computed using a simple search. Note that a more nuanced similarity metric would take in to account the amount of overlap, while a strict metric would require a minimum level of overlap for a pair to be considered a match. The strictest metric would require an exact match, that is $(s_i = s_j) \wedge (a_i = a_j) \wedge (b_i = b_j)$, however such precise discovery is essentially impossible and is not required for a system to be broadly useful [65, 66].

The second major issue for correspondence-based evaluation is how to handle extra discovered patterns that do not match any of the expected patterns. A strict evaluation would consider such detection to be false positives, which negatively effect the overall accuracy score. However, when manually investigating some of the additional patterns detected in real data sets, it was often determined that they referred to valid patterns that simply weren't predicted by the domain expert. For example, the first unexpected pattern detected in many cases represented a version of silence appropriate for the domain. As such, in the evaluation results presented in this chapter, additional patterns are ignored and thus do not improve nor reduce the performance scores.

4.1.4 Evaluation Metrics

A variety of methods have been developed in the pattern recognition and information retrieval communities for quantifying and presenting performance statistics. Graphical summaries such as receiver operator curves (ROCs) depict performance over a range of parameter values [20], while segment error tables (SETs) [94] and error division diagrams (EDDs) [60] are tailored to temporal event recognition tasks. Other approaches try to summarize the results with a single number, which allows easy comparison between different methods at the expense of providing a more complete picture of the performance.

In this research, accuracy and F-measure (specifically, the F_1-measure) are used to provide a quantitative summary of discovery performance. Accuracy measures how well the detected occurrences from a particular system match the expected, labeled instances. The evaluation of a temporal recognition or discovery system is somewhat more complicated than that of a traditional instance-based classification algorithm and combines several basic statistics:

- N – total number of true occurrences
- C – number of correct detections
- I – number of insertion errors (detections that do not match a true occurrence)
- D – number of deletion errors (true occurrences that were not detected)
- S – number of substitution errors (true occurrences of one class that were incorrectly detected as some other class)
- E – number of "extra" detections (detections that are occurrences of a discovered pattern that does not correspond to any of the expected patterns)

The primary departure from traditional accuracy calculations is the inclusion of the E statistic. In supervised classification, every test instance or detection is labeled with one from a set of predetermined labels. In the context of discovery, however, the system can detect additional patterns beyond those anticipated. All such detections increase the value of E. Note that E is different from I in that E detections are from an extra class (a pattern that was not expected), while I detections are from an expected class but do not correspond to a true occurrence. Finally, in the formulation used here, the extra detections are ignored and so E does not affect the final accuracy, although it could reduce the accuracy if a different, stricter formulation were adopted.

Accuracy is calculated from the above statistics from this simple formula:

$$acc = \frac{C - I}{N}$$

Note that this formula does actually account for all of the different error types (except for E as noted above) since $N = C + D + S$. Because the accuracy score penalizes for insertion, deletion, and substitution errors, and because the number of insertion errors is bounded by the data not by the number of true occurrences, the accuracy measure can be negative whenever $C > I$.

The F-measure, on the other hand, is a bounded score of performance that ranges between zero (the worst possible performance) and one (the best possible performance) in all cases. The F-measure combines the notion of recall and precision, evaluation metrics typically used to summarize the performance of information retrieval algorithms. Given a discovered motif that corresponds to an expected pattern, recall measures the fraction of the true occurrences that were detected, while precision measures the fraction of the detected occurrences that are correct. Both of these measures vary between zero and one, but note that either can be maximized in a fairly trivial and useless way. For example, a recall rate of 100% can be easily reached, at least for a single pattern by "detecting" every possible segment. In that case, all of the correct occurrences will be matched, but so will a huge number of incorrect segments. Similarly, precision can be maximized by finding a single correct occurrence.

The F-measure attempts to mitigate the deficiencies of the individual recall and precision measures by combining them. Specifically, the F-measure is computed as:

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

while the more general formula

$$F_\beta = \frac{(1 + \beta^2) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall}$$

(with $\beta \geq 0$) can be used if either recall or precision should be weighted more heavily in the combined score. Note that when $\beta = 1$, $F_\beta = F_1 = F$, and the precision and recall are weighted equally.

A complementary issue to the specific evaluation metric used to summarize results is which aspect of the results is actually analyzed. In the discussion so far, the analysis has been over the correspondence between the discovered motifs and the expected patterns at the level of the occurrence. Each occurrence is a temporally contiguous segment that has a particular start time, duration, and label. Performance is then measured as a function of the correspondence between the discovered and expected intervals, which is often called *event-level* analysis.

An alternative to this approach is to calculate the performance metric in terms of the individual samples, which is often called *frame-level* analysis. In this case, each frame is given a ground truth label based on the expectations of the domain expert, and also by the discovery system depending on which motif occurs at that particular time. Note that there is typically an implicit label for those frames that do not occur during a motif. The primary benefit of performing a frame-level analysis is that it avoids the complications involved in matching events. Because each frame is inspected independently, there is no ambiguity involved in determining whether the label provided by the discovery system matches the expected label. Note, however, that using a frame-level analysis does not preclude the need to determine which motif best corresponds to which ground truth label.

The main problem with using a frame-level analysis to summarize the performance of an algorithm is that it can give misleading results for real-world applications. A typical scenario calls for discovery or detection of a particular pattern so that the user can further investigate the underlying process at the relevant times to figure out what causes the pattern. In such cases, identifying the precise start and end time of each

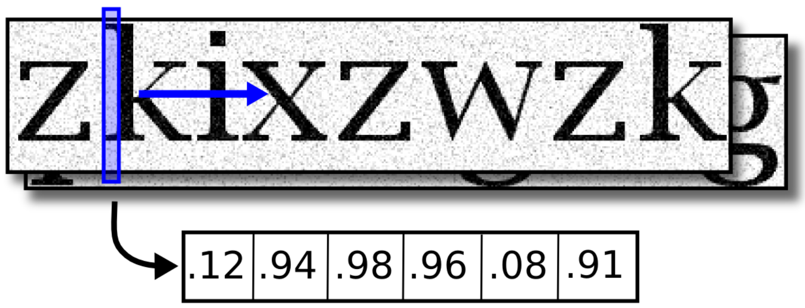

Figure 19: Synthetic data set that illustrates variable-length motifs. Each image represents a random string drawn from the alphabet {`zgwxfpmik`} and is corrupted by white noise. The "time series" frames correspond to features extracted from a vertical bar that slides across the image and measures the intensity of the underlying pixels.

occurrence is not particularly important. Completely missing an occurrence or incorrectly detecting a false occurrence, however, significantly undermines the usefulness of the system. Frame-level analysis will penalize systems for even the smallest error in the occurrence boundaries but will not penalize more severely for actual insertion or deletion errors. Frame-level analysis truly counts the number of mis-labeled frames and does not take the context into account. Therefore, it is often preferable to avoid using frame-level analysis to measure overall system performance but instead rely on event-level analysis or other, more nuanced summaries such as the SETs or EDDs mentioned previously [60, 94].

4.2 Optical Character Recognition

The first data set used for evaluation is made up of synthetically generated images containing different characters (see Figure 19). The goal of this data set is to provide a relatively easy discovery task suitable for testing the ability of different algorithms to locate motifs with widely varying lengths (*e.g.,* the character "i" is quite narrow while a "w" is several times wider). In total, 50 images were generated, each containing eight characters drawn from the alphabet {`zgwxfpmik`}. The characters are rendered with black letters on a white background using a basic serif font, and then the entire

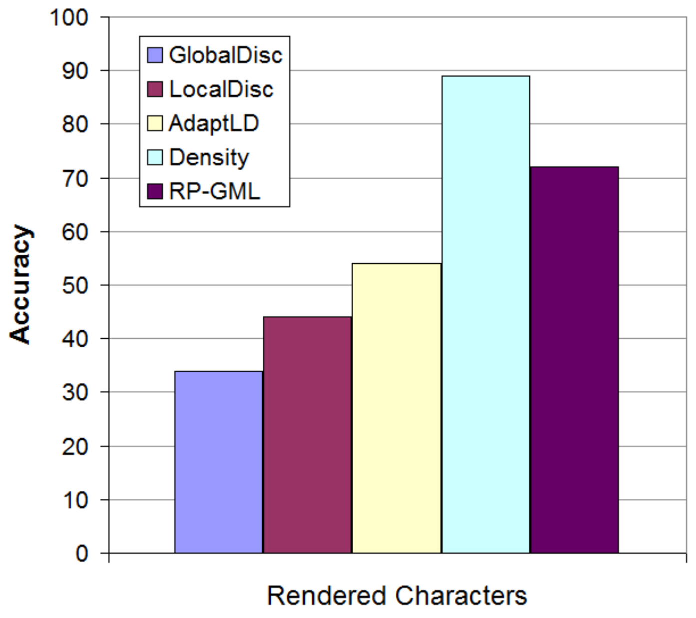

Figure 20: Accuracy results for the five algorithms compared in this research on the OCR data set. The two algorithms that model motifs using HMMs selected using a competitive learning framework perform the best due to their relative robustness to variable motif lengths.

image is corrupted by a small amount of monochromatic Gaussian noise. To convert the images into time series suitable for processing by our algorithm, a virtual sensor is used that scans the image from left to right. At each position, the virtual sensor extracts a multivariate descriptor based on the underlying pixel values. The sensor sees several pixel columns at each position and divides the vertical dimension into several bins. The sensor reading for each bin corresponds to the underlying pixels' mean intensity on a scale from zero (black) to one (white). Overall, the data set contains 50 sequences with 6,522 total frames and 15 dimensions. There are 400 total characters with roughly 44 examples of each of the nine possible characters. The characters vary in width from eight to 26 frames with an average length of 15.9 frames.

The graph in Figure 20 shows the performance of the five algorithms on the rendered character data. Throughout this chapter, the following names are used to refer to the algorithms described previously:

- **GlobalDisc** – the global discretization approach that uses suffix trees to detect patterns (described in Section 3.2)
- **LocalDisc** – the local discretization method that uses random projection to detect potential patterns and a fixed radius for motif neighborhoods (described in Section 3.3)
- **AdaptLD** – the adaptive version of the local discretization algorithm, which estimates the proper neighborhood radius from the data (described in Section 3.3.2)
- **Density** – the discovery algorithm that looks for local density maxima to identify candidate motifs and then uses a greedy mixture learning framework to select valid patterns (described in Section 3.4)
- **RP-GML** – the hybrid algorithm that combines random projection and greedy mixture learning (*i.e.,* the first phase from the `LocalDisc` algorithm and the second phase from `Density`; described in Section 3.5)

The results highlight the importance of explicitly supporting variable-length motifs. This data set was constructed to be very easy. There are no outliers, no temporal distortion, and minimal noise in the data set. Both methods that rely solely on fixed length subsequences (`LocalDisc` and `AdaptLD`) score poorly (44.0% and 54.0%, respectively) due to their inability to simultaneously model the very short motifs (such as `i` and `f`) and the long motifs (*e.g.,* `m` and `w`). Although the `GlobalDisc` method does support variable-length motif discovery, it performed even worse (34.5%). This poor result is likely due to difficulty in accurately modeling the range of virtual sensor readings with a small, global alphabet. The density-based method performed surprisingly well (89.0%) considering it relies on a search across fixed length subsequences to

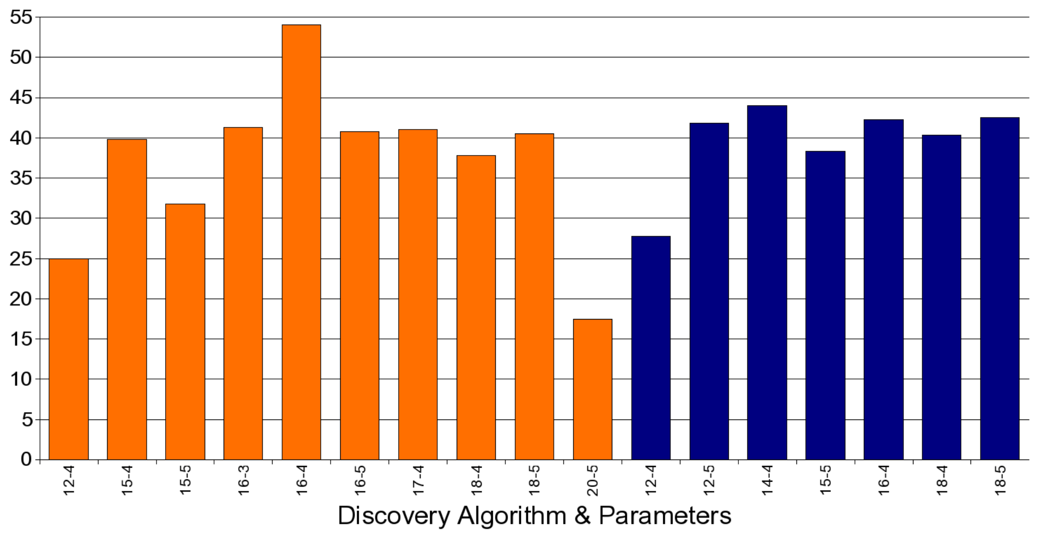

Figure 21: Accuracy of the `AdaptLD` (orange) and `LocalDisc` (blue) methods on the rendered character data across a range of parameter values. The parameters represent the window length and the number of PAA segments used. In each case, the accuracy reported is the maximum achieved via manual, supervised search across the relevant neighborhood size parameter.

find motif seeds. A major advantage of this method over `LocalDisc` and `AdaptLD` is that it does detect variable-length motif occurrences since the motif models can grow and shrink to fit the data[1]. Finally, the hybrid algorithm has an accuracy between that of the density-based approach and the local-discretization algorithms. This result implies that the HMM-based motif model is, in fact, superior to the hypersphere model, but that the random projection algorithm is not providing candidate patterns that are as accurate as those selected by the density-based algorithm.

Figure 21 provides a more detailed view of the performance of the `AdaptLD` and `LocalDisc` methods on the rendered character data set, shown using orange and blue bars, respectively. The graph shows accuracy over several runs using different parameter settings. In both cases, the first parameter represents the window length (ranging from 12 to 20 frames) while the second parameter represents the number of PAA segments used to compute the local discretization (ranging from 3-5 segments). In all of

[1]Preliminary experiments using a variant of the density-based algorithm that explicitly searches over variable-length subsequences shows accuracy over 99% at the expense of slower run time.

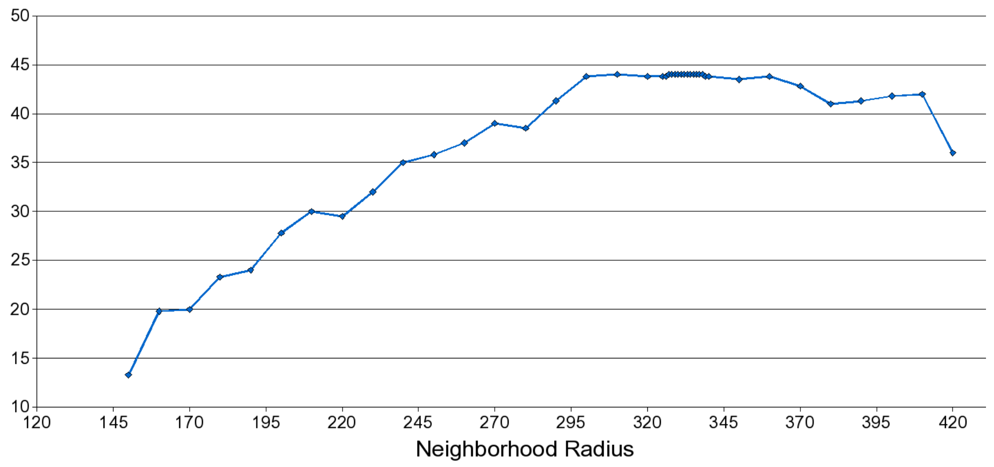

Figure 22: Accuracy of the `LocalDisc` method on the rendered character data across a range of neighborhood radius values. The reported accuracy is the maximum achieved after manually searching for the best settings for the other parameters. Note that values around 330 were more densely sampled after identifying that as a likely region to contain the maximum accuracy setting.

these trials, three SAX symbols were used for discretization. Additional experiments were performed using other parameter combinations and exploring different choices for the number of SAX symbols, but all other combinations led to results with lower accuracy.

Finally, Figure 22 shows the performance of the `LocalDisc` method over a range of neighborhood radius values using the best parameter settings from Figure 21. During experimentation, a wide range of values were tried, and the values around 320 were sampled more densely after that region appeared to be a promising region for a performance peak. The conclusion from these graphs is that that the low accuracy rates of the local discretization methods is not due to poor parameter choices. Instead, it is due to an algorithmic shortcoming, likely corresponding to the fixed-length motif assumption that does not hold for this data set.

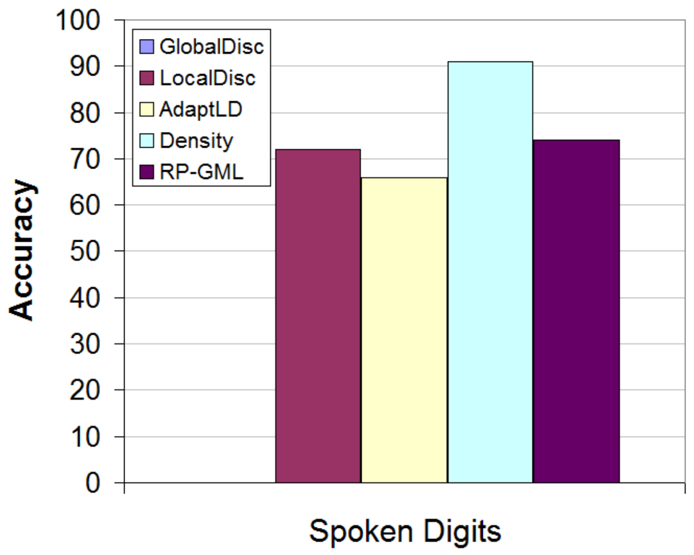

Figure 23: Accuracy results for the five evaluated algorithms on the TIDIGITS speech data set. This data set highlights the shortcomings of the `GlobalDisc` algorithm, which has a negative accuracy and thus is not visible on the graph.

4.3 Speech

The second data set used for evaluation purposes comes from the publicly available TIDIGITS data set, which contains audio recordings of spoken digits, originally intended to help evaluate automated telephone dialing speech recognition systems [44, 45]. For the purposes of evaluating the discovery algorithms considered here, analysis was performed on a subset of the available data consisting of 77 phrases composed of spoken digits from a single speaker. The data contains 11 classes (zero through nine plus "oh") and 253 total digit utterances or roughly 23 occurrences of each digit. The raw waveforms were transformed by extracting Mel frequency cepstral coefficients (MFCCs) over a range from zero to 4kHz using a frame period of 10ms and a window size of 25ms, which are all standard parameter choices for MFCC features in the speech recognition community. This feature extraction process leads to a final data set containing 11,917 frames.

All of the evaluated algorithms performed well on the TIDIGITS data set except for the `GlobalDisc` method. In that case, the algorithm actually has a negative

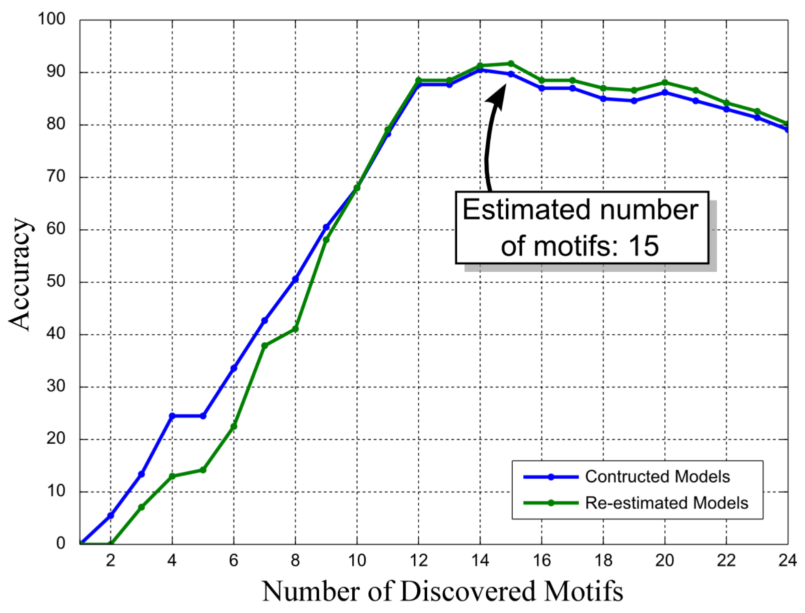

Figure 24: Graph of accuracy vs. number of discovered motifs for the TIDIGITS data set. The algorithm automatically selected 15 motifs for the TIDIGITS data (*truth:* 11 + background).

accuracy due to the detection of more false motifs occurrences than true ones (*i.e.,* a very large number of insertion errors). The poor performance is likely due to the difficulty of modeling the complex MFCCs in a relatively high dimensional space with a small set of discrete symbols. The quantization can induce insertion errors by representing a large region of space by a single symbol. Increasing the number of symbols can mitigate this problem, but that solution can easily lead to over-segmentation, which can cause the algorithm to separate a single pattern into multiple ones due to relatively small variations.

The best parameter setting for the adaptive local discretization method found 11 motifs with an overall accuracy of 68.0%, while the `LocalDisc` algorithm also found the 11 true motifs but with a slightly higher overall accuracy of 71.5%. Figure 23 summarizes these results, and Figure 25 shows two sequences from the data set with several correctly detected utterances of “one” highlighted. The lower performance of the adaptive algorithm is due to a difficulty in accurately estimating the neighborhood

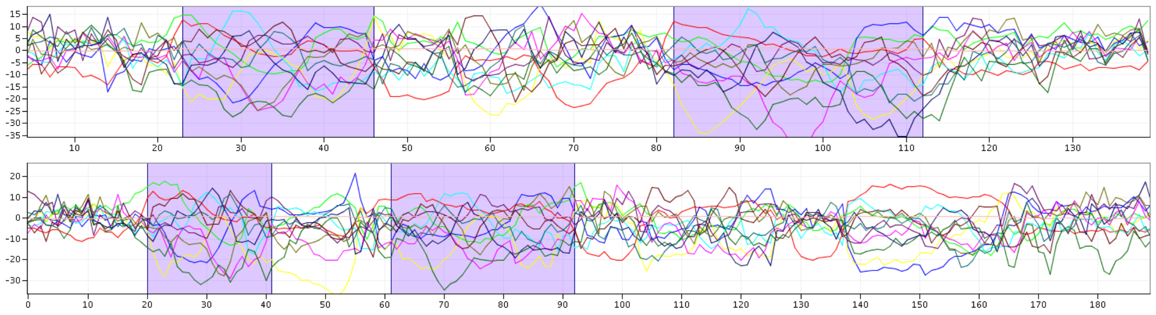

Figure 25: Two sequences from the spoken digits data set showing all 13 features. The highlighted segments represent occurrences of one of the discovered motifs, which corresponds to the utterance “one.”

radius. The estimation procedure requires a fairly distinct jump in distance separating true motif occurrences and non-motif segments, which corresponds to the assumption of fairly well-isolated motifs. The MFCC features in the speech domain can lead to a much more gradual shift, which could cause imprecise radius estimation and thus lower overall accuracy due to the resulting insertion (if the radius is over-estimated) or deletion errors (if it is under-estimated).

Figure 24 provides more information about the performance of the density-based discovery algorithm on the TIDIGITS data set. The algorithm overestimates the number of motifs (15 instead of 11) but the extra motifs did not interfere with the identification of the 11 spoken digits. Instead, they correspond to silence and other repeated noises that do not correspond to English words. The overall accuracy of the algorithm is 91.7%, which is significantly better than the other methods (as summarized in Figure 23).

The two accuracy curves shown in Figure 24 correspond to two variations of the density-based discovery algorithm. The blue curve corresponds to the performance of the “constructed models,” which gives the accuracy of the set of motif models as initialized from the k-nearest neighbors. The green curve corresponds to the performance of the “re-estimated models.” In this variation, each motif model is re-estimated using the Baum-Welch algorithm based on all of the segments that are matched by the

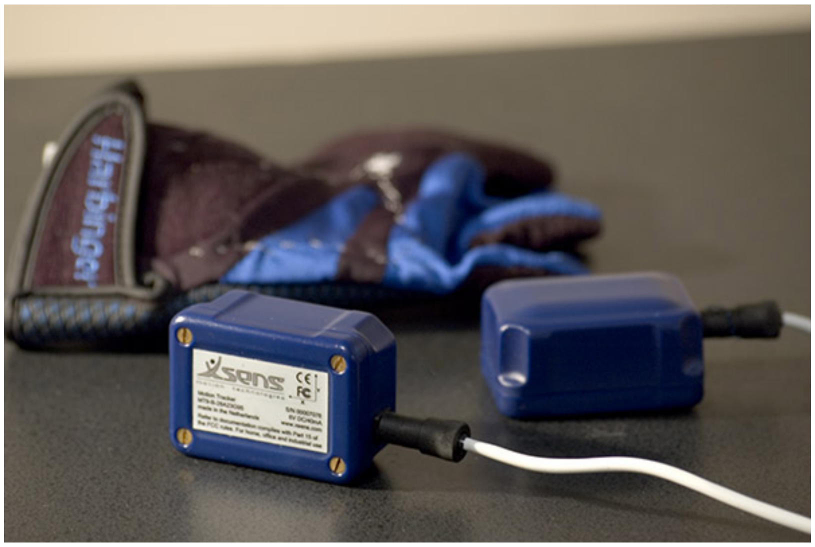

Figure 26: Two XSens MT9 inertial sensors and a modified glove used to mount a sensor on a subject's wrist. The MT9 measures acceleration and rate-of-rotation in three axes at 100Hz.

original, constructed model. In other words, the constructed models are added to the overall mixture model until the stopping criterion is reached. Then, all segments that correspond to each constructed motif model are extracted as training data. In the final step, each motif model is re-estimated using the corresponding training data and the updated mixture model is fit to the data. When too few motifs have been discovered, they often match extra, incorrect segments, which leads to a poor model parameters after re-estimation. This effect results in a lower accuracy rate for the re-estimated model version in the early stages of the discovery process. Once the system approaches and possibly exceeds the correct number of motifs, re-estimation can have a beneficial effect and lead to more accurate motif models. However, the improvement in accuracy is relatively small, and so it may not be worthwhile to re-estimate the models for time-critical applications.

4.4 Exercise via On-Body Inertial Sensors

The exercise data set consists of accelerometer and gyroscope readings from an on-body sensor that captured a mock exercise routine comprised of six different dumbbell

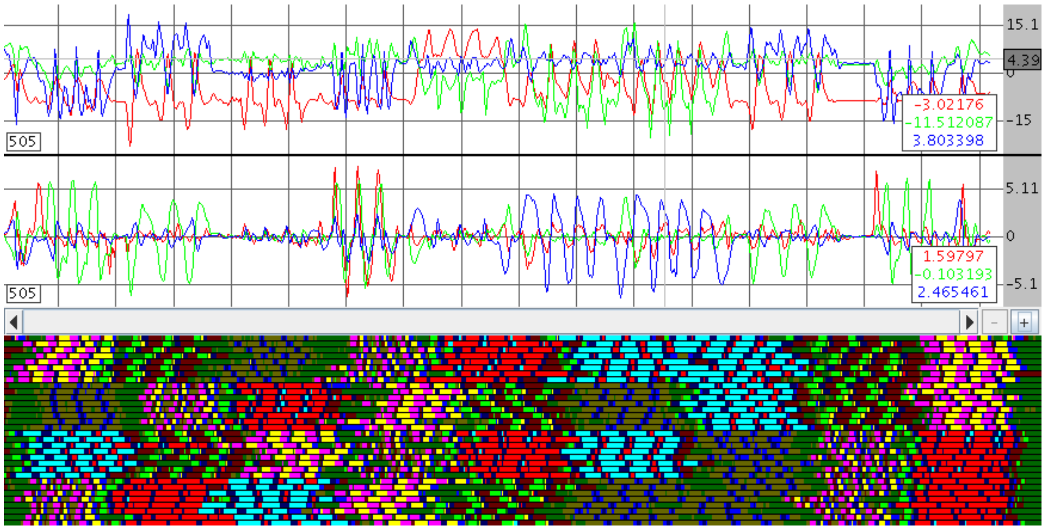

Figure 27: A visualization showing the raw accelerometer and gyroscope data from one sequence in the top portion and the quantized data from all 32 sequences in the bottom portion. The apparent regularity in the data is an artifact of orderly collection and does not help the discovery algorithm.

exercises (`back_row`, `flat_curl`, `twist_curl`, `shoulder_press`, `shoulder_extension`, and `tricep_extension`). An XSens MT9 inertial motion sensor was attached to the subject's wrist by fitting it into a pouch sewn to the back of a thin glove (see Figure 26). The MT9 sensor was sampled at 100Hz and recorded three-axis accelerometer and gyroscope readings. In total, 32 sequences were captured with a total of 166,582 frames resulting in 27 minutes and 46 seconds of data. For the experiment, however, the data was down-sampled to 12.5Hz leading to 20,711 frames. Given the expectation that each exercise is a different motif, the data set contains six motifs and 864 total repetitions or roughly 144 occurrences of each exercise.

For the `GlobalDisc` algorithm, the six-dimensional frames were quantized using a mixture of 10 Gaussian distributions as described in Section 3.2.1. Figure 27 shows the raw data for one of the sequences as well as the quantized representation of all 32 sequences. Each of the exercises shows up clearly in the color-coded representation of the quantized data. Note that the apparent block structure is due to the uniformity of the collection process. For the methods that use the hypersphere motif model

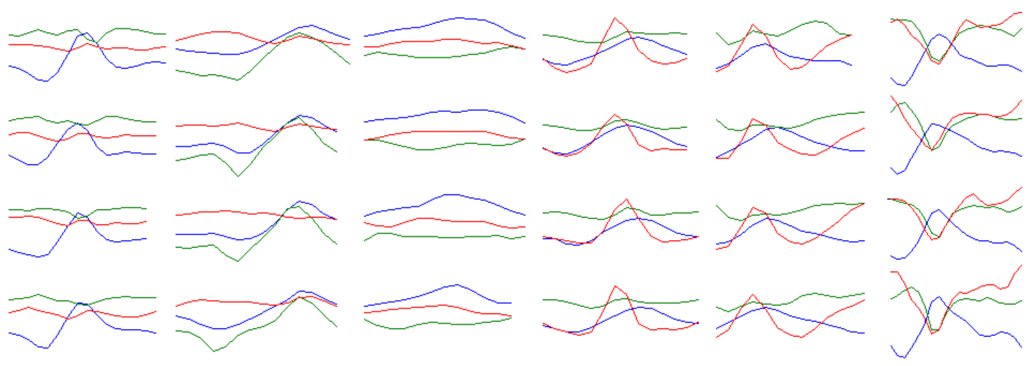

Figure 28: Sample occurrences of the six motifs discovered in the exercise data set showing the three-axis accelerometer readings. Each column corresponds to a different discovered dumbbell exercise.

(`LocalDisc` and `AdaptLD`), the dynamic time warp (DTW) distance was used with a Sakoe-Chiba band of 10%, which follows standard practice in the field, although some authors argue that a smaller, less flexible band is actually preferable in many situations [80].

All of the evaluated algorithms correctly discover the six motifs with high accuracy (see Figure 29 for a summary and Figure 28 for example motifs). As with the OCR data set, the HMM-based motif models outperformed the hypersphere models, and the greedy mixture learning approach (used by the `Density` and `RP-GML` algorithms) achieve higher accuracy than the direct detection method used by the `GlobalDisc` method.

The `GlobalDisc` algorithm successfully locates 96.3% of the occurrences, which corresponds to finding 832 of the 864 occurrences. It also locates 51 false occurrences (insertion errors) in addition to missing 32 real occurrences (deletion errors), but there are no substitution errors. Overall, the algorithm achieves an accuracy of 86.7% and precision of 88.4%.

The lack of substitution errors is likely due to the fact that the exercises are very distinct in the continuous domain. Note, however, that the `flat_curl` and `twist_curl` exercises were confused in the discrete domain, but the two motifs were

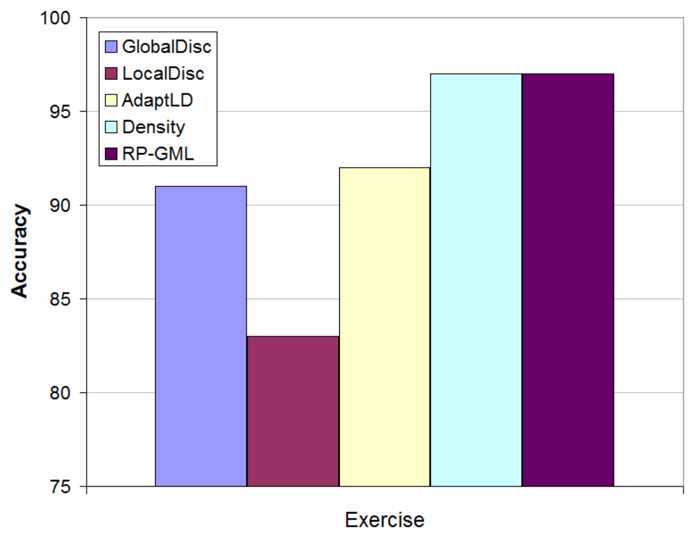

Figure 29: Accuracy results for the five evaluated algorithms on the exercise data set. Note that the graph starts at 75% accuracy. Although all of the algorithms perform well, using HMMs as motif models provides better accuracy results than using the hypersphere model in all cases.

then automatically split from two conflated motifs into four homogeneous ones during refinement. The algorithm then detected the need to merge the four motifs into two homogeneous clusters that correctly represented the two exercises.

Figure 31 shows the result of running the `LocalDisc` algorithm over a range of values for the neighborhood radius. The two curves represent two different choices for the distance metric: a straightforward sum of squared error (SSE) calculation and dynamic time warping (DTW). The accuracy rate of the two metrics peaks at different values. This result is expected because the user-specified threshold is given in terms of raw distances. More importantly, although performance is fairly smooth around the peaks, there are large fluctuations that correspond to instabilities in the algorithm (*e.g.*, around 5.0 and 6.0 for the SSE metric).

The performance of the `AdaptLD` algorithm is shown in Figure 32. The event-based accuracy is measured across a range of estimation portion sizes, which determines how much data is used to estimate the neighborhood radius for each motif. Specifically,

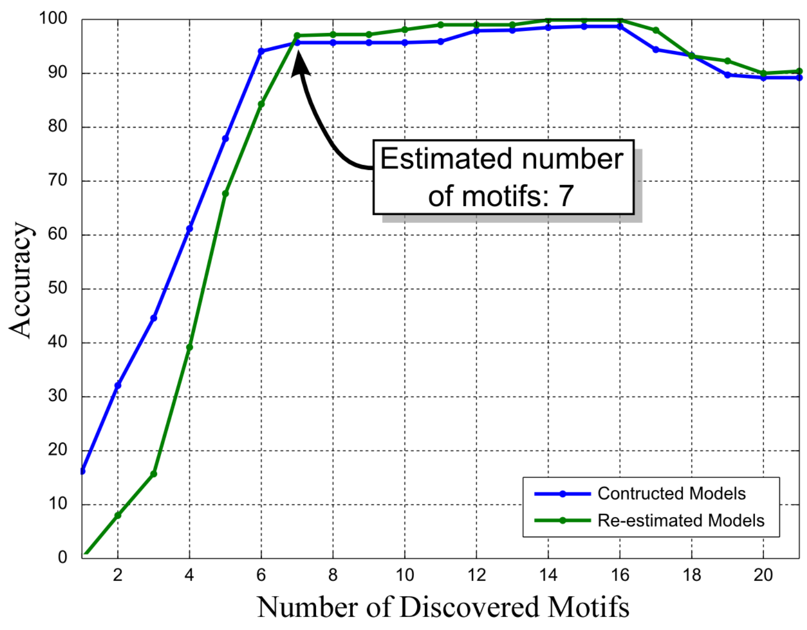

Figure 30: Graph of accuracy vs. number of discovered motifs for the exercise data set. The density-based algorithm automatically selects seven motifs for the exercise data (*truth:* six + background).

each portion size corresponds to the percentage of the distances, after sorting in increasing order, that is used to locate a knee in the distance curve and thus determine the neighborhood size. In addition to allowing different neighborhood sizes for different motifs, the portion size parameter is intended to be easier to specify than the distance threshold used by the `LocalDisc` algorithm because it does not directly depend on the underlying distance metric. Th graph in Figure 32, however, shows that the relationship between accuracy and portion size does still depend on the distance metric but the overall range is much smaller, performance stays high across a wider region, and performance is more stable.

For all of the other experiments performed in this section, the DTW distance metric was used with a value of 10.0% for the portion size because this choice led to high accuracy rates and is in the central part of a stable region. Similarly, the `LocalDisc` algorithm also uses the DTW distance metric. In this case, a radius of 9.25 was used since it was found to yield the best performance. Given these

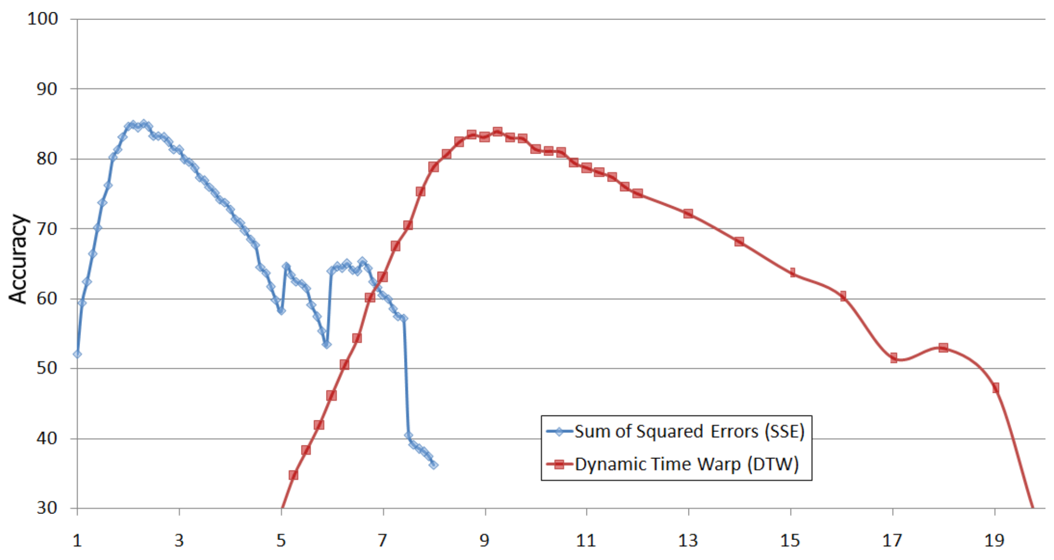

Figure 31: Graph showing the accuracy rate of the `LocalDisc` algorithm for a range of radii. Performance based on both the DTW and a straightforward squared error distance metric are included.

optimized parameter settings, the `AdaptLD` algorithm achieves an accuracy rate of 91.7% compared to 83.9% for `LocalDisc`, which corresponds to a 48.4% reduction in error.

The density-based and hybrid algorithms perform similarly and both achieve higher accuracy rates than the earlier methods (97.0% accuracy vs. 91.7% for the `AdaptLD` method). Figure 30 shows the performance of the density-based algorithm as the number of allowed motifs is increased. The exercise data set contains six expected motifs, and so accuracy is poor when the number of discovered motifs is artificially capped at values smaller than six since some true patterns are necessarily left undetected. Note that the algorithm automatically estimates that there are seven motifs. After manually analyzing the discovered patterns, it is clear that they correspond to the six expected motifs (*i.e.*, the actual exercises) with one additional motif that models a rest state.

For a more detailed summary of the performance of the algorithms on the exercise data, see Figure 33. This set of graphs shows four measures of performance for the different discovery algorithms. Each graph includes a confidence interval set at a

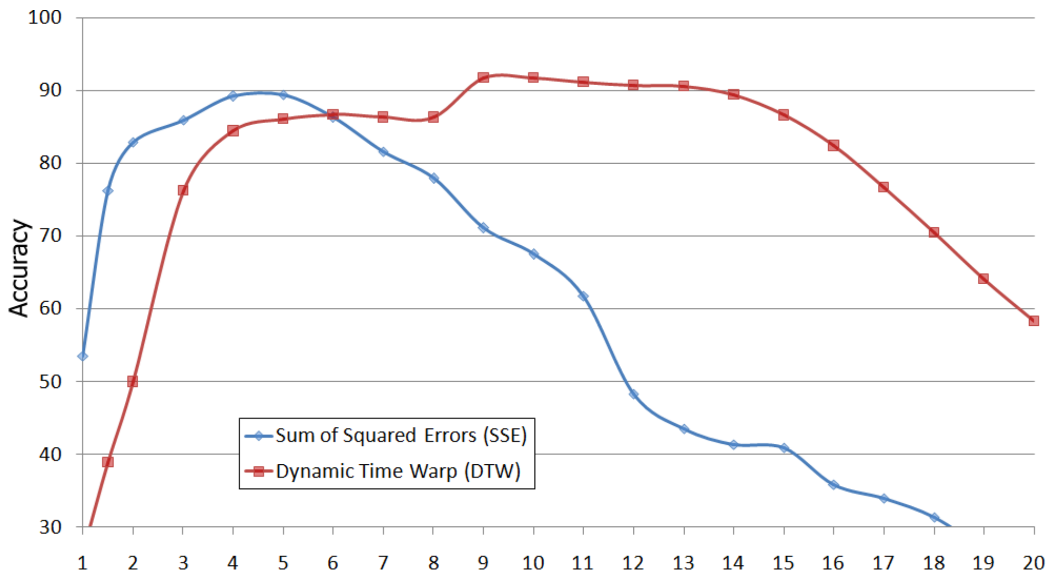

Figure 32: Graph showing the accuracy rate of the `AdaptLD` algorithm for a range of analysis proportions. Performance based on both the DTW and a straightforward squared error distance metric are included.

distance of one standard deviation from the mean performance level. These results were compiled from 16 separate trials where each trial used a bootstrap sample of 24 randomly selected sequences taken from the full set of 32 sequences in the exercise data set. The discovery algorithm was run independently for each trial, and then the mean and variance of the performance across the trials was computed. The graphs reinforce the results depicted in Figure 29, while the confidence interval gives more information about the variation in performance for different subsets of the data. Furthermore, the variation can be used to assess the significance of the accuracy differences by using a one-sided t-test[2]. The significance tests found that all of the differences are statistically significant with very small p-value with the lone exception of the improvement of `RP-GML` over `AdaptLD` where $p = 0.0335$. Although much less significant than the other performance differences, it is still sufficient to be considered statistically significant at the widely-used level of $p = 0.05$. Refer to Table 2 for a

[2]Specifically, we used the version of the t-test that allows for different variances; see Section 14.2 of [77] for details.

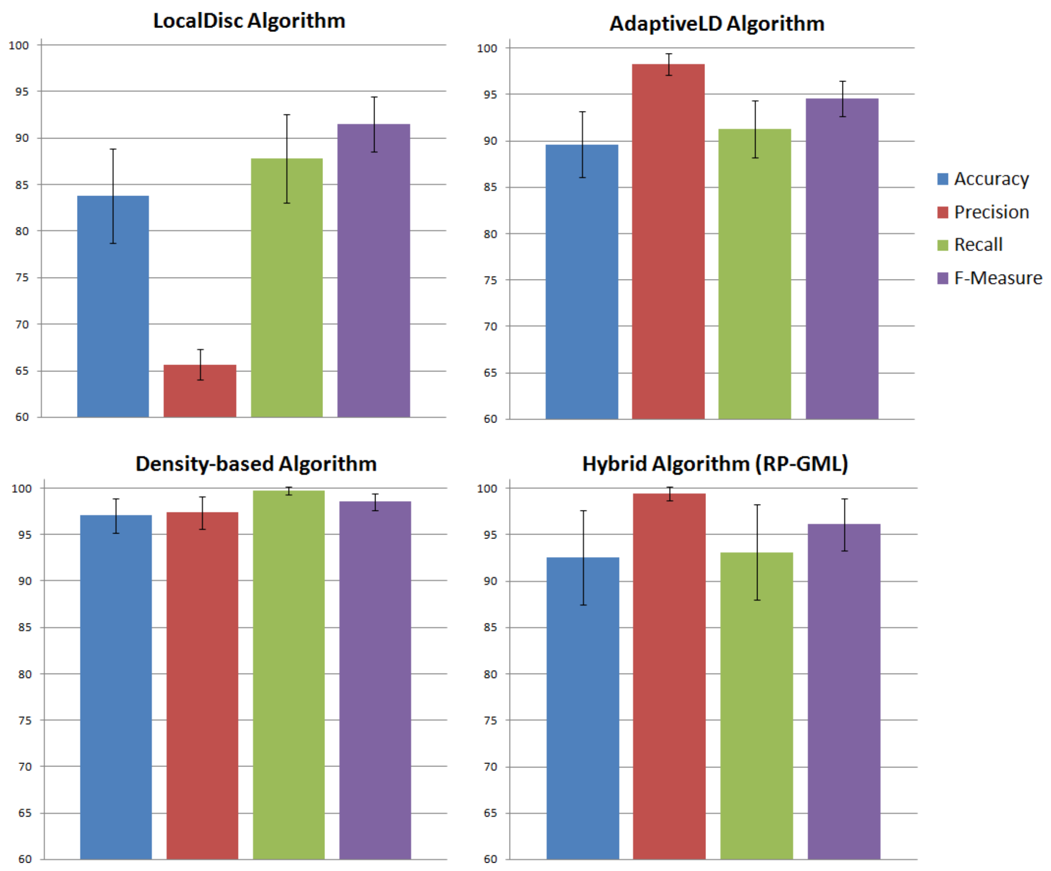

Figure 33: Each of the four graphs shows performance (accuracy, precision, recall, and f-measure) for a different algorithm. The results were averaged over 16 trials, each composed of 24 sequences randomly sampled from the 32 sequences in the exercise data set. The confidence interval corresponds to +/- one standard deviation across the 16 trials.

summary of all of the significance levels.

4.5 American Sign Language

The American Sign Language (ASL) data set is made up of 500 signed sentences captured by a head-mounted video camera that recorded 320x240 images at 10Hz (see Figure 34). The data was collected as part of a sign language recognition research project by Starner, Weaver, and Pentland [87]. Each frame of video was independently analyzed to detect skin-colored blobs that were classified as the left hand, right hand, or combined hands. The hand blobs were analyzed in order to compute an 8D feature vector that includes the centroid of the blob, mass, primary angle, eccentricity, the

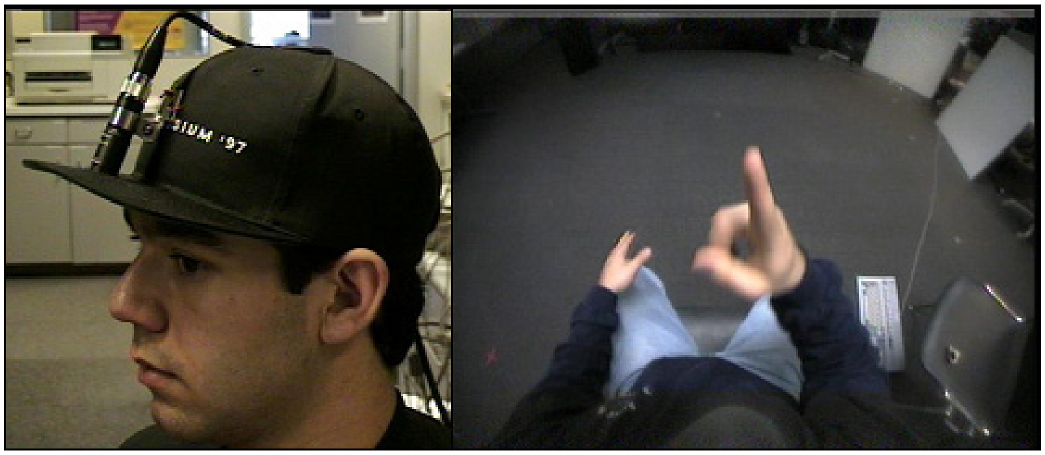

Figure 34: The ASL video was captured using a hat-mounted video camera (*left*) that captured a top-down, wide-angle view of the signer's hands (*right*).

length of the principal component, and the differential between the location in the current frame and the previous one. The analysis was performed separately for each hand, leading to a 16D feature vector, and when the hands were merged, the single set of features were recorded for both the left and right hands.

For the purposes of evaluating the discovery algorithms, the original features were left unchanged except for normalization to ensure that each feature had a similar range. For instance, the mass of the blobs is typically quite high since there were a relatively large number of pixels in each frame, while the angle and other statistics are typically quite small.

Each of the 500 sentences contains a single five word phrase composed from a vocabulary of 40 signs. While the words in each sentence are known, the precise boundaries between them were never manually labeled. Thus, the expected labels were

Table 2: Significance Levels for Accuracy Difference on the Exercise Data

	LocalDisc	AdaptLD	Density
AdaptLD	4.316e-4	–	–
Density	3.689e-9	7.167e-8	–
RP-GML	1.70e-5	0.0335	1.76e-3

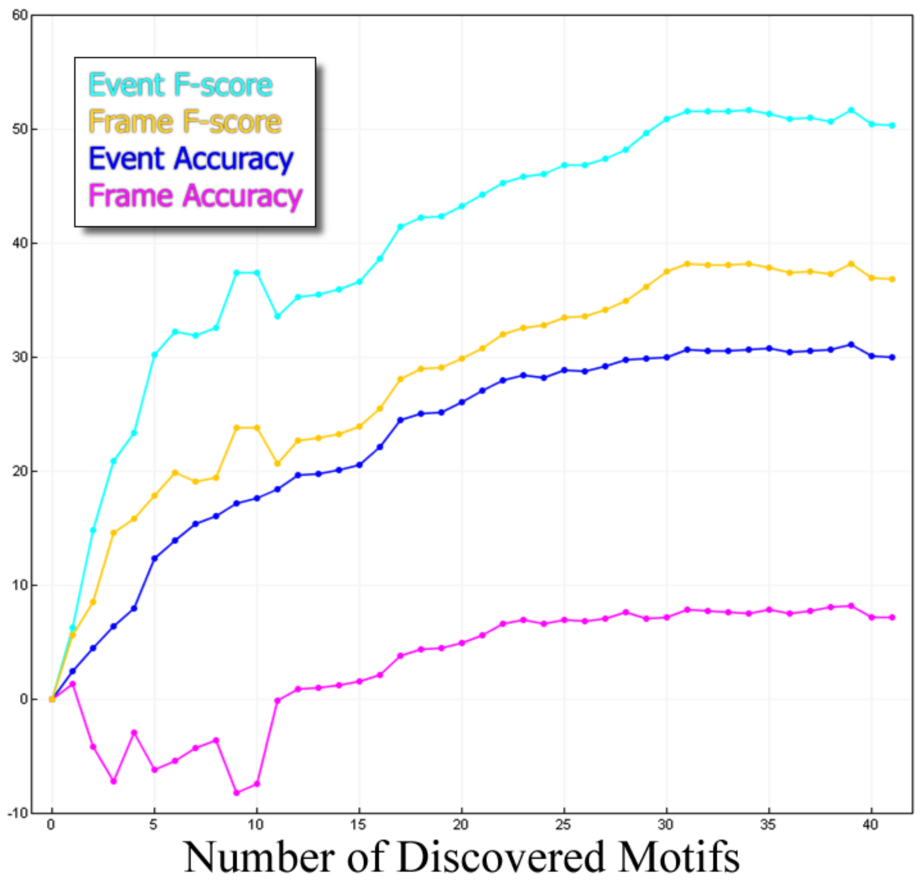

Figure 35: Performance of the hybrid algorithm on the ASL data set. The four curves provide the event and frame-based performance rates according to both the accuracy and F-measure statistics. The event-based accuracy (blue curve) is comparable to Figures 24 and 30.

learned from the known sentence structure and a supervised detection process. The algorithm simultaneously learns a HMM for each sign while adjusting the boundaries to best fit the data. Although this procedure is imperfect, it does provide reasonably accurate occurrence boundaries that are sufficient for the event-based evaluation that is used here.

Although the final accuracy rate on this data set is only 30%, achieving this rate is actually quite difficult on the ASL data set. First, there are a large number of signs, and the algorithm does a reasonable job by detecting 22 of the 40 real signs and only locating 19 additional patterns. Furthermore, the 10Hz sampling rate of the video

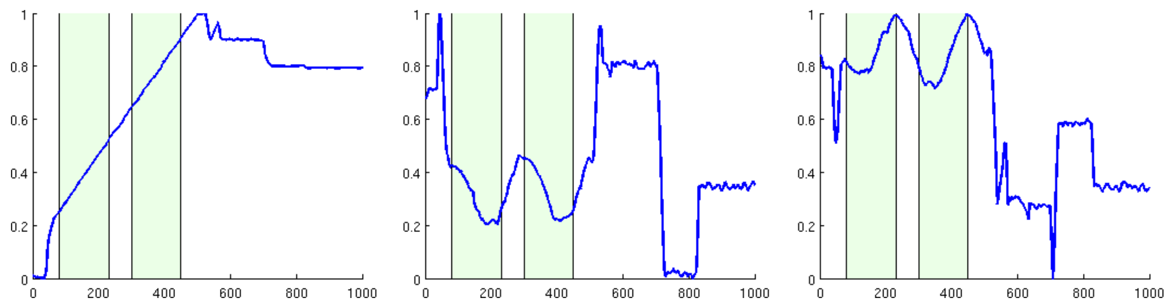

Figure 36: Six dimensional shuttle telemetry data with the discovered pattern highlighted (only the first three dimensions are shown)

does not provide much data for each sign, which typically lasts for around one second. The view of the camera is also not ideal for sign understanding as the language was designed for face-to-face communication instead of top-down interpretation.

4.6 Other Domains

In order to demonstrate the generality of the motif discovery algorithms, the methods were applied to data sets from other domains where the ground truth labels are unknown. The first two data sets have been used to demonstrate the performance of other methods in the the data mining community and are publicly available through the UCR Time Series Data Mining Archive [36]. The first data set represents shuttle telemetry data in which the algorithm discovers a single motif with two occurrences (see Figure 36). Although I am not able to directly interpret this motif due to a lack of domain knowledge, this result does match what was reported by other motif discovery researchers [10, 89].

The second data set is also taken from the UCR archive and represents a recording from a fetal ECG. The data has 2,500 frames and eight dimensions. Here, the algorithm finds two motifs corresponding to the thirteen occurrences of the heartbeat (see Figure 37) and then the "silence" between them.

Next, we analyzed data collected by a body-worn GPS unit. The data was originally collected as part of a research effort to automatically learn significant places

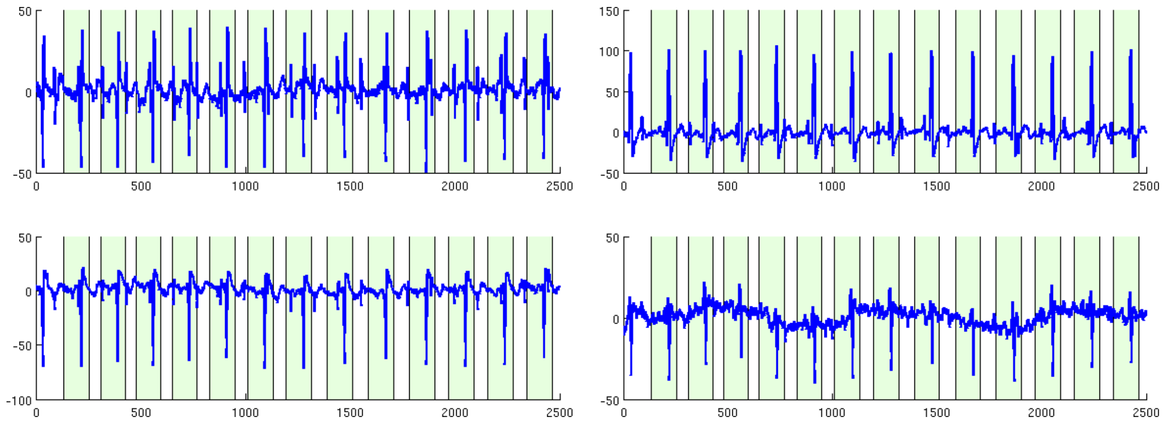

Figure 37: The primary motif discovered in the 8D fetal ECG data (the first four dimensions are shown)

in the wearer's daily life as described in Section 2.5. Here, the goal is to analyze the GPS paths instead of trying to cluster static locations heuristically selected as probable destinations. The primary difficulty in analyzing GPS data is the extreme level of variation in the duration of the paths encountered. For instance, a subject may drive to work (20 minutes), walk to the cafeteria (2 minutes), take a weekend get-a-way (a 3 hour drive), or drive to a conference (8 hours). Although the `Density` and `RP-GML` algorithms do allow for some temporal variation, neither supports the amount of variation found in GPS data. As such, the results of applying the discovery algorithm are not surprising: it did detect recurring routes, but it is clear that they are partial paths (see Figure 38 for an example).

Section 4.3 described the results of an experiment with speech data. The TIDIGITS data set was used, which is made up of high-quality recordings in a controlled environment and which has a small, known vocabulary. In this section, we describe additional experiments with more complex, realistic speech data. The first data set is a political lecture recorded at MIT. The recording is over 97 minutes long and has 585,982 samples after standard MFCC and log-energy features are extracted. The analysis first segments the data set by detecting silent regions using a conservative threshold on the log-energy feature. Then, the `RP-GML` algorithm is applied to the 12

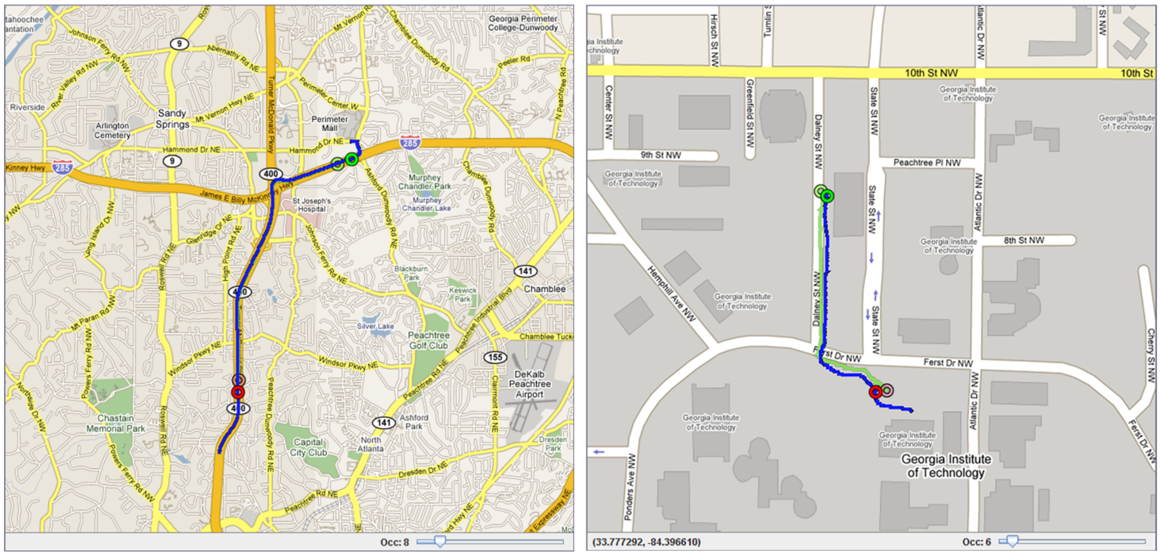
Figure 38: Two recurring paths in the GPS data. *Left:* a common driven route along the highway; *Right:* a recurring walking route between buildings on the Georgia Tech campus
.

MFCC features.

Because we do not have a transcript of the lecture and did not manually label the repeated words and phrases, there is no quantitative evaluation of the overall performance. Instead, a qualitative assessment follows. The first observation is that the algorithm was able to discover repeated words and phrases. For example, the bottom row of Figure 39 shows a spectrogram of three occurrences of a discovered motif that corresponds to the phrase "United States." The spectrogram is a matrix where each entry corresponds to the power in a certain frequency range at a particular time. Time progresses from left to right, while the frequency bands increase from the bottom to the top of the image. Each entry is color-coded according to the energy level, where the progression moves from yellow (low power), through blue, and finally to red. Although the discovery algorithm analyzes the raw MFCC features directly, the spectrogram provides a useful visualization that makes the similarity clear.

Although not as extreme as with the GPS data, the limited ability of the `RP-GML` algorithm to model motifs of different lengths complicated the analysis of this audio data. For instance, when a relatively short duration was specified (*e.g.,* 800ms), the system would typically find short words and parts of words, while longer durations (around 1,500ms) would find word phrases. Without the ability to search at multiple time scales or to temporally extend the initial matches (as discussed in Section 6.4), the system was unable to locate both short words and longer phrases.

In addition to words, the system also detects other repeated noises. For example, it detected recurring disfluencies such as when the speaker said "um," as well as instances when the audience was clapping. In the case of clapping, a typical human listener might only identify a single, relatively long period of clapping, but the discovery algorithm does not model such *fluid* patterns that have no well-defined duration. Instead, if it encounters five seconds of clapping with the analysis duration set to one second, it will likely interpret the data as five separate instances of clapping. This distinction between patterns with relatively stable durations and those that are highly variable arises quite often in the context of gesture and activity recognition. For example, a sign or other command gesture has a well-defined duration, while walking, waving, or shaking hands are periodic and may extend arbitrarily long (although social norms may impose a practical limit for waving and shaking). When both kinds of gestures are expected, they can be modeled by extracting appropriate features (often wavelet or Fourier coefficients are used) or by explicitly learning models that capture the repetitive nature of the gesture, such as an HMM with a left-right topology except for a single additional transition from the last state back to the first. None of the discovery algorithms discussed in this work support both kinds of patterns, although some other methods that seek to model broader classes of activity are able to discover "fluent" patterns (*e.g.,* [68] and [30]).

One problem that arose when analyzing the lecture data was that the learned motif models would match to far more segments than there were real occurrences of the dominant utterance (*i.e.,* there were a large number of insertion errors). In

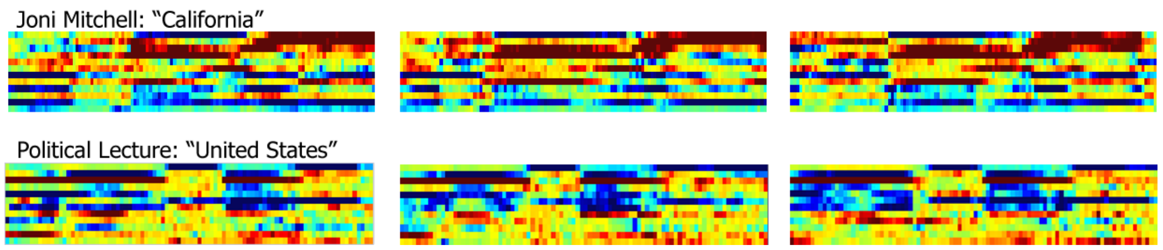

Figure 39: The top row shows spectrograms for three occurrences of the discovered word "California" in a recording of Joni Mitchell's song *California*, while the bottom row shows spectrograms for occurrences of the phrase "United States" in a recorded political lecture.

many of the discovered patterns, the first few detected occurrences, when sorted by distance from the seed occurrences or by probability given the learned motif model, would correspond to the same utterance. After the first few occurrences, however, the utterances would change to similar sounding but different words, and then finally into phrases that had no obvious similarity to the original phrase. I believe this problem is due to a poor background HMM, which is far too simple to yield a reasonable model for the regions of audio that were not yet modeled by other motif HMMs. When analyzing the TIDIGITS data, however, this problem was avoided because the recording was very clean, the vocabulary was small, and the overall data set was relatively short. In general, then, a better background model is needed to achieve high accuracy rates on realistic speech data with discovery algorithms that use a competitive learning framework.

In addition to the lecture data, we also analyzed musical audio recordings. The motivation for this domain was to investigate how the system would handle audio data with continuous high volume. In particular, some audio analysis methods have a practical requirement that the data be pre-segmented in order to maintain efficiency. This requirement is reasonable for relatively clean speech recordings due to the natural, frequent pauses that occur when speaking. The primary benefit of the volume-based segmentation is that the processed signal will typically be split into many relatively short time series (*e.g.,* on the order of two seconds each), which means that algorithms with a quadratic complexity can be applied to each short segment while maintaining an overall complexity that is sub-quadratic. Such methods break down, however, if the pre-segmentation is not possible. Although we did not perform a full quantitative evaluation, the `RP-GML` algorithm did run efficiently on music data where pre-segmentation is not possible. The top row of Figure 39 shows three examples of the word "California" from Joni Mitchell's song by the same name. As with the "United States" example from the lecture data, the similarity between the three occurrences is clear from the spectrogram. Interestingly, one can also see the large amount of energy in the high frequencies, a trait for which Joni Mitchell singing is well-known.

CHAPTER V

SUBDIMENSIONAL DISCOVERY

The motif discovery algorithms discussed in previous chapters have made an implicit assumption that all of the features in the time series data are relevant to each pattern. For the many univariate discovery algorithms that have been developed, this assumption is trivially accurate, but for the multivariate algorithms considered in this research, the assumption may be overly restrictive. To address this shortcoming, this chapter presents the problem of *subdimensional motif discovery*, and then a subdimensional algorithm is described that relaxes the previous assumption of global feature relevance.

Figure 40 provides a graphical depiction of four categories that successively reduce the assumptions about how the motifs relate to the features in the time series data. All of the algorithms presented in Chapter 3 address the problem depicted in Figure 40a where all of the features are relevant to all of the patterns. The first generalization is depicted in Figure 40b where some features are irrelevant. This effect is global, however, so it would suffice to detect the irrelevant dimensions and then run a traditional algorithm on only the relevant subset. Figure 40c removes the global relevancy assumption. Thus, the subset of the features that are included in each pattern is determined on a per-motif basis. The motifs can not temporally overlap, however. The final generalization, shown in Figure 40d, relaxes the temporal exclusion assumption and permits motifs to occur simultaneously so long as their relevant features are disjoint.

Subdimensional motifs arise in many circumstances. For example, in a distributed sensor system, patterns may arise independently in spatially separated sets of sensors.

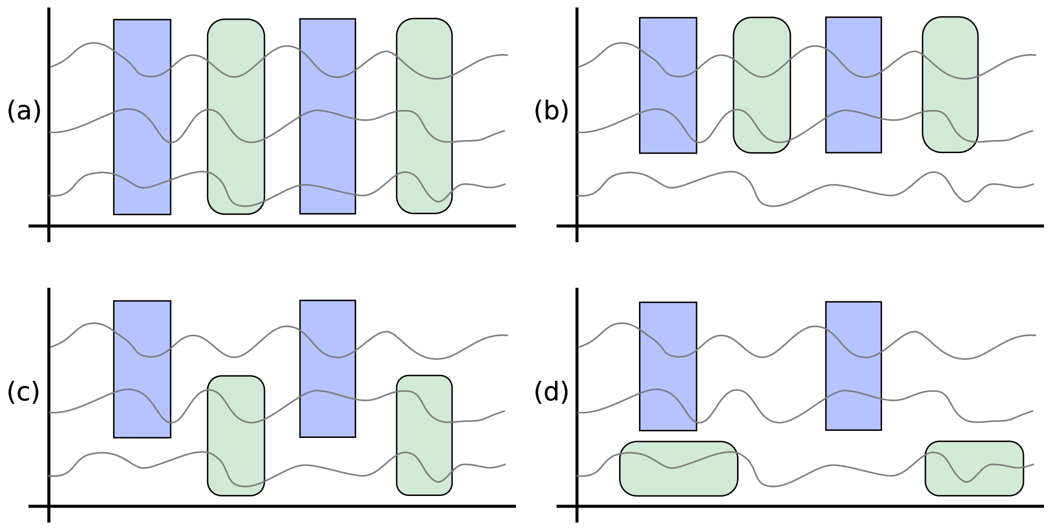

Figure 40: Four categories of multidimensional patterns: (a) “all-dimensional” patterns include all of the features, (b) a subset of the dimensions are relevant to all patterns, (c) each motif can have a different subset of relevant dimensions, and (d) feature relevance depends on the motifs and the patterns can temporally overlap if they include disjoint features.

In many cases, the independent subsets may not be obvious to the system engineer at design time, thus motivating the simultaneous discovery of statistically significant temporal patterns along with the relevant features. For computer vision applications, even though the individual photo sites are physically quite close, a wide field of view may allow the camera to capture activities that occur in different areas in the environment. The relevant regions in the video may vary over time and be difficult to predetermine even if the camera location is fixed. When analyzing data collected by on-body sensor systems or motion capture rigs, subdimensional motifs may arise when a recurring pattern generated by one part of the body, say the right arm, is decoupled from the behavior of other limbs or from other biometric sensors. As a final example, subdimensional motifs may arise when analyzing financial indicators in which different patterns involve disparate subsets of the underlying securities.

The main benefit of subdimensional motif discovery algorithms is that these methods can find patterns that would remain hidden to typical “all-dimensional” multivariate algorithms. The ability to automatically detect the relevance of each dimension

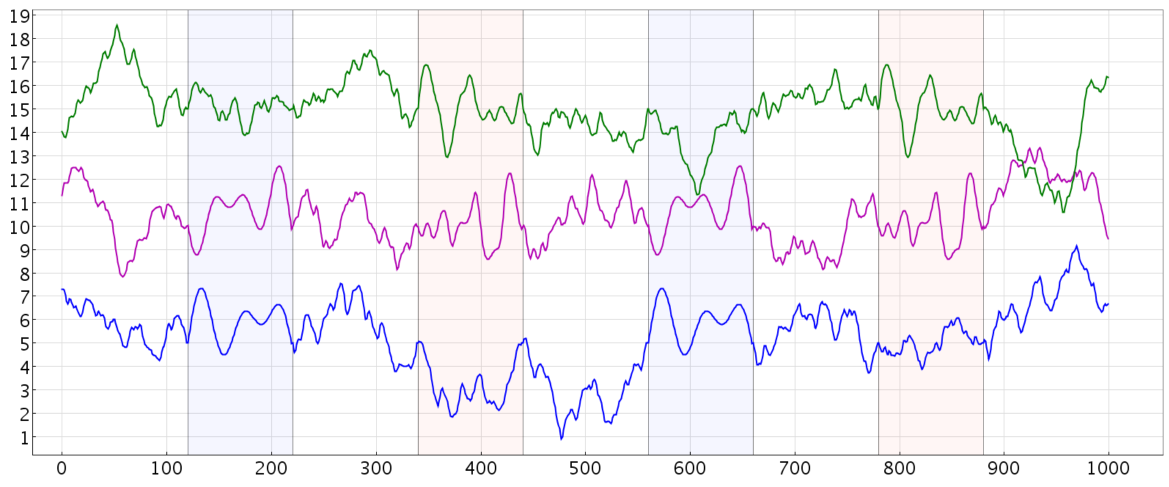

Figure 41: A three-dimensional signal with two subdimensional motifs. The first motif only spans dimensions two and three, while the second motif spans dimensions one and two.

on a per-motif basis allows great flexibility and provides data mining practitioners with the freedom to include additional features, indicators, or sensors without requiring them to be a part of every pattern. Subdimensional discovery also provides robustness to noisy or otherwise uninformative or distracting sensor channels.

5.1 *Subdimensional Discovery Algorithm*

The algorithm presented in this chapter addresses the subdimensional discovery problem up to the level of generality depicted in Figure 40c. The approach builds on the basic algorithm used in the `LocalDisc` and `AdaptLD` algorithms presented in Section 3.3. The algorithm starts with the same SAX-based local discretization method and uses random projection with a collision matrix to detect similar regions of the given time series data. However, where the earlier algorithms concatenated the SAX words generated in each dimension before applying random projection, the subdimensional algorithm projects the words separately. Then the corresponding entry in the collision matrix is incremented for each dimension that matches instead of at most once. This change can be understood as a switch from a logical `AND` policy in the all-dimensional case (*i.e., all* dimensions must match to qualify as a collision) to a

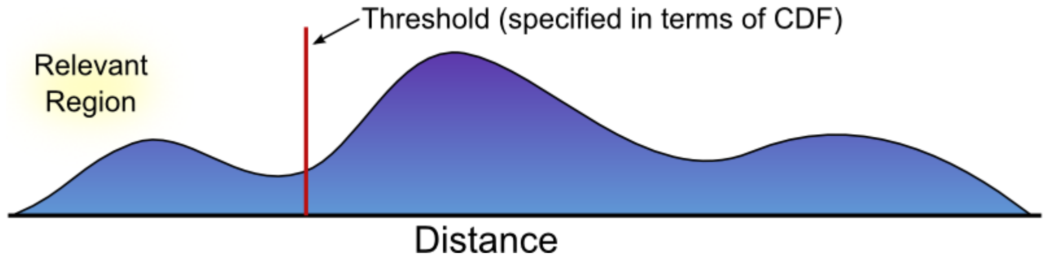

Figure 42: A feature is considered relevant to a particular seed motif if the seed distance in that dimension has high probability under the CDF estimated from random samples from the overall data set.

logical `OR` policy (*i.e.,* a collision occurs if *any* of the dimensions match).

This change in how the random projection algorithm is applied to the SAX data has two important effects. First and foremost, the collision matrix entries will now represent similarity across arbitrary subsets of the data features, which was the primary goal. The downside, however, is that the collision matrix can easily become dense, and thus have quadratic time and space complexity, since it is much more likely for two subsequences to collide under the `OR`-based policy. To combat the increase in complexity, the subdimensional algorithm automatically monitors the collision matrix and halts any searches that appear to be leading to a dense matrix. Upon detection, the algorithm adapts the discretization and projection parameters to reduce the number of collisions. For instance, it will increase the number of discrete symbols, which corresponds to dividing the real-valued data into smaller, more precise divisions, thus leading to a smaller chance of a match between two different regions.

The `OR`-based collision matrix policy allows the algorithm to detect similar subsequences even if some of the features do not match. It does not provide information about which features are actually relevant, however. To make this determination, the algorithm must analyze the subsequences that correspond to large entries in the collision matrix. Feature relevance is determined independently for each dimension. When the neighborhood radius for each motif is automatically estimated, as in the

`AdaptLD` algorithm, each feature distance in a candidate seed is compared to the overall data distribution. As a pre-processing step, the algorithm randomly samples subsequences from the data set and estimates the overall distribution of pair-wise distances for each dimension (see Figure 42). Then, during the discovery process, the distance between corresponding dimensions in a candidate seed motif is computed and compared to the estimated distribution. If the probability of the distance is large, then it was likely to have arisen randomly and so the dimension is probably irrelevant. If, on the other hand, the probability is low, then it is considered surprising and thus relevant. The classification is determined by a user-specified probability threshold, and the probability is computed via the cumulative distribution function (CDF) for the density estimate. In the experiments presented in Section 5.2, a Gaussian distribution was used to model the distribution of distances in each dimension, and the threshold was set so that relevant distances were smaller than 80% of the random distances:

$$\int_{-\infty}^{d} p(x)\, dx \leq 0.2$$

where d is the distance in question and $p(x)$ is the distribution of distances in the overall data set. For the Gaussian case, we have:

$$\int_{-\infty}^{d} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = \frac{1}{2}\left(1 + erf\left(\frac{d-\mu}{\sigma\sqrt{2}}\right)\right)$$

Note that in the general case a non-parametric kernel density estimate is straightforward to implement and can provide a more accurate model than the Gaussian model that proved sufficient here.

When the subdimensional algorithm is given a fixed neighborhood radius as in the `LocalDisc` algorithm, dimension relevance is much simpler to compute. In this case, no distance distribution is needed. Instead, the distance between the two subsequences in the candidate seed pair in each dimension are sorted in ascending order. They are then greedily selected until the total, multidimensional distance exceeds the user-specified threshold. Then the subset of dimensions that do not exceed the threshold are deemed relevant. In both the fixed and estimated radius case, the rest of the algorithm proceeds in the same way that the corresponding "all-dimensional" version does, except that the various computations are only performed across the relevant dimensions (see Algorithm 2 for details).

5.2 *Empirical Results*

To evaluate the subdimensional discovery algorithm, experiments were run with both synthetic data that contained planted motifs and with the exercise data described in Section 4.4. The goal of the evaluation was to characterize the empirical performance of the algorithm in terms of its run time and its accuracy in light of different kinds and different amounts of irrelevant data.

5.2.1 Computational Complexity on Synthetic Data

To facilitate empirical complexity analysis, experiments were run with synthetic time series data with planted motifs. The data and motifs were generated to have similar statistics and to avoid discontinuities at the motif boundaries (see Figure 41 as a short example). The benefit of using synthetic data is that we have complete control over the duration of the overall data set, the length and frequency of the motifs, and the scale and number of irrelevant dimensions.

For the case of a single planted motif, Figure 43 shows how the subdimensional algorithm scales as the length of the time series, T, increases (Figure 43a) and as the length of the motif, M, increases when T is held fixed (Figure 43b). The algorithm is able to accurately locate the planted motif in all cases, and, importantly, it correctly identifies the irrelevant dimension. From Figure 43a, we see that the time required to locate the planted motif scales linearly with the length of the time series. The dynamic time warp (DTW) distance has a much larger constant factor, however, due to the higher complexity of that distance measure compared to the sum of absolute

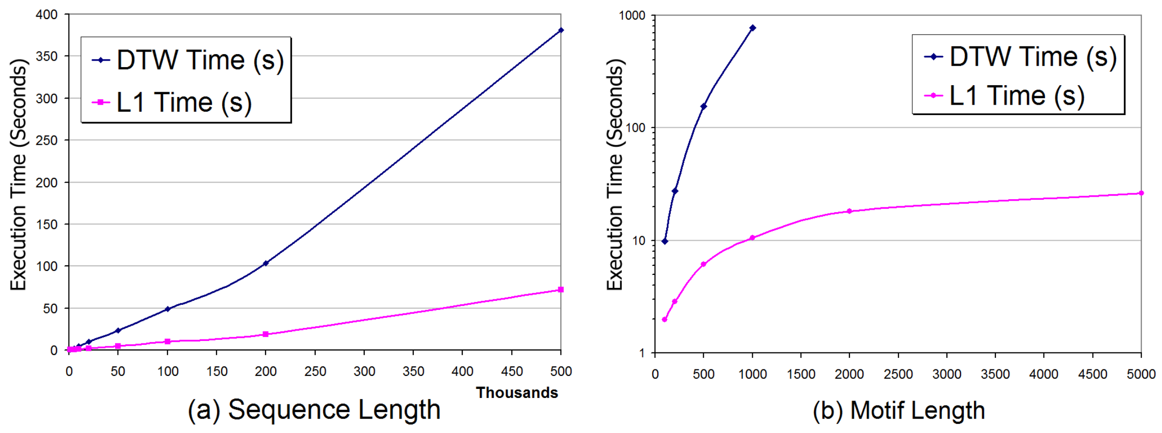

Figure 43: The basic subdimensional discovery algorithm scales linearly with both (a) the size of the time series data and with (b) the length of the motifs. However, the dynamic time warp distance measure scales quadratically with the length of the sequences being compared and so it leads to an $O(T^2)$ complexity as the motif length approaches the overall data length as shown in (b).

differences (*i.e.*, the L_1-norm for sequences).

The behavior is different as the motif length increases (Figure 43b). When the L_1 distance metric is used, the algorithm still scales linearly, but when the DTW distance measure is used, the computation time scales quadratically. This is not surprising since DTW is itself quadratic in M even when warping constraints are used (in all of the experiments, we used a 10% Sakoe-Chiba band). In typical cases of motif discovery, however, $M \ll T$, and so linear dependence on T still dominates the overall run time.

These experiments show the computation time in (wall-clock) seconds and were run on an Intel Core 2 E6400 CPU operating at 2.13GHz. They were not implemented using a multi-threaded algorithm, however, and thus did not take advantage of the dual-core CPU. The algorithm was written in Java and only minimal effort was made to optimize the implementation beyond using data structures that support the expected algorithmic time and space complexity (*e.g.*, sparse matrices and array-based sequence storage). As such, the graphs of execution time are intended to show relative performance and should not be taken as an accurate measure of the optimal

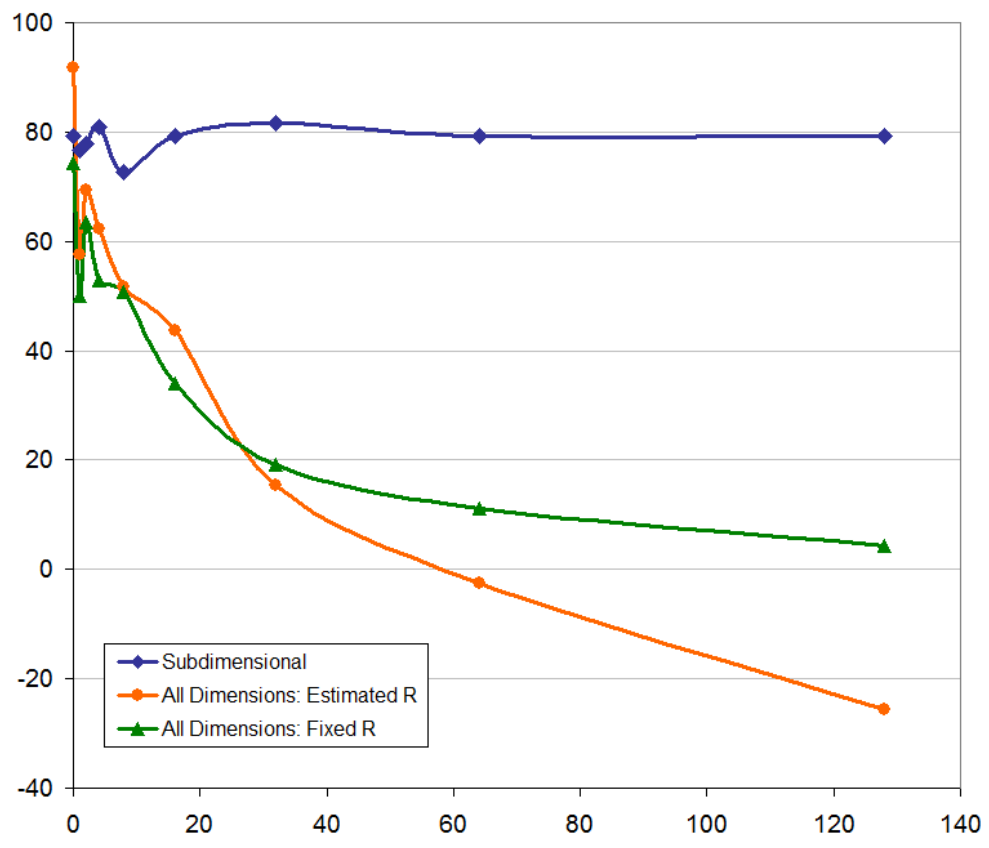

Figure 44: Graph showing event-based accuracy as the amount of noise increases in a single irrelevant dimension. The subdimensional algorithm can detect and ignore the noise, while the all-dimensional algorithms are disrupted.

run time of the algorithm.

5.2.2 Exercise Data with Irrelevant Features

The experiments in this section evaluate the ability of the subdimensional motif discovery algorithm to detect multivariate motifs in the exercise data set (described in Section 4.4). Two sets of experiments were run. In the first, the exercise data was modified by adding increasingly large amounts of noise to a single distracting noise dimension, while in the second, the data was augmented with many additional irrelevant dimensions, each with a moderate amount of noise.

Figure 44 shows the results of the first experiment in which a single dimension of

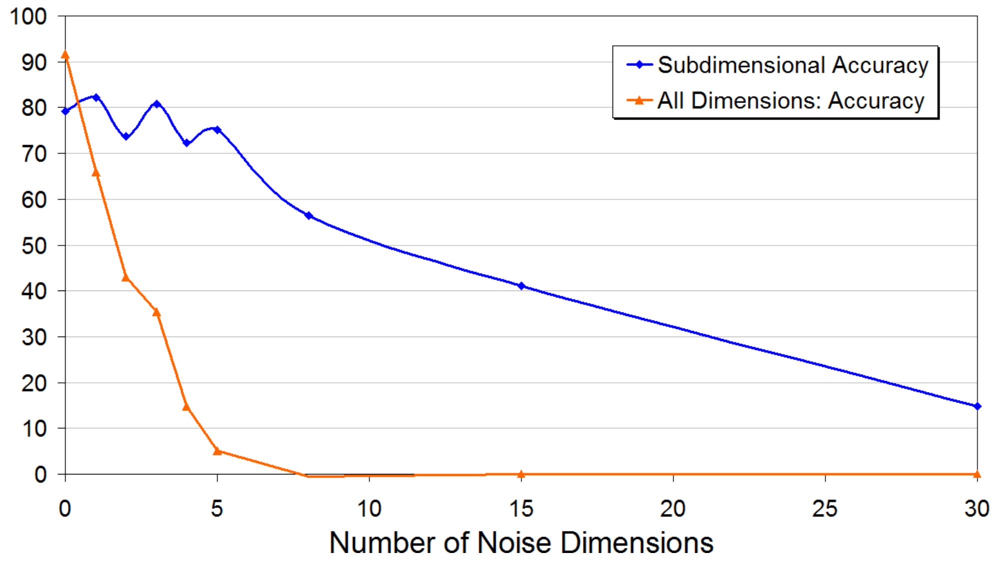

Figure 45: Graph showing event-based accuracy as the number of noisy, irrelevant dimensions increases. The subdimensional algorithm is negatively effected but not with the same severity as the all-dimensional methods.

noise was added to the six dimensional exercise data. With no additional noise dimension, the subdimensional algorithm achieve roughly 80% accuracy while the fixed radius all-dimensional algorithm performs slightly worse (74.2%) and the automatic radius estimation version performs substantially better (91.7%). As the scale of the noise in the extra dimension increases, however, the accuracy of both all-dimensional systems quickly falls, while accuracy of the subdimensional algorithms remains relatively unchanged. Note that this behavior is expected as the all-dimensional algorithms try to locate motifs that include the (increasingly overwhelming) noise dimension, while the subdimensional algorithm simply recognizes the irrelevance of that dimension and then only searches for motif occurrences that span the six remaining dimensions that contain valid sensor data.

In the second experiment, instead of increasing the scale of the noise, the number of irrelevant dimensions is increased (see Figure 47 for an example with three irrelevant dimension). The noise level in each additional dimension was set to correspond

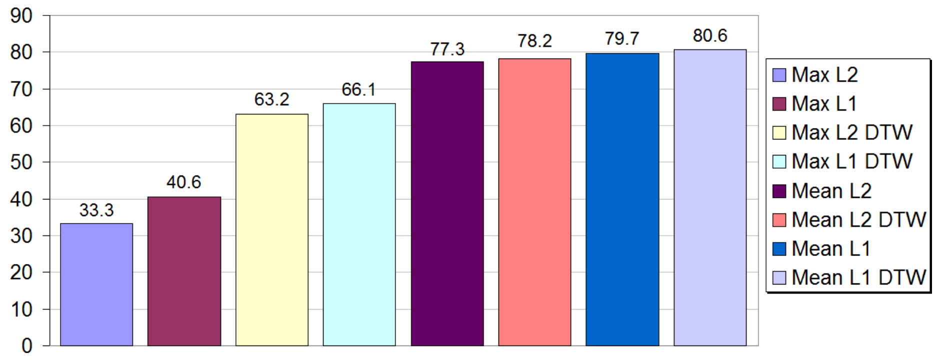

Figure 46: Event-based accuracy for the subdimensional algorithm applied to the exercise data set for eight different similarity measures.

to a value of four in Figure 44, which gives a signal amplitude similar to the real sensor data. The effect that additional noise dimensions have on accuracy is shown in Figure 45. The graph shows that the performance of both the all-dimensional and subdimensional algorithms decrease with extra noise dimensions, but the all-dimensional algorithm decreases much more rapidly. Ideally, the subdimensional algorithm would detect all of the additional noise dimensions as irrelevant and performance would stay level as it did in the previous experiment. Instead, the performance drops as extra dimensions are added. This behavior is likely caused by the algorithm discovering incidental patterns in the random dimensions which are counted as errors by the evaluation framework. The random patterns that arise here are not actually surprising because the probability of an unintentional pattern increases as the number of noise dimensions increases. The solution is to move to the discovery model represented in Figure 40d where motifs can temporally overlap if they use disjoint feature sets. Under this model, the incidental patterns may still be detected, but they would not disrupt the detection of the real patterns in the actual sensor data.

Finally, Figure 46 shows the accuracy achieved by the subdimensional discovery algorithm on the exercise data for different choices of the distance measure. The results show that calculating the mean distance between two sequences (*i.e.,* the average distance between the constituent frames) gives better performance than using

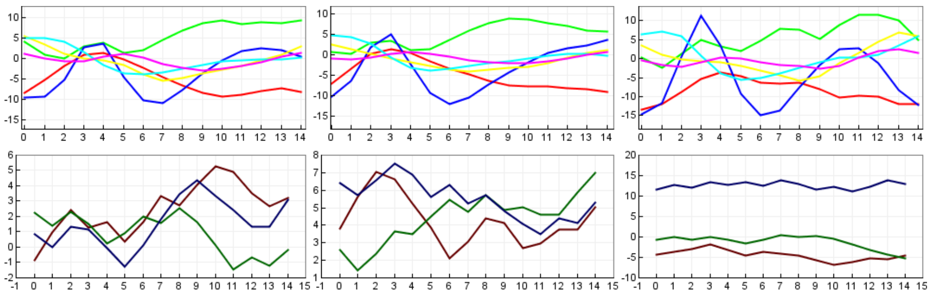

Figure 47: Three discovered occurrences of the `twist_curl` exercise. The top row shows the relevant dimensions while the bottom row shows the irrelevant (noise) dimensions.

the maximum distance across the sequences. For both choices (mean and max) the dynamic time warp distance provides higher accuracy than the temporally uniform L_p-norms. However, the difference is quite small when the mean distance is computed (78.2% vs. 77.3% for L_1 distance and 80.6% vs. 79.7% for the L_2-norm). In some domains, the use of the L_1-norm is preferred over the Euclidean distance because it provides additional robustness due to the more conservative penalty on outliers. For this problem, however, the performance is quite similar between the two distance measures with the L_1-norm achieving a slightly higher accuracy in both the direct and DTW cases. Because the DTW distance measure does not perform significantly better in terms of the detection rate, it is probably a worse choice than a direct, non-time warped metric, which is much more computationally efficient (see Figure 43). However, this result is highly domain-dependent and so it should be verified for other data sets and not considered to be a generally applicable preference. Also, the uniform scaling distance developed by Yankov *et al.* (described in Section 2.4) is an intermediate choice that is more efficient than the full DTW calculation but more sensitive than the direct norm [98].

Algorithm 2 Subdimensional Motif Discovery

Input: Time series data (S), subsequence length (w), word length (m), maximum number of random projection iterations (max_{rp}), threshold for dimension relevance ($thresh_{rel}$), and a distance measure ($D(\cdot, \cdot)$)

Output: Set of discovered motifs including occurrence locations and relevant dimensions

```
1. Collect all subsequences, $s_i$, of length $w$ from the time series $S$: $s_i = \langle S_i, .., S_{i+w-1} \rangle$ : $1 \leq i \leq |S| - w + 1$
2. Compute $\hat{p}(D) \approx p(D(s_{i,d}, s_{j,d}))$, an estimate of the distribution over the distance between all non-trivial matches for each dimension, $d$, by random sampling
3. Search for values of $\alpha$ (alphabet size) and $c$ (projection dimensionality) that lead to a sparse collision matrix
4. Compute the SAX word of length $m$ and alphabet size $\alpha$ for each dimension of each subsequence
5. Build the collision matrix using random projection over the SAX words; number of iterations $= min(\binom{m}{c}, max_{rp})$
6. Enumerate the motifs based on the collision matrix
    (a) Find the best collision matrix entry $(\dot{x}^1, \dot{x}^2)$
        i. Find the largest entry in the collision matrix and extract the set of all collisions with this value: $X = \{(x_1^1, x_1^2), (x_2^1, x_2^2), ..., (x_{|X|}^1, x_{|X|}^2)\}$
        ii. Compute the distance between the subsequences $x_j^1$ and $x_j^2$ in each collision, $1 \leq j \leq |X|$, and dimension, $d$: $dist_{j,d} = D(s_{x_j^1,d}, s_{x_j^2,d})$
        iii. Determine which dimensions are relevant:
             $rel(d) = \mathrm{I}(\int_{-\infty}^{dist_{j,d}} \hat{p}(D)) < thresh_{rel})$
        iv. Select the collision with smallest average distance per relevant dimension:
             $(x_j^1, x_j^2) : j = arg\,\underset{j}{min}(\frac{\sum_d dist_{j,d} \cdot rel(d)}{\sum_d rel(d)})$
    (b) Estimate the neighborhood radius, $R$, using only the relevant dimensions
    (c) Locate all other occurrences of this motif: $min(D(s_{\dot{x}^1}, s_i), D(s_{\dot{x}^2}, s_i)) \leq R$
    (d) Remove subsequences that would constitute trivial matches with the occurrences of this motif
```

CHAPTER VI

DISCUSSION

Chapter 3 described five all-dimensional discovery algorithms, while Chapter 4 presented the results of several experiments that compared the performance of these algorithms in terms of their ability to "discover" known patterns in both real and synthetic time series data. One of the primary research goals in developing the different algorithms (four of which were developed as part of this research, while one acts as a baseline and is essentially unchanged from Chiu *et al.*'s research [10]) was to explore different data representations used in the filtering phase of the general two-phase approach to efficient discovery.

To summarize, the different representations include (see Section 3.1 for details):

1. **Global Discretization** (`GlobalDisc`) – The real-valued, multivariate data is discretized such that each sample is replaced with a single symbol. A speed-up is achieved because comparing two symbols is fast and unequivocal. Furthermore, the string-processing community has developed highly efficient data structures for searching for recurring patterns in strings [24].

2. **Local Discretization** (`LocalDisc`, `AdaptLD`, `RP-GML`) – Each frame in the real-valued, multivariate time series data is contextually replaced with a symbol based on the surrounding data. When different local windows are considered, the frame may be replaced with different symbols. This representation precludes the use of string-processing methods, but researchers have developed random projection methods that can efficiently locate approximate matches despite the symbol multiplicity [10, 8]. The primary advantages of this approach over global discretization is that the actual discretization is much faster (primarily because

it is only based on the local data instead of the entire data set) and that the representation is more nuanced.

3. **No Discretization** (`Density`) – Although both discretization methods admit very fast search procedures, a discretization-free approach was developed for comparison. In theory, the discretization is necessarily an approximation to the original signal and so should introduce some error in the search. The goal of the discretization-free filtering algorithm is to avoid these errors while maintaining sub-quadratic asymptotic complexity as well as fast real-world performance. This goal was achieved by equating motifs with high-density regions in subsequence space and then using a fast, dual-tree based kNN algorithm to directly estimate the density around each observed subsequence.

The performance of the various algorithms is difficult to summarize due to the use of many user-specified parameters and the multiple performance criteria (*e.g.,* accuracy, efficiency, and parameter sensitivity). For instance, in terms of detection accuracy, the `Density` algorithm consistently performs at least as well as the other methods. However, its run time is also much slower than the local discretization algorithms, which might be critical for some applications.

Overall, the `GlobalDisc` algorithm performs worse than the other approaches in terms of both accuracy and run time. For both criteria, the primary culprit is the global discretization itself. In terms of computational complexity, the EM algorithm for learning the Gaussian mixture parameters scales linearly in the size of the data, however it may still require many passes through the entire data set if many iterations are needed for reasonable convergence. The implementation used in the experiments did not take advantage of any spatial partition data structures to accelerate the learning so that improvement may be one approach to ameliorating the problem. Other common discretization methods such as vector quantization via the k-means algorithm have similar shortcomings, though that does not mean that all global clustering and discretization algorithms are too slow. Nonetheless, the resulting accuracy is quite poor for complex, high-dimensional data sets likely due to the intrinsic difficulty in computing a global discretization that captures the important differences in the data while avoiding unwanted boundary effects.

The `AdaptLD` algorithm includes automatic neighborhood estimation for each discovered motif, whereas the original `LocalDisc` method uses a fixed, user-specified parameter. This change provides two important improvements: (1) a reduction in the number of parameters that must be manually specified and tuned, and (2) a more sensitive motif model. The `AdaptLD` algorithm does lead to more accurate results in some cases, but this improvement is not universal, even amongst the handful of domains explored here. The adaptive algorithm is strictly broader in its representational power since it could always estimate the neighborhood size of all discovered patterns to be equivalent to the fixed value provided to the `LocalDisc` version. In practice, however, the estimation algorithm is not robust enough to select values that lead to overall accuracy rates that are optimal or even consistently better than those achieved after tuning the `LocalDisc` algorithm. The adaptive algorithm may be useful to give a baseline value for the user, however. In some situations, the data analyst may have no idea what radius to specify to the `LocalDisc` algorithm. One option is to manually explore the results of different values, but this can lead to many iterations each of which includes a slow, manual inspection of the discovered motifs. The automatic estimation procedure may not always provide the best radius estimates, but it may be helpful to use it as a pre-processing step to provide the appropriate scale for the radius value to be used with the `LocalDisc` method.

Although the automatic neighborhood estimation procedure does not always outperform the fixed radius algorithm, it may still be preferred in some situations. For instance, the algorithm requires far less manual tuning and so it may give superior results when only a small amount of analysis time is available. Furthermore, all of the results presented for the `LocalDisc` algorithm in Chapter 4 are for the best accuracy rates achieved after exploring a wide range of parameter values within a supervised framework. The rationale is that the evaluation should reflect the performance of the algorithm itself and not the experimenter's ability to choose good parameter values. Although the tuning was automated in this case by using the (supervised) evaluation framework, in a real-world situation it is not unreasonable to assume that a user would have to try a range of parameter values and manually inspect the results from each run to select the best patterns. This use-case is facilitated by the extremely fast run time of the `LocalDisc` algorithm as it is by far the fastest method. However, to achieve the accuracy results presented here, it was often necessary to try dozens of parameter values, which led to a total run time, aggregated across all of the runs, that exceeded that of a single run of the `AdaptLD` algorithm. Thus, the adaptive algorithm often found a reasonable set of patterns much faster than the `LocalDisc` method, and it is certainly easier to use since the results are much less sensitive to the portion size parameter than to the radius size in `LocalDisc` (see Figures 31 and 32).

6.1 Feature Selection

A core aspect of all learning tasks is the selection of useful features. The importance of working with appropriate features is no less important for discovery tasks than it is in other areas of machine learning. The subdimensional discovery algorithm addresses one aspect of the problem, specifically, the case where there are some useful features but many others that should be ignored. Neither the subdimensional nor the all-dimensional algorithms developed as part of this research address the broader problem of extracting features from the raw data that are better suited for motif discovery. Instead, the algorithms rely on the user to pre-process the data and submit useful features as input.

Clearly, one can make the discovery process trivial by generating sufficiently discriminative features. In an attempt to avoid undermining the claim that it is the algorithm that is doing the learning rather than the system designer or user, the experiments presented in Chapter 4 used either raw sensor values or generic, community-accepted features. For instance, for the exercise data, the six dimensional feature vector was composed of the raw accelerometer and gyroscope readings. For the speech data, however, the raw waveform was transformed into multivariate samples composed of MFCCs. Although MFCCs are the result of intense research in the speech recognition community to generate features that aid the recognition task, they were deemed acceptable for these experience because they have become a standard preprocessing step for audio and were not adjusted or tailored to the discovery task in any way. In fact, the TIDIGITs data set was not processed by the author but was sent to a colleague who specializes in speech recognition with the request to extract "standard features for the field." Similarly, no specialized processing of the ASL video was performed. Instead, the features were taken from the original recognition experiments as calculated in 1995. The only additional calculation was the generic application of a normalization step to remove any bias for the naturally larger features (*e.g.,* blob mass) over those that are smaller (*e.g.,* orientation angle).

A natural question that arises is how one might go about extracting features that facilitate discovery. The task is quite difficult because in an unsupervised learning problem there are few constraints and little information on which to base the transformation. However, given a global objective function (discussed in more detail in Section 6.2), at least the system can score the overall process. This score may not directly provide information about how to manipulate the raw data to create useful features, but it can help rank the quality of the discovered patterns given different feature extraction methods. The ranking, then, provides the necessary information to select the best option and may also help guide the search for good features away from those transformations that hide or mask the embedded patterns.

6.2 Global Optimization

One design decision that affects each discovery algorithm but which has not yet been discussed is the global objective function that the algorithm seeks to optimize. Algorithms can be classified first by whether or not there is an overarching objective function and second by what form it takes and how it is optimized. Neither the `LocalDisc` nor the `AdaptLD` algorithms have a global objective function. Instead, these algorithms attempt to directly detect the recurring patterns and do not try to model the non-pattern data.

The `GlobalDisc` algorithm takes an intermediate position. The first phase includes a criterion based on minimum description length (MDL), which can be seen as a kind of objective function that directly tries to balance explanatory power with model complexity. Once motifs are detected in the discrete domain, however, the algorithm learns an HMM and then attempts to detect additional occurrences without modeling the entire data set.

Finally, the `Density` and `RP-GML` algorithms explicitly seek to maximize the total data likelihood. These algorithms learn generative models (HMMs) for each recurring pattern as well as for the non-pattern background frames. The assumption is that the closer the discovered motifs come to matching the "true" patterns in the underlying process, the better the motif models will fit the data. The shortcoming of this approach is that there is no model complexity penalty built in to the formulation. In a maximum likelihood framework that learns a mixture model, one can always learn trivial maxima by centering a model on a data point and letting the variance go to zero. Thus the model becomes a Dirac impulse that gives infinite probability to the data point over which it is centered. Typically, local maxima in the likelihood prevent algorithms from reaching such trivial parameter settings so long as reasonable initial parameter values are given. However, it is still the case that the likelihood can always be increased by adding more mixture components, at least until the number of components equals the number of data points. The `Density` and `RP-GML` algorithms avoid learning such large models through a heuristic stopping criterion, but a more principled approach based on a global model complexity penalty would be preferable.

6.3 Unsupervised to Supervised Learning Spectrum

The discovery algorithms presented in Chapter 3 take a fairly extreme position along the spectrum ranging from highly unsupervised algorithms at one end to tightly supervised learning methods at the other. While it is important to explore the feasibility and limitations of such strictly unsupervised algorithms, my research is not intended to make the argument that unsupervised methods are always the best choice or even that they are broadly applicable in real-world scenarios.

Instead, the intended argument is that unsupervised algorithms can find meaningful and useful recurring patterns in realistic data sets, even if very little domain knowledge is available. In a real world situation, reliable domain knowledge can and should be provided to the discovery algorithm, and the models and learning algorithms should be tailored to the known constraints of the data. However, a central tenet and motivation of this research is that "reliable domain knowledge" and "known constraints" may be much harder to acquire and far less common than often assumed. Unsupervised algorithms can help test, generate, and adapt the user's understanding of a domain, and so the algorithms can play an important role in the process of building a detection or analysis system, even if the final version is not learned solely through unsupervised discovery.

6.4 Variable-Length Patterns

Many motif discovery algorithms assume that all motifs, and all of the occurrences of each motif, have a fixed temporal duration [10, 18, 88, 75]. Several proposals have been made for ways of relaxing this assumption and allowing the detection of variable-length motifs:

- **Run-Length Encoding** – Although neither proposal in this category is truly run-length compression, both use the idea of detecting variable-length motifs by modifying a basic, fixed-length algorithm to be invariant to repetition of certain symbols in a discrete representation. For instance, Tanaka, Iwamoto, and Uehara developed an algorithm that assigns a different symbol to each unique SAX word and then combines runs of a single symbol [89]. When symbolic patterns are detected in the resulting string, each occurrence may have a different length due to the merging. Using a similar approach, the global discretization algorithm presented in Section 3.2 modifies the raw symbol string by replacing contiguous runs of a particular symbol with a single instance [57]. While the RLE approach to variable-length motif discovery does relax the restriction imposed by earlier methods, it is still far from the general case of allowing for motifs with arbitrary duration. This shortcoming is due to the fact that the pattern length in the compressed representation is still fixed.

- **Maximal Repeating Patterns** – The global discretization algorithm actually combines two methods to detect variable-length patterns. The RLE-like approach described above accounts for small, local temporal warping, while the overall search procedure accounts for patterns with fundamentally different durations. The search uses suffix trees to find *maximal repeating patterns*, which are repeating substrings in the data with maximal extent. Here, *maximal* means that if any of the occurrences were extended, they would then include different symbols and thus no longer match. To account for the fact that there may be N matches with length P but only M matches with length Q (where $M < N$ and $Q > P$), the algorithm uses an MDL framework for pattern selection. This approach is actually fully general with respect to locating variable-length motifs, but the requirement that the time series data be represented with a single string composed from a global alphabet restricts the representational power too much to be useful in some complex domains as discussed at the beginning of this chapter.

- **Temporally Flexible Models** – Several algorithms use probabilistic temporal models that inherently model occurrences with a range of durations [69, 57, 58]. These models do not fully address the variable-length pattern restriction, but at least they allow for variable-length occurrences. The models are used to detect occurrences by maximizing the data likelihood. This optimization can either be performed locally and independently (as in [69] and [57]) or within a competitive framework that also models the non-motif data (as in [58]). The primary drawback of the local, independent approach is that the basic models are biased toward shorter occurrences since every frame will typically reduce the overall likelihood. Accounting for this bias is possible, but doing so efficiently is quite difficult. The Viterbi algorithm, along with the variations used in the discovery literature, depends on a dynamic programming (DP) solution for efficiency. DP approaches require that the answers to subproblems are not affected by broader problems, but counteracting the bias typically requires incorporating the total match length, which breaks the optimal substructure property. One of the advantages of the competitive framework is to modify the overall computation in a way that reduces the bias. In that framework, all of the data must be modeled and so the motif model will match a frame if it assigns higher likelihood than any other motif model or the background model. Thus the need for a heuristic approach to counteract the short match bias is removed. The downside, however, is that a motif model may match a segment of the data that is not a real occurrence simply because the other motifs and the background model are even worse. This problem is very difficult to combat since it essentially requires a very accurate background model, which is rarely possible.

- **Temporally Flexible Distance Measures** – Yankov *et al.* developed an approach for detecting variable-length motif occurrences that relates to the use of temporally flexible models [98]. Instead of using a probabilistic temporal model, however, they adopted a distance measure that explicitly searches for the best match over a small range of durations by uniformly warping the query instance [40]. Note that this is different than a typical DTW formulation, which allows local, non-linear warping but generally does not allow the query to match to a shorter or longer version of the given target sequence (though see Park and Glass' segmental DTW [71]). The primary shortcoming of the uniform scaling approach is the same as that for the model-based warping method, specifically that it finds variable-length occurrences but does not address the restriction of fixed-length patterns.

- **Temporal Pattern Extension** – The final alternative that has been proposed in the discovery literature is the temporal extension of already-discovered patterns [69, 71, 57]. In this approach, a fixed-length discovery algorithm is used to locate base patterns, and then each pattern is analyzed to see if it should be temporally extended, thereby adding adjacent frames to the motif. Oates uses a statistical test based on the probability of the already-detected occurrences given the current, fixed-length model along with a user-specified confidence level to determine when a pattern should be extended [69]. The global discretization algorithm analyzes the real-valued data and compares the variance of the aligned frames within each pattern to that of the adjacent frames to iteratively extend each motif (see Section 3.2.2 for details). Finally, Park *et al.* use a different method where the region around each seed pair is analyzed to find the length-constrained minimum average (LCMA) within the segmental DTW matrix [71]. The exact occurrence location and duration is then adjusted to correspond to the contiguous region that has the lowest average deviation. This broader analysis is possible due to the development of a $O(N \, log \, L)$ algorithm (where N is the total segment length and L is the minimum length of the minimum average region) for computing the LCMA segment [52].

Although this approach is one of the most promising for locating variable-length motifs, there are two major problems. First, as argued by Yankov *et al.*, such methods require that the base algorithm search for patterns that have the smallest expected duration [98]. This search may be highly inefficient and lead to many false matches if the final patterns are often longer than the minimum duration. Second, all of the proposed temporal extension algorithms require a user-specified parameter. Although the various authors demonstrate improved results on some data sets, our experiments with the data sets described in Chapter 4 were mixed. It was often difficult to specify parameter values that improved accuracy and closer inspection showed that within a single data set, a particular parameter value might improve the accuracy of some patterns while reducing performance on others. Thus, this approach warrants additional research, but the current proposals appear to be too sensitive to be broadly applicable for variable-length motif discovery.

6.5 Run-Time Performance

All of the algorithms discussed in Chapter 3 meet the overall run-time goal of subquadratic complexity in the length of the input data. The absolute run-time and constant coefficients, however, vary dramatically between the different methods. Broadly, each algorithm embodies a different trade-off between run-time and sophistication, where simpler methods run much more quickly, but often at the expense of modeling

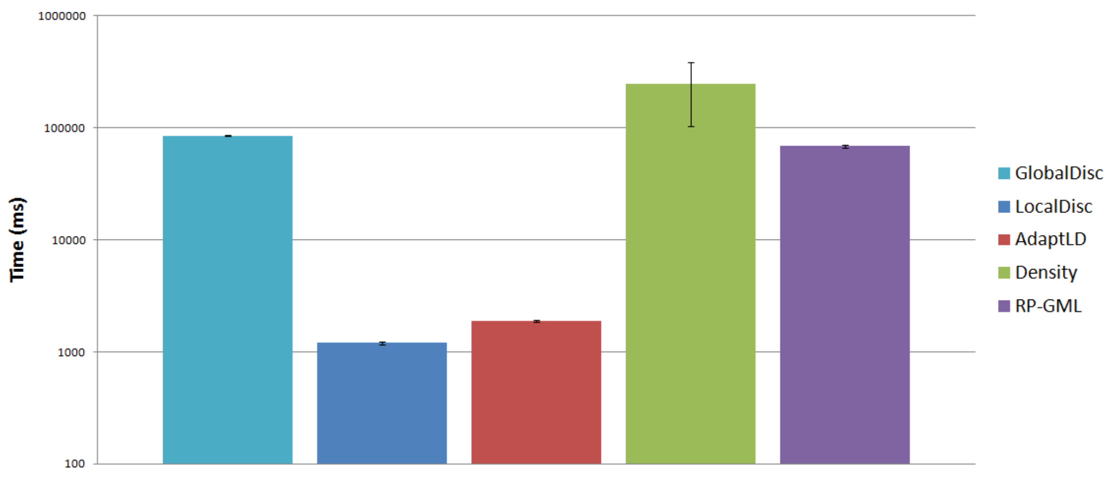

Figure 48: Run-time of a single run of each of the five algorithms on the exercise data set. Note the logarithmic time scale.

power, generality, or parameter complexity.

Figure 48 summarizes the run-time performance of the five discovery algorithms on the exercise data set. The values represent the time required to complete the discovery task for a single run using optimized parameters. In other words, the values do not include the time required to search for the best parameter values, which can be significant in some cases. To generate the graphs, each algorithm was executed three times, and the graph shows the mean run-time along with a one standard deviation confidence interval. The experiment was run on the same machine used to generate timing results for the subdimensional algorithm, an Intel Core 2 E6400 CPU operating at 2.13GHz with an implementation in Java.

The graph shows a clear ranking of the algorithms in terms of their expected run-time: `LocalDisc`, `AdaptLD`, `RP-GML`, `GlobalDisc`, and then `Density`. A logarithmic scale was required for the temporal axis to account for the wide disparity between the run-time of the different methods. For instance, `LocalDisc` requires around 1,200ms to process the exercise data, while `AdaptLD` uses approximately 1,900ms. This difference is relatively large when viewed as an increase of over 50%, but we can see that both algorithms are more than 36 times faster than the next most efficient method.

The RP-GML and GlobalDisc algorithms have similar run times (approximately 69 seconds and 85 seconds, respectively, a difference of around 23%). The RP-GML method yields much higher accuracy rates, however, and so it is the preferred method between the two. Across all of the domains explored in Chapter 4, the Density algorithm consistently yields the highest accuracy rates. It is now clear that this performance comes at a cost, as this method requires over 245 seconds to analyze the exercise data, which is roughly 3.5 times slower than the RP-GML method and over 200 times slower than the LocalDisc algorithm.

The use of manually optimized parameters to perform the run-time test skews the results. In real world scenarios, the manual tuning can be a very time consuming process, especially if verification of the discovered patterns requires extensive inspection of the data or additional experimental validation. Even for the exercise data, where manually labeled patterns are used to assess detection performance, the LocalDisc method required over 30 trials to find the best parameters. Thus, even though the AdaptLD method is over 50% slower than the LocalDisc method, the fact that it required far fewer trials to find near-optimal parameter values means that in practice it was on par or even slightly faster than the LocalDisc algorithm. The Density and RP-GML methods tend to be even more stable, but the corresponding reduction in the number of trials was not sufficient to compensate for their dramatically slower run-time.

The use of manually optimized parameters affects the run-time characteristics in another way as well. Because the algorithms assume that all patterns are temporally disjoint, each time a motif is discovered, its occurrences can be “removed” from the data set. This process reduces the amount of data that must be analyzed in subsequent iterations. In general, this effect is consistent across the different methods, but poor parameter choices can have a dramatic effect. For instance, for methods that use the hypersphere motif model, a very large neighborhood size will lead to many false positives (insertion errors), which means that much more data will be removed in each iteration than is actually warranted. In extreme cases, a large neighborhood size will lead to far fewer motifs, which can speed up the overall discovery process. Similarly, if the neighborhood size is too small, there will be a large number of false negatives (deletion errors), which can lead to many additional discovery iterations and thus a much longer overall run-time.

6.6 Pattern Hallucination

One can find patterns in any data set. Without some external, objective criterion, there is no guarantee that discovered patterns are "real" in the sense that they correspond to some underlying process or aberration with desirable properties. In Section 7.5, I will discuss the idea of *motif ranking* that attempts to capture the relative amount of support a particular pattern has within a data set. Using such a function, a discovery system can order the detected patterns according to how surprising they are or how confident the system is that they did not arise randomly.

One common approach to differentiating between "real" and spurious patterns is to assume a background model for the data. Then, one can compare the probability of a particular pattern and its associated occurrences under that model. There may be some threshold that specifies a lower-bound on the probability of a real pattern under the background model, or one can compare the background probability to that of a learned model and reject those patterns with a relatively poor ratio. For instance, Denton uses a random walk background model to infer a threshold in order to detect interesting recurrences [18]. Similarly, Chiu *et al.* compute the probability of a particular symbol sequence using a uniform model and reject potential matches with a similarity that fails to rise above the predicted level [10].

The competitive framework described in Section 3.4.2 incorporates a model-based method for rejecting spurious patterns that is related to those of Denton and Chiu *et al.* Before any motif models are added to the mixture, the algorithm learns the parameters of a background model. In the experiments of Chapter 4, a three-state, fully-connected HMM with Gaussian observation distributions was used. This model provides a baseline probability for each frame in the time series data, and, because of the overall structure of the competitive learning framework, motif models are only included in the final output if they lead to a higher probability than the background model for some segments.

The generic mixture model used as the baseline explanation in the experiments was sufficient to prevent the motif models from generating too many insertion errors. However, a more precise model, or even a set of background models, may be required in other domains. The framework supports such inclusion, although the generalized Viterbi algorithm currently used to fit the models may need to be adapted.

In order to further investigate the use of a tailored background model to help avoid false pattern detections, a series of experiments with synthetic data were performed. The experiments are divided into three categories. In the first category, random walk data was generated with no motifs. In the second category, a single recurrence was planted in the random walk data, while the third category is simply the exercise data. In each case, the `RP-GML` algorithm is used, and the probability of potential motifs are compared under the learned motif model, the random walk data model, and the generic three-state mixture model used in the other experiments. The expectation is that real patterns should have much higher probability under the learned model compared to either of the two background models.

In order to generate random walk data with a model that works with the existing discovery system, a HMM with a hand-crafted topology was built (see Figure 49). The model always starts in the state to the far left, which has a Gaussian observation

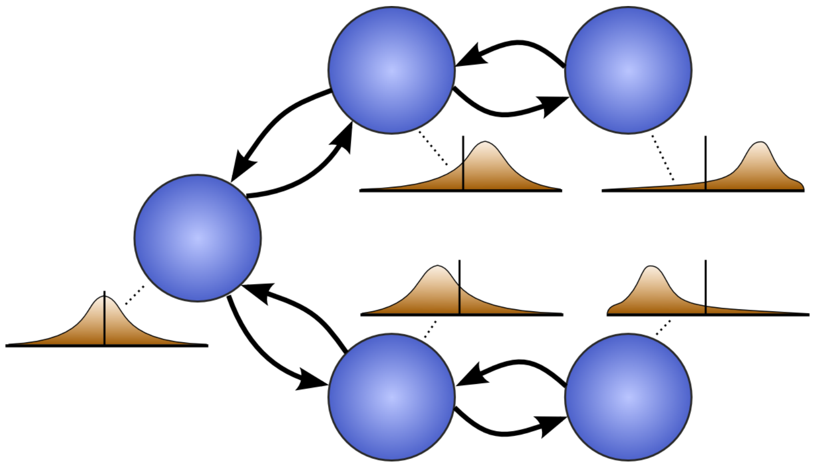

Figure 49: The HMM topology used to generate the random walk data (self-transitions are not shown)

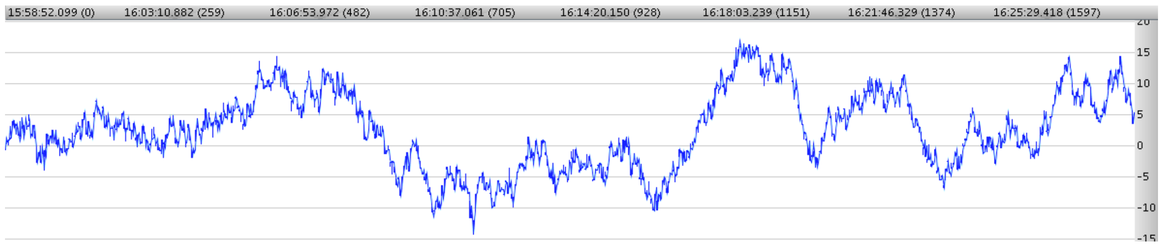

Figure 50: Graph showing a segment from the random walk data generated to test spurious pattern suppression.

distribution centered at zero. It can then transition to either "arm," where each state has a Gaussian with its mean shifted in either the positive (upper arm) or negative (lower arm) direction. For the experiments, the HMM was built with 33 states, which includes the zero state in the center and 16 states for each arm. The state transitions are also biased so that it is slightly more likely to move toward the zero state than farther away in the positive or negative direction. This bias helps the model from being "stuck" at the end of the arm, which corresponds to an artificial bound on the range of the generated data. Overall, the model is similar to a first-order autoregressive process with a parameter slightly less than one (see Figure 50 for an example).

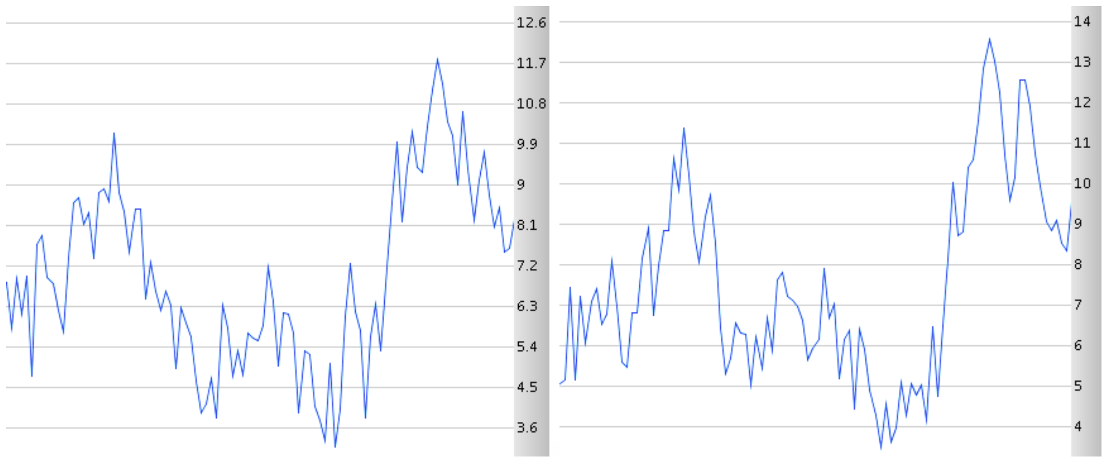

Figure 51: Graph showing a pair of similar subsequences detected by the discovery system in the random walk data. Although the sequences are visually similar, the random walk model still provides a higher probability than the learned model, thus allowing the proposed pattern to be rejected as spurious.

The first experiment involves data sampled from the random walk HMM. When the generating model is used as the background model, the discovery algorithm generates several candidate motifs, but none are actually matched by the motif models. In other words, the algorithm initializes a HMM for each potential motif, but in each case, the background model gives a higher probability to the relevant subsequences and so no occurrences are detected. On the other hand, when the standard three-state background model is used, the algorithm does find some motifs (see Figure 51). In each case, there is a clear similarity between the occurrences detected, and the learned model does give higher probability than the three-state background model. When a post-analysis is applied to the motif occurrences, however, it is clear that the learned models are no better (and certainly not significantly better) than the random walk model.

Three trials were run with pure random walk data. The only difference between them is the amount of data generated. The first trial included 10,000 frames, the second had 20,000, while the third had 100,000. Theoretically, it is more likely for a spurious pattern to arise in a larger data set, and so we wanted to avoid an artificial null result that might have resulted if only small data sets were analyzed. We found no significant difference in the results across the three data sets, however. In each case, the random walk model yielded higher probabilities than the three-state background model and the learned motif models.

In the second experiment, the same random walk HMM was used to generate synthetic data and a single pattern with two occurrences was planted in the data. This data is similar to that used to empirically evaluate the computational complexity of the subdimensional algorithm as described in Section 5.2.1. The expected result is the same as in that case, namely that the algorithm will find the planted motif and no others. Importantly, the probability of the planted motif given the learned model should be higher than either the generic background model or the random walk model.

The result of the experiment matches the expectation. When either the three-state or random walk model was used as the background model, the discovery algorithm detected the planted motif. Furthermore, in both cases, the learned model yielded a significantly higher probability than the others. Specifically, when the three-state model was used as the background model, the two planted occurrences had a log-likelihood of -25.5 and -32.3 under the learned model but only -126.2 and -125.1 under the random walk model. When the random walk model was used as the background model, the algorithm found slightly different boundaries, which gave a log-likelihood of -8.2 and -9.0 for the learned model but only -120.12 and -120.07 for the random walk model. In both cases, the log-likelihood under the generic, three-state model was less than -300.

The results for the second potential motif detected by the discovery algorithm is similar to the previous experiment where there were no actual patterns. For these occurrences, the random walk model model gave log-likelihood values of -158.5 and -155.0, while the learned model gave -166.1 and -162.3. Thus, this pattern, along with the other potential detections that have similar probabilities, can be rejected. Note that the three-state model again yielded a much smaller likelihood.

In the final experiment, the random walk model was applied to the results of the hybrid discovery algorithm when applied to the exercise data set. In this case, all of the discovered motifs had a higher probability under the learned model than under the random walk or three-state models. In fact, none of the likelihoods were even close (the most similar had a log-likelihood around -122 for the learned model compared to -190 for the random walk model). This implies that either the random walk model is a poor choice for a baseline or that none of the patterns are spurious.

Further, manual investigation revealed that the detected patterns beyond the six expected exercises and a seventh rest state were, in fact, interpretable. For instance, after detecting the actual exercises, the algorithm found four motifs that always occurred just before one of the exercises. As such, they can be interpreted as a kind of prefix or preparatory movement that serves as a transition from the rest state to the start of a new exercise. The subsequent motifs are somewhat harder to interpret, but can be summarized as a post-exercise transition (the pattern only occurred after the tricep and shoulder press exercises) and then a completion pattern, which only occurred after a shoulder press at the very end of a sequence.

6.7 Overlapping Patterns

A basic assumption made by all of the discovery algorithms developed in this work is that the patterns in a data set are temporally disjoint. This assumption is incorporated into both the traditional, "all-dimensional" algorithms described in Chapter 3 as well as the subdimensional algorithm presented in Chapter 5. The assumption, however, is not required for efficient pattern discovery. In fact, the original formulation of the `LocalDisc` algorithm by Chiu *et al.* allowed for partially overlapping patterns [10]. The problem of trivial matches was avoided by using a more precise testing mechanism based on the hypersphere motif model as described in Section 3.3.1.

Several questions naturally arise due to the assumption of temporally disjoint patterns. First, what happens when an algorithm that makes this assumption is applied to data in which the real patterns do, in fact, overlap? Second, how frequently should we expect the patterns to overlap in real data sets, and third, how can the discovery algorithms presented here be enhanced to support the proper detection of overlapping patterns?

The answer to the first question depends on the amount of overlap and on the frequency of each overlapping pair of patterns. The behavior will depend on the relative frequency of the combination. Consider two patterns, A and B, such that a long prefix of B overlaps a long suffix of A. There are four main cases based on how frequently each component occurs independently:

1. Neither A nor B frequently occur outside of the combination AB
2. A, but not B, frequently occurs outside of AB
3. B, but not A, frequently occurs outside of AB
4. Both A and B frequently occur independently and only sometimes occur together as the combination AB.

In the first case, the algorithm will simply detect the combination AB as a single pattern. There is little supporting evidence in the data from which the algorithm can infer that the combination should be decomposed. Presumably, the fact that there are actually two distinct patterns is only known due to external domain knowledge or other semantic information. Since the discovery algorithm does not have access to such information, it will have no way of knowing that the pattern AB should be divided.

The second and third cases are symmetric and so only the second case will be discussed explicitly. Because A frequently occurs outside of the combination, the algorithm will likely detect it as an independent pattern. It will then detect B as just the suffix of AB if the overlap is small or may identify the two patterns as A and AB if the overlap is large. In the latter case, the large amount of overlap causes the "tail" of AB to have too short a duration to be independently discovered.

The final case is similar to the previous situation except that both A and B will be independently discovered. Then, AB may still be identified as a separate pattern if the overlap is high. Otherwise, the discovery algorithm should be able to detect the separate A and B patterns in the combined AB occurrences. The small overlapping region will be included in one or the other pattern depending on the local noise and the variance learned by the motif models.

With regard to how common overlapping patterns are in real data sets, a definitive answer is difficult to provide since the question is inherently empirical. A common case for overlapping patterns is when there is co-articulation (or the appropriate generalization for data sets outside of the speech domain) between consecutive patterns. For instance, an English speaker may blur the boundaries of two adjacent words as in the transition between "want" and "to" in the phrase, "I want to go to the store." Similarly, a signer may blur the end of one gesture with the beginning of the next sign. Co-articulation effects that only have a short duration are probably quite common, but this case is also the least disruptive to algorithms that make the disjoint assumption.

Finally, adapting the discovery algorithms to remove the disjoint pattern assumption is relatively easy given a procedure to detect trivial matches. The key limitation in the current implementation is that after a motif is detected, the algorithm removes all of the data frames "explained" by the motif occurrences from the list of "unexplained" frames. This change precludes further processing of those frames, which ensures that no future motif will overlap the one that was just detected.

If the algorithm does not remove the relevant frames from the analysis list, then it can detect overlapping motifs in future iterations, but it will also likely detect false or duplicate patterns due to trivial matches. The solution is to store more detailed information about each frame and then to remove a smaller number of frames around each occurrence. Instead of keeping track of which frames have been explained by a motif and which have not, the discovery system will need to store whether or not a particular frame is a valid starting location for motif occurrences. Clearly, the first frame of a detected occurrence is no longer valid as it would lead to a duplicate detection. In addition, all frames surrounding it should be marked as invalid if the corresponding occurrences would constitute trivial matches. By only removing the offending frames from the list of potential starting points for future pattern occurrences, the system can avoid trivial matches while still allowing overlap.

For those algorithms that do not use the hypersphere motif model, a new definition of a trivial match is needed. By analogy, a viable definition based on probabilistic temporal models could be developed as follows:

> Given an occurrence O_i starting at frame i with motif model λ_M and background model λ_B, a subsequence O_j starting at frame j is a trivial match of O_i iff:

$$\neg\exists k : ((i < k < j) \vee (j < k < i)) \wedge p(O_k|\lambda_B) > p(O_k|\lambda_M)$$

In other words, if there is a model for one occurrence, another subsequence is a trivial match if there is no subsequence between the two that is more probable under the background model than under the relevant motif model.

A final complication arises from the Viterbi-based model fitting method used by the `Density` and `RP-GML` methods. An efficient implementation of the algorithm is based on a dynamic programming formulation, which depends on the restriction that each frame of data is explained by a single model. It is not clear how to relax this restriction while maintaining the desired run-time characteristics. Adapting this aspect of the discovery algorithm to allow overlapping patterns is an open research problem, but one can still use a pure search-based approach as in the `LocalDisc` and `AdaptLD` algorithms.

CHAPTER VII

FUTURE WORK AND CONCLUSIONS

The development of unsupervised recurring pattern detection algorithms for multivariate, real-valued signals is a relatively unexplored research area and so it is not surprising that there are many open issues that warrant further investigation. The most obvious direction is to try to improve the algorithms presented here in terms of their basic goals, specifically to reduce the overall run time, to improve detection accuracy, and to remove or automatically estimate the user-specified parameters required by the various methods. The remainder of this chapter outlines several other research directions that can help improve the basic algorithms and broaden their applicability.

7.1 Online Discovery

All of the methods presented here are batch algorithms. In other words, the analysis takes place after all of the data has been collected rather than in an online fashion either as the data is collected or in a one-pass, sequential manner. While this approach makes sense in many situations, it can cause problems for massive data sets that do not fit into main memory or for streaming data that is never fully collected. The memory constraint is an issue because retrieving data from a hard drive or other inexpensive mass storage devices is orders of magnitudes slower than accessing data in main memory. The need to continually retrieve different parts of the data from secondary storage means that a low complexity algorithm may actually run more slowly than a simpler algorithm with a worse asymptotic bound. This effect is especially pronounced if the simpler algorithm accesses the data in a manner that is more efficient for the underlying medium. For instance, data can be retrieved more quickly from a traditional hard drive when it is accessed sequentially rather than in a random fashion that requires the drive head to seek more often.

Catalano *et al.* developed an online recurring pattern discovery algorithm for real-valued signals [9]. To my knowledge it is the only online algorithm of its kind. The approach is very efficient but relies on a limited memory both for efficiency reasons and to allow online processing (see Section 2.4 for a brief summary of their algorithm). Improvements could be made by improving the temporal range over which the algorithm can detect repeated patterns and by incorporating online feature selection to allow subdimensional discovery.

7.2 Coarse-to-Fine Processing

It should be possible to realize efficiency gains in the discovery algorithm with only a very small accuracy penalty by utilizing a coarse-to-fine processing strategy. Such methods are often used in image processing and other signal processing areas to accelerate search algorithms, typically via early rejection schemes. Note that in the context of multivariate pattern discovery, the coarse-to-fine approach can be applied in two ways. First, the temporal sampling rate can be downsampled to produce a "pyramid" of signals at successively lower sampling frequencies. Then, the similarity between two segments can be estimated from the signal at a lower sampling rate, potentially detecting poor matches without analyzing the full data set.

The second approach is to embed the multivariate samples into a lower-dimensional space. For instance, one can compute the principal components of the signal, treating each sample as a separate data point and then reconstruct a low-dimensional approximation from the top few components. Similarly, other dimensionality reduction methods such as Isomap, locally-linear embedding, manifold sculpting, random projection, or any of the other variations proposed in the literature could be used. The goal is the same as for temporal downsampling: if two segments are sufficiently different in the low-dimensional space then they do not need to be compared in the original, high-dimensional space. Although this approach only gives a constant speed-up, since the maximal reduction is from the original D dimension down to one, it may be possible to combine this method with other speed-up techniques to give a non-negligible boost to the real-world performance.

7.3 Higher-level Learning & Feedback

The typical output of motif discovery algorithms is a list of motifs including a model for each pattern and a list of occurrences in the given time series data. In a successful application of pattern discovery for structured activity data, the motifs will correspond to the primitive actions that make up the activity. Thus, one should be able to analyze the temporal relationships between the motifs to uncover the higher-level structure of the underlying activity. For instance, for speech data, the motifs will likely correspond to the words in the language, while the temporal relationships provide information about the grammar.

The additional information contained in the higher-level structure can be very helpful for forming a complete understanding of the data, and it is especially useful if the learning system intends to use the discovered primitives as the basis for future planning or inference. The temporal relationships can indicate causal relationships or at least significant temporal correlations, and knowledge of these correlations can extend the horizon over which the system can accurately reason and predict.

If the discovery system is able to infer reliable temporal relationships between different motifs, it may also be able to use that information to improve the base pattern detection. For example, if the system observes that `Motif 9` typically follows `Motif 2` and it detects an unpaired occurrence of `Motif 2`, it can allocate more resources to the search or can relax the detection threshold for a subsequent `Motif 9`. Of course more complex relationships are also possible, and the entire process can be wrapped in an iterative loop that alternates between detecting motifs using the current estimate of the higher-level structure and then re-estimating the structure from the detected motifs.

Another benefit of inferring higher-level structure as part of the discovery process is that it can alleviate the confusion of multi-part motifs. For example, consider the `flat_curl` and `twist_curl` movements from the exercise data set. Both of these exercises start with the same motion but then diverge mid-way through due to the rotation in the forearm in the `twist_curl` exercise. A discovery system could reasonably detect the two parts separately, especially if the system is designed to allow large variations in the motif duration. The first half of the exercises has much better support since it occurs roughly twice as often as either individual exercise. The same situation occurs in speech data with multi-syllabic words. A discovery system could detect common word parts instead of individual words. Similarly, it could err on the side of extension and detect common phrases. If the system discovers a hierarchy of motifs and temporal relationships between them, then this confusion is reduced since the hierarchy encodes the information that pattern subparts have a strong correspondence.

7.4 Automatic Timescale Estimation

The user-specified parameter that is most difficult to specify yet is required by all of the motif discovery algorithms is the one that controls temporal scale. The various algorithms differ with regard to how much deviation is allowed in the duration of a pattern before the algorithm becomes unable to detect it, but almost all of them rely on the user-specified time scale parameter to some extent. The sole exception is the global discretization algorithm (see Section 3.2) which avoids the time scale parameter by using a suffix tree that can efficiently find recurring patterns of any length. That approach is plagued by other shortcomings, however, and so it is not a general solution.

One approach to automatically estimating the best time scale for analysis and to detect patterns at multiple time scales is to use a temporal multiresolution analysis as discussed in the previous section. At each temporal resolution, the algorithm can search for recurring patterns and notify the user. A more powerful unsupervised algorithm would globally rank the patterns (see the next section for details on motif ranking) in an effort to avoid overwhelming the user with redundant, overlapping patterns.

7.5 Motif Ranking

Motif ranking is the process of calculating the relative support of two motifs or otherwise assessing their "interestingness." When pattern discovery is tied to a particular domain, as in the discovery of motifs in nucleotide sequences in bioinformatics, this problem can be addressed by applying domain knowledge. For more general data mining, such domain knowledge is not available and so a different solution is needed. Researchers have adopted various approaches that implicitly sort the motifs without explicitly investigating the effects of the implicit ranking function. For instance, the `LocalDisc` method addresses the problem by making simplifying assumptions about the form of a motif and greedily detecting motifs according to their similarity as measured by the collision matrix, which does not directly depend on each pattern's support.

In some cases, the relative ranking of two motifs is very clear. For instance, if `Motif 1` has 35 occurrences with an average similarity of 1.4 while `Motif 2` has 47 occurrence with an average similarity of 1.1, `Motif 2` clearly has higher support and thus should be ranked above `Motif 1`. Note that the concept of subsequence density as a single measure of support captures exactly this relationship. The situation becomes more complicated when the base measures diverge, for instance if one motif has more occurrences but lower similarity than another. The inclusion of variable-length motifs also complicates motif ranking. Although it is clear that if two motifs have otherwise similar characteristics, the one with longer duration should be preferred, the best interaction between duration, similarity, and number of occurrences for motif ranking is unknown.

It is unlikely that a solution to the motif ranking problems exists that is optimal in all cases, and little research has investigated methods customized for specific domains. One approach is to develop a formula that aligns closely with human judgement to help provide a general purpose baseline. In the field of computer vision, researchers have developed mathematical models of human surprise in an effort to understand and simulate the human visual attention mechanism [32]. An interesting research direction is to investigate ways of adapting this work to estimate the surprise associated with a particular motif given the data set in which it is embedded. This framework may improve our understanding of the relationship between existing methods and aid in the development of new algorithms designed around explicit ranking functions.

7.6 Interactive Discovery

As discussed in Section 6.3, there exists a spectrum ranging from unsupervised algorithms with minimal bias to supervised algorithms that take advantage of extensive domain knowledge and training data. One intermediate approach that has received relatively little attention is interactive discovery. In such a system, the system begins the discovery process and can solicit feedback and suggestions from the user when needed. The user can also view intermediate results as they are computed and can correct mistakes or guide the system toward other solutions.

The goal of an interactive discovery system is to utilize the strengths and avoid the weaknesses of both the automated algorithm and the human user. Typically, the user has some idea of the kinds of patterns that may arise or can at least identify

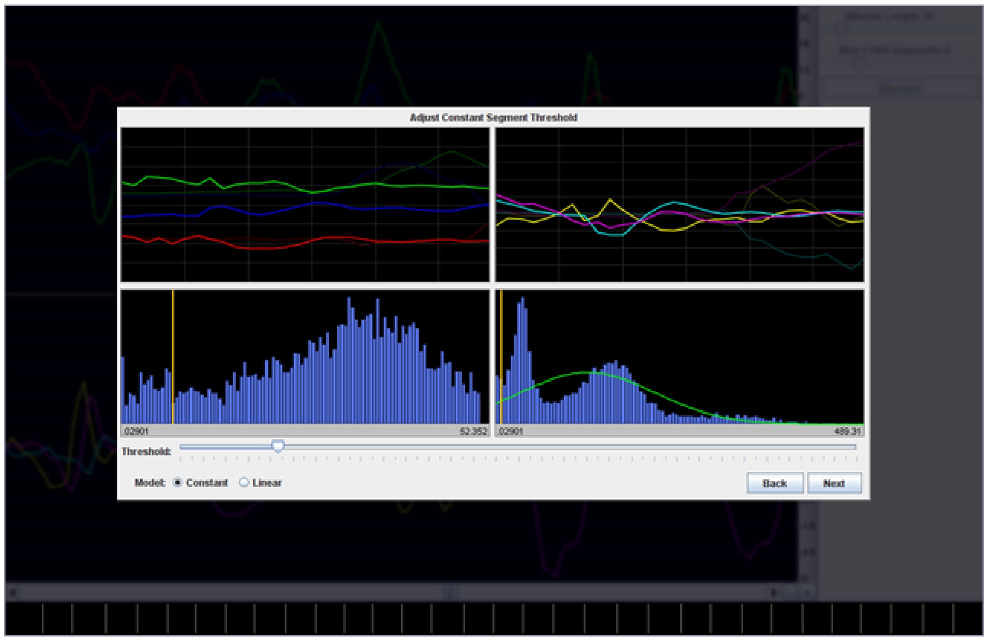

Figure 52: Interface for manually adjusting the threshold for removing “silence” in the time series data. The interface supports both a constant and locally linear silence model.

corrections, uninteresting repetitions, or subtle patterns that an automated algorithm may have difficulty detecting. Conversely, computational systems lack extensive experience in the world and the common sense that naturally arises in humans due to that experience. They do not, however, suffer from fatigue, boredom, or technical imprecision and thus can be much more efficient at analyzing massive data sets and carrying out straightforward calculations.

The idea of interactive learning has been previously explored for supervised learning tasks [95, 21]. For instance, Fails and Olsen developed an interactive system for computer vision that learns to segment hands over cluttered backgrounds by detecting skin-colored pixels [21]. The system starts with a very small training set and then iterates between learning a classifier and asking the user for additional training examples. The key insight is that the user is not required to provide extensive training data nor carefully selected examples. Instead, the training data and thus the classifier is continually refined in each iteration, and the user is aided because the system will

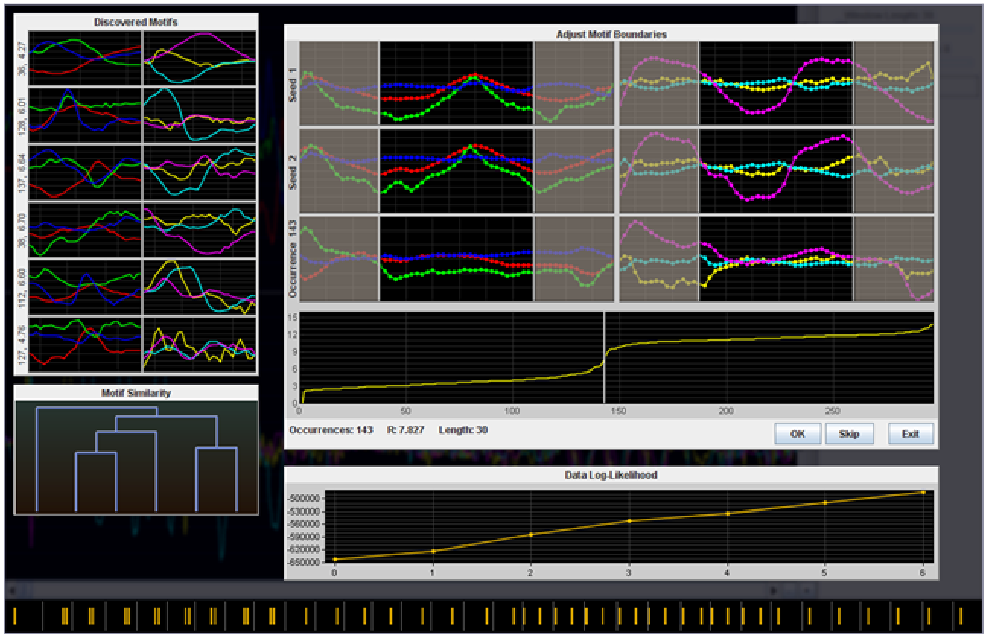

Figure 53: Graphical interface for interactive motif discovery. This display shows sample occurrences, the occurrence distance graph and associated threshold, previously discovered patterns, and a dendrogram summarizing the relative similarity between the patterns.

classify all of the pixels and present a visual display of the results. The user can then focus on just the mistakes made by the classifier and make a quick annotation that drives the next round of learning.

Within the data mining and visual analytics communities, many systems have been developed to allow large-scale data visualization, interactive queries, and summarization. I am not aware of any interactive systems for the kind of temporal pattern discovery explored in this research, however. I have implemented a preliminary interface for exploring the different approaches to interactive discovery (see Figures 52 and 53). This interface provides a visualization of the data being analyzed, the patterns detected, and the similarity between those patterns. It also allows the user to directly manipulate the pattern boundaries, which facilitates variable-length motif detection, and it provides a mechanism for merging patterns that were erroneously detected as distinct motifs.

The interface is designed to be as domain independent as possible, but it does allow the incorporation of custom modules that are tailored to the kind of data being analyzed. For example, there is an interactive map component that renders GPS traces, a video component that can render multiple video segments side-by-side to facilitate a visual assessment of similarity, and an audio component that shows the spectrogram of a segment and allows playback.

Initial experiments with the exercise data set show that higher accuracy rates can be achieved using the interactive system compared to a fully unsupervised algorithm. This result is not surprising given that the unsupervised algorithm already performs well and considering that the data set is relatively simple. Additional experiments are needed to evaluate the usefulness of the interface with different data sets, different domains, and with users with varying levels of familiarity with both the tool and the data.

7.7 Predictive Motifs

In this research, a motif has been defined as a set of similar subsequences embedded in time series data. Typically, a discovery algorithm will also learn a model that captures the salient features of the set so that additional occurrences can be detected in new time series. It is assumed that motifs with high similarity are unlikely to arise randomly and thus will provide insight into the underlying phenomena and will be useful for prediction or planning. The discovery algorithms presented here and elsewhere in the literature, however, do not guarantee that the motifs are useful nor is any notion of predictive power built in to the overall search or optimization procedures.

For some applications, a more useful tool would explicitly search for patterns that help make predictions about upcoming observations. A simple way to detect such patterns is to run a standard discovery algorithm and then test each motif to determine its predictive power. This approach may prove to be quite inefficient, however, if most of the discovered patterns turn out to provide little information about future data. A more interesting and challenging approach would incorporate the predictive requirement into the search procedure itself.

7.8 Transfer Learning

Transfer learning is a form of machine learning in which a solution for one problem is used to help find a solution for another problem more quickly than would be possible with a direct search. For pattern discovery, ideas from transfer learning can be applied by analyzing data in one domain and learning useful parameter values that can be used for future analysis in related domains. For instance, the typical word duration and the number of temporal bins needed to analyze speech data within a local discretization framework may be relatively stable across speakers and across languages. Thus, after performing a "blind" search on several base speech data sets, the discovery system can save the best parameter values and then use them to guide the search on future speech data. This approach becomes even more useful if the base data sets are manually annotated so that the parameters can be revised under supervision. The goal is that the analysis of future speech data sets would require less time since the parameters would not need to be adapted and that they would be more accurate.

7.9 Conclusions

In this dissertation, we have explored several approaches to real-valued, multivariate motif discovery. Two variations were investigated. In the "all-dimensional" case, several algorithms were developed to efficiently detect unknown recurring patterns when all of the dimensions in the time series data were relevant to the patterns. Broadly, the algorithms first run a fast, coarse analysis to detect candidate patterns, and then perform a more detailed analysis to select the patterns with the best support.

Three approaches for the coarse analysis were analyzed: global discretization, local discretization, and a real-valued density-based method. The experiments indicated that the global discretization could be very brittle, the density-based approach was the most accurate, while the local-discretization provided a nice trade-off between execution speed and detection accuracy.

In addition, two motif models were evaluated: a prototype-based hypersphere detector and a HMM-based model. For data sets with relatively dense patterns or for those with simple non-pattern regions, the HMM-based model used within a competitive framework led to higher accuracy rates than the hypersphere model. Detecting patterns using the hypersphere model lead to better results, however, when the background data was complex. In those situations, the background HMM was too general, which led to false positive errors when the motif models gave higher likelihood scores to non-pattern regions than the background model.

The second pattern discovery variation explored in this dissertation is the subdimensional version. Subdimensional discovery requires that the algorithm determine which dimensions of the time series data are relevant to each pattern. An extension of the `AdaptLD` algorithm was developed that is able to detect patterns despite distracting features and can then determine which features are relevant.

The discovery algorithms and associated experiments presented here demonstrate that unsupervised detection of recurring temporal patterns can be both fast and accurate across many different domains. There is still much research that needs to be done to broaden the applicability of discovery algorithms and make them easier to use and less sensitive to the given parameters. The methods developed as part of this research, however, represent a step forward for multivariate time series analysis.

REFERENCES

[1] Alon, J., Sclaro, S., Kollios, G., and Pavlovic, V., "Discovering clusters in motion time-series data," in *IEEE Computer Vision and Pattern Recognition Conference*, 2003.

[2] Ashbrook, D. and Starner, T., "Using GPS to learn significant locations and predict movement across multiple users," in *Personal and Ubiquitous Computing*, pp. 275–286, October 2003.

[3] Bailey, T. and Elkan, C., "Fitting a mixture model by expectation maximization to discover motifs in biopolymers," in *Second International Conference on Intelligent Systems for Molecular Biology*, pp. 28–36, AAAI Press, 1994.

[4] Black, M. and Jepson, A., "Recognizing temporal trajectories using the condensation algorithm," in *3rd IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pp. 16–21, April 1998.

[5] Blekas, K., Fotiadis, D., and Likas, A., "Greedy mixture learning for multiple motif discovery in biological sequences," *Bionformatics*, vol. 19, no. 5, 2003.

[6] Bobick, A. F. and Wilson, A. D., "A state based approach to the representation and recognition of gesture," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 19, pp. 1325–1337, December 1997.

[7] Brand, M., Oliver, N., and Pentland, A., "Coupled hidden markov models for complex action recognition," in *Computer Vision and Pattern Recognition*, 1997.

[8] Buhler, J. and Tompa, M., "Finding motifs using random projections," in *International Conference on Computational Biology*, pp. 69–76, 2001.

[9] Catalano, J., Armstrong, T., and Oates, T., "Discovering patterns in real-valued time series," in *Proc. of the Tenth European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, September 2006.

[10] Chiu, B., Keogh, E., and Lonardi, S., "Probabilistic disovery of time series motifs," in *Conf. on Knowledge Discovery in Data*, pp. 493–498, 2003.

[11] Clarkson, B. and Pentland, A., "Unsupervised clustering of ambulatory audio and video," in *ICASSP*, 1999.

[12] Cleveland, W., "Robust locally weighted regression and smoothing scatterplots," *Journal of the American Statistical Association*, vol. 74, pp. 829–836, December 1979.

[13] Cohen, P. R., Oates, J. T., and Beal, C. R., "Robots that learn meanings," in *submitted to First Joint Conference on Autonomous Agents and Multiagent Systems*, 2002.

[14] Comaniciu, D. and Meer, P., "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 24, pp. 603–619, May 2002.

[15] Comaniciu, D., Ramesh, V., and Meer, P., "The variable bandwidth mean shift and Data-Driven scale selection," in *IEEE Int. Conf. Computer Vision*, pp. 438–445, July 2001.

[16] Cormen, T., Leiserson, C., and Rivest, R., *Introduction to Algorithms.* The MIT Press, 1997.

[17] d'Avella, A. and Bizzi, E., "Shared and specific muscle synergies in natural motor behaviors," *Proc. of the National Academy of Science of the United States of America*, vol. 102, no. 8, pp. 3076–3081, 2005.

[18] Denton, A., "Kernel-density-based clustering of time series subsequences using a continuous random-walk noise model," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, November 2005.

[19] Doucet, A., de Freitas, N., and Gordon, N., *Sequential Monte Carlo Methods in Practice.* Springer, 2001.

[20] Duda, R., Hart, P., and Stork, D., *Pattern Classification, 2nd ed.* John Wiley & Sons, Inc., 2001.

[21] Fails, J. A. and Olsen, Jr., D. R., "Interactive machine learning," in *8th International Conference on Intelligent User Interfaces*, pp. 39–45, ACM, 2003.

[22] Gaffney, S. and Smyth, P., "Joint probabilistic curve clustering and alignment," in *Advances in Neural Information Processing*, vol. 17, 2004.

[23] Gray, A. G. and Moore, A. W., "Very fast multivariate kernel density estimation via computational geometry," in *Proc. of the ASA Joint Statistical Meeting*, 2003.

[24] Gusfield, D., *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cmbridge University Press, 1997.

[25] Hamid, R., Maddi, S., Bobick, A., and Essa, I., "Structure from statistics - unsupervised activity analysis using suffix trees," in *Proc. of Int. Conf. on Computer Vision*, 2007.

[26] Hamid, R., Johnson, A., Batta, S., Bobick, A., Isbell, C., and Coleman, G., "Detection and explanation of anomalous activities: Representing activities as bags of even n-grams," in *Computer Vision and Pattern Recognition*, June 2005.

[27] Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I., and Isbell, C., "Unsupervised activity discovery and characterization from event-streams," in *Uncertainty in Artificial Intelligence*, July 2005.

[28] Hand, D. J. and Bolton, R. J., "Pattern discovery and detection: A unified statistical methodology," *Journal of Applied Statistics*, vol. 31, no. 8, pp. 885–924, 2004.

[29] HTK, "HTK Speech Recognition Toolkit. Machine Intelligence Laboratory, Cambridge University. http://htk.eng.cam.ac.uk," 2007.

[30] Huynh, T. and Schiele, B., "Unsupervised discovery of structure in activity data using multiple eigenspaces," in *2nd International Workshop on Location- and Context-Awareness (LoCA)*, Springer, May 2006.

[31] Isard, M. and Blake, A., "CONDENSATION – conditional density propagation for visual tracking," *Int. Journal on Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[32] Itti, L. and Baldi, P., "A principled approach to detecting surprising events in video," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (San Siego, CA), pp. 631–637, June 2005.

[33] Ivanov, Y. and Bobick, A., "Recognition of visual activities and interactions by stochastic parsing," *PAMI*, vol. 22, pp. 852–872, August 2000.

[34] Jensen, K., Styczynski, M. P., Rigoutsos, I., and Stephanopoulos, G., "A generic motif discovery algorithm for sequential data," *Bioinformatics*, vol. 22, no. 1, pp. 21–28, 2006.

[35] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S., "Dimensionality reduction for fast similarity search in large time series databases," *Journal of Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, 2000.

[36] Keogh, E. and Folias, T., "UCR time series data mining archive," 2002.

[37] Keogh, E., Lin, J., and Truppel, W., "Clustering of time series subsequences is meaningless: Implications for past and future research," in *ICDM*, pp. 115–122, 2003.

[38] Keogh, E., "Personal communication," 2006.

[39] Keogh, E. and Ratanamahatana, C. A., "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[40] Keogh, E., "Efficiently finding arbitrarily scaled patterns in massive time series databases," in *Principles and Practice of Knowledge Discovery*, pp. 253–265, 2003.

[41] Ketkar, N., Holder, L., Cook, D., Shah, R., and Coble, J., "Subdue: Compression-based frequent pattern discovery in graph data," in *ACM KDD Workshop on Open-Source Data Mining*, August 2005.

[42] Kuhn, H. W., "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[43] Lafferty, J., McCallum, A., and Pereira, F., "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Int. Conf on Machine Learning*, 2001.

[44] Leonard, R. G., "A database for speaker-independent digit recognition," in *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 328–331, March 1984.

[45] Leonard, R. G. and Doddington, G., "TIDIGITS Linguistic Data Consortium, Philadelphia (http://www.ldc.upenn.edu/catalog/catalogentry.jsp?catalogid=ldc93s10)," 1993.

[46] Leonardis, A., Bischof, H., and Maver, J., "Multiple eigenspaces," *Pattern Recognition*, vol. 35, no. 11, pp. 2613–2627, 2002.

[47] Lester, J., Choudhury, T., Kern, N., Borriello, G., and Hannaford, B., "A hybrid discriminative-generative approach for modeling hman activities," in *Proc. of Int. Joint Conf. on Artificial Intelligence*, July 2005.

[48] Liao, L., Choudhury, T., Fox, D., and Kautz, H., "Training conditional random fields using virtual evidence boosting," in *International Joint Conference on Artificial Intelligence*, January 2007.

[49] Liao, T. W., "Clustering of time series data - a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[50] Lin, J., Keogh, E., Lonardi, S., and Chiu, B., "A symbolic representation of time series, with implications for streaming algorithms," in *8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, June 2003.

[51] Lin, J., Vlachos, M., Keogh, E., and Gunopulos, D., "Iterative incremental clustering of time series," in *Int. Conf. on Extending Database Technology*, pp. 106–122, 2004.

[52] LIN, Y.-L., JIANG, T., and CHAO, K.-M., "Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis," *Journal of Computer and System Sciences*, vol. 65, pp. 570–586, January 2002.

[53] LIU, T., *Fast Nonparametric Machine Learning Algorithms for High-dimensional Massive Data and Applications*. PhD thesis, Carnegie Mellon University, 2006.

[54] LOFTSGAARDEN, D. and QUESENBERRY, C., "A nonparametric estimate of a multivariate density function," *The Annals of Mathematical Statistics*, vol. 36, pp. 1049–1051, 1965.

[55] LUKOWICZ, P., WARD, J., JUNKER, H., STÄGER, M., TRÖSTER, G., ATRASH, A., and STARNER, T., "Recognizing workshop activity using body worn microphones and accelerometers," in *Pervasive Computing*, pp. 18–22, 2004.

[56] MINNEN, D., ESSA, I., and STARNER, T., "Expectation grammars: Leveraging high-level expectations for activity recognition," in *Computer Vision and Pattern Recognition*, June 2003.

[57] MINNEN, D., STARNER, T., ESSA, I., and ISBELL, C., "Discovering characteristic actions from on-body sensor data," in *To appear in Int. Symp. on Wearable Computers (ISWC)*, October 2006.

[58] MINNEN, D., STARNER, T., ESSA, I., and ISBELL, C., "Discovering multivariate motifs using subsequence density estimation," in *AAAI Conf. on Artificial Intelligence*, July 2007.

[59] MINNEN, D., STARNER, T., ISBELL, C., and ESSA, I., "Improving activity disocvery with automatic neighborhood estimation," in *Submitted to Int. Joint Conf. on Artificial Intelligence IJCAI*, January 2007.

[60] MINNEN, D., WESTEYN, T., STARNER, T., WARD, J., and LUKOWICZ, P., "Performance metrics and evaluation issues for continuous activity recognition," in *Performance Metrics for Intelligent Systems*, August 2006.

[61] MINNEN, D. and WREN, C., "Finding temporal patterns by data decomposition," in *Sixth Int. Conf. on Automatic Face and Gesture Recognition*, May 2004.

[62] MOORE, A., "Predicting real-valued outputs: an introduction to regression," 2003.

[63] MOORE, D., ESSA, I., and HAYES, M., "Context management for human activity recognition," in *Proc. of Audio and Vision-based Person Authentication 1999*, 1999.

[64] Moore, D., Essa, I., and Hayes, M., "Exploiting human actions and object context for recognition tasks," in *ICCV'99*, pp. 80–86, 1999.

[65] Mörchen, F., "A better tool than allen's relations for expressing temporal knowledge in interval data," in *Proc. of Temporal Data Mining Workshop at ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[66] Mörchen, F. and Ultsch, A., "Efficient mining of understandable patterns from multivariate interval time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 181–215, 2007.

[67] Munkres, J., "Algorithms for the assignment and transportation problems," *Journal of the Society of Industrial and Applied Mathematics*, vol. 5, pp. 32–38, March 1957.

[68] Naphade, M. and Huang, T., "Discovering recurrent events in video using unsupervised methods," in *IEEE Int. Conf. on Image Processing*, September 2002.

[69] Oates, T., "PERUSE: An unsupervised algorithm for finding recurring patterns in time series," in *Int. Conf. on Data Mining*, pp. 330–337, 2002.

[70] Oates, T., Firoiu, L., and Cohen, P. R., *Sequence Learning: Paradigms, Algorithms and Applications*, ch. Using Dynamic Time Warping to Bootstrap HMM-Based Clustering of Time Series. Springer-Verlag, 2000.

[71] Park, A., *Unsupervised Pattern Discovery in Speech: Applications to Word Acquisition and Speaker Segmentation.* PhD thesis, Massachusetts Institute of Technology, 2006.

[72] Park, A. and Glass, J. R., "Towards unsupervised pattern discovery in speech," in *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop*, December 2005.

[73] Park, A. and Glass, J. R., "Unsupervised word acquisition from speech using pattern discovery," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, May 2006.

[74] Parry, R. M. and Essa, I., "Blind source separation using repetitive structure," in *International Conference on Digital Audio Effects*, (Madrid), pp. 143–148, September 2005.

[75] Patel, P., Keogh, E., Lin, J., and Lonardi, S., "Mining motifs in massive time series databases," in *Int. Conf. on Data Mining*, pp. 370–377, 2002.

[76] Patterson, D., Liao, L., Fox, D., and Kautz, H., "Inferring high-level behavior from low-level sensors," in *5th International Conference on Ubiquitous Computing*, October 2003.

[77] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical REcipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[78] Rabiner, L. and Juang, B.-H., *Fundamentals of Speech Recognition.* Signal Processing Series, Prentice Hall, 1993.

[79] Rabiner, L., "A tutorial on hidden markov models and selected applications in speech recognition," *Readings in speech recognition*, pp. 267–296, 1990.

[80] Ratanamahatana, C. and Keogh, E., "Three myths about dynamic time warping," in *SIAM Int. Conf. on Data Mining*, pp. 506–510, 2005.

[81] Sain, S., "Multivariate locally adaptive density estimation," tech. rep., Department of Statistical Science, Southern Methodist University, 1999.

[82] Schlenzig, J., Hunter, E., and Jain, R., "Recursive identification of gesture inputs using hidden markov models," in *WACV94*, pp. 187–194, 1994.

[83] Shashanka, M., Raj, B., and Smaragdis, P., "Probabilistic latent variable model for sparse decompositions of nonnegative data," *To appear: Trans. on Pattern Analysis and Machine Intelligence*, 2007.

[84] Shi, Y., Huang, Y., Minnen, D., Bobick, A., and Essa, I., "Propagation networks for recognition of partially ordered sequential action," in *IEEE Conference on Computer Vision and Pattern Recognition*, (Washington DC, USA), pp. II862–870, IEEE Computer Society, June 2004.

[85] Silverman, B., *Density Estimation.* London: Chapman and Hall, 1986.

[86] Smaragdis, P. and Raj, B., "Shift-invariant probabilistic latent component analysis," *To appear: Journal of Machine Learning Research*, 2007.

[87] Starner, T., Weaver, J., and Pentland, A., "Real-time american sign language recognition using desk and wearable computer based video," *PAMI*, vol. 20, pp. 1371–1375, December 1998.

[88] Tanaka, Y. and Uehara, K., "Discover motifs in multi-dimensional time-series using the principal component analysis and the MDL principle," in *International Conference on Machine Learning and Data Mining*, pp. 252–265, 2003.

[89] Tanaka, Y., Iwamoto, K., and Uehara, K., "Discovery of time-series motif from multi-dimensional data based on mdl principle," *Machine Learning*, vol. 58, no. 2-3, pp. 269–300, 2005.

[90] Ukkonen., E., "Constructing suffix-trees on-line in linear time," *Algorithms*, vol. 1, no. 92, pp. 484–492, 1992.

[91] Vlassis, N. and Likas, A., "A greedy EM algorithm for Gaussian mixture learning," *Neural Processing Letters*, vol. 15, February 2002.

[92] Vogler, C. and Metaxas, D., "ASL recognition based on a coupling between hmms and 3d motion analysis," in *ICCV98*, pp. 363–369, 1998.

[93] Wang, P., Lee, D., Gray, A., and Rehg, J. M., "Fast mean shift with accurate and stable convergence," in *AI and Statistics*, March 2007.

[94] Ward, J. A., Lukowicz, P., and Tröster, G., "Evaluating performance in continuous context recognition using event-driven error characterisation," in *Int. Workshop on Location and Context-Awareness*, pp. 239–255, 2006.

[95] Ware, M., Frank, E., Holmes, G., Hall, M., and Witten, I. H., "Interactive machine learning: letting users build classifiers," *Int. J. Hum.-Comput. Stud.*, vol. 55, no. 3, pp. 281–292, 2001.

[96] Xie, L., Change, S.-F., Divakaran, A., and Sun, H., *Unsupervised Mining of Statistical Temporal Structures in Video*, ch. 10. Kluwer Academic Publishers, 2003.

[97] Yamato, J., Ohya, J., and Ishii, K., "Recognizing human action in time-sequential images using a hidden Markov model," in *CVPR1992*, pp. 379–385, 1994.

[98] Yankov, D., Keogh, E., Medina, J., Chiu, B., and Zordan, V., "Detecting motifs under uniform scaling," in *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, August 2007.

[99] Young, S., Russell, N., and Thornton, J., "Token passing: a simple conceptual model for connected speech recognition systems," Tech. Rep. 38, Cambridge University, 1989.

[100] Zhong, H., Shi, J., and Visontai, M., "Detecting unusual activity in video," in *Computer Vision and Pattern Recognition*, June 2004.