ANALYZING REPETITIVE SEQUENCES WITH STRUCTURED DYNAMIC BAYESIAN NETWORKS

Thomas L. Lake, M.S.

Western Michigan University, 2015

Time series often feature structure that is known a priori and easily described using natural language terms such as repetitive, symmetric, seasonal, and self-similar. However, the typical priors used in Bayesian analysis, normally chosen due to the convenience of conjugacy, do not capture such complex phenomena well. As a result of this mismatch, known structure is modeled poorly or completely ignored. Focusing on time series with repetitive structure, this thesis proposes to overcome this problem by reducing, rather than increasing, the capacity of a well know time series model, the Hidden Markov Model. Through a careful choice in the way model capacity is reduced, the model is forced to use its latent variables in an interpretable way which accurately reflects known structure. In addition to increased modeling performance, the lower capacity model also admits more efficient inference and parameter estimation procedures. Experimental results are presented demonstrating the effectiveness of this approach both in a supervised and unsupervised learning context.

ANALYZING REPETITIVE SEQUENCES WITH STRUCTURED DYNAMIC BAYESIAN NETWORKS

by

Thomas L. Lake

A dissertation submitted to the Graduate College
in partial fulfillment of the requirements
for the degree of Master of Science
in Computer Science
Western Michigan University
October 2015

Thesis Committee:

Elise de Doncker, Ph.D., Chair
Robert Trenary, Ph.D.
John Kapenga, Ph.D.

Acknowledgments

I want to thank my advisor Elise deDoncker and Robert Trenary for all their support.

Contents

List of Tables

List of Figures

Acronyms

AMI	Adjusted Mutual Information
CHMM	Cyclic Hidden Markov Model
CRF	Conditional Random Field
DBN	Dynamic Bayesian Network
DFA	Deterministic Finite State Automaton
DPMoCHMM	Dirichlet Process Mixture of Cyclic Hidden Markov Models
DPMoHMM	Dirichlet Process Mixture of Hidden Markov Models
EM	Expectation Maximization
HAR	Human Activity Recognition
HHMM	Hierarchical Hidden Markov Model
HMM	Hidden Markov model
iid	independent and identically distributed
LDA	Latent Dirichlet Allocation
MAP	Maximum a Posteriori
MCMC	Markov Chain Monte Carlo
MEMM	Maximum Entropy Markov Model
MoCHMM	Mixture of Cyclic Hidden Markov Models
MoHMM	Mixture of Hidden Markov Models
NLP	Natural Language Processing
PCFG	Probabilistic Context Free Grammar

pdf	probability density function
PGM	Probabilistic Graphical Model
SVM	Support Vector Machine

Notation

As this thesis deals almost exclusively with time series, t will always be used as a time index and T as the length of the time series. A sequence from time t_1 to t_2 is denoted by $x_{t_1:t_2} = (x_{t_1}, x_{t_1+1}, \ldots, x_{t_2})$. Following this notation the entire sequence is denoted $x_{1:T}$.

Random variables are denoted by uppercase letters X, Y and the values these variables take by the corresponding lower case letter x, y. The probability that a particular random variable X takes a value x is denoted $P(X = x)$, and $P(X = x \mid Y = y)$ denotes the conditional probability that $X = x$ given the value of some other variable $Y = y$. If clear from context the uppercase letter will be omitted and the previous expressions will simply be written $P(x)$ and $P(x \mid y)$. If x is distributed according to some distribution D with parameters θ we write $x \sim \mathrm{D}(\theta)$. Common abbreviations used for well known parametric probability distributions are listed in Table 1.

Distribution	**Abbreviation**
Categorical	Cat
Geometric	Geom
Uniform	Unif
Beta	Beta
Dirichlet	Dir
Normal	Norm

Table 1: Distribution abbreviations.

Chapter 1

Introduction

Uncovering the underlying structure giving rise to a set of observations is a fundamental theme in Machine Learning. The canonical representation of this structure is the Probabilistic Graphical Model (PGM). The inclusion or exclusion of edges in a PGM explicitly imposes structural relationships among variables and controls the ways in which they may interact. Considerable attention has been given to the problem of automated structure discovery. Algorithms for structure discovery typically start as a blank slate, attempting to induce structure from observations alone.

On the other hand, when applying Machine Learning to solve real world problems, there is often background knowledge available about the target domain. The ability to exploit this knowledge to enable or improve potential solutions is of obvious virtue. When this domain knowledge can be expressed in the form of dependencies which are known to exist or not exist a priori, this knowledge can be easily incorporated into the PGM formalism. When domain knowledge is more nebulous, and can not straightforwardly be expressed in the form of dependencies, incorporating this knowledge becomes more difficult. Examples of this sort of knowledge include repetition in music, symmetry in images, and locality in space. In the context of PGMs, the later has been addressed effectively through the use of grid-structured models, and similar techniques could likely be employed to capture notions of symmetry. Taking inspiration from these methods this work focuses on repetition in time series, using several tasks in the activity recognition domain as motivating examples.

1.1 Contributions

The most important contribution of this work is the identification and empirical exploration of a subclass of Hidden Markov models (HMMs) especially well suited to capture repetitive structure in time series. The subclass of HMM

considered here is less computationally demanding than a standard HMM, and is shown to perform well on a variety of activity recognition tasks both within the supervised and unsupervised learning framework. Furthermore, interpreting the latent state sequence found by these models yields highly accurate repetition counts.

To further demonstrate the effectiveness of this subclass of HMM, it is utilized as the base distribution in a Bayesian nonparametric mixture model. To the author's knowledge, nonparametric mixtures of HMMs have not been previously discussed in the literature. To support inference in this model this work extends the collapsed block Gibbs sampler of [15] to jointly sample in the space of cluster assignments and latent state sequences.

1.2 Organization

The remainder of this thesis is organized as follows:

Chapter 2 formalizes the Human Activity Recognition problem and reviews the existing literature, with a focus on sensor based systems.

Chapter 3 describes the dataset used for the real world experiments presented in this thesis. This includes both the methodologies used for data collection, and labeling schemes developed to support subsequent experiments.

Chapter 4 provides relevant background related to Hidden Markov Models and the more general Dynamic Bayesian Network framework. This chapter also introduces a novel Bayesian Nonparametric extension of Mixtures of Hidden Markov Models.

Chapter 5 introduces a structured subclass of Hidden Markov Models which are particularly well suited to the types of problems explored in this thesis.

Chapter 6 presents synthetic experiments which are intended to explore the capabilities of the techniques introduced in Chapter 4 and 5.

In Chapter 7 the proposed model is shown to compare favorably to a strong performing system influenced by the existing literature in a supervised exercise classification setting.

Chapter 8 demonstrates the use of the proposed model for unsupervised time series segmentation.

Chapter 9 presents experiments using Bayesian nonparametrics to cluster exercises.

Chapter 10 contains concluding remarks and future directions.

Chapter 2

Analyzing Human Activity with Sensors

The availability of small low cost microprocessors and sensors has enabled the creation of numerous wearable devices for research, industrial, and commercial applications. These applications target a diverse set of domains including fitness [9], health status monitoring [20, 33], and workplace safety [52]. Despite the variety of applications, a common underlying theme to most of these works is Human Activity Recognition (HAR), the act of identifying what activity the wearer is engaged in at a given time. Upon further consideration this is to be expected; the ability of a system to provide meaningful output is dependent on the ability to *understand* the current context. Informally a HAR system can be described as a mapping from sequential input data to an ordered list of activities. The specifics of the input data, as well as the type and granularity of the recognized activities, varies greatly from system to system.

2.1 Sensors

HAR systems have been designed to work with numerous input sources including video [37], audio [52], RFID [53], location, temperature, motion sensors [24], or combinations thereof. Keeping in line with much of the HAR literature, the focus of this work shall be on wearable sensor based systems, in particular systems utilizing accelerometers and gyroscopes [2] as input. In addition to sensor type, systems vary both in body location and the number of locations used.

From an implementation perspective there are obvious advantages to using a variety of sensors placed at a number of different locations on the body. Multiple locations allow a more complete picture of the activity being performed, especially since the motion associated with an activity may be isolated to a

single limb or portion of the body, e.g., writing or riding a bike. However, if such systems are to be actually utilized, requiring a user to attach multiple devices to different body locations is likely impractical and obtrusive [24]. As an extreme example of location proliferation, [19] presents a system composed of twelve devices attached at distinct locations.

2.2 Activities

Given the various application domains targeted by HAR systems it is not surprising that the definition of *activity* adopted by these systems varies greatly as well. To frame the current work in the context of the broader HAR literature systems will be divided into two categories based on activity specificity.

On one end of the spectrum are systems which recognize broad category daily activities such as walking, driving, making a phone call, or watching TV [2]. Activities in this group typically last for a few minutes to several hours, and can largely be characterized by general levels of motion rather than a precise series of movements. A useful feature of such broad categories is that a wearer's day can in theory be segmented into a series of such activities, allowing for a comprehensive analysis of general lifestyle patterns. Unfortunately, broad categories have the disadvantage of being plagued by ambiguity, non-mutual exclusivity, and intra-category variability. [2] use 20 broad categories including Watching TV, Walking, Running, and Folding Laundry. Given these categories: What is the correct classification of someone folding laundry while watching TV? At what speed does one stop walking and begin running? Is a jogger walking or running? The answers to such questions will largely depend on the target domain and fundamentally rule out certain post-hoc analysis [4].

On the other end of the spectrum are systems with narrow activity categories. A common example from the literature, and the one explored in this work, is that of Exercise classification [28]. Within this regime activities last for much briefer periods of time, on the order of seconds to minutes. Furthermore, there is little chance for category ambiguity as categories are sufficiently narrow to avoid many of problems associated with broad categories mentioned above. However, narrow categories are not without problems. The most obvious is that it requires either the definition of an unreasonably large number of categories, or the introduction of a *none of these* type category. The former is problematic both from a computational and dearth of data perspective, while the later will be plagued by problems associated with class imbalance [13]. Recognizing narrow category activity is closely related to problem of Gesture Recognition [54].

Several works have focused on crossing the boundaries between broad and narrow categories using either an implicit or explicit hierarchical decomposition of activities [6, 49]. In this framework, potentially shared, narrow activities are combined to form broad activities. To date, these systems do not use a number of broad activities that would be necessary to fully segment an entire day into activity categories.

2.3 Problem Formalization

The HAR task will be formally defined as follows: Let $x_{1:T}$ be a sequence of possibly multivariate observations with $x_t \in \mathcal{X}$, and $\mathcal{Y}$ be a finite set of activities[1]. Then the output of a HAR system is a sequence $y_{1:T}$ with $y_t \in \mathcal{Y}$ such that y_t is the activity being performed at time t.

The definition presented above is the most general output one could expect from a HAR system, and specific HAR tasks present in the literature can be seen as special cases or transformations of the output. For example if, one only wishes to account for the amount of time spent performing each activity, if suffices to set

$$s_y = \sum_t \mathbb{1}\{y_t = y\}$$

for each $y \in \mathcal{Y}$. As another example, Algorithm 1 may be used to obtain an ordered list of the activities performed from the output $y_{1:T}$ described above.

Algorithm 1 HAR Activity List.

Input: $y_{1:T}$ a sequence of activity predictions from time 1 to T.

```
S ← stack()
push(S, y_1)
for t ← 2, ..., T do
    if last(S) ≠ y_t then
        push(S, y_t)
    end if
end for
return S
```

2.4 Windows and Features

The HAR problem if commonly decomposed into a series of independent decisions problems [52, 5, 28, 21]. When this approach is used the resulting

[1]For example, if $x_{1:T}$ is a sequence of readings from a triaxial accelerometer then $\mathcal{X} = \mathbf{R}^3$.

classifier operates over fixed size temporal windows of the data, which will be referred to as *windowing*. The typical windowing setup proceeds as follows:

- Split the sequence into a series of fixed size possibly overlapping windows of size n, $w_1, w_2, \ldots, w_m$, $w_i = x_{t:t+n}$.
- Map each window w_i to a corresponding feature vector $\Phi(w_i)$.
- For each i use any classification algorithm to predict the activity occurring within the window, $\hat{y}_i = h(\Phi(w_i))$.

Assuming d real valued inputs are observed at each time step, the feature function Φ can be described as

$$\begin{aligned} \Phi \colon \mathbb{R}^{n \times d} &\longrightarrow \mathbb{R}^k \\ x_{t:t+n} &\longmapsto (\phi_1(x_{t:t+n}), \ldots, \phi_k(x_{t:t+n})), \end{aligned}$$

where n is the size of the window, ϕ_i are the component feature functions, and k is the the number of features computed for each window.

A variety of feature function ϕ_i have been proposed in the literature, several of which are described below. For ease of exposition the input will be assumed to be a one dimensional real valued time series and the features computed over the first window spanning from time 1 to n. Generalization to multiple dimensions and other window sizes and positions are straightforward.

Mean and Variance:

$$\begin{aligned} \mu &= \frac{1}{n} \sum_{t=1}^{n} x_t \\ \sigma^2 &= \frac{1}{n-1} \sum_{t=1}^{n} (x_t - \mu)^2 \end{aligned}$$

Root Mean Square:

$$\mathrm{RMS} = \sqrt{\frac{1}{n} \sum_{t=1}^{n} x_t^2}$$

Autocorrelation: (lag k)

$$a_k = \frac{1}{(n-k)\sigma^2} \sum_{t=1}^{n-k} (x_t - \mu)(x_{t+k} - \mu)$$

Energy:

$$E = \frac{1}{n} \sum_{t=1}^{n} F_i^2$$

where F_i is the i^{th} component of the Fourier transform of x.

There are many advantages to a windowed approach to sequential classification problems. Most notably, using a fixed sized window as input allows one to sidestep the need to deal explicitly with the temporal aspects of the problem. This frees the practitioner to leverage the vast array of classification techniques suitable only for static inputs. Another advantage of windowing is it allows predictions to be obtained in a near realtime manner, as prediction lag is only fundamentally limited by the size of the window. This is in contrast with classification methods designed specifically to work with sequential data, which typically involve some sort of inference procedure to handle temporal dependencies between observations and/or predictions.

Despite the advantages of windowing, neglecting the sequential nature of the problem is the largest drawback of the approach. As only a small window is available to the classifier, context outside this window is unavailable, rendering the predicted activity of each window independent. Furthermore, because the window size must fixed a priori, an activity which requires more than this time to be identified will potentially fail to be recognized as accurately.

2.5 Related Work

2.5.1 Supervised Approaches

Most approaches to HAR in the literature fall within the supervised Machine Learning framework [24], i.e., sensor sequences are paired with label sequences denoting the activity being done at each point in time.

The closest work in terms of domain and scope to that presented here is [28]. Like this work, they work within the exercise classification domain utilizing data from 96 participants for 26 exercises collected from a single device attached to the participant's arm. Although they collected data for 26 activities, results are only presented for discrimination among 13 different exercises (this work presents experiments using 50 exercises). Their focus is on the development and validation of an end-to-end realtime classification system, while this work focuses primarily on several tasks related to the development of such a system. Their full system is based on a three stage pipeline consisting of: segmentation, classification, and repetition counting. The segmentation and classification systems are essentially the same, differing only slightly in features used and the former being a binary classifier and the later being a multi-class classifier. The classifier used in both cases is a Support Vector Machine (SVM) operating over 5 second windows. Their repetition counting procedure is based on a heuristic peak counting algorithm.

Another closely related work is [6], although they focus primarily on active learning. Their work presents a discriminative PGM for the HAR task based on a decomposition of activities into a sequence of attributes, a.k.a gestures. The attributes and their relation to the activity are hand engineered from domain expertise and ultimately serve the same role as features. At test time the attributes are predicted by a lower level classifier operating over fixed sized input windows. They apply their model to both daily activities and exercises, although not simultaneously.

2.5.2 Unsupervised Approaches

Unlike the supervised setting, ground truth labels are not provided to the learning algorithm in the unsupervised setting. Lacking labels, the goal of unsupervised HAR reduces to clustering, and possibly segmenting, sequences. This task has received considerably less attention from the HAR community, although some prior work does exist.

[12] use Latent Dirichlet Allocation (LDA) [3], an admixture model originally developed for modeling document topics, to infer daily activities such as working or commuting. However, rather than directly inferring activities from raw data, they use the outputs of traditional HAR classifiers trained in a supervised manner on lower level activities (walking, sitting, etc) as inputs to their unsupervised system.

In [27] a combination of algorithms for motif discovery [14] and PGMs are used to segment and cluster raw activity data. Their proposed approach con-

sists of iteratively building up a set a set of motifs from the raw data, refining the motifs, and then training an HMM for each motif. The HMMs are then used to segment and classify sequences containing 6 unique dumbbell exercises. Although their reported results are promising, details regarding the number of participants, and whether their system was evaluated on out-of-sample participants were omitted. Their proposed system also requires the user manually specify a large number of hyperparameters such as window size, number of motifs, number of HMM states, and several parameters for the motif refinement algorithm.

Chapter 3

Activity Data

Since there does not exist publicly available sensor data for the target domain, a privately funded large annotated dataset of different people performing a variety of exercises was utilized. Each instance in this dataset consists of contiguous sensor readings of a user performing a series of exercises (a *routine*), where each exercise is separated by non-exercise activities which includes activities such as stretching, drinking water, setting up equipment or recovery. Any non-exercise activity is referred to as *rest* throughout. Table 3.1 summarizes the dataset.

	Train	**Test**
Users	85	37
Classes	50	50
Routines	194	107
Activities	3364	1382
Repetitions	28476	12396
Minutes of Routine Data	3636	1357
Minutes of Activity Data	1247	557

Table 3.1: Summary of exercise dataset.

3.1 Hardware

Sensor data is collected from a wrist worn device positioned slightly above the head of the ulna on the left arm. The device collects 3-axis accelerometer and 3-axis gyroscope readings at 15Hz. This data is transferred in real-time to a PC using Blue Tooth Low Energy (BLE).

3.2 Data Collection Methodology

Participants in the data collection project were volunteers recruited utilizing social media and word of mouth. In addition to sensor data, demographic information including age, sex, height, weight, and a self reported expertise level was recorded.

Participants were instructed that they will be prompted to perform a series of exercises (between 10 and 20) along with a suggested number of repetitions. For exercises including variable weight they were directed to choose a weight with which they are comfortable. Furthermore, they were instructed to perform the exercises as naturally as possible, to skip any exercises with which they are unfamiliar, take frequent breaks, ask questions, and vary the number of repetitions to match their comfort level.

Each exercise instance within a routine was annotated by an observer with an exercise name, form variation (when applicable), start time, stop time, end of repetition times, and weight (when applicable). Figure 3.1 shows an example routine with exercise start time and stop time annotations.

3.3 Exercise Taxonomy

Exercise names are often ambiguous; several names can refer to the same exercise, and the same name can refer to different variations of the same exercise. The first type of naming ambiguity (many to one) is easily solved by defining a fixed reference name which is always used, but the second type (one to many) is more difficult to handle.

The second type of ambiguity typically involves positioning, resulting in exercises that are visually similar but whose sensor readings can be drastically different. Examples include supinated (underhand) vs pronated (overhand) for pull-ups and dumbbell curls, or hands on temple vs hands behind head vs arms crossed for sit-ups and crunches. To solve this problem a standard naming convention for exercises which include form variations was defined. This allows grouping form variations into different classes, which was found to be beneficial for some classifiers. The final naming convention we adopted is

```
[modifier] [equipment] <exercise> [(form variation)]
```

where square brackets and angle brackets denote optional and required arguments respectively. This convention results in exercise names such as

$$\underbrace{\text{alternating}}_{\text{modifier}}\ \underbrace{\text{dumbbell}}_{\text{equipment}}\ \underbrace{\text{bicep curl}}_{\text{exercise}}\ \underbrace{\text{(start with wrist facing forward)}}_{\text{form variation}}$$

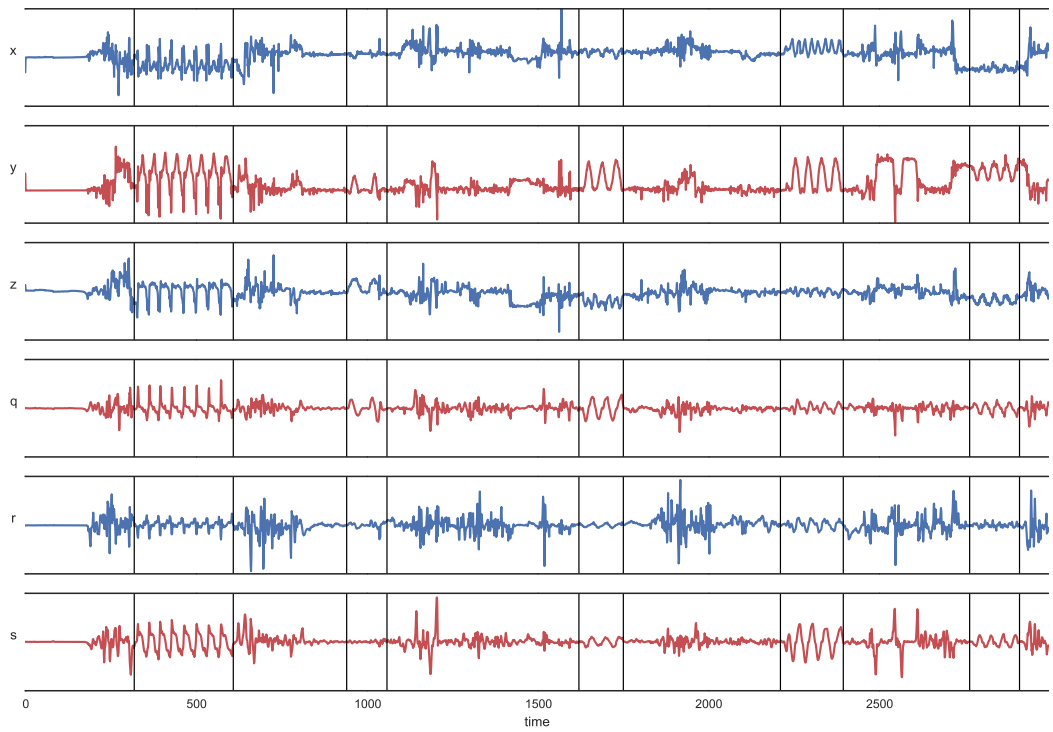


Figure 3.1: Example routine time series, vertically ordered by x, y, z (accelerometer, axis range from -3 to 3) and q, r, s (gyroscope, axis range from -600 to 600).

and

$$\overbrace{\phantom{\text{modifier}}}^{\text{modifier}}\ \overbrace{\phantom{\text{equipment}}}^{\text{equipment}}\ \underbrace{\text{pull-up}}_{\text{exercise}}\ \underbrace{(\text{supinated})}_{\text{form variation}}$$

Chapter 4

Hidden Markov Models

HMMs [38] are a common theme throughout this thesis. In the spirit of self containment this chapter introduces common terminology, methods and inference algorithms employed when working with HMMs. In particular the focus is on Bayesian methods. For a more standard introduction see Kevin Murphy's excellent thesis [29], which also discusses the more general class of probabilistic models known as Dynamic Bayesian Networks (DBNs), of which HMMs and the popular Kalman filter [18], are a member.

4.1 Introduction to Hidden Markov Models

HMMs are probabilistic state-space models for sequential data. The states in an HMM are members of a finite discrete set, and can be partitioned into a set of observed variables X, and latent (unobserved) variables Y. The latent variables evolve according to a first order Markov process:

$$P(y_t \mid y_{1:t-1}) = P(y_t \mid y_{t-1})$$

i.e., the latent state at time t depends only on the latent state at time $t-1$. Generalization to higher order dependencies is straightforward. It is typically assumed that the observation at time t is conditionally independent of all other variables given the latent variable at time t,

$$P(x_t \mid y_{1:T}) = P(x_t \mid y_t).$$

Again, this assumption can be relaxed in obvious ways [43]. For ease of exposition the simplest formulation presented above is used here.

Given the above assumptions the joint probability of a sequence of observations $x_{1:T}$ and latent states $y_{1:T}$ is given by

$$P(x_{1:T}, y_{1:T}) = P(y_1)P(x_1 \mid y_1) \prod_{t=2}^{T} P(y_t \mid y_{t-1})P(x_t \mid y_t). \tag{4.1}$$

The corresponding generative story is given by

1. Sample an initial state $y_1 \sim \text{Cat}(\pi_0)$.
2. Sample an initial observation $x_t \sim \text{D}(\phi_{y_1})$.
3. For $t = 2$ to T
   (a) Sample the next state $y_t \mid z_t \sim \text{Cat}(\pi_{y_{t-1}})$ conditioned on the previous state.
   (b) Sample the next observation $x_t \mid y_t \sim \text{D}(\phi_{y_t})$ conditioned on the current state.

where D is some parametric emission distribution. Dependencies can be visualized graphically in Figure 4.1, where vertices represent variables, edges represent dependencies, and grey vertices represent observed variables.

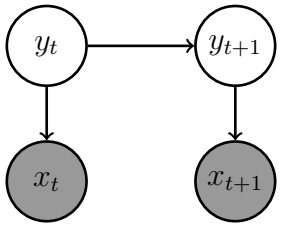


Figure 4.1: Slice representation of a HMM. Conditional dependencies and random variables are denoted by directed edges and vertices respectively. Shaded vertices correspond to random variables which are be observed.

4.1.1 Inference

This Section outlines two useful algorithms for HMMs assuming a fixed set of parameters. The first computes the likelihood of an observed sequence under the given parameters. The second computes the most likely sequence of states given an observed sequence. The number of states in the HMM will be denoted by K.

Likelihood

Computing the likelihood of an observed sequence $x_{1:T}$ can be done naively by marginalizing over all possible length T states sequences

$$P(x_{1:T}) = \sum_{y_{1:T}} P(x_{1:T}, y_{1:T})$$

Computation in this manner requires time $O(TK^T)$ and is infeasible for all but the smallest K and T. A more efficient algorithm exists requiring only time $O(TK^2)$. It is typically referred to as the forward, alpha, or simply sum-product algorithm in the more general context of PGMs.

Define $\alpha_{t,y_t} = P(y_t, x_{1:t})$ as the joint probability of the state at time t being y_t and the entire observation sequence up to time t. The key insight is that $\alpha_{t,*}$ may be computed recursively from $\alpha_{t-1,*}$ as

$$\begin{aligned}
\alpha_{t,y_t} &= P(y_t, x_{1:t}) \\
&= P(x_t \mid y_t) \sum_{y_{t-1}} P(y_t \mid y_{t-1}) P(y_{t-1}, x_{1:t-1}) \\
&= P(x_t \mid y_t) \sum_{y_{t-1}} P(y_t \mid y_{t-1}) \alpha_{t-1,y_{t-1}}
\end{aligned}$$

with the base case given by

$$\alpha_{1,y_1} = P(y_1) P(x_1 \mid y_1)$$

It follows that

$$P(x_{1:T}) = \sum_{y_T} P(y_T, x_{1:T}) = \sum_{y_T} \alpha_{T,y_T}.$$

The Maximum a Posteriori State Sequence

The algorithm for efficiently computing the most likely state sequence given an observed sequence is closely related to the algorithm for computing the likelihood of an observed sequence discussed above. It is commonly referred to as the Viterbi or max-product algorithm.

Define

$$v_{t,i} = \max_{y_{1:t}} \{P(x_{1:t}, y_{1:t}) \mid y_t = i\}$$

as the probability of the most likely sequence of t states ending in state i given $x_{1:t}$. Then $v_{t,*}$ may be computed recursively from $v_{t-1,*}$ as

$$\begin{aligned} v_{t,y_t} &= \max_{y_{t-1}} \{P(x_t \mid y_t) P(y_t \mid y_{t-1}) v_{t-1,y_{t-1}}\} \\ &= P(x_t \mid y_t) \max_{y_{t-1}} \{P(y_t \mid y_{t-1}) v_{t-1,y_{t-1}}\} \end{aligned}$$

with the base case given by

$$v_{1,y_1} = P(y_1) P(x_1 \mid y_1).$$

The most likely path is easily recovered by storing a $T \times K$ table of back pointers whose t, k entry is the most likely state leading to state k at time t.

4.2 The Bayesian Approach

Bayesian statistics treats quantities of interest as random variables. In the context of Machine Learning, Bayesian methods are frequently used for parameter estimation and model selection, although only the former is considered here. The Bayesian approach to parameter estimation begins by defining a prior distribution over model parameters before observing any data, $P(\theta)$. After observing data D one is typically interested in the posterior, i.e., the probability distribution over parameters, conditioned on observing D.

$$P(\theta \mid D) = \frac{P(\theta) P(D \mid \theta)}{P(D)}.$$

The term $P(D \mid \theta)$ appearing above is referred to as the likelihood. Using this terminology one is able to write

$$\text{Posterior} \propto \text{Prior} \times \text{Likelihood}.$$

4.2.1 The Dirichlet-Categorical Distribution

The Dirichlet distribution is a probability distribution defined on the $K-1$ dimensional simplex $\triangle$. $\pi \in \triangle \implies \pi_k > 0$, $\sum_k \pi_k = 1$. It is useful to conceptualize the Dirichlet distribution as a distribution over parameters of Categorical or Multinomial distributions. From this point of view samples from a Dirichlet distribution are themselves probability distributions.

If $\pi \sim \text{Dir}(\alpha)$ then

$$P(\pi) = \frac{1}{\mathbf{B}(\alpha)} \prod_k \pi_k^{\alpha_k - 1}, \tag{4.2}$$

where $\mathbf{B}$ is the multivariate beta function,

$$\mathbf{B}(\alpha) = \frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k)}, \tag{4.3}$$

and Γ is the Gamma function.

Let $D = \{x_1, \ldots, x_m\}$ and $n_k = \sum_{x \in D} \mathbb{1}\{x = k\}$ count the number of occurrences of $k \in D$. Assuming

$$\begin{aligned} \pi &\sim \mathrm{Dir}(\alpha) \\ x_i &\sim \mathrm{Cat}(\pi),\ i = 1, \ldots, m. \end{aligned}$$

Then

$$\begin{aligned} P(\pi \mid D, \alpha) &= \frac{P(\pi \mid \alpha) P(D \mid \pi, \alpha)}{P(D)} \\ &\propto P(\pi \mid \alpha) P(D \mid \pi) \\ &= \frac{1}{\mathbf{B}(\alpha)} \prod_k \pi_k^{\alpha_k - 1} \prod_k \pi_k^{n_k} \\ &= \frac{1}{\mathbf{B}(\alpha)} \prod_k \pi_k^{n_k + \alpha_k - 1}, \end{aligned}$$

The Dirichlet distribution is referred to as the *conjugate* prior for the Categorical distribution since the posterior $P(\pi \mid D, \alpha)$ and prior $P(\pi \mid \alpha)$ have the same form [11]. The marginal likelihood $P(D \mid \alpha)$ can be obtained by integrating over π as follows.

$$\begin{aligned} P(D \mid \alpha) &= \int_{\triangle} P(\pi \mid \alpha) \prod_{x \in D} P(x \mid \pi) d\pi \\ &= \int_{\triangle} \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \pi_k^{\alpha_k - 1} \prod_k \pi_k^{n_k} d\pi \\ &= \frac{1}{\mathbf{B}(\alpha)} \int_{\triangle} \prod_k \pi_k^{n_k + \alpha_k - 1} d\pi. \end{aligned} \tag{4.4}$$

The term inside the integral in (4.4) is the unnormalized probability density function (pdf) of a Dirichlet distribution with parameters $n_k + \alpha_k$. Since (4.2) is a pdf it follows that

$$
\begin{aligned}
1 &= \int_{\triangle} \frac{1}{\mathbf{B}(\boldsymbol{n}+\alpha)} \prod_k \pi_k^{n_k+\alpha_k-1} d\pi \\
&= \frac{1}{\mathbf{B}(\boldsymbol{n}+\alpha)} \int_{\triangle} \prod_k \pi_k^{n_k+\alpha_k-1} d\pi \\
\implies \mathbf{B}(\boldsymbol{n}+\alpha) &= \int_{\triangle} \prod_k \pi_k^{n_k+\alpha-1} d\pi \qquad (4.5)
\end{aligned}
$$

where $\boldsymbol{n} = (n_1, \ldots, n_K)$ is the vector of counts defined above. Plugging (4.5) into (4.4) yields

$$
P(D \mid \alpha) = \frac{1}{\mathbf{B}(\alpha)} \int_{\triangle} \prod_k \pi_k^{n_k+\alpha_k-1} d\pi = \frac{\mathbf{B}(\boldsymbol{n}+\alpha)}{\mathbf{B}(\alpha)}. \qquad (4.6)
$$

The process of marginalizing over a parameter as above is common in Bayesian analysis. It is particularly helpful in the context of Gibbs sampling since reducing the number of parameters that need to be sampled frequently results in increased efficiency.

4.2.2 The Bayesian Hidden Markov Model

Since the initial and transition distributions of an HMM are Categorical distributions it is convenient, due to the above described conjugacy relation, to place Dirichlet priors with parameters α_0 and α_i over their parameters π_0 and π_i. Assuming each state's emission distribution is also Categorical with parameters θ_i, a Dirichlet prior with parameters α_i' may be placed over these parameters as well. This results in the following generative story.

$$
\begin{aligned}
\pi_0 &\sim \mathrm{Dir}(\alpha_0) \\
\pi_i &\sim \mathrm{Dir}(\alpha_i),\ i = 1, \ldots, |Y| \\
\theta_i &\sim \mathrm{Dir}(\alpha_i'),\ i = 1, \ldots, |Y| \\
y_1 &\sim \mathrm{Cat}(\pi_0) \\
y_{t+1} \mid y_t &\sim \mathrm{Cat}(\pi_{y_t}),\ t = 1, \ldots, T-1 \\
x_t \mid y_t &\sim \mathrm{Cat}(\theta_{y_t}),\ t = 1, \ldots, T.
\end{aligned}
$$

Let $(\mathcal{X}, \mathcal{Y}) = (\{x^1_{1:T_1}, \ldots, x^m_{1:T_m}\}, \{y^1_{1:T_1}, \ldots, y^m_{1:T_m}\})$ be a set of observation and corresponding state sequences, and

$$
\begin{aligned}
n_{0,i} &= \sum_{y \in \mathbf{y}} \mathbb{1}\{y_1 = i\} \\
n_{i,j} &= \sum_{y \in \mathbf{y}} \sum_{t} \mathbb{1}\{y_{t-1} = i \wedge y_t = j\} \\
n'_{i,o} &= \sum_{(x_{1:T}, y_{1:T}) \in (\mathcal{X}, \mathcal{Y})} \sum_{t} \mathbb{1}\{y_t = i \wedge x_t = o\}
\end{aligned}
$$

count the number of times the initial states is i, the number of transition from state i to j, and the number of times a particular observation o is observed in state i respectively. Then

$$
\begin{aligned}
P(\mathcal{X}, \mathcal{Y} \mid \Theta, \alpha) = P(\pi_0 \mid \alpha_0) \prod_i P(\pi_i \mid \alpha_i) \prod_i P(\theta_i \mid \alpha') \\
\times \prod_{(x_{1:T}, y_{1:T}) \in (\mathcal{X}, \mathcal{Y})} P(x_{1:T}, y_{1:T})
\end{aligned} \tag{4.7}
$$

where $\Theta = \{\pi_0, \pi_i, \theta_i\}$ and $\alpha = \{\alpha_0, \alpha_i, \alpha'_i\}$.

Substituting the equations for the of the priors into (4.7) and rewriting in terms of the counting variables gives,

$$
\begin{aligned}
P(\mathcal{X}, \mathcal{Y} \mid \Theta, \alpha) = \frac{1}{\mathbf{B}(\alpha_0)} \prod_i \pi_{0,i}^{n_{0,i} + \alpha_{0,i} - 1} \\
\times \prod_i \left(\frac{1}{\mathbf{B}(\alpha_i)} \prod_j \pi_{i,j}^{n_{i,j} + \alpha_{i,j} - 1} \right) \\
\times \prod_i \left(\frac{1}{\mathbf{B}(\alpha'_i)} \prod_v \pi_{i,j}^{n'_{i,v} + \alpha'_{i,v} - 1} \right).
\end{aligned} \tag{4.8}
$$

Repeatedly applying the same steps used to derive (4.5) allows one to write the marginal joint probability of $(\mathcal{X}, \mathcal{Y})$ in terms of hyperparameters and counts.

$$
P(\mathcal{X}, \mathcal{Y} \mid \alpha_0, \alpha_i, \alpha'_i) = \frac{\mathbf{B}((\boldsymbol{n}_0 + \alpha_0)}{\mathbf{B}(\alpha_0)} \prod_i \frac{\mathbf{B}(\boldsymbol{n}_i + \alpha_i}{\mathbf{B}(\alpha_i)} \frac{\mathbf{B}((\boldsymbol{n}'_i + \alpha'_i)}{\mathbf{B}(\alpha'_i)} \tag{4.9}
$$

where the counts have been extended to vectors as was done previously,

$$
\begin{aligned}
\boldsymbol{n}_0 &= (n_{0,1}, \ldots, n_{0,|Y|}) \\
\boldsymbol{n}_i &= (n_{i,1}, \ldots, n_{i,|Y|}),\ i = 1, \ldots, |Y| \\
\boldsymbol{n}'_i &= (n'_{i,1}, \ldots, n'_{i,|X|}),\ i = 1, \ldots, |Y|.
\end{aligned}
$$

From an algorithmic perspective (4.9) has a particularly convenient form. Given the counting variables the time required to compute the marginal likelihood does not depend on the number of sequences or their length, only the number of states and and possible observations.

4.3 A Block Collapsed Gibbs Sampler for Hidden Markov Models

This Section develops a collapsed Gibbs Sampler for $P(\mathcal{Y} \mid \mathcal{X}, \alpha)$. We do so using a block-wise approach, resampling an entire state sequence $y^i_{1:T}$ given $\mathcal{X}$, $\mathcal{Y}_{-i}, \alpha$, where $\mathcal{Y}_{-i}$ denotes $\mathcal{Y} \setminus \{y^i_{1:T}\}$. This sampler follows the approach described in [15] for Probabilistic Context Free Grammars (PCFGs), and utilized for HMMs in [10].

The primary difficulty in constructing such a sampler is due to marginal-

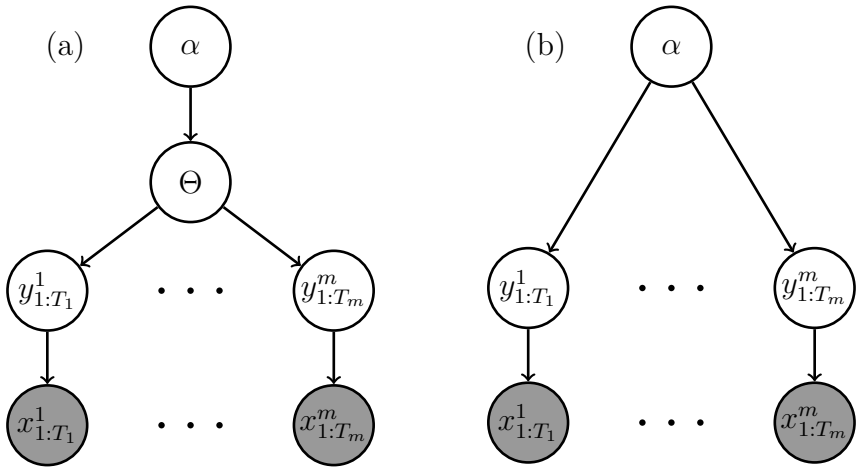


Figure 4.2: Graphical representation of a Bayesian HMM without (a) and with (b) marginalization of the parameters Θ. Notice that marginalization of Θ introduces dependencies between latent states y, since they now share a common parent α in (b).

izing over the parameters, as doing so introduces dependencies among the

latent state sequences $\mathcal{Y}$, Figure 4.2. Let us proceed in the usual manner for constructing such a Gibbs sampler. We have

$$P(y^i_{1:T} \mid \mathcal{Y}_{-i}, \mathcal{X}, \alpha) = \frac{P(x^i_{1:T} \mid y^i_{1:T}) P(y^i_{1:T} \mid \mathcal{Y}_{-i}, \mathcal{X}_{-i}, \alpha)}{P(x^i_{1:T} \mid \mathcal{Y}_{-i}, \mathcal{X}_{-i}, \alpha)}. \tag{4.10}$$

The $P(x^i_{1:T} \mid \mathcal{Y}_{-i}, \mathcal{X}_{-i}, \alpha)$ term in the denominator of (4.10) is particularly problematic, as noted in [15]. The need to marginalize over all possible state sequences (without reference to a set of HMM parameters) is particularly difficult, and it is not clear how one would approach such a problem in a computationally efficient manner.

Instead, since the denominator of (4.10) does not depend on $y^i_{1:T}$, we can appeal to a Metropolis-Hastings approach, knowing this troublesome term will cancel when computing the acceptance probability.

A Metropolis-Hastings sampler [1] is a popular Markov Chain Monte Carlo (MCMC) algorithm for drawing samples from a target distribution $P(X)$, provided one is able to compute a value $f(x)$ such that $f(x) \propto P(x)$. The sampler works by repeatedly drawing a sample x^* from a proposal distribution $q(\cdot \mid x)$ which is probabilistically accepted, replacing the current x, or rejected. The result is a series of samples whose stationary distribution is $P(X)$. In the special case that the proposal distribution does not depend on the current sample, $q(x^* \mid x) = q(x^*)$, the proposal is said to be *independent*. The generic Metropolis-Hastings algorithm is given in Algorithm 2.

Algorithm 2 The Metropolis-Hastings Algorithm.

Input: x^0 starting value

for $i \leftarrow 1, \ldots, n$ **do**
  $x^* \sim q(\cdot \mid x^{i-1})$
  $u \sim \text{Unif}(0, 1)$
  if $u < \dfrac{f(x^*) q(x^{i-1} \mid x^*)}{f(x^i) q(x^* \mid x^{i-1})}$ **then**
    $x^i \leftarrow x^*$
  else
    $x^i \leftarrow x^{i-1}$
  end if
end for
return $(x^1, \ldots, x^n)$

For the problem of sampling a state sequence $y_{1:T}$, the Metropolis-Hastings algorithm is utilized as a subroutine to sample from the conditional distribution

required by the Gibbs sampler. This is known as Metropolis-within-Gibbs [48]. Let $\mathcal{Y}^* = \{y_{1:T}\}^* \cup \mathcal{Y}_{-i}$, where $y^*_{1:T}$ is drawn from some an independent distribution $q(\cdot)$. Then the proposed $y^*_{1:T}$ is accepted with probability

$$\begin{aligned} p &= \frac{P(y^*_{1:T} \mid \mathcal{Y}_{-i}, \mathcal{X}, \alpha)}{P(y^i_{1:T} \mid \mathcal{Y}_{-i}, \mathcal{X}, \alpha)} \frac{q(y^i_{1:T})}{q(y^*_{1:T})} \\ &= \frac{P(\mathcal{Y}^*, \mathcal{X} \mid \alpha)}{P(\mathcal{Y}, \mathcal{X} \mid \alpha)} \frac{q(y^i_{1:T})}{q(y^i_{1:T}*)} \\ &= \frac{\mathbf{B}(\boldsymbol{n}^*_0 + \alpha_0)}{\mathbf{B}(\boldsymbol{n}_0 + \alpha_0)} \prod_j \left[\frac{\mathbf{B}(\boldsymbol{n}^*_j + \alpha_j)}{\mathbf{B}(\boldsymbol{n}_j + \alpha_j)} \frac{\mathbf{B}(\boldsymbol{n}'^*_j + \alpha'_j)}{\mathbf{B}(\boldsymbol{n}'_j + \alpha'_j)} \right] \frac{q(y^i_{1:T})}{q(y^*_{1:T})} \end{aligned} \tag{4.11}$$

where the $\boldsymbol{n}^*$ are analogous to the $\boldsymbol{n}$ terms, but contain the counts from the proposal $y^*_{1:T}$ rather than $y^i_{1:T}$.

The proposal distribution $q(\cdot)$ is itself a HMM with parameters $\hat{\Theta} = \{\hat{\pi}_0, \hat{\pi}_i, \hat{\theta}_i\}$ derived from the current $\mathcal{Y}_{-i}$, $\mathcal{X}_{-i}$. In particular $\hat{\Theta}$ is set to its expected value, $\hat{\Theta} = \mathrm{E}[\Theta \mid Y_{-i}, X_{-i}, \alpha]$, and then the proposal is sampled from

$$y^*_{1:T} \sim P(\cdot \mid x^i_{1:T}, \hat{\Theta}).$$

Empirically this strategy results in a majority of proposals $y^*_{1:T}$ being accepted, which aligns with the observations reported by [10].

To sample a latent state sequence the algorithm described in [41] is used. The algorithm takes as input an observation sequence $x_{1:T}$ and iteratively computes $p^t_i = P(Y_t = i \mid x_{1:t})$ from $P(y_{t-1} \mid x_{1:t-1})$ in a manner similar to the forward algorithm described in Section 4.1. However, along the way a series of matrices A^t are computed such that $a^t_{ij} = P(Y_{t-1} = i, Y_t = j, x_t \mid x_{1:t-1})$. Once p^T_i is computed the latent state sequence is sampled in reverse order starting by sampling $y_T \sim \mathrm{Cat}(p^T)$. Having obtained a value for y_t, the algorithm then samples $y_{t-1} \sim \mathrm{Cat}(b)$, where b is proportional to the y_t column of A^t. A complete description is given in Algorithm 3.

Algorithm 3 Sample States.

Input: Observation sequence $x_{1:T}$.
Output: A sequence of laent state $y_{1:T}$.

▷ Forward Traversal
$u_j^1 \leftarrow P(y_i)P(x_1 \mid y_i)$
$p_j^1 \leftarrow \dfrac{u_j^1}{\sum_i u_i^1}$
for $t \leftarrow 2, \ldots, T$ **do**
    $u_{ij}^t \leftarrow p_i^{t-1} P(j \mid i) P(x_t \mid j)$
    $a_{ij}^t \leftarrow \dfrac{u_{ij}^t}{\sum_{i'j'} u_{i'j'}^t}$
    $p_j^t \leftarrow \sum_i a_{ij}^t$
end for

▷ Backward Sampling
$y_T \sim \mathrm{Cat}(p^T)$
for $t \leftarrow T, \ldots, 2$ **do**
    $b_i \leftarrow \dfrac{a_{iy_t}^t}{\sum_{i'} a_{i'y_t}^t}$
    $y_{t-1} \sim \mathrm{Cat}(b)$
end for
return $y_{1:T}$

4.4 Mixtures of Hidden Markov Models

Extension to a finite mixture of K HMMs is straightforward and results in the following generative story

$$
\begin{aligned}
\phi &\sim \mathrm{Dir}(\beta) \\
\pi_{k,0} &\sim \mathrm{Dir}(\alpha_0), k = 1, \ldots, K \\
\pi_{k,i} &\sim \mathrm{Dir}(\alpha_i),\ k = 1, \ldots, K,\ i = 1, \ldots, |Y| \\
\theta_{k,i} &\sim \mathrm{Dir}(\alpha_i'),\ k = 1, \ldots, K,\ i = 1, \ldots, |Y| \\
z &\sim \mathrm{Cat}(\phi) \\
x_{1:T}, y_{1:T} \mid z &\sim \mathrm{HMM}(\Theta_z).
\end{aligned}
$$

where $\mathrm{HMM}(\Theta_z)$ is used as shorthand for the HMM generative story outlined in Section 4.1. Again, the marginal likelihood $\mathcal{Z}$, $\mathcal{Y}$, and $\mathcal{X}$ can be expressed in closed form in terms of hyper parameters and counts.

$$P(\mathcal{Z}, \mathcal{X}, \mathcal{Y} \mid \alpha, \beta) = \frac{\mathbf{B}(\boldsymbol{c}+\beta)}{\mathbf{B}(\beta)} \prod_k \left(\frac{\mathbf{B}(\boldsymbol{n}_{k,0}+\alpha_0)}{\mathbf{B}(\alpha_0)} \prod_i \left(\frac{\mathbf{B}(\boldsymbol{n}_{k,i}+\alpha_i)}{\mathbf{B}(\alpha_i)} \frac{\mathbf{B}(\boldsymbol{n}'_{k,i}+\alpha'_i)}{\mathbf{B}(\alpha'_i)} \right) \right). \quad (4.12)$$

where $c_k = \sum_{z \in \mathcal{Z}} \mathbb{1}\{z = k\}$ and $\boldsymbol{c} = (c_1, \ldots, c_K)$.

In many situations K is unknown. Estimation of K can be carried out using standard techniques such as cross validation or Bayesian Information Criterion, but this is often computationally difficult and methodologically unsatisfying. A modern approach to dealing with this problem is Bayesian nonparametrics [16]. In brief, Bayesian nonparametric techniques allow one to sidestep the problems associated with finite capacity models, by assuming observations are generated from infinite dimensional objects, for which only a finite number are involved in the generation of a finite set of samples.

In the case of mixture models, the above model can naturally be extended to handle a countably infinite number of cluster components using a Dirichlet Process [46]. The constructive definition of the Dirichlet process is given in terms of a *stick-breaking process* [42] with parameter $\beta \in \mathbb{R}_{>0}$, frequently denoted GEM(β).

$$\begin{aligned} \xi_k &\sim \text{Beta}(1, \beta), k = 1, \ldots, \infty \\ \phi_k &= \xi_k \prod_{l=1}^{k-1} (1 - \xi_l) \\ \implies \phi &\sim \text{GEM}(\beta) \end{aligned}$$

Extension of the finite mixture of HMMs to the nonparametric case thus assumes cluster assignments are sampled from an infinite dimensional ϕ, itself sampled from a GEM(β) distribution.

$$\begin{aligned} \phi &\sim \text{GEM}(\beta) \\ \pi_{k,0} &\sim \text{Dir}(\alpha_0), k = 1, \ldots, \infty \\ \pi_{k,i} &\sim \text{Dir}(\alpha_i),\ k = 1, \ldots, \infty,\ i = 1, \ldots, |Y| \\ \theta_{k,i} &\sim \text{Dir}(\alpha'_i),\ k = 1, \ldots, \infty,\ i = 1, \ldots, |Y| \\ z &\sim \phi \\ x_{1:T}, y_{1:T} \mid z &\sim \text{HMM}(\Theta_z). \end{aligned}$$

Note that for any finite set of $\{z_1, \ldots, z_m\} = \mathcal{Z}$, $\boldsymbol{c}$ contains at most m unique elements. In analogy to the finite mixture model, denote the number of unique elements K. Then the marginal likelihood of $\boldsymbol{c}$ given β has the following form [22]

$$P(\boldsymbol{c} \mid \beta) = \frac{m^K \Gamma(\beta)}{\Gamma(\beta + m)} \prod_k \Gamma(c_k). \tag{4.13}$$

Collapsing all parameters results in the following slight alteration of equation (4.12).

$$\begin{aligned} P(\mathcal{Z}, \mathcal{X}, \mathcal{Y} \mid \alpha) = &\left(\frac{m^K \Gamma(\beta)}{\Gamma(\beta + m)} \prod_k \Gamma(c_k) \right) \\ &\times \prod_k \left(\frac{\mathbf{B}(\boldsymbol{n}_{k,0} + \alpha_0)}{\mathbf{B}(\alpha_0)} \prod_i \left(\frac{\mathbf{B}(\boldsymbol{n}_{k,i} + \alpha_i)}{\mathbf{B}(\alpha_i)} \frac{\mathbf{B}(\boldsymbol{n}'_{k,i} + \alpha'_i)}{\mathbf{B}(\alpha'_i)} \right) \right). \end{aligned} \tag{4.14}$$

Having derived an expression for the the marginal likelihood of an infinite mixture of HMMs, the same generic Metropolis-within-Gibbs approach presented in Section 4.3 could be applied to carry out inference in the infinite mixture of HMMs.

However, collapsed Gibbs sampling for Dirichlet processes is a well studied problem for which numerous solutions exist [31]. In particular it would be easy to alternate between sampling z_i variables and latent state sequences $y^i_{1:T}$. Unfortunately this would likely mix poorly, since the sequence of latent states should be highly dependent on the cluster assignment. The ability to propose a new cluster and a new state sequence simultaneously avoids this problem, motivating the desire to blockwise sample in the combined space $(z_i, y^i_1, y^i_2, \ldots, y^i_T)$.

Luckily the marginal posterior predictive distribution of a Dirichlet process has a particularly convenient form [40] given by

$$P(z_i = k | \mathcal{Z}_{-i}) \propto \begin{cases} c_{k,-i} & k \in \mathcal{Z}_{-i} \\ \beta & k \notin \mathcal{Z}_{-i}. \end{cases}$$

Unlike proposing a latent state sequence, proposing z_i from this distribution is straightforward. Incorporating this into the proposal distribution means the terms related to the Dirichlet Process will cancel, and can be neglected when calculating the acceptance ratio of the Metropolis within Gibbs sampler.

Chapter 5

A Generative Model for Repetitive Sequences

A significant advantage of PGMs over popular discriminative Machine Learning methods such as SVMs, Random Forests, or Neural Networks, is the ability to easily incorporate structured knowledge via latent variables and conditional dependencies into the models [26, 35, 47]. This is true even in the case when the structure itself is unobserved, and must be learned in an unsupervised manner. This chapter describes a structured generative model for repetitive sequences. The model is a HMM with cyclically structured transitions which, for brevity, will be dubbed the Cyclic Hidden Markov Model (CHMM).

This model is specifically designed to reflect the structure of the repetitive sequences often encountered in activity recognition. While there have been several previous studies demonstrating the effectiveness of HMMs for recognizing activities from wearable sensors [25, 49], to date none have explicitly designed models to account for the repetitive nature of many activities. Repetition is a powerful inductive bias that can easily be incorporated into DBN models (such as HMMs). This is especially relevant in the exercise recognition domain as all activities considered are repetitive.

Using HMMs with structured transitions to model specific properties of sequences is not entirely novel. [36] use a similarly structured model to develop a high accuracy unsupervised grammar induction system.

5.1 Motivation

The structure of the model used is motivated by the observation that artificial sequences which resemble sensor reading during an exercise can be recognized by a Deterministic Finite State Automaton (DFA) with cyclic transition struc-

tures. Consider the oscillatory discrete time series depicted in Figure 5.1.

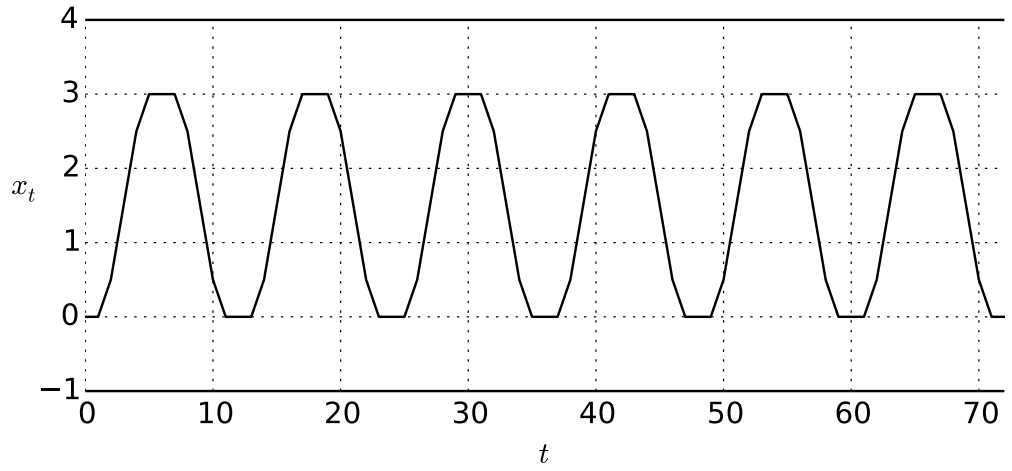


Figure 5.1: Example of a synthetic oscillatory discrete time series.

Taking the difference between successive values x_t and x_{t+1} as symbols in an alphabet, this sequence belongs to the regular language

$$\left[[+0]^* \left[+\tfrac{1}{2}\right]^+ [+1]^* \left[+\tfrac{1}{2}\right]^+ [+0]^* \left[-\tfrac{1}{2}\right]^+ [-1]^* \left[-\tfrac{1}{2}\right]^+ \right]^*,$$

which is recognized by the DFA in Figure 5.2.

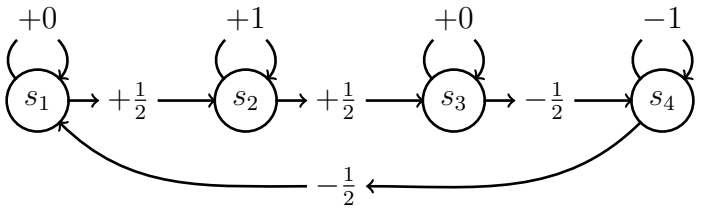


Figure 5.2: Cyclic DFA with four states.

The natural probabilistic generalization of this DFA is a HMM with a cyclic transition structure and parametric emission distributions replacing discrete symbols, Figure 5.3.

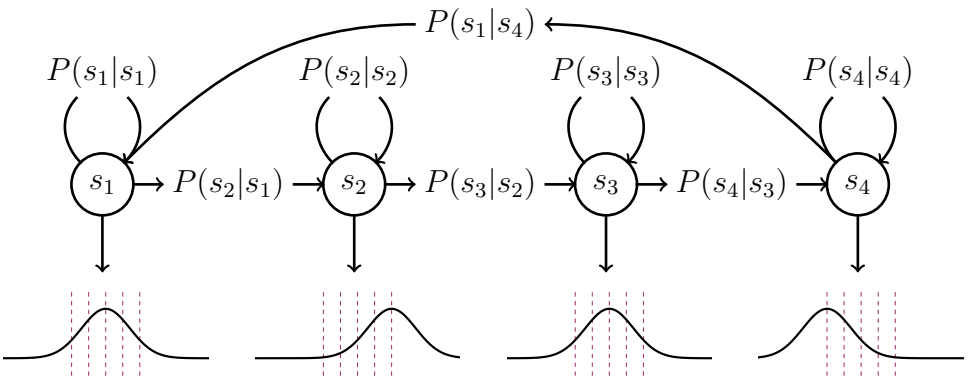


Figure 5.3: CHMM with four states.

5.2 Advantages

This basic, and surprisingly simple model is elaborated in various ways to solve a number of synthetic and real world tasks related to motion recognition in the following chapters. Not only does the CHMM inherently capture the repetitive nature of exercise, but due to the highly structured transition distribution a CHMM with K states only has $O(K)$ transition parameters[1]. This is unlike a HMM in which every state can transition to every other state which has $O(K^2)$ transition parameters. This reduction in parameters should require less data to estimate, and reduces the computational complexity of typical inference tasks such as filtering, smoothing, and decoding from $O(TK^2)$ to $O(TK)$.

[1] A CHMM with K states has exactly $2K$ transition parameters.

Chapter 6

Synthetic experiments with Hidden Markov Models

This chapter describes several synthetic experiments carried out utilizing the models and inference procedures developed in the previous Chapters. The focus on synthetic data is primarily intended as a demonstration and exploration of model aptitude. Experiments with real world data are presented in the following chapters.

6.1 The Trick Coin

The *trick coin* is a classic HMM problem as it is described exactly by the HMM generative story. In this scenario there are two coins, one fair, and the other biased to come up tails with significantly higher probability than heads. A sequence of flip outcomes is observed, however at each time step the flipper chooses to swaps coins, replacing the fair coin with the biased coin or vice versa with some probability. This scenario can be modeled by an HMM in which the latent states represent the coin being used, the observations the observed outcomes, and the state transitions the probability of swapping coins.

A single sequence of 500 flips is observed, to which the parameters of a HMM is fit using either the collapsed Gibbs sampler or the standard Expectation Maximization (EM) [1] estimation method.

The resulting HMM's performance was assessed on two tasks. 1.) Inferring the coin being used at each time of the training sequence, and 2.) Discriminating between an out of sample sequence of flips drawn from the two coin switching process or a single coin with the same expected value as the trick coin.

[1] Application of EM to HMMs is commonly also referred to as the Baum-Welch algorithm.

The point

For the first task, the HMM fit with the collapsed Gibbs sampling performed quite well, and the Maximum a Posteriori (MAP) state sequence closely aligned with the true state sequence. On the other hand the HMM fit with EM typically only utilized a single state and was therefore unable to distinguish between actual states in a better than random fashion, Figure 6.1.

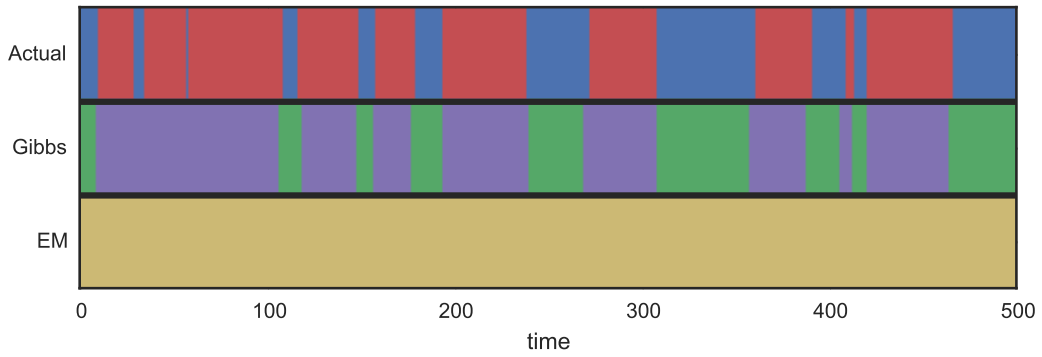


Figure 6.1: Actual state sequence and example inferred state sequences from the HMM fit with Gibbs sampling and EM. The HMM fit with EM only uses a single state to model the observed sequence.

For the second task 500 $(x_{1:T}^{\text{trick}}, x_{1:T}^{\text{single}})$ pairs of sequences were generated. The ability of each model to discriminate (by comparing likelihoods) is assessed for each pair. This same experiment was repeated 25 times using a different single training sequence and pairs of testing sequences. Again, the HMM fit with Gibbs sampling significantly outperformed the HMM fit with EM, Figure 6.2.

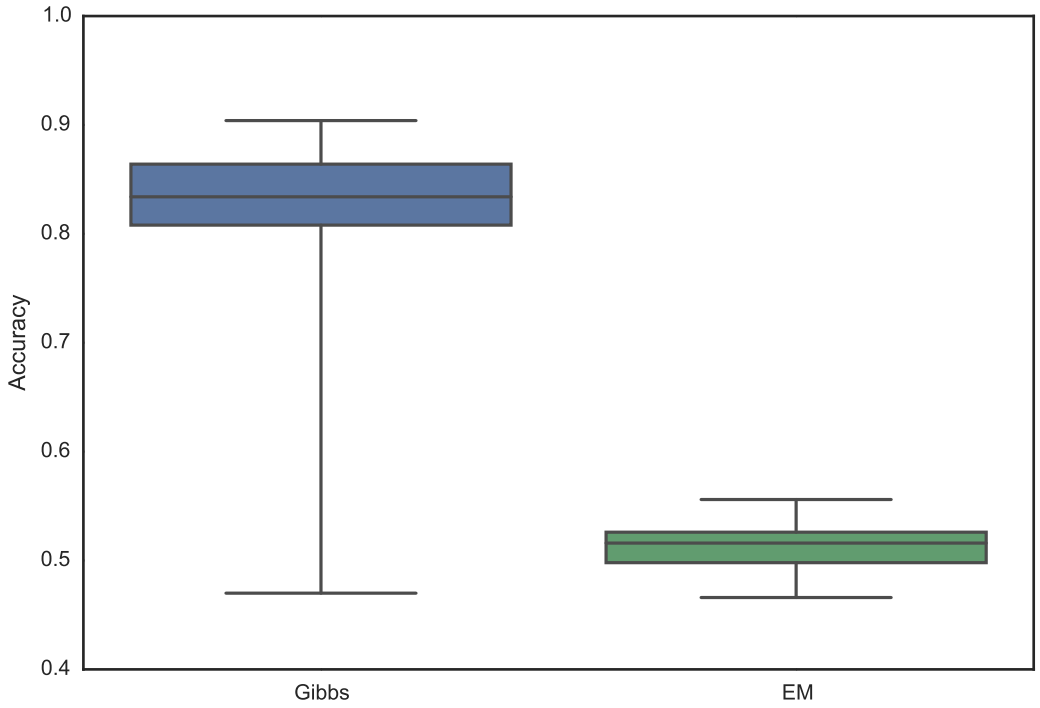


Figure 6.2: Accuracy for the trick coin discriminative task. Whiskers depict the maximum and minimum values.

6.2 Learning to Segment Repetitive Sequences

Having demonstrated the effectiveness of the collapsed Gibbs sampling approach, we now describe experiments which demonstrate the ability of the CHMM to utilize its repetitive latent structure in an artificial segmentation task. Sequences were generate by embedding a recurrent sequence between two constant sequences and adding Gaussian random noise. In particular an example sequence is generated from the following distribution.

$$
\begin{aligned}
t_1 &\sim 100 \cdot \text{Beta}(5,5) + 10 \\
t_2 &\sim 100 \cdot \text{Beta}(5,5) + 10 + t_1 \\
t_3 &\sim 100 \cdot \text{Beta}(5,5) + 10 + t_2 \\
\epsilon_t &\sim \text{Norm}(0, 1/2),\ t = 1, \ldots, t_3 \\
x_t &= \epsilon_t,\ t = 1, \ldots, t_1 \\
x_t &= \sin(t) + \epsilon_t,\ t = t_1 + 1, \ldots, t_2 \\
x_t &= \epsilon_t,\ t = t_2 + 1, \ldots, t_3
\end{aligned}
$$

A model for segmenting such sequences must distinguish between the recurrent portion, $x_{t_1+1:t_2}$, and the constant portions, $x_{1:t_1}$, $x_{t_2+1:t_3}$. More formally, each timestep x_t in the sequence is annotated with a tag $a_t \in \{I, O\}$ denoting whether the sequence is *in* (I), or *out* (O) of the repetitive portion of the sequence at time t. The annotation is similar to the BIO (begin/in/out) representation commonly used in Natural Language Processing (NLP) chunking applications [39].

Rather than using these annotations within the learning procedure (for example, to learn a discriminative classifier) an unsupervised approach is taken, in which only the raw sequence is available during training. Thus the ability to discover structure in the time series rests entirely on a priori structure built into the model itself.

To capture this structure the prior distribution over transition matrices (a Dirichlet distribution for each row) is split into three regime blocks. The first and third block corresponds to a standard HMM in which each state may transition to every other state, the *out* portions of the sequence. The second block corresponds to the CHMM structure in which states transition only to themselves or their successor states, the *in* portion of the sequence. To allow movement between these blocks, transitions are added from each state in the first block to the start state of the second block, and from the final state of the second block to each state in the third block. Diagrammatically this results in the structure depicted below.

$$\boldsymbol{\alpha} = \left[\begin{array}{c|c|c} F & A_1 & \mathbf{0} \\ \hline \mathbf{0} & S & A_2 \\ \hline \mathbf{0} & \mathbf{0} & F \end{array}\right]$$

$$F = \begin{bmatrix} \alpha_0 & \cdots & \alpha_0 \\ \vdots & & \vdots \\ \alpha_0 & \cdots & \alpha_0 \end{bmatrix} \qquad S = \begin{bmatrix} \alpha_3 & \alpha_4 & 0 & \cdots & 0 \\ 0 & \alpha_3 & \alpha_4 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_3 & \alpha_4 \\ \alpha_4 & 0 & \cdots & 0 & \alpha_3 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} \alpha_1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \alpha_1 & 0 & \cdots & 0 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \\ \alpha_2 & \cdots & \alpha_2 \end{bmatrix}$$

Here the α_i terms are parameters of a prior on transitions from

α_0 : an *out* state to an *out* state,
α_1 : an *out* state to an *in* state,
α_2 : an *in* state to an *out* state,
α_3 : an *in* state to itself,
α_4 : an *in* state to its successor *in* state.

Using just 2 sample sequences and the above described prior structure with 1 *out* state and 5 *in* states allows the collapsed Gibbs sampler to discover highly accurate segmentations in a completely unsupervised manner, Figure 6.3. To demonstrate the necessity for the prior, and rule out the possibility that the model is doing something uninteresting (such as modeling differences in variance within the distinct regimes) the same experiment was repeated with a standard HMM with two states. In this case, the model is completely unable to discover any interesting structure in the sequence, Figure 6.4.

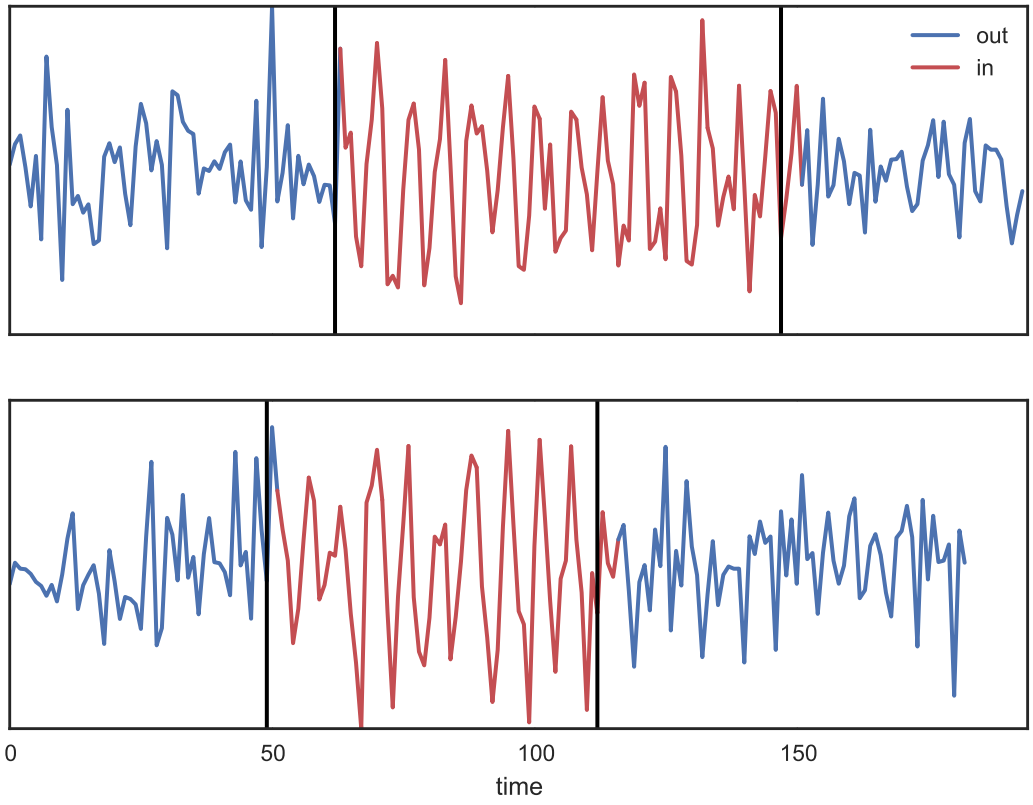


Figure 6.3: Unsupervised segmentation by a HMM with structured prior.

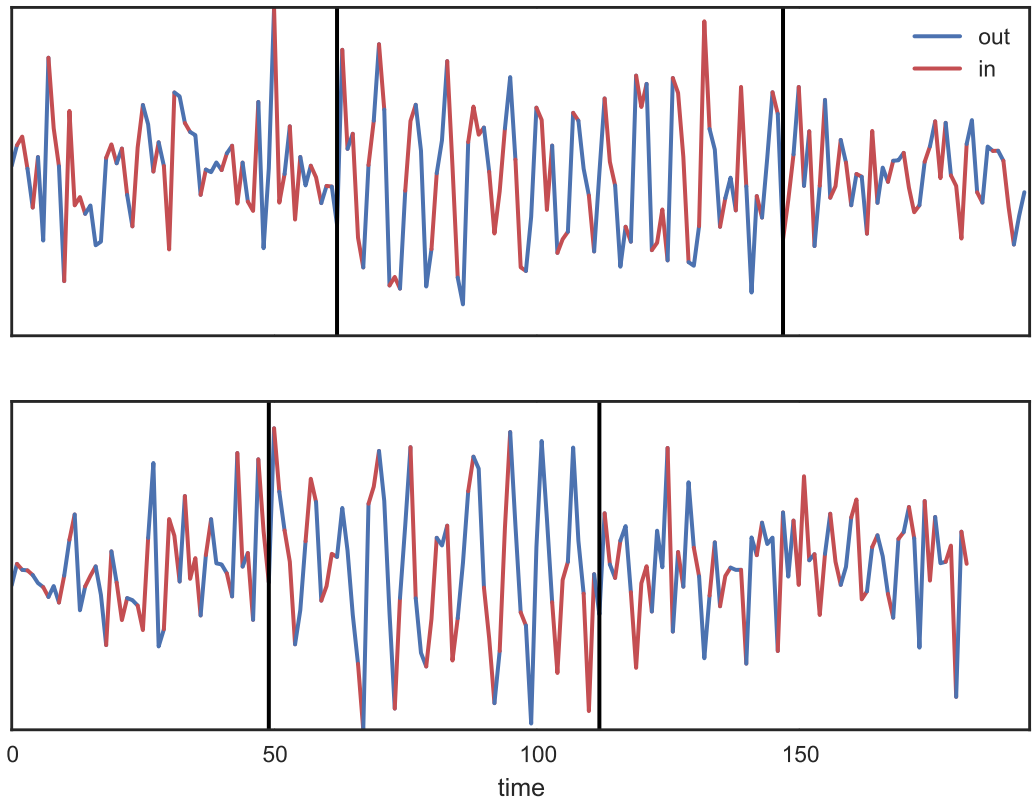


Figure 6.4: Unsupervised segmentation by a HMM with 2 states.

6.3 Clustering Sequences

The final synthetic experiment explores the Bayesian nonparametric approach outlined in Section 4.4 for the purpose of sequence clustering. As ground truth clusters the following four recurrent functions are used.

$$f_1(t) = \sin(t) \qquad f_3(t) = \max(-2, \min(2, \tan(t/4)))$$
$$f_2(t) = \sin(t) + \sin(t/2) \qquad f_4(t) = \max(0, 2\cos(t/4))$$

An example sequence is then generated from the following process.

$$\begin{aligned} t_0 &\sim \text{Unif}(0, \ldots, 200) \\ t_1 &\sim \text{Geom}(1/10) + 50 + t_0 \\ \epsilon_t &\sim \text{Norm}(0, 1/2) \\ x_t &= f_k(t) + \epsilon_t, \; t = t_0, \ldots, t_1. \end{aligned}$$

This procedure is repeated 10 times for each $k \in \{1, 2, 3, 4\}$ resulting in 40 total training sequences.

Two clustering approaches are considered. The Dirichlet Process Mixture of Hidden Markov Models (DPMoHMM) utilizes standard HMMs as the base distribution and the Dirichlet Process Mixture of Cyclic Hidden Markov Models (DPMoCHMM) uses CHMMs as the base distribution. The experiment is repeated 10 times and the resulting clusterings are compared using Adjusted Mutual Information (AMI) [50], a version of Mutual Information corrected to adjust for chance agreements. Figure 6.6 shows that as well as increased performance, in terms of AMI, the DPMoCHMM finds the correct number of clusters in 80% of experiments. This is in contrast to the DPMoHMM, which is unable to find the actual number of clusters in any of the experiments.

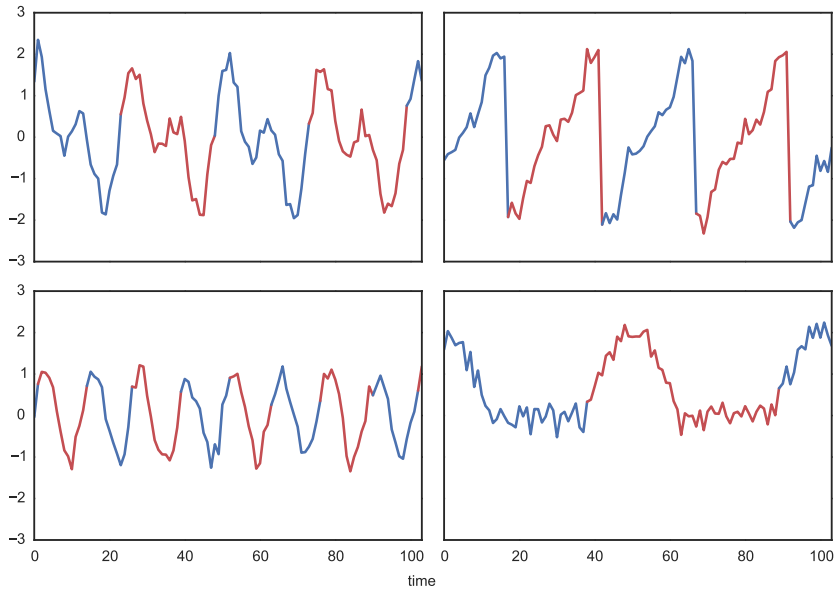


Figure 6.5: Example from each cluster and recurrent structure discovered by the DPMoCHMM.

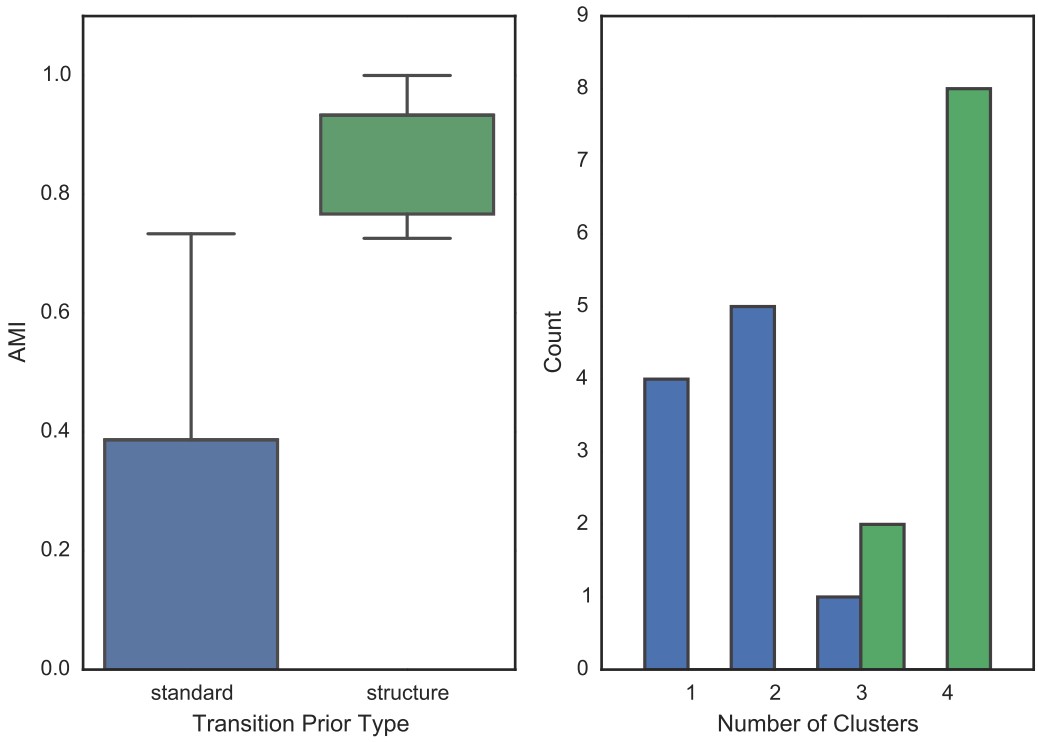


Figure 6.6: Results from clustering artificial sequences. Comparison of Adjusted mutual information and number of discovered clusters with (green) and without (blue) structured transition prior.

Chapter 7

Segmented Exercise Classification

An obvious precursor to classifying each timestep of an entire routine is to classify the hand segmented exercises into 1 of K mutually exclusive exercise classes. This task is significantly easier than than the full routine classification task since several simplifying assumptions can be made. First, we can assume each sequence belongs to a single class. Second, we can assume each class is approximately equally likely a priori and avoid dealing with the class imbalance problem that arises when working with full routines, which are overwhelmingly dominated by *rest*.

7.1 Setup

For training and evaluation purposes we used a random train/test split of the available data (approximately 70% of the data was used for training and the remaining 30% was used for testing), randomized over users. Keeping the set of users in these sets disjoint is particularly important as it mimics the situation one would actually encounter if deploying such a system. Classifier hyperparameters were set using grid search performed on an analogous split of the training data. Table 7.1 summarizes the splits.

	Train	**Test**
Users	85	37
Classes	50	50
Activities	3364	1382
Repetitions	28476	12396
Minutes of Data	1247	557

Table 7.1: Summary of segment classification dataset.

7.2 Models

The proposed CHMM was compared to a Structured Perceptron [7] and a strong performing system heavily influenced from the activity recognition literature [28]. As only sequences with a single label are considered in this experiment, a voting procedure is used to aggregate predictions for individual time steps to produce the final classification for both the Random Forest and Structured Perceptron. Note that this gives an unfair advantage to these models. In a real world activity classification setting one would not a priori have knowledge of the bounds (segmentation) over which to aggregate votes.

7.2.1 Baseline from Literature

A classifier inspired by [28] (discussed in Section 2.5) was developed to provide a strong performing baseline. Significantly better results were obtained on the data considered here using a Random Forest rather than a SVM, and only the performance of the former is reported. This discrepancy is likely due to [28] utilizing a number of heuristic feature partitioning schemes which were not able to be replicated in this work.

The following features were computed over 4 second windows (60 timesteps) for each of the 3 accelerometer and 3 gyroscope axis:

- The number of peaks in the autocorrelation function.
- The height of the first peak of the autocorrelation function after crossing zero.
- The maximum value of the autocorrelation function.
- The mean of the signal.
- The standard deviation of the signal.
- The norm of the signal.
- The magnitude of the power spectrum summed across 10 linearly spaced bands.

The free software scikit-learn [34] was used to estimate parameters of the Random Forest. Grid search was performed over the following hyper-parameters:[1]

- The number of trees in the forest: $(25, 50, \mathbf{100})$.
- The minimum number of samples in each leaf node: $(2, \mathbf{5}, 10)$.
- The maximum depth of each tree: $(10, 20, \mathbf{none})$.
- The maximum number of features considered for each split: $(\sqrt{\mathbf{d}}, all)$.
- The spit evaluation function: **Gini impurity** or information gain.

[1] Best performing settings on the validation set are noted in bold.

7.2.2 Structured Perceptron

The Structured Perceptron is a discriminative analog of a HMM. It is worth noting this claim is somewhat tenuous, and it might be considered more appropriate to say this is only true of the closely related Conditional Random Field (CRF), which models $P(Y|X)$ as opposed to a HMM which models $P(Y, X)$. Regardless, both are chain structured discriminative models which avoid the local normalization problems, *label bias*, associated with HMMs and Maximum Entropy Markov Models (MEMMs) [23].

Despite avoiding the label bias problem, the gradient based algorithms frequently used for training such models do not easily extend to models with latent variables. As latent variables offer a significant amount of flexibility this is a potential drawback for such models.

CRFSuite [32], a free structured linear model implementation, was used for training the Structured Perceptron. CRFs and Structured Perceptrons with Passive Aggressive weight updates were also evaluated for several settings of hyper-parameters. The Averaged Structured Perceptron obtained lower overall error on the data considered here. The Averaged Structured Perceptron has no hyperparameters.

7.2.3 Mixtures of Cyclic Hidden Markov Models

The conditional probability of a sequence belonging to one of the K exercise classes can be modeled by forming a mixture of CHMMs and calculating the predictive distribution as

$$P(k \mid x_{1:T}, y_{1:T}) \propto P(k)P(x_{1:T}, \mid k)$$

where $P(k)$ is prior probability of exercises k. In these experiments, $P(k) = 1/K$ is fixed. Given labeled segments, parameter estimation of the component CHMMs can be carried out independently. Two methods of parameter estimation are considered.

The first method searches for MAP parameters using Viterbi Training [17]. Viterbi Training seeks model parameters which maximize the probabilities of the most likely latent sequences given the observed sequences, rather than parameters that maximize the probability of the observed sequences as is done in classical EM. This criteria was found to perform slightly better than the typical EM approach for estimating HMM parameters. Recent work provides both empirical and theoretical support for this observation, especially when fitting highly misspecified models, and/or in cases in which the final performance

will be measured using the MAP state sequence. For further discussion on this subject we defer the interested reader to Sections 7 and 8 of [44] as well as [51].

The second method is based on the collapsed block Gibbs sampling approach outlined in Section 4.3. The MAP sampled assignment is used to produce a point estimate of the parameters.

7.3 Results

Table 7.2 gives train and test set accuracy numbers for the considered classification methods. In general the Random Forest outperforms the other models in terms of raw accuracy, although the Mixture of Cyclic Hidden Markov Models (MoCHMM) is quite competitive. The Structured Perceptron significantly under performs the other models, likely due to being restricted to linear decision boundaries as noted above.

The same experiments as above were repeated for the Random Forest and MoCHMM fit with the collapsed Gibbs sampler, but this time including a number of *rest* sequences equal the most frequent activity. While the accuracy of the Random Forest degrades in this regime, surprisingly the accuracy of the MoCHMM increases. Inspecting performance for just the *rest* class reveals this is at least in some part due to the MoCHMM doing a better job of modeling the *rest* class, the last two columns of Table 7.3. The lower recall number for the Random Forest imply that approximately 25% of the *rest* sequences are misclassified as an activity.

Classifier	**Train Accuracy**	**Test Accuracy**
MAP Baseline	0.043	0.037
Structured Perceptron	0.681	0.598
Random Forest	0.997	**0.827**
MoCHMM (Gibbs)	0.848	0.775
MoCHMM (Viterbi)	0.837	0.762

Table 7.2: Segment classification performance of different classifiers.

Another important observation is the significant performance degradation of the Random Forest on unseen data. Performance of the Random Forest decreases by 17% and 18.5% on unseen data for the two experiments described here, a significant amount compared to the more mild 8% and 4% percent decreases obtained by the Mixture of Hidden Markov Models (MoHMM). Several attempts were made to combat overfitting in the Random Forest, largely by

Classifier	Train Acc.	Test Acc.	F1	Precision	Recall
Random Forest	0.995	**0.81**	0.789	**0.925**	0.755
MoCHMM (Gibbs)	0.829	0.799	0.790	0.81	**0.829**

Table 7.3: Segment classification performance including the *rest* class. The F1-score is reported as a macro (unweighted) average of the individual classes. Precision and recall are for *rest* only.

reducing the complexity of the individual ensemble members, however in all cases this resulted in decreased test performance.

Chapter 8

Segmenting Exercises

The strong performing baseline classifier from Chapter 8 has the disadvantage that it requires relatively tight ground truth segmentations. Significant performance degradation was observed when many examples had large portions of *rest* leading and/or trailing the activity. Given the inherent non-sequential treatment of the input, this observation should not be surprising. Analogous behavior would be observed in a independent and identically distributed (iid) classification setting if many negative samples (*rest*) were incorrectly labeled positively (*exercise*). In contrast the CHMM can perform this segmentation in an unsupervised manner simply by inspecting the inferred latent states.

8.1 Unsupervised Segmentation

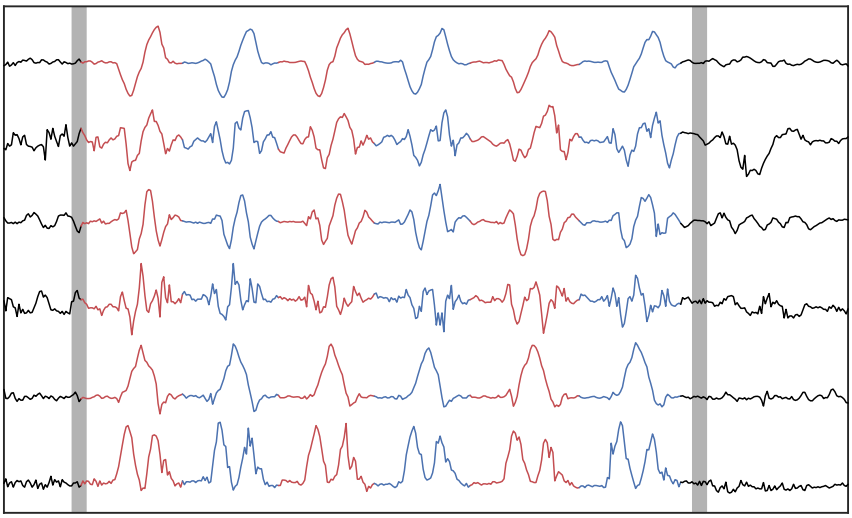
The system described in Section 6.2 was applied unmodified to the task of segmenting exercise boundaries. For each class, the parameters of a single CHMM were estimated using all available data for that class. The resulting parameters were then used to infer the MAP latent state sequences $y_{1:T}$ for each sequence.

Using the terminology introduced previously, let $\mathcal{O},\ \mathcal{I} \subseteq Y$ denote the set of *out* states and *in* states respectively. Then the inferred start and stop times of the activity are given by

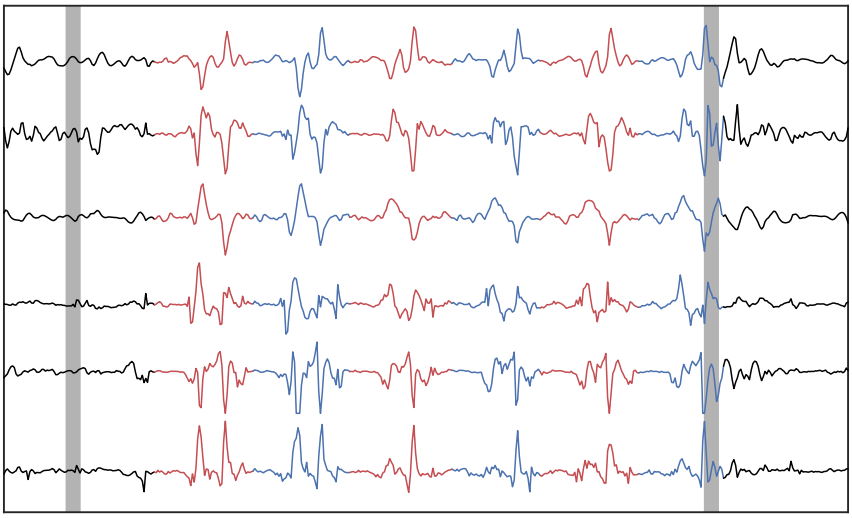
$$t_{\text{start}} = \min_t \{y_t \in \mathcal{I}\}$$
$$t_{\text{stop}} = \min_t \{y_{t-1} \in \mathcal{I} \wedge y_t \in \mathcal{O}\}.$$

Unfortunately, as the only annotations available are the ones which are to be corrected, it is difficult to define a performance metric to quantitatively

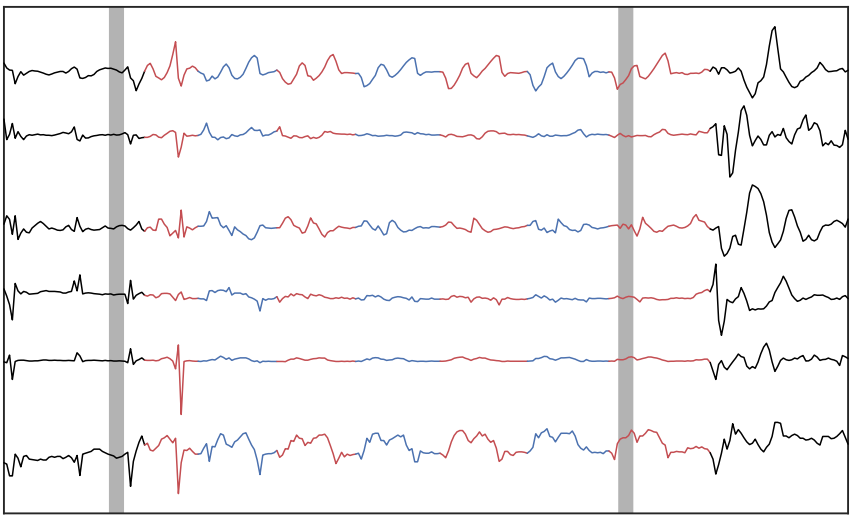
evaluate the resulting segmentations. Instead, visual inspection of a random subset of exercises was used to qualitatively assess model performance.



(a) Alternating Dumbbell Bicep Curls.



(b) Burpees.



(c) Push-Ups.

Figure 8.1: Visualization of segmentations produced by the CHMM. Large grey bars denote observer annotations. Parts of the sequence drawn in black correspond to *out* predictions, and parts drawn in color correspond to *in* predictions. Red and blue highlight the repetitive structure discovered by the algorithm. The ability of the CHMM to correct erroneous annotations is clearly visible in (b) and (c).

8.2 Results

For a majority of inspected sequences the algorithm was able to produce nearly equivalent or superior segmentations when compared to those produced by the observer. Figure 8.1 visually contrasts the segmentations produced by the CHMM and those produced by the exercise observer. the ability of the CHMM to produce superior annotations is particularly prevelant in Figures 8.1b and 8.1c.

The ability to "clean up" erroneous or noisy segmentations puts the CHMM in a unique position. As well as being able to form the basis of a competitive classifier, it can also serve as a valuable preprocessing tool to produce more accurate examples in classification settings where discriminative models ultimately prevail.

Chapter 9

Clustering Exercise Types

The experiments presented in Chapter 7 and Chapter 8 assume labels for the type of exercises being performed are given during the parameter estimation phase. This Chapter explores the scenario when the type of exercise being performed, and number of unique types is unknown.

Given these assumptions, a natural task is to attempt to partition the set of individual sequences into clusters. Ideally these clusters would correspond to the types of exercise being performed. Since the number of unique types is also unknown the Bayesian nonparametric approach described in Section 4.4, and utilized to cluster artificial data in Section 6.3, is used. To assess the value of the CHMM structure experiments are carried out using either a CHMM or HMM base distribution.

9.1 Experiment Setup

The ability of the DPMoCHMM and DPMoHMM to cluster exercises is assessed using three and nine exercises classes. To measure the effect of dataset size the number of exercise sequences used to estimate model parameters is varied as well. For the experiments carried out with 3 classes 10, 20, 40, 60, and 100 example sequences from each class are used, resulting in 30, 60, 120, 180, and 262 total samples[1]. For the experiments carried out with 9 classes 10 and 90 example sequences from each class are used, resulting in 90 and 360 total samples.

For each experimental setup (number of classes and number of sample sequences) 15 models of each type were fit to allow averaging performance metrics and assess model stability. The parameters of each model are fit using

[1]For some exercise we did not have 100 samples, resulting in 262 samples rather than 300

a different randomly selected subset of the available data, although the same random subsets are used for both the DPMoHMM and DPMoCHMM to enable staightforward comparison. Table 9.1 lists the exercises used.

Classes	Exercises
3	burpees, sit-ups, dumbbell lateral raises
9	dumbbell tricep extension, jumping jacks, burpees, sit-ups, dumbbell lateral raises, box jumps, alternating dumbbell bicep curls, push-ups, pull-ups

Table 9.1: Exercises used in clustering experiments.

9.2 Results

In both the three and nine class experiments the DPMoCHMM is able to produce fairly accurate clusterings as measured by AMI, and in all cases outperforms the DPMoHMM, Table 9.2. This provides further support for the benefit of incorporating prior structure into the model. There also appears to be a much stronger relationship between AMI and the complete data log likelihood for the DPMoCHMM than the DPMoHMM, Figure 9.2. This is especially important since one would not have access to the label required to compute AMI in a real world clustering scenario, and alternate measures of performance, such as likelihood, would need to be used to evaluate model performance.

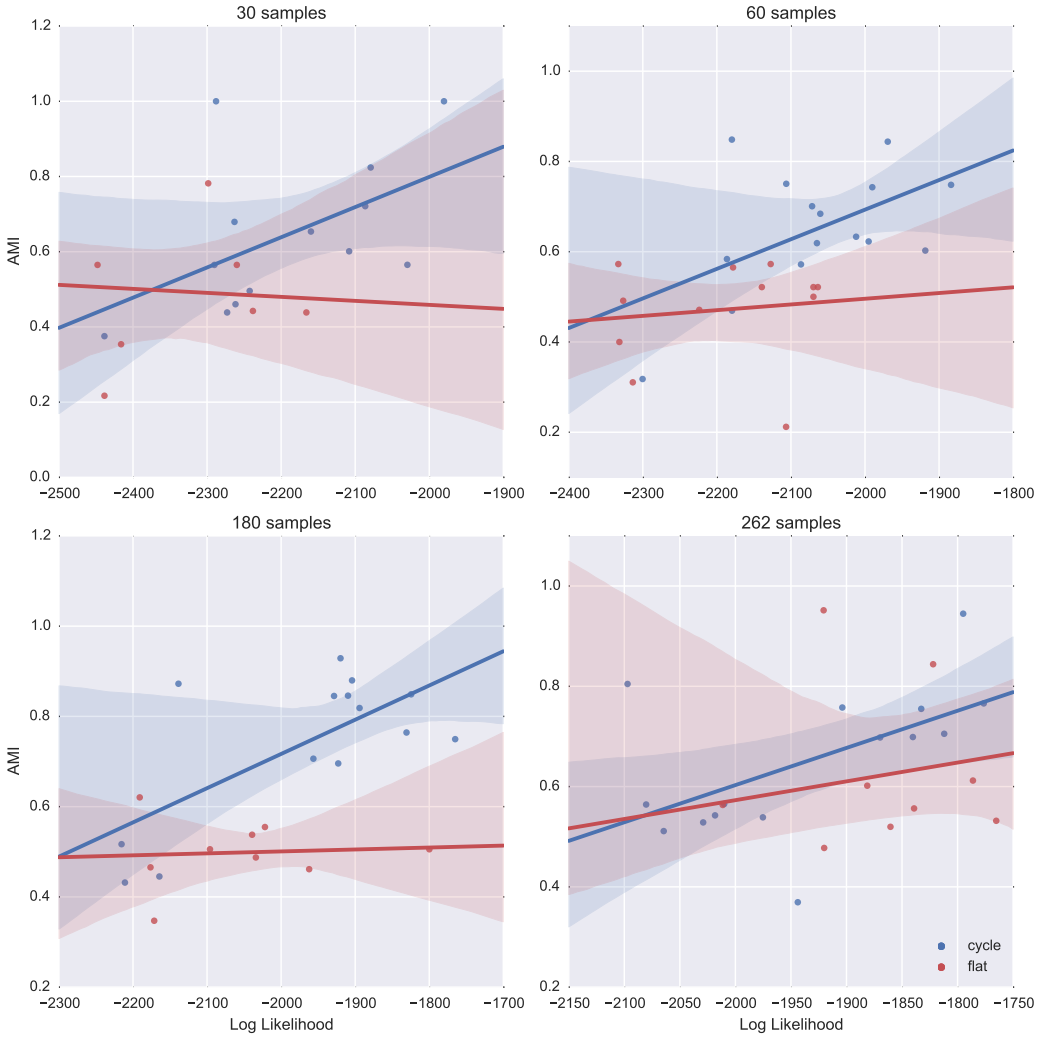
Classes	Size	Flat			Cycle		
		AMI	Fail	Success	AMI	Fail	Success
3	30	0.30 ± 0.27	6	2	0.56 ± 0.28	2	7
3	60	0.38 ± 0.21	3	0	0.65 ± 0.13	0	11
3	120	0.23 ± 0.25	8	0	0.59 ± 0.25	1	8
3	180	0.30 ± 0.25	6	1	0.69 ± 0.24	1	11
3	262	0.41 ± 0.31	5	4	0.65 ± 0.14	0	8
9	90	0.30 ± 0.11	0	0	0.49 ± 0.07	0	1
9	360	0.34 ± 0.13	0	0	0.62 ± 0.07	0	10

Table 9.2: Comparison of unsupervised clustering with HMM and CHMM base distributions. Reported AMI is the mean $\pm$ 1 standard deviation. The fail and success columns count the number of runs (of 15) resulting in an AMI less than 0.1 and greater than 0.6 respectively. The model using CHMM base distributions consistently has higher performance overall, more successful runs, and is less prone to failure in the 3 class case.

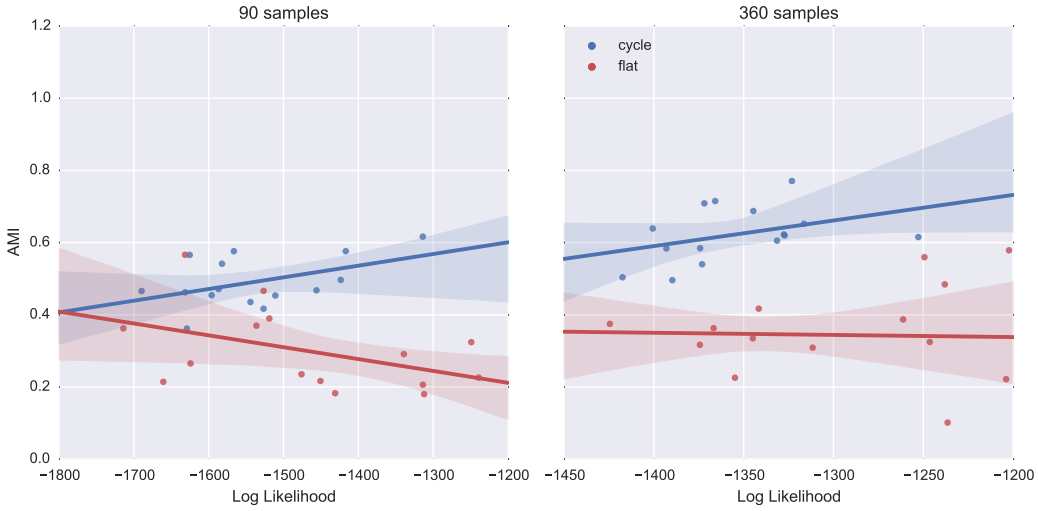
As well as AMI the resulting clusters were also assessed in terms of supervised accuracy. To do so each cluster was assigned the label of the most frequently appearing exercise class within that cluster. The resulting confusion matrices and raw accuracies are given in Figure 9.4. Although such numbers are not meaningful on their own, putting every sample in its own cluster would result in 100% accuracy, Figure 9.3 shows that this is not the case in the experiments presented here.

Although the results presented here are quite positive, there are several ways in which the DPMoCHMM does not perform adequately. The first, and most disheartening, is largely a failure of the sampler. Occasionally the sampler would get stuck in poor initial clusterings from which it was unable to escape. Typically this resulted in all sequences being assigned a single cluster. This is not entirely surprising, considering that the sampler is unable to move multiple sequences in a single step. An attempt to approximately count the number of such failures are given in the *Fail* column in Table 9.2.

The second noteable point of failure is the inability of the sampler to discover the correct number of true exercises, especially in the nine class case, Figure 9.3. This is likely due to the mismatch between the actual distribution of clusters and the Dirichlet process. Replacing the Dirichlet process with other stochastic pocesses which allow more control over tail behavior, such as the Pitman-Yor process [45], is a promising direction for future research.

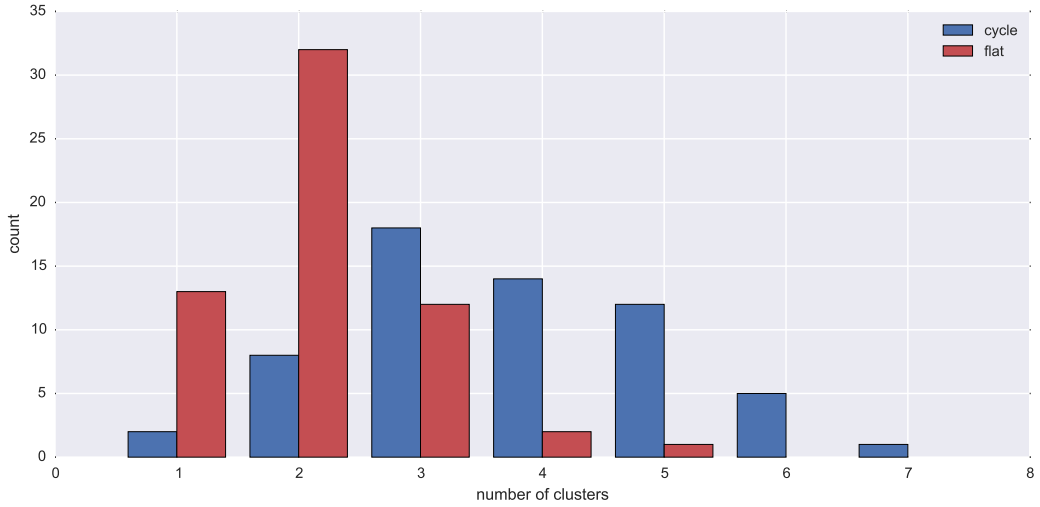


(a) Three Exercises

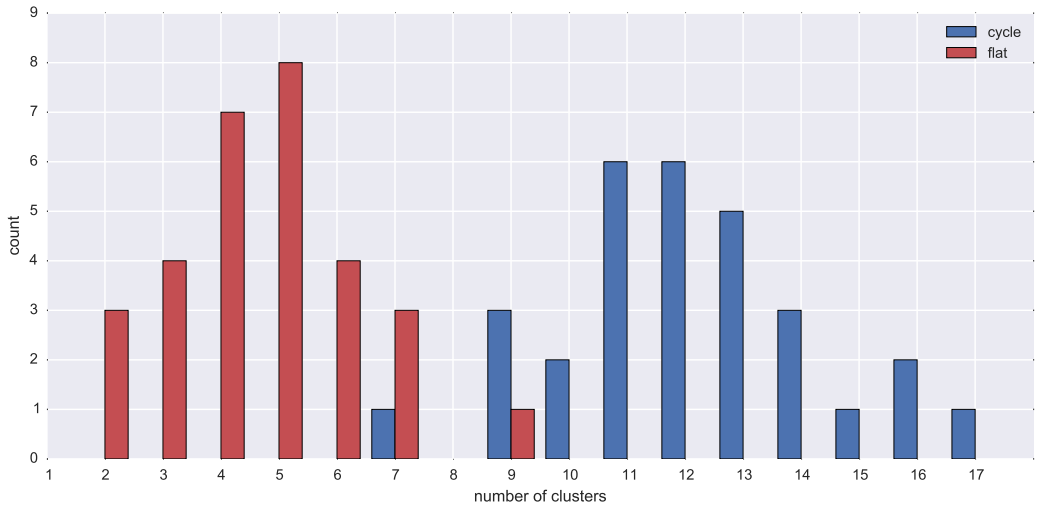


(a) Nine Exercises

Figure 9.2: Relationship between joint log likelihood and clustering performance as measured by AMI for various number of sample seqeunces. Solid lines give best linear fit to the observed data. Shaded regions depict 95% confidence intervals computed using 5,000 bootstrap resamples. As can be seen, there is a much stronger trend for higher log likelihood to result in increased performance as measured by AMI when using a CHMM (blue) base distribution than when using a standard HMM (red).

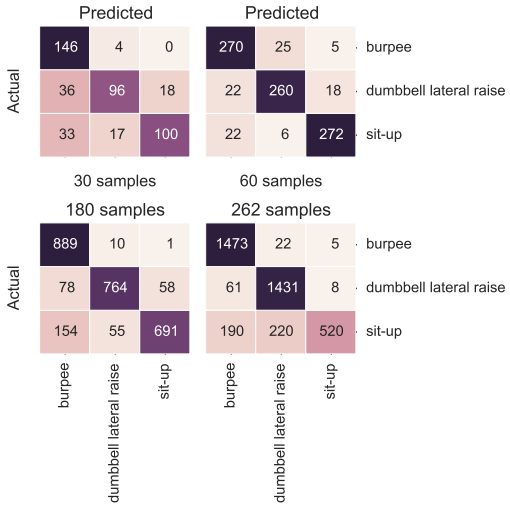


(a) Three Exercises.

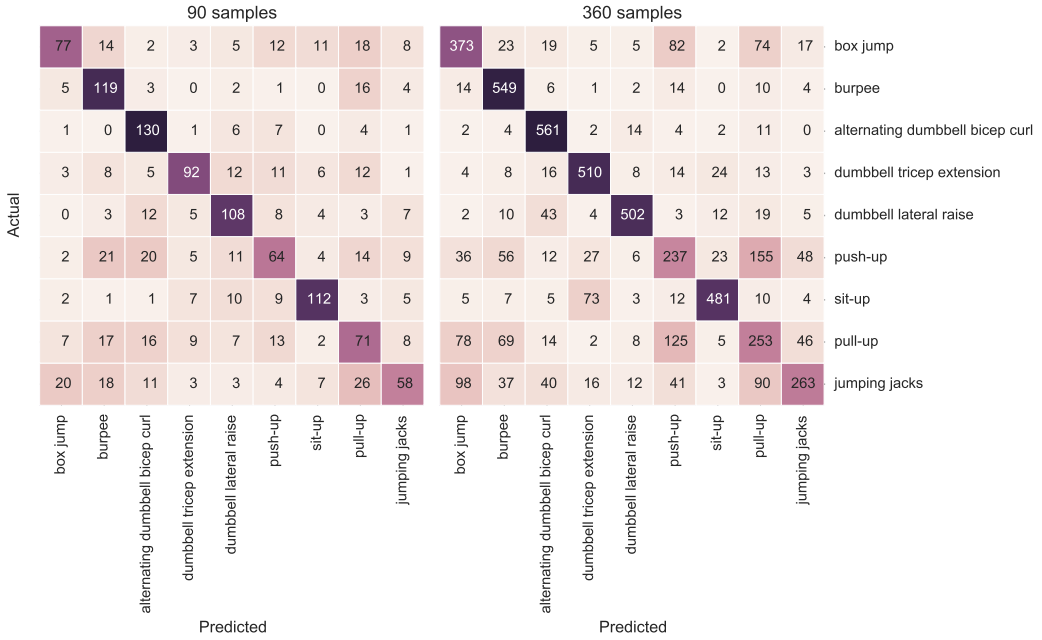


(b) Nine Exercises.

Figure 9.3: Histogram of the number of clusters discovered by the DPMoHMM (red) and DPMoCHMM (blue) when using 3 (a) and 9 (b) different exercises.



(a) Three Exercises.



(b) Nine Exercises.

Figure 9.4: Confusion matrices for the DPMoCHMM using 3 and 9 exercises classes. Each cluster is assigned the label of the most frequently appearing class. The resulting accuracies from upper left to lower right in (a) are 76%, 89%, 87%, and 87%, and from left to right in (b) are 61% and 69%.

Chapter 10

Conclusions and Future Directions

This thesis introduced the CHMM, a generative model specifically designed to capture repetitive structure in time series. In contrast to much work, the CHMM is a restriction, rather than an elaboration, of a popular and well known Machine Learning technique, the HMM. Restricting capacity in such a way that a model is encouraged to use its remaining capacity to discover structure was empirically shown to be a powerful technique through classification, segmentation, and clustering experiments with real world time series data. In Chapter 5 the CHMM is motivated by an automata theoretical analogy. The use of similar techniques to capture other types of structure and in other PGMs is an interesting direction for future research.

To cluster time series with an unknown number of types, Dirichlet process mixtures of HMMs were also introduced. A novel collapsed block Gibbs sampler was derived to support inference in this model. Due to working exclusively in the collapsed space, this sampler is able to simultaneously update the cluster assignment and descendent latent states associated with a particular example. This technique is likely to be useful in other mixture type models which feature a strong coupling between the mixture assignment and latent variables of the base distribution.

Despite performing well in the task considered here, the proposed sampler occasionally becomes stuck in poor modes when applied to Dirichlet process mixture models. The analysis carried out in Chapter 9 suggest several ways to overcome this limitation. Due the the success of most runs, methods featuring multiple independent chains starting from different random initializations are especially promising. This includes techniques such as parallel tempering [8] and tempered transitions [30], which are well suited to dealing with

poor mixing in multi-modal settings.

Bibliography

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[2] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *Pervasive computing*, pages 1–17. Springer, 2004.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] A.J. Bernheim Brush, John Krumm, and Scott James. Activity recognition research: The good, the bad, and the future. In *Proceedings of the Pervasive 2010 Workshop on How to do Good Research in Activity Recognition, Helsinki, Finland*, volume 1720, 2010.

[5] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33, 2014.

[6] Heng-Tze Cheng. *Learning and Recognizing The Hierarchical and Sequential Structure of Human Activities*. PhD thesis, Carnegie Mellon University, 2013.

[7] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, volume 10, pages 1–8. Association for Computational Linguistics, 2002.

[8] Radu V Craiu, Jeffrey Rosenthal, and Chao Yang. Learn from thy neighbor: Parallel-chain and regional adaptive mcmc. *Journal of the American Statistical Association*, 104(488):1454–1466, 2009.

[9] Thomas Fritz, Elaine M Huang Gail C Murphy, and Thomas Zimmermann. Persuasive technology in the real world: a study of long-term use

of activity sensing devices for fitness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 487–496. ACM, 2014.

[10] Jianfeng Gao and Mark Johnson. A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 344–352. Association for Computational Linguistics, 2008.

[11] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*, volume 2. Taylor & Francis, 2014.

[12] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 10–19. ACM, 2008.

[13] Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In *Proceedings of the International Conference on Artificial Intelligence*, 2000.

[14] Kyle L Jensen, Mark P Styczynski, Isidore Rigoutsos, and Gregory N Stephanopoulos. A generic motif discovery algorithm for sequential data. *Bioinformatics*, 22(1):21–28, 2006.

[15] Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. Bayesian inference for PCFGs via markov chain monte carlo. In *HLT-NAACL*, pages 139–146, 2007.

[16] Michael I Jordan. Bayesian nonparametric learning: Expressive priors for intelligent systems. *Heuristics, Probability and Causality: A Tribute to Judea Pearl*, 11:167–185, 2010.

[17] Biing-Hwang Juang and Lawrence Rabiner. The segmental k-means algorithm for estimating parameters of hidden markov models. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(9):1639–1641, 1990.

[18] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[19] Nicky Kern, Bernt Schiele, and Albrecht Schmidt. Multi-sensor activity context detection for wearable computing. In *Ambient Intelligence*, pages 220–232. Springer, 2003.

[20] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Lee, and Tae-Seong Kim. Accelerometers position independent physical activity recognition system for long-term activity monitoring in the elderly. *Medical & biological engineering & computing*, 48(12):1271–1279, 2010.

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

[22] Minjung Kyung, Jeff Gill, and George Casella. Estimation in dirichlet random effects models. *The Annals of Statistics*, 38(2):979–1009, 2010.

[23] John Lafferty, Andrew McCallum, and Fednando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[24] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *Communications Surveys & Tutorials, IEEE*, 15(3):1192–1209, 2013.

[25] Young-Seol Lee and Sung-Bae Cho. Activity recognition using hierarchical hidden markov models on a smartphone with 3d accelerometer. In *Hybrid Artificial Intelligent Systems*, pages 460–467. Springer, 2011.

[26] Vikash Mansinghka, Charles Kemp, Thomas Griffiths, and Joshua B Tenenbaum. Structured priors for structure learning. *arXiv preprint arXiv:1206.6852*, 2012.

[27] David Minnen, Thad Starner, Irfan Essa, and Charles Isbell. Discovering characteristic actions from on-body sensor data. In *Wearable computers, 2006 10th IEEE international symposium on*, pages 11–18. IEEE, 2006.

[28] Dan Morris, Scott T Saponas, Andrew Guillory, and Ilya Kelner. Recofit: using a wearable sensor to find, recognize, and count repetitive exercises. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3225–3234. ACM, 2014.

[29] Kevin Patrick Murphy. *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

[30] Radford M Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and computing*, 6(4):353–366, 1996.

[31] Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[32] Naoaki Okazaki. CRFsuite: a fast implementation of conditional random fields, 2007.

[33] Alexandros Pantelopoulos and Nikolaos G Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(1):1–12, 2010.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[35] Amy Perfors, Joshua B Tenenbaum, and Terry Regier. Poverty of the stimulus? A rational approach. In *In the Proceedings of the 2006 Cognitive Science conference. 2006*, pages 663–668, 2006.

[36] Elias Ponvert, Jason Baldridge, and Katrin Erk. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1077–1086. Association for Computational Linguistics, 2011.

[37] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[38] Lawrence Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[39] Lance A Ramshaw and Mitchell P Marcus. Text chunking using transformation-based learning. *arXiv preprint cmp-lg/9505040*, 1995.

[40] Carl Edward Rasmussen. The infinite Gaussian mixture model. In *NIPS*, volume 12, pages 554–560, 1999.

[41] Steven L Scott. Bayesian methods for hidden Markov models. *Journal of the American Statistical Association*, 97(457), 2002.

[42] Jayaram Sethuraman. A constructive definition of dirichlet priors. Technical report, DTIC Document, 1991.

[43] Matt Shannon and William Byrne. Autoregressive HMMs for speech synthesis. In *Interspeech*, pages 400–403, 2009.

[44] Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17. Association for Computational Linguistics, 2010.

[45] Yee Whye Teh. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics, 2006.

[46] Yee Whye Teh. Dirichlet process. In *Encyclopedia of machine learning*, pages 280–287. Springer, 2010.

[47] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285, 2011.

[48] Luke Tierney. Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728, 1994.

[49] Dung T Tran, Dinh Q Phung, Hung H Bui, and Svetha Venkatesh. Factored state-abstract hidden markov models for activity recognition using pervasive multi-modal sensors. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on*, pages 331–336. IEEE, 2005.

[50] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1073–1080. ACM, 2009.

[51] Martin J Wainwright. Estimating the wrong graphical model: Benefits in the computation-limited setting. *The Journal of Machine Learning Research*, 7:1829–1859, 2006.

[52] Jamie A Ward, Paul Lukowicz, Gerhard Troster, and Thad E Starner. Activity recognition of assembly tasks using body-worn microphones and accelerometers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1553–1567, 2006.

[53] Jianxin Wu, Adebola Osuntogun, Tanzeem Choudhury, Matthai Philipose, and James M Rehg. A scalable approach to activity recognition based on object use. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[54] Xu Zhang, Xiang Chen, Yun Li, Vuokko Lantz, Kongqiao Wang, and Jihai Yang. A framework for hand gesture recognition based on accelerometer and emg sensors. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(6):1064–1076, 2011.