Analyzing Repetitive Sequences with Structured Dynamic Bayesian Networks

Thom L. Lake
Western Michigan University
&
Atlas Wearables

2015.08.02

Disclaimer

Abstract

Abstract goes here

Dedication

To my wife Brooke Lake.

Declaration

I declare that this has taken too long

Acknowledgments

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

Contents

1 Introduction **8**
- 1.1 Related Work . . . 8
- 1.2 Exercise Classification . . . 8

2 Activity Data **9**
- 2.1 Hardware . . . 9
- 2.2 Data Collection Methodology . . . 10
- 2.3 Exercise Taxonomy . . . 10

3 Hidden Markov Models **12**
- 3.1 Notation . . . 12
- 3.2 Introduction to Hidden Markov Models . . . 13
  - 3.2.1 Inference . . . 14
- 3.3 The Bayesian Approach . . . 16
  - 3.3.1 The Dirichlet-Categorical Distribution . . . 16
  - 3.3.2 The Bayesian Hidden Markov Model . . . 18
- 3.4 A Block Collapsed Gibbs Sampler for Hidden Markov Models 20
- 3.5 Mixtures of Hidden Markov Models . . . 23

4 A Generative Model for Repetitive Sequences **26**
- 4.1 Motivation . . . 27
- 4.2 Advantages . . . 28

5 Synthetic experiments with Hidden Markov Model **29**
- 5.1 The Trick Coin . . . 29
- 5.2 Learning to Segment Repetitive Sequences . . . 31
- 5.3 Clustering Sequences . . . 35

6 Segmented Exercise Classification 38
6.1 Setup 38
6.2 Models 39
6.2.1 Baseline from Literature 39
6.2.2 Structured Perceptron 40
6.2.3 Mixtures of Cyclic Hidden Markov Models 41
6.3 Results 41

7 Segmenting Exercises 43

8 Simultaneous Segmentation and Classification of Exercises 45

Chapter 1

Introduction

Words words words

1.1 Related Work

TODO: This Section obviously needs work. Have references ([3], [17], [11], [27], [2], [32]) need to write.

1.2 Exercise Classification

Like this work, [16] apply Machine Learning methods to segment, classify, and count repetitions of exercises using an arm mounted accelerometer and gyroscope (their sensors were mounted on the forearm near the elbow rather than the wrist).

Chapter 2

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 uniformly as *rest* and denoted with the label $\varnothing$ where applicable, Table 2.1.

	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 2.1: Summary of exercise dataset.

2.1 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).

2.2 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 2.1 shows an example routine with exercise start time and stop time annotations.

2.3 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

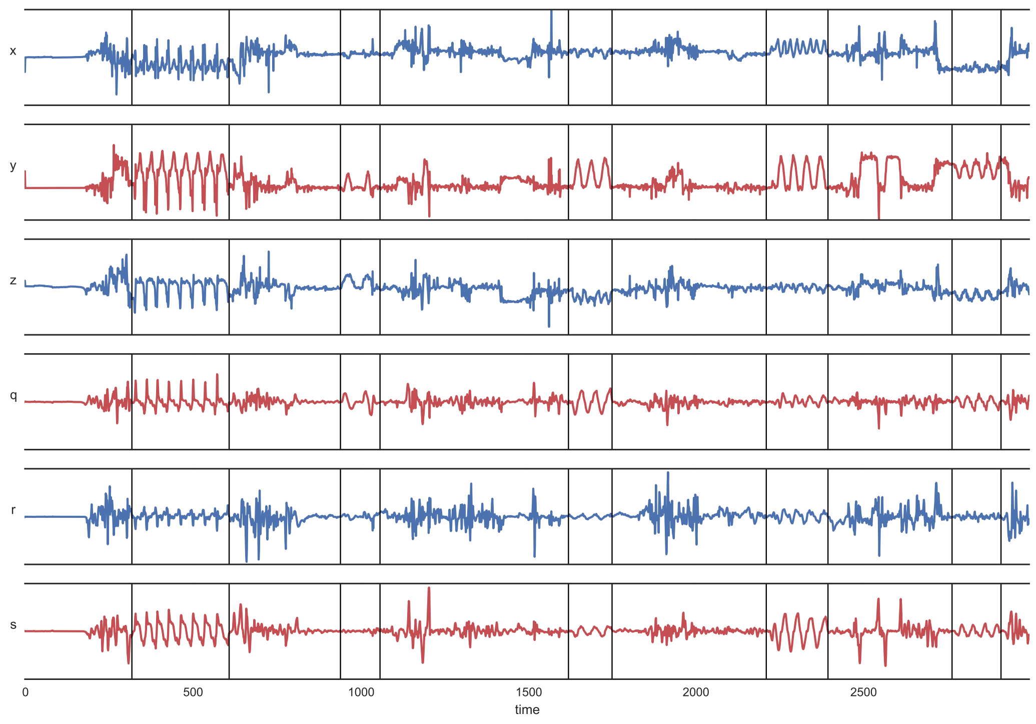

Figure 2.1: Example routine time series, vertically ordered by x, y, z (accelerometer, axis range from -3 to 3) and q, r, s (gyroscope, axis range from -600 to 600).

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

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

and

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

Chapter 3

Hidden Markov Models

Hidden Markov models (HMMs) [24] 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 [18], which also discusses the more general class of probabilistic models known as Dynamic Bayesian Networks (DBNs), of which HMMs and the popular Kalman filter [10], are a member.

3.1 Notation

Standard notation used throughout is reviewed here. 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 $x \sim \mathrm{D}(\theta)$. The Uniform, Bernoulli, Categorical, Dirichlet, and Normal distribution are denoted by Unif, Bern, Cat, Dir, and Norm respectively.

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 $x_{1:T}$.

3.2 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 [30]. 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

$$P(x_{1:T}, y_{1:T}) = P(y_1)P(x_1 \mid y_1) \prod_{t=2}^{T} P(y_t \mid y_{t-1})P(x_t \mid y_t). \tag{3.1}$$

The corresponding generative story is given by

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

where D is some parametric emission distribution. Dependencies can be visualized graphically in Figure 3.1, where vertices represent variables, edges represent dependencies, and grey vertices represent observed variables.

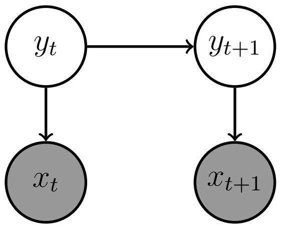

Figure 3.1: Slice representation of an HMM.

3.2.1 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 Probabilistic Graphical Models (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 (MAP) 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.

3.3 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

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

3.3.1 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 $\pi \sim \mathrm{Dir}(\alpha)$ then

$$P(\pi) = \frac{1}{\mathbf{B}(\alpha)} \prod_k \pi_k^{\alpha_k - 1}, \tag{3.2}$$

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

$$\mathbf{B}(\alpha) = \frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k)}, \tag{3.3}$$

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 [6]. The marginal likelihood $P(D \mid \alpha)$ can be obtained by integrating over π as follows.

$$
\begin{aligned}
P(D \mid \alpha) &= \int_{\triangle} P(\pi \mid \alpha) \prod_{x \in D} P(x \mid \pi) d\pi\\
&= \int_{\triangle} \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \pi_k^{\alpha_k - 1} \prod_k \pi_k^{n_k} d\pi\\
&= \frac{1}{\mathbf{B}(\alpha)} \int_{\triangle} \prod_k \pi_k^{n_k + \alpha_k - 1} d\pi. \qquad (3.4)
\end{aligned}
$$

The term inside the integral in (3.4) is the unnormalized probability density function of a Dirichlet distribution with parameters $n_k + \alpha_k$. Since (3.2) is a pdf it follows that

$$\begin{aligned} 1 &= \int_{\triangle} \frac{1}{\mathbf{B}(\boldsymbol{n}+\alpha)} \prod_k \pi_k^{n_k+\alpha_k-1} d\pi \\ &= \frac{1}{\mathbf{B}(\boldsymbol{n}+\alpha)} \int_{\triangle} \prod_k \pi_k^{n_k+\alpha_k-1} d\pi \\ \implies \mathbf{B}(\boldsymbol{n}+\alpha) &= \int_{\triangle} \prod_k \pi_k^{n_k+\alpha-1} d\pi \end{aligned} \tag{3.5}$$

where $\boldsymbol{n} = (n_1, \ldots, n_K)$ is the vector of counts defined above. Plugging (3.5) into (3.4) yields

$$P(D \mid \alpha) = \frac{1}{\mathbf{B}(\alpha)} \int_{\triangle} \prod_k \pi_k^{n_k+\alpha_k-1} d\pi = \frac{\mathbf{B}(\boldsymbol{n}+\alpha)}{\mathbf{B}(\alpha)}. \tag{3.6}$$

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.

3.3.2 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

$$\begin{aligned} P(\mathcal{X}, \mathcal{Y} \mid \Theta, \alpha) = P(\pi_0 \mid \alpha_0) \prod_i P(\pi_i \mid \alpha_i) \prod_i P(\theta_i \mid \alpha') \\ \times \prod_{(x_{1:T}, y_{1:T}) \in (\mathcal{X}, \mathcal{Y})} P(x_{1:T}, y_{1:T}) \end{aligned} \tag{3.7}$$

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 (3.7) and rewriting in terms of the counting variables gives,

$$\begin{aligned} P(\mathcal{X}, \mathcal{Y} \mid \Theta, \alpha) = \frac{1}{\mathbf{B}(\alpha_0)} \prod_i \pi_{0,i}^{n_{0,i} + \alpha_{0,i} - 1} \\ \times \prod_i \left(\frac{1}{\mathbf{B}(\alpha_i)} \prod_j \pi_{i,j}^{n_{i,j} + \alpha_{i,j} - 1} \right) \\ \times \prod_i \left(\frac{1}{\mathbf{B}(\alpha'_i)} \prod_v \pi_{i,j}^{n'_{i,v} + \alpha'_{i,v} - 1} \right). \end{aligned} \tag{3.8}$$

Repeatedly applying the same steps used to derive (3.5) allows one to write the marginal joint probability of $(\mathcal{X}, \mathcal{Y})$ in terms of hyperparameters and counts.

$$P(\mathcal{X}, \mathcal{Y} \mid \alpha_0, \alpha_i, \alpha_i') = \frac{\mathbf{B}((\boldsymbol{n}_0 + \alpha_0)}{\mathbf{B}(\alpha_0)} \prod_i \frac{\mathbf{B}(\boldsymbol{n}_i + \alpha_i}{\mathbf{B}(\alpha_i)} \frac{\mathbf{B}((\boldsymbol{n}_i' + \alpha_i')}{\mathbf{B}(\alpha_i')} \tag{3.9}$$

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 (3.9) 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.

3.4 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 $y_{1:T}^i$ given $\mathcal{X}$, $\mathcal{Y}_{-i}, \alpha$, where $\mathcal{Y}_{-i}$ denotes $\mathcal{Y} \setminus \{y_{1:T}^i\}$. This sampler follows the approach described in [7] for Probabilistic Context Free Grammars (PCFGs), and utilized for HMMs in [5].

The primary difficulty in constructing such a sampler is due to marginalizing over the parameters, as doing so introduces dependencies amongst the latent state sequences $\mathcal{Y}$, Figure 3.2. Let us proceed in the usual manner for constructing such a Gibbs sampler. We have

$$P(y_{1:T}^i \mid \mathcal{Y}_{-i}, \mathcal{X}, \alpha) = \frac{P(x_{1:T}^i \mid y_{1:T}^i) P(y_{1:T}^i \mid \mathcal{Y}_{-i}, \mathcal{X}_{-i}, \alpha)}{P(x_{1:T}^i \mid \mathcal{Y}_{-i}, \mathcal{X}_{-i}, \alpha)}. \tag{3.10}$$

The $P(x_{1:T}^i \mid \mathcal{Y}_{-i}, \mathcal{X}_{-i}, \alpha)$ term in the denominator of (3.10) is particularly problematic, as noted by [7]. 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

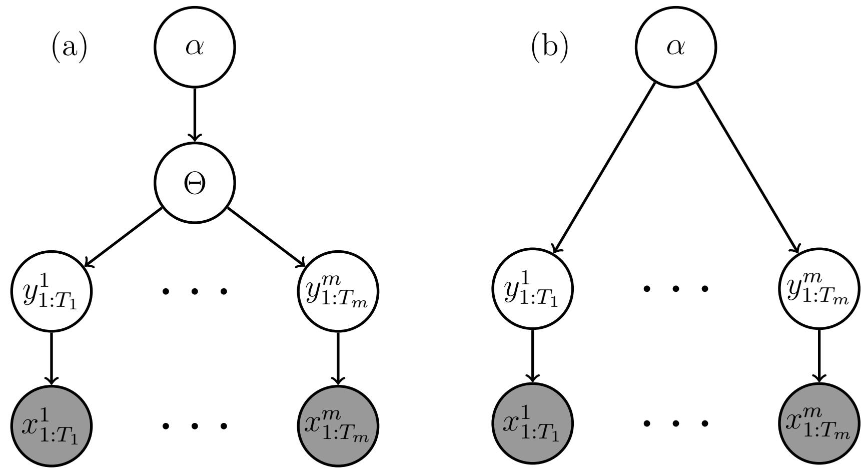

Figure 3.2: Graphical representation of a Bayesian HMM without (a) and with (b) marginalization of the parameters Θ. Notice that marginalization of Θ introduces dependencies between latent states y, since they now share a common parent α in (b).

computationally efficient manner.

Instead, since the denominator of (3.10) 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 1.

Algorithm 1 The Metropolis-Hastings Algorithm.

```
Input: $x^0$ starting value
 1: for $i \leftarrow 1, \ldots, n$ do
 2:     $x^* \sim q(\cdot \mid x^{i-1})$
 3:     $u \sim \mathrm{Unif}(0, 1)$
 4:     if $u < \dfrac{f(x^*)q(x^{i-1} \mid x^*)}{f(x^i)q(x^* \mid x^{i-1})}$ then
 5:         $x^i \leftarrow x^*$
 6:     else
 7:         $x^i \leftarrow x^{i-1}$
 8:     end if
 9: end for
10: 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 [35]. 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

$$\begin{aligned} p &= \frac{P(y^*_{1:T} \mid \mathcal{Y}_{-i}, \mathcal{X}, \alpha)}{P(y^i_{1:T} \mid \mathcal{Y}_{-i}, \mathcal{X}, \alpha)} \frac{q(y^i_{1:T})}{q(y^*_{1:T})} \\ &= \frac{P(\mathcal{Y}^*, \mathcal{X} \mid \alpha)}{P(\mathcal{Y}, \mathcal{X} \mid \alpha)} \frac{q(y^i_{1:T})}{q(y^i_{1:T}*)} \\ &= \frac{\mathbf{B}(\boldsymbol{n}^*_0 + \alpha_0)}{\mathbf{B}(\boldsymbol{n}_0 + \alpha_0)} \prod_j \left[\frac{\mathbf{B}(\boldsymbol{n}^*_j + \alpha_j)}{\mathbf{B}(\boldsymbol{n}_j + \alpha_j)} \frac{\mathbf{B}(\boldsymbol{n}'^*_j + \alpha'_j)}{\mathbf{B}(\boldsymbol{n}'_j + \alpha'_j)} \right] \frac{q(y^i_{1:T})}{q(y^*_{1:T})} \end{aligned} \tag{3.11}$$

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

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

utilizing the algorithm outlined in [28].

TODO: Explain Scott's algorithm.

Empirically this results in a majority of proposals $y^*_{1:T}$ being accepted, which aligns with the observations reported by [5].

3.5 Mixtures of Hidden Markov Models

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

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

where $\mathrm{HMM}(\Theta_{z_i})$ is used as shorthand for the HMM generative story outlined in Section 3.2. Again, the marginal likelihood $\mathcal{Z}$, $\mathcal{Y}$, and $\mathcal{X}$ can be expressed in closed form in terms of hyper parameters and counts.

$$P(\mathcal{Z}, \mathcal{X}, \mathcal{Y} \mid \alpha, \beta) = \\
\frac{\mathbf{B}(\boldsymbol{c} + \beta)}{\mathbf{B}(\beta)} \prod_k \left(\frac{\mathbf{B}(\boldsymbol{n}_{k,0} + \alpha_0)}{\mathbf{B}(\alpha_0)} \prod_i \left(\frac{\mathbf{B}(\boldsymbol{n}_{k,i} + \alpha_i)}{\mathbf{B}(\alpha_i)} \frac{\mathbf{B}(\boldsymbol{n}'_{k,i} + \alpha'_i)}{\mathbf{B}(\alpha'_i)} \right) \right). \tag{3.12}$$

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

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 philosophically unsatisfying. A modern approach to dealing with this problem is Bayesian nonparametrics [8]. 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 naturally be extended to handle a countably infinite number of cluster components using a Dirichlet Process [33]. The constructive definition of the Dirichlet process is given in terms of a *stick-breaking process* [29] with parameter $\beta \in \mathbb{R}_{>0}$, frequently denoted GEM(β).

$$\begin{aligned}
\xi_k &\sim \mathrm{Beta}(1,\beta), k = 1,\dots,\infty \\
\phi_k &= \xi_k \prod_{l=1}^{k-1}(1-\xi_l)
\end{aligned}$$

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.

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

Note that for any finite set of $\{z_1,\dots,z_m\} = \mathcal{Z}$, $\boldsymbol{c}$ contains at most m unique elements. In analogy to the finite mixture me, denote the number of unique elements K. Then the marginal likelihood of $\boldsymbol{c}$ given β has the following form [12]

$$P(\boldsymbol{c} \mid \beta) = \frac{m^K \Gamma(\beta)}{\Gamma(\beta+m)} \prod_k \Gamma(c_k). \tag{3.13}$$

Collapsing all parameters results in the following slight alteration of equation (3.12).

$$P(\mathcal{Z},\mathcal{X},\mathcal{Y}\mid\alpha)=\left(\frac{m^K\Gamma(\beta)}{\Gamma(\beta+m)}\prod_k\Gamma(c_k)\right)$$
$$\times\prod_k\left(\frac{\mathbf{B}(\boldsymbol{n}_{k,0}+\alpha_0)}{\mathbf{B}(\alpha_0)}\prod_i\left(\frac{\mathbf{B}(\boldsymbol{n}_{k,i}+\alpha_i)}{\mathbf{B}(\alpha_i)}\frac{\mathbf{B}(\boldsymbol{n}'_{k,i}+\alpha'_i)}{\mathbf{B}(\alpha'_i)}\right)\right). \quad (3.14)$$

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 3.4 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 [19]. In particular it would be easy to alternate between sampling z_i variables and latent state sequences $y^i_{1:T}$. Unfortunately this will mix poorly, since the cluster allocation is highly coupled with the sequence of latent states. The ability to propose a new cluster and a new state sequence simultaneously avoids stagnation of the Markov chain, motivating the desire to blockwise sample $(z_i, x^i_{1:T})$.

Luckily the marginal posterior predictive distribution of a Dirichlet process has a particularly convenient form [26] 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 can be neglected when calculating the acceptance ratio of the Metropolis within Gibbs sampler.

Chapter 4

A Generative Model for Repetitive Sequences

A significant advantage of PGMs over popular discriminative Machine Learning methods such as Support Vector Machines (SVMs), Random Forests, and Neural Networks, is the ability to easily incorporate structured knowledge via latent variables and conditional dependencies into the models [15], [22], [34]. 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, [14], [36], 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. [23] use a similarly structured model to develop a high accuracy unsupervised grammar induction system.

4.1 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 structures. Consider the oscillatory discrete time series depicted in Figure 4.1.

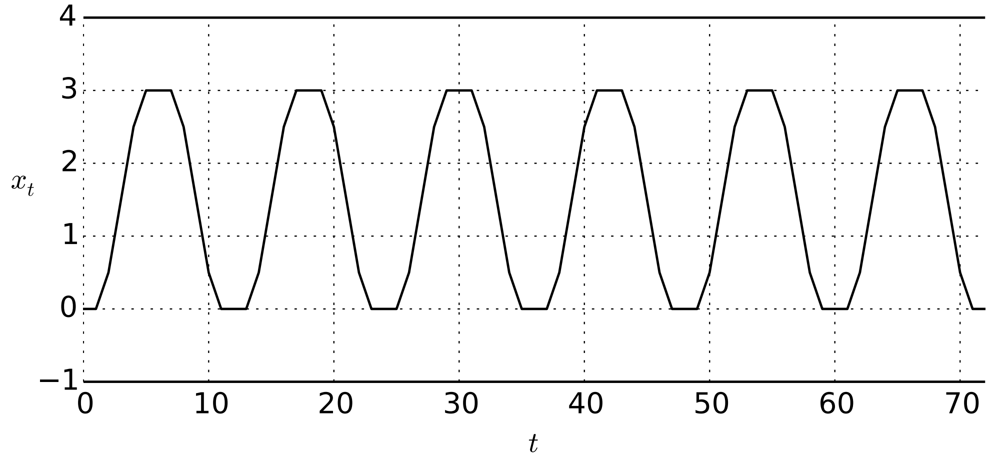

Figure 4.1: Oscillatory discrete time series.

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 4.2.

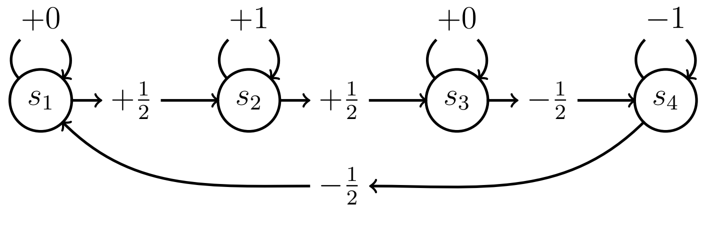

Figure 4.2: 4 state cyclic DFA.

The natural probabilistic generalization of this DFA is a HMM with a cyclic transition structure and parametric emission distributions replacing discrete symbols, Figure 4.3.

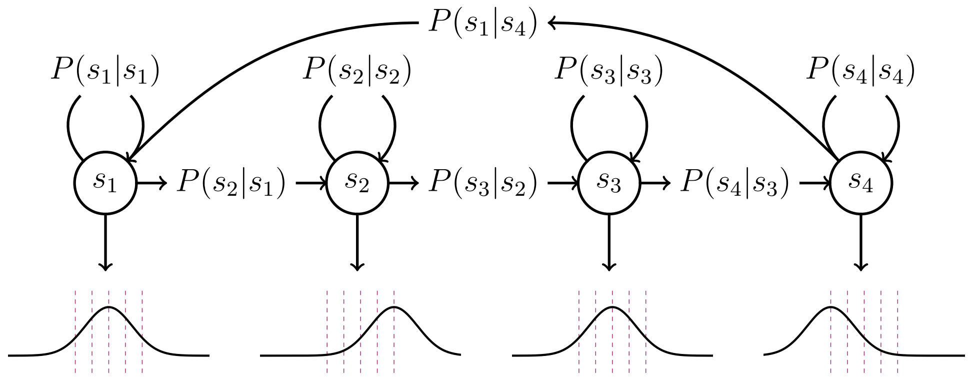

Figure 4.3: 4 state CHMM.

4.2 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 aCHMM 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 5

Synthetic experiments with Hidden Markov Model

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.

5.1 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.
The point

For the first task, the HMM fit with the collapsed Gibbs sampling performed quite well, and the 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 5.1.

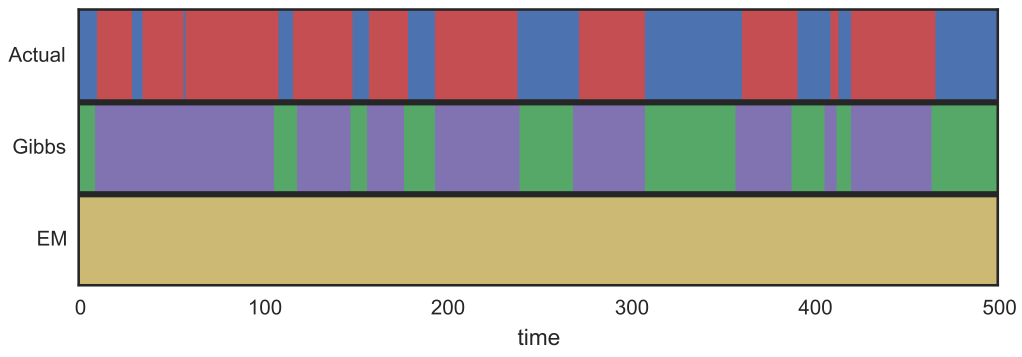

Figure 5.1: Actual state sequence and example inferred state sequences from the HMM fit with Gibbs sampling and EM. The HMM fit with EM only uses a single state to model the observed sequence.

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 5.2.

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

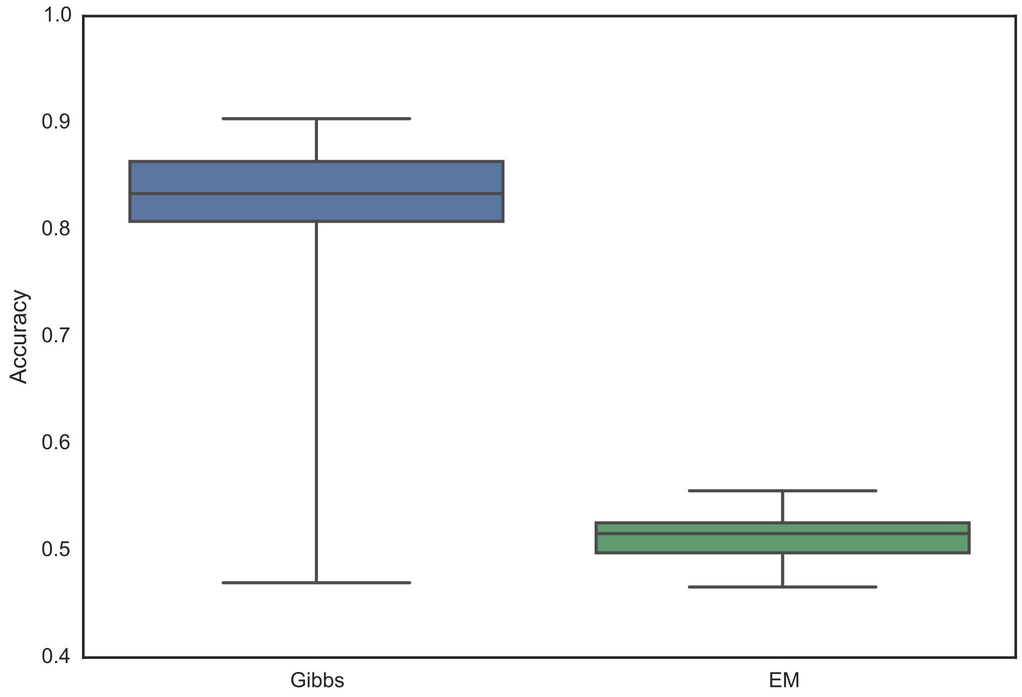

Figure 5.2: Accuracy for the trick coin discriminative task. Whiskers depict the maximum and minimum values.

5.2 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.

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

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 [25].

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 5.3. 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 5.4.

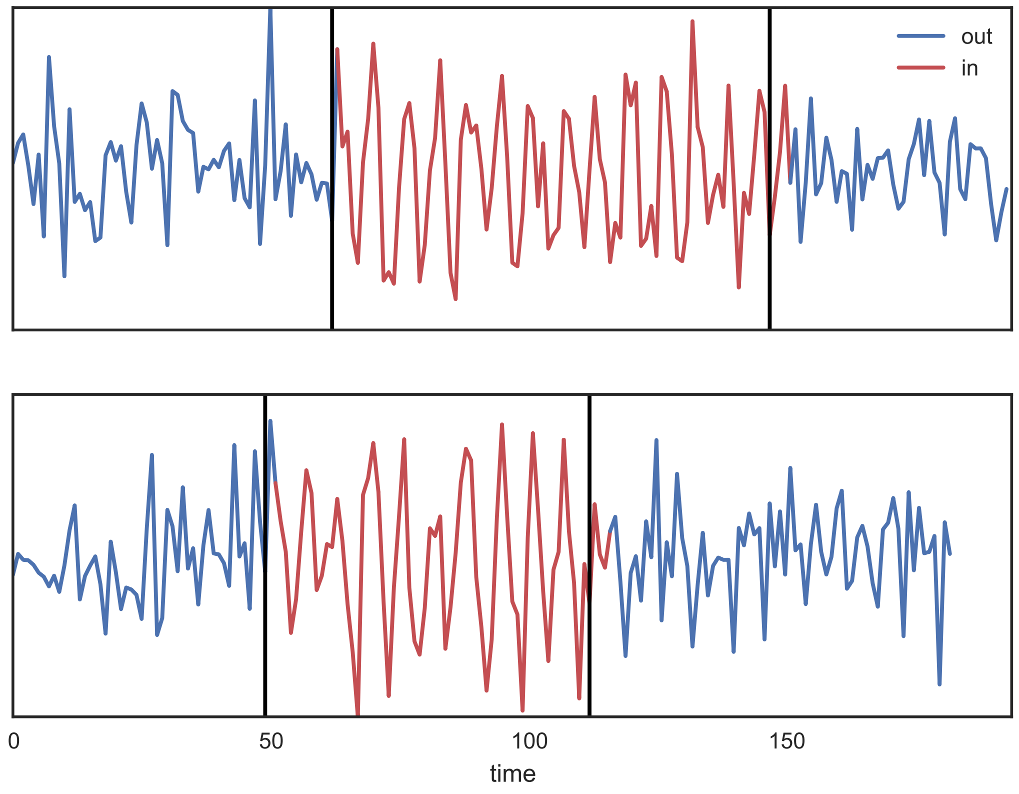

Figure 5.3: Unsupervised segmentation by a HMM with structured prior.

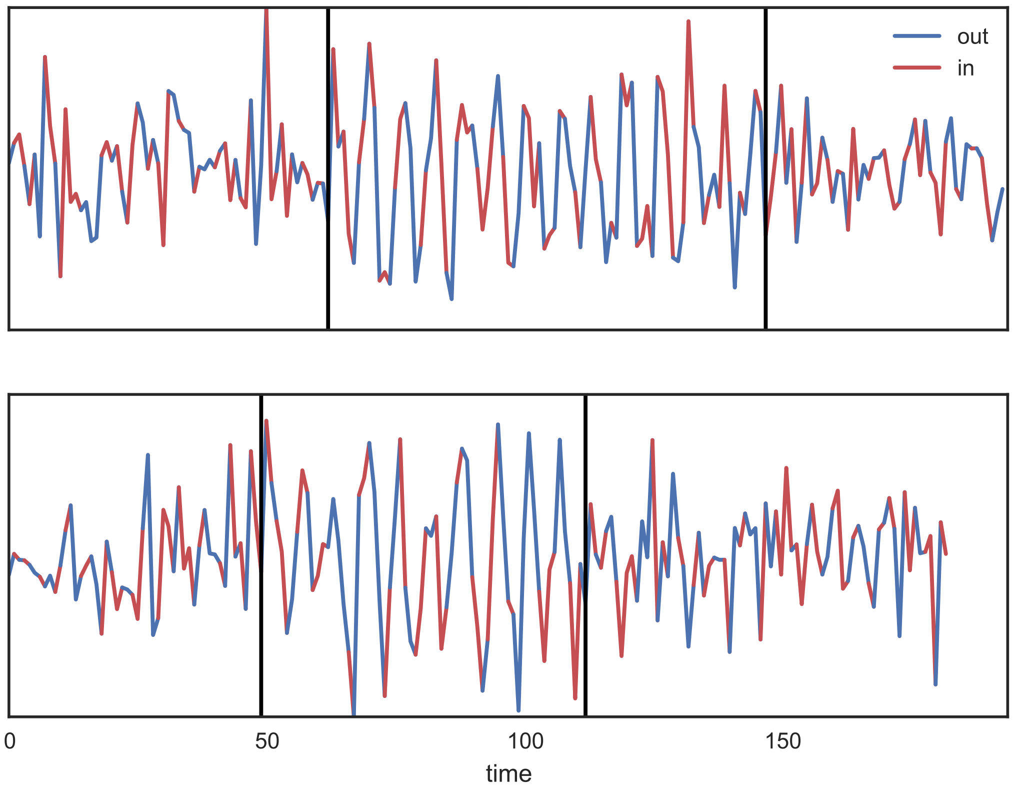

Figure 5.4: Unsupervised segmentation by a HMM with 2 states.

5.3 Clustering Sequences

The final synthetic experiment explores the Bayesian nonparametric approach outlined in Section 3.5 for the purpose of sequence clustering. As ground truth clusters the following four recurrent functions are used.

$$
\begin{aligned}
f_1(t) &= \sin(t) & f_3(t) &= \max(-2, \min(2, \tan(t/4))) \\
f_2(t) &= \sin(t) + \sin(t/2) & f_4(t) &= \max(0, 2\cos(t/4))
\end{aligned}
$$

An example sequence is then generated from the following process.

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

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 first utilizes standard HMMs as the base distribution and the second uses CHMMs as the base distribution. The experiment is repeated 10 times and the resulting clusterings are compared using Adjusted Mutual Information (AMI) [37], a version of Mutual Information corrected to adjust for chance agreements. Figure 5.6 shows that as well as increased performance, in terms of AMI, the Dirichlet Process with CHMM base distributions finds the correct number of clusters in 80% of experiments. This is in contrast to the standard HMM which is unable to find the actual number of clusters in any of the experiments.

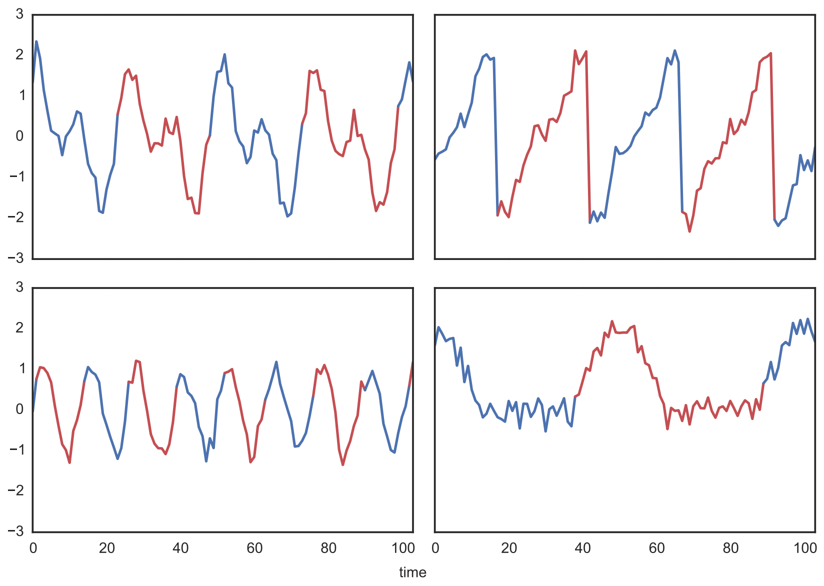

Figure 5.5: Example from each cluster and recurrent structure discovered by the Dirichlet Process mixture of HMMs with structured prior.

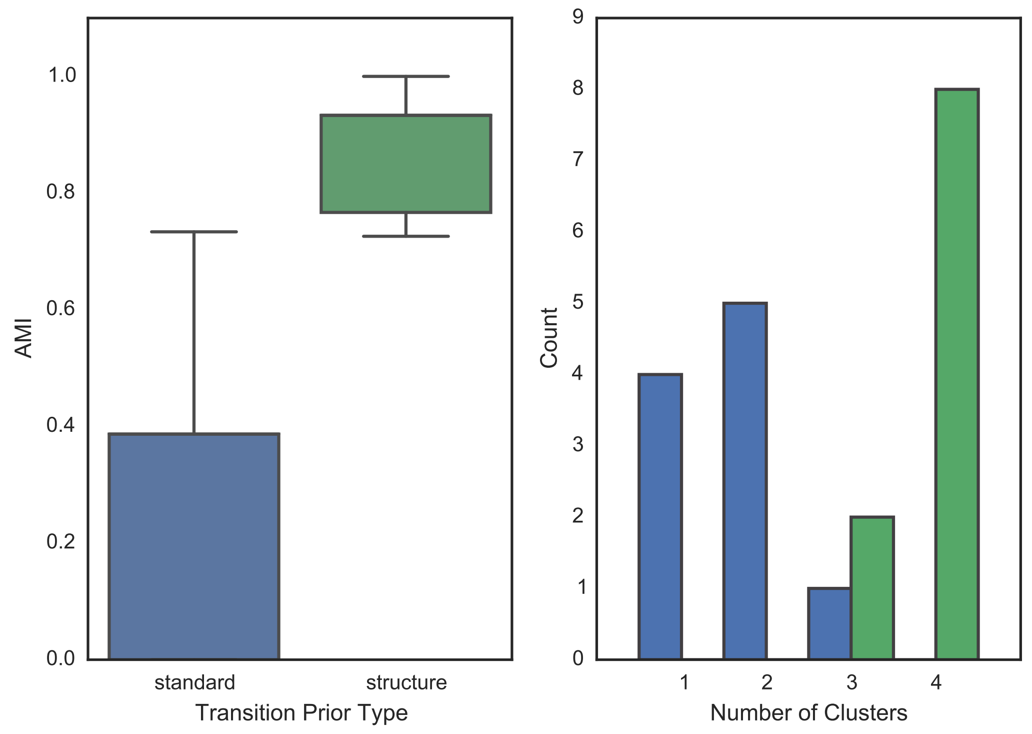

Figure 5.6: Results from clustering artificial sequences. Comparison of Adjusted mutual information and number of discovered clusters with (green) and without (blue) structured transition prior.

Chapter 6

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*.

6.1 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 6.1 summarizes the splits.

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

Table 6.1: Summary of segment classification dataset.

6.2 Models

The proposed CHMM was compared to a Structured Perceptron [4] and a strong performing system heavily influenced from the activity recognition literature [16]. 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.

6.2.1 Baseline from Literature

Decomposition of a temporal classification problem into a series of fixed size independent and identically distributed (iid) decisions is a common approach to activity recognition in the literature. The approach in [16] is to first derive a series of standard features from the raw sensor data. A SVM is then applied to overlapping fixed size windows consisting of several seconds of sensors. Significantly better results were obtained on the data considered here using a Random Forest rather than a SVM, and only the former is reported here. This discrepancy is likely due to [16] 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 specturm summed across 10 linearly spaced bands.

The free software scikit-learn [21] 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: $(10, 20, \textbf{none})$.
- The maximum number of features considered for each split: $(\sqrt{\mathbf{d}}, all)$.
- The spit evaluation function: **Gini impurity** or information gain.

6.2.2 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) [13].

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 [20], 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.

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

6.2.3 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 [9]. 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 [31] as well as [38].

The second method is based on the collapsed block Gibbs sampling approach outlined in Section 3.4. The MAP sampled assignment is used to produce a point estimate of the parameters.

6.3 Results

Table 6.2 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 Hidden Markov Models (MoHMM) 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 MoHMM 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 MoHMM increases. Inspecting performance for just the *rest* class reveals this is at least in some part due to the MoHMM doing a better job of modeling the *rest* class, the last two columns of Table 6.3. 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**
MoHMM (block Gibbs)	0.848	0.775
MoHMM (Viterbi)	0.837	0.762

Table 6.2: Segment classification performance of different classifiers.

Classifier	**Train Acc.**	**Test Acc.**	**F1**	**Precision**	**Recall**
Random Forest	0.995	**0.81**	0.789	**0.925**	0.755
MoHMM	0.829	0.799	0.790	0.81	**0.829**

Table 6.3: 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.

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 MoHMM. Several attempts were made to combat overfitting in the Random Forest, largely by reducing the complexity of the individual ensemble members, however in all cases this resulted in decreased test performance.

Chapter 7

Segmenting Exercises

The strong performing baseline classifier from Chapter 6 has the disadvantage that it requires ground truth segmentations. In contrast the CHMM system presented for segmentation in Section 5.2 can perform this segmentation in an unsupervised manner. Doing so simply requires inspection of the latent state sequences. Portions of the sequence corresponding to the

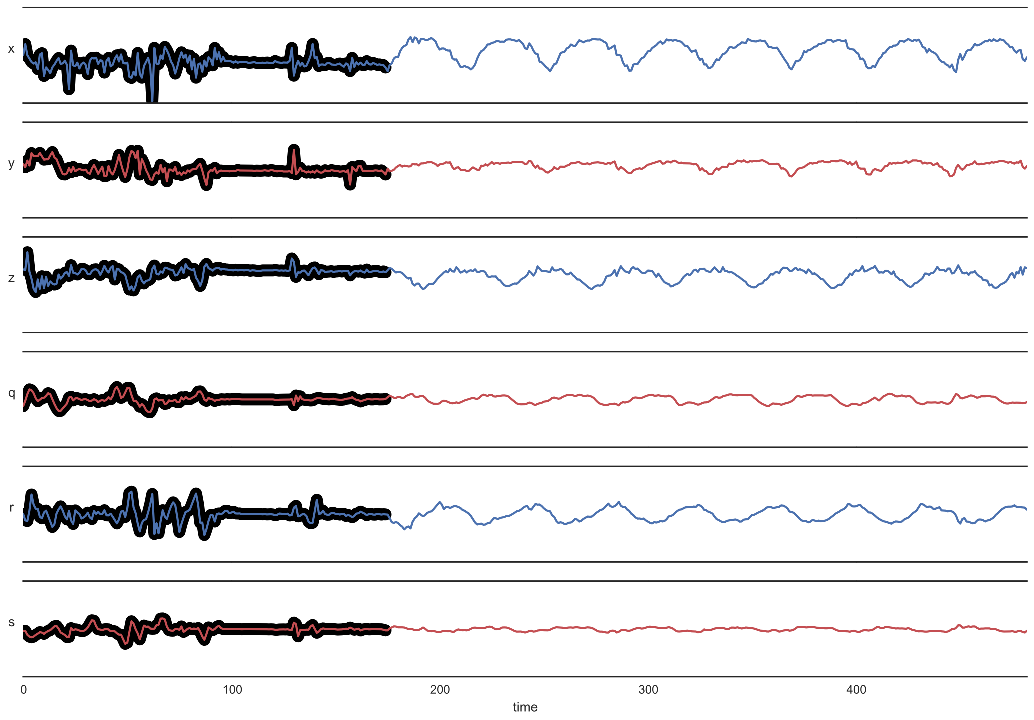

Figure 7.1: Example annotation error.

Chapter 8

Simultaneous Segmentation and Classification of Exercises

This chapter focuses on the most difficult case where the time series $x_{1:T}$ contains instances of different exercises interleaved with *rest*, i.e., a routine. Thus for each time $t = 1, \ldots, T$ there is an associated label $a_t \in \{\varnothing\} \cup \{1, \ldots, K\}$ denoting the current activity, and the algorithm must output a sequence of labels $\hat{a}_{1:T}$. Metrics are a problem here as predicting interleaved periods of rest during an exercise are extremely harmful to downstream components (like repetition counting and displaying information about the entire routine), but are only slightly penalized by something like mean hamming distance. Furthermore one needs to correct for labeling every time step with the majority class ($\varnothing$). For these reasons I've mostly been resorting to visual inspection. Possible options include:

Maximal Overlapping Subsequence: This isn't something I've found a reference for, but which seems quite reasonable. For each homogeneous region $a_{t_1:t_2}$ find the maximal overlapping predicted subsequence

$$\hat{s} = \arg\max\{t_2' - t_1' \mid t_1 \leq t_1' < t_2' \leq t_2,\ \hat{a}_i = \hat{a}_j,\ t_1' \leq i, j \leq t_2'\}$$

$$\text{err}(\hat{s}, a_{t_1:t_2}) = t_1' - t_1 + \sum_{i=t_1'}^{t_2'} \mathbb{1}\{\hat{a}_i = a_i\} + t_2 - t_2'.$$

For example

$$\begin{array}{rcl}
a & = & [\varnothing, \ \varnothing, \ 1, \ 1, \ 1, \ 1, \ 1, \ \varnothing, \ \varnothing, \ 2, \ 2, \ 2] \\
\hat{a} & = & [\varnothing, \ \varnothing, \ 1, \ \varnothing, \ 1, \ 1, \ 1, \ 1, \ \varnothing, \ 2, \ 2, \ 2] \\
\text{err} & = & [0, \ 0, \ 1, \ 1, \ 0, \ 0, \ 0, \ 1, \ 0, \ 0, \ 0, \ 0]
\end{array}$$

The total error for the entire sequence is then the error of the individual s terms.

Edit Distance: Treat each homogeneous region as a single symbol and compute the edit distance between the resulting ground truth and predicted strings.

Acronyms

DBN	Dynamic Bayesian Network
HMM	Hidden Markov model
CHMM	Cyclic Hidden Markov Model
HHMM	Hierarchical HMM
MoHMM	Mixture of Hidden Markov Models
DFA	Deterministic Finite State Automaton
PGM	Probabilistic Graphical Model
CRF	Conditional Random Field
MEMM	Maximum Entropy Markov Model
SVM	Support Vector Machine
MAP	Maximum a Posteriori
EM	Expectation Maximization
PCFG	Probabilistic Context Free Grammar
MCMC	Markov Chain Monte Carlo
NLP	Natural Language Processing

AMI	Adjusted Mutual Information
iid	independent and identically distributed

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] 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.

[4] 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.

[5] 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.

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

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

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

[9] 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.

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

[11] 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.

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

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

[14] 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.

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

[16] 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.

[17] Michael Muehlbauer, Gernot Bahle, and Paul Lukowicz. What can an arm holster worn smart phone do for activity recognition? In *Proceedings of the 2011 15th Annual International Symposium on Wearable Computers*, ISWC, pages 79–82, Washington, DC, USA, 2011. IEEE Computer Society.

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

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

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

[21] 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.

[22] 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.

[23] 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.

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

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

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

[27] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity recognition from accelerometer data. In *AAAI*, volume 5, pages 1541–1546, 2005.

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

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

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

[31] 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.

[32] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2006.

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

[34] 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.

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

[36] 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.

[37] 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.

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