Hidden Markov Models and Dynamical Systems

February 21, 2025

Preface

This book arose from a pair of symposia on hidden Markov models that Kevin Vixie organized at the 2001 SIAM Dynamical Systems meeting. At the end of the first symposium someone asked for a simple reference that explains the basic ideas and algorithms for applying HMMs. We were stumped[1]. A group of the participants suggested writing this book to answer the question. Two years later, Fraser alone delivered a proposal for the book that included about two chapters to SIAM at the 2003 Dynamical Systems meeting. Sometime after SIAM published the first edition in 2008, we (Fraser alone again) began working on updating the software we used to generate the book. In an effort to use better software practices, we wrote tests, and one of those tests revealed a conceptual error in the last chapter of the book. In the current edition Chapter 6 is a thorough revision that addresses the same tasks with a simpler approach.

The book is intended for readers who have backgrounds and interests typical of those who attend the SIAM Dynamical Systems meetings. In particular, our view is that HMMs are discrete state discrete time stochastic dynamical systems, and that they are often used to approximate dynamical systems with continuous state spaces operating in continuous time. Thus, by using familiar analogies, it should be easy to explain HMMs to readers who have studied dynamical systems.

The basic techniques were well developed in the 1970's for work on speech and language processing. Many in speech research in the 1980's through the end of the century learned about the techniques at a symposium held at the Institute for Defense Analysis in Princeton. A small number of proceedings of that symposium [2] were printed, and the volume was called *the blue book* by workers in the field. We were not part of that community, but we have a copy of the blue book. It explains the basic algorithms and illustrates them with simple concrete examples. We hope our book is as simple, useful, and clear.

In chapters 3, 4 and 7 we extend ideas from HMMs with discrete states and observations to more general states and observations. In literature about such more general systems and observations, the techniques are called *data assimilation*. Because the discrete character of the states and observations of basic HMMs makes the ideas and algorithms for them easier to explain and under-

[1] In subsequent years several good references that explain the basic ideas of HMMs have appeared, e.g., **ToDo:** Find some

stand than more general data assimilation techniques, we recommend studying HMMs before working on applications that require more those general data assimilation tools.

Although there are now other books and papers that are about HMMs exclusively or in part, we believe that the present volume is unique in that:

It is introductory Sophomore or Junior study in engineering, mathematics, or science provides the prerequisites for most of the book. The exceptions are ideas from dynamical systems and information theory. In particular, we use the Gibbs inequality (Eqn. (2.54)) in developing the EM algorithm in Chapter 2. Although Chapter 5 *Toy Problems and Performance Bounds* deals with Lyapunov exponents and entropies, it is not a prerequisite for any other chapter.

Algorithms are explained and justified We present enough of the theory behind the basic algorithms in Chapter 2 so that a reader can use it as a guide to developing her own variants.

We provide Python and data for the algorithms and examples You can fetch the code and data and build the book yourself. You can examine the source code to resolve questions about how figures were made, and you can build on the software to try out your own ideas for variations. (See *Notes on Software* on page 135 of the appendix.)

It uses analogies to dynamical systems For example, we demonstrate the HMM training algorithm by applying it to data derived from the Lorenz system. The result, as Fig. 1.10 illustrates, is that the algorithm estimates a discrete state generating mechanism that is an approximation to the state space of the original Lorenz system.

We illustrate with practical examples In Chapter 6 we present an application to experimental measurements of electrocardiograms (ECGs). **ToDo:** I want a practical application of particle filtering in Chapter 7

ToDo: Point to review papers and material (I've collected some of the kind of stuff I have in mind at `http://fraserphysics.com/~andy/HMMs/`). Perhaps organize this stuff under the following topics:

- MCMC annealing and optimization
- Other applications
- Other papers books and web sites
- Bayes nets

Notation

In general we introduce notation as it gets used in the text and collect it in a section at the end of the book on page 137. However to avoid confusion, we now note our use of Python notation for sequences[2], namely

$$y[0:T] \equiv [y[0], y[1], \ldots, y[T-1]]\,.$$

There are a few different notation conventions for probabilities and stochastic processes each of which has drawbacks. We describe the conventions we've chosen in the notation section.

Acknowledgments

In writing the first edition of this book, we relied on much that we learned from colleagues in Portland. In particular, Todd Leen kept us informed about developments in the machine learning community. We've relied on the review of the ergodic theory of dynamical systems in Kevin Vixie's dissertation[24] for Chapter 5 of this book. Shari Matzner and Gerardo Lafferriere helped us understand the convergence properties of the EM algorithm. Also the following colleagues have gave us constructive comments on drafts of the book: Patrick Campbell, Ralf Juengling, Shari Matzner, Cosma Shalizi, and Rudolph Van der Merwe.

We thank the following people and organizations for providing the data that we used for examples:

Carl Otto Weiss for providing Tang's[45] laser data

PhysioNet[52] For providing Penzel's[20] ECG data and the WFDB python package[49] for accessing that data from python.

Project Gutenberg For digitizing and distributing *A Book of Prefaces*, by H. L. Mencken

We were fortunate to meet the late Karl Hegbloom in Portland. Karl contributed to several free software projects and he helped us with the figures and software for the fist edition. In addition to personal help with software, we relied on free software written by too many contributors to list. We used the NixOS distribution of Linux to organize the software.

For writing the first edition, we acknowledge Portland State University for support of a sabbatical leave

[2]Thus we write the probability of observation y at time t given previous observations as $P\left(y[t] \mid y[0:t]\right)$ which at first is confusing because the sequence $y[0:t]$ does not include $y[t]$. We've found using notation in the text that is inconsistent with the notation in the code to be even more confusing.

Contents

Chapter 1

Introduction

In a dynamical system, a rule maps states $x \in \mathcal{X}$ forward in time. Familiar examples include discrete time maps and differential equations in which the states are elements of $\mathbb{R}^n$. At time t, the current state $x[t]$ of a dynamical system provides all the information necessary to calculate future states and information about past states is redundant. Thus the sequence of states in a dynamical system satisfies the *Markov property*, which we will more formally define in Eqn. (1.7). In applications, we often think of measured data as being a function of states of a dynamical system with $y[t] = G(x[t])$. The function F that maps states forward in time and the function G that maps states to observations make up a *state space model* for sequences of observations. If the observation function is not invertible, then knowledge of the measurement $y[t]$ at a single time t is not sufficient to specify the state $x[t]$ uniquely, and one could say that $x[t]$ is *hidden*. That is the sense of *hidden* in the term "hidden Markov model".

Given a short sequence of observations, say $y[0:3] = [y[0], y[1], y[2])$, one might hope to *reveal* the state $x[2]$ by looking for all initial states that are consistent with the observations. The strategy will only work if the images of all initial states that are consistent with the observations all fall on the same state, i.e., if for all x such that $G(x) = y[0], G \circ F(x) = y[1]$, and $G \circ F \circ F(x) = y[2]$, we find that $F \circ F(x) = \hat{x}$, then the measurements are sufficient to identify $x[2] = \hat{x}$ uniquely. If such a revelation procedure works, then one can use it to map long sequences of observations to long sequences of states and from there to do forecasting of both states and observations.

For most of the state space models that we consider, the function that governs the state dynamics and the observation function both have random elements. Only imagination limits what constitutes the set of states in a state space model. We will consider discrete state spaces that are sets with a finite number of elements and state spaces that are real vector spaces. The sets of observations are similarly varied. As a prelude, we look at some measurements of a laser system that is "Lorenz like".

1.1 Laser Example

In 1963 E.N. Lorenz[37] reported interesting behavior in numerical solutions of the system of equations

$$\dot{x}_1 = sx_2 - sx_1 \tag{1.1a}$$
$$\dot{x}_2 = -x_1x_3 + rx_1 - x_2 \tag{1.1b}$$
$$\dot{x}_3 = x_1x_2 - bx_3 \tag{1.1c}$$

which he had proposed as an approximation for fluid convection. In Eqn. (1.1), $x = (x_1, x_2, x_3)$ is a vector of mode amplitudes, and the parameters s, r, and b describe properties of the fluid. The paper is widely cited, not because it is a good model for convection, but because the interesting behavior of the solutions has characteristics that are typical of what is now called *chaos*. The Lorenz system has been used countless times as an example of a system whose solution trajectories are unstable, aperiodic and bounded, i.e., *chaotic*. We will use numerical simulations of the system as illustrations throughout this book.

In 1975 Haken[33] observed that under certain conditions a laser should obey the same equations. For a laser, one interprets the components of the state x as the electric field, the polarization, and the population inversion in the laser medium. In 1992 Tang et al. [45, 56] reported measurements of the time dependent intensity of the electric field for such a laser. The measured quantity corresponds to $(x_1[t])^2$. Figure 1.1 shows a sequence of Tang's measurements. We produced the second trace in the figure by numerically integrating Eqn. (1.1) with initial conditions and parameters selected to optimize the match. The similarity of the two traces convincingly supports the claim that the laser system is like the Lorenz system.

In working with Tang's laser data, we used a stochastic state space model with the form

$$x[t+1] = F(x[t]) + \eta[t] \tag{1.2a}$$
$$y[t] = G(x[t]) + \epsilon[t]. \tag{1.2b}$$

We implemented the function F by integrating the Lorenz system for an interval $\Delta\tau$, and we used independently identically distributed Gaussian noise with mean zero and covariance $\mathbf{I}\sigma_\eta^2$ to implement the state noise $\eta[t]$. Our measurement model is $G(x[t]) = S_g \cdot (x_1[t])^2 + O_g$ where S_g and O_g are fixed scale and offset parameters, and the measurement noise $\epsilon[t]$ is independently identically distributed Gaussian noise with mean zero and covariance $\mathbf{I}\sigma_\epsilon^2$. The model has the following eleven free parameters:

Lorenz system parameters The values of r, s, and b in (1.1) constitute three free parameters.

Integration time The single parameter $\Delta\tau$.

Offset and scale The pair of parameters O_g and S_g.

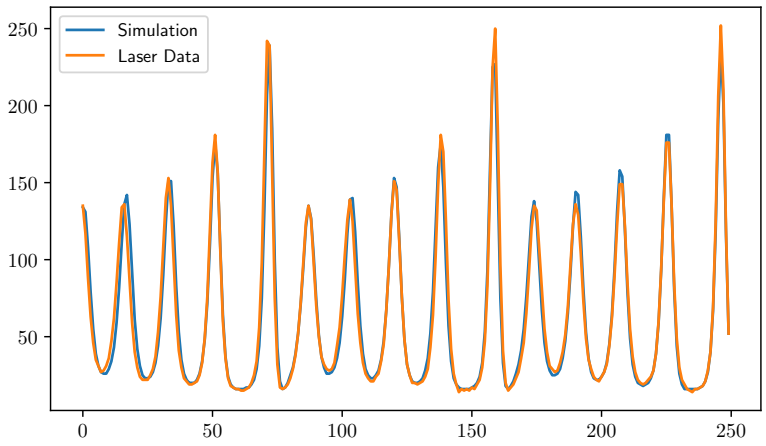

Figure 1.1: Laser intensity measurements. The trace labeled *Laser* is a plot of laser intensity measurements provided by Tang et al. The trace labeled *Simulation* plots a numerical simulation of the Lorenz system (1.1) with parameters that optimize the match.

Measurement noise The single parameter σ_ϵ.

State noise The single parameter σ_η.

Initial state distribution We model the distribution of the initial state as a Gaussian. We treat the mean as three parameters and set the covariance to σ_η.

Using this parameterization, we wrote a routine based on the *extended Kalman filter* techniques described in Chapter 4 to calculate approximate probabilities which we write as $P_{*|\theta}$ where θ denotes the collection of parameters. By passing that routine and Tang's data to the SciPy optimization package, we found a parameter vector that satisfies

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \; P(y[0:250] \mid \theta), \tag{1.3}$$

where $P(y[0:250] \mid \theta)$ is the conditional probability that a sequence of 250 observations will have the values $y[0], y[1], \ldots, y[249]$ given the parameters θ. The parameter vector $\hat{\theta}$ is called the *maximum likelihood estimate* of the parameter vector. Figure 1.2 sketches a piece of the log-likelihood function.

Given $\hat{\theta}$, the maximum likelihood parameters, and the observations, we can calculate many interesting quantities. For example, in Fig. 1.3 we have plotted the sequence of states that has the highest probability, i.e.,

$$\hat{x}[0:250] = \underset{x[0:250]}{\operatorname{argmax}} P(x[0:250] \mid y[0:250], \hat{\theta}), \tag{1.4}$$

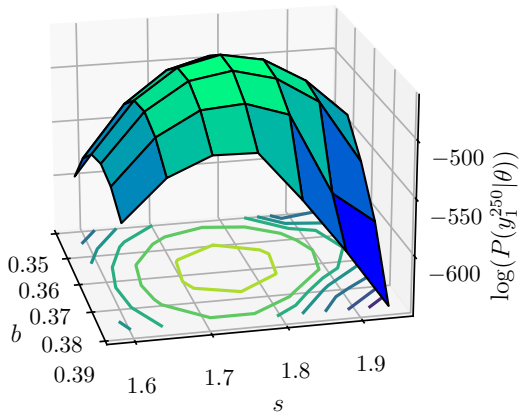

Figure 1.2: Log likelihood as function of s and b. Other parameters were taken from the vector $\hat{\theta}$ that maximizes the likelihood $P(y[0:250] \mid \theta)$ (see Eqn. (1.3)).

and in Fig. 1.4 we have plotted a forecast that we made by iterating the function F on the state $\hat{x}$ that has highest probability given the first 250 observations, i.e.,

$$\hat{x} = \underset{x[250]}{\operatorname{argmax}}\, P(x[250] \mid y[0:250], \hat{\theta}). \tag{1.5}$$

1.2 State Space Models

To get state-space-models that are more general than the form (Eqn. (1.2)) that we used to describe the laser data, we suppose only that a conditional probability distribution $P_{X[t+1]|X[t]}$ governs evolution in state space and another conditional distribution $P_{Y[t]|X[t]}$ governs the observations $Y[t]$. Combining these two conditional distribution functions with a distribution $P_{X[0]}$ of initial states defines probabilities for any collection of states and observations. In particular, it defines the joint probabilities of the *stochastic process* or *information source* consisting of sequences of observations. We refer to such a combination as a *model* and denote it as $P_{*|\theta}$. So defined, the class of state space models is so broad that to do anything useful, we must use smaller subclasses. Typically, we assume that the conditional distributions are time invariant and that a finite set of parameters θ specifies the model. Notice that we have not specified the sets from which we draw the states or observations; they could be discrete, real

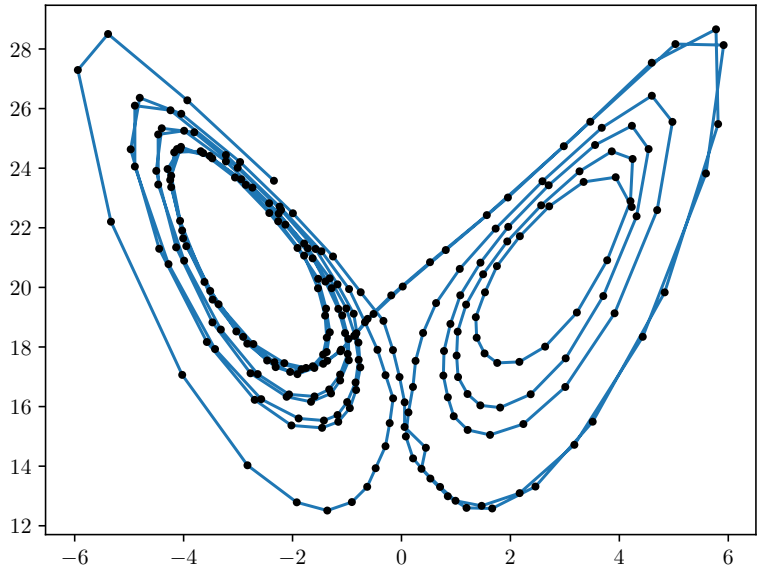

Figure 1.3: State trajectory $\hat{x}[0:250]$ estimated from observation sequence $y[0:250]$. (see Eqn. (1.4).) Components x_1 and x_3 of the Lorenz system (see Eqn. (1.1)) are plotted.

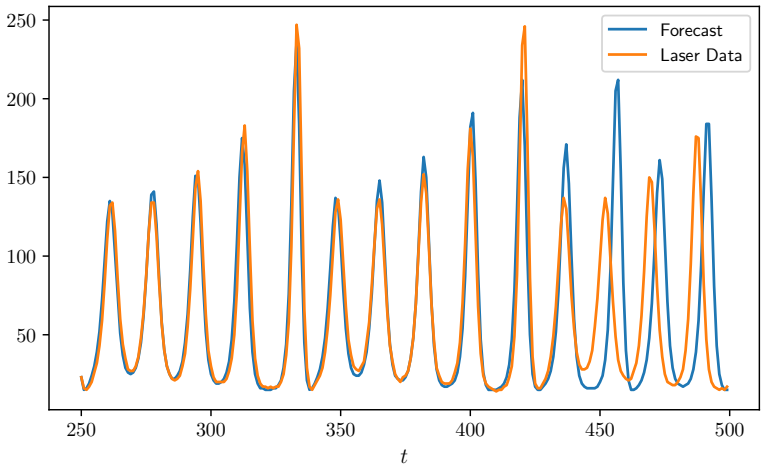

Figure 1.4: Forecast observation sequence. We set the noise terms η and ϵ to zero and iterated Eqn. (1.2) 250 times to generate the forecast $\hat{y}[250:500]$. We started with the initial condition $\hat{x}$ defined by Eqn. (1.5). The forecast begins to fail noticeably around $t = 450$. The failure suggests that the period five cycle in the model is unstable. The period five cycle must have been stable in the actual laser system to appear in the data. Thus an essential characteristic of the model is wrong.

scalars, real vectors, or something else.

1.2.1 Tasks

One can use a parameterized class of state space models $\left\{P_{*|\theta}\right\}$ in many ways including the following:

Model Parameter Estimation Given a model class $\left\{P_{*|\theta}\right\}$ and a sequence of observations $y[0:T]$, we often use the maximum likelihood estimate

$$\hat{\theta}_{MLE} \equiv \underset{\theta}{\operatorname{argmax}} P(y[0:T] \mid \theta) \tag{1.6}$$

to characterize the source Y.

Trajectory Estimation Given a particular model $P_{*|\theta}$ and a sequence of observations $y[0:T]$, one can calculate the conditional distribution of states $P(x[0:T] \mid y[0:T], \theta)$. For example, Fig. 1.3 plots the result of a calculation of

$$\hat{x}[0:250] = \underset{x[0:250] \in \mathcal{X}^{250}}{\operatorname{argmax}} P(x[0:250] \mid y[0:250], \theta).$$

Short Term Forecasting Given a model $P_{*|\theta}$ and a distribution of states at time t, $P_{X[t]}$, one can calculate the conditional distribution of future states or observations. For example, Fig. 1.4 plots the result of a calculation of

$$\hat{y}[250:500] = \underset{y[250:500]}{\operatorname{argmax}} P(y[250:500] \mid y[0:250], \theta).$$

Simulation Given a model $P_{*|\theta}$, one can characterize its long term behavior, answering questions like "What is a hundred year flood?". We often find that models that we fit are not good for such long term extrapolation. For example, the laser data that we described in the previous section seems to come from a stable period five orbit, but the periodic orbit that the trajectory in Fig. 1.3 approximates is linearly unstable. Thus the long term behavior of our estimated model is very different from the actual laser system.

Classification Given sample signals like $y[0:T]$ and two possible signal sources, h and d where $P_{*|h}$ characterizes healthy units and $P_{*|d}$ characterizes defective units, one can can classify a unit on the basis of the *likelihood ratio*

$$R(y[0:T]) = \frac{P(y[0:T] \mid d)}{P(y[0:T] \mid h)}.$$

If $R(y[0:T])$ is above some threshold, it is classified as defective.

1.3 Discrete Hidden Markov Models

In this section we describe the simplest state space models: those that are discrete in time, state, and observation. We begin with a couple of definitions. Three random variables $X[0]$, $X[1]$, and $X[2]$ constitute a *Markov chain* if

$$P_{X[2]|X[0],X[1]} = P_{X[2]|X[1]}, \tag{1.7}$$

which is equivalent to $X[0]$ and $X[2]$ being conditionally independent given $X[1]$, i.e.,

$$P_{X[2],X[0]|X[1]} = P_{X[2]|X[1]} P_{X[0]|X[1]}.$$

An indexed sequence of random variables $X[0:T]$ is a *Markov process* if for any $t : 1 < t < T$ the variables before and after t are conditionally independent given $X[t]$, i.e.,

$$P_{X[0:t],X[t+1:T]|X[t]} = P_{X[0:t]|X[t]} P_{X[t+1:T]|X[t]}. \tag{1.8}$$

We will restrict our attention to time invariant models, i.e., those for which the transition probabilities are constant over time. Begin by considering the ordinary (*unhidden*) Markov model or process sketched in Fig. 1.5. The set of states $\mathcal{S} = \{u, v, w\}$, the probability distribution for the initial state

$$P_{S[0]} = \left[\tfrac{1}{3}, \quad \tfrac{1}{3}, \quad \tfrac{1}{3}\right], \tag{1.9}$$

and the transition matrix

		$S[t+1]$		
$P(s[t+1]\|s[t])$		u	v	w
	u	0	1	0
$S[t]$	v	0	$\frac{1}{2}$	$\frac{1}{2}$
	w	$\frac{1}{2}$	$\frac{1}{2}$	0

(1.10)

define the model, and the model determines the probability of any sequence of states $s[0:T]$, which we write[1] as $P_{S[0:T]}(s[0:T])$. For example we calculate the probability that a sequence of 4 states has the values $s[0:4] = (u, v, w, v)$, (i.e., $s[0] = u$, $s[1] = v$, $s[2] = w$, and $s[3] = v$) as follows:

[1] We use upper case letters to denote random variables and P to denote probability distribution functions. A random variable used as subscript on P specifies that we mean the distribution of that random variable. We can give P an argument to specify the value of the distribution function at that value, e.g. $P_X(3)$ is the probability that the random variable X has the value 3 and $P_X(x)$ is the probability that the random variable X has the value x. We usually drop subscripts on P when the context or argument resolves ambiguity as to which probability function we mean.

$$P(s[0:4]) = P(s[0]) \prod_{\tau=1}^{3} P(s[\tau] \mid s[0:\tau-1]) \tag{1.11}$$

$$= P(s[0]) \prod_{\tau=1}^{3} P(s[\tau] \mid s[\tau-1]) \tag{1.12}$$

$$P(u,v,w,v) = P(v \mid u,v,w) \cdot P(w \mid u,v) \cdot P(v \mid u) \cdot P(u) \tag{1.13}$$

$$= P(v \mid w) \cdot P(w \mid v) \cdot P(v \mid u) \cdot P(u) \tag{1.14}$$

$$= \frac{1}{2} \cdot \frac{1}{2} \cdot 1 \cdot \frac{1}{3} = \frac{1}{12}. \tag{1.15}$$

Applying Bayes-rule $\left(P_{A|B}P_B = P_{A,B}\right)$ recursively, yields Eqn. (1.11) and the special case, Eqn. (1.13). Equations (1.12) and (1.14) follow from Eqns. (1.11) and (1.13) respectively by the *Markov assumption*, Eqn. (1.8). which says that in determining the probability of the t^{th} state given any sequence of previous states only the $(t-1)^{\text{th}}$ state is relevant.

A common exercise is to find a *stationary* probability distribution, i.e., given a transition matrix T find the probability vector V (nonnegative entries that sum to one) that satisfies

$$VT = V. \tag{1.16}$$

If (1.16) holds for $V = P_{S[0]}$, then

$$P_{S[1]} = P_{S[0]}T = P_{S[0]} = P_{S[t]} \forall t,$$

and in fact all probabilities are independent of shifts in time, i.e.,

$$P_{S[0:t]} = P_{S[0+\tau:t+\tau]} \forall (t,\tau),$$

which is the definition of a stationary process. Quick calculations verify that the initial probability and transition matrix in (1.9) and (1.10) do not satisfy (1.16) but that the distribution $V = \left[\frac{1}{7}, \quad \frac{4}{7}, \quad \frac{2}{7}\right]$ does. Although our example is not a stationary stochastic process, it relaxes towards such a process in the sense that

$$\lim_{t\to\infty} P_{S[t]} = \lim_{t\to\infty} P_{S[0]}T^t = \left[\frac{1}{7}, \quad \frac{4}{7}, \quad \frac{2}{7}\right].$$

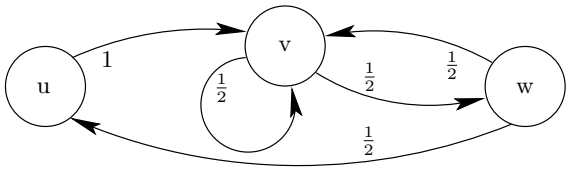

Figure 1.5: A Markov model

The important points about a Markov model also apply to hidden Markov models, namely that:

- The model determines the probability of arbitrary sequences of observations,

- and the assumptions about independence and time invariance permit specification of the model by a small number of parameters.

Now suppose, as sketched in Fig. 1.6, that when the system arrives in a state that rather than observing the state directly, one observes a random variable that depends on the state. The matrix that specifies the random map from states to observations, i.e.:

		Y		
$P(y \mid s)$		d	e	f
	u	1	0	0
S	v	0	$\frac{1}{3}$	$\frac{2}{3}$
	w	0	$\frac{2}{3}$	$\frac{1}{3}$

combined with the distribution of initial states (1.9) and transition matrix (1.10) specifies this hidden Markov model. The notion is that the underlying Markov process chugs along unaware of the observations, and that when the process arrives at each successive state $s[t]$, an observation $y[t]$ is produced in a fashion that depends only on the state $s[t]$.

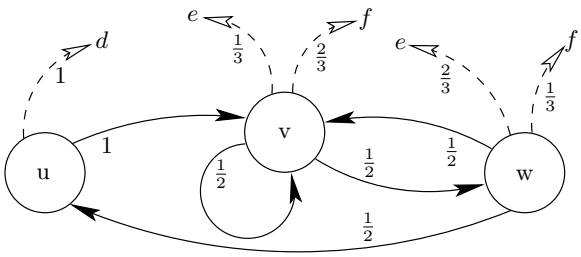

Figure 1.6: A hidden Markov model

Let us calculate the probability that a sequence of four observations from this process would have the values $y[0:4] = (d, e, f, e)$. As an intermediate step we calculate $P(y[0:4], s[0:4])$ for the given observation sequence and all possible state sequences. Then we add to obtain

$$\sum_{s[0:4]} P(y[0:4], s[0:4]) = P(y[0:4]). \tag{1.17}$$

It is convenient that the only state sequences that could have produced the observation sequence are (u, v, v, v), (u, v, v, w), and (u, v, w, v). For any other state sequence $P(y[0:4], s[0:4]) = 0$.

$s[0:4]$	$P(s[0:4])$	$P(y[0:4] \mid s[0:4])$	$P(y[0:4], s[0:4])$	
$uvvv$	$\frac{1}{3} \cdot 1 \cdot \frac{1}{2} \cdot \frac{1}{2}$	$1 \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3}$	$\frac{2}{324}$	(1.18a)
$uvvw$	$\frac{1}{3} \cdot 1 \cdot \frac{1}{2} \cdot \frac{1}{2}$	$1 \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{2}{3}$	$\frac{4}{324}$	(1.18b)
$uvwv$	$\frac{1}{3} \cdot 1 \cdot \frac{1}{2} \cdot \frac{1}{2}$	$1 \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3}$	$\frac{1}{324}$	(1.18c)

Adding the fractions in the right hand column yields

$$P(d, e, f, e) = \frac{7}{324}.$$

Now examine this calculation more carefully beginning with a statement of the model assumptions.

The state process is Markov: Given the current state, the probability of the next state is independent of earlier states and observations, i.e.,

$$P_{S[t+1]|S[0:t+1],Y[0:t+1]} = P_{S[t+1]|S[t]}. \tag{1.19}$$

The observations are conditionally independent given the states: Given the current state, the probability of the current observation is independent of states and observations at all earlier times, i.e.,

$$P_{Y[t]|S[0:t+1],Y[0:t]} = P_{Y[t]|S[t]}. \tag{1.20}$$

Though the assumptions appear asymmetric in time, they are not[2]. From the assumptions, one can derive that

The joint process is Markov:

$$P_{Y[t+1:T],S[t+1:T]|Y[0:t+1],S[0:t+1]} = P_{Y[t+1:T],S[t+1:T]|Y[t],S[t]}$$

[2]We often use the following facts about independence relations:

$$\begin{aligned} P(A \mid B, C) = P(A \mid B) &\iff P(A, C \mid B) = P(A \mid B) \cdot P(C \mid B) \\ &\iff P(C \mid A, B) = P(C \mid B) \end{aligned} \tag{1.21}$$

$$\begin{aligned} P(A \mid B, C, D) = P(A \mid B) &\iff P(A, C, D \mid B) = P(A \mid B) \cdot P(C, D \mid B) \\ &\implies P(A, C \mid B) = P(A \mid B) \cdot P(C \mid B) \\ &\iff P(A \mid B, C) = P(A \mid B). \end{aligned} \tag{1.22}$$

The first chain of implications, (1.21), says that if a process is Markov with time going forward, then it is also Markov with time going backwards. The second chain, (1.22), says that if A is conditionally independent of C and D given B, then A is conditionally independent of C alone given B. By symmetry, A is also conditionally independent of D given B.

Given $S[t]$, $Y[t]$ is conditionally independent of everything else:

$$P_{Y[t]|Y[0:t],Y[t+1:T],S[0:T]} = P_{Y[t]|S[t]}$$

Equations (1.19) and (1.20) are assumptions about *conditional independence* relations. Figure 1.7 represents these relations as a *Bayes net*[11].

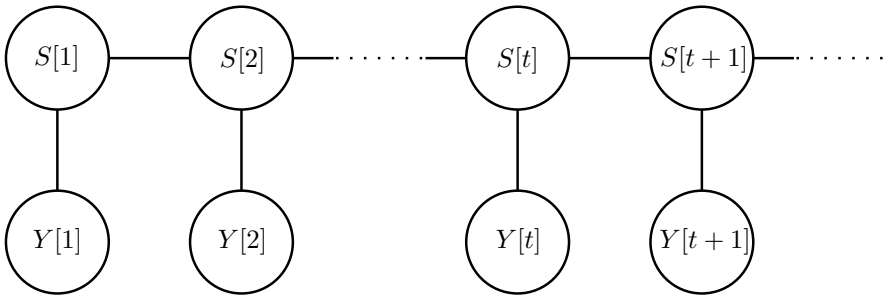

Figure 1.7: Bayes net schematic for a hidden Markov model. The drawn edges indicate the dependence and independence relations: Given $S[t]$, $Y[t]$ is conditionally independent of everything else, and given $S[t-1]$, $S[t+1]$, and $Y[t]$, $S[t]$ is conditionally independent of everything else.

Bayes rule and the assumptions justify

$$P_{Y[0:T],S[0:T]} = P_{S[0:T]}\, P_{Y[0:T]|S[0:T]},$$

$$P_{S[0:T]} = P_{S[0]} \prod_{t=1}^{T-1} P_{S[t]|S[t-1]} \tag{1.23}$$

$$P_{Y[0:T]|S[0:T]} = \prod_{t=1}^{T-1} P_{Y[t]|S[t]} \tag{1.24}$$

and we conclude

$$P_{Y[0:T],S[0:T]} = P_{S[0]} \prod_{t=1}^{T-1} P_{S[t]|S[t-1]} \prod_{t=0}^{T-1} P_{Y[t]|S[t]}.$$

Since the state u produces the observation d exclusively and no other state can produce d, the observation sequence (d, f, e, f) is only possible if the state sequence begins with u and does not return to u. That constraint reduces the number of possible state sequences to eight. The impossibility of state w following itself, further constrains the possible state sequences to the three listed in the calculations of Eqn. (1.18). One can verify the values for $P(s[0:T])$ and $P(y[0:T] \mid s[0:T])$ in those calculations by applying Eqns. (1.23) and (1.24).

The calculation of $P(y[0:4])$ in Eqn. (1.18) is easy because, of the $3^4 = 81$ conceivable state sequences, only three are consistent with the observations and

model structure. In general however, if there are N_S states and an observation sequence with length T, then implementing

$$P(y[0:T]) = \sum_{s[0:T]} P\left(s[0:T], y[0:T]\right)$$

naively requires order $\left(N_S\right)^T$ calculations. If N_S and T are as large as one hundred, $\left(N_S\right)^T$ is too many calculations for any conceivable computer.

There is a family of algorithms whose complexities are linear in the length T that make it possible to use HMMs with interestingly long time series. The details of these algorithms constitute Chapter 2; here we only list their names and objectives. In these descriptions, we denote by θ the vector of parameters that define an HMM, namely the state transition probabilities, the initial state probabilities, and the conditional observation probabilities,

$$\theta \equiv \left\{ \begin{array}{l} \left\{ \; P_{S[t+1]|S[t]}\left(s' \mid s\right) \;\; \forall s, s' \; \right\}, \\ \left\{ \; P_{S[0]}\left(s\right) \;\; \forall s \; \right\}, \\ \left\{ \; P_{Y[t]|S[t]}\left(y_i \mid s'\right) \;\; \forall y_i, s' \; \right\} \end{array} \right\}.$$

The Viterbi Algorithm: Given a model θ and a sequence of observations $y[0:T]$, the Viterbi algorithm finds the most probable state sequence $\hat{s}[0:T]$, i.e.,

$$\hat{s}[0:T] = \underset{s[0:T]}{\operatorname{argmax}} P\left(s[0:T] \mid y[0:T], \theta\right). \tag{1.25}$$

The Baum-Welch Algorithm: (Often called the *Forward Backward Algorithm*) Given a sequence of observations $y[0:T]$ and an initial set of model parameters θ_0, a single pass of the Baum-Welch algorithm calculates a new set of parameters θ_1 that has higher likelihood

$$P\left(y[0:T] \mid \theta_1\right) \geq P\left(y[0:T] \mid \theta_0\right). \tag{1.26}$$

Equality can only occur at critical points of the likelihood function (where $\partial_\theta P\left(y[0:T] \mid \theta\right) = 0$). In generic cases, running many iterations of the Baum-Welch algorithm yields a sequence $\theta[0:n]$ that approaches a local maximum of the likelihood.

The Forward Algorithm: For each time step t and each state s, the forward algorithm calculates the conditional probability of being in state s at time t given all of the observations up to that time, i.e., $P_{S[t]|Y[0:t+1],\theta}\left(s \mid y[0:t+1], \theta\right)$. It also calculates $P\left(y[t] \mid y[0:t], \theta\right)$, the conditional probability of each observation given previous observations. Using these terms it calculates the probability of the entire data sequence given the model,

$$P\left(y[0:T] \mid \theta\right) = P\left(y[0] \mid \theta\right) \cdot \prod_{t=1}^{T-1} P\left(y[t] \mid y[0:t], \theta\right).$$

The forward algorithm is the first phase of the Baum-Welch algorithm.

1.3.1 Example: Quantized Lorenz Time Series

To illustrate these algorithms we have applied them to data that we synthesized by numerically integrating the Lorenz system (Eqn. (1.1) with parameter values $r = 28$, $s = 10$, and $b = \frac{8}{3}$) and recording 40000 vectors $x(\tau)$ with a sampling interval $\Delta\tau = 0.15$. Then we produced a sequence of integer valued observations y_1^{40000} by binning x_1 with boundaries at $-10.0, 0.0$ and 10.0. The result is that for each integer $1 \le t \le 40000$, $x_1[t \cdot 0.15]$ yields $y[t] \in \{1, 2, 3, 4\}$. Figure 1.8 depicts the first few observations.

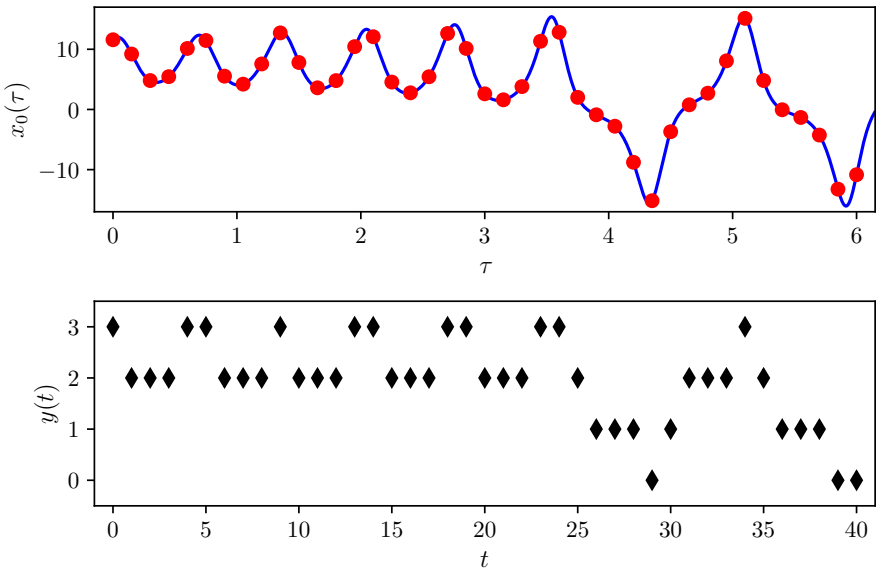

Figure 1.8: Generating the observations $y[0:40]$. The curve in the upper plot depicts the first component $x_1[\tau]$ of an orbit of the Lorenz system (Eqn. 1.1), and the points marked with red dots indicate the values sampled with an interval $\Delta\tau = 0.15$. The points in the lower plot are the $y[t]$ values quantized with boundaries $\{-10.0, 0.0, 10.0\}$.

We randomly generated an HMM with twelve hidden states[3] and 4 possible observations, and then used 100iterations of the Baum-Welch algorithm to select a set of parameters $\hat{\theta}$ with high likelihood for the data. Finally we used the Viterbi algorithm to find the most likely state sequence

$$\hat{s}[0:T] = \underset{s[0:T]}{\operatorname{argmax}}\, P\left(s[0:T] \mid y[0:T], \hat{\theta}\right).$$

Although the plot of *decoded* state values in Fig. 1.9 is not very enlightening,

[3]We chose the number of hidden states to be twelve capriciously so that we could organize Fig. 1.10 on a 4×4 grid.

we can illustrate that there is a relationship between the learned decoded states and the original Lorenz system states by going back to the original data. For each state s, we identify the set of integer times t such that the decoded state is s, i.e., $\{t : \hat{s}[t] = s\}$, and then we find what the Lorenz system state was at each of these times and plot that set of points. In the upper right box of Fig. 1.10 we have plotted points in the original state space that correspond to hidden state number one, i.e., the set of pairs $\{(x_1[t \cdot \Delta\tau], x_3[t \cdot \Delta\tau]) : \hat{s}[t] = 1\}$. In searching for model parameters that give the observed data high likelihood, the Baum-Welch algorithm "discovers" a discrete hidden state structure, and Fig. 1.10 shows that the discrete hidden state structure is an approximation of the continuous state space that generated the data.

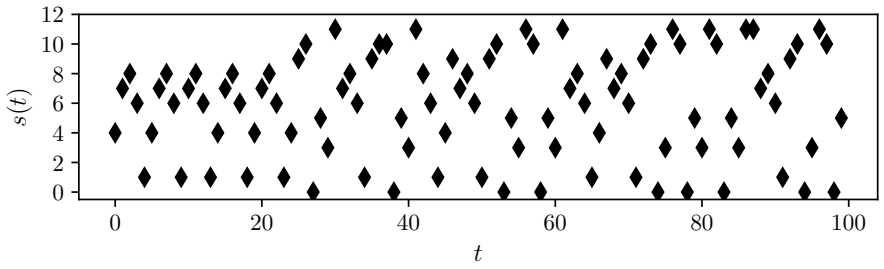

Figure 1.9: A plot of the state sequence found by Viterbi decoding a quantized time series from the Lorenz system. Here the number of the decoded state $s[t]$ is plotted against time t. Although it is hard to see any structure in the plot because the numbers assigned to the states are not significant, Fig. 1.10 illustrates that the decoded states are closely related to positions in the generating state space.

ToDo: Improve Fig. 1.11

1.3.2 Example: Hidden States as Parts of Speech

Hidden Markov models were developed for speech and text processing, and for unscientific audiences, we find the application to language modeling the easiest way to motivate HMMs. Consider for example the sentence, "The dog ate a biscuit." and its reduction to a sequence of parts of speech: *article noun verb article noun*. By choosing different particular articles nouns and verbs and placing them in the order specified by the sequence of parts of speech, we can produce many other sentences such as, "An equation describes the dynamics." The parts of speech are like hidden states and the particular words are like observations.

Rather than using a dictionary or our own knowledge to build a model of language, here we describe the experiment of applying the Baum-Welch algorithm to some sample text to create an HMM. We hope that the experiment will

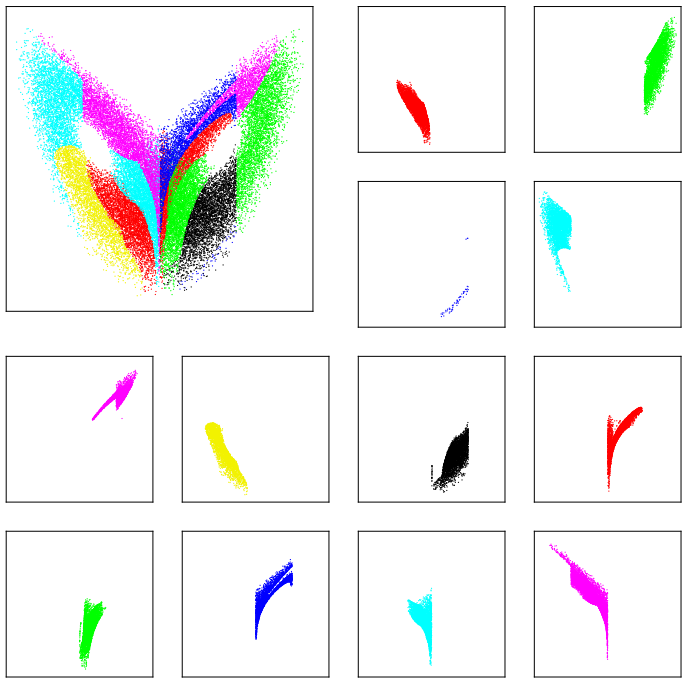

Figure 1.10: The relationship between the hidden states of an HMM and the original coordinates of the Lorenz system. A state transition graph appears in Figure 1.11.

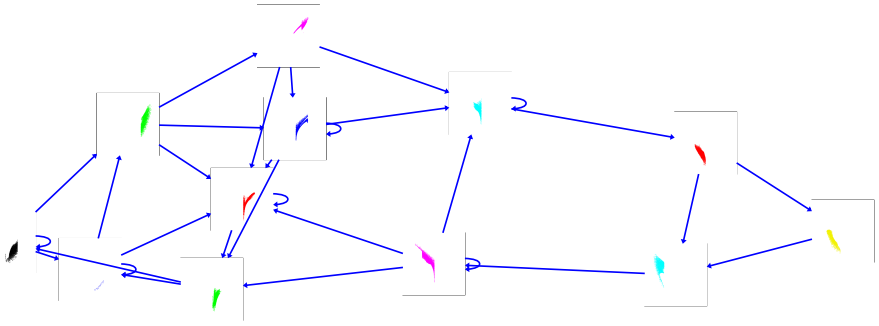

Figure 1.11: Graph of transitions between states depicted in Figure 1.10 drawn by the *graphviz* program.

discover parts of speech. We fetched the fourth edition of *A Book of Prefaces* by H. L. Mencken from www.gutenberg.org, fit an HMM with the Baum-Welch algorithm, and decoded a state sequence with the Viterbi algorithm. We chose a book of essays rather than a novel because we expected that it would not have as much dialog. We feared that different speakers in a novel would require different models.

The experiment consisted of the following steps:

Parse the text: We reduced the text to a sequence of tokens. Each token is a word, a number, or a special character such as punctuation. We retained distinctions between lower and upper case. The length of the resulting sequence was 71459tokens, i.e., $w[0:T]$ with $T = 71459$.

Identify unique tokens: There were 9998unique tokens of which 2,759 appear in the text more than twice.

Create a map from tokens to rank: We sorted the tokens by the frequency of their occurrence so that for the most frequent token w', $R(w') = 1$ and for the most infrequent token, $\bar{w}$, $R(\bar{w}) = 9998$.

Map tokens to integers: We created a sequence of integers $y[0:T]$ where $y[t] \in \{0, \ldots, 2850\} \forall t$. If the token $w[t]$ appeared in the text less than three times, we set $y[t] = 2850$. Otherwise, we set $y[t]$ to $R(w[t])$.

Train an HMM: Starting from a fifteen state model with random parameters, we used 100 iterations of the Baum-Welch algorithm to obtain a trained model.

Decode a sequence of states: By applying the Viterbi algorithm to $y[0:T]$ we obtained $s[0:T]$ where $s[t] \in \{0, \ldots, 11\}$.

Print the most frequent words for each state: For each state s, count the number of times each integer y occurs, i.e., $c(y,s) = \sum_{t:s[t]=s} \delta(y, y[t])$. Then print the words corresponding the ten most frequently occurring integers (excluding the special value $y = 2850$).

The results appear in Table 1.1.**ToDo:** Perhaps drop punctuation from table.

1.3.3 Remarks

One might imagine that HMMs are simply higher order Markov processes. For example, consider the suggestion that the states depicted in Fig. 1.10 correspond to sequential pairs of observations and that the model is a second order Markov model that is characterized by $P_{Y[t+1]|Y[t],Y[t-1]}$, the set of observation probabilities conditioned on the *two* previous observations. Although the number of unique sequential pairs $y[t:t+1]$ that occur in our data is in fact twelve, the fact that some of the states in Fig. 1.10 straddle the quantization boundaries at $x = -10.0$ and $x = 10.0$ belies the suggestion. In general, the class of HMMs

Table 1.1: Words most frequently associated with each state. While we have no interpretation for some of the states, the following interpretations of other states are plausible.

1 – Nominative pronouns	10 – Prepositions
2 – Prepositions	11 – Nouns
3 – Helping verbs	13 – Nouns
5 – Relative pronouns	15 – Adjectives
9 – Articles	

1	he	it	and	I	they	He	It	there	who	we
2	to	in	as	by	with	not	at	for	on	that
3	is	was	are	has	have	had	were	may	would	not
4	.	indeed	vs	Co	S	Sec	U	H	Dreiser	W
5	,	;	that	”	as	and	which	:	what	but
6	-	them	it	all	him	life	course	which	what	America
7	it	be	him	more	all	not	them	out	X	have
8	.	,	]	?	”	!	(	:	;	)
9	the	a	his	an	its	”	their	this	that	any
10	of	and	-	in	to	for	or	as	from	with
11	The	New	American	other	moral	English	one	Sister	Jennie	such
12	”	and	The	but	[	In	But	as	or	that
13	book	man	work	story	way	sense	end	sort	artist	books
14	York	years	Carrie	hand	Gerhardt	men	Titan	out	)	States
15	”	same	first	own	Puritan	whole	new	great	very	other

is more powerful than the class of simple Markov models in the sense that the former includes the later but not vice versa.

Let us emphasize the following points about discrete HMMs:

1. Although the hidden process is first order Markov, the observation process may not be a Markov process (of any order).

2. Any Markov model of any order can be represented by an HMM.

3. Even if the functions governing the dynamics and observations of a continuous state space system are nonlinear, a discrete HMM can approximate the system arbitrarily well[4] by using large numbers of states N_S and possible observation values N_Y.

4. For estimating model parameters, larger numbers of training data are required as N_S and N_Y are increased.

As an illustration of Point 1, consider the process depicted in Fig. 1.12, which produces observation strings with runs of about seven a's interspersed with occasional b's and c's. In the observation stream, the b's and c's alternate

[4]By using a discrete HMM to approximate dynamics governed by a continuous function one sacrifices the opportunity to exploit continuity. That sacrifice will degrade model performance in many applications.

no matter how many a's fall in between. Such behavior can not be captured by a simple Markov process of any order.

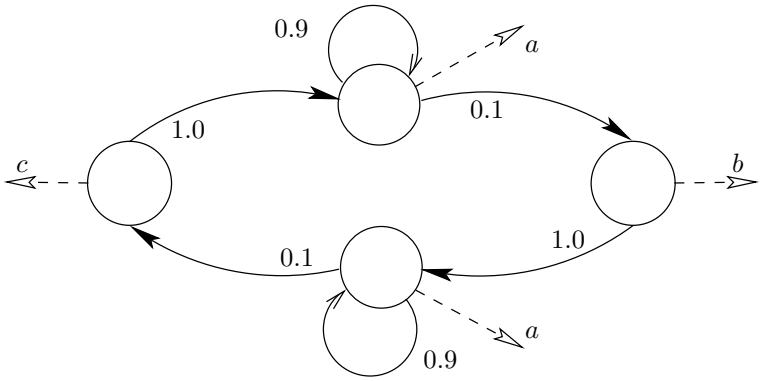

Figure 1.12: An HMM that cannot be represented by a Markov model of any order. Consider the string of observations "$b, a, a, \ldots, a, a, a$". Since the previous non-"a" observation was "b" and the model will not produce another "b" before it produces a "c", the next observation can be either a "c" or another "a", but not a "b". Because there is no limit on the number of consecutive "a's" that can appear, there is no limit on how far back in the observation sequence you might have to look to know the probabilities of the next observation.

The possibility of long term memory makes state space models, e.g., HMMs, more powerful than Markov models. That observation suggests that if there is noise, then the *delay vector reconstructions* described in the chaos literature[39, 44, 29, 19] are suboptimal because they discard information from earlier observations that could be used to more accurately specify the state.

Chapter 2

Basic Algorithms

In Section 1.3 we mentioned the Viterbi algorithm for finding the state sequence that has the highest probability given an observation sequence, the forward algorithm for calculating the probability of an observation sequence, and the Baum-Welch algorithm for finding a parameter vector that at least locally maximizes the likelihood given an observation sequence. This chapter explains the details of those algorithms. Much of the literature on the algorithms and much of the available computer code uses Ferguson's [16] notation. In particular, Rabiner's [18] widely cited article follows Ferguson's notation. Let us begin by establishing our notation for the model parameters.

$\boldsymbol{P_{S[t+1]|S[t]}(s \mid \tilde{s})}$ The probability that at time $t+1$ the system will be in state s given that at time t it was in state $\tilde{s}$. Notice that the parameter is independent of time t. Ferguson called these parameters *the A matrix* with $a_{i,j} = P_{S[t+1]|S[t]}(s_j \mid s_i)$.

$\boldsymbol{P_{S[1]}(s)}$ The probability that the system will start in state s at time 1. Ferguson used a to represent this vector of probabilities with $a_i = P_{S[1]}(s_i)$.

$\boldsymbol{P_{Y[t]|S[t]}(y \mid s)}$ The probability that the observation value is y given that the system is in state s. Ferguson used b to represent this with $b_j(k) = P_{Y[t]|S[t]}(y_k \mid s_j)$.

$\boldsymbol{\theta}$ The entire collection of parameters. For example, iteration of the Baum-Welch algorithm produces a sequence of parameter vectors $\theta[0:N]$ with $P(y[0:T] \mid \theta[n+1]) \geq P(y[0:T] \mid \theta[n])$: $1 < n < N$. Instead of θ, Ferguson used λ to denote the entire collection of parameters.

2.1 The Forward Algorithm

Given θ, the forward algorithm calculates $P(y[0:T] \mid \theta)$, and several useful intermediate terms. Figure 2.1 sketches the basic recursion's structure. Since we are considering only one set of parameters, we will drop the dependence of probabilities on θ from the notation for the remainder of this section. In the example of Eqn. (1.18) we found that we could write the right hand terms of

$$P\left(y[0:T]\right) = \sum_{s[0:T]} P\left(y[0:T], s[0:T]\right)$$

easily from the model assumptions. Unfortunately, the number of possible sequences $s[0:T]$ is exponential in T, and it is not feasible to do the sum for even modest lengths T.

The forward algorithm regroups the terms and produces the desired result using order T calculations. For each time $t: \ 1 < t \leq T$ we calculate $P(y[t] \mid y[0:t])$, and in principle we can write

$$P(y[0:T]) = P(y[0]) \prod_{t=1}^{T-1} P(y[t] \mid y[0:t]).$$

However the values of $P(y[t] \mid y[0:t])$ are typically small compared to 1, and the product of many such terms is too small to be represented even in double precision. Working with logarithms avoids underflow:

$$\log\left(P(y[0:T])\right) = \log\left(P(y[0])\right) + \sum_{t=1}^{T} \log\left(P(y[t] \mid y[0:t])\right). \tag{2.1}$$

For each time step we do the four calculations specified in the equations below[1]. In these equations we use color highlighting to emphasize repeated instances of the same quantity. The algorithm saves the following intermediate results

$$\alpha(t,s) \equiv P_{S[t]|Y[0:t+1]}\left(s \mid y[0:t+1]\right) \quad \text{The updated state distribution} \tag{2.2a}$$

$$\gamma[t] \equiv P(y[t] \mid y[0:t]) \quad \text{The incremental likelihood.} \tag{2.2b}$$

In words[2], $\alpha(t,s)$ is the conditional probability of being in state s at time t given all of the past observations including $y[t]$ and $\gamma[t]$ is the conditional probability of the observation at time t given all of the previous observations. We also

[1] On page 136 in Appendix C we present four lines of python code that implement these four calculations.

[2] In Ferguson's notation $\alpha(t,s) \equiv P_{S[t],Y[0:t]}\left(s, y[0:t]\right)$. Our notation differs from that by a factor of $P(y[0:t])$. Ferguson did not have notation for the term $P(y[t] \mid y[0:t])$ which we call γ, but he did use γ for quantities that we will denote by w in Eqns. (2.22) and (2.23).

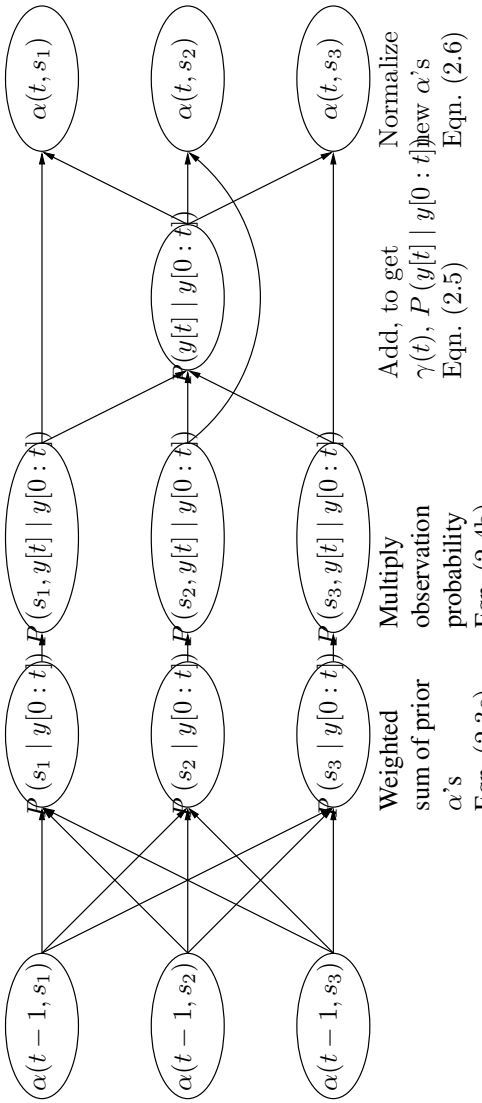

Figure 2.1: Dependency relations in the forward algorithm (See Eqns. (2.3)-(2.6) in the text).

define two intermediate terms that are not saved

$$a(s,t) \equiv P_{S[t]|Y[0:t]}\left(s \mid y[0:t]\right) \qquad \text{The state forecast} \tag{2.2c}$$

$$\tilde{a}(s,t) \equiv P_{S[t],Y[t]|Y[0:t]}\left(s, y[t] \mid y[0:t]\right) \qquad \text{The joint forecast.} \tag{2.2d}$$

To initialize the algorithm, one assigns

$$\gamma[0] = P_{Y[0]}(y[0]) = \sum_s P_{S[0]}(s) P_{Y[t]|S[t]}(y[0] \mid s)$$

$$\alpha(0,s) = P_{S[0]|Y[0]}(s \mid y[0]) = \frac{P_{S[0]}(s) P_{Y[t]|S[t]}(y[0] \mid s)}{\gamma[0]} \ \forall s \in \mathcal{S}$$

where the distributions $P_{S[0]}$ and $P_{Y[t]|S[t]}$ are model parameters. After initialization, the algorithm iterates in a loop doing the following four calculations:

Forecast the Distribution of States Find for time t and each possible state s the conditional probability, given the previous observations, of the state being s at time t:

$$\begin{aligned} P_{S[t]|Y[0:t]}\left(s \mid y[0:t]\right) &= \sum_{\tilde{s}\in\mathcal{S}} P_{S[t],S[t-1]|Y[0:t]}\left(s, \tilde{s} \mid y[0:t]\right) && \text{(2.3a)} \\ &= \sum_{\tilde{s}\in\mathcal{S}} \Big(P_{S[t]|S[t-1],Y[0:t]}\left(s \mid \tilde{s}, y[0:t]\right) \\ &\quad \times P_{S[t-1]|Y[0:t]}\left(\tilde{s} \mid y[0:t]\right)\Big) && \text{(2.3b)} \\ &= \sum_{\tilde{s}\in\mathcal{S}} P_{S[t]|S[t-1]}\left(s \mid \tilde{s}\right) \cdot \alpha(t-1,\tilde{s}). && \text{(2.3c)} \end{aligned}$$

We justify the operations as follows:

(2.3a) $P(a) = \sum_b P(a,b)$

(2.3b) $P(a \mid b) \cdot P(b) = P(a,b)$

(2.3c) Assumption of Eqn. (1.19) on page 11: The state process is Markov

Forecast the Joint Probability of States and the Current Observation Find for time t and each possible state s, the conditional probability, given the previous observations, of the state being s and the observation being $y[t]$ (the value actually observed):

$$\begin{aligned} P_{S[t],Y[t]|Y[0:t]}\left(s, y[t] \mid y[0:t]\right) &= P_{Y[t]|S[t],Y[0:t]}\left(y[t] \mid s, y[0:t]\right) \\ &\quad \times P_{S[t]|Y[0:t]}\left(s \mid y[0:t]\right) && \text{(2.4a)} \\ &= P_{Y[t]|S[t]}\left(y[t] \mid s\right) \\ &\quad \times P_{S[t]|Y[0:t]}\left(s \mid y[0:t]\right). && \text{(2.4b)} \end{aligned}$$

We justify the equations as follows:

(2.4a) $P(a \mid b) \cdot P(b) = P(a, b)$

(2.4b) Assumption of Eqn. (1.20) on page 11: The observations are conditionally independent given the states

Calculate the Conditional Probability of the Current Observation Find for time t, the conditional probability, given the previous observations, of the observation being $y[t]$ (the observation actually observed):

$$P(y[t] \mid y[0:t]) = \sum_{s \in \mathcal{S}} P_{S[t],Y[t]|Y[0:t]}\left(s, y[t] \mid y[0:t]\right). \tag{2.5}$$

Equation (2.5) is an application of $P(a) = \sum_b P(a, b)$.

Calculate the Updated Distribution of States For each possible state s, find the conditional probability of being in that state at time t given all of the observations up to time t. Note that this differs from the first calculation, (2.3), in that the conditioning event includes $y[t]$:

$$\alpha(t, s) \equiv P_{S[t]|Y[0:t+1]}\left(s \mid y[0:t+1]\right) \tag{2.6a}$$

$$= P_{S[t],Y[t]|Y[0:t]}\left(s, y[t] \mid y[0:t]\right) \div P(y[t] \mid y[0:t]) \tag{2.6b}$$

Equation (2.6b) is an application of Bayes rule, i.e., $P(a \mid b) \cdot P(b) = P(a, b)$.

Note that given the incremental likelihoods $\gamma[t]$ the forward algorithm simply alternates between multiplying a state distribution by a likelihood and multiplying the result by the matrix of state transition probabilities, i.e.,

$$a(t, s) = \sum_{\tilde{s}} P_{s[1]|s[0]}\left(s \mid \tilde{s}\right) \alpha(t-1, \tilde{s}) \tag{2.7a}$$

$$\alpha(t, s) = \frac{P(y[t]|s)a(t, s)}{\gamma[t]}. \tag{2.7b}$$

2.2 The Viterbi Algorithm

For some applications, one must estimate the sequence of states based on a sequence of observations. The Viterbi algorithm finds the *best* sequence $\hat{s}[1:T]$ in the sense of maximizing the probability $P\left(s[0:T] \mid y[0:T]\right)$. That is equivalent to maximizing $\log\left(P\left(y[0:T], s[0:T]\right)\right)$ because $P\left(y[0:T]\right)$ is simply a constant, and the log is monotonic, ie,

$$\begin{aligned}
\hat{s}[0:T] &\equiv \operatorname*{argmax}_{s[0:T]} P(s[0:T] \mid y[0:T]) \\
&= \operatorname*{argmax}_{s[0:T]} \left(P(s[0:T] \mid y[0:T]) \cdot P(y[0:T])\right) \\
&= \operatorname*{argmax}_{s[0:T]} \log\left(P(y[0:T], s[0:T])\right).
\end{aligned}$$

As in the implementation of the forward algorithm, we use logs to avoid numerical underflow. If we define $\log\left(P\left(y[0:t], s[0:t]\right)\right)$ as the *utility* (negative cost) of the state sequence $s[0:t]$ given the observation sequence $y[0:t]$, then the Viterbi algorithm finds the maximum utility state sequence.

Initially one calculates $\log\left(P_{Y[0],S[0]}\left(y[0], s\right)\right)$ for each state $s \in \mathcal{S}$. Then for each successive time step $t : 1 \leq t < T$ one considers each state and determines the best predecessor for that state and the utility of the best state sequence ending in that state. The Viterbi algorithm and the forward algorithm have similar structures (see Fig. 2.1 and Fig. 2.3). Roughly, the forward algorithm does a sum over predecessor states, while the Viterbi algorithm finds a maximum over predecessor states. We use the following notation and equations to describe and justify the algorithm.

Notation:

$\tilde{s}(t, s)$ The best $s[0:t+1]$ ending in s Of all length $t+1$ state sequences ending in s at time t, we define $\tilde{s}(t, s)$ to be the sequence with the highest joint probability with the data, i.e.,

$$\tilde{s}(t, s) \equiv \underset{s[0:t+1]:s[t]=s}{\operatorname{argmax}} P\left(y[0:t+1], s[0:t+1]\right).$$

$\nu(t, s)$ The *utility* of the best $s[0:t+1]$ ending in s This is simply the log of the joint probability of the data with the best state sequence defined above, i.e.

$$\nu(t, s) = \log\left(P\left(y[0:t+1], \tilde{s}(t, s)\right)\right)$$

$s'(t, s)$ The immediate predecessor of state s in $\tilde{s}(t, s)$ In other words, given that the best sequence of length $t+1$ that ends in s_d is

$$\tilde{s}(t, s_d) = [s[0] = s_a, s[1] = s_b, \ldots, s[t-1] = s_c, s[t] = s_d]$$

the best predecessor of s is the penultimate, i.e. at time $t-1$, element of that sequence, i.e. s_c.

Equations:

Equations (2.8) and (2.9) summarize the Viterbi algorithm. When finally deciphered, one finds Eqn. (2.8) is the vacuous statement that the best state sequence ending in s at time t consists of s concatenated with the best sequence ending in s' at time $t-1$, where s' is the best predecessor of s.

$$\tilde{s}(t, s) = [\tilde{s}(t-1, s'(t, s), s] \tag{2.8}$$

Equation (2.9) says that the total utility of the best path of length $t+1$ to state s is the utility of the best predecessor plus two terms, one that accounts for the transition from the predecessor to s and one that accounts for the probability of

state s producing the observation $y[t]$. From the model assumptions (Eqn. (1.19) and Eqn. (1.20)) we find

$$P_{Y[t+1],Y[0:t+1],S[t+1],S[0:t+1]}\left(y[t+1], y[0:t+1], s, s[0:t+1]\right) = \\ P\left(y[0:t+1], s[0:t+1]\right) \cdot P_{S[t+1]|S[t]}\left(s \mid s[t]\right) \cdot P_{Y[t+1]|S[t+1]}\left(y[t] \mid s\right).$$

Taking logs yields

$$\begin{aligned} \nu(t+1, s) = \nu(t, s'(t, s)) &+ \log\left(P_{S[t+1]|S[]}\left(s \mid s'(t, s)\right)\right) \\ &+ \log\left(P_{Y[t+1]|S[t+1]}\left(y[t] \mid s\right)\right). \end{aligned} \tag{2.9}$$

Algorithm:

Following the steps in Fig. 2.2, note that the algorithm starts by assigning a utility for each state using the first observation and then iterates forward through time. For each time step t and each possible next state s_{next} at time $t+1$, the algorithm finds and records the best predecessor state s_{best} at time t. Then the algorithm calculates the utility of s_{next} at time $t+1$ on the basis of the utility of the best predecessor, the conditional transition probability, and the conditional observation probability. At the final time T the algorithm selects the highest utility endpoint, i.e.,

$$\hat{s}[T-1] = \underset{s}{\operatorname{argmax}}\, \nu(T-1, s),$$

and then backtracks through the optimal predecessor links to produce the entire highest utility path.

2.3 The Baum-Welch Algorithm

Given an initial vector of model parameters θ for an HMM and a sequence of observations $y[0:T]$, iteration of the Baum-Welch algorithm produces a sequence of parameter vectors $\theta[1:N]$ that almost always converges to a local maximum of the likelihood function $P_\theta\left(y[0:T]\right)$. The algorithm was developed by Baum and collaborators [26, 25] in the 1960's at the Institute for Defense Analysis in Princeton. In each iteration, it estimates the distribution of the unobserved states and then maximizes the expected log likelihood with respect to that estimate. Although Baum et al. limited their attention to HMMs, the same kind of iteration works on other models that have unobserved variables. In 1977, Dempster Laird and Rubin [28] called the general procedure the *EM algorithm*.

The EM algorithm operates on models $P_{\mathbf{Y},\mathbf{S},\theta}$ with parameters θ for a mix of data that is observed ($\mathbf{Y}$) and data that is unobserved ($\mathbf{S}$). (For our application, $\mathbf{Y}$ is a sequence of observations $Y[0:T]$ and $\mathbf{S}$ is a sequence of discrete hidden states $S[0:T]$.) The steps in the algorithm are:

Initialize:

for each s

$\nu_{\text{next}}(s) = \log\left(P_{Y[0],S[0]}\left(y[0], s\right)\right)$

Iterate:

for t from 1 to T

Swap $\nu_{\text{next}} \leftrightarrow \nu_{\text{old}}$

for each s_{next}

Find best predecessor

$s_{\text{best}} = \text{argmax}_{s_{\text{old}}}\left(\nu_{\text{old}}(s_{\text{old}}) + \log\left(P_{S[t]|S[t-1]}\left(s_{\text{next}} \mid s_{\text{old}}\right)\right)\right)$

Update ν

$$\begin{aligned}\nu_{\text{next}}(s_{\text{next}}) = \ & \nu_{\text{old}}(s_{\text{best}}) \\ & + \log\left(P_{S[t]|S[t-1]}\left(s_{\text{next}} \mid s_{\text{best}}\right)\right) \\ & + \log\left(P_{Y[t]|S[t]}\left(y[t] \mid s_{\text{next}}\right)\right)\end{aligned}$$

Update predecessor array

$\text{Predecessor}[s_{\text{next}}, t] = s_{\text{best}}$

Backtrack:

$s[0:T] = \hat{s}[0:T](\bar{s})$, where $\bar{s} = \text{argmax}_s\, \nu_{\text{next}}(s)$ at $t = T-1$

Figure 2.2: Pseudocode for the Viterbi Algorithm

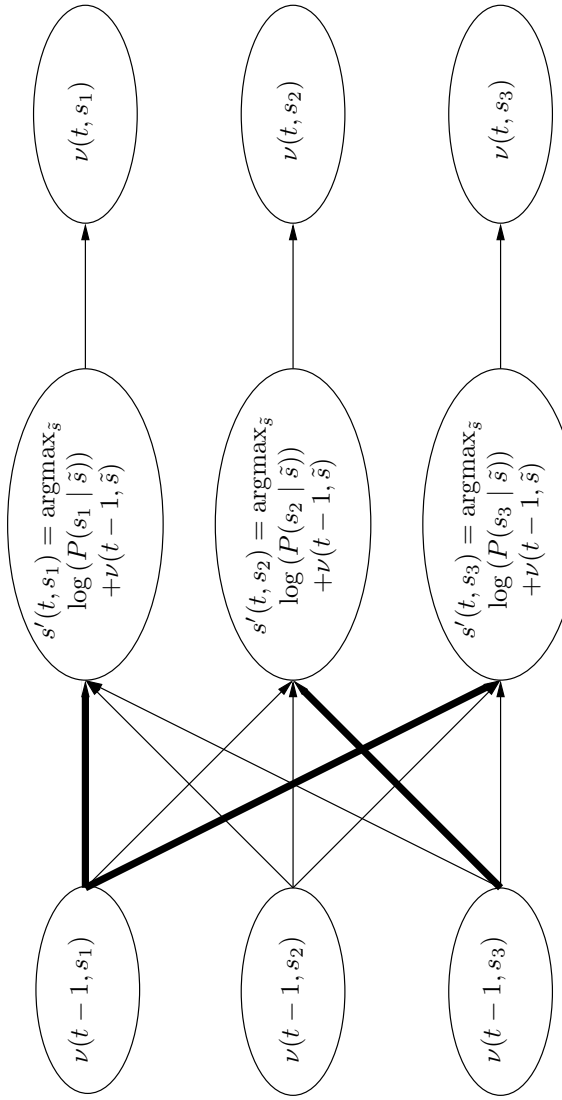

Figure 2.3: Dependency relations in the Viterbi algorithm.

1. Guess[3] a starting value of $\theta[0]$, and set $n = 0$.
2. Choose $\theta[n+1]$ to maximize an *auxiliary function* Q

$$\theta[n+1] = \underset{\theta}{\operatorname{argmax}}\, Q(\theta, \theta[n]) \tag{2.10}$$

   where

$$Q(\theta', \theta) \equiv \mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)} \left(\log P(\mathbf{y}, \mathbf{S}, \theta')\right) \tag{2.11}$$

   (Here the notation $\mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)}(F(\mathbf{S}))$ means the expected value of $F(\mathbf{S})$ over all values of $\mathbf{S}$ using the distribution $P(\mathbf{S} \mid \mathbf{y}, \theta)$.)
3. If not converged, go to 2 with $n \leftarrow n + 1$.

We defer further discussion of the general EM algorithm to Section 2.5 and now proceed to the details of its application to HMMs, i.e., the Baum-Welch algorithm. The work of an iteration of the EM algorithm is done in step 2. To apply it to an HMM, we first characterize $P(s[0:T] \mid y[0:T], \theta)$ by a combination of the *forward algorithm* that we already described in Section 2.1 and the *backward algorithm* which we will describe in the next section. Using the characterization of $P(s[0:T] \mid y[0:T], \theta)$, we describe the optimization specified by Eqn. (2.10) in Section 2.3.2.

2.3.1 The Backward Algorithm

The backward algorithm is similar to the forward algorithm in structure and complexity, but the terms are neither as easy to interpret nor as clearly useful. After running both the forward algorithm and the backward algorithm, one can calculate $P_{S[t]|Y[0:T]}(s \mid y[0:T])$, the conditional probability of being in any state $s \in \mathcal{S}$ at any time $t : 1 \leq t \leq T$ given the entire sequence of observations. The forward algorithm provides the terms $\alpha(t, s) \equiv P_{S[t]|Y[0:t]}(s \mid y[0:t])\ \forall(t, s)$. Thus the backward algorithm must provide terms, call them $\beta(t, s)$, with the values

$$\beta(t, s) = \frac{P_{S[t]|Y[0:T]}(s \mid y[0:T])}{\alpha(t, s)} = \frac{P_{S[t]|Y[0:T]}(s \mid y[0:T])}{P_{S[t]|Y[0:t]}(s \mid y[0:t])}. \tag{2.12}$$

Invoking Bayes rule and the model assumptions we find

$$\beta(t, s) = \frac{P_{Y[0:T],S[t]}(y[0:T], s) \cdot P(y[0:t])}{P_{Y[0:t],S[t]}(y[0:t], s) \cdot P(y[0:T])} \tag{2.13}$$

$$= \frac{P_{Y[t+1:T]|Y[0:t],S[t]}(y[t+1:T] \mid y[0:t], s)}{P(y[t+1:T] \mid y[0:t])} \tag{2.14}$$

$$= \frac{P_{Y[t+1:T]|S[t]}(y[t+1:T] \mid s)}{P(y[t+1:T] \mid y[0:t])}. \tag{2.15}$$

[3] Although a symmetric model in which the transition probability from each state to every other state is the same and the observation probabilities are all uniform is easy to describe, such a model is a bad choice for $\theta[1]$ because the optimization procedure will not break the symmetry of the states.

(Note that if $\alpha(t,s) = 0$, Eqn. (2.12) is undefined, but we can nonetheless implement Eqn. (2.15).)

The algorithm starts at the final time T with β set to one[4] for each state, $\beta(t,s) = 1,\ \forall s \in \mathcal{S}$, and solves for β at earlier times with the following recursion that goes *backwards* through time:

$$\beta(t-1,\tilde{s}) = \sum_{s\in\mathcal{S}} \beta(t,s) \frac{P_{Y[t]|S[t]}\left(y[t] \mid s\right) \cdot P_{S[t]|S[t-1]}\left(s \mid \tilde{s}\right)}{\gamma[t]} \tag{2.16}$$

Note that $\gamma[t] \equiv P\left(y[t] \mid y[0:t]\right)$ is calculated by the forward algorithm and that the terms in the numerator are model parameters. We justify Eqn. (2.16) by using Eqn. (2.15)

$$\beta(t-1,\tilde{s}) \equiv \frac{P_{Y[t:T]|S[t-1]}\left(y[t:T] \mid \tilde{s}\right)}{P\left(y[t:T] \mid y[0:t]\right)} \tag{2.17}$$

$$= \frac{\sum_{s\in\mathcal{S}} P_{Y[t:T],S[t]|S[t-1]}\left(y[t:T], s \mid \tilde{s}\right)}{P\left(y[t:T] \mid y[0:t]\right)}. \tag{2.18}$$

Next, factor each term in the numerator of Eqn. (2.18) using Bayes rule twice then apply the model assumptions and the expression for $\beta(t,s)$ from Eqn. (2.15):

$$\begin{aligned}
&P_{Y[t:T],S[t]|S[t-1]}\left(y[t:T], s \mid \tilde{s}\right) \\
&\quad = P_{Y[t+1:T]|Y[t],S[t],S[t-1]}\left(y[t+1:T] \mid y[t], s, \tilde{s}\right) \\
&\qquad \cdot P_{Y[t]|S[t],S[t-1]}\left(y[t] \mid s, \tilde{s}\right) \cdot P_{S[t]|S[t-1]}\left(s \mid \tilde{s}\right) \\
&\quad = P_{Y[t+1:T]|S[t]}\left(y[t+1:T] \mid s\right) \cdot P_{Y[t]|S[t]}\left(y[t] \mid s\right) \cdot P_{S[t]|S[t-1]}\left(s \mid \tilde{s}\right) \\
&\quad = \beta(t,s) \cdot P\left(y[t+1:T] \mid y[0:t]\right) \cdot P_{Y[t]|S[t]}\left(y[t] \mid s\right) \cdot P_{S[t]|S[t-1]}\left(s \mid \tilde{s}\right)
\end{aligned}$$

Similarly, simplify the denominator of Eqn. (2.18) using Bayes rule:

$$P\left(y[t:T] \mid y[0:t]\right) = P\left(y[t+1:T] \mid y[0:t]\right) \cdot P\left(y[t] \mid y[0:t]\right)$$

Finally by substituting these values into the fraction of (2.18), we verify the recursion (2.16)

$$\beta(t-1,\tilde{s}) = \frac{\sum_{s\in\mathcal{S}} \beta(t,s) \cdot P\left(y[t+1:T] \mid y[0:t]\right) \cdot P_{Y[t]|S[t]}\left(y[t] \mid s\right) \cdot P_{S[t]|S[t-1]}\left(s \mid \tilde{s}\right)}{P\left(y[t+1:T] \mid y[0:t]\right) \cdot P\left(y[t] \mid y[0:t]\right)} \tag{2.19}$$

$$= \sum_{s\in\mathcal{S}} \beta(t,s) \frac{P_{Y[t]|S[t]}\left(y[t] \mid s\right) \cdot P_{S[t]|S[t-1]}\left(s \mid \tilde{s}\right)}{\gamma[t]}. \tag{2.20}$$

By introducing the intermediate quantity $b(t,s)$ we can break (2.20) into the

[4]From the premises that $\alpha(t,s)\beta(t,s) = P_{S[t]|Y[0:T]}\left(s \mid y[0:T]\right)$ and $\alpha(t,s) \equiv P_{S(T)|Y[0:T]}\left(s \mid y[0:T]\right)$, we conclude that $\beta(t,s) = 1\ \forall s$.

two steps

$$b(s,t) = \frac{\beta(t,s)y[t]}{\gamma[t]} \tag{2.21a}$$

$$\beta(t-1,\tilde{s}) = \sum_{s\in\mathcal{S}} b(s,t) P_{S[1]|S[0]}\left(s \mid \tilde{s}\right), \tag{2.21b}$$

which clarifies that, aside from normalization, the backward algorithm like the forward algorithm alternates between multiplying a state distribution by an observation likelihood and then multiplying the result by the transition probability matrix or its transpose.

2.3.2 Weights and Reestimation

Each pass of the Baum-Welch algorithm consists of the following steps: Run the forward algorithm described in Section 2.1 to calculate the values of $\alpha(t,s)$ and $\gamma(t)$ for each time $t \in [0,\ldots,T-1]$ and each state $s \in \mathcal{S}$; Run the backward algorithm described in Section 2.3.1 to calculate the values of $\beta(t,\tilde{s})$; Reestimate the model parameters using the formulas in Table 2.1.

We write the reestimation formulas in terms of *weights* which express the conditional probability of being in specific states at specific times given the observed data $y[0:T]$. We denote the conditional probability of being in state s at time t given all of the data by

$$w(t,s) \equiv P_{S[t]|Y[0:T]}\left(s \mid y[0:T]\right), \tag{2.22}$$

and we denote the conditional probability, given all of the data, of being in state s at time t and being in state $\tilde{s}$ at time $t+1$ by

$$\tilde{w}(t,\tilde{s},s) \equiv P_{S[t+1],S[t]|Y[0:T]}\left(\tilde{s},s \mid y[0:T]\right). \tag{2.23}$$

Table 2.1 (page 33) summarizes the formulas for the updated model parameters after one pass of the Baum-Welch algorithm.

To derive reestimation formulas for $P_{S(1)}$ and $P_{Y[t]|S[t]}$ we will consider a sum over all possible state sequences $s[0:T]$, i.e.,

$$w(t,s) = \sum_{s[0:T]:s[t]=s} P\left(s[0:T] \mid y[0:T]\right). \tag{2.24}$$

Since this is virtually unimplementable, the actual algorithm uses

$$w(t,s) = \alpha(t,s)\beta(t,s). \tag{2.25}$$

In fact, in Eqn. (2.12), we chose the expression for β to make Eqn. (2.25) true.

Similarly, we will derive the reestimation formula for $P_{S[t+1]|S[t]}$ using

$$\tilde{w}(t,\tilde{s},s) = \sum_{\substack{s[0:T]:s[t+1]=\tilde{s},\\ s[t]=s}} P\left(s[0:T] \mid y[0:T]\right), \tag{2.26}$$

but in the algorithm we use

$$\tilde{w}(t,\tilde{s},s) = \frac{\alpha(t,s)\cdot P_{S[t+1]|S[t]}\left(\tilde{s}\mid s\right)\cdot P_{Y[t+1]|S[t+1]}\left(y[t+1]\mid\tilde{s}\right)\cdot\beta(t+1,\tilde{s})}{\gamma[t+1]}. \tag{2.27}$$

One can verify Eqn. (2.27) using the model assumptions, the definitions of α, β, and γ, and Bayes rule.

Reestimation

With Eqns. (2.25) and (2.27) for $w(t,i)$ and $\tilde{w}(t,\tilde{s},s)$ in terms of known quantities (α, β, γ, $y[0:T]$, and the old model parameters $\theta[n]$), we are prepared to use the formulas in Table 2.1 to calculate new estimates of the model parameters, $\theta[n+1]$ with higher likelihood.

Table 2.1: Summary of reestimation formulas.
Note that formulas for $w(t,s)$ and $\tilde{w}(t,\tilde{s},s)$ appear in Eqns. (2.25) and (2.27) respectively.

Description	Expression	New Value
Initial State Prob.	$P_{S[0]\mid\theta[n+1]}\left(s\mid\theta[n+1]\right)$	$w(0,s)$
State Transition Prob.	$P_{S[t+1]\mid S[t],\theta[n+1]}\left(\tilde{s}\mid s,\theta[n+1]\right)$	$\frac{\sum_{t=1}^{T-1}\tilde{w}(t,\tilde{s},s)}{\sum_{s'\in\mathcal{S}}\sum_{t=1}^{T-1}\tilde{w}(t,s',s)}$
Cond. Observation Prob.	$P_{Y[t]\mid S[t],\theta[n+1]}\left(y\mid s,\theta[n+1]\right)$	$\frac{\sum_{t:y[t]=y} w(t,s)}{\sum_t w(t,s)}$

To derive the formulas in Table 2.1, start with Step 2 of the EM algorithm (see Eqn. (2.11)) which is to maximize the auxiliary function

$$Q(\theta',\theta) \equiv \mathbb{E}_{P(\mathbf{S}|\mathbf{y}),\theta}\left(\log P(\mathbf{y},\mathbf{S}\mid\theta')\right)$$

with respect to θ'. For an HMM, substitute the sequence of *hidden* states $s[0:T]$ for $\mathbf{S}$ and the sequence of observations $y[0:T]$ for $\mathbf{y}$. Note that the joint probability of a state sequence $s[0:T]$ and the observation sequence $y[0:T]$ is

$$P\left(s[0:T],y[0:T]\right) = P_{S[0]}\left(s[0]\right)\cdot\prod_{t=1}^{T-1}P_{S[1]|S[0]}\left(s[t]\mid s[t-1]\right)\cdot\prod_{t=0}^{T-1}P_{Y[0]|S[0]}\left(y[t]\mid s[t]\right),$$

or equivalently,

$$\log P\left(y[0:T],s[0:T]\right) = \log P_{S[0]}\left(s[0]\right) + \sum_{t=1}^{T-1}\log P_{S[1]|S[0]}\left(s[t]\mid s[t-1]\right) + \sum_{t=0}^{T-1}\log P_{Y[0]|S[0]}\left(y[t]\mid s[t]\right).$$

We can optimize Q by breaking it into a sum in which each of the model parameters only appears in one of the terms and then optimizing each of the terms independently:

$$Q(\theta', \theta) = \sum_{s[0:T] \in \mathcal{S}^T} \left(P\left(s[0:T] \mid y[0:T], \theta\right) \log P\left(s[0:T], y[0:T], \theta'\right)\right) \quad (2.28)$$

$$\equiv Q_{\text{initial}}(\theta', \theta) + Q_{\text{transition}}(\theta', \theta) + Q_{\text{observation}}(\theta', \theta), \quad (2.29)$$

where

$$Q_{\text{initial}}(\theta', \theta) \equiv \sum_{s[0:T] \in \mathcal{S}^T} \left(P\left(s[0:T] \mid y[0:T], \theta\right) \log P_{S[0]|\theta'}\left(s[0] \mid \theta'\right)\right) \quad (2.30)$$

$$Q_{\text{transition}}(\theta', \theta) \equiv \sum_{s[0:T] \in \mathcal{S}^T} \left(P\left(s[0:T] \mid y[0:T], \theta\right) \sum_{t=1}^{T-1} \log P_{S[1]|S[0],\theta'}\left(s[t] \mid s[t-1], \theta'\right)\right) \quad (2.31)$$

$$Q_{\text{observation}}(\theta', \theta) \equiv \sum_{s[0:T] \in \mathcal{S}^T} \left(P\left(s[0:T] \mid y[0:T], \theta\right) \sum_{t=1}^{T} \log P_{Y[0]|S[0],\theta'}\left(y[t] \mid s[t], \theta'\right)\right) \quad (2.32)$$

To simplify the appearance of expressions as we optimize Q, we introduce notation for logs of parameters

$$L_{\text{initial}}(i) \equiv \log P_{S[0]|\theta'}\left(i \mid \theta'\right) \quad (2.33)$$

$$L_{\text{transition}}(i, j) \equiv \log P_{S[1]|S[0],\theta'}(i \mid j, \theta') \quad (2.34)$$

$$L_{\text{observation}}(y, i) \equiv \log P_{Y[0]|S[0],\theta'}\left(y \mid i, \theta'\right). \quad (2.35)$$

Now to optimize Q_{initial} write Eqn. (2.30) as

$$Q_{\text{initial}}(\theta', \theta) = \sum_{s[0:T] \in \mathcal{S}^T} P\left(s[0:T] \mid y[0:T], \theta\right) L_{\text{initial}}(s[0]) \quad (2.36)$$

$$= \sum_{s \in \mathcal{S}} L_{\text{initial}}(s) \sum_{s[0:T]:s[0]=s} P\left(s[0:T] \mid y[0:T], \theta\right) \quad (2.37)$$

$$= \sum_{s} L_{\text{initial}}(s) P_{S[0]|Y[0:T],\theta}\left(s \mid y[0:T], \theta\right) \text{ see Eqn. (2.24)} \quad (2.38)$$

$$= \sum_{s} L_{\text{initial}}(s) w(0, s) \quad (2.39)$$

We wish to find the set $\{L_{\text{initial}}(s)\}$ that maximizes $Q_{\text{initial}}(\theta', \theta)$ subject to the constraint

$$\sum_{s} e^{L_{\text{initial}}(s)} \equiv \sum_{s} P_{S[0]|\hat{\theta}}\left(s \mid \hat{\theta}\right) = 1.$$

The method of Lagrange multipliers yields

$$L_{\text{initial}}(s) = \log w(0, s) \ \forall s, \tag{2.40}$$

ie, the new estimates of the initial probabilities are

$$P_{S[0]|\theta[n+1]}\left(s \mid \theta[n+1]\right) = w(0, s). \tag{2.41}$$

To derive new estimates of the state transition probabilities, write

$$Q_{\text{transition}}(\theta', \theta) = \sum_{s[0:T] \in \mathcal{S}^T} P\left(s[0:T] \mid y[0:T], \theta\right) \sum_{t=1}^{T-1} L_{\text{transition}}(s[t-1], s[t]) \tag{2.42}$$

$$= \sum_{s, \tilde{s}} L_{\text{transition}}(s, \tilde{s}) \sum_{t=1}^{T-1} \sum_{\substack{s[0:T]: s[t]=\tilde{s}, \\ s[t-1]=s}} P\left(s[0:T] \mid y[0:T], \theta\right) \tag{2.43}$$

$$= \sum_{s, \tilde{s}} L_{\text{transition}}(s, \tilde{s}) \sum_{t=1}^{T-1} \tilde{w}(t, \tilde{s}, s). \tag{2.44}$$

Optimization yields

$$P_{S[1]|S[0],\theta[n+1]}\left(\tilde{s} \mid s, \theta[n+1]\right) = \frac{\sum_{t=1}^{T-1} \tilde{w}(t, \tilde{s}, s)}{\sum_{s'} \sum_{t=1}^{T-1} \tilde{w}(t, s', s)}. \tag{2.45}$$

Similarly, we derive the new estimates of the conditional observation probabilities from

$$Q_{\text{observation}}(\theta', \theta) = \sum_{s[0:T] \in \mathcal{S}^T} P\left(s[0:T] \mid y[0:T], \theta\right) \sum_{t=1}^{T-1} L_{\text{observation}}(y[t], s[t]) \tag{2.46}$$

$$= \sum_{y \in \mathcal{Y}, s \in \mathcal{S}} L_{\text{observation}}(y, s) \sum_{t: y[t]=y} \sum_{s[0:T]: s[t]=s} P\left(s[0:T] \mid y[0:T], \theta\right) \tag{2.47}$$

$$= \sum_{y, s} L_{\text{observation}}(y, s) \sum_{t: y[t]=y} w(t, s). \tag{2.48}$$

Optimization yields

$$P_{Y[t]|S[t],\theta[n+1]}\left(y \mid s, \theta[n+1]\right) = \frac{\sum_{t: y[t]=y} w(t, s)}{\sum_{t=1}^{T-1} w(t, s)}. \tag{2.49}$$

Notation:

$\theta[n]$ is the model, or equivalently the set of parameters, after n iterations of the Baum-Welch algorithm.

$\boldsymbol{\alpha}_n$ is the set of conditional state probabilities calculated on the basis of the n^{th} model and the data $y[0:T]$. See Eqns. (2.2a) and (2.6).

$$\boldsymbol{\alpha}_n \equiv \left\{ P_{S(t)|Y[0:t],\theta[n]}\left(s \mid y[0:t], \theta[n]\right) : \forall s \in \mathcal{S} \,\&\, 0 \leq t < T \right\}$$

$\boldsymbol{\beta}_n$ is a set of values calculated on the basis of the n^{th} model $\theta[n]$ and the data $y[0:T]$. See Eqns. (2.15) and (2.16).

$$\boldsymbol{\beta}_n \equiv \left\{ \frac{P_{Y[t+1:T]|S[t]}\left(y[t+1:T] \mid s\right)}{P\left(y[t+1:T] \mid y[0:t]\right)} : \forall s \in \mathcal{S} \,\&\, 0 \leq t < T \right\}$$

$\boldsymbol{\gamma}_n$ is the set of conditional observation probabilities calculated on the basis of the n^{th} model $\theta[n]$ and the data $y[0:T]$. See Eqns. (2.2b) and (2.5).

$$\boldsymbol{\gamma}_n \equiv \left\{ P\left(y[t] \mid y[0:t], \theta[n]\right) : \, 0 \leq t < T \right\}$$

Initialize:

Set $n = 0$ and choose $\theta[0]$

Iterate:

$(\boldsymbol{\alpha}_n, \boldsymbol{\gamma}_n) \leftarrow \text{forward}(y[0:T], \theta[n])$	See Section 2.1 page 22
$\boldsymbol{\beta}_n \leftarrow \text{backward}(\boldsymbol{\gamma}_n, y[0:T], \theta[n])$	See Section 2.3.1 page 30
$\theta[n+1] \leftarrow \text{reestimate}\left(y[0:T], \boldsymbol{\alpha}_n, \boldsymbol{\beta}_n, \boldsymbol{\gamma}_n, \theta[n]\right)$	See Table 2.1 page 33
$n \leftarrow n + 1$	
Test for completion	

Figure 2.4: Summary and pseudo-code for optimizing model parameters by iterating the Baum-Welch algorithm.

2.4 Remarks

2.4.1 MAP Sequence of States or Sequence of MAP States?

Consider the difference between the sequence of maximum a posteriori states and the maximum a posteriori sequence of states. The maximum a posteriori state at a particular time t is the best guess for where the system was at that time given all of the observations, i.e. $\hat{s}[t] = \text{argmax}_{s'} P(s'[t] \mid y[0:T]) = \text{argmax}_{s'} \alpha(t,s')\beta(t,s')$ (see (2.22) – (2.25)). While it seems reasonable that a sequence of such guesses would constitute a good guess for the entire trajectory, it is not an optimal trajectory estimate. In fact, as the following example demonstrates, such a trajectory may even be impossible.

Consider the HMM drawn in Fig. 2.5 and the sequence of observations $y[0:6] = (a,b,b,b,b,c)$. Any sequence of states that is consistent with $y[0:6]$ must begin in e, end in g, and pass through state f exactly once. The only unknown remaining is the time at which the system was in state f. Here is a tabulation of the four possible state sequences:

$s[1]$	$s[2]$	$s[3]$	$s[4]$	$s[5]$	$s[6]$	$P(y[0:6], s[0:6])/z$	$P(s[0:6] \mid y[0:6])$
e	e	e	e	f	g	0.9^3	0.30
e	e	e	f	g	g	$0.9^2 \cdot 0.8$	0.26
e	e	f	e	g	g	$0.9 \cdot 0.8^2$	0.23
e	f	g	g	g	g	0.8^3	0.21

In the table, the term z represents the factors that are common in $P(y[0:6], s[0:6])$ for all of the possible state sequences. Only the factors that are different appear in the seventh column. The largest entry in the last column, where only two significant figures appear, is 0.30 which corresponds to the MAP estimate: $\hat{s}[1:6] = (e,e,e,e,f,g,)$.

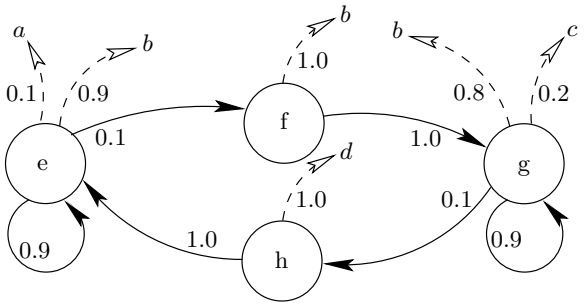

Figure 2.5: HMM used to illustrate that the maximum a posteriori sequence of states is not the same as the sequence of maximum a posteriori states.

The next table displays the values of $P(s[t] \mid y[0:6])$, the a posteriori probability for the three possible states:

$P(s[t] \mid y[0:6])$		t = 1	2	3	4	5	6
	e	**1.0**	**0.79**	**0.56**	0.30	0	0
s	f	0	0.21	0.23	0.26	0.30	0
	g	0	0	0.21	**0.44**	**0.70**	**1.0**

The table quantifies the intuition that the a posteriori probability starts in state e at time $t = 1$ and sort of diffuses completely over to state g by time $t = 6$. Notice that although all of the probability passes through state f, at no time is it the most probable state. Thus the sequence of maximum a posteriori states is e, e, e, g, g, g which is an *impossible* sequence. On the other hand, the maximum a posteriori sequence of states, e, e, e, e, f, g, is entirely plausible.

2.4.2 Training on Multiple Segments

Simple modifications to code for the Baum-Welch algorithm enable it to train on data that consists of a collection of independent segments $\tilde{\mathbf{y}} \equiv \{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{n-1}\}$ where $\mathbf{y}_k = y_k[0 : T_k]$. In particular for each iteration, one should:

- Run the forward and backward algorithms on each segment $\mathbf{y}_k$ to calculate $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$
- Create $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by concatenating $\boldsymbol{\alpha}_k$ $\forall k$ and $\boldsymbol{\beta}_k$ $\forall k$ respectively.
- Reestimate all model parameters by applying the formulas in Table 2.1 to the concatenated $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\tilde{\mathbf{y}}$.
- Modify the reestimated initial state probabilities using

$$P_{S[0]|\theta[m+1]}\left(s \mid \theta[m+1]\right) = \frac{1}{n}\sum_{k=0}^{n-1} \alpha_k(s,0)\beta_k(0,s)$$

2.4.3 Probabilities of the initial state

Using the procedure of the previous section for a few independent observation sequences with several iterations of the Baum-Welch algorithm produces a model in which the estimates of the probabilities of the initial states, $P_{S[0]}(s)\ \forall s \in \mathcal{S}$, reflect the characteristics at the beginning of the given sequences. Those estimates are appropriate if all observation sequences come from state sequences that start in a similar fashion. Such models are not stationary. To accommodate the many applications in which we wish to model the state dynamics as stationary, we also calculate stationary initial state probability estimates using

$$P_{S[0](\text{stationary})}(s) = \frac{\sum_t w(t,i)}{\sum_{j,t} w(t,j)} \tag{2.50}$$

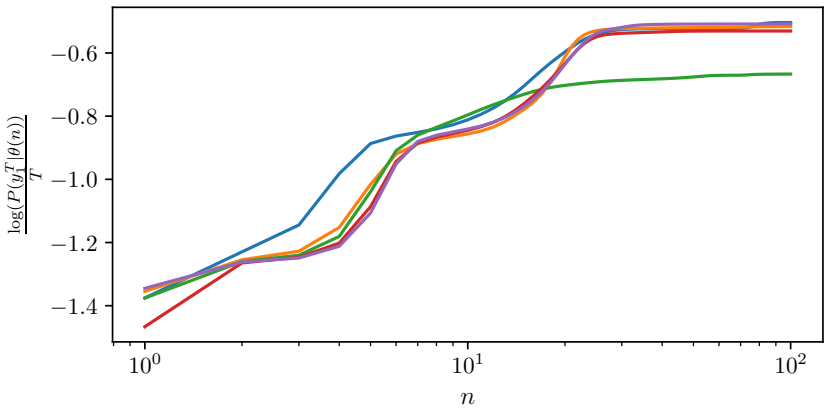

Figure 2.6: Convergence of the Baum-Welch algorithm. Here we have plotted the log likelihood per step as a function of the number of iterations n of the Baum-Welch algorithm for five different initial models $\theta[0]$. We used the same sequence of observations $y[0:T]$ that we used for Fig. 1.10, and we used different seeds for a random number generator to make the five initial models. Note the following characteristics: The five different initial models all converge to different models with different likelihoods; The curves intersect each other as some models improve more with training than others; Convergence is difficult to determine because some curves seem to have converged for many iterations and later rise significantly.

2.4.4 Maximizing likelihood over unrealistic classes

We often fit simple hidden Markov models to data that come from systems that have complicated continuous state spaces. For example, in Chapter 6 we fit models with roughly 10 states to electrocardiograms even though we believe that partial differential equations over vector fields better describe physiological dynamics that affect the signal. By fitting unrealistically simple models we reduce the variance of the parameter estimates at the expense of having less accurate models and parameters that are harder to interpret. It is a version of the classic bias-variance trade-off.

2.4.5 Multiple Local Maxima

The Baum-Welch algorithm generically converges to a *local* maximum of the likelihood function. For example, we obtained the model used to generate Fig. 1.10 by iterating the Baum-Welch algorithm on an initial model with random parameters. By re-running the experiment with five different seeds for the random number generator, we obtained the five different results that appear in Fig. 2.6.

2.4.6 Disappearing Transitions

For some observation sequences $y[0:T]$ and initial models, multiple iterations of the Baum-Welch algorithm leads to state transition probabilities that are too small to be represented in double precision. In our implementation of the Baum-Welch algorithm, we set those transition probabilities to zero. Such pruning:

- Prevents numerical underflow exceptions
- Simplifies the models
- Lets the code that process the models run faster when we use sparse matrices

There are methods (see for example [38]) for addressing situations in which one believes that some transitions should be allowed by a model even though maximizing the likelihood by the Baum-Welch algorithm would numerically drive their probability to zero.

2.4.7 Bayesian Estimates Instead of Point Estimates

A Bayesian parameter estimation scheme begins with an *a priori* distribution P_θ that characterizes knowledge about what values are possible and then uses Bayes rule to combine that knowledge with observed data y to calculate an *a posteriori* distribution of parameters $P_{\theta|y}$. We don't really believe that the maximum likelihood estimate (MLE) produced by the Baum-Welch algorithm is precisely the *one true answer*. It is what Bayesians call a *point estimate*. Parameter values near the MLE are just as plausible given the data. A proper Bayesian procedure characterizes the plausibility of other parameter values with an *a posteriori* distribution. In the next chapter, we present a variant on the Baum-Welch algorithm that uses a prior distribution on parameter values and produces an estimate that maximizes the *a posteriori* probability, i.e., a *MAP* estimate. However, like the MLE, the MAP is a point estimate. Neither does a good job of characterizing the set of plausible parameters.

In addition to yielding only a point estimate, the Baum-Welch algorithm is indirect in that each pass optimizes an auxiliary function rather than optimizing the likelihood, and it converges to *local* maxima. A Bayesian *Markov chain Monte Carlo* approach would address all of these objections at the expense of being slow.

Although others have obtained Bayesian *a posteriori* parameter distributions for HMMs using *Markov chain Monte Carlo* and *variational Bayes* procedures (see for example [41] and [23]), we will restrict our attention to point estimates from the Baum-Welch algorithm and simple variations.

2.5 The EM algorithm

In Section 2.3, we described the Baum-Welch algorithm for finding parameters of an HMM that maximize the likelihood, and we noted that the Baum-Welch al-

gorithm is a special case of the EM algorithm. Here, we examine the general EM algorithm in more detail. Readers willing to accept the Baum-Welch algorithm without further discussion of Eqn. 2.10 should skip this section. Dempster Laird and Rubin[28] coined the term *EM algorithm* in 1977. More recent treatments include Redner and Walker[42], McLachlan and Krishnan[10] and Watanabe and Yamaguchi[15]. Recall that the algorithm operates on models $P_{\mathbf{Y},\mathbf{S},\theta}$ with parameters θ for a mix of data that is observed ($\mathbf{Y}$) and data that is unobserved ($\mathbf{S}$) and that the steps in the algorithm are: Want 2024 recent

1. Guess a starting value of $\theta[0]$, and set $n = 0$.
2. Choose $\theta[n+1]$ to maximize an *auxiliary function* Q

$$\theta[n+1] = \underset{\theta}{\operatorname{argmax}}\, Q(\theta, \theta[n]) \tag{2.51}$$

   where

$$Q(\theta', \theta) \equiv \mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)} \left(\log P(\mathbf{y}, \mathbf{S} \mid \theta')\right) \tag{2.52}$$

3. Increment n.
4. If not converged, go to 2.

Step 2 does all of the work. Note that if the unobserved data ($\mathbf{S}$) is discrete, then the auxiliary function (Q) is $\mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)} \left(\log P(\mathbf{y}, \mathbf{S} \mid \theta')\right) = \sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{y}, \theta) \left(\log P(\mathbf{y}, \mathbf{s} \mid'_\theta)\right)$. Although Dempster Laird and Rubin [28] called the characterization of $P\left(\mathbf{S} \mid \mathbf{y}, \theta\right)$ the *estimation step* and the optimization of $Q(\theta, \theta[n])$ over θ the *maximization step*, the steps are now referred to as *expectation* and *maximization*.

The advantages of the EM algorithm are that it is easy to implement and it monotonically increases the likelihood. These often outweigh its slow convergence and the fact that it calculates neither the second derivative of the likelihood function nor any other indication of the reliability of the results it reports. Proving monotonicity is simple. If the likelihood is bounded, convergence follows directly from monotonicity, but convergence of the parameters does not follow. Also, the likelihood might converge to a local maximum. Papers by Baum et al.[26], Dempster, Laird, and Rubin[28], and Wu[47] analyze the issues. In the next two subsections we touch on some of the ideas and analyze an example.

2.5.1 Monotonicity

Denoting the log likelihood of the observed data given the model θ' as

$$L(\theta') \equiv \log\left(P\left(\mathbf{y} \mid \theta'\right)\right)$$

and the cross entropy of the unobserved data with respect to a model θ' given a model θ as

$$H(\theta, \theta') \equiv -\mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)} \left(\log P(\mathbf{S} \mid \mathbf{y}, \theta')\right),$$

we can write the auxiliary function as

$$\begin{aligned} Q(\theta', \theta) &\equiv \mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)} \left(\log P(\mathbf{S}, \mathbf{y} \mid \theta') \right) \\ &= \mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)} \left(\log P(\mathbf{S} \mid \mathbf{y}, \theta') + \log \left(P(\mathbf{y} \mid \theta') \right) \right) \\ &= L(\theta') - H(\theta, \theta') \end{aligned}$$

or

$$L(\theta') = Q(\theta', \theta) + H(\theta, \theta'). \tag{2.53}$$

The fact that

$$H(\theta, \theta') \geq H(\theta, \theta) \forall \theta' \tag{2.54}$$

with equality *iff*

$$P(\mathbf{s} \mid \mathbf{y}, \theta') = P(\mathbf{s} \mid \mathbf{y}, \theta) \quad \forall \mathbf{s}$$

is called the *Gibbs Inequality*[5]. It is a consequence of Jensen's inequality.

Given the model $\theta[n]$ after n iterations of the EM algorithm, we can write

$$L(\theta[n]) = Q(\theta[n], \theta[n]) + H(\theta[n], \theta[n]). \tag{2.55}$$

If for some other model θ'

$$Q(\theta', \theta[n]) > Q(\theta[n], \theta[n]), \tag{2.56}$$

then the Gibbs inequality implies $H(\theta[n], \theta') \geq H(\theta[n], \theta[n])$ and consequently, $L(\theta') > L(\theta[n])$. Monotonicity of the log function further implies

$$P\left(\mathbf{y} \mid \theta'\right) > P\left(\mathbf{y} \mid \theta[n]\right).$$

Since the EM algorithm requires the inequality in (2.56), for $\theta' = \theta[n+1]$, the algorithm monotonically increases the likelihood.

2.5.2 Convergence

Here we analyze how the Baum-Welch algorithm converges in a simple example. Some of the analysis applies to EM algorithms in general. An EM algorithm operates on Θ, a set of allowed parameter vectors with a map $\mathcal{T} : \Theta \mapsto \Theta$ that implements an iteration of the algorithm, i.e.,

$$\theta[n+1] = \mathcal{T}(\theta[n]).$$

Wu[47] has considered convergence of EM algorithms generally and observed that more than one value of θ' may maximize $Q(\theta', \theta)$. Consequently he considered $\mathcal{T}$ to be a point to set map. However, there are many model classes –including all of the models that we describe in this book– for which one can write algorithms that calculate a unique $\mathcal{T}(\theta)$.

[5]While many statisticians attribute this inequality to Kullback and Leibler[36], it appears earlier in chapter *XI Theorem II* of Gibbs[3].

If there is a bound[6] $\bar{P}$ on $P(\mathbf{y} \mid \theta)$ with

$$P(\mathbf{y} \mid \theta) \leq \bar{P} \; \forall \theta \in \Theta$$

then the monotonicity of the sequence $(P(\mathbf{y} \mid \theta[1]), P(\mathbf{y} \mid \theta[2]), P(\mathbf{y} \mid \theta[3]), \ldots)$ and the bounded convergence theorem ensures that the limit $P^* \equiv \lim_{n\to\infty} P(\mathbf{y} \mid \theta[n])$ exists. The bounded convergence theorem does not promise that $P^* = \sup_{\theta\in\Theta} P(\mathbf{y} \mid \theta)$ or that it is even a local maximum, nor does it promise that $\theta^* \equiv \lim_{n\to\infty} \theta[n]$ exists. However, in the following example, iterations of $\mathcal{T}$ converge to a local maximum[7] of the likelihood, and we believe it provides good intuition for behavior of the Baum-Welch algorithm.

Consider the model defined by the initial state probability

$$P_{S[0]} = \left[\tfrac{1}{2}, \quad \tfrac{1}{2}\right], \tag{2.57a}$$

the state transition probability

		$S[t+1]$	
$P(s[t+1]\|s[t])$		0	1
	0	.9	.1
$S[t]$	1	u	$1-u$,

(2.57b)

and the observation probability

		$Y[t]$	
$P(y[t]\|s[t])$		0	1
	0	.9	.1
$S[t]$	1	v	$1-v$.

(2.57c)

A schematic of the model appears in Figure 2.7. We set

$$\theta_{\text{true}} \equiv (u_{\text{true}}, v_{\text{true}}) = (0.1, 0.2) \tag{2.58}$$

and simulated the model for 10,000 time steps to produce $y[0:10,000]$. Then starting with

$$\theta_{\text{initial}} \equiv (u_{\text{initial}}, v_{\text{initial}}) = (0.001, 0.01) \tag{2.59}$$

we ran fifty iterations of the Baum-Welch algorithm to calculate an estimate of θ^*. During training we held all the model parameters fixed at their true values except u and v .

Figure 2.8 illustrates the trajectory in the parameter space, $U \times V$, of the first eleven training iterations. The eigenvectors and eigenvalues that appear in the lower plot of that figure are from a spectral decomposition of the second term in the Taylor series expansion $\mathcal{T}$ near θ^*, ie,

$$\mathcal{T}(\theta) = \theta^* + D(\theta - \theta^*) + \text{Remainder}.$$

[6] See Section 3.1.2 for an example involving probability densities with no such bound. For discrete $\mathbf{y}$, however the bound $P(\mathbf{y} \mid \theta) \leq 1$ holds.

[7] While Wu[47] provides an example of a trajectory that converges to a saddle point of the likelihood, we argue in the appendix that such trajectories are not generic.

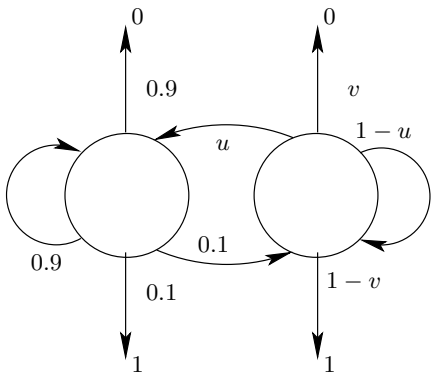

Figure 2.7: A schematic of the HMM defined by Equations (2.57). We chose to have only two free parameters in the model, u and v, so that we can illustrate the convergence of training with 2-d plots as we have in Figure 2.8.

In Appendix B on page 132 we derive

$$D \equiv \left.\frac{\partial \mathcal{T}(\theta)}{\partial \theta}\right|_{\theta^*} = \left[J_y + I_{S|y}\right]^{-1} I_{S|y}.$$

To evaluate D one can obtain the term $I_{S|y}$ from $P(s \mid y, \theta)$ which comes from the forward and backwards algorithms. On the other hand, we find

$$J_y \equiv -\frac{\partial^2}{\partial \theta^2} \log\left(P(y \mid \theta\right),$$

which is called the *observed information*, more difficult to calculate. For the figure, we approximated J_y with numerically expensive calculations.

Notice that the trajectory quickly lines up with the eigenvector of D whose eigenvalue is closest to 1 and then decays exponentially towards θ^*. We believe that behavior is typical. In the appendix we also show that if all of the eigenvalues of J_y are positive[8] then $\mathcal{T}$ is linearly stable, i.e., $|\lambda| < 1$ for all eigenvalues of D. If one could cheaply estimate the spectral decomposition of D, one could speed up the convergence of HMM training.

[8]Positive eigenvalues implies that θ^* is a local maximum.

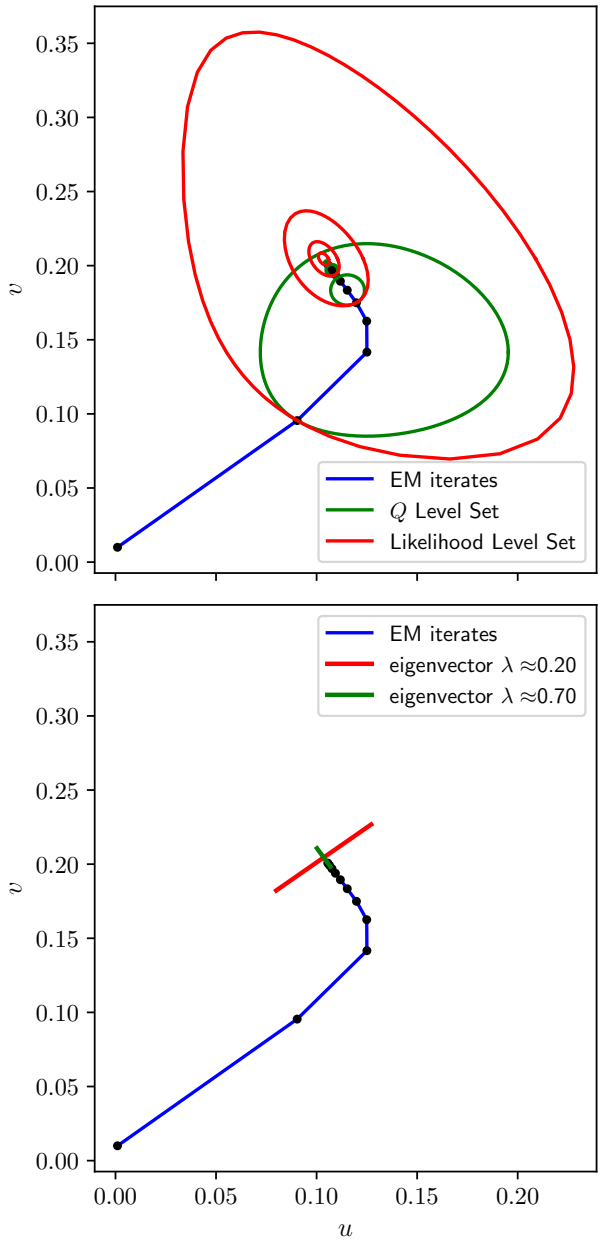

Figure 2.8: An illustration of the EM algorithm converging to a maximum likelihood estimate θ^*. The data $y[0:10,000]$ come from simulating the HMM described in (2.57) and Figure 2.7. The first eleven estimates of $\theta \equiv (u, v)$ in the training sequence appear as a blue trajectory in both plots. In the upper plot level sets of the log-likelihood, $L(\theta') = \log\left(P(y[0:10,000] \mid \theta')\right)$, and auxiliary function, $Q(\theta', \theta[n])$ appear for every third value of n. Notice that every value of θ' except $\theta[n]$ on the level set of $Q(\theta', \theta[n])$ is inside of (and therefore has higher likelihood) the level set of L. The eigenvectors depicted in the lower plot are from a spectral decomposition of the derivative $\left.\frac{\partial \mathcal{T}(\theta)}{\partial \theta}\right|_{\theta^*}$ given by Equation (B.24) on page 132 of the appendix. The lengths of the vectors are proportional to the logs of the eigenvalues.

Chapter 3

Variants and Generalizations

Hidden Markov models are special cases of discrete time state space models characterized by a state transition probability function and an observation probability function, i.e.,

$$P_{S_{n+1}|S_n}, \text{ and} \tag{3.1a}$$

$$P_{Y_n|S_n}. \tag{3.1b}$$

In Chapter 2 we described algorithms for fitting and using the probability distributions specified in Eqn. (3.1) if both the set of possible states $\mathcal{S}$ and the set of possible observations $\mathcal{Y}$ have an unstructured finite number of discrete values. However, in many applications the measurements, and perhaps the states also, are thought of as being drawn from continuous vector spaces.

Since most experimental observations are measured and recorded digitally, one could argue that discrete approximations are adequate and attempt to use the algorithms of Chapter 2 anyway. That approach is disastrous because it precludes exploiting either the metric or topological properties of the space of measurements. Consider the histogram of the first 600 samples of Tang's laser data in Fig. 3.1. Neither 5 nor 93 occurs, but it seems more plausible that 93 will occur in the remainder of the samples because there are 14 occurrences between 90 and 96 and none between 2 and 8. To make more effective use of measured data, one usually approximates the probabilities by functions with a small number of free parameters. For many such families of *parametric models* one can use the algorithms of Chapter 2 with minor modifications[1]. For a practitioner, the challenge is to find or develop both a parametric family that closely matches the measured system and algorithms for fitting and using the models.

In this chapter we will describe some model families with Gaussian observations. We will use the failure of the maximum likelihood approach with such

[1] At the 1988 ICASSP meeting, Poritz[17] reviewed several HMM variants.

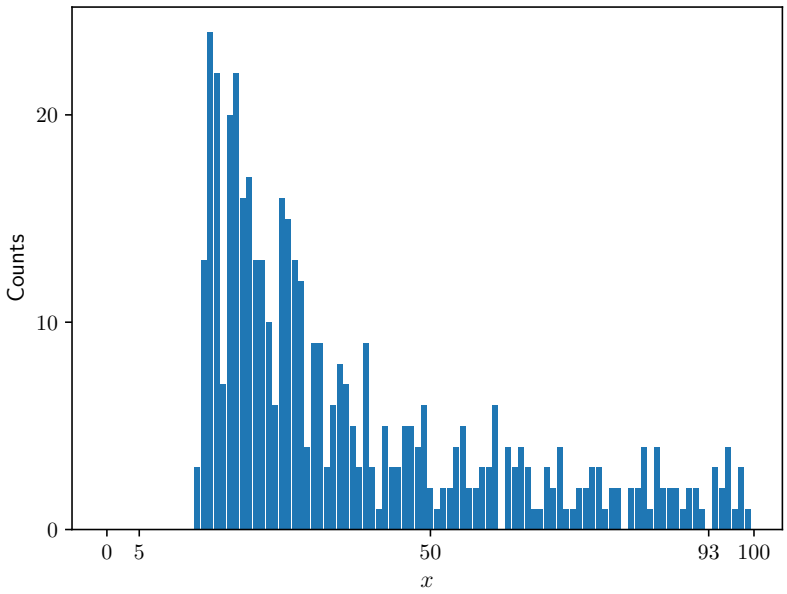

Figure 3.1: Histogram of Tang's laser measurements. Even though neither $y = 5$ nor $y = 93$ occurs in $y[0 : 600]$, it is more plausible that $y = 93$ would occur in future measurements because of what happens in the neighborhood. Discarding the numerical significance of the bin labels would preclude such an observation.

models to motivate and develop *regularization*. Also, we will touch on the relationships between HMM model families and other kinds of models.

3.1 Gaussian Observations

3.1.1 Independent Scalar Observations

A simple model for continuously distributed measurements is an HMM with an independent scalar Gaussian observation model associated with each state. In many cases it is adequate, but risky (See Fig. 3.3), to simply use the algorithms of Chapter 2 with minor modifications for Gaussian observations. Such algorithms performed satisfactorily for the exercises depicted in Fig. 3.2 in which we estimated an approximate state sequence and model parameters from a sequence of observations.

The code that generated the data for Fig. 3.2 implemented algorithms from Chapter 2 with the following modifications:

$\boldsymbol{P_{Y[t]|S[t]}(y \mid s)}$ The Viterbi algorithm, the forward algorithm, and the backward algorithm all use the observation probability conditioned on the state. In each case one simply uses the value of the probability density conditioned on the state

$$P_{Y[t]|S[t]}(y \mid s) = \frac{1}{\sqrt{2\pi\sigma_s^2}} e^{-\frac{(y-\mu_s)^2}{2\sigma_s^2}}.$$

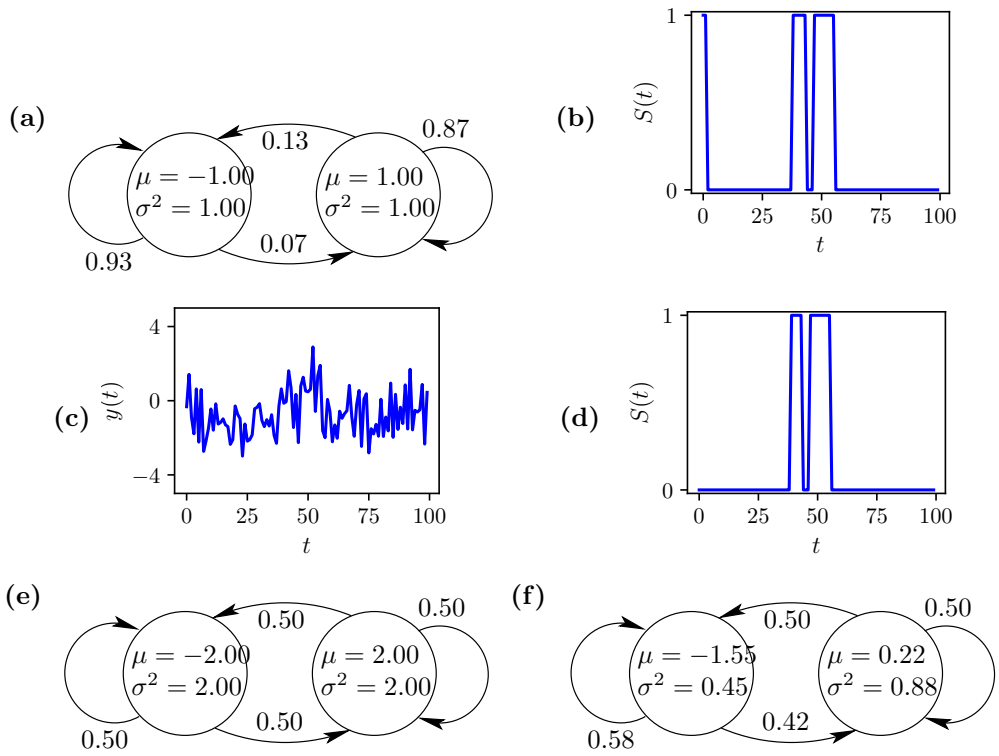

Figure 3.2: An HMM with scalar Gaussian observations. A state diagram appears in *(a)*. The half-life of the first state is about ten and the half life of the second state is about five, i.e., $0.93^{10} \approx 0.87^5 \approx 0.5$. A simulated state sequence and observation sequence appear in *(b)* and *(c)* respectively. Using the model parameters from *(a)* and the observation sequence from *(c)*, the Viterbi algorithm estimates the state sequence that appears in *(d)* which is satisfyingly similar to the state sequence in *(b)*. Finally, starting from the initial model depicted in *(e)* and using the observation sequence depicted in *(c)*, 50 iterations of the Baum-Welch algorithm produces the model depicted in *(f)* which is satisfyingly similar to *(a)*.

Reestimation Reviewing the derivations in Section 2.3.2, we find that the first two formulas in Table 2.1 (those for the initial state probability, $P_{S[0]|\theta[n+1]}\left(i \mid \theta[n+1]\right)$, and the state transition probability, $P_{S[1]|S[0],\theta[n+1]}\left(\tilde{s} \mid s\right)$) still work with the new observation model. To derive reestimation formulas for the Gaussian observation model parameters, note that Eqn. (2.35) becomes

$$L_{\text{observation}}(y,s) \equiv \log P_{Y[0]|S[0],\theta'}\left(y \mid s,\theta'\right) \tag{3.2}$$

$$= -\frac{1}{2}\log(2\pi) - \log(\sigma_s) - \frac{(y-\mu_s)^2}{2\sigma_s^2}, \tag{3.3}$$

and starting from Eqn. (2.46) calculate

$$Q_{\text{observation}}(\theta',\theta) = \sum_{q[0:T]\in\mathcal{S}^T} P_\theta\left(q[0:T] \mid y[0:T]\right) \sum_{t=0}^{T-1} L_{\text{observation}}(y[t],q[t]) \tag{3.4}$$

$$= \sum_{s\in\mathcal{S}} \sum_{t=0}^{T-1} L_{\text{observation}}(y[t],s) \sum_{q[0:T]:q[t]=s} P_\theta\left(q[0:T] \mid y[0:T]\right) \tag{3.5}$$

$$= -\sum_{s\in\mathcal{S}} \sum_{t=0}^{T-1} w(t,s)\left(\frac{1}{2}\log(2\pi) + \log(\sigma_s) + \frac{\left(y[t]-\mu_s\right)^2}{2\sigma_s^2}\right). \tag{3.6}$$

Since the formulas

$$\mu_s = \sum_{t=0}^{T-1} w(t,s)y[t] \tag{3.7}$$

$$\sigma_s^2 = \sum_{t=0}^{T-1} w(t,s)\left(y[t]-\mu_s\right)^2 \tag{3.8}$$

maximize $Q_{\text{observation}}(\theta',\theta)$, we use them in place of the discrete observation reestimation formula of Chapter 2 (Table 2.1 and Eqn. (2.49)).

3.1.2 Singularities of the likelihood function and regularization

Running 2 iterations of the Baum-Welch algorithm on the observation sequence in Fig. 3.2 (c) starting with the model in Fig. 3.3 (a) produces the model in Fig. 3.3 (b) in which the variance of the observations produced by the second state looks suspiciously small. In fact with additional iterations of the Baum-Welch algorithm that variance continues to shrink, and the code soon stops with a floating point exception. The algorithm is pursuing a singularity in the

likelihood function in which the second state fits observation $y[52]$ exactly and the first state fits all of the other observations. If $\mu_2 = y[52]$ the likelihood $P_{Y[t]|S[t]}\left(y[52] \mid 2\right)$ increases without limit as $\sigma_2^2 \to 0$, i.e.,

$$\lim_{\sigma_2^2 \to 0} P_{Y[t]|S[t]}\left(y[52] \mid 2\right) = \infty.$$

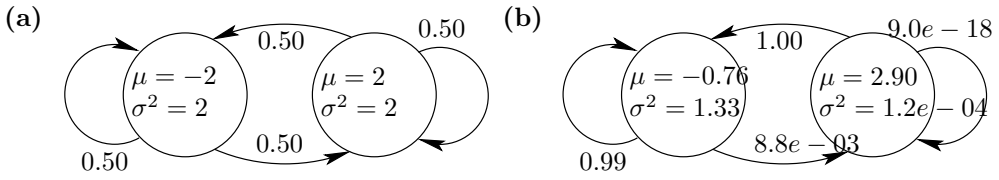

Figure 3.3: An illustration of trouble with maximum likelihood. Here we have used the same implementation of the Baum-Welch algorithm that we used to produce Fig. 3.2*(f)*, but rather than starting with the model in Fig. 3.2 *(c)*, we started the algorithm with the initial model depicted in (a) above. After 2 iterations of the algorithm we get the suspicious model depicted in *(b)* above.

Such singularities of likelihood are common among parametric probability density functions. A particularly simple example is the *Gaussian mixture model*

$$f(y) = \lambda \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(y-\mu_1)^2}{2\sigma_1^2}} + (1-\lambda) \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(y-\mu_2)^2}{2\sigma_2^2}}, \tag{3.9}$$

which has the five parameters $\mu_1,\ \sigma_1,\ \mu_2,\ \sigma_2$, and λ. Assuming that the data are i. i. d., one might attempt a maximum likelihood fit to the observations in Fig. 3.2 (e) with the likelihood function

$$g(\mu_1, \sigma_1, \mu_2, \sigma_2, \lambda) = \prod_{t=0}^{T-1} f(y[t]). \tag{3.10}$$

While it is possible to find a useful *local* maximum of g near

$$\mu_1 = -1, \quad \sigma_1 = 1, \quad \mu_2 = 1, \quad \sigma_2 = 1, \quad \lambda = \frac{2}{3},$$

the likelihood is higher near the singularities of g specified by the equations

$$\mu_s = y[t]$$
$$\sigma_s = 0,$$

for each pair $(s,t) \in \{1,2\} \times \{1,2,\ldots,T\}$.

If, as is the case here, we want to exclude parameter vectors for which the likelihood function is larger than its value at the solution we prefer, then likelihood doesn't really express our goal. *Regularization* refers to a variation on maximum likelihood that more accurately reflects what we want. In the next subsection, we explain how to use Bayesian *priors* to regularize *maximum a posteriori* parameter estimates.

3.1.3 The EM algorithm for maximum a posteriori estimation

Bayesian-estimation starts by characterizing the acceptability of models $P_{\mathbf{Y}|\theta}$ in terms of a *prior* probability distribution P_θ. Initial observations $\mathbf{y}_e$, called *evidence* or training data, modify the prior through Bayes rule to yield the *a posteriori* distribution

$$P(\theta \mid \mathbf{y}_e) = \frac{P(\mathbf{y}_e, \theta)}{P(\mathbf{y}_e)} = \frac{P(\mathbf{y}_e \mid \theta) P(\theta)}{\int P(\mathbf{y}_e \mid \theta) P(\theta)\, d\theta}, \tag{3.11}$$

and the probability of future observations $\mathbf{y}_f$ is

$$P(\mathbf{y}_f \mid \mathbf{y}_e) = \int P(\mathbf{y}_f \mid \theta) P(\theta \mid \mathbf{y}_e)\, d\theta. \tag{3.12}$$

The parameter vector that maximizes the *a posteriori* probability,

$$\theta_{\text{MAP}} \equiv \underset{\theta}{\text{argmax}}\, P(\theta \mid \mathbf{y}_e) \tag{3.13a}$$

$$= \underset{\theta}{\text{argmax}}\, P(\theta, \mathbf{y}_e), \tag{3.13b}$$

is called the MAP estimate. Using θ_{MAP} one may approximate[2] Eqn. (3.12) with $P\left(\mathbf{y}_f \mid \theta_{\text{MAP}}\right)$.

A slight variation of the algebra in Section 2.5 produces an EM algorithm for MAP estimation. Dropping the subscript on $\mathbf{y}_e$ if we replace the auxiliary function of Eqn. (2.11), i.e.,

$$Q_{\text{MLE}}(\theta', \theta) \equiv \mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)} \left(\log P(\mathbf{S}, \mathbf{y} \mid \theta')\right),$$

with

$$Q_{\text{MAP}}(\theta', \theta) \equiv \mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)} \left(\log P(\mathbf{S}, \mathbf{y}, \theta')\right) \tag{3.14a}$$

$$= Q_{\text{MLE}}(\theta', \theta) + \mathbb{E}_{P(\mathbf{S}|\mathbf{y},\theta)} \left(\log P(\theta')\right) \tag{3.14b}$$

$$= Q_{\text{MLE}}(\theta', \theta) + \log P(\theta'), \tag{3.14c}$$

then the derivation of

$$Q_{\text{MAP}}(\theta', \theta) > Q_{\text{MAP}}(\theta, \theta) \Rightarrow P(\mathbf{y}, \theta') > P(\mathbf{y}, \theta),$$

is completely parallel to the argument on page 42 that concludes $Q_{\text{MLE}}(\theta', \theta) > Q_{\text{MLE}}(\theta, \theta)$. If in addition the components of $P(\theta)$ are independent, i.e.,

$$P(\theta) = P(\theta_{\text{initial}}) \cdot P(\theta_{\text{transition}}) \cdot P(\theta_{\text{observation}}),$$

[2] Although it is every bit as reasonable to use the mean to characterize the *a posteriori* distribution as it is to use the maximum, we prefer the maximum because changing from MLE to MAP requires only minor modifications to the Baum-Welch algorithm. A strictly Bayesian approach would retain the entire *a posteriori* distribution in parameter space rather than characterizing the distribution by a single point estimate.

then like the decomposition of Q_{MLE} in Eqn. (2.29) we find

$$Q_{\text{MAP}}(\theta', \theta) = Q_{\text{MAP, initial}}(\theta', \theta) + Q_{\text{MAP, transition}}(\theta', \theta) + Q_{\text{MAP, observation}}(\theta', \theta),$$

with

$$Q_{\text{MAP, initial}} = Q_{\text{MLE, initial}} + \log P(\theta_{\text{initial}}) \tag{3.15a}$$
$$Q_{\text{MAP, transition}} = Q_{\text{MLE, transition}} + \log P(\theta_{\text{transition}}) \tag{3.15b}$$
$$Q_{\text{MAP, observation}} = Q_{\text{MLE, observation}} + \log P(\theta_{\text{observation}}). \tag{3.15c}$$

With a suitable prior, the simple form of Eqn. (3.15) makes it easy to convert a program that implements the Baum-Welch algorithm for maximum likelihood estimation into a program that implements maximum a posteriori estimation.

3.1.4 Vector Autoregressive Observations

Overview of the model

Rather than choosing a prior and developing algorithms for the independent scalar observations of Section 3.1.1 we will work on more general models with vector autoregressive Gaussian observations associated with each hidden state. If at time t such a system is in state s, then the mean of the conditional distribution for $y[t]$ is a linear function of the d previous observations $y[t-d:t]$ added to a fixed offset[3]. Specifically, the conditional distributions for observations are n dimensional vector Gaussians as follows:

Covariance The covariance is a state dependent positive definite $n \times n$ matrix Σ_s.

Mean Given that the system is in state s, the parameters $\{c_{s,i,\tau,j}, \bar{y}_{s,i}\}$ and the d previous observations determine the mean of the observation distribution through a linear function, with components

$$\mu_i\left(s, y[t-d:t]\right) = \bar{y}_{s,i} + \sum_{\tau=1}^{d}\sum_{j=1}^{n} c_{s,i,\tau,j} y_j[t-\tau]. \tag{3.16}$$

We implement this as a *linear* function of a *context vector* x consisting of d previous observations and a constant one, i.e.,

$$\begin{aligned} x[t] &\equiv \left(\overrightarrow{y[t-1]}, \overrightarrow{y[t-2]}, \ldots \overrightarrow{y[t-d]}, 1\right) \\ &\equiv \left(y_1[t-1], y_2[t-1], \ldots, y_n[t-1], y_1[t-2], \ldots, y_n[t-d], 1\right). \end{aligned}$$

[3]The function g in $y = g(x) = mx$ is *linear* because: $g(cx) = c(gx)$ and $g(x+z) = g(x) + g(z)$. On the other hand in the equation for a line, $y = f(x) = mx + b$, f is an *affine* function, in other words a linear function plus a constant offset. Subsequently we will not make the distinction and refer to functions like f as linear.

Using notation in which the i^{th} component of the observation τ time steps before time t is $y_i[t-\tau]$, the k^{th} component of the context at time t is

$$x_k[t] = \begin{cases} y_i[t-\tau] & 1 \le \tau \le d \\ 1 & \tau = d+1, \quad i = 1, \end{cases}, \tag{3.17}$$

where $k = n \cdot (\tau - 1) + i$, and

$$\mu_s\left(s, y[t-d:t]\right) = A_s x[t] \tag{3.18}$$

where A_s is an $n \times (nd+1)$ matrix consisting of $\{c_{s,i,\tau,j}\}$ and $\{\bar{y}_{s,i}\}$.

Using this notation the model assumptions are (compare to Eqns. (1.19) and (1.20) which describe the assumptions for HMMs with discrete observations.) that the states follow a Markov process and that the conditional distribution of an observation given the state s is Gaussian with mean $A_s x[t]$ and covariance Σ_s, i.e.,

$$P(s[t] \mid s[-\infty:t], y[-\infty:t]) = P\left(s[t] \mid s[t-1]\right) \tag{3.19}$$

$$P(y[t] \mid s[-\infty:t+1], y[-\infty:t]) = \frac{1}{\sqrt{(2\pi)^n \left|\Sigma_{s(t)}\right|}} \exp\left(-\frac{1}{2} z_{s(t)}^\top(t) \Sigma_{s(t)}^{-1} z_{s(t)}(t)\right) \tag{3.20}$$

where

$$z_s(t) \equiv y(t) - A_s x(t).$$

The model accounts for relationships between an observation and its predecessors two ways, first the observation at time $y[t]$ is related to the d previous observations through the matrix A_s, and it is also related to *all* previous observations through the state s.

Reestimation

A derivation like the one leading to (3.6) yields

$$Q_{\text{observation}}(\theta', \theta) = \frac{1}{2} \sum_{s \in \mathcal{S}} \sum_{t=0}^{T-1} w(t,s) \left[\log\left(\left|\Sigma_s^{-1}\right|\right) - n\log(2\pi) - z_s^\top(t) \Sigma_s^{-1} z_s(t)\right]. \tag{3.21}$$

Hence each term in the sum over s can be optimized separately. One can write code that maximizes $\sum_{t=0}^{T-1} w(t,s) \left[\log\left|\Sigma_s^{-1}\right| - z_s^\top(t) \Sigma_s^{-1} z_s(t)\right]$ using operations on vectors and matrices as follows:

1. Create a weighted *context* matrix X_s with columns $x[t]\sqrt{w(t,s)}$, where $w(t,s) = P_{S[t]|y[0:T]}\left(s \mid y[0:T]\right)$ and Eqn. (3.17) defines $x[t]$.

2. Create a weighted *observation* matrix Y_s with columns $y[t]\sqrt{w(t,s)}$.

3. Solve
$$A_s = \underset{M}{\operatorname{argmin}} |Y_s - MX_s|^2 \tag{3.22}$$
(We use singular value decomposition methods here because they are stable and make diagnosing problems easy.)

4. Calculate a matrix of residuals
$$Z_s = Y_s - A_s X_s$$

5. Calculate a new covariance matrix[4]
$$\Sigma_s = \frac{\sum_{t=0}^{T-1} w(t,s) z_s(t) z_s^\top(t)}{\sum_{t=0}^{T-1} w(t,s)} \tag{3.23}$$

Regularization

As Eqn. (3.15) indicates, we can influence the a posteriori distributions of the initial states, the transitions, and the observations by our selection of the respective priors, $P(\theta_{\text{initial}})$, $P(\theta_{\text{transition}})$, and $P(\theta_{\text{observation}})$. Since singularities in the likelihood are associated with singular covariance matrices Σ_s, the prior for the covariance matrices is most urgent.

Following [30, 38] we use inverse-Wishart ~distributions as priors for the inverse covariance matrices. See pages 150-155 of Schafer [13] for a description of these distributions. The inverse-Wishart prior has the following probability density for an $n \times n$ inverse covariance matrix

$$P_{\text{IW}}(\Sigma^{-1}) \equiv C \left|\Sigma^{-1}\right|^{\frac{m+n+1}{2}} e^{-\frac{1}{2}\mathbf{tr}(\Lambda\Sigma^{-1})},$$

where C is a normalization constant, m is called the *degrees of freedom*, and Λ is called the *scale*. The mean and the maximum of the inverse-Wishart are

$$\mathbb{E}\Sigma^{-1} = \frac{1}{m-n-1}\Lambda^{-1}, \text{ and}$$
$$\operatorname{argmax} P_{\text{IW}}\left(\Sigma^{-1}\right) = \frac{1}{m+n+1}\Lambda^{-1}$$

respectively.

[4] One may derive this formula by differentiating the right-hand side of Eqn. 3.21 with respect to the elements of Σ_s^{-1} and setting the result equal to zero. The key is the observation that for a positive definite matrix M,
$$\frac{\partial \log|M|}{\partial m_{i,j}} = \left[M^{-1}\right]_{i,j},$$
i.e., the derivative of the log of the determinant with respect to the i,j^{th} element of M is the i,j^{th} element of M^{-1}.

Since the value of A_s that minimizes $\sum_{t=0}^{T-1} w(t,s) - z_s^\top(t)\Sigma_s^{-1} z_s(t)$ is independent of Σ_s^{-1}, it is correct to do step 3 before step 5.

Assuming neutral priors for the coefficient matrices A_s and the form

$$\Lambda = \beta \mathbf{I},$$

we find

$$P(\theta_y) \propto \prod_s \left|\Sigma_s^{-1}\right|^{\frac{\alpha}{2}} e^{-\frac{1}{2}\mathbf{tr}(\beta\Sigma_s^{-1})},$$

(where $\alpha = n + m + 1$) and Eqn. 3.15c becomes

$$Q_{\text{observation}} = C + \sum_s \left(\frac{\alpha}{2} \log \left|\Sigma_s^{-1}\right| - \frac{1}{2}\mathbf{tr}(\beta\Sigma_s^{-1}) \right) + \frac{1}{2} \sum_s \sum_{t=0}^{T-1} w(t,s) \left[\log \left|\Sigma_s^{-1}\right| - z_s^\top(t)\Sigma_s^{-1} z_s(t) \right],$$

where $z_s(t) \equiv y(t) - A_s x(t)$ and $w(t,s)$ is the probability of being in state s at time t given the data $y[0:T]$. Reestimation of the regularized model is the same as reestimation for the unregularized model except that Eqn. 3.23 is replaced with

$$\Sigma_s = \frac{\beta\mathbf{I} + \sum_{t=0}^{T-1} w(t,s) z_s(t) z_s^\top(t)}{\alpha + \sum_{t=0}^{T-1} w(t,s)}. \tag{3.24}$$

Thus in the absence of new information, a covariance matrix is given the default value $\frac{\beta\mathbf{I}}{\alpha}$, and α is the weight of that default.

3.2 Related Models

Results from applying our code for HMMs with the vector autoregressive observations described in Section 3.1.4 to a time series of 3-d vectors from the Lorenz system appear in Figure 3.4. Aside from the different observation models, we used the same procedure and much of the same code to make Figure 3.4 and Figure 1.10 on page 16. Here the model for each state uses a single previous observed vector to forecast the next observation. Optimization should find regions over which dynamics are approximately linear and assign those regions to single states. We have no intuitive explanation for the many stripe like regions in the center of the figure. The model class used to make the figure does not exploit all of the available features of the Lorenz data. In particular it does not use the Lorenz equations. Modeling techniques exist that can exploit knowledge of nonlinear generating equations, e.g. *extended Kalman filters* and *particle filters* that we mention below.

Here we list a few of the many techniques that are related to the basic ideas we've described:

Nonstationary HMMs The definition of a *stationary* model is that all probabilities are independent of shifts in time, i.e., $P_{Y[0:t]} = P_{Y[0+\tau:t+\tau]} \forall (t,\tau)$. Many processes, for example weather, are not even approximately stationary. By making HMMs with state transition probabilities that depend on time, many have created nonstationary HMMs for such applications.

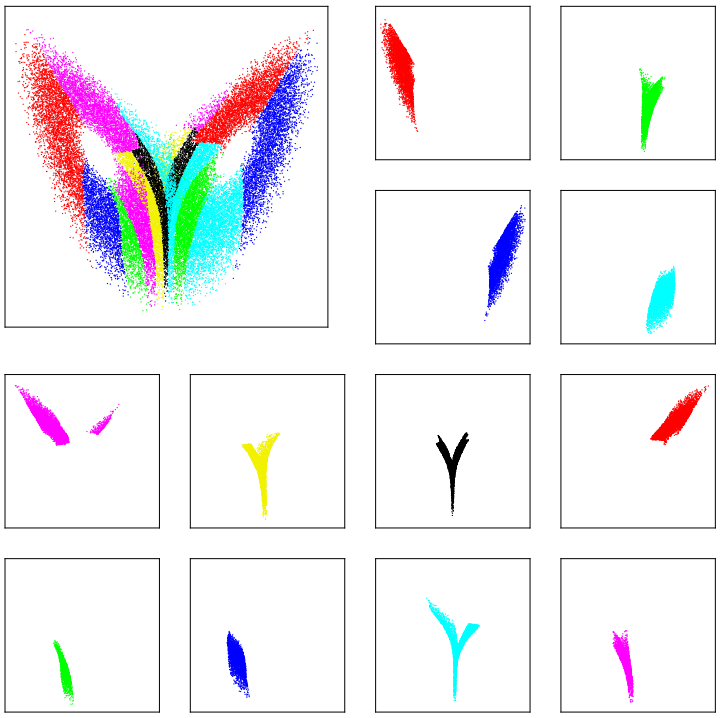

Figure 3.4: Plots of decoded states using an HMM with vector autoregressive observations. Here the observations are a trajectory of three dimensional state vectors from the Lorenz system. In each state the observation $y[t]$ is modeled as a Gaussian with a mean that is a linear function of one previous observation $y[t-1]$.

Gaussian mixtures In a Gaussian mixture, the probability density is weighted average of Gaussian densities. We described a simple one dimensional two component example in Eqn. (3.9). Although a simple Gaussian mixture model does not use the notion of *state*, they are frequently used as observation models in HMMs. Note that a Gaussian mixture model is equivalent to the degenerate case of an HMM with Gaussian observations in which all state transition probabilities are identical.

Cluster weighted modeling For an HMM, the state transition probabilities depend only on the state. Looking at Fig. 3.4, it seems that letting the current observation $y[t]$ influence the transition probabilities could be an improvement, i.e., the probability of the next state would depend on both the current state and the current observation. By ignoring the current state entirely, one obtains a *cluster weighted model*[31]; the observation at time $t+1$ is a mixture whose component weights are determined by the observation at time t.

Kalman filter The Kalman filter is a version of the forward algorithm one obtains for a model with continuous states $x[t]$ and observations $y[t]$ with the form

$$x[t+1] = F(x[t], t) + \eta[t]$$
$$y[t] = G(x[t]) + \epsilon[t]$$

where the functions F and G are linear in x and the noise terms $\eta[t]$ and $\epsilon[t]$ are Gaussian. We will describe algorithms that operate with such models in the next chapter.

Extended Kalman filter Using linear approximations to nonlinear functions F and G in a Kalman filter is called an *extended Kalman filter*. Linear approximations work well as long as the spreads of the state distributions are small compared to the second derivatives of F and G.

Unscented Kalman filter Rather than use linear approximations to F and G, an unscented Kalman filter[34] uses exact functions and a collection of samples to estimate means and variances.

Particle filter The idea of using the empirical distribution of many simulated trajectories (particles) to approximate the conditional distribution $P_{X[t]|Y[0:t+1]}$ is called particle filtering[32, 35]. In the procedure, particles that seem inconsistent with measurements are eliminated and new particles are created.

Chapter 4

Continuous States and Observations and Kalman Filtering

We think of state space systems with continuously distributed states and observations in terms of equations like

$$x[t] = F(x[t-1], t) + \eta[t] \tag{4.1a}$$
$$y[t] = G(x[t], t) + \epsilon[t], \tag{4.1b}$$

where $X \in \mathbb{R}^n$ is the state variable, $Y \in \mathbb{R}^m$ is the observation, and $\eta[t]$ and $\epsilon[t]$ are noise terms. Equations (4.1) define the conditional probability densities

$$P_{X[t+1]|X[t]} \tag{4.2}$$

and

$$P_{Y[t]|X[t]}. \tag{4.3}$$

Having each of the noise terms $\eta[t]$ and $\epsilon[t]$ in Eqns. 4.1 be independent of all other noise terms is sufficient to ensure that the following assumptions hold:

1. The dynamics of the state variable X are Markov.
2. Given the state at time t, the observation $Y[t]$ is independent of everything else.

These are the same as the assumptions of Eqns. (1.19) and (1.20) which characterize HMMs with discrete observations. In this chapter, we write forward and backward algorithms by replacing sums of finite probabilities with integrals over probability densities. If the functions F and G are linear in X and the noise terms $\eta[t]$ and $\epsilon[t]$ are independent and Gaussian, then the forward algorithm is called *Kalman Filtering*.

In this chapter we go over the forward and backward algorithms for continuous states and observations three times. First in Section 4.1 we emphasize the parallel to Chapter 2 by simply replacing sums in the development of that chapter with integrals. By themselves, the results are not immediately useful because one must specify parametric forms for the probability densities and *do* the integrals to implement the algorithms. Next, in Section 4.2, we concisely present the algorithms one obtains when the functions F and G in Eqns. (4.1) are linear and the noise terms $\eta[t]$ and $\epsilon[t]$ are Gaussian. We hope this concise presentation will be useful for readers who want to implement the algorithms. Finally, for completeness, in Section 4.3 we demonstrate that in fact the integrals of Section 4.1 do yield the formulas of Section 4.2.

4.1 Algorithms with Integrals

Here we write out the forward and backward algorithms for models with continuously distributed states and observations by simply replacing sums in Chapter 2 with integrals.

4.1.1 Forward Algorithm

Like the forward algorithm for a simple HMM (see Equations (2.2) on page 22) the forward algorithm for continuous variables uses each successive observation and calculates the incremental likelihood

$$\gamma[t] \equiv P(y[t] \mid y[0:t])$$

and the three distributions

$$\alpha(t,x) \equiv P_{X[t]|Y[0:t+1]}\left(x \mid y[0:t+1]\right) \qquad \text{Updated state distribution}$$

$$a(t,x) \equiv P_{X[t]|Y[0:t]}\left(x \mid y[0:t]\right) \qquad \text{Forecast state distribution}$$

$$\tilde{a}(t,x) \equiv P_{X[t],Y[t]|Y[0:t]}\left(x, y[t] \mid y[0:t]\right) \qquad \text{Joint forecast distribution.}$$

Using the distributions $P_{X[0]}$ and $P_{Y[0]|X[0]}$ which are parts of the model, we start by initializing in step 1 and then we loop over the remaining steps for the subsequent observations:

1. **Initialize** The calculations

$$P_{X[0],Y[0]}\left(x, y[0]\right) = P_{Y[0]|X[0]}\left(y[0] \mid x\right) P_{X[0]}\left(x\right) \tag{4.4}$$

$$\gamma[0] = P\left(y[0]\right) = \int P_{X[0],Y[0]}\left(x, y[0]\right) dx \tag{4.5}$$

$$\alpha(0,x) \equiv P_{X[0]|Y[0:1]}\left(x \mid y[0:1]\right) = \frac{P_{X[0],Y[0]}\left(x, y[0]\right)}{P\left(y[0]\right)} \tag{4.6}$$

specify $\gamma[0]$, the likelihood for the first observation and $\alpha(0,x)$, the conditional distribution of the first state given the first observation.

2. **Forecast the state distribution** Find the conditional distribution of the state at the present time given the past observations.

$$\begin{aligned} P\left(x[t], x[t-1] \mid y[0:t]\right) &= P\left(x[t] \mid x[t-1], y[0:t]\right) && (4.7)\\ &\quad \times P\left(x[t-1] \mid y[0:t]\right) \\ &= P\left(x[t] \mid x[t-1]\right) P\left(x[t-1] \mid y[0:t]\right) && (4.8)\\ &= P\left(x[t] \mid x[t-1]\right) \alpha(t-1, x[t-1]) \\ a(t,x) &\equiv P_{X[t]|Y[0:t]}\left(x \mid y[0:t]\right) \\ &= \int P\left(x[t], x[t-1] \mid y[0:t]\right) dx[t-1] && (4.9)\\ &= \int P_{X[1]|X[0]}\left(x|x'\right) \alpha(t-1, x') dx'. && (4.10) \end{aligned}$$

Here we justify (4.7) by Bayes rule, (4.8) by the model assumptions, and (4.9) by the definition of a marginal distribution.

3. **Calculate the joint forecast** Leaving the state at the present time as a free parameter, calculate the joint conditional distribution of the state and the observation $y[t]$ given past observations.

$$\begin{aligned} \tilde{a}(t,x) &\equiv P_{X[t],Y[t]|Y[0:t]}\left(x, y[t] \mid y[0:t]\right) \\ &= P\left(y[t] \mid x[t], y[0:t]\right) P_{X[t]|Y[0:t]}\left(x \mid y[0:t]\right) && (4.11)\\ &= P\left(y[t] \mid x[t]\right) a(t,x) && (4.12) \end{aligned}$$

Here we justify (4.11) by Bayes rule and (4.12) by the model assumptions.

4. **Calculate the incremental likelihood** Integrate out $x[t]$ to get the conditional probability of this observation given previous observations

$$\begin{aligned} \gamma[t] \equiv P(y[t] \mid y[0:t]) &= \int P_{X[t],Y[t]|Y[0:t]}\left(x, y[t] \mid y[0:t]\right) dx[t] \\ &= \int \tilde{a}(t,x) dx && (4.13) \end{aligned}$$

5. **Update the conditional state distribution** Use Bayes rule to combine (4.12) and (4.13) to get $\alpha(t,x)$, the conditional distribution of the present state given all past observations and the present observation.

$$\begin{aligned} \alpha(t,x) &\equiv P_{X[t]|Y[0:t+1]}\left(x \mid y[0:t+1]\right) && (4.14)\\ &= \frac{P_{X[t],Y[t]|Y[0:t]}\left(x, y[t] \mid y[0:t]\right)}{P(y[t] \mid y[0:t])} && (4.15)\\ &= \frac{\tilde{a}(t,x)}{\gamma[t]}. && (4.16) \end{aligned}$$

Note that aside from normalization, the forward algorithm alternates between multiplying by the observation likelihood in (4.12) and integrating with the state transition probability in (4.10).

4.1.2 Backward Algorithm

Similarly, the backward algorithm finds a function called the backwards-forecast,

$$\beta(t,x) \equiv \frac{P_{Y[t+1:T]|X[t]}\left(y[t+1:T] \mid x\right)}{P\left(y[t+1:T] \mid y[0:t+1]\right)}, \tag{4.17}$$

for each time t by starting with $\beta(T-1,x) = 1$ and following with the recursion

$$\beta(t-1,x) = \int \frac{\beta(t,x')P_{Y[t]|X[t]}\left(y[t] \mid x'\right)P_{X[t]|X[t-1]}\left(x' \mid x\right)}{P\left(y[t] \mid y[0:t]\right)}dx'. \tag{4.18}$$

Note that by replacing the sum over discrete states s in (2.20) on page 31 with an integral over continuous states x' one obtains (4.18), and that as with the discrete case we have chosen to define $\beta(t,x)$ so that the conditional distribution of the state at time t given all of the observations is

$$\alpha(t,x)\beta(t,x) = P_{X[t]|Y[0:T]}\left(x \mid y[0:T]\right). \tag{4.19}$$

Defining and evaluating a *backwards update* function

$$b(t,x) \equiv \frac{\beta(t,x)P_{Y[t]|X[t]}\left(y[t] \mid x\right)}{P\left(y[t] \mid y[0:t]\right)} \tag{4.20}$$

lets one write Eqn. (4.18) as the *backwards forecast*

$$\beta(t-1,x) = \int b(t,x')P_{X[t]|X[t-1]}\left(x' \mid x\right)dx' \tag{4.21}$$

which may make evaluating the integral easier and emphasizes the alternation between multiplying by the observation likelihood in (4.20) and integrating with the transition probability (4.21).

4.2 Linear Gaussian Systems

If the functions F and G in Eqn. (4.1) are linear in x, the noise terms $\eta[t]$ and $\epsilon[t]$ are i. i. d. and Gaussian[1] with $\eta[t] \sim \mathcal{N}(0,\Sigma_\eta)$ and $\epsilon[t] \sim \mathcal{N}(0,\Sigma_\epsilon)$, and the

[1]We assume *identical* distributions only to simplify the notation. It lets us write η and ϵ instead of $\eta[t]$ and $\epsilon[t]$. The procedures generalize easily to time dependent noise terms. The notation $X \sim \mathcal{N}(\mu,\Sigma)$ means, "The random variable X has a Gaussian distribution with mean μ and covariance Σ.

distribution of the initial state $X[0]$ is also Gaussian, then we can write

$$X[0] \sim \mathcal{N}\left(\mu_{X[0]}, \Sigma_{X[0]}\right) \tag{4.22a}$$
$$x[t] = F[t] \cdot x[t-1] + \eta \qquad \eta \sim \mathcal{N}\left(0, \Sigma_\eta\right) \tag{4.22b}$$
$$P(x[t] \mid x[t-1]) = \mathcal{N}\left(Fx[t-1], \Sigma_\eta\right)\big|_{x[t]} \tag{4.22c}$$
$$y[t] = G[t] \cdot x[t] + \epsilon \qquad \epsilon \sim \mathcal{N}\left(0, \Sigma_\epsilon\right) \tag{4.22d}$$
$$P(y[t] \mid x[t]) = \mathcal{N}\left(Gx[t], \Sigma_\epsilon\right)\big|_{y[t]}, \tag{4.22e}$$

where $F[t]$ and $G[t]$ are matrices. We use the notation $\mathcal{N}(\mu, \Sigma)$ for a Gaussian distribution with mean μ and covariance Σ, and we denote the value of a Gaussian probability density at v by $\mathcal{N}(\mu, \Sigma)|_v$ i.e. if V is an n-dimensional random variable $V \sim \mathcal{N}(\mu, \Sigma)$ then $P(v) = \mathcal{N}(\mu, \Sigma)|_v \equiv \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(v-\mu)^\top \Sigma^{-1}(v-\mu)}$.

We describe the forward and backward algorithms for linear Gaussian systems in terms of the quantities listed below. Our notation for these quantities emphasizes the similarity to the calculations for discrete processes in Chapter 2. In particular the Greek subscripts α and β denote the forward updated distribution and the backwards forecast distribution respectively while the Roman subscripts a and b denote the forward forecast distribution and the backwards updated distributions respectively. The assumptions of Eqns. (4.22) imply that the distributions described by these parameters are also Gaussian.

$\boldsymbol{\mu_\alpha[t]}$ and $\boldsymbol{\Sigma_\alpha[t]}$ (Note *Greek* subscript.) are the parameters of the *updated* state distribution[2], i.e.,

$$P(x[t] \mid y[0:t+1]) \equiv \mathcal{N}(\mu_\alpha[t], \Sigma_\alpha[t])\big|_{x[t]}.$$

$\boldsymbol{\mu_a[t]}$ and $\boldsymbol{\Sigma_a[t]}$ (Note *Roman* subscript.) are the parameters of the one step *forecast* of the state distribution[3], i.e.,

$$P(x[t] \mid y[0:t]) \equiv \mathcal{N}(\mu_a[t], \Sigma_a[t])\big|_{x[t]}.$$

$\boldsymbol{\mu_\gamma[t]}$ and $\boldsymbol{\Sigma_\gamma[t]}$ are the parameters of the conditional probability of the observation at time t given all previous observations, i.e.

$$P(y[t] \mid y[0:t]) \equiv \mathcal{N}(\mu_\gamma[t], \Sigma_\gamma[t])\big|_{y[t]} \equiv \gamma[t].$$

Neither the forward nor the backwards Kalman filter uses the γ terms, but they are useful for calculating the likelihood for model parameters.

$\boldsymbol{N_\beta[t]}$, $\boldsymbol{\mu_\beta[t]}$ and $\boldsymbol{\Sigma_\beta[t]}$ are the parameters[4] of the backwards forecast function $\beta(t, x)$, which as a ratio of Gaussian functions is itself an unnormalized

[2]For the quantities that we call $\mu_\alpha[t]$ and $\Sigma_\alpha[t]$, Maybeck[9] uses the notation $\hat{x}(t_i^+)$ and $P(t_i^+)$ and Kailath et al.[5] use $\hat{x}_{i|i}$ and $P_{i|i}$ respectively.

[3]For the quantities that we call $\mu_a[t]$ and $\Sigma_a[t]$, Maybeck[9] uses the notation $\hat{x}(t_i^-)$ and $P(t_i^-)$ and Kailath et al.[5] use $\hat{x}_{i+1}$ and P_{i+1} respectively.

[4]On page 342, Kailath et al.[5] use $\hat{x}_i^b$ to denote the quantity that we call $\mu_\beta[t]$.

Gaussian. We define $N_\beta[t]$ by

$$\frac{\beta(t,x)}{N_\beta[t]} \equiv \mathcal{N}\left(\mu_\beta[t], \Sigma_\beta[t]\right)\big|_x .$$

$\boldsymbol{\mu_b[t]}$ and $\boldsymbol{\Sigma_b[t]}$ are the mean[5] and covariance of the backwards update function $b(t,x)$ (see Eqn. (4.26)) which is an intermediate term in the backward algorithm. Notice that the parameters of the forward forecast have Roman subscripts while the parameters of the backward forecast have Greek subscripts.

$\boldsymbol{\mu_{\alpha\beta}[t]}$ and $\boldsymbol{\Sigma_{\alpha\beta}[t]}$ are the mean and covariance of the best estimate of the state at time t given all of the observations, i.e.

$$P\left(x[t] \mid y[0:T]\right) \equiv \mathcal{N}\left(\mu_{\alpha\beta}[t], \Sigma_{\alpha\beta}[t]\right)\big|_x .$$

4.2.1 Kalman Filter: The Forward Algorithm

The following two step recursion is called Kalman filtering[6] and it implements the forward algorithm:

Calculate the forecast of the distribution of the state

$$\mu_a[t] = F[t] \cdot \mu_\alpha[t-1] \tag{4.23a}$$

$$\Sigma_a[t] = F[t] \cdot \Sigma_\alpha[t-1] \cdot \left(F[t]\right)^\top + \Sigma_\eta \tag{4.23b}$$

Update the distribution of the current state using $\boldsymbol{y[t]}$

$$\left(\Sigma_\alpha[t]\right)^{-1} = \left(\Sigma_a[t]\right)^{-1} + \left(G[t]\right)^\top \Sigma_\epsilon^{-1} G[t] \tag{4.24a}$$

$$\mu_\alpha[t] = \mu_a[t] + \Sigma_\alpha[t]\left(G[t]\right)^\top \Sigma_\epsilon^{-1}\left[y[t] - G[t]\mu_a[t]\right] \tag{4.24b}$$

Note:

- Equations (4.24) are usually presented in the equivalent but computationally more efficient form

$$\mu_\alpha[t] = \mu_a[t] + K[t]\left(y[t] - G[t]\mu_a[t]\right) \tag{4.25a}$$

$$\Sigma_\alpha[t] = \left(\mathbf{I} - K[t]G[t]\right)\Sigma_a[t] \tag{4.25b}$$

where

$$K[t] \equiv \Sigma_a[t]\left(G[t]\right)^\top \Big(G[t]\Sigma_a[t]\left(G[t]\right)^\top + \Sigma_\epsilon\Big)^{-1} \tag{4.25c}$$

is called *the Kalman gain matrix*[7] (See Eqn. (4.25).).

[5] Kailath et al.[5] use $\hat{x}^b_{i|i}$ to denote the quantity that we call $\mu_b[t]$.

[6] There are many more detailed presentations of Kalman filters, e.g., Maybeck[9], Kailath et al.[5], Jacobs[53], and Brown and Hwang[50].

[7] This computationally more efficient form was Kalman's principle contribution.

- One can calculate the log likelihood of a model by summing the increments

$$\log\left(P\left(y[0:T]\right)\right) = \sum_{t=0}^{T-1} \log\left(P\left(y[t] \mid y[0:t]\right)\right)$$

and calculating each increment as

$$P\left(y[t] \mid y[0:t]\right) = \mathcal{N}\left(\mu_\gamma[t], \Sigma_\gamma[t]\right)\big|_{y[t]}$$

where

$$\begin{aligned}
\mu_\gamma[t] &= G[t]\mu_a[t] \\
\Sigma_\gamma[t] &= G[t]\Sigma_a[t]\left(G[t]\right)^\top + \Sigma_\epsilon \qquad \text{and} \\
\log\left(P\left(y[t] \mid y[0:t]\right)\right) &= -\frac{1}{2}\Big(n\log(2\pi) + \log\left(|\Sigma_\gamma[t]|\right) \\
&\qquad + \left(y[t] - \mu_\gamma[t]\right)^\top \left(\Sigma_\gamma[t]\right)^{-1}\left(y[t] - \mu_\gamma[t]\right)\Big).
\end{aligned}$$

4.2.2 The Backward Algorithm

We calculate $\mu_\beta[t]$ and $\Sigma_\beta[t]$ as defined in Eqn. (4.17) with the following recursion that goes through the observations in reverse order.

Update the distribution of the current state using $y[t]$

$$\left(\Sigma_b[t]\right)^{-1} = \left(\Sigma_\beta[t]\right)^{-1} + \left(G[t]\right)^\top \left(\Sigma_\epsilon\right)^{-1} G[t] \tag{4.26a}$$

$$\mu_b[t] = \mu_\beta[t] + \Sigma_b[t]\left(G[t]\right)^\top \Sigma_\epsilon^{-1}\big[y[t] - G[t]\mu_\beta[t]\big], \tag{4.26b}$$

Forecast of the distribution of the state backward in time

$$\mu_\beta[t-1] = \left(F[t]\right)^{-1}\mu_b[t] \tag{4.27a}$$

$$\Sigma_\beta[t-1] = \left(F[t]\right)^{-1}\left(\Sigma_\eta + \Sigma_b[t]\right)\left(\left(F[t]\right)^{-1}\right)^\top \tag{4.27b}$$

Note:

- The the backward update formulas, (4.26) are exactly the same as the forward update formulas, (4.24), and the forecast formulas, (4.27) and (4.23), principally differ in that the backward formula uses F^{-1} where the forward formula uses F.
- As for the forward recursion, the update formulas are usually presented in the computationally more efficient form

$$\Sigma_b[t] = \left(\mathbf{I} - K_b[t]G[t]\right)\Sigma_\beta[t] \tag{4.28a}$$

$$\mu_b[t] = \mu_\beta[t] + K_b[t]\left(y[t] - G[t]\mu_\beta[t]\right) \tag{4.28b}$$

where

$$K_b[t] \equiv \Sigma_\beta[t]G[t]^\top \left(G[t]\Sigma_\beta[t]G[t]^\top + \Sigma_\epsilon[t]\right)^{-1} \tag{4.28c}$$

is called the backwards-Kalman-gain-matrix.

- Ideally one would initialize the backward algorithm with

$$(\Sigma_\beta[T-1])^{-1} = 0$$
$$\mu_\beta[T-1] = 0$$

but that would make $(\Sigma_b[T-1])^{-1}$ uninvertible and preclude using Eqn. (4.27b) to evaluate $\Sigma_\beta[T-1]$. One can address the problem by initializing $\Sigma_\beta^{-1}[T-1]$ with small values, or by using the *inverse covariance form* of the algorithm (see Section 4.3.4.).

4.2.3 Smoothing

The conditional distribution of the state at time t given all of the data is Gaussian and therefore specified by its covariance and mean, i.e.,

$$P\left(x[t] \mid y[0:T]\right) = \mathcal{N}\left(\mu_{\alpha\beta}[t], \Sigma_{\alpha\beta}[t]\right)\big|_{x[t]},$$

where

$$(\Sigma_{\alpha\beta}[t])^{-1} = (\Sigma_\alpha[t])^{-1} + (\Sigma_\beta[t])^{-1} \quad \text{and} \tag{4.29a}$$
$$\mu_{\alpha\beta}[t] = \Sigma_{\alpha\beta}[t]\left((\Sigma_\alpha[t])^{-1}\mu_\alpha[t] + (\Sigma_\beta[t])^{-1}\mu_\beta[t]\right) \tag{4.29b}$$

are combinations of the *forward update* parameters and the *backward prediction* parameters. Such use of all of the observations to estimate the state sequence is called *smoothing*. Combining forward update parameters and backward update parameters, i.e., α and b, for smoothing is an error.

4.3 Algorithm Derivations and Details

Here we connect the integrals that describe the forward and backward algorithms (Eqns. (4.4)-(4.26)) to the formulas (Eqns. (4.23)-(4.29)). In this context each distribution is Gaussian. So we can characterize each distribution by its mean and covariance and ignore normalization. Given two d-dimensional Gaussian density functions

$$P_1(x) \equiv \frac{1}{\sqrt{(2\pi)^d\,|\Sigma_1|}}\exp\left(-\frac{1}{2}(x-\mu_1)^\top\Sigma_1^{-1}(x-\mu_1)\right) \equiv \mathcal{N}\left(\mu_1,\Sigma_1\right)\big|_x \quad \text{and}$$
$$P_2(x) \equiv \frac{1}{\sqrt{(2\pi)^d\,|\Sigma_2|}}\exp\left(-\frac{1}{2}(x-\mu_2)^\top\Sigma_2^{-1}(x-\mu_2)\right) \equiv \mathcal{N}\left(\mu_2,\Sigma_2\right)\big|_x,$$

we can characterize their product or ratio by respectively adding or subtracting their exponents (see Equation (A.14) on page 130).

4.3.1 Forward Kalman Filter

Recall that the forward algorithm for a simple HMM (see Eqn. (2.7) on page 25) alternates between multiplying a state distribution by a likelihood and multiplying the result by the state transition probability matrix. In the forward Kalman filter the state transition matrix is a Gaussian and the sum is replaced by an integral.

The basic loop consists of steps 2 through 5 below. For the first iteration, we start with the model parameters that describe $P(x[0])$ and proceed directly to step 3.

1. **Initialize with $\boldsymbol{\mu_a[0]}$ and $\boldsymbol{\Sigma_a[0]}$.** We initialize the recursion using the model parameters $\mu_a[0]$ and $\Sigma_a[0]$ of $P(x[0])$ and entering the loop at step 3 to calculate $\mu_\gamma[0]$ and $\Sigma_\gamma[0]$ by setting $t = 0$.

2. **Calculate the state forecast, $\boldsymbol{P(x[t] \mid y[0:t])}$.** From Eqn. (4.10) on page 63
$$a(t,x) = \int \mathcal{N}\left(Fx', \Sigma_\eta\right)\big|_x \alpha((t-1), x')dx' \tag{4.30}$$
Rather than evaluating the integral to find the distribution, we can specify it completely by calculating its first two moments. Since
$$x[t] = F[t]x[t-1] + \eta.$$
the mean is
$$\begin{aligned}
\mu_a[t] &= \mathbb{E}_{x[t-1],\eta|y[0:t]}\left[F[t]X[t-1] + \eta\right] && \text{(4.31a)}\\
&= \mathbb{E}_{x[t-1],\eta|y[0:t]}F[t]X[t-1] + \mathbb{E}_{x[t-1],\eta|y[0:t]}\eta && \text{(4.31b)}\\
&= \mathbb{E}_{x[t-1]|y[0:t]}F[t]X[t-1] + \mathbb{E}_\eta\eta && \text{(4.31c)}\\
&= F[t]\mu_\alpha[t-1]. && \text{(4.31d)}
\end{aligned}$$
The key step in the above sequence is Eqn. (4.31c) which we justify by observing that the distribution of the noise η is independent of time and independent of $x[\tau]$, and $y[\tau]$ for all earlier times τ. Similarly we calculate the variance by
$$\begin{aligned}
\Sigma_a[t] &= \mathbb{E}_{x[t-1],\eta|y[0:t]}\big[\left(F[t]X[t-1] + \eta - \mu_a[t-1]\right)\\
&\qquad \times \left(F[t]X[t-1] + \eta - \mu_a[t-1]\right)^\top\big] && \text{(4.31e)}\\
&= F[t]\Sigma_\alpha[t-1]F[t]^\top + \Sigma_\eta. && \text{(4.31f)}
\end{aligned}$$
Thus Eqn. (4.23) implements the integral of Eqn. (4.10).

3. **Calculate $\boldsymbol{P(y[t] \mid y[0:t])}$.** Using
$$y[t] = G[t]x[t] + \epsilon$$

we calculate the mean and covariance of the distribution of the forecast observation as follows

$$\begin{aligned}\mu_\gamma[t] &= \mathbb{E}_{x[t],\epsilon|y[0:t]} G[t]X[t] + \epsilon \\ &= G[t]\mu_a[t] \qquad (4.32)\\ \Sigma_\gamma[t] &= \mathbb{E}_{x[t],\epsilon|y[0:t]} \left(G[t]X[t] + \epsilon - \mu_\gamma[t]\right) \\ &\quad \times \left(G[t]X[t] + \epsilon - \mu_\gamma[t]\right)^\top \\ &= G[t]\Sigma_a[t]G[t]^\top + \Sigma_\epsilon \qquad (4.33)\end{aligned}$$

4. **Calculate $\boldsymbol{P\left(x[t], y[t] \mid y[0:t]\right)}$.** The forecast distribution of the joint variable

$$z[t] = \begin{bmatrix} x[t] \\ y[t] \end{bmatrix}$$

is characterized by

$$\mu_z = \begin{bmatrix} \mu_a[t] \\ \mu_\gamma[t] \end{bmatrix}$$

and

$$\Sigma_z = \begin{bmatrix} \Sigma_a[t] & \Sigma_a[t](G[t])^\top \\ G[t]\Sigma_a[t] & \Sigma_\gamma[t] \end{bmatrix} \equiv \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix}. \qquad (4.34)$$

We derive the off diagonal terms of Σ_z in Eqn. (4.34) as follows

$$\begin{aligned}C &= \mathbb{E}_{x[t],\epsilon|y[0:t]} \big(x[t] - \mu_a[t]\big)\big(y[t] - \mu_\gamma[t]\big)^\top \\ &= \mathbb{E}_{x[t]|y[0:t]} \big(x[t] - \mu_a[t]\big)\big(x[t] - \mu_a[t]\big)^\top \big(G[t]\big)^\top \\ &\quad + \mathbb{E}_{x[t],\epsilon|y[0:t]} \big(x[t] - \mu_a[t]\big)\big(\epsilon\big) \\ &= \Sigma_a[t]\big(G[t]\big)^\top. \qquad (4.35)\end{aligned}$$

We use the terms A, B and C in the right block matrix of (4.34) to match (A.5) on page 128 of the appendix where we derive expressions for D, E and F in

$$\begin{bmatrix} A & C \\ C^\top & B \end{bmatrix}^{-1} \equiv \begin{bmatrix} D & F \\ F^\top & E \end{bmatrix}.$$

we will use those expressions when we calculate the conditional distribution of $x[t]$ given $y[t]$ in step 5.

Note that the calculation in this step is equivalent to completing the square in the sum of the exponents we would get by multiplying $P(y[t] \mid x[t])$ and $a(t, x)$ in (4.12), i.e.,

$$\begin{aligned}\mathcal{N}\left(\mu_{\tilde{a}}, \Sigma_{\tilde{a}}\right)|_{x,y[t]} &\equiv P_{X[t],Y[t]|Y[0:t]}\left(x, y[t] \mid y[0:t]\right) \\ &= \mathcal{N}\left(\mu_a, \Sigma_a\right)|_x \mathcal{N}\left(G[t]x, \Sigma_\epsilon\right)|_{y[t]} \qquad (4.36)\end{aligned}$$

5. **Calculate the state update, $P(x[t] \mid y[0:t+1])$.** With the actual value of $y[t]$ and the joint distribution of $x[t]$ and $y[t]$ given $y[0:t]$ from step 4, we use Eqn. (A.7) from page 129 in the appendix to write the parameters of the conditional $P(x[t] \mid y[0:t+1])$ as

$$\Sigma_\alpha[t] = D^{-1} \tag{4.37a}$$
$$\mu_\alpha[t] = \mu_a[t] + CB^{-1}\left(y[t] - \mu_\gamma[t]\right). \tag{4.37b}$$

Note the following equations and justifications:

$$\begin{aligned}
E^{-1} &= B - C^\top A^{-1} C && \text{See (A.5b)} \\
&= \Sigma_\gamma[t] - G[t]\Sigma_a[t]\left(G[t]\right)^\top && \text{See (4.34)} \\
&= \Sigma_\epsilon && \text{See (4.33)} \\
CB^{-1} &= D^{-1}A^{-1}CE && \text{See (A.5c)} \\
&= \Sigma_\alpha[t]\left(G[t]\right)^\top E && \text{See (4.35)} \\
&= \Sigma_\alpha[t]\left(G[t]\right)^\top \left(\Sigma_\epsilon\right)^{-1} \\
D &= A^{-1} + A^{-1}CEC^\top A^{-1} && \text{See (A.5a).}
\end{aligned}$$

Thus

$$\left(\Sigma_\alpha[t]\right)^{-1} = \left(\Sigma_a[t]\right)^{-1} + \left(G[t]\right)^\top \Sigma_\epsilon^{-1} G[t] \tag{4.38a}$$
$$\mu_\alpha[t] = \mu_a[t] + \Sigma_\alpha[t]\left(G[t]\right)^\top \Sigma_\epsilon^{-1}\left[y[t] - G[t]\mu_a[t]\right] \tag{4.38b}$$

which are the update equations in (4.24).

As presented in Eqns. (4.23) and (4.24), the forward algorithm requires two matrix inversions per iteration. First one must invert $\left(\Sigma_\alpha[t-1]\right)^{-1}$ for Eqn. (4.23b), then one must invert $\Sigma_a[t-1]$ for Eqn. (4.24a). Each of these is an $n \times n$ matrix where n is the dimension of the state x. One can avoid these inversions either by keeping track of state space covariances or by keeping track of *inverse* state space covariances. Either choice improves the numerical speed and accuracy of the algorithm. Kalman's form keeps track of state space covariances and uses the *Kalman gain matrix* which in the notation of Eqn. (4.34) is $K = CB^{-1}$. To get Eqn. (4.37) into Kalman's form, note that by Eqn. (A.5a) on page 128, $D^{-1} = A - CB^{-1}C^\top$, and thus

$$\Sigma_\alpha[t] = \big(\mathbf{I} - K[t]G[t]\big)\Sigma_a[t] \tag{4.39a}$$
$$\mu_\alpha[t] = \mu_a[t] + K[t]\left(y[t] - \mu_\gamma[t]\right). \tag{4.39b}$$

Using Eqns. (4.33), (4.34) and (4.35), we find the Kalman gain matrix in terms of known quantities to be

$$\begin{aligned}
CB^{-1} &\equiv K[t] \\
&= \Sigma_a[t]\big(G[t]\big)^\top \big(\Sigma_\gamma[t]\big)^{-1} \\
&= \Sigma_a[t]\big(G[t]\big)^\top \big(G[t]\Sigma_a[t]G[t]^\top + \Sigma_\epsilon\big)^{-1}.
\end{aligned} \tag{4.40}$$

The formula requires inversion of an $m \times m$ matrix where m, the dimension of the observation y, is usually less than the dimension of the state x.

4.3.2 Backward Recursion

On page 64 we described the backwards recursion as alternating between multiplying a prior by a likelihood in (4.20) and integrating the product of a state probability and a state transition probability in (4.21).

We should initialize the backward recursion with a uniform distribution $\beta(T-1, x) = 1$ which requires $\Sigma^{-1}_{\beta(T-1)} = 0$. We will ignore that for now and focus on the two alternating steps:

1. **Backwards Update** ignoring normalization, given the backwards forecast $\beta(t, x)$, the update formula is

$$\begin{aligned} b(t,x) &= \beta(t,x) P(y[t] \mid x) \\ &= \mathcal{N}\left(\mu_{\beta(t,x)}, \Sigma_{\beta(t,x)}\right)\Big|_x \mathcal{N}\left(G[t]x, \epsilon\right)\Big|_{y[t]}. \end{aligned} \tag{4.41}$$

   Notice that the form of (4.41) matches the form of (4.36) on page 70 in the forward recursion. After taking the conditional given the actual value of $y[t]$ we copy the result from (4.38) and obtain

$$\left(\Sigma_b[t]\right)^{-1} = \left(\Sigma_\beta[t]\right)^{-1} + \left(G[t]\right)^\top \Sigma_\epsilon^{-1} G[t] \tag{4.42a}$$

$$\mu_b[t] = \mu_\beta[t] + \Sigma_b[t] \left(G[t]\right)^\top \Sigma_\epsilon^{-1} \left[y[t] - G[t]\mu_\beta[t]\right] \tag{4.42b}$$

2. **Backwards Forecast** From Equation (4.21) on page 64 the integral for the backward forecast is

$$\beta(t-1, x) = \int \mathcal{N}\left(\mu_{b[t]}, \Sigma_{b[t]}\right)\Big|_{x'} \mathcal{N}\left(Fx, \Sigma_\eta\right)\Big|_{x'} dx' \tag{4.43}$$

   and from Equation (4.10) on page 63 the corresponding integral for the forward forecast is

$$a(t+1, x) = \int \mathcal{N}\left(\mu_{\alpha[t]}, \Sigma_{\alpha[t]}\right)\Big|_{x'} \mathcal{N}\left(Fx', \Sigma_\eta\right)\Big|_{x} dx'. \tag{4.44}$$

   On page 69 via Equations (4.31) we showed that the forward forecast is

$$\begin{aligned} \mu_a[t] &= F[t] \cdot \mu_\alpha[t-1] \\ \Sigma_a[t] &= F[t] \cdot \Sigma_\alpha[t-1] \cdot \left(F[t]\right)^\top + \Sigma_\eta. \end{aligned}$$

   Manipulating the quadratic for in the exponent of $\mathcal{N}\left(\mu_{b[t]}, \Sigma_{b[t]}\right)\Big|_{x'}$ we

find

$$\begin{aligned}
Q &\equiv (x' - Fx)^\top \Sigma_\eta^{-1} (x' - Fx) \\
&= \left(F^{-1}x' - x\right)^\top F^\top \Sigma_\eta^{-1} F \left(F^{-1}x' - x\right) \quad \text{so} \\
\mathcal{N}\left(Fx, \Sigma_\eta\right)\big|_{x'} &= \mathcal{N}\left(F^{-1}x', F^\top \Sigma_\eta F\right)\big|_x \qquad \text{using this in (4.44) yields} \\
\mu_\beta[t-1] &= (F[t])^{-1} \mu_b[t] \\
\Sigma_\beta[t-1] &= (F[t])^{-1} \left(\Sigma_\eta + \Sigma_b[t]\right) \left((F[t])^{-1}\right)^\top,
\end{aligned}$$

which exactly matches (4.27) on page 67.

Since the form of the forward update formula, (4.38) on page 71, is the same as the form of the backwards update formula, (4.42) on page 72, the formula for the backwards gain, (4.28c) on page 67, matches the formula for the forward gain, (4.40) on page 71.

4.3.3 Smoothing

We have defined $\alpha(t,x)$ and $\beta(t,x)$ so that

$$P_{X[t]|Y[0:T]}\left(x \mid y[0:T]\right) = \alpha(t,x)\beta(t,x).$$

(See Eqn. (4.19).) In fact $P_{X[t]|Y[0:T]}$ is Gaussian, and we denote its parameters $\mu_{\alpha\beta}[t]$ and $\Sigma_{\alpha\beta}[t]$. Examining the exponential terms in $\alpha(t,x)\beta(t,x)$ we find (suppressing the time indices)

$$\begin{aligned}
&(x-\mu_\alpha)^\top \Sigma_\alpha^{-1} (x-\mu_\alpha) + (x-\mu_\beta)^\top \Sigma_\beta^{-1} (x-\mu_\beta) \\
&= x^\top \left((\Sigma_\alpha[t])^{-1} + (\Sigma_\beta[t])^{-1}\right) x - 2x^\top \left((\Sigma_\alpha[t])^{-1} \mu_\alpha[t] + (\Sigma_\beta[t])^{-1} \mu_\beta[t]\right) \\
&+ \mu_\alpha^\top (\Sigma_\alpha[t])^{-1} \mu_\alpha + \mu_\beta^\top (\Sigma_\beta[t])^{-1} \mu_\beta,
\end{aligned}$$

which implies

$$(\Sigma_{\alpha\beta}[t])^{-1} = (\Sigma_\alpha[t])^{-1} + (\Sigma_\beta[t])^{-1} \quad \text{and} \tag{4.45a}$$

$$\mu_{\alpha\beta}[t] = \Sigma_{\alpha\beta}[t] \left((\Sigma_\alpha[t])^{-1} \mu_\alpha[t] + (\Sigma_\beta[t])^{-1} \mu_\beta[t]\right). \tag{4.45b}$$

4.3.4 Inverse Covariance Form

If the inverse covariance of the state distribution is singular, one must propagate inverse covariances rather than covariances (see page 239 of Maybeck[8]). Kailath et al. call this *The Information Form* (see page 332 of [5]). The procedure is useful when one makes the backwards pass through the data for smoothing because the initial inverse covariance is zero.

4.3.5 Extended Kalman Filter

Recall that Eqn. (4.1) on page 61 at the beginning of the chapter is

$$\begin{aligned} x[t+1] &= F(x[t],t) + \eta \\ y[t] &= G(x[t],t) + \epsilon, \end{aligned}$$

and that if the functions F and G are linear and the noise terms η and ϵ are independent and Gaussian that the Kalman filter implements the forward algorithm. If on the other hand, the functions F and G are nonlinear but the errors of first order approximations to them are small compared to the size of the state covariance, then one can reasonably apply the same algorithm to the approximations. The resulting procedure is called an *extended Kalman filter*. While, as we noted in Section 3.2 there are more robust alternatives, the simplicity of extended Kalman filters explains their frequent use. In Section 1.1 we applied an extended Kalman filter to laser measurements for the following purposes:

- Estimate model parameters
- Estimate state space trajectory
- Forecast measurements

And in the next chapter we will compare the likelihood of a model implemented as an extended Kalman filter to a performance bound obtained from Lyapunov exponent estimates.

Chapter 5

Toy Problems and Performance Bounds

Having developed algorithms for fitting model parameters, one might reasonably ask how well the models so produced perform. In this chapter we argue that the exercise of fitting models to data from chaotic dynamical systems is interesting because *Lyapunov exponent* calculations give a quantitative benchmark against which to compare model performance. The idea is that the stretching or local instability of dynamics, which Lyapunov exponents characterize, limits the predictability of sequences of observations. We will start by examining a toy example derived from the Lorenz system that informally introduces the main ideas. From there we will review definitions of entropy and Lyapunov exponents and results from information theory and ergodic theory that connect the ideas. Finally we explain a simple calculation that can determine that a proposed model is fundamentally suboptimal.

We suppose that a *true* stochastic process that assigns probabilities to sequences of observations exists. Many of the terms that we define are *expected values* with respect to those probabilities, and we find that sample sequences converge to those expected values. We use $P_{*|\mu}$ to denote these *true* probabilities[1], and we use them to define expected values without delving in to theoretical questions about their existence.

[1]Following Kolmogorov, modern probability theory is cast a subfield of measure theory. The measure theory literature uses the Greek letter μ for a function or *measure* that maps sets to $\mathbb{R}^+$. In earlier chapters, we have used $P_{*|\theta}$ to denote parametric families of distributions. We introduce the mongrel notation $P_{*|\mu}$ here to make our notation for comparisons between a true distribution and a parametric model natural. The meaning of the Greek letter μ here is not related to its use to denote the mean of a distribution.

Lorenz Example

As an example, we have simulated a version of the Lorenz-system (Eqn. (1.1)) modified to fit the form of Eqn. (4.1),

$$\begin{aligned} x[t+1] &= F(x[t], t) + \eta[t] \\ y[t] &= G(x[t], t) + \epsilon[t]). \end{aligned}$$

We have used the extended-Kalman-filter described in chapter 4 to obtain parametric probability functions $P\left(y[t] \mid y[0:t], \theta\right)$ that approximate $P\left(y[t] \mid y[0:t], \mu\right)$, i.e., the conditional distribution of the measurement at time t given all previous measurements. Our code for generating sequences of measurements has the following characteristics:

State

$$x \equiv \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

Time step We obtain the map F by numerically integrating Eqn. (1.1) for time intervals of length τ_s with an absolute error tolerance of 10^{-8}.

iid state noise

$$\eta[t] \sim \mathcal{N}\left(0, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \sigma_\eta^2\right)$$

Measurement function A simple projection

$$G(x) = x_0 = \begin{bmatrix} 1, 0, 0 \end{bmatrix} \cdot x$$

iid measurement noise

$$\epsilon[t] \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Quantization The observations are quantized with a resolution $\Delta = 10^{-4}$. We analyze quantized measurements rather than continuous measurements because they provide a *finite* rather than *infinite* amount of information and they are characterized by *coordinate invariant* probability mass functions rather than *coordinate dependent* probability density functions.

Recall that for the extended Kalman filter the means $\mu_\gamma[t]$ and variances $\sigma_\gamma^2[t]$ completely characterize $P(y[t] \mid y[0:t], \theta)$ with

$$P(y[t] \mid y[0:t], \theta) = \mathcal{N}\left(\mu_\gamma[t], \sigma_\gamma^2[t]\right)\big|_{y[t]}.$$

(We use a lower case sigma here because the observations are scalars.) We obtain affine maps for the approximation $F(x+\delta, t) \approx \left[DF(x)\right]\delta + F(x)$ by numerically integrating both the Lorenz system and the tangent equations. We use those approximations with Eqns. (4.23a) and (4.24) on page 66 to implement

the recursive calculation of $\mu_\gamma[t]$ and $\sigma_\gamma[t]$ described by Eqns. (4.7) to (4.16) on page 63.

Figures 5.1 and 5.2 depict a simulation in which dynamical stretching, i.e., the linear instability of $[DF(x)]$, occasionally limits predictability. We chose the parameters, specified in the caption of Fig. 5.1, so that dynamical noise and measurement quantization are negligible compared to the effects of measurement noise and dynamical stretching. In the middle plot of Fig. 5.1 notice that while for most times the forecast deviation of the prediction $\sigma_\gamma[t]$ is very close to the size of the state noise **ToDo:** or close to the measurement noise? $\sigma_\eta = 0.020$, occasionally the forecast deviations are many times larger. The log likelihood per time step which appears in the bottom plot of the figure is low when either the forecast deviations are large or when the difference between the mean of the forecast and the actual observation is much larger than the predicted deviation, i.e., $\sigma_\gamma^2[t] << (y[t] - \mu_\gamma[t])^2$.

The largest excursion of $\sigma_\gamma[t]$ in Fig. 5.1 occurs at $t = 20$. Figure 5.2 illustrates the stretching action of the map $[DF]$ that occurs then.

Figure 5.3 illustrates the behavior of $-\hat{h}$, the sample average of the log likelihood of forecasts, for a group of simulations with parameters that are quite different from those in Figs. 5.1 and 5.2. Given model parameters θ and a sample sequence $y[0:T]$ of length T, we define

$$\begin{aligned} -\hat{h} &\equiv \frac{1}{T}\sum_{t=0}^{T-1} \log\left(P\left(y[t] \mid y[0:t], \theta\right)\right) \\ &= \frac{1}{T}\log\left(P\left(y[0:T] \mid \theta\right)\right). \end{aligned}$$

The negative of this sample log likelihood is an estimate of the *cross entropy rate*

$$h(\mu||\theta) \equiv \lim_{T\to\infty} -\frac{1}{T}\mathbb{E}_\mu \log\left(P\left(Y[0:T] \mid \theta\right)\right)$$

which in turn is bounded from below by the *entropy rate*. We discuss both entropy rate and cross entropy rate in Section 5.2, and in Section 5.3 we review the *Pesin Formula*. That formula says that the largest *Lyapunov exponent* λ_0, a characterization of the stretching action of the dynamics, is equal to the entropy rate, a characterization of the predictability. For good models, we expect the log likelihood of forecasts to fall off with a slope of $-\lambda_0$ as the sample time τ_s increases, and for the best possible model we expect

$$h(\tau_s) = \lambda_0 \tau_s. \tag{5.1}$$

We have chosen the parameters[2] specified in the caption of Fig. 5.3 with a measurement quantization size large enough that the log likelihood is limited primarily by dynamical stretching and the Gaussian model for $P\left(y[t] \mid y[0:t], \theta\right)$.

[2]In making Fig. 5.3, we wanted simulations close to the bound of Eqn. (5.1). We found that at larger values of τ_s and Δ, extended Kalman filters performed better if given models with larger state noise than the noise actually used to generate the data, i.e. $\tilde{\sigma}_\eta > \sigma_\eta$. We believe that the effect is the result of the larger errors that occur as the affine approximation $F(x+\delta) \approx [DF(x)]\delta + F(x)$ fails to track the nonlinearities of the Lorenz system over larger

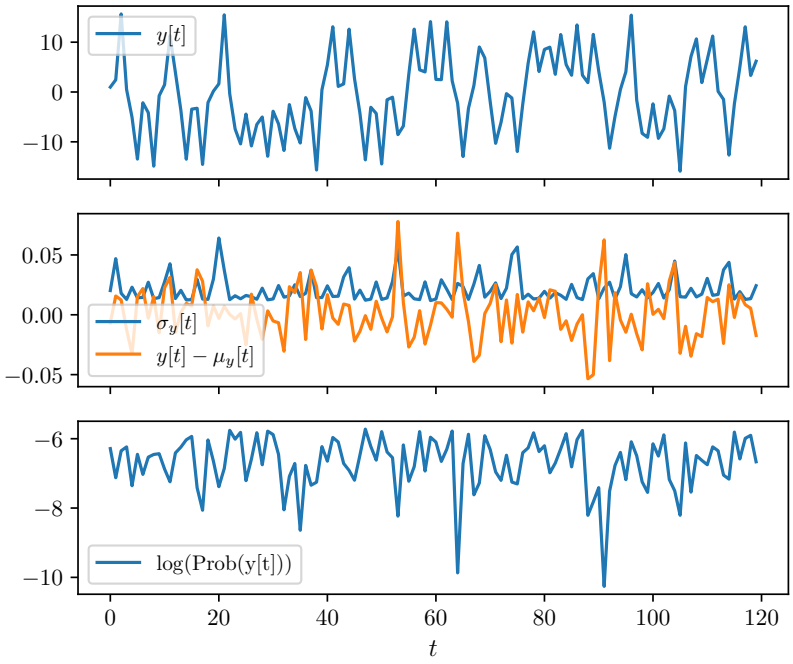

Figure 5.1: Extended Kalman filter for one step forecasting with simulation parameters:

$\tau_s = 0.250$	Sample interval
$\sigma_\eta = 0.020$	Standard deviation of state noise
$\sigma_\epsilon = 0.0030$	Standard deviation of measurement noise
$\Delta = 10^{-4}$	Measurement quantization

A time series of observations appears in the upper plot. The middle plot characterizes the one-step forecast distributions $P_\gamma(y[t]) \equiv P(y[t] \mid y[0:t], \theta) = \mathcal{N}\left(\mu_\gamma[t], \sigma_\gamma^2[t]\right)\big|_{y[t]}$. The standard deviations of the forecasts appear in the first trace, and the differences between the actual observations and the means of the forecasts appear in the second trace. The logs of the likelihoods of the forecasts, $\log(P_\gamma(y[t]))$, appear in the bottom plot.

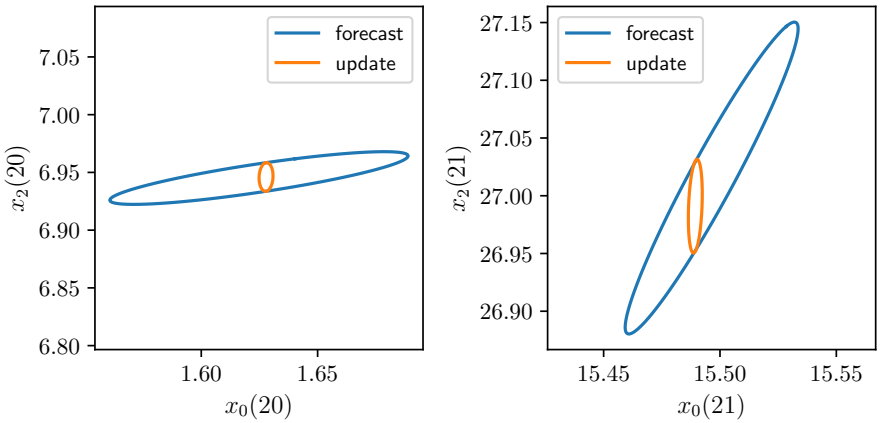

Figure 5.2: These plots illustrate dynamical stretching increasing the variance of the conditional distribution in state space between time steps 20 and 21 in Fig. 5.1. In each plot, the *forecast* state distribution ellipse represents $P_a\left(x[t]\right) \equiv P\left(x[t] \mid y[0:t], \theta\right) = \mathcal{N}\left(\mu_a, \Sigma_a\right)|_{x[t]}$ and the *update* state distribution ellipse represents $P_\alpha\left(x[t]\right) \equiv P\left(x[t] \mid y[0:t+1], \theta\right) = \mathcal{N}\left(\mu_\alpha, \Sigma_\alpha\right)|_{x[t]}$. For each distribution, an ellipse depicts the level set $(x-\mu)^\top \Sigma^{-1}(x-\mu) = 1$ in the $x_0 \times x_2$ plane. To support visual comparisons, the sizes of the ranges for x_0 and x_2 are the same in each of the plots. In mapping the updated distribution at $t = 20$ (on the left) to the forecast distribution at time $t = 21$ (on the right), the function F that implements state dynamics stretches the ellipse by about a factor of 10 in both directions.

We are pleased to observe that the overall slope of the plot on the left in Fig. 5.3 is consistent with the estimate $\hat{\lambda}_0 = 0.901$ (base e) that we obtain using the Benettin procedure described in Section 5.4.

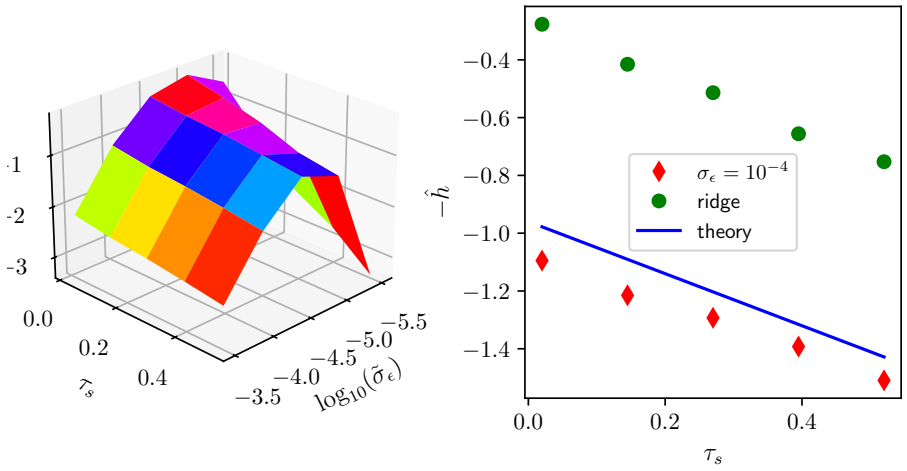

Figure 5.3: Average log likelihood of one step forecasts as a function of time step τ_s and filter parameter $\tilde{\sigma}_\epsilon$. To **ToDo:** Check code that produces these values simulate measurements for this figure, we used the parameters:

$\sigma_\eta = 10^{-6}$	Standard deviation of state noise
$\sigma_\epsilon = 10^{-10}$	Standard deviation of measurement noise
$\Delta = 10^{-4}$	Measurement quantization
$T = 200$	Number of samples

For both plots, the vertical axis is the average log likelihood of the one-step forecast $-\hat{h} \equiv \frac{1}{T}\sum_{t=0}^{T-1} \log\left(P\left(y[t] \mid y[0:t], \theta\right)\right)$. On the left we plot $-\hat{h}$ as a function of both τ_s, the time step, and $\tilde{\sigma}_\epsilon$, the standard deviation of the measurement noise model used by the extended Kalman filter. On the right the top row of dots indicates the performance of filters that use measurement noise models that depend on the sampling time through the formula $\tilde{\sigma}_\epsilon(\tau_s) = 10^{m\cdot\tau_s+b}$, with $m = 0.4$ and $b = -4.85$ chosen by hand to follow the ridge top in the plot on the left. Also on the right, the bottom row of dots indicates the performance of filters that use $\tilde{\sigma}_\epsilon = 10^{-4}$, i.e. the measurement quantization level, and the solid line traces Eqn. (5.2) in the text.

In the plot on the right in Fig. 5.3 we see that for a class of filters in which the standard deviation of the model measurement noise $\tilde{\sigma}_\epsilon$ is set to the quantization

intervals in state space. By making $\tilde{\sigma}_\eta$ larger, the errors are accommodated as state noise. We chose the state noise of the generating process to be an order of magnitude larger than the absolute integration tolerance of 10^{-8}. We then chose the quantization level and sample times to be as large as possible, but still small enough that we could have $\tilde{\sigma}_\eta = \sigma_\eta$ without losing performance. That led to the values $\tilde{\sigma}_\eta = \sigma_\eta = 10^{-6}$, $\Delta = 10^{-4}$, and $0 < \tau_s \leq 0.5$.

size Δ, the log likelihood closely follows the approximation

$$\hat{h}(\tau_s) = \log\left(\operatorname{erf}\left(\frac{1}{2\sqrt{2}}\right)\right) + \lambda_0 \tau_s, \tag{5.2}$$

where *erf* is the error function[3]. We explain the nonzero intercept in Eqn. (5.2) by observing that in the limit of small sampling interval ($\tau_s \to 0$) and zero noise ($\sigma_\eta \to 0$ and $\sigma_\epsilon \to 0$), only one discrete observation $y[t] = \bar{y}$ is possible given a history $y[0:t]$. For data drawn from that limiting case, a Kalman filter with parameters $\tilde{\sigma}_\epsilon = \Delta$ and $\sigma_\eta \to 0$ would make a forecast with a density $P(y[t] \mid y[0:t]) = \mathcal{N}\left(\bar{y}, (\Delta)^2\right)\big|_{y[t]}$. Integrating that density over the quantization interval yields

$$\begin{aligned}
P_{y[t]|\theta}(\bar{y}) &= \int_{\bar{y}-\frac{\Delta}{2}}^{\bar{y}+\frac{\Delta}{2}} \frac{1}{\sqrt{2\pi(\Delta)^2}} e^{-\frac{(y-\bar{y})^2}{2(\Delta)^2}} dy \\
&= \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}s^2} ds \\
&= \operatorname{erf}\left(\frac{1}{2\sqrt{2}}\right) \\
&\approx 0.3829 \\
\log\left(P_{y[t]|\theta}(\bar{y})\right) &\approx -0.9599.
\end{aligned}$$

Given the simplicity of the analysis, Eqn. (5.2) fits the simulations in Fig. 5.3 remarkably well.

5.1 Fidelity Criteria and Entropy

Stochastic models are fit to an enormous variety of measured phenomena and the most appropriate measure of the fidelity of a model to measurements depends on the application. Such phenomena include long and short term weather, financial markets, computer data, electric power demand, and signals and noise in communication or instrumentation. In many cases one makes decisions based on a model and those decisions change the *cost* of future events. The expected cost of basing decisions on a model $P_{*|\theta}$ depends in a complicated fashion on many factors including how the cost of acting on a decision depends on lead time and which aspects of the modeled phenomenon are important. For the Lorenz example at the beginning of this chapter we implicitly assumed *stationarity* and *ergodicity* and characterized model quality in terms of the average of the log of the likelihood. For a stationary ergodic system, the log-likelihood is tied to $D(\mu||\theta)$, the *relative entropy* of the model $P_{*|\theta}$ given $P_{*|\mu}$ (see Eqn. (5.5)). Although relative entropy is not an appropriate performance measure for every application, it is a common tool for problems in information theory and statistics. See Cover and Thomas[1] for many of these including the application of

[3]The error function is defined by $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$.

relative entropy to a theory of gambling. Relative entropy is exactly the right performance measure for data compression. The arithmetic coding algorithm (see the review by Witten, Neal, and Cleary[21]) for compressing a sequence of symbols uses a model $P_{*|\theta}$ to make decisions that affect the cost in a manner that depends on the symbol values that actually occur. The relative entropy $D(\mu||\theta)$ is the expected value of the number of bits wasted by the algorithm if it uses a model $P_{*|\theta}$ for decisions when in fact $P_{*|\mu}$ is true. More accurate models lead to better compression.

5.1.1 Definitions

Now, to solidify the discussion, we make some formal definitions.

Stochastic process

We are interested in sequences of states $X[0:T]$ and measurements $Y[0:T]$ each of which can be thought of as a *random function* on the domain $\{1, 2, \ldots, T\}$, i.e., a *stochastic process*.

Entropy of a discrete random variable

If a discrete random variable U takes on the values $u_0, u_1, \ldots, u_n$ with probabilities $P(u_0), P(u_1), \ldots, P(u_n)$, then the *entropy* of U is **ToDo:** should sum be from 0?

$$H(U) \equiv -\mathbb{E}\log\left(P(U)\right) = -\sum_{k=1}^{n} P(u_k)\log\left(P(u_k)\right). \tag{5.3}$$

Entropy quantifies the the uncertainty in U before its value is known and the information or degree of surprise in discovering its value. If the base of the logarithm in Eqn. (5.3) is 2, then the units of $H(U)$ are called *bits*. We will use natural logarithms with Euler constant e as the base. For natural logarithms the units of $H(U)$ are called *nats*.

Differential entropy of a continuous random variable

If U is a continuous random variable with a probability density function P then the *differential entropy* of U is

$$\tilde{H}(U) \equiv -\mathbb{E}\log\left(P(U)\right) = -\int P(u)\log\left(P(u)\right)du. \tag{5.4}$$

Notice that the differential entropy depends on the coordinates of U.

Conditional entropy

The *conditional entropy* of U given V is

$$H(U|V) \equiv -\mathbb{E}\log\left(P(U|V)\right) = -\sum_{i,j} P(u_i, v_j)\log\left(P(u_i|v_j)\right).$$

Factoring the probability of sequences

$$P_{z[0:T]} = P_{z[0]} \prod_{t=0}^{T-1} P_{z[t]|z[0:t]}$$

is equivalent analyzing entropy into the sum

$$H(Z[0:T]) = H(Z[0]) + \sum_{t=1}^{T-1} H\left(Z[t]|Z[0:t]\right).$$

Relative entropy of two probability functions

The *relative entropy* between two probability functions $P_{*|\mu}$ and $P_{*|\theta}$ with the same domain $\mathcal{Z}$ is

$$\begin{aligned} D(\mu||\theta) &\equiv \mathbb{E}_\mu \log\left(\frac{P(Z \mid \mu)}{P(Z \mid \theta)}\right) \\ &= \sum_{z\in\mathcal{Z}} P(z \mid \mu) \log\left(\frac{P(z \mid \mu)}{P(z \mid \theta)}\right). \end{aligned} \tag{5.5}$$

The relative entropy is coordinate independent. The relative entropy between two probability functions $P_{*|\mu}$ and $P_{*|\theta}$ is never negative and is zero if and only if the functions are the same on all sets with finite probability. We use $D(\mu||\theta)$ to characterize the fidelity of a model $P_{*|\theta}$ to a *true* distribution $P_{*|\mu}$

Cross entropy of two probability functions

While some authors use the terms *relative entropy* and *cross entropy* interchangeably to mean the quantity $D(\mu||\theta)$ that we defined in Eqn.(5.5), we define the cross entropy to be

$$\begin{aligned} H(\mu||\theta) &\equiv -\mathbb{E}_\mu \log\left(P(Z \mid \theta)\right) \\ &= -\sum_{z\in\mathcal{Z}} P(z \mid \mu) \log(P(z \mid \theta)) \end{aligned} \tag{5.6}$$

and note that

$$D(\mu||\theta) = H(\mu||\theta) - H(\mu).$$

The cross entropy is the negative expected log-likelihood of a model. It is greater than the entropy unless the model $P_{*|\theta}$ is the same as $P_{*|\mu}$ for all sets with finite probability.

Stationary

A stochastic process is *stationary* if probabilities are unchanged by constant shifts in time, i.e., for any two integers $T \geq 1$ and $\tau \geq 0$

$$P_{Z[0:T]} = P_{Z[0+\tau:T+\tau]}.$$

Ergodic

Roughly, in an *ergodic* process you can get anywhere from anywhere else. Let $\mathcal{X}$ be the set of states for a stationary stochastic process with probabilities $P_{*|\mu}$. The process is ergodic if for any two subsets of $\mathcal{X}$, A and B with $P(A \mid \mu) > 0$ and $P(B \mid \mu) > 0$ there is a time T such that the probability of going from set A to set B in time T is greater than zero. Birkhoff's ergodic theorem says that for an ergodic process, time averages converge to expected values with probability one.

Entropy rate

For a discrete stochastic process X, the entropy rate is

$$h(X) \equiv \lim_{T\to\infty} \frac{1}{T} H(X[0:T]). \tag{5.7}$$

If the process is stationary

$$h(X) = \lim_{T\to\infty} H(X[T]|X[0:T]). \tag{5.8}$$

If the process is stationary and Markov

$$h(X) = H(X[T+1]|X[T]) \ \forall T. \tag{5.9}$$

And if the process is ergodic

$$h(X) = \lim_{T\to\infty} -\frac{1}{T} \log\left(P\left(x[0:T] \mid \mu\right)\right) \tag{5.10}$$

with probability one.

We similarly define the relative entropy rate and the cross entropy rate. For an ergodic process X with true probabilities $P_{*|\mu}$ and model probabilities $P_{*|\theta}$

$$\begin{aligned} h(\mu||\theta) &\equiv \lim_{T\to\infty} -\frac{1}{T} \mathbb{E}_\mu \log\left(P\left(X[0:T] \mid \theta\right)\right) \\ &= \lim_{T\to\infty} -\frac{1}{T} \log\left(P\left(x[0:T] \mid \theta\right)\right) \end{aligned} \tag{5.11}$$

with probability one.

Entropy rate of a partition $\mathcal{B}$

Let $\mathcal{X}$, the set of states for a stationary stochastic process with probabilities $P_{*|\mu}$, be a continuum, and let $\mathcal{B} = \{\beta_0, \beta_1, \ldots \beta_n\}$ be a partition of $\mathcal{X}$ into a finite number of non-overlapping subsets. By setting $b[t]$ to the index of the element of $\mathcal{B}$ into which $x[t]$ falls, we can map any sequence $x[0:T]$ into a sequence $b[0:T]$ thus defining a discrete stationary stochastic process B. Applying the definition of entropy rate to the process B yields the definition of the entropy rate as a function of partition $\mathcal{B}$. Suppose in particular that on some set $\mathcal{X}$ the

map $F : \mathcal{X} \mapsto \mathcal{X}$ and the probability $P_{*|\mu}$ define an ergodic process, that $\mathcal{B}$ is a partition of $\mathcal{X}$, and that the model probability function $P_{*|\theta}$ assigns probabilities to sequences of partition indices $b[0:T]$. We define the entropy rate $h(\mathcal{B}, F, \mu)$ and the cross-entropy-rate $h(\mathcal{B}, F, \mu||\theta)$ as follows

$$\begin{aligned} h(\mathcal{B}, F, \mu) &\equiv \lim_{T\to\infty} \frac{1}{T} H(B[0:T]) \\ &= \lim_{T\to\infty} -\frac{1}{T} \mathbb{E}_\mu \log\left(P\left(B[0:T] \mid \mu\right)\right) \\ h(\mathcal{B}, F, \mu||\theta) &\equiv \lim_{T\to\infty} -\frac{1}{T} \mathbb{E}_\mu \log\left(P\left(B[0:T] \mid \theta\right)\right). \end{aligned} \tag{5.12}$$

Kolmogorov Sinai entropy h_{KS}

As before, suppose that on some set $\mathcal{X}$, the map $F : \mathcal{X} \mapsto \mathcal{X}$ and the probability $P_{*|\mu}$ define an ergodic process. The least upper bound over all partitions $\mathcal{B}$ on the entropy rate $h(\mathcal{B}, F, \mu)$ is the called the *Kolmogorov Sinai entropy*

$$h_{KS}(F, \mu) \equiv \sup_{\mathcal{B}} h(\mathcal{B}, F, \mu). \tag{5.13}$$

5.2 Stretching and Entropy

Here we will outline theory that connects ideas from dynamics to ideas from probability. The main results say that average dynamical stretching (Lyapunov exponents) is proportional to average uncertainty (entropy) in measurement sequences. First we will give some examples, then we will quote definitions and theorems without proof.

5.2.1 Maps of the unit circle

Two x mod one

The map of the unit interval $[0, 1)$ into itself defined by

$$x[n+1] = F_2(x[n]) \tag{5.14}$$

$$\equiv 2x \mod 1 \tag{5.15}$$

is continuous if we identify the points 0 and 1. Notice that if we use the partition

$$\mathcal{B}_2 = \left\{ \beta_0 = [0, \frac{1}{2}), \beta_1 = [\frac{1}{2}, 1) \right\}, \tag{5.16}$$

a symbol sequence $b[0:\infty]$ provides the coefficients of a base two power series that identifies the starting point, i.e.

$$x[0] = \sum_{t=0}^{\infty} b[t] \left(\frac{1}{2}\right)^{t+1}.$$

Further, if we assign a uniform probability measure μ to the interval, then

$$h_{KS}(F_2, \mu) = h(\mathcal{B}_2, F, \mu) = \log(2).$$

Three x mod one

Analogously, by defining the map

$$x[n+1] = F_3(x[n]) \tag{5.17}$$
$$\equiv 3x \mod 1, \tag{5.18}$$

and using the partition

$$\mathcal{B}_3 = \left\{ \beta_0 = [0, \frac{1}{3}), \beta_1 = [\frac{1}{3}, \frac{2}{3}), \beta_2 = [\frac{2}{3}, 1) \right\} \tag{5.19}$$

and a uniform probability measure μ, we find

$$h_{KS}(F_3, \mu) = h(\mathcal{B}_3, F_3, \mu) = \log(3).$$

This is the result that motivated Kolmogorov and Sinai to define the entropy h_{KS}. They were addressing the isomorphism problem, e.g., "Is there a relabeling of points in the unit interval that makes F_2 the same as F_3?". Since the characteristic h_{KS} is independent of the coordinates or labeling used, the fact that $h_{KS}(F_3, \mu) \neq h_{KS}(F_2, \mu)$ provided a negative answer to the question.

Notice that the Kolmogorov entropy is equal to the average of the log of the slope of the map. Specifically, the slope of F_2 is 2 and $h_{KS}(F_2, \mu) = \log(2)$ while the slope of F_3 is 3 and $h_{KS}(F_3, \mu) = \log(3)$. The rule that entropy is proportional to the log of the average slope is not true in general. The next example provides a counter example and suggests a correction factor.

Dynamics on a Cantor set

While every point in the entire unit interval can be represented as a base three power series, i.e.

$$\forall x \in [0, 1),\ \exists d_0^\infty :\ x = \sum_t^\infty d[t] \left(\frac{1}{3}\right)^{t+1} \text{ with } d[t] \in \{0, 1, 2\}\ \forall t,$$

the middle third Cantor set consists of the points in the unit interval that can be represented as base three power series that exclude the digit "1". The symbol sequences produced by applying the map F_3 and partition $\mathcal{B}_3$ to the middle third Cantor set are the sequences of coefficients in the base three expansions of the starting points, i.e., they consist exclusively of $0's$ and $2's$. Given any finite sequence $d[0:n]$, we define the set of infinite coefficient sequences $\{d[0:n], \ldots\}$ as those that begin with the sequence $d[0:n]$. Now we define a probability measure μ_c in terms of such sets of infinite sequences,

$$\mu_c\left(\{d[0:n], \ldots\}\right) \equiv \begin{cases} 2^{-(n+1)} & \text{if "1" does not appear in } d[0:n] \\ 0 & \text{if "1" does appear in } d[0:n] \end{cases} \tag{5.20}$$

With this measure we find

$$h_{KS}(F_3, \mu_c) = \log(2).$$

The following isomorphism or relabeling of the unit interval connects (F_2, μ) to (F_3, μ_c):

1. Find the binary expansion $b[0:\infty]$ of the original point x
2. Create a new sequence $d[0:\infty]$ by replacing every occurrence of "1" in $b[0:\infty]$ with "2"
3. Map x to y where y is described by the base three expansion $d[0:\infty]$

The Hausdorff-dimension[4] of the middle third Cantor set is $\delta = \frac{\log(2)}{\log(3)}$, and that is the factor that is missing in the formula connecting entropy and stretching.

$$\begin{aligned} h_{KS}(F_3, \mu_c) &= \log(2) \\ &= \frac{\log(2)}{\log(3)} \log(3) \\ &= \delta \log(\text{stretching factor}) \end{aligned} \tag{5.21}$$

Now we turn to the definitions and theorems that express the above idea precisely.

5.3 Lyapunov Exponents and Pesin's Formula

Vixie[24] has reviewed the work of Ruelle[43], Pesin[40], Young[22], and others who established the relationship between smooth dynamics and entropy. Here we reiterate a few of those results using the following notation:

X An n-dimensional manifold

$F : X \mapsto X$ An invertible C^2 (continuous with continuous first and second derivatives) map of the manifold into itself

μ A probability measure on X that is invariant under F

x A point on the manifold

$TX(x)$ The tangent space of X at x

v An element of $TX(x)$

We define the asymptotic growth rate of the direction v at x as

$$\lambda(F, x, v) \equiv \lim_{t \to \infty} \frac{1}{t} \log\left(\left\|[DF^t(x)]v\right\|\right). \tag{5.22}$$

[4]Sets and characteristics of sets with non-integer dimensions are sometimes called *fractal*.

Oseledec's-theorem [22, 6, 7] says that at almost every x the limit exists for every v, and that although the value of the limit depends on v, that as v varies, it only takes on $r \leq n$ discrete values called the *spectrum of Lyapunov exponents*

$$\lambda_0(F,x) > \lambda_1(F,x) > ... > \lambda_r(F,x). \tag{5.23}$$

The tangent space $TX(x)$ is the direct sum of subspaces $E_i \subset TX(x)$ associated with each exponent, with

$$\lambda(F,x,v) = \lambda_i(F,x) \ \forall v \in E_i$$

and

$$TX(x) = \bigoplus_{i=0}^{r-1} E_i.$$

The dimension of E_i is called the *multiplicity* m_i of the exponent λ_i. If μ is ergodic with respect to F, then the spectrum $\{\lambda_i\}$ is the same almost everywhere.

We want to use Pesin's-formula[40] which implies that if μ is smooth and ergodic, then the entropy is equal to the sum of the positive Lyapunov exponents, i.e.

$$h_{KS}(F,\mu) = \sum_{i:\lambda_i>0} m_i \lambda_i. \tag{5.24}$$

In light of the correction for fractal dimension that we saw in Eqn. (5.21) and the ubiquity of fractal measures in chaotic systems, we should review Ledrappier and Young's explanation (See [22] for an overview) of the effect of fractal measures on Pesin's formula.

Ledrappier and Young's formula is given in terms of the dimensions of the conditional measures on the nested family of *unstable foliations* of F. For a point $x \in X$ and i such that $\lambda_i > 0$ we define

$$W^i(x) \equiv \left\{ y \in X \text{ such that } \lim_{n\to\infty} \sup \frac{1}{t} \log\left(d\left(F^{-t}(x), F^{-t}(y)\right)\right) < -\lambda_i \right\}. \tag{5.25}$$

For an intuitive picture, consider a trajectory $x(t)$ that passes through x at time $t = 0$; any trajectory $y(t)$ that has separated from $x(t)$ at a rate of at least λ_i, passes through the manifold $W^i(x)$ at time $t = 0$.

Let δ_i be the Hausdorff dimension of the conditional measure that μ defines on $W^i(x)$. For an ergodic μ, δ_i will be constant almost everywhere. Further, let γ_i be the incremental dimension

$$\gamma_i \equiv \begin{cases} \delta_0 & i = 0 \\ \delta_i - \delta_{i-1} & i > 0 \end{cases}$$

Now Ledrappier-and-Young's-formula is

$$h_{KS}(F,\mu) = \sum_{i:\lambda_i>0} \lambda_i \gamma_i. \tag{5.26}$$

Note that Pesin's formula holds if the measure μ is smooth in the unstable directions. Such measures are called SRB (Sinai Ruelle Bowen) measures. Tucker has found that the Lorenz system has an SRB measure and says that numerical simulations of Lorenz's system are "real" [46].

5.3.1 A theoretical bound on model likelihood

Now we have the terms that we need to discuss theoretical bounds on the expected log likelihood of models of discrete observations of a chaotic dynamical system. Given X, F, and μ as described above, if the multiplicity of each exponent is $m_i = 1$ then we know:

$$h(\mathcal{B}, F, \mu) \equiv -\lim_{t\to\infty} \mathbb{E}_\mu \log\left(P\left(b[t] \mid b[0:t], \mu\right)\right) \tag{5.27}$$

$$h(\mathcal{B}, F, \mu||\theta) \equiv -\lim_{t\to\infty} \mathbb{E}_\mu \log\left(P\left(b[t] \mid b[0:t], \theta\right)\right) \tag{5.28}$$

$$h(\mathcal{B}, F, \mu) \leq h(\mathcal{B}, F, \mu||\theta) \qquad \text{Equality} \iff \theta = \mu \text{ a.e.} \tag{5.29}$$

$$h(\mathcal{B}, F, \mu) \leq h_{KS}(F, \mu) \qquad \text{Equality} \iff \mathcal{B} \text{ generating} \tag{5.30}$$

$$h_{KS}(F, \mu) = \sum_{i:\lambda_i>0} \lambda_i \gamma_i \tag{5.31}$$

$$h_{KS}(F, \mu) \leq \sum_{i:\lambda_i>0} \lambda_i \qquad \mu \text{ smooth on } W^i \Rightarrow \text{Equality} \tag{5.32}$$

with the following justifications

(5.27) and (5.28)**:** Definition

(5.29)**:** Gibbs inequality, (2.54)

(5.30)**:** The definition of $h_{KS}(F, \mu)$ is that it is the *supremum* over all partitions $\mathcal{B}$

(5.31)**:** This is Ledrappier and Young's formula (5.26)

(5.32)**:** Because in (5.31) $0 \leq \gamma_i \leq 1 \ \forall i$

Thus we have the following two theorems:

Theorem 1 (Lyapunov exponent bound on likelihood) *If μ is ergodic and smooth in the unstable directions and $\mathcal{B}$ is a generating partition, then for any model θ of the stochastic process B consisting of F, μ, and $\mathcal{B}$*

$$h(\mathcal{B}, F, \mu||\theta) \geq \sum_{i:\lambda_i>0} \lambda_i = h(\mathcal{B}, F, \mu) \tag{5.33}$$

Theorem 2 (Entropy gap) *If μ is ergodic (not necessarily smooth in the unstable directions). Then for an optimal model θ of the stochastic process B consisting of F, μ, and $\mathcal{B}$ ($\mathcal{B}$ not necessarily generating)*

$$h(\mathcal{B}, F, \mu||\theta) = h(\mathcal{B}, F, \mu) \leq \chi \equiv \sum_{i:\lambda_i>0} \lambda_i \tag{5.34}$$

and if for some other model ν

$$h(\mathcal{B}, F, \mu||\nu) \geq \chi \tag{5.35}$$

then the model ν is not optimal.

In the next section, we will describe a numerical procedure for estimating Lyapunov exponents, and in the following section we will argue that one can reasonably use Eqn. (5.35) with numerical simulations to quantitatively characterize the non-optimality of a model.

5.4 Benettin's Procedure for Calculating Lyapunov Exponents Numerically

We begin reviewing Benettin's-procedure[27] for estimating Lyapunov exponents by using the Lorenz system as an example. The Lorenz system is

$$\dot{x} = F(x) = \begin{bmatrix} s(x_1 - x_0) \\ x_0(r - x_2) - x_1 \\ x_0 x_1 - b x_2. \end{bmatrix} \tag{5.36a}$$

Note that

$$DF(x) = \begin{bmatrix} -s & s & 0 \\ r - x_2 & -1 & -x_0 \\ x_1 & x_0 & -b \end{bmatrix} \tag{5.36b}$$

where $(DF(x))_{i,j} \equiv \frac{\partial F_i(x)}{\partial x_j}$. Let Φ denote solutions to the differential equation with

$$x[\tau] \equiv \Phi(x[0], \tau).$$

Lyapunov exponents are defined (recall Eqn. (5.22)) in terms of the long time behavior of the derivative matrix

$$\mathcal{D}[x[0], \tau] \equiv D_{x[0]}\Phi(x[0], \tau).$$

Interchanging the order of differentiation with respect to $x[0]$ and τ and applying the chain rule yields a *linear* differential equation for $\mathcal{D}$:

$$\begin{aligned} \dot{\mathcal{D}}(x[0], \tau) &= \frac{d}{d\tau} D_{x[0]}\Phi(x[0], \tau) \\ &= DF(x)|_{x=\Phi(x[0],\tau)}\, \mathcal{D}(x[0], \tau). \end{aligned}$$

Thus, given initial conditions $x[0]$ and $\mathcal{D}[0] = \mathbf{I}$ one can use an off-the-shelf routine to find $\begin{bmatrix} x[\tau] \\ \mathcal{D}[\tau] \end{bmatrix}$ by integrating

$$\begin{bmatrix} \dot{x[\tau]} \\ \dot{\mathcal{D}[\tau]} \end{bmatrix} = \begin{bmatrix} F(x) \\ [DF(x)]\mathcal{D} \end{bmatrix}. \tag{5.37}$$

Given a computer with infinite precision, for a range of time intervals τ, one could:

1. Integrate Eqn. (5.37) to obtain $\mathcal{D}[\tau]$
2. Do singular value decompositions (SVD's)

$$U[\tau]S[\tau]V^{\top}[\tau] = \mathcal{D}[\tau], \tag{5.38}$$

   where $U[\tau]$ and $V[\tau]$ are orthogonal and $S[\tau]$ is diagonal

3. Look for approximate convergence of the finite time Lyapunov exponent estimates:

$$\tilde{\lambda}_i[\tau] \equiv \frac{1}{\tau}\log(S_{i,i}[\tau]). \tag{5.39}$$

On a real computer, the procedure fails because the ratio of the largest and smallest singular values $\frac{S_0[\tau]}{S_d[\tau]}$ grows exponentially with τ and becomes larger than the precision of the machine.

Rather than using an SVD decomposition for each τ in step 2 above, one could use a QR decomposition:

$$Q[\tau]R[\tau] = \mathcal{D}[\tau]. \tag{5.40}$$

A QR decomposition factors the matrix $\mathcal{D}[\tau]$ into a product of two matrices the first of which $Q[\tau]$ is orthogonal and the second of which $R[\tau]$ is upper triangular. One could use the intuitive Gram Schmidt procedure, but there are algorithms that behave better numerically (see, e.g. [4] or [12]). Although the diagonal elements of $R[\tau]$ are not equal to the diagonal elements of $S[\tau]$, the finite time estimates

$$\hat{\lambda}_i[\tau] \equiv \frac{1}{\tau}\log(|R_{i,i}[\tau]|) \tag{5.41}$$

and the $\tilde{\lambda}_i[\tau]$ defined in Eqn. (5.39) converge to the same values[5].

Using Eqn. (5.41) does not address the problem of finite machine precision for long time intervals τ, but Benettin et al. recommend calculating $\log(|R_{i,i}[\tau]|)$

[5]In the SVD of Eqn. (5.38), the first column of $V[\tau]$ specifies the direction of the initial vector in the tangent space with the largest stretching. The exponential stretching rate is the Lyapunov exponent λ_0. However, with probability one, a randomly chosen vector will have the same stretching rate. The estimate $\hat{\lambda}_0[\tau]$ of Eqn (5.41) is based on the stretching rate of the first standard basis vector, i.e., $[1, 0, 0]$. Similar arguments using the growth rates of areas, volumes, hyper-volumes, etc. support using the estimates $\hat{\lambda}_i[\tau]$ of Eqn (5.41) for $i = 2, 3, \ldots$

by breaking the interval into N smaller steps of duration $\Delta\tau$ in a way that does address finite precision. Letting $A[n]$ denote the one time step derivative

$$A[n] \equiv D\Phi(x[(n-1)\Delta\tau], \Delta\tau) \tag{5.42}$$

the chain rule implies

$$D\Phi(x[0], N\Delta\tau) = \prod_{n=1}^{N} A[n].$$

If, for each n, one calculates[6] the pair $(Q[n], r[n])$ defined by

$$\begin{aligned} Q[0] &= \mathbf{I} \\ Q[n]r[n] &\equiv A[n]Q[n-1], \end{aligned}$$

where $Q[n]$ and $r[n]$ are obtained by a QR factorization of the product $A[n]Q[n-1]$, then induction yields

$$\prod_{n=1}^{N} A[n] = Q[N] \prod_{n=1}^{N} r[n].$$

Since $\prod_{n=1}^{N} r[n]$ is upper triangular, we have the QR factorization

$$D\Phi(x[0], N\Delta\tau) = Q[N]R[N] \tag{5.43}$$

$$R[N] = \prod_{n=1}^{N} r[n]. \tag{5.44}$$

Because the diagonal elements are the products $R_{i,i}[N] = \prod_{n=1}^{N} r_{i,i}[n]$, their logs are the sums

$$\log(|R_{i,i}[n]|) = \sum_{n=1}^{N} \log(|r_{i,i}[n]|). \tag{5.45}$$

Substituting this result into Eqn. (5.41) constitutes the Benettin procedure. The action of a matrix on a unit square is factored into components Q and R and sketched in Fig. 5.4. Results of applying the procedure to the Lorenz system appear in Fig. 5.5.

[6] To calculate $Q[n]$ and $r[n]$ for each n, one can either:

1. Integrate Eqn. (5.37) for a time interval $\Delta\tau$ with the initial condition $\begin{bmatrix} x[(n-1)\Delta\tau] \\ Q[n-1] \end{bmatrix}$ to obtain $\begin{bmatrix} x[n\Delta\tau] \\ A[n]Q[n-1] \end{bmatrix}$ and then calculate a QR factorization of $A[n]Q[n-1]$, the second component of the result.
2. As above, but use the identity matrix instead of $Q[n-1]$ as the second component of the initial condition for the integration which yields the result $\begin{bmatrix} x[n\Delta\tau] \\ A[n] \end{bmatrix}$, then calculate a QR factorization of the product $A[n]Q[n-1]$

$[[e_0] \quad [e_1]]$ $\qquad$ $R\,[[e_0] \quad [e_1]]$ $\qquad$ $QR\,[[e_0] \quad [e_1]]$

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\qquad$ $\begin{bmatrix} 5 & 0.15 \\ 0 & 0.2 \end{bmatrix}$ $\qquad$ $\begin{bmatrix} 3 & 0.25 \\ -4 & 0 \end{bmatrix}$

Figure 5.4: The action of the Q R factors of a matrix on a unit square. Here $A = \begin{bmatrix} 3 & 0.25 \\ -4 & 0 \end{bmatrix}$, $Q = \begin{bmatrix} 0.6 & 0.8 \\ -0.8 & 0.6 \end{bmatrix}$, and $R = \begin{bmatrix} 5 & 0.15 \\ 0 & 0.2 \end{bmatrix}$. R stretches the x component by a factor of five and shears y components in the x direction and shrinks them by a factor of five with a net effect of preserving areas (The determinants of A and R are both 1.0). Q simply rotates the stretched figure.

5.5 A Practical Performance Bound

Consider the following two cases:

- State space dynamics perturbed by noise
- Simulated dynamics perturbed by numerical truncation

The definition of Kolmogorov-Sinai-entropy in the two cases yields extremely different answers. If the perturbations are random noise, then the supremum of $h(\mathcal{B}, F, \mu)$ over $\mathcal{B}$ does not exist and h_{KS} is unbounded. On the other hand, if the perturbations are numerical truncation and the process is a digital simulation, then all observation sequences converge to periodic cycles and $h_{KS} = 0$. Thus, the *strict* definition of the Kolmogorov Sinai entropy is useless as a bound on the cross entropy of models in numerical simulations. Here we argue however, that numerical Lyapunov exponent estimates nonetheless provide a *practical* reference for the performance of models.

If you are working on a new model building procedure that takes *training* samples $\{y[\tau_0 : T_0], y[\tau_1 : T_1], \ldots, y[\tau_N : T_N],\}$ and produces a family of parameterized conditional probability functions $P\left(y[t] \mid y[0 : t], \theta\right)$, we recommend numerically estimating the *entropy gap* (see Theorem 2) $\delta_{\mu||\theta} = h(\mathcal{B}, F, \mu||\theta) - \sum_{i:\lambda_i>0} \lambda_i$ to characterize the fidelity of the resulting models θ to generating processes. As a debugging tool, it is reasonable to choose some parameters θ' for a model class, use that model to generate training data, and then verify that as the size of the training data set increases the proposed model building procedure recovers the parameters θ'. However such a test fails to consider how well the proposed procedure and model class work on the realistic case of data generated by processes outside the model class. Even though the test we propose does not provide *correct* model parameters against which to compare fitted parameters, it does provide a reference against which to compare model performance. Specifically, we advocate the following numerical experiment for evaluating a model building procedure:

1. Use Φ obtained by numerically integrating a chaotic dynamical system with a sampling interval $\Delta\tau$ to generate training data and testing data. For simplicity, consider a system with a single positive exponent λ_0
2. Quantize the data with a partition $\mathcal{B}$
3. Run the Benettin procedure on the system, to estimate Lyapunov exponents
4. Substitute the estimated exponents into Ledrappier and Young's formula, Eqn. (5.26) with $\gamma_0 = 1$ to get $\hat{h}(\Phi, \mu)) = \hat{\lambda}_0$, an estimated entropy rate. If $\mathcal{B}$ is fine enough or a generating partition, $\hat{h}(\mathcal{B}, \Phi, \mu)) = \hat{\lambda}_0$ will be a good estimate.
5. Produce $P_{*|\theta}$ by applying the new model building procedure to the training data

6. Estimate the cross entropy by evaluating the likelihood of the model on long sequences of testing data

$$\hat{h}(\mathcal{B}, \Phi, \mu||\theta) = \frac{-1}{T} \sum_{t=1}^{T} \log\left(P\left(y[t] \mid y[0:t], \theta\right)\right) \tag{5.46}$$

7. Calculate an entropy gap by subtracting the two estimates

$$\hat{\delta}_{\mu||\theta} = \hat{h}(\mathcal{B}, \Phi, \mu||\theta) - \hat{h}(\mathcal{B}, \Phi, \mu)).$$

   For an optimal model, expect the gap to be zero. If the gap is much larger than zero, conclude that the new procedure is suboptimal.

The test is reasonable only if, subject to some constraints, $\mathbb{E}\hat{\delta}_{\mu||\theta} \geq 0$ is a tight bound and the variance of $\hat{\delta}_{\mu||\theta}$ is small. Below, we argue that a model that uses knowledge of the generating process and has smooth probability densities in state space achieves the bound with equality and thus the bound is tight.

In this class of models, the probability measure for the generating process is not necessarily ergodic or even stationary; it is derived from a uniform density over a box that covers possible starting conditions, and it includes a little bit of noise **ToDo:** I don't like *a little bit of noise.* in the dynamics so that even in the long time limit it does not become fractal. Because the probability is smooth, the models cannot exploit fractal properties that might exist in the modeled system and consequently γ, the Ledrappier and Young correction to the Pesin formula, is irrelevant. More specifically we consider a model class with the following properties:

Probability density The probability density for the initial state $P(x[1] \mid \theta)$ is a uniform distribution on a cube in X that has length $L_{\text{i.c.}}$ on each side.

State noise The model has noise in the state dynamics,

$$x[t+1] = \Phi(x[t]) + \eta[t], \tag{5.47}$$

where $\eta[t]$ are i. i. d. Gaussian with $\eta[t] \sim \mathcal{N}(0, \mathbf{I}\sigma_\eta^2)$. We suppose that Φ is the same as the *true* function of the *modeled* system, but that noise in the modeled system is smaller or zero.

Measurement function We let the measurement function be the same for the model as for the true system, i.e., a discrete partition with resolution Δ. We have in mind a uniform quantization of a single component of X such as we used for Fig. 5.3.

The only difference between the true system and the model is that the state noise in the model may be larger than the state noise in the true system.

With this framework we can draw samples randomly from a true distribution $P_{*|\mu}$ and consider model probabilities $P_{*|\theta}$ without having to find a stationary distribution. In sidestepping the key issue of a stationary distribution, we have

sacrificed ergodicity which is the basis of the definition of a Lyapunov exponent as a global property. Empirically, however, the convergence of the Benettin procedure is similar for any initial condition (See Fig. 5.5). Relying on this empirical observation, we suppose for some time interval τ that the stretching factor is roughly independent of initial conditions with

$$S[\tau] \approx \prod_{t=1}^{\tau-1} r_{0,0}[t] = e^{\hat{\lambda}\tau}. \tag{5.48}$$

In the limit of small noise $\sigma_\eta \to 0$, one can calculate $P(y[0:T] \mid \theta)$ for any sequence of observations as the probability of the set of initial conditions that are consistent with $y[0:T]$, i.e., the pre-image of $y[0:T]$,

$$P(y[0:T] \mid \theta) = \int_{\{x:Y[0:T](x)=y[0:T]\}} P(x \mid \theta) dx.$$

For a d dimensional system, the volume of such pre-images is typically

$$\frac{\Delta e^{-\hat{\lambda}T}}{O(T^d)} < \text{Vol} < \Delta e^{-\hat{\lambda}T},$$

and since the density of initial conditions is smooth, for large T we find

$$\frac{1}{T} \log\left(P(y[0:T] \mid \theta)\right) \approx -\hat{\lambda}. \tag{5.49a}$$

Rather than going backwards in time to analyze the pre-image of $y[0:T]$, we can think about the forward image of the volume of initial conditions under the map $\Phi(T)$. To first order, the distribution is a uniform probability density over a parallelepiped **ToDo:** What is $L_{\text{i.c.}}$? that extends a distance $L_{\text{i.c.}} s[T]$ in the direction of the first column of the orthogonal matrix $Q[t]$ in Eqn. (5.43). The measurement partition divides the image into elements that have a characteristic size of Δ, yielding

$$\frac{1}{T} \log\left(P(y[0:T] \mid \theta)\right) \approx -\hat{\lambda} \tag{5.49b}$$

again. Given the enormous stretching that occurs, it is clear that the image of the volume of allowed initial conditions will resemble steel wool more than a parallelepiped, but the exponential nature of the stretching is all that matters, and in the small noise limit we have

$$h(\mathcal{B}, \Phi, \mu || \theta) \approx \hat{\lambda} \tag{5.50}$$

One might wonder whether finite state noise σ_η invalidates (5.50) or if perhaps a correction of order $\frac{\sigma_\eta}{\Delta}$ will suffice. As long as $\sigma_\eta << L_{\text{i.c.}} e^{\hat{\lambda}T}$, the analysis of the size of the image of the volume of initial conditions under $\Phi(T)$ that leads to (5.49b) is adequate. The noise term σ_η in the model will however transfer probability from observation sequences permitted by the true system

to sequences that it does not allow, thereby increasing the cross entropy. At each time step the effect roughly augments the stretching in each state space direction with a term of size $\frac{\sigma_\eta}{\Delta}$. Since the noise at each time is independent of the noise at all other times, the effects add in quadrature. We can estimate an upper bound on the total effect by replacing $|r_{i,i}[n]|$ with $|r_{i,i}[n]| + \frac{\sigma_\eta}{\Delta}$ for each i and n in Eqn. (5.45) of the Benettin procedure, i.e.,

$$\hat{\lambda}_{\text{aug},i} \equiv \frac{1}{N} \sum_{n=1}^{N} \log(|r_{i,i}[n]| + \frac{\sigma_\eta}{\Delta}). \tag{5.51}$$

Notice that knowledge of $\hat{\lambda}_i$ and $\frac{\sigma_\eta}{\Delta}$, is not sufficient to calculate $\hat{\lambda}_{\text{aug},i}$. If the stretching were uniform with $|r_{i,i}[n]| = e^\lambda \ \forall n$, the augmented result would be $\hat{\lambda}_{\text{aug},i} = \log\left(e^\lambda + \frac{\sigma_\eta}{\Delta}\right)$, but the result increases without bound[7] as $|r_{i,i}[n]|$ varies more with n. In Fig. 5.5 we compare $\hat{\lambda}_{\text{aug},1}$ with $\hat{\lambda}_1$ for the Lorenz system. For noise with an amplitude $\frac{\sigma_\eta}{\Delta} = 0.0100$, the figure indicates an augmentation of $\hat{\lambda}_{\text{aug},0} - \hat{\lambda}_0 \approx 0.068$, which is roughly an order of magnitude larger than the augmentation that uniform stretching would produce. From the figure, we conclude that the Benettin procedure produces a robust practical upper bound on model performance.

Specifically, integrating the Lorenz system, (5.36), over intervals of $\Delta\tau = 0.150$ to create Φ we find

$$\hat{\lambda}_0(\Phi) = 0.135 \text{ nats} = 0.195 \text{ bits}$$
$$\hat{\lambda}_0(F) = \frac{\hat{\lambda}_0(\Phi)}{\Delta\tau} = 0.901 \text{ nats}$$

which is close to the value of 0.906 obtained by more careful calculations.

5.6 Approaching the Bound

Although the *slope* of the plot in Fig. 5.3 (the log likelihood per time step attained by extended Kalman filters) matches the entropy bound, and we are satisfied with our explanation of the nonzero intercept, an example of a model with a likelihood close to the bound without any offset would be more satisfying. To find such model, we return to the source of coarsely quantized Lorenz observations that we used for Fig. 1.10 in the introduction. That figure illustrates

[7] If,

$$|r_{i,i}[n]| = \begin{cases} \frac{1}{\delta^{N-1}} e^\lambda & n = 1 \\ \delta e^\lambda & \text{otherwise} \end{cases},$$

then $\prod_{n=1}^{N} |r_{i,i}[n]| = e^{N\lambda}$ and $\hat{\lambda}_i = \lambda$, but

$$\hat{\lambda}_{\text{aug},i} = \frac{1}{N}\left(\log\left(\frac{e^\lambda}{\delta^{N-1}} + \frac{\sigma_\eta}{\Delta}\right) + (N-1)\log\left(\delta e^\lambda + \frac{\sigma_\eta}{\Delta}\right)\right)$$
$$\lim_{\delta \to 0} = \frac{1}{N}\left(\lambda + (N-1)\log\left(\frac{\sigma_\eta}{\delta\Delta}\right)\right) \to \infty.$$

the association of each of the twelve discrete hidden states of an HMM with particular regions in $\mathbb{R}^3$, the state space of the Lorenz system. Although the cross entropy of that twelve state model is not very close to the bound based on our Lyapunov exponent estimate, it seems plausible that by using HMMs with more states we might get higher likelihoods. In fact it is true, but we are surprised at how many states we need. As for Fig. 1.10, we generated the observations by integrating the Lorenz system with a time step of $\tau_{\texttt{sample}} = 0.150$ and quantized the first component into one of 4 levels.

Our first attempt was to train a sequence of models with ever more hidden states. We initialized each model randomly and ran many iterations of the Baum-Welch algorithm on quantized observations. Even with many iterations, we did not build any promising models.

In our second attempt, we exploited our knowledge of the Lorenz dynamics in $X = \mathbb{R}^3$ as follows:

1. Generate training data $x[0:T]$ and $y[0:T]$ by integrating the Lorenz system. Here $x[t]$ is a point in the original state space and $y[t]$ is a quantized observation that can take one of four values.

2. Generate testing data $y[T:T+N]$ by continuing the integration.

3. Find a set of discrete states $\{s_0, s_1, \ldots, s_m\}$ by partitioning the original space with a uniform grid of resolution Δ_x. We only constructed states for those partition elements that were occupied by at least one member of $x[0:T]$.

4. Build an HMM using the training sequence. We set the state transition probabilities by counting the frequency with which the partitioned X sequence made each possible transition. Similarly, we set the observation model by counting the frequency with which each partition element was associated with each possible observation.

5. Estimate the cross entropy ($\hat{h}(\mathcal{B}, F, \mu||\theta)$ See Eqn. (5.46)) of the model by calculating its log likelihood per time step on the testing data.

As hoped, we found that as we reduced the resolution, the number of states increased and the cross entropy estimates decreased. Since there is no training, i.e., Baum-Welch iterations, in this procedure, we could calculate the likelihood with a variant of the forward algorithm that does not store or return α or γ values for the entire sequence. In fact, given typical observations $y[0:t+1]$ up to time t, only a small fraction of the states have nonzero probability. Code that uses sparse matrices to exploit these features is many orders of magnitude cheaper than vanilla code for the forward algorithm.

Results of the procedure appear in Fig. 5.6. For the rightmost point on the curve[8] we found a model with 4,890,280 hidden states and a cross entropy,

[8]While the address space limit of 32 bit Python constrained the number of states for the first edition of this book to about 1,500,000, computation time constrains the number of number of states in 2025. Using a high end 2023 desktop computer, it takes about 100 minutes to generate Fig. 5.6.

$\hat{h}(\mathcal{B}, \Phi, \mu||\theta)$, of 0.150 nats or 0.216 bits. By connecting this model to a simple data compression routine one could compress the test data (or presumably any other long sequence from the source) down to 0.216 bits per sample, which is 11% more than the 0.195 bits per sample that the Lyapunov exponent estimate suggests is possible.

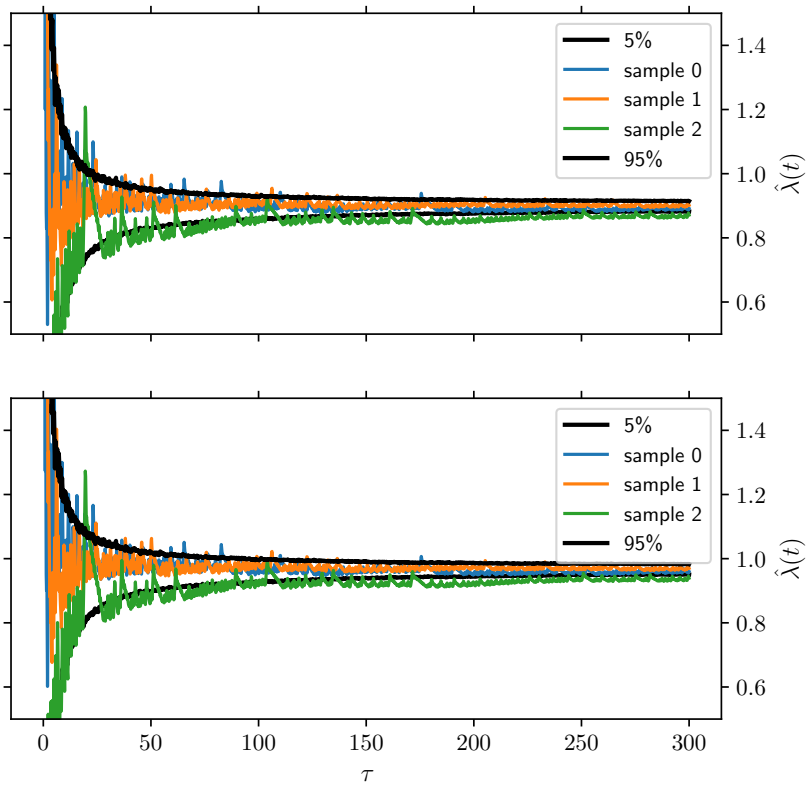

Figure 5.5: The effect of noise on Lyapunov exponent calculations for the Lorenz system. In the upper plot, the colored lines trace $\frac{1}{T}\sum_{t=0}^{T-1}\log\left(|r_{0,0}[t]|\right)$ (See Eqn. (5.45)) for three different initial conditions and the black lines trace the 5% and 95% limits on 1,000 separate runs. The lower plot is the same except that $|r_{0,0}[t]|$ is augmented by a noise term with amplitude $\frac{\sigma_\eta}{\Delta} = 0.0100$ (See Eqn. (5.51)). At $\tau = 300.0$ the Laypunov exponent estimates from the upper and lower plots are 0.899 ± 0.0100 and 0.967 ± 0.0100 respectively. The difference gives an indication of the sensitivity of the estimates to noise.

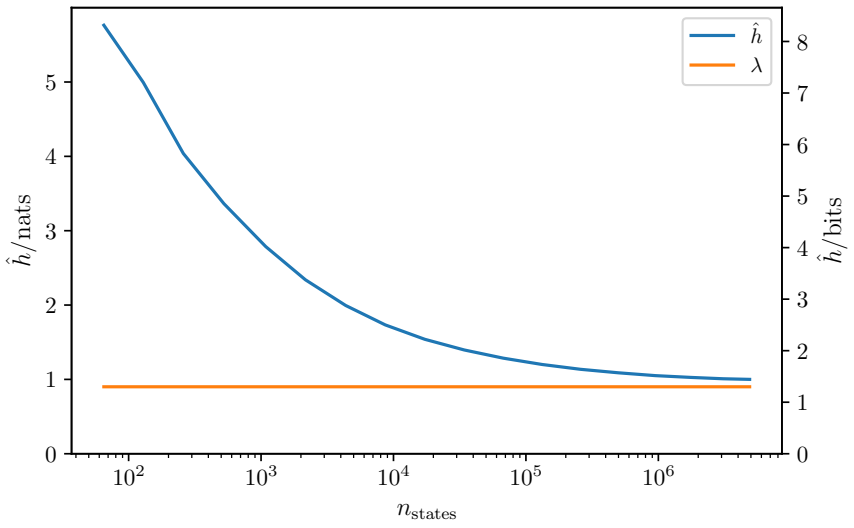

Figure 5.6: Entropy gap, $\hat{\delta}_{\mu||\theta}$ vs number of states in HMMs. The upper curve indicates estimates of cross entropy $\hat{h}(\mathcal{B}, F, \mu||\theta)$ for a sequence of HMMs vs the number of discrete states in the models. We built the models using simulated Lorenz state space trajectories as described in the text. The lower line indicates an estimate of the entropy rate, $\hat{h}(F, \mu)) = \hat{\lambda}_0$, of the true process based on Lyapunov exponents estimated by the Benettin procedure. The distance between the curves is the *entropy gap* $\hat{\delta}_{\mu||\theta}$. The gap seems to be going to zero, suggesting that an HMM with enough states would perform at least as well as any other model based any other technology. Each model was built using the same 10,000,000 sample trajectory in the original state space, and the cross entropy estimates are based on a test sequence of 10,000 observations.

Chapter 6

Obstructive Sleep Apnea

The challenge for the *Computers in Cardiology* meeting in 2000 (CINC2000) was to identify *obstructive sleep apnea* on the basis of electrocardiograms alone. Our colleague James McNames at Portland State University, with some support from us, won the prize for the best minute by minute analysis of data. We prepared our first entry using HMMs, and McNames prepared the subsequent (and winning) entries by hand using spectrograms[1] to visualize the data.

In preparing the first entry, we noticed that apnea produced many different characteristics in heart rate time series while the characteristics in non-apnea times were more similar to each other. We used a larger number of states in HMMs to model the various characteristics of apnea and a smaller number of states to model the more consistent normal times. However using the maximum a posteriori probability (MAP) estimate sequence of states from the Viterbi algorithm, ie,

$$\hat{s}[0:T] \equiv \underset{s[0:T]}{\operatorname{argmax}} P\left(s[0:T] \mid y[0:T]\right)$$

to estimate the sequence of classifications $c[\hat{0:T}]$ with

$$\hat{c}[t] \equiv c : \hat{s}[t] \in c$$

produced obvious errors. We found that during many periods of apnea the probability of being in each of the apnea states was lower than the probability of being in the normal state with the highest probability while the sum of the probabilities of apnea states was larger than the sum over normal states. Our effort to model the diversity of apnea characteristics led to decoded sequences that were normal too often.

We tried to address the issue by estimating the MAP sequence of classes rather than the MAP sequence of states, ie,

$$\hat{c}[0:T] \equiv \underset{c[0:T]}{\operatorname{argmax}} P\left(c[0:T] \mid y[0:T]\right). \tag{6.1}$$

[1] A spectrogram is display of power in Fourier spectral bands as a function of time. The x-axis is time, the y-axis is frequency, and the image intensity at point (t, f) is the power in that frequency estimated in a window centered at that time.

While the computational complexity of Viterbi algorithm which finds the MAP sequence of states is linear in T, the length of the sequence of observations, the computational complexity required to solve (6.1) is exponential[2] in T.

In this chapter, we will first describe the CINC2000 challenge and how McNames addressed it. Then we will address the challenge using HMMs.

6.1 The Challenge and the Data

The PhysioNet website[3] announced the challenge and described the data as follows:

Introduction: *Obstructive sleep apnea (intermittent cessation of breathing) is a common problem with major health implications, ranging from excessive daytime drowsiness to serious cardiac arrhythmias. Obstructive sleep apnea is associated with increased risks of high blood pressure, myocardial infarction, and stroke, and with increased mortality rates. Standard methods for detecting and quantifying sleep apnea are based on respiration monitoring, which often disturbs or interferes with sleep and is generally expensive. A number of studies during the past 15 years have hinted at the possibility of detecting sleep apnea using features of the electrocardiogram. Such approaches are minimally intrusive, inexpensive, and may be particularly well-suited for screening. The major obstacle to use of such methods is that careful quantitative comparisons of their accuracy against that of conventional techniques for apnea detection have not been published.*

We therefore offer a challenge to the biomedical research community: demonstrate the efficacy of ECG-based methods for apnea detection using a large, well-characterized, and representative set of data. The goal of the contest is to stimulate effort and advance the state of the art in this clinically significant problem, and to foster both friendly competition and wide-ranging collaborations. We will award prizes of US$500 to the most successful entrant in each of two events.

Data for development and evaluation: *Data for this contest have kindly been provided by Dr. Thomas Penzel of Philipps-University, Marburg, Germany [available on the website].*

The data to be used in the contest are divided into a learning set and a test set of equal size. Each set consists of 35 recordings, containing a single ECG signal digitized at 100 Hz with 12-bit resolution, continuously for approximately 8 hours (individual recordings vary in length from slightly less than 7 hours to nearly 10 hours). Each recording includes a set of reference annotations, one for each minute of the recording, that indicate the presence or absence of apnea during that minute. These reference annotations were made by human experts on the basis of simultaneously recorded respiration signals. Note that the reference annotations for the test set will not be made available until the conclusion

[2] In the first edition of this book, we presented an algorithm that we claimed solved (6.1) in linear complexity. In some cases that algorithm yields plausible but not necessarily correct results and in others it simply crashes.

[3] `http://www.physionet.org/challenge/2000`

of the contest. Eight of the recordings in the learning set include three respiration signals (oronasal airflow measured using nasal thermistors, and chest and abdominal respiratory effort measured using inductive plethysmography) each digitized at 20 Hz, and an oxygen saturation signal digitized at 1 Hz. These additional signals can be used as reference material to understand how the apnea annotations were made, and to study the relationships between the respiration and ECG signals. [...]

Data classes: *For the purposes of this challenge, based on these varied criteria, we have defined three classes of recordings:*

Class A (Apnea): *These meet all criteria. Recordings in class A contain at least one hour with an apnea index of 10 or more, and at least 100 minutes with apnea during the recording. The learning and test sets each contain 20 class A recordings.*

Class B (Borderline): *These meet some but not all of the criteria. Recordings in class B contain at least one hour with an apnea index of 5 or more, and between 5 and 99 minutes with apnea during the recording. The learning and test sets each contain 5 class B recordings.*

Class C (Control): *These meet none of the criteria, and may be considered normal. Recordings in class C contain fewer than 5 minutes with apnea during the recording. The learning and test sets each contain 10 class C recordings.*

Events and scoring: *Each entrant may compete in one or both of the following events:*

1. **Apnea screening:** *In this event, your task is to design software that can classify the 35 test set recordings into class A (apnea) and class C (control or normal) groups, using the ECG signal to determine if significant sleep apnea is present. [...]*

2. **Quantitative assessment of apnea:** *In this event, your software must generate a minute-by-minute annotation file for each recording, in the same format as those provided with the learning set, using the ECG signal to determine when sleep apnea occurs. Your annotations will be compared with a set of reference annotations to determine your score. Each annotation that matches a reference annotation earns one point; thus the highest possible score for this event will be approximately 16800 (480 annotations in each of 35 records). It is important to understand that scores approaching the maximum are very unlikely, since apnea assessment can be very difficult even for human experts. Nevertheless, the scores can be expected to provide a reasonable ranking of the ability of the respective algorithms to mimic the decisions made by human experts.*

6.1.1 The Data

Briefly, one can fetch the following records from PhysioNet:

a01-a20: The *a records* from individuals that display *apnea*

b01-b05: The *b records*[4] from individuals diagnosed as *borderline*

c01-c10: The *c records* from *control* or normal individuals

a01er-a04er and b01er and c01er: Identical to *a01-a04* and *b01* and *c01* except augmented with respiration and SpO_2 (percent of arterial hemoglobin saturated with oxygen) signals

summary_of_training: Expert classifications of each minute in the *a*, *b*, and *c* records

x01-x35: The test set[5]; records without classification

Using data visualization tools[6] one can see striking oscillations in the apnea time series. The patients stop breathing for tens of seconds, gasp a few breaths, and stop again. Each cycle takes about 45 seconds and can go on for most of the night. We've plotted two periods of such an oscillation from record *a03* in Fig. 6.1. The reference or *expert* classifications provided with the data indicate these are the last oscillations in the first apnea episode of the night. Over the entire night's record of 8 hours and 39 minutes, the patient had apnea for a total of 4 hours and five minutes in 11 separate episodes. In that time, more than once a minute, he was waking up enough to start breathing. Ten and a half minutes after the end of the first apnea episode, the record, as plotted in Fig. 6.2, looks normal.

In plots like Fig. 6.1, the heart rate visibly increases at the end of the gasping phase and then decreases during the phase of interrupted respiration. That heart rate oscillation is the key we used in our initial attempts to classify periods of apnea. In Fig. 6.3, we've plotted both a heart rate derived from the ECG and the SpO_2 signal. The oscillations in the signals track each other, and the expert classifies only the region of large heart rate oscillation as apnea.

6.2 Using Information from Experts to Train

We first considered the CINC2000 challenge when Andreas Rechtsteiner told us that he was going to use it as a final project for a class that Professor McNames was teaching. We suggested that Rechtsteiner try a two state HMM with autoregressive observation models, i.e., a model like those described in section 3.1.4 but with scalar observations.

To use the expert classification information in training, Rechtsteiner found that one can just modify the observation model. The technique is not limited

[4]The amplitude of the *b05* record varies dramatically over different segments of time. We found it unusable and discarded it entirely.

[5]The records *x33* and *x34* are so similar that we suspect they are simultaneous recordings from different ECG leads. We did not explicitly exploit the similarity in our analysis.

[6]One can use the script hmmds/applications/apnea/explore.py from the software we used to create this book to explore aspects of the data.

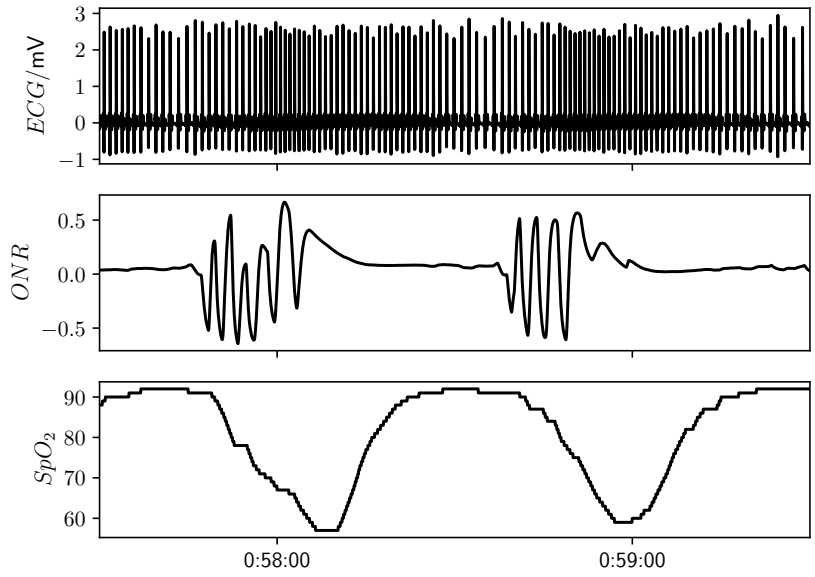

Figure 6.1: A segment of record a03. Two cycles of a large apnea induced oscillation in SpO_2 are drawn in the lower plot. The middle plot is the oronasal airflow signal, and the upper plot is the ECG. The time axis is marked in *hours:minutes:seconds*. Notice the increased heart rate just after 0:58:00 and just before 0:59:00.

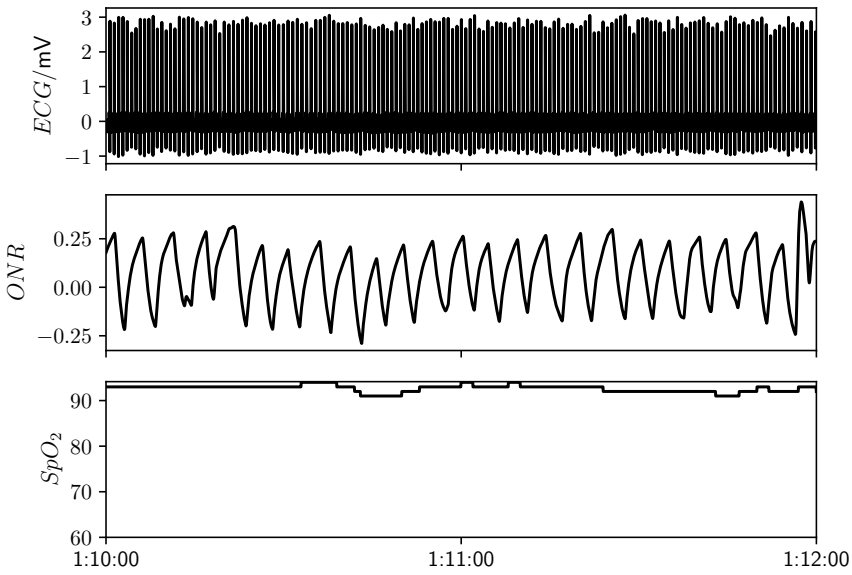

Figure 6.2: A segment of record a03 taken during a period of normal respiration. The signals are the same as in Fig. 6.1.

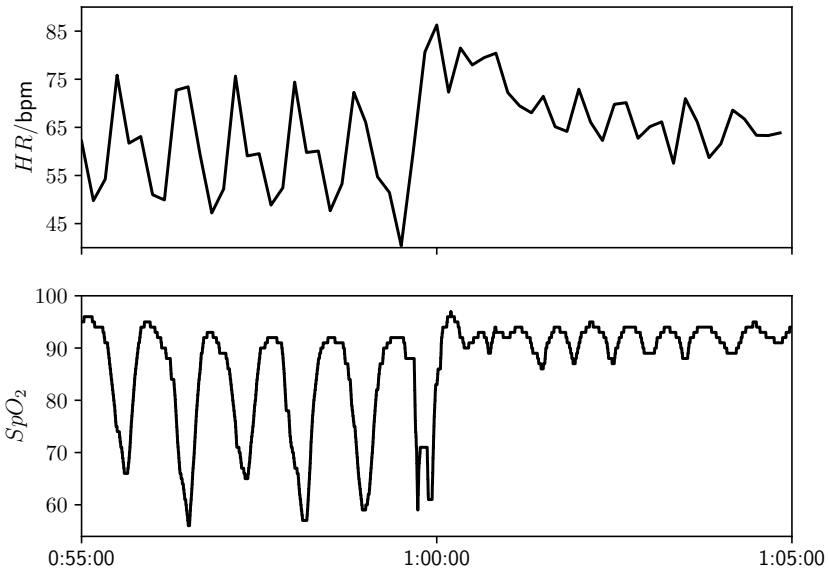

Figure 6.3: A segment of record 03 at the end of an episode of apnea with indications in both the SpO_2 signal and the heart rate (HR) signal. The expert marked the time before 1:00 as apnea and the time afterwards as normal.

to HMMs with only two states or two classes; it applies to an arbitrary number of classes and to arbitrary numbers of states associated with each class. At each time t, let $c[t]$ denote classification information from an expert about which states are possible. Specifically $c[t]$ is a vector that indicates that some states are possible and that the others are impossible. One simply replaces $P_{Y[t]|S[t],Y[0:t]}$ with $P_{Y[t],C[t]|S[t],Y[0:t]}$ wherever it occurs in the Baum-Welch algorithm. The modification forces the system into states associated with the right class during training.

While the performance of Rechtsteiner's two state model was not memorable, models with more states were promising.

6.2.1 The Excellent Eye of Professor McNames

To diagnose the errors, McNames printed spectrograms of every record. As he reported in [54], the first step in McNames analysis was implementing his own QRS[7] detection algorithm. Then he derived spectrograms from the resulting QRS analyses. Although the spectrograms discarded phase information that we hoped was important, they clearly indicated the oscillations that we had tried to capture with complicated HMMs as intense bands of power at frequencies below 2 bpm (beats per minute). In addition to those low frequency oscillations, he noticed bands of power between 10 and 20 bpm in the spectrograms (See Fig. 6.4). That higher frequency power is evidence of respiration. Using both of those features, he classified the test data by hand. First he classified each entire record for *event 1*. Since almost none of the minutes in the normal records are apnea, he classified each minute in those records as normal. His third attempt at *event 2* was the best entry at the close of the contest.

The question that motivated the contest is "Is the information in an ECG alone sufficient to classify apnea?" McNames work answered the question affirmatively. For attempts to do the classification automatically, his work suggests the following points:

1. Heart rate oscillations at about 1.3 bpm indicate apnea.

2. A clean band at about 14 bpm in the power spectrum indicates normal respiration.

6.3 Using HMMs to Address the Challenge

In the following sections we use HMMs to address the CINC2000 challenge. We begin with an HMM scheme for estimating heart rate because, like McNames, we find off-the-shelf QRS detection codes are not adequate for the variety of the CINC2000 data. From the estimated heart rate signal for each record, we derive a sequence of two dimensional observations. The first component is the low pass

[7]The ECG of a typical heart beat has five distinctive features or *waves* unimaginatively called P,Q,R,S, and T. The largest feature is the spike called R. We have labeled the features in Figure 6.7.

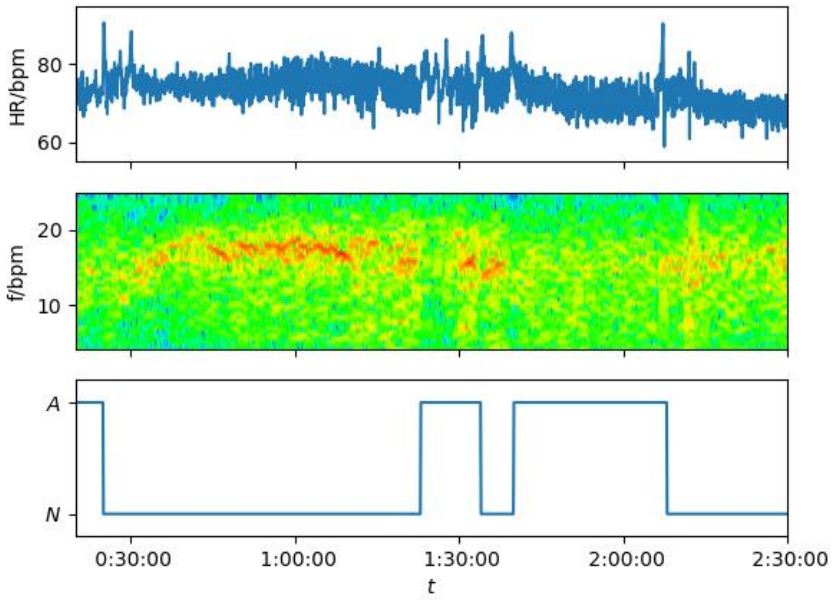

Figure 6.4: Information about respiration in high frequency bands of power spectra. This is derived from the $a11$ record between 20 minutes and 2 hours and 30 minutes after the start. The upper plot is heart rate (bandpass filtered 0.09-3.66 bpm), the middle plot is a spectrogram of the heart rate, and the lower plot is the expert classification. A single band of spectral power between about 10 and 20 bpm without much power below the band in the spectrogram indicates normal respiration.

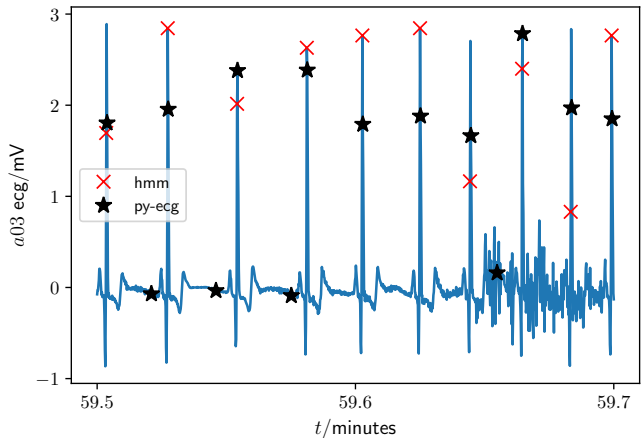

Figure 6.5: In this segment of the ECG for record a03, stars indicate estimates of the R wave locations by an off-the-shelf detector[55], and ×s indicate estimates from the algorithm described in Section 6.3.1.

filtered heart rate, and the second is roughly the intensity of the spectrogram in the range of respiration near 15 beats per minute. Next we fit an HMM with vector autoregressive observation models as described in Section 3.1.4 to the two dimensional observations derived from the training data records. And finally we apply that HMM to classify each minute of the test data.

6.3.1 Estimating Heart Rate

At first we tried using off-the-shelf code to extract heart rates from the PhysioNet ECG data. Figure 6.5 illustrates results of using one of the detectors[51] implemented by Porr et al.[48], and Figure 6.6 illustrates the challenging diversity of waveforms in the PhysioNet data. To address the diversity of waveforms, we developed an approach to estimating heart rates from ECGs based on HMMs that relies on the observation that for each record the shape and duration of the $PQRST$ pattern doesn't vary much. For different heart rates the time between the T wave and the next P wave does change, but the rest of the pattern varies very little. Figure 6.7 illustrates the invariance.

State topology

Figure 6.8 illustrates the topology of the HMMs we built to capture the invariant shapes of the PQRST sequences for each individual record. In addition to the states drawn, there is a special *outlier* state accommodates ECG-lead noise. Here are the essential characteristics of the topology:

- A loop of 52 discrete states.

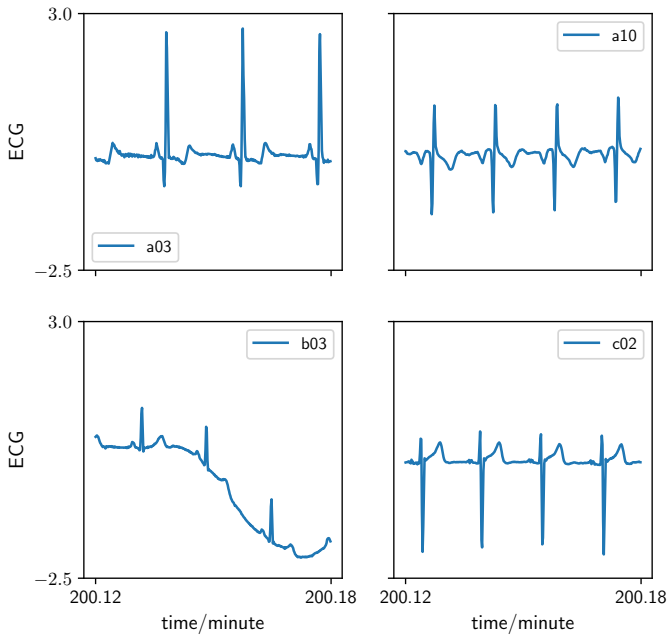

Figure 6.6: Segments of ECGs from four records. The ECGs for the different data records were so different from each other that off-the-shelf code was not adequate for estimating heart rate signals.

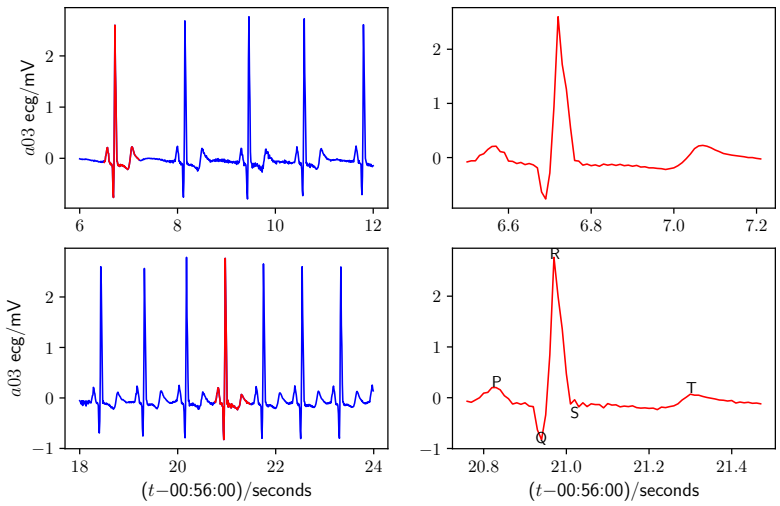

Figure 6.7: At different heart rates the shape and duration of the PQRST pattern doesn't change. Only the delay between the sequences changes. Notice that the lengths of the time intervals in the upper and lower plots are identical.

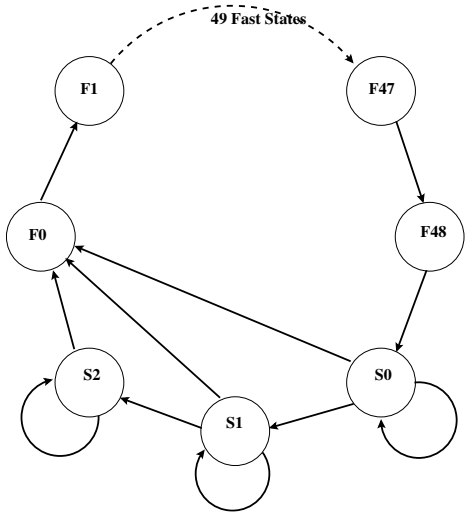

Figure 6.8: Illustration of the topology of HMMs for exploiting invariance of PQRST shapes. The chain of 49 fast states models the invariant PQRST shape. The variable duration paths through the slow states, S_0, S_1, and S_2, model the flexible time between T and P waves.

- A sequence of 49 *fast* states that don't branch. Each state n in that sequence transitions exclusively to its successor $n + 1$ at each time step. These fast states model the unvarying PQRST shape.
- Three *slow* states that model heart rate variations. Each of these three states transitions to one of the following:
  - Itself
  - Its successor in the loop
  - The first fast state

The minimum number of states visited in a loop is 50, or 500 ms since the ECG data was sampled at 100 Hz. Consequently the model is not appropriate for heart rates above 120 bpm.

Observation models and training

We used a Gaussian scalar autoregressive observation model with means for each state being a linear function of the previous three observations with a fixed offset and variances fit to each state separately. The models are one dimensional versions of those described in Section 3.1.4.

We used *scipy.signal.find_peaks* from *SciPy* to supervise training of an initial model for one of the records from CINC 2000. We derived models for most[8] of the other records from that initial model via unsupervised training.

Applying trained ECG models

As intended, the trained HMMs track PQRST shapes in the ECG data well. Figure 6.9 illustrates the performance on two records, *a01* and *c02*, in which the ECG shapes are quite different. To be clear, we repeat that we trained a separate HMM on each record.

In addition to analyzing measured ECGs, the models can generate simulated ECGs, eg, Figure 6.10.

Mapping state sequences to heart rate

We built HMMs of ECG signals to estimate heart rate signals because we believe that the timing of heart beats has the information that is relevant for detecting apnea. Decoded state sequences like those that appear in Figure 6.9 are sufficient for estimating heart rate. Considering the circular character of the topology depicted in Figure 6.8, the estimated heart rate is simply the rate at which decoded sequences go around the circle, and the phase of state sequence is not important for the estimation. It is nice however that one state in the chain of fast states corresponds closely to the peak of the R wave in the ECGs as appears in Figure 6.11.

[8] For a few of the records we used other techniques to obtain initial models. However models for all records were trained similarly.

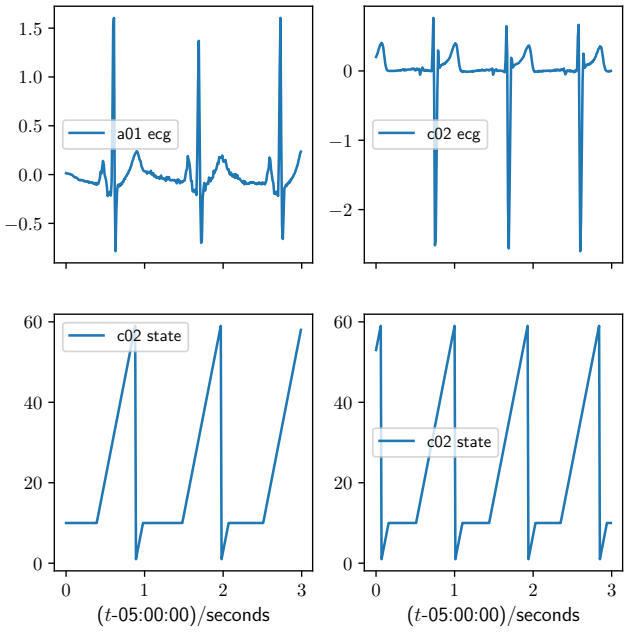

Figure 6.9: State sequences from Viterbi decoding of ECG signals for two records appear. The invariant $PQRST$ pattern maps to the lines of constant slope. The varying lengths of the horizontal segments accounts for variable time between heart beats.

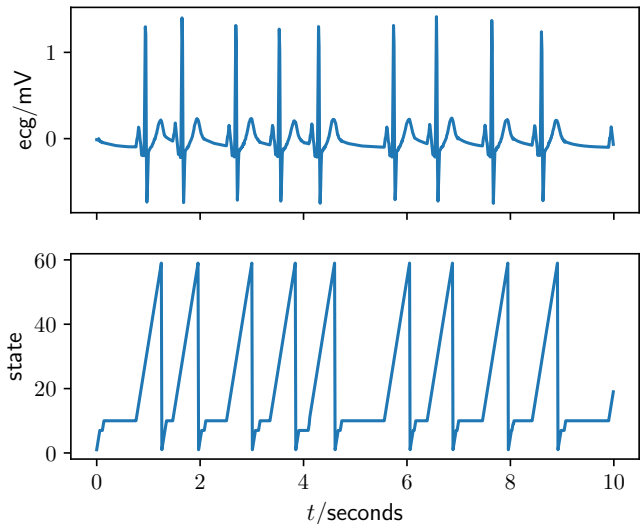

Figure 6.10: A plausible ECG appears in the upper plot, and the corresponding state sequence appears in the lower plot. We created both plots by fitting a model to the *a01* ECG data and then driving that model with a random number generator.

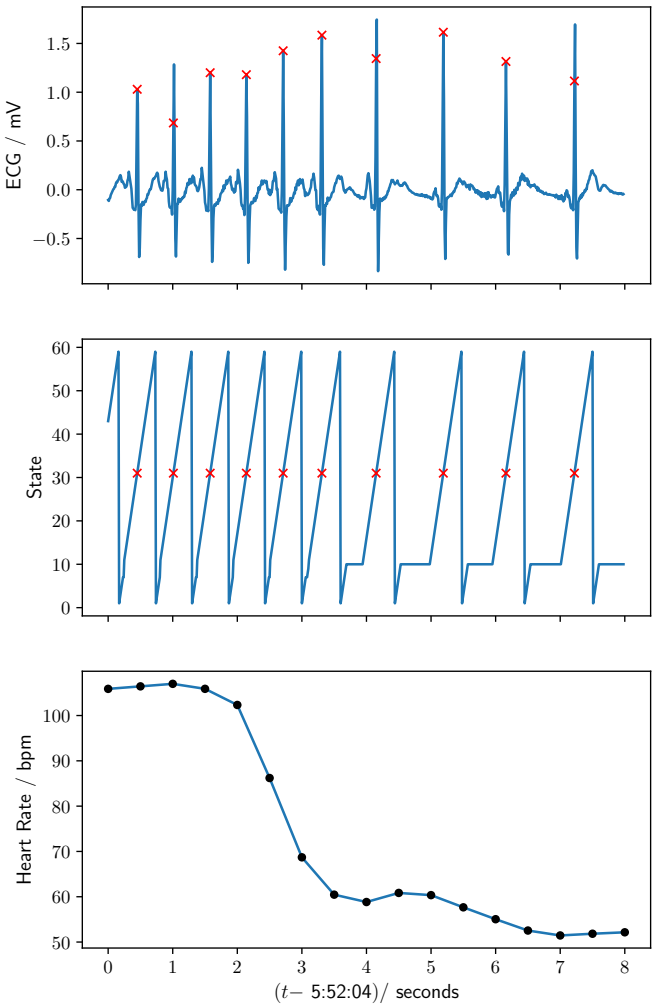

Figure 6.11: An illustration of using Viterbi decoding to derive heart rate from an ECG. An ECG segment appears in the upper plot, and the corresponding segment of the decoded state sequence appears in the middle plot. The × marks in the middle plot indicate the times when the state is 31. The same times are also indicated in the upper plot. The inverse of the difference in time between adjacent ×s provides the estimated heart rate. A smoothed version of those estimates sampled periodically at 2 Hz appears in the lower plot.

6.3.2 An Observation Model and HMMs of Heart Rate

While the previous section was about designing HMMs of ECG signals in order to estimate heart rate signals, here we will address designing HMMs of heart rate signals in order to detect apnea. In Section 6.2.1 we described two characteristics that McNames used for classifying each minute, namely:

- Oscillations of the heart rate with periods of about 45 seconds. Such oscillations appear in Fig. 6.1.
- Modulation of the heart rate by respiration at around 15 beats per minute. Such modulation is captured by the spectrogram in Fig. 6.4.

We derive these two characteristics from heart rate signals (as appears in Fig. 6.11) sampled at 2 Hz using spectral techniques that begin with a fast Fourier transform (FFT) of each entire record of about 8 hours. Figure 6.12 illustrates a transition into apnea in which both characteristics indicate the transition. The HMM we built for detecting apnea has eleven states and uses fifth order vector autoregressive observation models of two dimensional observations, with components called **low pass heart rate** and **respiration**.

Of the host of parameters, we chose the following for the observation model by hand using one dimensional studies on the training data like Figure 6.13:

Autoregressive Order 5

Model Sample Frequency 4 samples per minute

Low Pass Period 51.1 seconds

Respiration Center Frequency 11.53 bpm

Respiration Filter Width 3.2 bpm

Respiration Smoothing 0.486 bpm

Figure 6.12 illustrates the derivation of HMM observation data from the estimated raw heart rate signal using these parameters.

6.3.3 Using a Sequence of Class Probabilities

Equation (2.22), ie,

$$w(t,s) \equiv P_{S[t]|Y[0:T]}\left(s \mid y[0:T]\right),$$

provides the probability of each state at each time given the model and all of the observations. From such state weights, we construct a sequence of ratios with

$$r[t] \equiv \frac{\sum_{s\in A} w(t,s)}{\sum_{s\in N} w(t,s)}$$

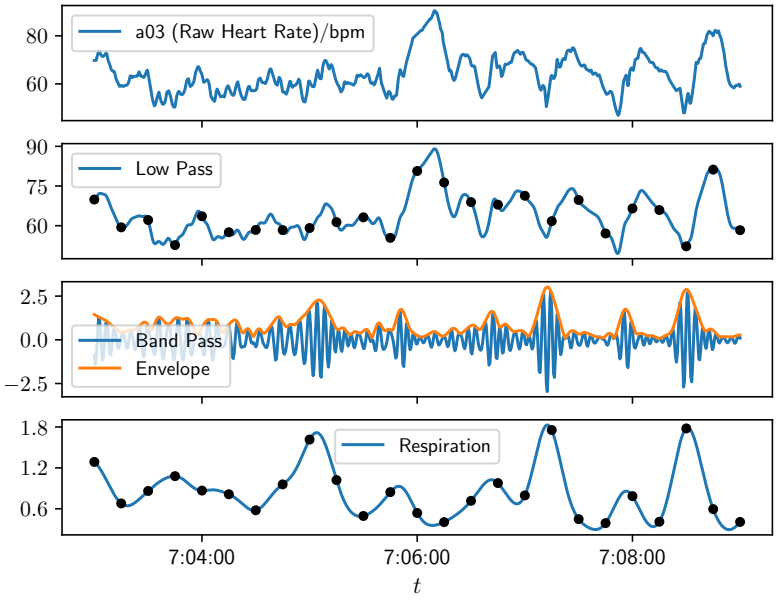

Figure 6.12: An illustration of apnea characteristic data derived from heart rate at a transition into apnea. The expert marked a transition from normal respiration to apnea in record *a03* at minute 427=7:07. A low pass Gaussian filter applied to the raw heart rate signal produces the trace labeled *Low Pass*, and the dots on that trace indicate the data passed as one component of the observation data to the HMM. A Gaussian filter that passes frequencies in a band typical for respiration yields the *Band Pass* trace. The *Respiration* trace is a low pass filtered version of the envelope of the *Band Pass* trace. Again, the dots indicate data passed to the HMM. The characteristics of apnea are the large slow oscillations in the *Low Pass* signal and the drops of the *Respiration* signal to low levels.

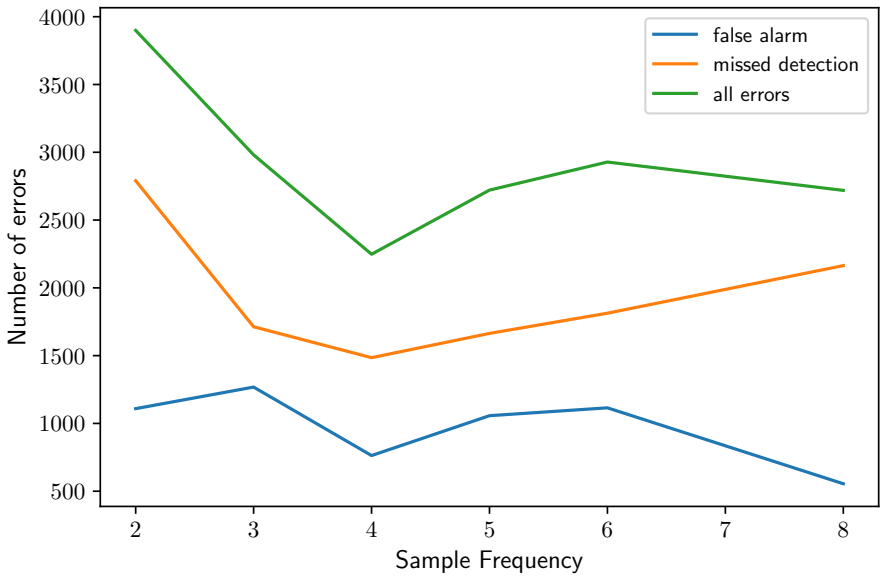

Figure 6.13: Performance on the training records (a b and c records excluding defective records) vs observations per minute. The models have eleven states and the observation models are vector autoregressive models for heart rate and respiration.

where A is the set of apnea states and N is the set of normal states. Finally we derive a sequence of classifications, $c[0:T]$, from the sequence of ratios, $r[0:T]$, by comparing to a threshold, R,

$$c[t] = \begin{cases} N & r[t] < R \\ A & r[t] >= R \end{cases}. \tag{6.2}$$

A search over parameter values like that illustrated in Figure 6.13 finds $R = 6.2$ minimizes the number of classification errors on the training data.

6.4 Results

We trained an HMM with the structure and parameters described in Section 6.3.2 on the training data (the usable *a*, *b* and *c* records) and then used it to classify each minute of the training data as either normal, N, or apnea, A. The results appear in Table 6.1. Results of applying the HMM to the test data (the x records appear in Table 6.2. We are disappointed that the score on the test data does not place among the official entries in the *CinC Challenge 2000 Top Scores* listed at https://archive.physionet.org/challenge/2000/top-scores.shtml which have error rates from 0.0738 to 0.1446. While we were able to get slightly better performance with more complex ideas, the small improvements did not seem to justify the complexity.

6.5 Classification Versus Estimation

In the early chapters of this book, we have emphasized choosing model parameters to maximize the likelihood of the data, the tweaks and fudges we've used in this chapter are concerned with improving classification performance rather than improving likelihood. While it is true that if one had access to accurate probabilistic characterizations of observations conditional on class membership the best classifier would be a likelihood ratio classifier, it is not true that without such characterizations the best approach to classification is to estimate them. This point was made forcefully by Vapnik[14] who said, "one should solve the [classification] problem directly and never solve a more general problem as an intermediate step."

Name	N_{Apnea}	N_{Normal}	Apnea→Normal		Normal→Apnea		N_{Error}	P_{Error}
a01	470	19	71	0.15	0	0.00	71	0.15
a02	420	108	83	0.20	25	0.23	108	0.20
a03	246	273	6	0.02	68	0.25	74	0.14
a04	453	39	24	0.05	8	0.21	32	0.07
a05	276	178	60	0.22	12	0.07	72	0.16
a06	206	304	165	0.80	1	0.00	166	0.33
a07	322	189	159	0.49	25	0.13	184	0.36
a08	189	312	17	0.09	106	0.34	123	0.25
a09	381	114	73	0.19	32	0.28	105	0.21
a10	100	417	58	0.58	9	0.02	67	0.13
a11	222	244	129	0.58	1	0.00	130	0.28
a12	534	43	17	0.03	27	0.63	44	0.08
a13	244	251	33	0.14	41	0.16	74	0.15
a14	383	126	121	0.32	9	0.07	130	0.26
a15	368	142	42	0.11	37	0.26	79	0.15
a16	320	162	60	0.19	12	0.07	72	0.15
a17	158	327	83	0.53	8	0.02	91	0.19
a18	438	51	147	0.34	10	0.20	157	0.32
a19	205	297	24	0.12	18	0.06	42	0.08
a20	315	195	63	0.20	23	0.12	86	0.17
b01	19	468	8	0.42	16	0.03	24	0.05
b02	93	424	11	0.12	131	0.31	142	0.27
b03	73	368	12	0.16	49	0.13	61	0.14
b04	10	419	7	0.70	6	0.01	13	0.03
c01	0	484	0	0.00	3	0.01	3	0.01
c02	1	501	1	1.00	5	0.01	6	0.01
c03	0	454	0	0.00	1	0.00	1	0.00
c04	0	482	0	0.00	0	0.00	0	0.00
c05	3	463	3	1.00	2	0.00	5	0.01
c06	1	467	1	1.00	0	0.00	1	0.00
c07	4	425	4	1.00	7	0.02	11	0.03
c08	0	513	0	0.00	45	0.09	45	0.09
c09	2	466	2	1.00	7	0.02	9	0.02
c10	1	430	1	1.00	19	0.04	20	0.05
Total	6457	10155	1485	0.23	763	0.08	2248	0.14

Table 6.1: Performance of the HMM described in 6.3.2 on the training data.

Name	N_{Apnea}	N_{Normal}	Apnea→Normal		Normal→Apnea		N_{Error}	P_{Error}
x01	375	148	157	0.42	3	0.02	160	0.31
x02	209	260	23	0.11	68	0.26	91	0.19
x03	12	453	6	0.50	47	0.10	53	0.11
x04	0	482	0	0.00	4	0.01	4	0.01
x05	316	189	6	0.02	88	0.47	94	0.19
x06	0	450	0	0.00	3	0.01	3	0.01
x07	240	269	93	0.39	26	0.10	119	0.23
x08	324	193	14	0.04	12	0.06	26	0.05
x09	167	341	4	0.02	10	0.03	14	0.03
x10	96	414	85	0.89	17	0.04	102	0.20
x11	13	444	9	0.69	15	0.03	24	0.05
x12	57	470	25	0.44	35	0.07	60	0.11
x13	292	214	124	0.42	45	0.21	169	0.33
x14	439	51	264	0.60	0	0.00	264	0.54
x15	200	298	150	0.75	6	0.02	156	0.31
x16	65	450	6	0.09	157	0.35	163	0.32
x17	1	399	1	1.00	15	0.04	16	0.04
x18	2	457	2	1.00	28	0.06	30	0.07
x19	407	80	12	0.03	46	0.57	58	0.12
x20	264	249	45	0.17	36	0.14	81	0.16
x21	120	390	10	0.08	156	0.40	166	0.33
x22	2	480	2	1.00	13	0.03	15	0.03
x23	119	408	15	0.13	30	0.07	45	0.09
x24	1	428	0	0.00	7	0.02	7	0.02
x25	291	219	14	0.05	63	0.29	77	0.15
x26	344	176	99	0.29	51	0.29	150	0.29
x27	487	11	53	0.11	6	0.55	59	0.12
x28	433	62	127	0.29	7	0.11	134	0.27
x29	0	470	0	0.00	0	0.00	0	0.00
x30	326	185	87	0.27	67	0.36	154	0.30
x31	516	41	44	0.09	1	0.02	45	0.08
x32	425	113	71	0.17	1	0.01	72	0.13
x33	3	470	3	1.00	5	0.01	8	0.02
x34	4	471	4	1.00	8	0.02	12	0.03
x35	0	483	0	0.00	5	0.01	5	0.01
Total	6550	10718	1555	0.24	1081	0.10	2636	0.15

Table 6.2: Performance of the HMM described in 6.3.2 on the test data.

Chapter 7

Particle Filters

More later.

Appendix A

Formulas for Matrices and Gaussians

Here we review some material necessary for deriving Eqns. (4.24)-(4.29) on page 66. Similar material appears in Appendix A of Kailath et al.[5].

Block Matrix Inverse

If G is an $n \times n$ invertible matrix, K is an $m \times m$ invertible matrix, and H and J are $n \times m$ and $m \times n$ respectively, then direct matrix multiplication verifies that

$$\begin{bmatrix} (G - HK^{-1}J)^{-1} & -(G - HK^{-1}J)^{-1}HK^{-1} \\ -(K - JG^{-1}H)^{-1}JG^{-1} & (K - JG^{-1}H)^{-1} \end{bmatrix} \begin{bmatrix} G & H \\ J & K \end{bmatrix} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \tag{A.1a}$$

$$\begin{bmatrix} G & H \\ J & K \end{bmatrix} \begin{bmatrix} (G - HK^{-1}J)^{-1} & -G^{-1}H(K - JG^{-1}H)^{-1} \\ -K^{-1}J(G - HK^{-1}J)^{-1} & (K - JG^{-1}H)^{-1} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{I} \end{bmatrix}, \tag{A.1b}$$

assuming that $(G - HK^{-1}J)^{-1}$ and $(K - JG^{-1}H)^{-1}$ exist. One can derive other expressions for the inverse by using the Sherman Morrison Woodbury formula (Eqn. (A.3)) to expand terms in Eqn. (A.1).

By noting

$$\begin{bmatrix} (G - HK^{-1}J)^{-1} & -(G - HK^{-1}J)^{-1}HK^{-1} \\ 0 & (K - JG^{-1}H)^{-1} \end{bmatrix} \begin{bmatrix} G & H \\ J & K \end{bmatrix} = \begin{bmatrix} \mathbf{I} & 0 \\ (K - JG^{-1}H)^{-1}J & (K - JG^{-1}H)^{-1}K \end{bmatrix}$$

and taking the determinant of both sides

$$\left|(G - HK^{-1}J)^{-1}\right| \cdot \left|(K - JG^{-1}H)^{-1}\right| \cdot \left|\begin{bmatrix} G & H \\ J & K \end{bmatrix}\right| = \left|(K - JG^{-1}H)^{-1}\right| \cdot |K|$$

one finds the following formula for determinants

$$\left|\begin{bmatrix} G & H \\ J & K \end{bmatrix}\right| = |K| \left|(G - HK^{-1}J)\right| \tag{A.2}$$

Sherman Morrison Woodbury Formula

If G and K are invertible matrices, H and J have dimensions so that $(G + HKJ)^{-1}$ makes sense and exists, and $\left(JG^{-1}H + K^{-1}\right)^{-1}$ exists, then

$$(G + HKJ)^{-1} = G^{-1} - G^{-1}H\left(JG^{-1}H + K^{-1}\right)^{-1} JG^{-1}. \tag{A.3}$$

Multiplying both sides by $(G + HKJ)$ verifies the formula. Equation (A.3) is called the Sherman Morrison Woodbury formula.

To invert $(A^{-1} + C^\top B^{-1}C)$, when A is an $n \times n$ matrix and B is an $m \times m$ matrix, if $n > m$ one can use (A.3) to write

$$\left(A^{-1} + C^\top B^{-1}C\right)^{-1} = A - AC^\top \left(CAC^\top + B\right)^{-1} CA. \tag{A.4}$$

The right hand side requires inverting an $m \times m$ matrix while the left hand side requires inverting an $n \times n$ matrix.

Marginal and Conditional Distributions of a Gaussian

Suppose that $W = \begin{bmatrix} U \\ V \end{bmatrix}$ is a Gaussian random variable with an n dimensional component U and an m dimensional component V. We write its distribution

$$W \sim \mathcal{N}(\mu_W, \Sigma_W) \text{ or equivalently } P(w) = \mathcal{N}(\mu_W, \Sigma_W)|_w$$

with

$$\mu_W = \begin{bmatrix} \mu_U \\ \mu_V \end{bmatrix} \text{ and } \Sigma_W = \begin{bmatrix} \Sigma_{UU} & \Sigma_{UV} \\ \Sigma_{VU} & \Sigma_{VV} \end{bmatrix} \equiv \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix},$$

where we have introduced $A \equiv \Sigma_{UU}$, $B \equiv \Sigma_{VV}$, and $C \equiv \Sigma_{UV}$ to shorten the notation. If we denote

$$\Sigma_W^{-1} = \begin{bmatrix} D & F \\ F^\top & E \end{bmatrix},$$

then from Eqns. (A.1) and (A.3)

$$D = (A - CB^{-1}C^\top)^{-1} \qquad = A^{-1} + A^{-1}CEC^\top A^{-1} \tag{A.5a}$$

$$E = (B - C^\top A^{-1}C)^{-1} \qquad = B^{-1} + B^{-1}C^\top DCB^{-1} \tag{A.5b}$$

$$F = -A^{-1}CE \qquad = -DCB^{-1}. \tag{A.5c}$$

In this notation, the marginal distributions are

$$P(u) = \int P(u,v)\,dv \tag{A.6a}$$
$$= \mathcal{N}(\mu_U, A)\big|_u \tag{A.6b}$$
$$P(v) = \int P(u,v)\,du \tag{A.6c}$$
$$= \mathcal{N}(\mu_V, B)\big|_v, \tag{A.6d}$$

and the conditional distributions are

$$P(u \mid v) = \frac{P(u,v)}{P(v)} \tag{A.7a}$$
$$= \mathcal{N}\left(\mu_U + CB^{-1}(v-\mu_v), D^{-1}\right)\big|_u \tag{A.7b}$$
$$P(v \mid u) = \frac{P(v,u)}{P(u)} \tag{A.7c}$$
$$= \mathcal{N}\left(\mu_V + C^\top A^{-1}(u-\mu_U), E^{-1}\right)\big|_v \tag{A.7d}$$

Notice that the covariance of the *marginal* distribution of U is given by the UU block of Σ_W, but that the inverse covariance of the *conditional* distribution of U is given by the UU block of Σ_W^{-1}.

As a check of these formulas, we examine $P(u \mid v)\,P(v)$ and find

$$\begin{aligned}
P(u \mid v)\,P(v) &= \frac{\sqrt{|D|}}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}\left(u-\mu_U-CB^{-1}(v-\mu_V)\right)^\top D\left(u-\mu_U-CB^{-1}(v-\mu_V)\right)} \\
&\quad \times \frac{1}{\sqrt{(2\pi)^m\,|B|}} e^{-\frac{1}{2}(v-\mu_V)^\top B^{-1}(v-\mu_V)} \\
&= \frac{1}{\sqrt{(2\pi)^{n+m}\,|\Sigma_W|}} \exp\Bigg(-\frac{1}{2}\Big[\\
&\quad \left(u-\mu_U-CB^{-1}(v-\mu_V)\right)^\top D\left(u-\mu_U-CB^{-1}(v-\mu_V)\right) \\
&\quad + (v-\mu_V)^\top B^{-1}(v-\mu_V)\Big]\Bigg) \\
&= \frac{1}{\sqrt{(2\pi)^{n+m}\,|\Sigma_W|}} \\
&\quad \times e^{-\frac{1}{2}\left((u-\mu_U)^\top D(u-\mu_U)+2(v-\mu_V)^\top F^\top(u-\mu_U)+(v-\mu_V)^\top E(v-\mu_V)\right)} \\
&= P(u,v)
\end{aligned}$$

all is right with the world. In the above, Eqn. (A.2) implies that $\frac{\sqrt{|D|}}{\sqrt{|B|}} = \frac{1}{\sqrt{|\Sigma_W|}}$.

Completing the Square

Some of the derivations in section 4.3 rely on a procedure called *completing the square*, which we illustrate with the following example. Suppose that the

function $f(u)$ is the product of two n dimensional Gaussians, $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$, i.e.

$$f(u) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_1|}} e^{-\frac{1}{2}(u-\mu_1)^\top \Sigma_1^{-1}(u-\mu_1)} \frac{1}{\sqrt{(2\pi)^n |\Sigma_2|}} e^{-\frac{1}{2}(u-\mu_2)^\top \Sigma_2^{-1}(u-\mu_2)} \tag{A.8}$$

$$= \frac{1}{\sqrt{(2\pi)^{2n} |\Sigma_1| |\Sigma_2|}} e^{-\frac{1}{2}\left[(u-\mu_1)^\top \Sigma_1^{-1}(u-\mu_1) + (u-\mu_2)^\top \Sigma_2^{-1}(u-\mu_2)\right]} \tag{A.9}$$

$$\equiv \frac{1}{\sqrt{(2\pi)^{2n} |\Sigma_1| |\Sigma_2|}} e^{-\frac{1}{2}\left[Q(u)\right]}. \tag{A.10}$$

By expanding the function $Q(u)$ in the exponent, we find:

$$Q(u) = u^\top \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right) u - 2u^\top \left(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2\right) + \mu_1^\top \Sigma_1^{-1}\mu_1 + \mu_2^\top \Sigma_2^{-1}\mu_2 \tag{A.11}$$

$$= u^\top q u - 2u^\top l + s \tag{A.12}$$

where the quadratic, linear, and scalar terms are

$$q = \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)$$
$$l = \left(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2\right)$$
$$s = \mu_1^\top \Sigma_1^{-1}\mu_1 + \mu_2^\top \Sigma_2^{-1}\mu_2$$

respectively.

Completing the square means finding values μ, Σ, and R for which Eqn. (A.12) takes the form

$$Q(u) = (u-\mu)^\top \Sigma^{-1}(u-\mu) + R, \tag{A.13}$$

where R is not a function of u. One can verify by substitution that the solution is

$$\Sigma^{-1} = q$$
$$\mu = \Sigma l$$
$$R = s - \mu^\top \Sigma^{-1}\mu.$$

For the product of Gaussians example (A.8),

$$\Sigma^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} \tag{A.14a}$$

$$\mu = \Sigma\left(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2\right) \tag{A.14b}$$

$$= \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1}\left(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2\right) \tag{A.14c}$$

$$R = \mu_1^\top \Sigma_1^{-1}\mu_1 + \mu_2^\top \Sigma_2^{-1}\mu_2 - \mu^\top \Sigma^{-1}\mu \tag{A.14d}$$

$$= \mu_1^\top \Sigma_1^{-1}\mu_1 + \mu_2^\top \Sigma_2^{-1}\mu_2 - \left(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2\right)^\top \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1}\left(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2\right). \tag{A.14e}$$

In words the product of two Gaussian density functions is an unnormalized Gaussian density function in which the inverse covariance is the sum of the inverse covariances of the factors and the mean is the average of the factor means weighted by the inverse covariances.

Appendix B

EM Convergence Rate

Now we calculate a linear approximation of the behavior of the EM algorithm in the neighborhood of a fixed point. The manipulations here require that the probability and likelihood functions have continuous first and second derivatives which we implicitly assume. We let $\mathcal{T}$ denote the action of one iteration of the algorithm and let θ^* denote a fixed point with

$$\theta[n+1] = \mathcal{T}\left(\theta[n]\right) \tag{B.1}$$

$$\mathcal{T}(\theta^*) = \theta^* \qquad \text{with Taylor series} \tag{B.2}$$

$$\mathcal{T}(\theta) = \theta^* + \left[\frac{\partial \mathcal{T}(\theta)}{\partial \theta}\right]_{\theta^*} (\theta - \theta^*) + \text{Remainder} \tag{B.3}$$

$$Q(\theta', \theta) \equiv \mathbb{E}_{S|y,\theta} \log\left(P(y, S \mid \theta'\right) \tag{B.4}$$

$$\mathcal{T}(\theta) = \underset{\theta'}{\operatorname{argmax}}\, Q(\theta', \theta). \tag{B.5}$$

For a given value of θ, the derivative of $Q(\theta', \theta)$ at a maximum is zero, and we write

$$\Psi(\theta', \theta) \equiv \frac{\partial Q(\theta', \theta)}{\partial \theta'} \tag{B.6}$$

$$\Psi(\mathcal{T}(\theta), \theta) = 0 \tag{B.7}$$

$$\frac{d\Psi(\mathcal{T}(\theta), \theta)}{d\theta} = 0 \tag{B.8}$$

$$\frac{d\Psi(\mathcal{T}(\theta), \theta)}{d\theta} = \left.\frac{\partial \Psi(\theta', \theta)}{\partial \theta'}\right|_{\mathcal{T}(\theta),\theta} \left.\frac{\partial \mathcal{T}(\theta)}{\partial \theta}\right|_{\theta} + \left.\frac{\partial \Psi(\theta', \theta)}{\partial \theta}\right|_{\mathcal{T}(\theta),\theta} \tag{B.9}$$

$$= \left.\frac{\partial^2 Q(\theta', \theta)}{\partial \theta'^2}\right|_{\mathcal{T}(\theta),\theta} \left.\frac{\partial \mathcal{T}(\theta)}{\partial \theta}\right|_{\theta} + \left.\frac{\partial^2 Q(\theta', \theta)}{\partial \theta \partial \theta'}\right|_{\mathcal{T}(\theta),\theta} = 0 \tag{B.10}$$

$$\left.\frac{\partial \mathcal{T}(\theta)}{\partial \theta}\right|_{\theta} = -\left[\left.\frac{\partial^2 Q(\theta', \theta)}{\partial \theta'^2}\right|_{\mathcal{T}(\theta),\theta}\right]^{-1} \left[\left.\frac{\partial^2 Q(\theta', \theta)}{\partial \theta \partial \theta'}\right|_{\mathcal{T}(\theta),\theta}\right]. \tag{B.11}$$

Manipulating the first of the two second derivatives in (B.11) we find

$$\frac{\partial^2 Q(\theta',\theta)}{\partial \theta'^2} = \frac{\partial^2}{\partial \theta'^2}\mathbb{E}_{S|y,\theta}\log\left(P(y,S\mid\theta'\right) \tag{B.12}$$

$$= \frac{\partial^2}{\partial \theta'^2}\mathbb{E}_{S|y,\theta}\left(\log\left(P(y\mid\theta'\right)+\log\left(P(S\mid y,\theta')\right)\right) \tag{B.13}$$

$$= \frac{\partial^2}{\partial \theta'^2}\log\left(P(y\mid\theta'\right)+\mathbb{E}_{S|y,\theta}\left(\frac{\partial^2}{\partial \theta'^2}\log\left(P(S\mid y,\theta')\right)\right) \tag{B.14}$$

$$= -J_y - I_{S|y}, \tag{B.15}$$

where $J_y \equiv -\frac{\partial^2}{\partial\theta^2}\log\left(P(y\mid\theta\right)$ is called the *observed information* that y provides about θ', and $I_{S|y}$ is the *Fisher information* of the unobserved data. Now manipulating the second of the two second derivatives in (B.11) we find

$$\frac{\partial^2 Q(\theta',\theta)}{\partial \theta'\partial\theta} = \frac{\partial^2}{\partial \theta'\partial\theta}\mathbb{E}_{S|y,\theta}\log\left(P(y,S\mid\theta'\right) \tag{B.16}$$

$$= \frac{\partial^2}{\partial \theta'\partial\theta}\mathbb{E}_{S|y,\theta}\left(\log\left(P(y\mid\theta'\right)+\log\left(P(S\mid y,\theta')\right)\right) \tag{B.17}$$

$$= \frac{\partial^2}{\partial \theta'\partial\theta}\log\left(P(y\mid\theta'\right)+\frac{\partial}{\partial\theta}\mathbb{E}_{S|y,\theta}\left(\frac{\partial}{\partial\theta'}\log\left(P(S\mid y,\theta')\right)\right) \tag{B.18}$$

$$= \frac{\partial}{\partial\theta}\mathbb{E}_{S|y,\theta}\left(\frac{\partial}{\partial\theta'}\log\left(P(S\mid y,\theta')\right)\right) \tag{B.19}$$

$$= \frac{\partial}{\partial\theta}\sum_s P(s\mid y,\theta)\frac{\frac{\partial P(s|y,\theta')}{\partial\theta'}}{P(s\mid y,\theta')} \tag{B.20}$$

$$= \sum_s P(s\mid y,\theta)\frac{\frac{\partial P(s|y,\theta)}{\partial\theta}}{P(s\mid y,\theta)}\frac{\frac{\partial P(s|y,\theta')}{\partial\theta'}}{P(s\mid y,\theta')} \tag{B.21}$$

At the fixed point $\theta^* = \theta' = \theta$ and[1]

$$\left.\frac{\partial^2 Q(\theta',\theta)}{\partial\theta'\partial\theta}\right|_{\theta^*,\theta^*} = \mathbb{E}_{S|y,\theta^*}\left(\frac{\partial}{\partial\theta'}\log\left(P(S\mid y,\theta')\right)\right)^2 \tag{B.22}$$

$$\equiv I_{S|y}. \tag{B.23}$$

Combining (B.23) and (B.15) with (B.11) we write

$$\left.\frac{\partial\mathcal{T}(\theta)}{\partial\theta}\right|_{\theta^*} = \left[J_y + I_{S|y}\right]^{-1} I_{S|y}. \tag{B.24}$$

Equation (B.24) matches our intuition. If the observed information, J_y, is much larger than the unobserved information, $I_{S|y}$, the derivative is small and the convergence is fast. Alternatively, if the unobserved information dominates, then the derivative is close to one and the convergence is slow.

[1]This is an alternative calculation of Fisher information.

Linear Stability of EM

Here we use an idea inspired by Sylvester's law of inertia[2] to show that if J_y is positive definite then $\mathcal{T}$ is linearly stable at θ^*.

We need the following lemma: If A is positive definite and symmetric and B is positive definite and symmetric then the eigenvalues of their product $C = AB$ are positive. Because A is positive definite and symmetric, there is an X with

$$\begin{aligned} A &= XX^\top && \text{and we can define} \\ \Gamma &\equiv X^{-1}CX = X^{-1}XX^\top BX \\ \Gamma &= X^\top BX \end{aligned}$$

By assumption B is positive definite so $y^\top By > 0 \ \forall y \neq 0$. Now $\forall z \neq 0$, $z^\top X^\top BXz > 0$ because Xz is a y. So Γ is positive definite. The symmetry of B implies that Γ is also symmetric. Thus Γ is positive definite and symmetric, and all of its eigenvalues are positive. Because they are related by a similarity transformation, Γ and C have the same eigenvalues, and we know that all of the eigenvalues of $AB = C$ are positive.

At a fixed point θ^*, $\mathcal{T}$ is linearly stable if and only if $|\lambda| < 1$ for all eigenvalues λ of its derivative D. From (B.24) we find

$$\begin{aligned} D &= \left[I_{S|y}^{-1} J_y + 1 \right]^{-1} \\ D^{-1} &= I_{S|y}^{-1} J_y + 1 \\ D^{-1} - 1 &= I_{S|y}^{-1} J_y \end{aligned} \tag{B.25}$$

Since the right hand side of (B.25) satisfies the premises of the lemma, each its eigenvalues λ_R is positive. Now for each eigenvalue of the right hand side there is an eigenvalue of D with

$$\begin{aligned} \frac{1}{\lambda_D} - 1 &= \lambda_R \\ \lambda_D &= \frac{1}{1+\lambda_R} && \text{and since } \lambda_R > 0 \\ 0 &< \lambda_D < 1. \end{aligned} \tag{B.26}$$

Thus if the eigenvalues of J_y are positive (because it is symmetric, this is equivalent to it being positive definite) then $\mathcal{T}$ is linearly stable. A similar argument shows that if $J_y \equiv -\frac{\partial^2}{\partial \theta^2} \log\left(P(y \mid \theta\right)$ has negative eigenvalues then $\mathcal{T}$ is linearly unstable.

In summary: Qualitatively $\mathcal{T}$ acts like a gradient flow on $L \equiv \log\left(P(S \mid y, \theta\right)$; convergence to fixed point of $\mathcal{T}$ that is a local maximum of the likelihood is generic and convergence to a saddle point of the likelihood is not generic.

[2]Sylvester's law of inertia is: If B is a symmetric matrix, then for any invertible matrix A, the number of positive, negative and zero eigenvalues (called the inertia of the matrix) of $C = ABA^\top$ is constant.

Appendix C

Notes on Software

We found that writing the text for this book took less time than writing the supporting software. All of the software we've used (both the code we've written and the software that that code depends on) for this book is free software. After fetching our code, typing "make book" in the top level directory of the hmmds project will create a copy of the book in a file called *main.pdf* after some hours of computation. **ToDo:** Where can one get the software? On what systems does the code run?

Data

We used the following sets of data for examples:

Tang's laser data Carl Otto Weiss mailed us a CD full of data from various experiments that he and Tang did in the 1980s and 1990s. Although we analyzed many of the files, we finally used only a file called *LP5.DAT* in the book (see Section 1.1). The file *LP5.DAT* is included in *hmmdsbook*.

H. L. Mencken's *A Book of Prefaces* We used Mencken's book for the parts of speech example in Section 1.3.2. Although the code fetches the book from www.gutenberg.org as of April, 2007, we planned to include the parsed text in *hmmdsbook*.

CINC2000 ECG data We used Dr. Thomas Penzel's ECG measurements throughout Chapter 6. Although the code fetches the the data from www.physionet.org/physiobank/database/apnea-ecg as of April, 2007, we planned to include much smaller files that contain estimates of the timing of heart beats in *hmmdsbook*.

Clarity and Efficiency

Before the SciPy or NumPy packages existed, we wrote early versions of the code for this book in C to make it run fast. Since we wrote those early versions, SciPy and NumPy have made most of that old C code obsolete. Now we have Python code for all of the algorithms described in the book.

We also provide Cython code for a few of the algorithms. While the Cython code is faster, it is harder to read and debug. The interfaces to call Cython code match the interfaces to the Python code. We recommend developing with the Python code and after that if you need the speed, try the Cython versions.

Here is the heart of our simple Python implementation of the forward algorithm described in about 4 pages in Section 2.1. It looks pretty simple here.

```
# last is a conditional distribution of state probabilities.
# What it is conditioned on changes as the calculations
# progress.
last = numpy.copy(self.p_state_initial.reshape(-1))
for t, likelihood_t in enumerate(self.state_likelihood):
    last *= likelihood_t  # Element-wise multiply
    self.gamma_inv[t] = 1 / last.sum()
    last *= self.gamma_inv[t]
    self.alpha[t, :] = last
    last[:] = numpy.dot(last, self.p_state2state)
```

And here is the heart of the corresponding implementation of the backward algorithm. It is even simpler.

```
# last and beta are analogous to last and alpha in forward(),
# but the precise interpretations are more complicated.
last = numpy.ones(self.n_states)
for t in range(len(self.state_likelihood) - 1, -1, -1):
    self.beta[t, :] = last
    last *= self.state_likelihood[t] * self.gamma_inv[t]
    last[:] = numpy.dot(self.p_state2state, last)
```

Notation

Throughout the book we write about random variables and stochastic processes. We have tried to select notation that is as simple as possible without being ambiguous. To illustrate the challenge suppose that we have been talking about gambling and the weather and that we say "the probability of 5 is 0.16". Among the many things that we could mean are the following:

$$\textbf{Prob}(\text{top face of die} = 5) = 0.16 \tag{C.1}$$

$$\lim_{\epsilon \to 0} \frac{\textbf{Prob}(5 < \text{total rainfall today in millimeters} < 5 + \epsilon)}{\epsilon} = 0.16 \tag{C.2}$$

As a less cumbersome notation we prefer $P_D(5) = 0.16$ or $P_R(5) = 0.16$, where P denotes either a probability mass function or a probability density function. We only use a subscript when it is necessary to specify which function we mean.

In general we use the following conventions:

X	Upper case indicates a random variable. Although the notation does not suggest it, the notion of a random variable includes a set of possible values or outcomes and a probability distribution for those values.
x	Lower case indicates an *outcome* or value of a random variable.
$\mathcal{X}$	Occasionally we use a calligraphic font to indicate the *alphabet* or set of all possible values of a random variable X.
$X[0:T]$	A stochastic process indexed by the sequence $[0, 1, 2, \ldots, T-1]$, i.e., $X[0], X[1], \ldots, X[T-1]$.
$x[0:T]$	A particular possible outcome of the stochastic process $X[0:T]$.
$X[t]$	The random variable that results from picking a single component of a stochastic process.
$P(x)$	The probability (or density) that a random variable will have the particular value x. We do not put a subscript on P when the context permits us to drop it without ambiguity.

$P_{X[t]}(x)$ The probability that the value of $X[t]$ is x.

$P_{X[t+1]|X[t]}(x_a \mid x_b)$ The conditional probability that the value of $X[t+1]$ is x_a given that the value of $X[t]$ is x_b.

$\mu(\beta)$ The probability that an outcome is in the *set* β. We occasionally use this notation from measure theory in Chapter 5.

$P(x \mid \theta)$ Rather than a subscript, we occasionally use a conditioning variable to specify one of many possible probability distributions.

Symbols

The following symbols usually have the meanings described below:

$\alpha(t, s)$ The conditional probability that at time t the system is in state s given the observations up to the present,

$$\alpha(t, s) \equiv P_{S(t)|Y[0:t]}(s \mid y[0:t]).$$

Also called the forward updated distribution in the Kalman filter literature. (page 22)

$\beta(t, s)$ An intermediate quantity calculated in the backwards algorithm which is used somewhat like α in the forward algorithm. One may use either of the following two equations to define β

$$\begin{aligned}\beta(t, s) &= \frac{P_{Y[t+1:T]|S[t]}(y[t+1:T] \mid s)}{P(y[t+1:T] \mid y[0:t])} \\ &= \frac{P_{S[t]|Y[0:T]}(s \mid y[0:T])}{\alpha(t, s)} = \frac{P_{S[t]|Y[0:T]}(s \mid y[0:T])}{P_{S[t]|Y[0:t]}(s \mid y[0:t])}.\end{aligned}$$

The interpretation of β is less intuitive than the interpretation of α. It is also called the backward forecast distribution in the Kalman filter literature. (page 30)

$\gamma(t)$ The probability of the present observation given the history. (page 22)

$\boldsymbol{\theta}$ The entire collection of parameters that defines a model. (page 21)

$\nu(t, s)$ Used in discussing the Viterbi algorithm to denote the *utility* of the best sequence ending in state s.

$$\nu(t, s) \equiv \log\left(P(y[0:t+1], \tilde{s}[0:t+1](s))\right)$$

(page 26)

bpm — Beats per minute. (page 110)

$[DF(x)]\,\delta$ — The *derivative* of the function F at x applied to the vector δ. This notation emphasizes that $DF(x)$ is a linear map. (page 76)

ECG — Electrocardiogram (page 106)

$\mathbb{E}_{q(X)}(F(X))$ — The expected value of the function F over random variable X with distribution q. (page 30)

$H(U)$ — The entropy of a discrete random variable U. (page 82)

$\tilde{H}(U)$ — Differential entropy of the continuous random variable U. (page 82)

h_{KS} — Kolmogorov Sinai entropy. (page 85)

$h(X)$ — The *entropy rate* of a stochastic process X. (page 84)

$\mathbf{I}$ — The *identity* operator; a diagonal matrix of ones. (page 2)

$\mathcal{N}(\mu, \Sigma)$ — A *normal* or Gaussian distribution function; μ is an n dimensional vector and Σ is an $n \times n$ matrix. Writing $X \sim \mathcal{N}(\mu, \Sigma)$ means X is distributed normally with mean μ and covariance Σ and the probability density at any particular vector x is $\mathcal{N}(\mu, \Sigma)|_x$. (page 128)

SpO_2 — Percent of arterial hemoglobin saturated with oxygen. (page 106)

$\tilde{w}(t, \tilde{s}, s)$ — In the reestimation phase of the Baum-Welch algorithm, the *weight* assigned to the transition from state s at time t to state $\tilde{s}$ at time $t+1$,

$$\tilde{w}(t, \tilde{s}, s) \equiv P_{S[t+1],S[t]|Y[0:T]}\left(\tilde{s}, s \mid y[0:T]\right).$$

(page 32)

$w(t, s)$ — In the reestimation phase of the Baum-Welch algorithm, the *weight* assigned to state s at time t,

$$w(t, s) \equiv P_{S[t]|Y[0:T]}\left(s \mid y[0:T]\right).$$

(page 32)

Bibliography

Books and Collections

[1] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley, NY, 1991.

[2] J. Ferguson, ed., *Symposium on the application of hidden Markov models to text and speech*, Institute for Defense Analyses, Communications Research Division, Princeton, NJ, October 1980.

[3] J. Gibbs, *Elementary Principles in Statistical Mechanics Developed with Especial Reference to the Rational Foundation of Thermodynamics*, Yale University Press, 1902. Republished by Dover in 1960.

[4] G. H. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 3 ed., 1996.

[5] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*, Prentice Hall, 2000.

[6] A. Katok and B. Hasselblatt, *Introduction to the Modern Theory of Dynamical Systems*, Cambridge University Press, New York, NY, 1995.

[7] R. Mane, *Ergodic theory and differentiable dynamics*, Springer-Verlag, Berlin, 1987. Translated from the Portuguese by Silvio Levy.

[8] P. S. Maybeck, *Stochastic models, estimation, and control*, Academic Press, 1979.

[9] ——, *Stochastic models, estimation, and control, Vol 2*, Academic Press, 1982.

[10] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, Wiley, 1996.

[11] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, San Mateo (Calif.), second ed., 1991.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: the art of scientific computing*, Cambridge University Press, Cambridge, 2nd ed., 1992.

[13] J. Schafer, *Analysis of Incomplete Multivariate Data*, no. 72 in Monographs on Statistics and Applied Probability, Chapman Hall/CRC, 1997.

[14] V. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, 1998.

[15] Watanabe and Yamaguchi, *The EM algorithm and related statistical models*, Marcel Dekker, 2004.

Review Articles

[16] J. Ferguson, *Hidden Markov analysis: An introduction*, in Proc. of the Symposium on the Applications of Hidden Markov Models to Text and Speech, Princeton, 1980, IDA-CRD, pp. 8–15.

[17] A. Poritz, *Hidden Markov models: A guided tour*, in Proc. IEEE Intl. Conf. on Acoust. Speech and Signal Proc., Piscataway, NJ, 1988, IEEE.

[18] L. Rabiner, *A tutorial on hidden markov models and selected applications in speech recognition*, Proceedings of the IEEE, 77 (1989), pp. 257–286.

[19] T. Sauer, J. Yorke, and M. Casdagli, *Embedology*, J. Stat. Phys., 65 (1991), pp. 579–616.

[20] P. T., M. J., de Chazal P., R. B., M. A., and M. G., *Systematic comparison of different algorithms for apnea detection based on electrocardiogram recordings*, Medical & Biological Engineering & Computing, 40 (2002), pp. 402–407.

[21] I. Witten, R. Neal, and J. Cleary, *Arithmetic coding for data compression*, Comm. ACM, 30 (1987), pp. 520–540.

[22] L. Young, *Ergodic theory of differentiable dynamical systems*, in Real and Complex Dynamical Systems, B. Branner and P. Hjorth, eds., Kluwer, 1995.
`http://citeseer.ist.psu.edu/young95ergodic.html`.

Dissertations and Theses

[23] M. J. Beal, *Variational Algorithms for Approximate Bayesian Inference*, PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003. Chapter 3, entitled "Variational Bayesian Hidden Markov Models", review HMMs in general and describes a Bayesian approach to estimation. The full text of the thesis is available at
`http://www.cse.buffalo.edu/faculty/mbeal/papers.html`.

[24] K. R. Vixie, *Signals and Hidden Information*, PhD thesis, Portland State University, 2002. available as Los Alamos Lab report LA-13881-T.

Research Articles

[25] L. Baum and J. Eagon, *An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology*, Bulletin of the American Mathematical Society, 73 (1967), pp. 360–363.

[26] L. Baum, T. Petrie, G. Soules, and N. Weiss, *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains*, The Annals of Mathematical Statistics, 41 (1970), pp. 164–171.

[27] G. Benettin, L. Galgani, A. Giorgilli, and J.-M. Sterlcyn, *Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them*, Meccanica, 15 (1980), p. 9.

[28] A. Dempster, N. Laird, and D. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, J.R. Stat. Soc. B, 39 (1977), pp. 1–38.

[29] A. M. Fraser and H. L. Swinney, *Independent coordinates for strange attractors from mutual information*, Phys. Rev. A, 33 (1986), pp. 1134–1140.

[30] J. Gauvain and C. Lee, *Maximum* a posteriori *estimation for multivariate Gaussian mixture observations of Markov chains.*, IEEE Trans. Speech and Audio Processing, 2 (1994), pp. 291–298.

[31] N. Gershenfeld, B. Schoner, and E. Metois, *Cluster-weighted modeling for time series analysis*, Nature,, 379 (1999), pp. 329–332.

[32] N. Gordon, D. Salmond, and A. Smith, *Novel approach to nonlinear/non gaussian bayesian state estimation*, IEE Proceedings F, (1993), pp. 107–113.

[33] H. Haken, *Analogy between higher instabilities in fluids and laser*, Phys. Lett A, (1975), pp. 77–78.

[34] S. Julier and J. Uhlmann, *A new extension of the kalman filter to nonlinear systems*, in The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls, Orlando Florida, vol. Multi-Sensor Fusion, Tracking and Resource Management II of Proc of AeroSense, 1997.

[35] G. Kitagawa, *Monte-carlo filter and smoother for non-gaussian nonlinear state space models*, Journal on Computational and Graphical Statistics, 5 (1996), pp. 1–25.

[36] S. Kullback and R. Leibler, *On information and sufficiency*, Ann. Math. Stat., 22 (1951), pp. 79–86.

[37] E. Lorenz, *Deterministic nonperiodic flow*, J. Atmos. Sci., 20 (1963).

[38] D. Ormoneit and V. Tresp, *Improved Gaussian mixture density estimates using Bayesian penalty term and network averaging*, in Advances in Neural Information Processing Systems, 1995.

[39] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. Shaw, *Geometry from a time series*, Phys. Rev. Lett., 45 (1980), pp. 712–716.

[40] Y. B. Pesin, *Characteristic lyapunov exponents and smooth ergodic theory*, Russ. Math. Surv., 32 (1977), pp. 55–112.

[41] R. RA, F. M, and E. AL, *Calcium regulation of single ryanodine receptor channel gating analyzed using hmm/mcmc statistical methods*, J Gen Physiol, 123 (2004), pp. 533–53.

[42] R. Redner and H. Walker, *Mixture densities, maximum likelihood and the EM algorithm*, SIAM Review, 26 (1984), pp. 195–202.

[43] D. Ruelle, *An inequality for the entropy of differentiable maps*, Bol. Soc. Brasil. Mat., 9 (1978), pp. 83–87.

[44] F. Takens, *Detecting strange attractors in turbulence*, in Dynamical Systems and Turbulence, Warwick, 1980, Lecture notes in mathematics vol. 898, D. A. Rand and L. Young, eds., Berlin, 1981, Springer, pp. 366–381.

[45] D. Tang, C. Weiss, E. Roldan, and G. de Valcarcel, *Deviation from lorenz-type dynamics of an nh_3 ring laser*, Optics Communications, 89 (1992), pp. 47–53.

[46] W. Tucker, *The lorenz attractor exists*, C. R. Acad. Sci. Ser. I Math., 328 (1999), pp. 1197–1202.
`http://citeseer.ist.psu.edu/tucker99lorenz.html`.

[47] C. J. Wu, *On the convergence properties of the EM algorithm*, The Annals of Statistics, 11 (1983), pp. 95–103.

Web Sites

[48] B. Porr and L. Howell, *Ecg detectors*. Web Site.
`https://github.com/berndporr/py-ecg-detectors`.

[49] C. Xie, L. McCullum, A. Johnson, T. Pollard, B. Gow, and B. Moody, *Waveform database software package (wfdb) for python*. Web Site, 2023.
https://doi.org/10.13026/9njx-6322.

Uncategorized)

[50] R. G. Brown and P. Y. C. Hwang, *Introduction to random signals and applied Kalman filtering : with MATLAB exercises and solutions*, Wiley, New York, third edition. ed., 1997.

[51] M. Elgendi, M. Jonkman, and F. DeBoer, *Frequency bands effects on qrs detection*, in Proceedings of the Third International Conference on Bio-inspired Systems and Signal Processing (BIOSTEC 2010) - BIOSIGNALS, INSTICC, SciTePress, 2010, pp. 428–431.

[52] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, *PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals*, Circulation, 101 (2000 (June 13)), pp. e215–e220. Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[53] O. L. R. Jacobs, *Introduction to control theory / O.L.R. Jacobs.*, Oxford University Press, Oxford ;, 2nd ed. ed., 1993.

[54] J. Mcnames and A. M. Fraser, *Obstructive sleep apnea classification based on spectrogram patterns in the electrocardiogram*, Computers in Cardiology 2000. Vol.27 (Cat. 00CH37163), (2000), pp. 749–752.

[55] B. Porr, L. Howell, I. Stournaras, and Y. Nir, *Ecg detectors*, Jan. 2024.

[56] D. Y. Tang and C. O. Weiss, *Uniqueness of the chaotic attractor of a single-mode laser*, Phys. Rev. A, 49 (1994), pp. 1296–1300.

Index

Page numbers set in **bold** type indicate that the entry refers to a definition.