Software for Hidden Markov Models and Dynamical Systems

Andrew M. Fraser

May 15, 2013

Preface

{chap:preface}

This document describes *hmmds3*, a revision of the software I used to produce *Hidden Markov Models and Dynamical Systems*, hereinafter *the book*. When I started working on the book, I decided to structure it as a software project using version control to help with collaboration and keep track of changes. I used gnu *make* to control building the figures and formatting the text. I hoped that using the structure of a software project would produce a single package that achieved the following goals:

- Keep a record of all techniques
- Allow readers to read, run, modify, and redistribute the code
- Allow readers to verify results
- Make formatting techniques available to other authors

Building

{sec:building}

To build this document, from a command line, go to the root directory (just above *TeX*) and enter the following

```
scons TeX/software.pdf
```

Some of the derived files (those that take days to build and those for which the present code does not work) are included in the *derived_data* directory of distribution. If you want to rebuild those files, simply remove them and rebuild using the above procedure.

showlabels

{sec:showlabels}

The text {`sec:showlabels`} that appears in the margin is produced by the package *showlabels*. I find it helpful as I write LaTeXdocuments to be able to see the symbolic names of labeled items. You can eliminate the labels by removing the line `\usepackage{showlabels}` from the file *software.tex* and rebuilding this document, *software.pdf*.

Changes

{sec:changes}

A preliminary version of Release 1.0 of the hmmds3 produced the figures in this draft document. When I release the software, each figure in the book will map to a figure in this document. Below, I list the major differences between hmmds3 and the software that I actually used to produce the book:

Error described and addressed: The algorithm for decoding *class* sequences described in Chapter 6 of the book does not work in general. In designing the flawed algorithm, I imitated the Viterbi algorithm which finds the best *state* sequence with complexity that is *linear* in the number of time steps. In the book, I claimed that my algorithm was also linear in the number time steps and that it finds the best *classification* sequence. While the complexity is linear, the flawed algorithm may fail to find the best sequence, and in fact it may fail to find a *possible* sequence. For details, see Section 6.3 of this document.

Test Suite: I have written tests using the variant of the *Nose testing system* that numpy uses. Those tests are part of the new distribution. That system revealed to me the error of the previous item.

Favor Pretty Code Over Pretty Pictures: Karl Hegbloom and I wrote complex code to make the book pretty. In this version, I sacrifice appearance of the plots if that lets me write simpler code. I do not recommend the formatting techniques of this version to other authors because the results are not very pretty. Neither do I recommend the techniques of the first version because *they* were too complex.

Python3: Most of the new code runs under python3 exclusively. The major exceptions are plotting scripts that use matplotlib and run under python2 because matplotlib is not compatible with python3 yet.

Numpy, Scipy, Cython, . . . The new and improved tools that have been created since I started the first version combined with things I've learned about writing code has produced much clearer code.

Sphinx: I use sphinx to format documentation embedded in the source.

Matplotlib and Xfig: I make all of the figures with matplotlib and xfig rather than the mix of tools including gnuplot that Karl and I used for the book. My code that uses matplotlib is ugly. While I take some credit for that ugliness, I don't recommend matplotlib with any enthusiasm over gnuplot.

Scons: I've switched from gnu-make to scons for building. I don't recommend scons over gnu make with any enthusiasm either.

Differences

The data for most of the figures and tables in this document are not identical to data for the figures and tables in the book. My code is different and the libraries on which my code builds are different. Many details of the results depend on features like random number generators. I have focused on making the new code easier to read rather than getting it to reproduce the results of the old code exactly. While I've found the old code embarrassingly difficult to read, I am pleased to have found that it mostly runs and gives correct results. As of 2013-02-12 the only substantial error that I've found in the old is the bogus algorithm described in Chapter 6 of the book for decoding class sequences. If I find other major flaws, I will note them in revisions to this document.

Copyright and License

Structure of the Document

Chapters 1 through 6 mirror chapters in the book. The figures in those chapters map bijectively to the figures in the corresponding chapters of the book and the numbering is the same. While the figures in the book illustrate concepts in the text, in this document the text explains the software that makes the figures. I will place other material in this preface and in appendices. The first appendix addresses the erroneous analysis of class decoding that appears in Chapter 6 of the book.

To Do

fig:LikeLor Perhaps try for larger n_{states}

[1]The "Los Alamos Computer Code" for this work is LA-CC-13-008.

Fix fig:mm:

Class-Decode: Study the performance of ad hoc algorithms for decoding class. Write an appendix describing the study. Apply such an algorithm to the apnea problem of Chapter 6.

Clean up EKF and put it in code/hmm Get scons to build all files in derived_data/laser and remove them from the repository

Write new Hview code Data for Figs. 5.1 and 5.2

Chapter 1

Introduction

{chap:introduction}

1.1 Laser Example

The red trace in Fig. 1.1 plots the first 250 points in Tang's data file *LP5.DAT*. The blue trace is the result of an optimization that searches for parameters of the model described in the book that minimizes the sum of the squared differences between the simulation and the data. The program *Laser_data.py* calls the scipy.optimize.fmin_powell routine to do the optimization and makes the file *data/LaserLP5*. The optimization takes about 6 minutes. The script *Laser_plots.py* makes *figs/LaserLP5.pdf* from *data/LaserLP5*.

After finishing the optimization for Fig. 1.2, the data for Figs.1.3, 1.4, and 3.1 are made in less than a minute.

Figure 1.4 is not as nice as the figure in the book. I hope to rework the optimization code yet again to find a stable period five orbit that matches the data.

1.3 Discrete HMMs

{sec:intro_hmm}

Karl Hegbloom and I drew Fig. 1.5 using the xfig utility. Karl set up a method that let us use the same drawing for Fig. 1.6 too. In his words:

> Both *Markov_mm.pdf_t* and *Markov_dhmm.pdf_t* are generated from the same .fig file. The output arcs and their labels are at depth 40 and the rest is at depth 50. When *mm.pdf_t* is created, the space taken up by the output arcs is blank, so we need to trim it off using the '-K' switch to 'fig2dev'.

1.3.1 Example: Quantized Lorenz Time Series

{sec:QuantizedLorenz}

I used *scipy.odeint* or the fourth order Runge Kutta formula implemented in Cython, to integrate the Lorenz system and make the data for Figs. 1.7-1.9. I

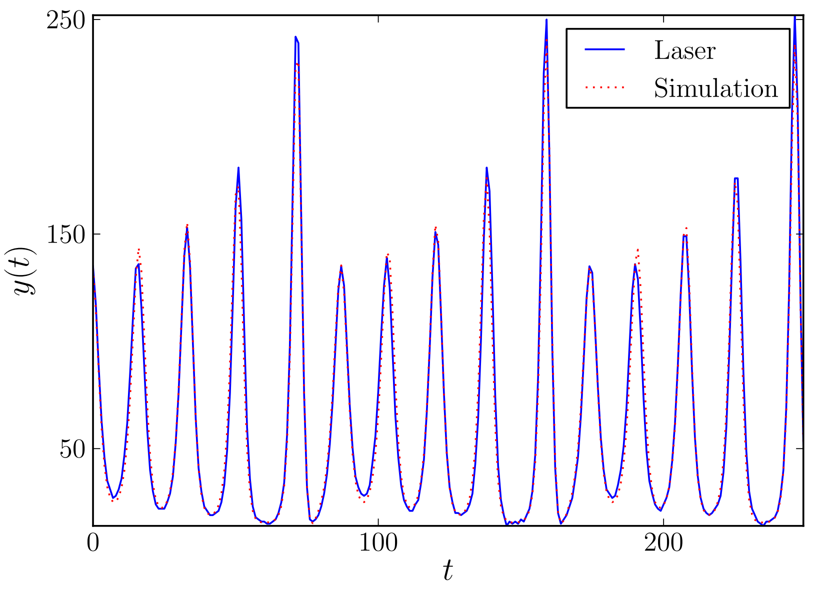


Figure 1.1: Laser intensity measurements. The trace labeled *Laser* is a plot of laser intensity measurements provided by Tang et al.. The trace labeled *Simulation* plots a numerical simulation of the Lorenz system (1.1) with parameters $r = 21.16$, $s = 1.792$, $b = 0.3670$, and measurement parameters $\tau_s = 0.1435$, $S_g = 7.071$, and $O_g = 15.16$. I used the optimization procedure described in the text to select these parameters. The simulated intensities were derived from the state by $y(t) = S_g \cdot (x_1(t))^2 + O_g$. I specified an absolute error tolerance of 10^{-7} per time step for the numerical integrator. {fig:LaserLP5}

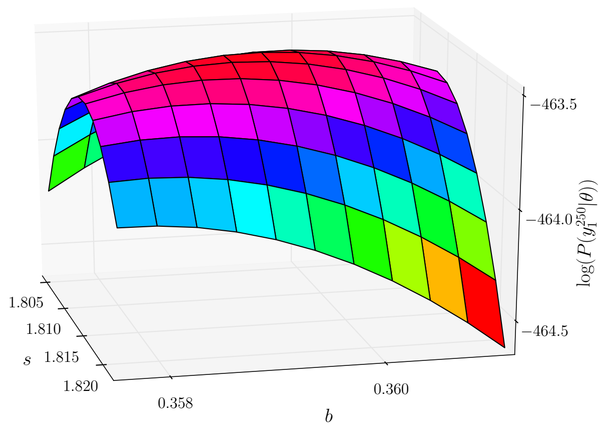


Figure 1.2: Log likelihood as function of s and b. Other parameters were taken from the vector $\hat{\theta}$ that maximizes the likelihood $P(y_1^{250}|\theta)$ (see Eqn. 1.3). {`fig:LaserLogLike`}

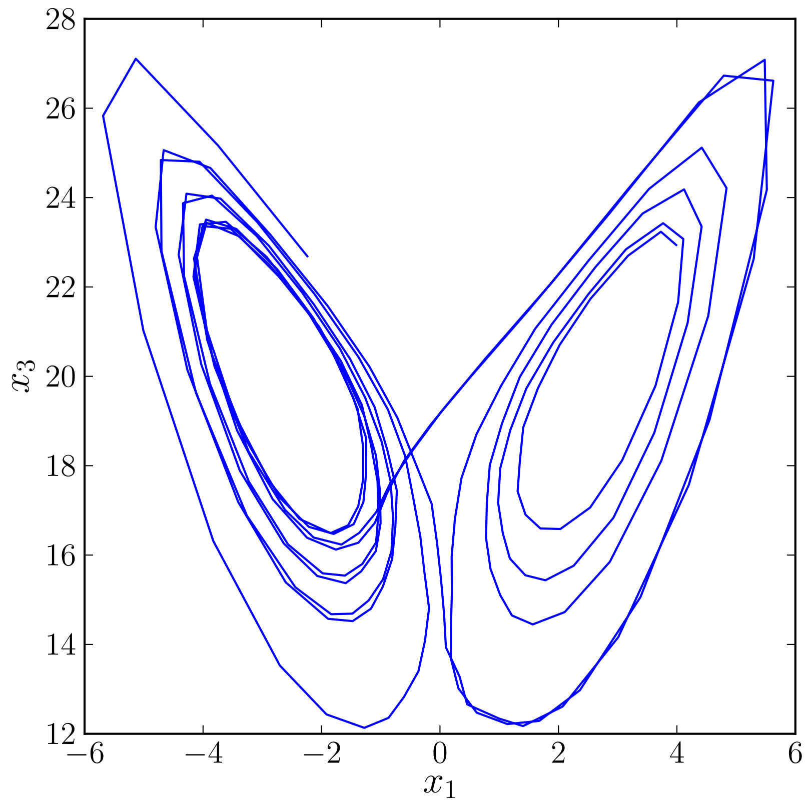


Figure 1.3: State trajectory $\hat{x}_1^{250}$ estimated from observation sequence y_1^{250}. (see Eqn. 1.4.) Components x_1 and x_3 of the Lorenz system (see Eqn. 1.1) are plotted. Recovering the familiar Lorenz figure suggests both that the laser data is *Lorenz like* and that the algorithm for estimating states from observations is reasonable. {fig:LaserStates}

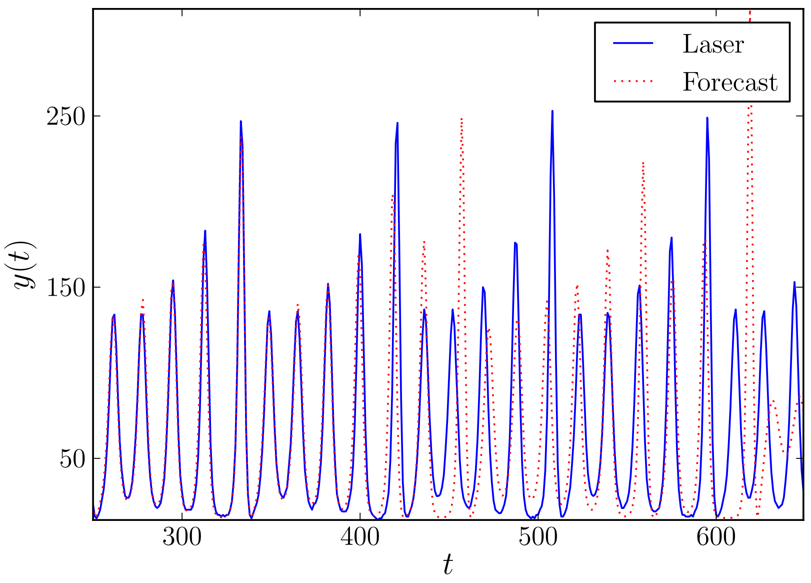


Figure 1.4: Forecast observation sequence. I set the noise terms η and ϵ to zero and iterated Eqn. 1.2 400 times to generate the forecast $\hat{y}_{251}^{650}$. I started with the initial condition $\hat{x}$ defined by Eqn. 1.5. The forecast begins to fail noticeably after $t = 500$. The failure suggests that the period five cycle in the forecast is unstable. The laser cycle must have been stable to appear in the data. Thus an essential characteristic of the model is wrong. {fig:LaserForecast}

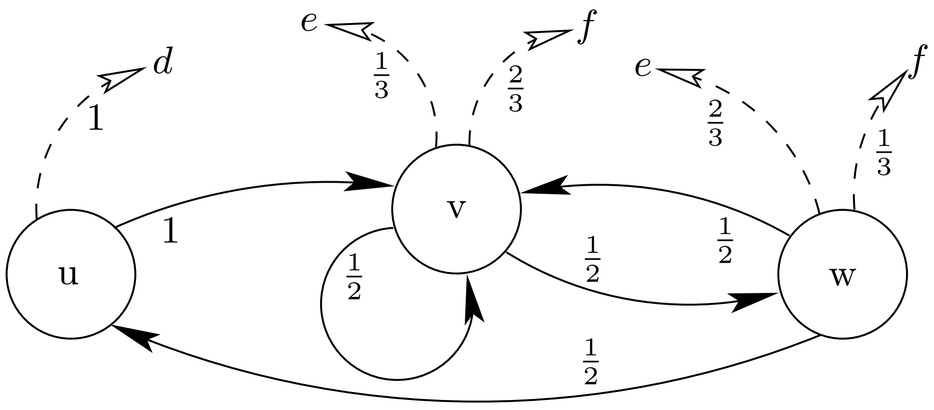


Figure 1.5: A Markov model {fig:mm}

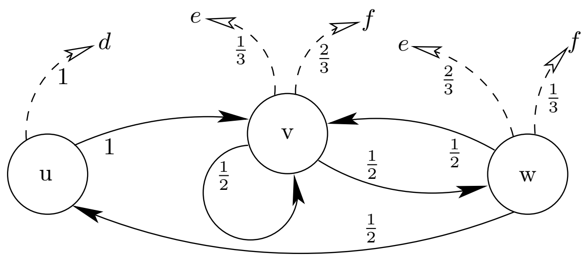


Figure 1.6: A hidden Markov model {fig:dhmm}

have not found matplotlib to be much easier to use than gnuplot.

Figure 1.9 is the picture on the cover of the book.

1.3.2 Example: Hidden States as Parts of Speech

{sec:POSpeech}

I've copied Table 1.1 from the source for the book. Since the actual table of words is the result of the new code (*po_speech.py*), the explanations do not match the word groups. Using different seeds for the random number generator produces different groupings. For now, each run takes 52 minutes. In the future, I will write the core routines in Cython which will reduce the time by an order of magnitude.

1.3.3 Remarks

{sec:DHMMRemarks}

Figures 1.10 and 1.11 are xfig drawings.

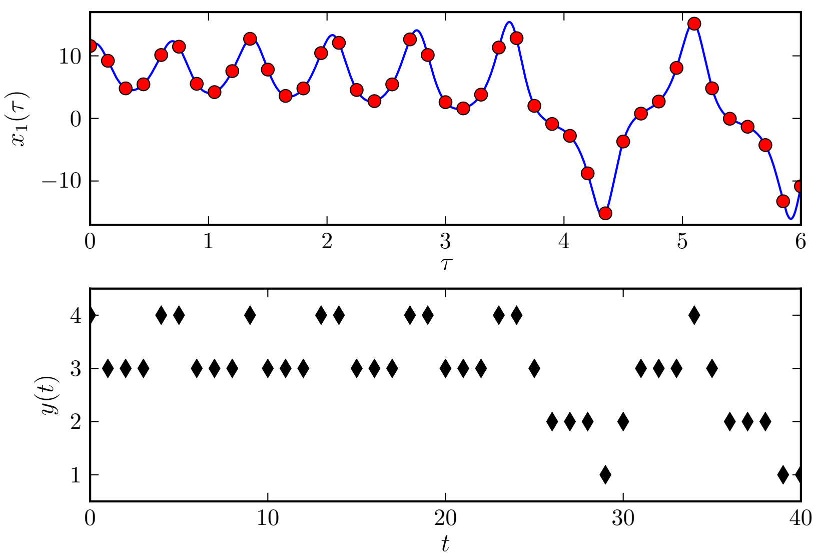


Figure 1.7: Generating the observations y_1^{40}. The curve in the upper plot depicts the first component $x_1(\tau)$ of an orbit of the Lorenz system (1.1), and the points marked ⋄ indicate the values sampled with an interval $\tau_s = 0.15$. The points in the lower plot are the quantized values $y(t) \equiv \left\lceil \frac{x_1(t \cdot \tau_s)}{10} + 2 \right\rceil$, where $\lceil u \rceil$ is the least integer greater than or equal to u. {fig:TSintro}

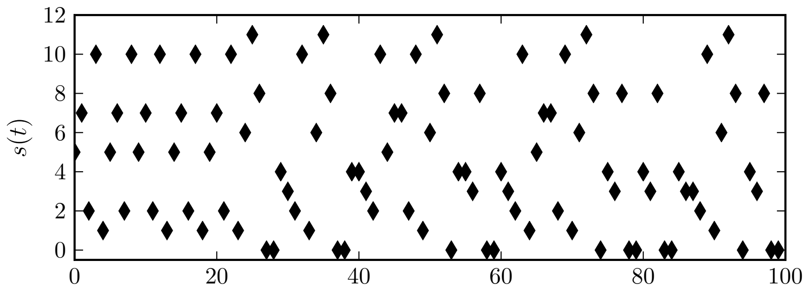


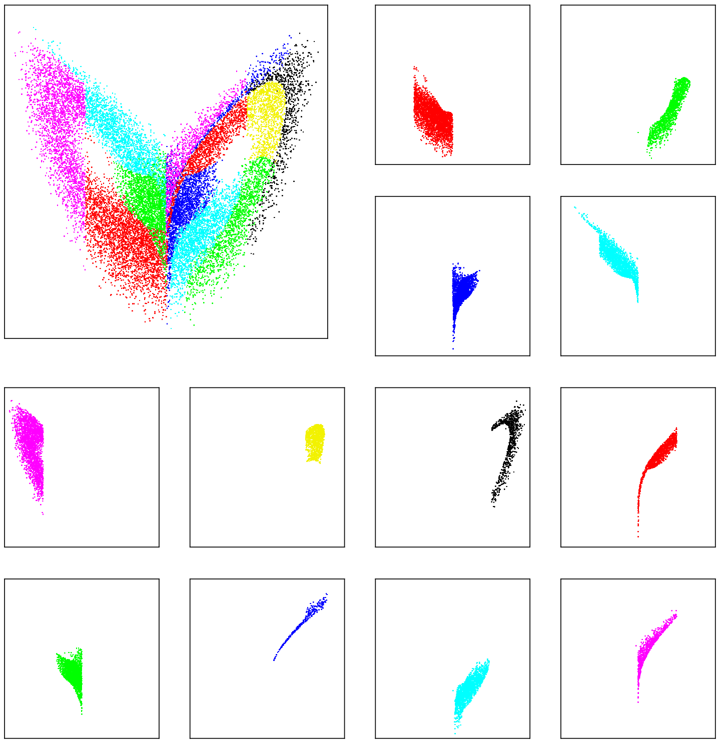
Figure 1.8: A plot of the state sequence found by Viterbi decoding a quantized time series from the Lorenz system. Here the number of the decoded state $s(t)$ is plotted against time t. Although it is hard to see any structure in the plot because the numbers assigned to the states are not significant, Fig. 1.9 illustrates that the decoded states are closely related to positions in the generating state space. {fig:STSintro}

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

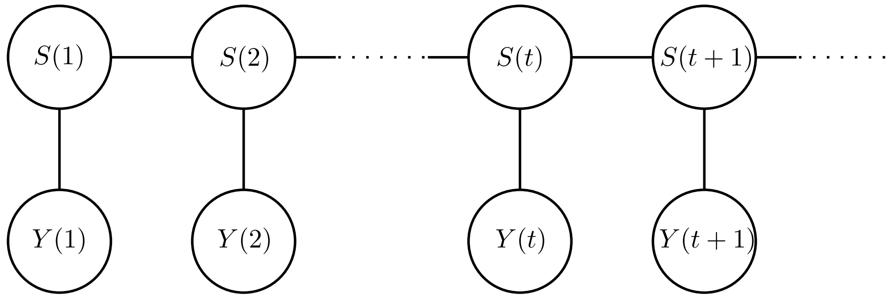
1 – Adjectives
2 – Punctuation and other tokens that appear at the end of phrases
6 – Capitalized articles and other tokens that appear at the beginning of phrases
7 – Objective pronouns and other words with similar functions
8 – Nouns
9 – Nouns
10 – Helping verbs
11 – Nominative pronouns
12 – Articles
13 – Conjunctions
14 – Prepositions
15 – Relative pronouns

1	work	book	man	York	books	story	hand	years	men	way
2	to	in	not	be	as	by	with	no	been	on
3	,	and	-	.	;	et	”	&	by	or
4	other	American	first	”	new	moral	same	own	whole	great
5	is	and	was	but	has	are	have	had	may	even
6	-	X	and	tm	”	half	self	but	electronic	rate
7	it	all	him	them	more	out	one	life	which	course
8	the	a	his	an	its	this	their	that	any	no
9	.	?	Project	!	:	;	Archive	*	where	Eyes
10	”	indeed	seq	Co	and	The	or	105	1917	12
11	The	In	But	Gutenberg	”	And	His	A	Here	That
12	he	it	I	and	He	they	It	there	we	you
13	of	and	in	to	for	as	with	by	or	from
14	,	that	;	”	which	who	as	out	what	(
15	”	.	,	]	[	The	and	1	A	)

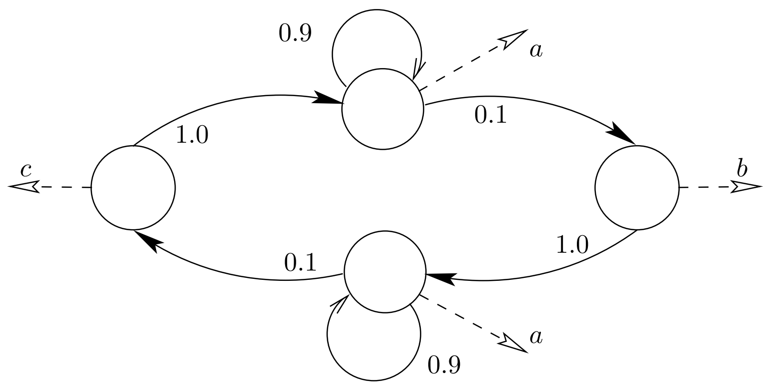
{tab:POS}



{fig:Statesintro} Figure 1.9: The relationship between the hidden states of an HMM and the original coordinates of the Lorenz system.



{fig:dhmm_net} Figure 1.10: 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.



{fig:nonmm} Figure 1.11: An HMM that cannot be represented by a Markov model of any order. Consider the string of observations "$b, a, a, \ldots, a, a, a$". For each "a" in the string, the previous non-"a" observation was "b". Since 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.

Chapter 2

Basic Algorithms

{chap:algorithms}

2.1 The Forward Algorithm

{sec:forward}

Figure 2.1 is an xfig drawing that invokes many LaTeXcommands defined in the file *software.tex*.

2.3 The Viterbi Algorithm

{sec:viterbi}

Figure 2.2 is straight LaTeXsource that is part of the file *software.tex*. Figure 2.3 is an xfig drawing that invokes commands defined in *software.tex*.

2.3.1 General Decoding

{sec:GenDecode}

2.3.2 MAP Sequence of States or Sequence of MAP States?

{sec:sequenceMAP}

Figure 2.4 is an xfig drawing.

2.4 The Baum-Welch Algorithm

{sec:baum_welch}

Reestimation

Table 2.1 is somewhat obscure straight LaTeX.
Fig. 2.5 is also obscure straight LaTeX.

2.4.2 Remarks

{sec:AlgApp}

Multiple Maxima

{sec:MultiMax}

2.5 The EM algorithm

{sec:EM}

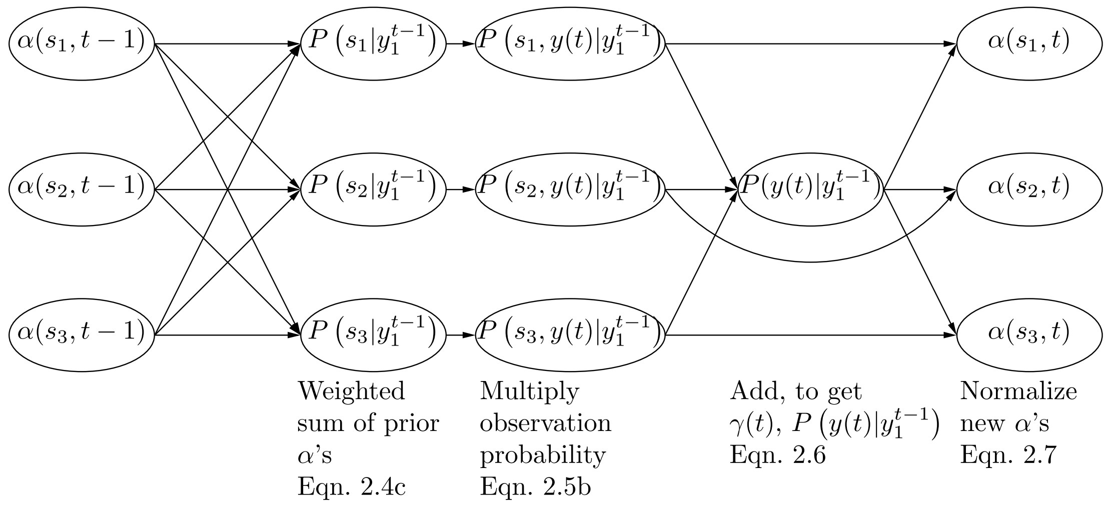

$\alpha(s_1, t-1)$
$\alpha(s_2, t-1)$
$\alpha(s_3, t-1)$
$P\left(s_1|y_1^{t-1}\right)$
$P\left(s_2|y_1^{t-1}\right)$
$P\left(s_3|y_1^{t-1}\right)$
$P\left(s_1, y(t)|y_1^{t-1}\right)$
$P\left(s_2, y(t)|y_1^{t-1}\right)$
$P\left(s_3, y(t)|y_1^{t-1}\right)$
$P(y(t)|y_1^{t-1})$
$\alpha(s_1, t)$
$\alpha(s_2, t)$
$\alpha(s_3, t)$
Weighted sum of prior α's Eqn. 2.4c
Multiply observation probability Eqn. 2.5b
Add, to get $\gamma(t)$, $P\left(y(t)|y_1^{t-1}\right)$ Eqn. 2.6
Normalize new α's Eqn. 2.7

Figure 2.1: Dependency relations in the forward algorithm (See Eqns. 2.4-2.7 in the text). The figure indicates the calculations the algorithm executes to incorporate the observation at time t for a three state model. {fig:forward}

```
Initialize:
    for each $s$
        $\nu_{\text{next}}(s) = \log\left(P_{Y(1),S(1)}\left(y(1), s\right)\right)$

Iterate:
    for $t$ from 2 to $T-1$
        Swap $\nu_{\text{next}} \leftrightarrow \nu_{\text{old}}$
        for each $s_{\text{next}}$

            for each $s_{\text{old}}$
                $\omega(s_{\text{old}}, s_{\text{next}}) = \nu(s_{\text{old}}, t) + \log\left(P(s_{\text{next}}|s_{\text{old}})\right)$
                    $+ \log\left(P(y(t+1)|s_{\text{next}})\right)$

            # Find best predecessor
            $B(s_{\text{next}}, t+1) = \text{argmax}_{s_{\text{old}}}\, \omega(s_{\text{old}}, s_{\text{next}})$

            # Update $\nu$
            $\nu_{\text{next}}(s_{\text{next}}) = \omega(B(s_{\text{next}}, t+1), s_{\text{next}})$
Backtrack:
    $\bar{s} = \text{argmax}_s\, \nu_{\text{next}}(s)$
    $\hat{s}(T) = \bar{s}$
    for $t$ from $T-1$ to 1
        $\bar{s} = B(\bar{s}, t+1)$
        $\hat{s}(t) = \bar{s}$
```

Figure 2.2: Pseudocode for the Viterbi Algorithm

{fig:viterbi}

Table 2.1: Summary of reestimation formulas. Note that formulas for $w(s,t)$ and $\tilde{w}(\tilde{s},s,t)$ appear in Eqns. 2.20 and 2.22 respectively.

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

{tab:reestimation}

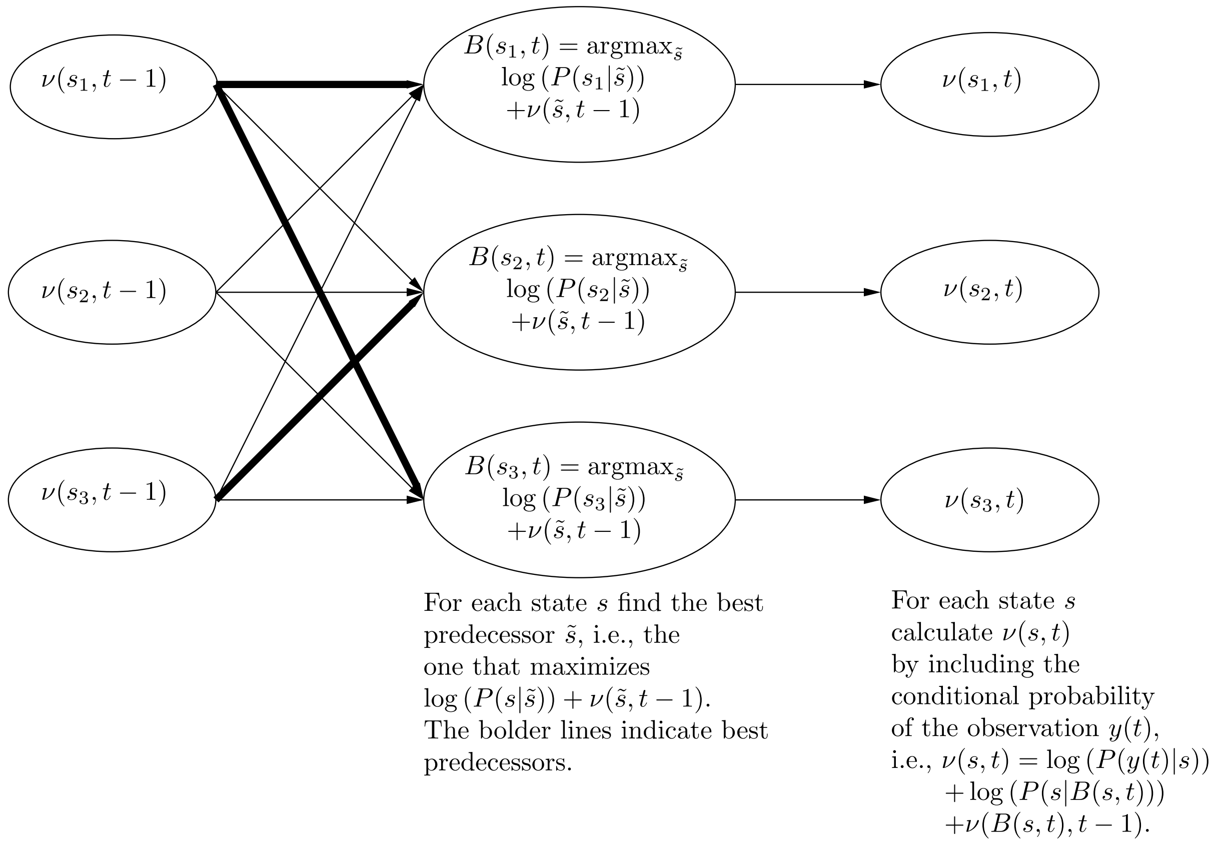

$\nu(s_1, t-1)$
$\nu(s_2, t-1)$
$\nu(s_3, t-1)$
$B(s_1, t) = \text{argmax}_{\tilde{s}}$ $\log(P(s_1|\tilde{s}))$ $+\nu(\tilde{s}, t-1)$
$B(s_2, t) = \text{argmax}_{\tilde{s}}$ $\log(P(s_2|\tilde{s}))$ $+\nu(\tilde{s}, t-1)$
$B(s_3, t) = \text{argmax}_{\tilde{s}}$ $\log(P(s_3|\tilde{s}))$ $+\nu(\tilde{s}, t-1)$
$\nu(s_1, t)$
$\nu(s_2, t)$
$\nu(s_3, t)$
For each state s find the best predecessor $\tilde{s}$, i.e., the one that maximizes $\log(P(s|\tilde{s})) + \nu(\tilde{s}, t-1)$. The bolder lines indicate best predecessors.
For each state s calculate $\nu(s,t)$ by including the conditional probability of the observation $y(t)$, i.e., $\nu(s,t) = \log(P(y(t)|s)) + \log(P(s|B(s,t))) + \nu(B(s,t), t-1)$.

Figure 2.3: Dependency relations in the Viterbi algorithm. {fig:viterbiB}

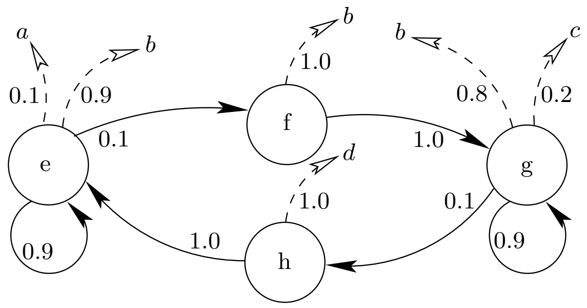


Figure 2.4: HMM used to illustrate that the maximum *a posteriori* sequence of states is not the same as the sequence of maximum *a posteriori* states. {fig:sequenceMAP}

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_1^T. See Eqns. 2.2 and 2.7.

$$\boldsymbol{\alpha}_n \equiv \left\{ P_{S(t)|Y_1^t,\theta(n)}\left(s|y_1^t,\theta(n)\right) : \forall s \in \mathcal{S} \,\&\, 1 \leq t \leq T \right\}$$

$\boldsymbol{\beta}_n$ is a set of values calculated on the basis of the n^{th} model $\theta(n)$ and the data y_1^T. See Eqns. 2.11 and 2.12.

$$\boldsymbol{\beta}_n \equiv \left\{ \frac{P_{Y_{t+1}^T|S(t)}\left(y_{t+1}^T|s\right)}{P\left(y_{t+1}^T|y_1^t\right)} : \forall s \in \mathcal{S} \,\&\, 1 \leq t < T \right\}$$

$\boldsymbol{\gamma}_n$ is the set of conditional observation probabilities calculated on the basis of the n^{th} model $\theta(n)$ and the data y_1^T. See Eqns. 2.3 and 2.6.

$$\boldsymbol{\gamma}_n \equiv \left\{ P\left(y(t)|y_1^{t-1},\theta(n)\right) : 2 \leq t \leq T \right\}$$

```
Initialize:
    Set $n = 1$ and choose $\theta(1)$

Iterate:
    $(\boldsymbol{\alpha}_n, \boldsymbol{\gamma}_n) \leftarrow$ forward$(y_1^T, \theta(n))$                                  See Section 2.1 page 20
    $\boldsymbol{\beta}_n \leftarrow$ backward$(\boldsymbol{\gamma}_n, y_1^T, \theta(n))$                                   See Section 2.2 page 25
    $\theta(n+1) \leftarrow$ reestimate $\left(y_1^T, \boldsymbol{\alpha}_n, \boldsymbol{\beta}_n, \boldsymbol{\gamma}_n, \theta(n)\right)$     See Table 2.2 page 33
    $n \leftarrow n + 1$
    Test for completion
```

Figure 2.5: Summary and pseudo-code for optimizing model parameters by iterating the Baum-Welch algorithm. {fig:train}

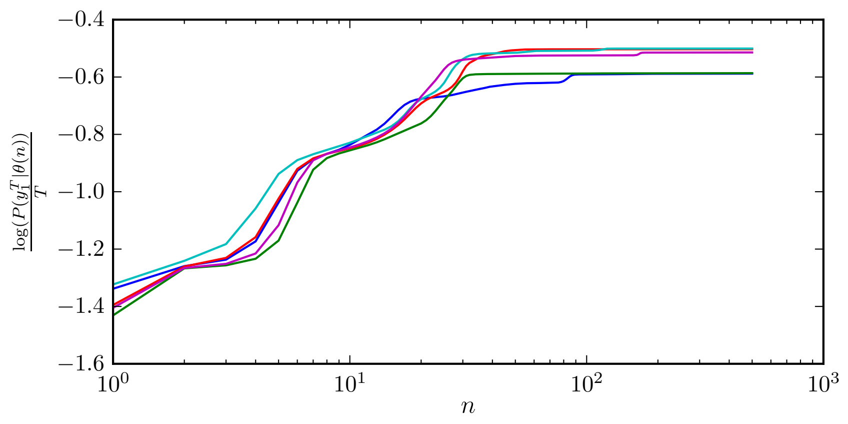


Figure 2.6: Convergence of the Baum-Welch algorithm. Here I have plotted $\frac{\log\left(P\left(y_1^T|\theta(n)\right)\right)}{T}$ (the log likelihood per step) as a function of the number of iterations n of the Baum-Welch algorithm for five different initial models $\theta(1)$. I used the same sequence of observations y_1^T that I used for Fig. 1.9, and I 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. Although it appears that three of the initial models all converge to -0.5, close examination of the data suggests that they are converging to different models with different log likelihoods per step. {fig:TrainChar}

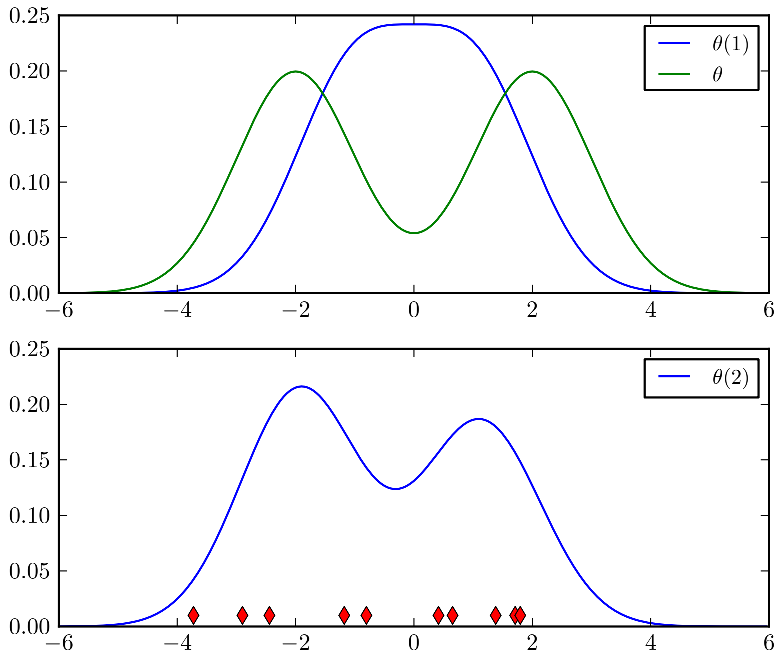


Figure 2.7: Two iterations of the EM algorithm. I use the algorithm to search for the parameters of Eqn. 2.49 that maximize the likelihood of the ten simulated observations that appear above in the row labeled $y(t)$. The triple of rows labeled $\theta(1)$ report the weighting calculated using Eqn. 2.50 used in Eqns. 2.51 to recalculate the conditional means, μ_1 and μ_2, for $\theta(2)$ in the M step, and the next triple of rows, labeled $\theta(2)$, report the same quantities for the calculation of $\theta(3)$. The parameters of the first three models appear in the row just below the boxed table. $\theta(3)$ is the triple of parameters produced by two iterations of the EM algorithm, $\lambda = 0.603$, $\mu_1 = -2.028$, $\mu_2 = 1.885$. On the axes of the upper plot, I illustrate $P(x|\theta)$ as defined in Eqn. 2.49 for two sets of model parameters: The dashed line depicts $\theta = (0.5, -2, 2)$, the distribution used to simulate the data, and the solid line depicts $\theta(1) = (0.5, -1, 1)$, the starting distribution I chose for the EM algorithm. On the bottom axes I plot the simulated observations as marks on the abscissa and $P(y|\theta(3))$ as a solid line. {fig:GaussMix}

2.5.2 Convergence

A contrived example

{sec:contrived}

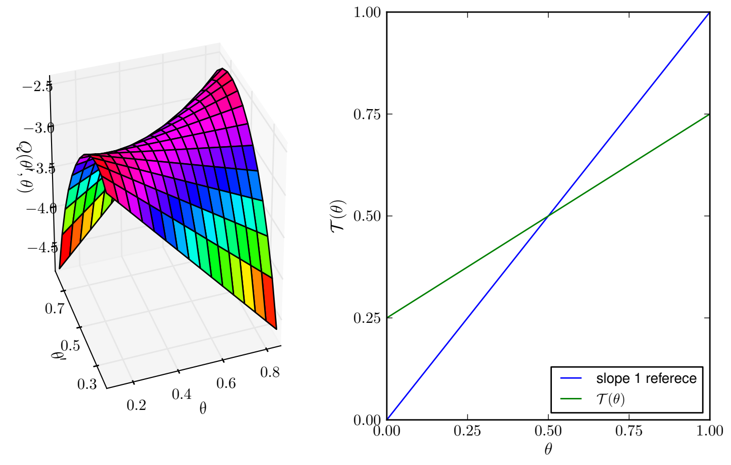


Figure 2.8: An illustration of the EM algorithm for an experiment in which a coin is thrown four times, first a head is observed ($y(1) = 1$), then a tail is observed ($y(2) = 0$), and finally two results are unobserved with s_h and s_t being the number of unobserved heads and tails respectively. The goal is to find the maximum likelihood value of θ, the probability of heads. The log likelihood function for the complete data is $L_\theta = (s_h + 1)\log(\theta) + (s_t + 1)\log(1 - \theta)$. The auxiliary function $Q(\theta', \theta) = (1 + 2\theta)\log(\theta') + (1 + 2(1 - \theta))\log(1 - \theta')$ appears on the left, and the map $\mathcal{T}(\theta)$ appears on the right. Note that $\theta^* = \frac{1}{2}$ is the fixed point of $\mathcal{T}$ (where the plot intersects the slope 1 reference line) and it is stable because the slope of $\mathcal{T}$ is less than one. {fig:EM}

Chapter 3

Variants and Generalizations

{chap:variants}

Laser_plots.py uses pylab to make Fig. 3.1 directly from Tang's file *data/LP5.DAT*.

3.1 Gaussian Observations

{sec:gaussian}

3.1.1 Independent Scalar Observations

{sec:ScalarGaussian}

3.1.2 Singularities of the likelihood function and regularization

{sec:regularization}

Figure 3.3 is an xfig drawing.

3.2 Related Models

{sec:related}

It takes 11 minutes for *VStatePic.py* to make the state data for Fig. 3.4 from *data/lorenz.xyz*.

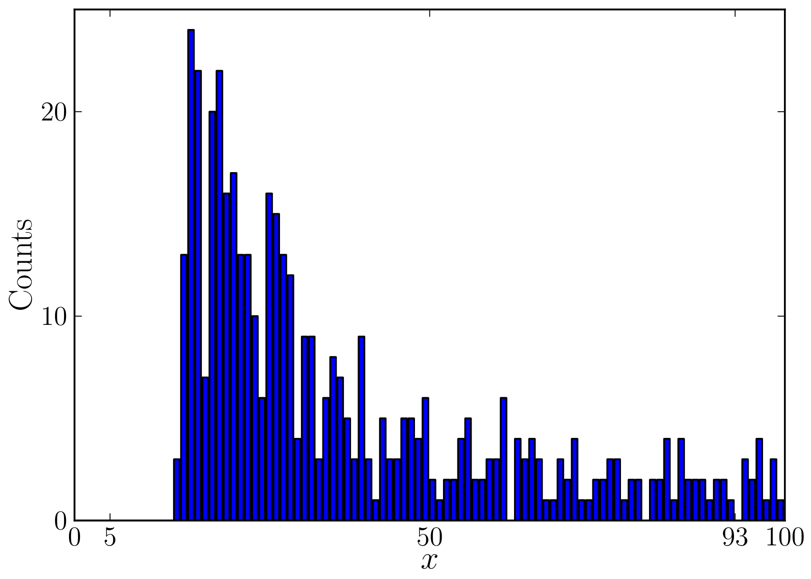


Figure 3.1: Histogram of Tang's laser measurements. Even though neither $y = 5$ nor $y = 93$ occurs in y_1^{600}, it is more plausible that $y = 93$ would occur in future measurements because of what happens in the neighborhood. Discarding the numerical significance of the bin labels would preclude such an observation. {fig:LaserHist}

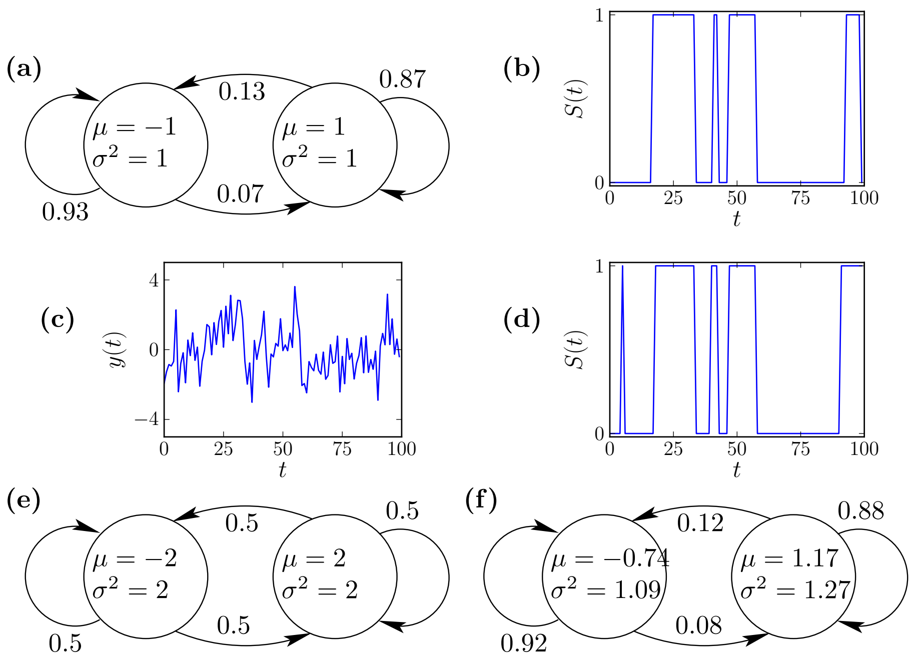


Figure 3.2: An HMM with scalar Gaussian observations. A state diagram appears in *(a)*. The half-life of the first state is about ten and the half life of the second state is about five, i.e., $0.93^{10} \approx 0.87^5 \approx 0.5$. A simulated state sequence and observation sequence appear in *(b)* 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)*. {fig:ScalarGaussian}

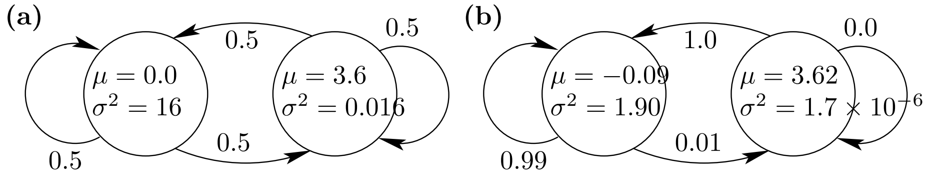

(a)
$\mu = 0.0$ $\sigma^2 = 16$
$\mu = 3.6$ $\sigma^2 = 0.016$
0.5
0.5
0.5
0.5
(b)
$\mu = -0.09$ $\sigma^2 = 1.90$
$\mu = 3.62$ $\sigma^2 = 1.7 \times 10^{-6}$
1.0
0.0
0.01
0.99

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

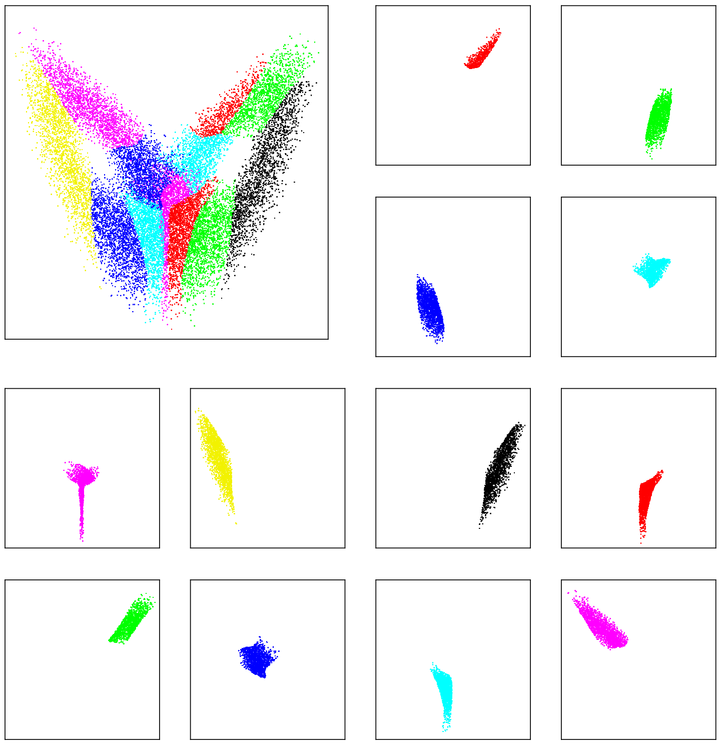


Figure 3.4: Plots of decoded states using an HMM with vector autoregressive observations. Here the observations are a trajectory of three dimensional state vectors from the Lorenz system. In each state the observation $y(t)$ is modeled as a Gaussian with a mean that is an affine (linear plus fixed offset) function of the observation $y(t-1)$. The empty boxes correspond to states that do not appear in the decoded sequence of states. In comparing with Fig. 1.9 which used a model with coarsely quantized observations, notice that large regions near the fixed points at centers of the spirals are represented by a single state. These large regions occur because the dynamics are approximately linear over their extents. {fig:VARGstates}

Chapter 5

Performance Bounds and a Toy Problem

{chap:toys}

Lorenz Example

I make Figs. 5.1 and 5.2 from the files *Save_Hview_T_100*, *Save_Hview_T_118*, and *Save_Hview_T_119* which I made using the old version of GUI program *Hview.py*. You can use the new version to make similar files and figures, but I like the way the old ones look. The script *ToyA.py* makes the figures from the data.

5.4 Benettin's Procedure for Calculating Lyapunov Exponents Numerically

{sec:Benettin}

Figure 5.4 is a boring collection of straight LaTeX and three xfig drawings.

5.6 Approaching the Bound

{sec:approach}

Fig. 5.6 from the data. Note: To make this figure you must have at least two gigabytes of RAM.

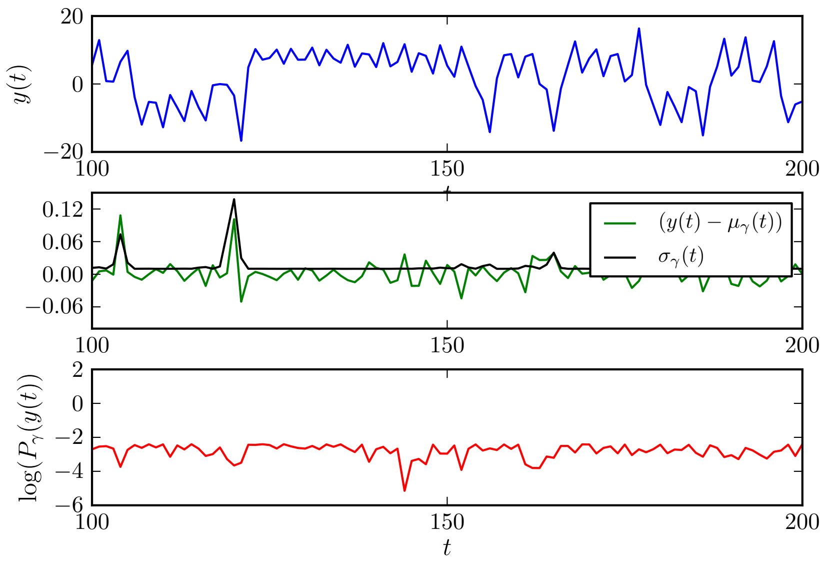


Figure 5.1: Extended Kalman filter for one step forecasting with simulation parameters:

$\tau_s = 0.25$	Sample interval
$\sigma_\eta = 10^{-6}$	Standard deviation of state noise
$\sigma_\epsilon = 0.01$	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 $P_\gamma\left(y(t)\right) \equiv P\left(y(t)|y_1^{t-1},\theta\right) = \mathcal{N}\left(\mu_\gamma(t),\sigma_\gamma^2(t)\right)\Big|_{y(t)}$; the first trace is the standard deviations of the forecasts and the second trace is the difference between the actual observation and the mean of the forecast. The logs of the likelihoods of the forecasts, $\log(P_\gamma\left(y(t)\right))$, appear in the bottom plot. **Note: The data comes from old software not in hmmds3.** {fig:ToyTS1}

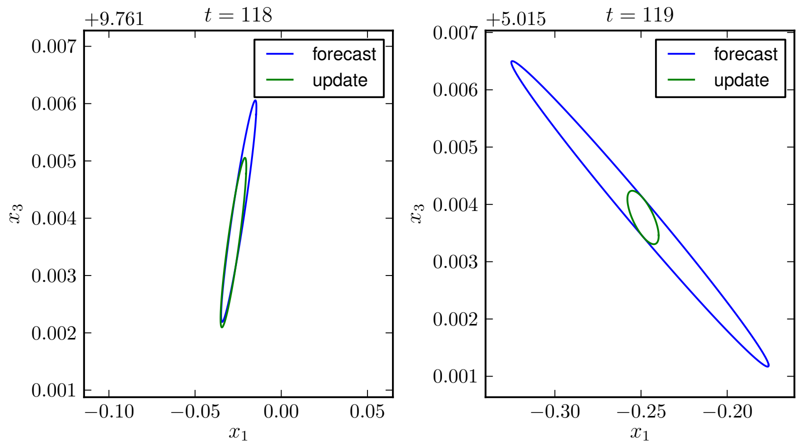


Figure 5.2: These plots illustrate dynamical stretching increasing the variance of the conditional distribution in state space corresponding to time steps 118 and 119 in Fig. 5.1. In each plot, the larger ellipse represents the *forecast* state distribution $P_a\left(x(t)\right) \equiv P\left(x(t)|y_1^{t-1},\theta\right) = \mathcal{N}\left(\mu_a,\Sigma_a\right)|_{x(t)}$ and the smaller ellipse represents the *updated* state distribution $P_\alpha\left(x(t)\right) \equiv P\left(x(t)|y_1^t,\theta\right) = \mathcal{N}\left(\mu_\alpha,\Sigma_\alpha\right)|_{x(t)}$. For each distribution, an ellipse depicts the level set $(x-\mu)^\top\Sigma^{-1}(x-\mu) = 1$ in the $x_1 \times x_3$ plane. Since the observations provide information about the value of x_1, the updated distributions vary less in the x_1 direction than the corresponding forecasts. To aid comparisons, the x_1 range is 0.2 and the x_3 range is 0.01 in each plot. In the x_1 direction, the standard deviation of the updated distribution $P_\alpha(x(t))$ at $t = 118$ (the smaller of the two ellipses on the left) is 0.007. The dynamics map that distribution to the forecast distribution $P_a(x(t))$ at $t = 119$ (the larger of the two ellipses on the right) for which the standard deviation in the x_1 direction is more than ten times larger. **Note: The data comes from old software not in hmmds3.** {fig:ToyStretch}

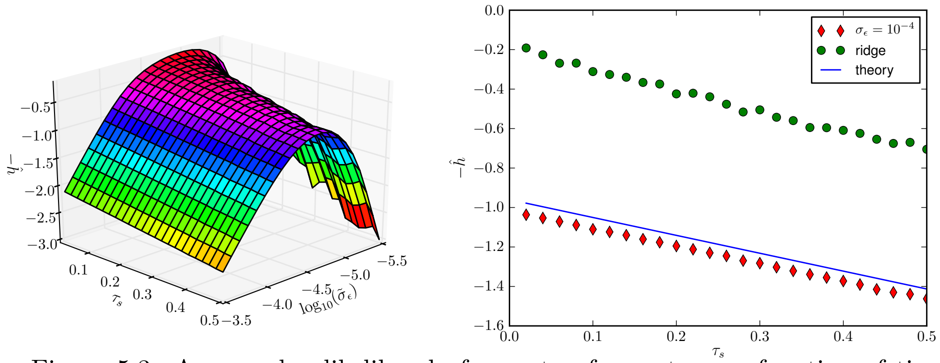


Figure 5.3: Average log likelihood of one step forecasts as a function of time step τ_s and filter parameter $\tilde{\sigma}_\epsilon$. To simulate measurements for this figure, I used the parameters:

$\sigma_\eta = 10^{-6}$	Standard deviation of state noise
$\sigma_\epsilon = 10^{-10}$	Standard deviation of measurement noise
$\Delta = 10^{-4}$	Measurement quantization
$T = 5,000$	Number of samples

For both plots, the vertical axis is the average log likelihood of the one-step forecast $-\hat{h} \equiv \frac{1}{T}\sum_{t=1}^{T} \log\left(P\left(y(t)|y_1^{t-1},\theta\right)\right)$. On the left I 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 Kalman filter. On the right "$\circ$"indicates the performance of filters that use measurement noise models that depend on the sampling time through the formula $\tilde{\sigma}_\epsilon(\tau_s) = 10^{0.4\tau_s - 4.85}$, which closely follows the ridge top in the plot on the left, "$\diamond$"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. {fig:ToyH}

$[[e_1]\quad [e_2]]$ $\qquad R\,[[e_1]\quad [e_2]]$ $\qquad QR\,[[e_1]\quad [e_2]]$

$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ $\qquad \begin{bmatrix}5 & 0.15\\ 0 & 0.2\end{bmatrix}$ $\qquad \begin{bmatrix}3 & 0.25\\ -4 & 0\end{bmatrix}$

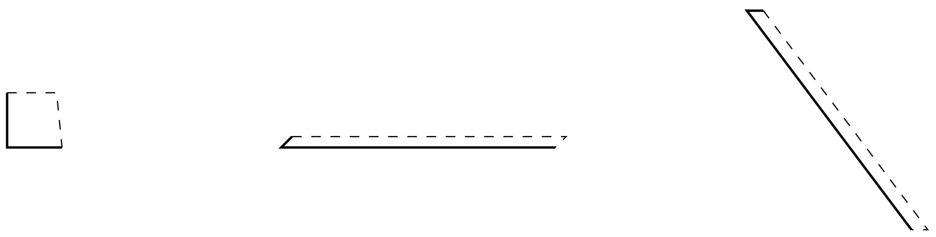


Figure 5.4: The action of the Q R factors of a matrix on a unit square. Here $A = \begin{bmatrix}3 & 0.25\\ -4 & 0\end{bmatrix}$, $Q = \begin{bmatrix}0.6 & 0.8\\ -0.8 & 0.6\end{bmatrix}$, and $R = \begin{bmatrix}5 & 0.15\\ 0 & 0.2\end{bmatrix}$. R stretches the x component by a factor of five and shears y components in the x direction and shrinks them by a factor of five with a net effect of preserving areas. Q simply rotates the stretched figure. Each parallelepiped in the bottom row is constructed from the columns of the corresponding matrix in the middle row. The algebraic formulas for those vectors appear in the top row. Note that R determines the changes in length and area, and that Q does not effect either. {fig:QR}

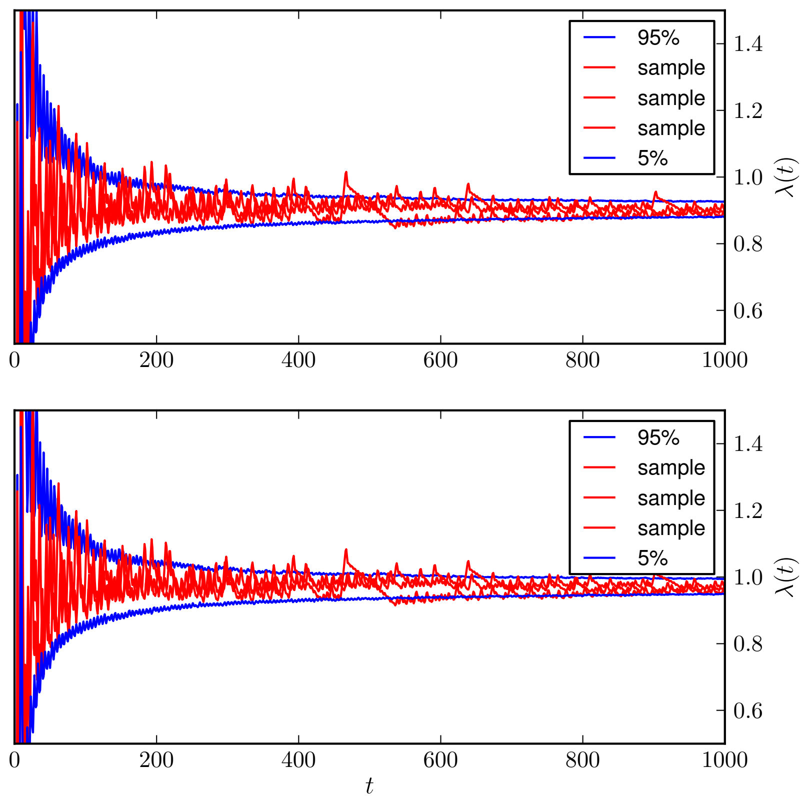


Figure 5.5: Lyapunov exponent calculation for the Lorenz system. In the upper part, the three lighter traces are plots of $\frac{1}{T}\sum_{t=1}^{T}\log\left(|r_{1,1}(t)|\right)$ and the heavier traces the 5% and 95% limits on 1,000 separate runs. The lower part is the same except that $|r_{1,1}(t)|$ is augmented by a noise term with amplitude $\frac{\sigma_\eta}{\Delta} = 0.01$ (See Eqn. 5.49. The shapes of the traces are almost unchanged except for uniform shift up of about 0.03. I conclude that a test model that is the same as the generating model except that the state noise is $\frac{\sigma_\eta}{\Delta} = 0.01$ would have a cross entropy of about 0.936 nats, while the largest Lyapunov exponent is $\hat{\lambda}_1 \approx 0.906$ nats. {fig:benettin}

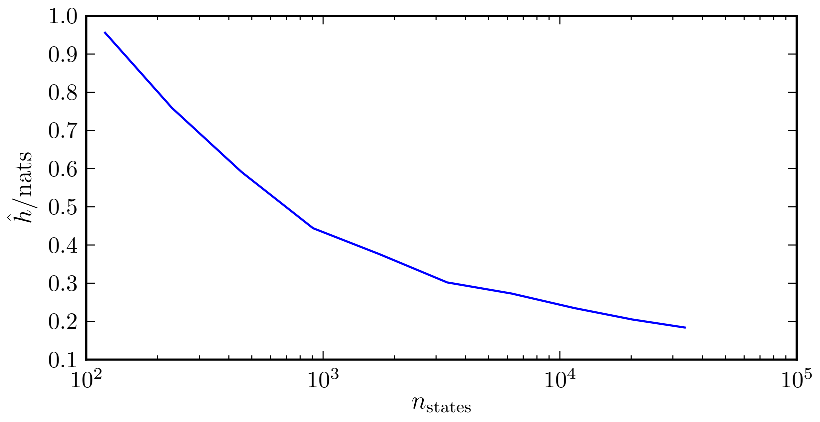


Figure 5.6: Entropy gap, $\hat{\delta}_{\mu||\theta}$ vs number of states in HMMs. The upper trace plots estimates of cross entropy $\hat{h}(\mathcal{B}, F, \mu||\theta)$ for a sequence of HMMs vs the number of discrete state in the models. I built the models using actual Lorenz state space trajectories as described in the text. The lower trace is an estimate of the entropy rate, $\hat{h}(F, \mu)) = \hat{\lambda}_1$, 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 might perform at least as well as any other model based on any other technology. Each model was built using the same sample trajectory of 8,000,000 points in the original state space, and the cross entropy estimates are based on a test sequence of 10,000 observations. {fig:LikeLor}

Chapter 6

Obstructive Sleep Apnea

{chap:apnea}

6.1 The Challenge and the Data

{sec:challenge}

6.1.1 The Data

{sec:data}

6.2 First Classification Algorithms and Two Useful Features

{sec:NVG}

6.2.2 Nonlinear Dynamics

{sec:NLD}

6.2.3 The Excellent Eye of Dr. McNames

{sec:mcnames}

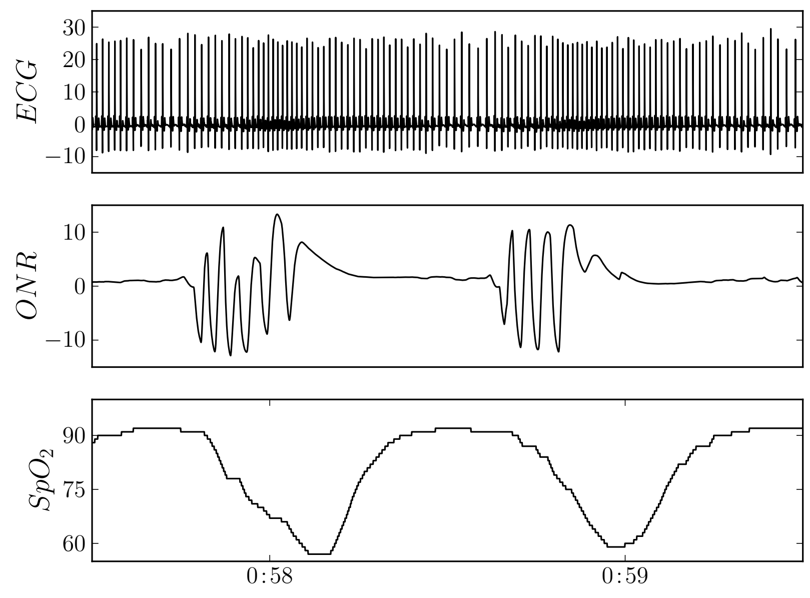


Figure 6.1: A segment of record a03. Two cycles of a large apnea induced oscillation in SpO_2 are drawn in the lower plot. The middle plot is the oronasal airflow signal, and the upper plot is the ECG (units of both ONR and ECG are unknown). The time axis is marked in *hours:minutes*. Notice the increased heart rate just after 0:58 and just before 0:59. {fig:a03erA}

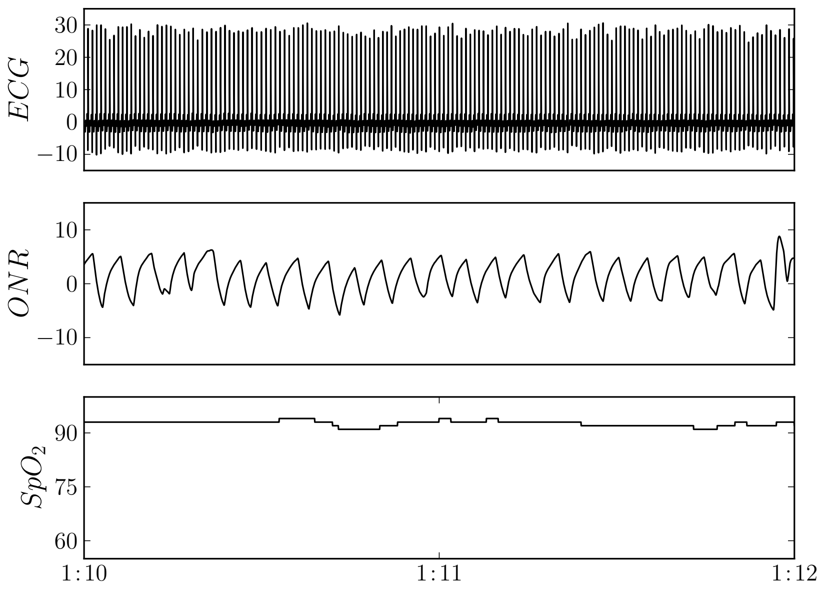


Figure 6.2: A segment of record a03 taken during a period of normal respiration. Signals the same as in Fig. 6.1.

{fig:a03erN}

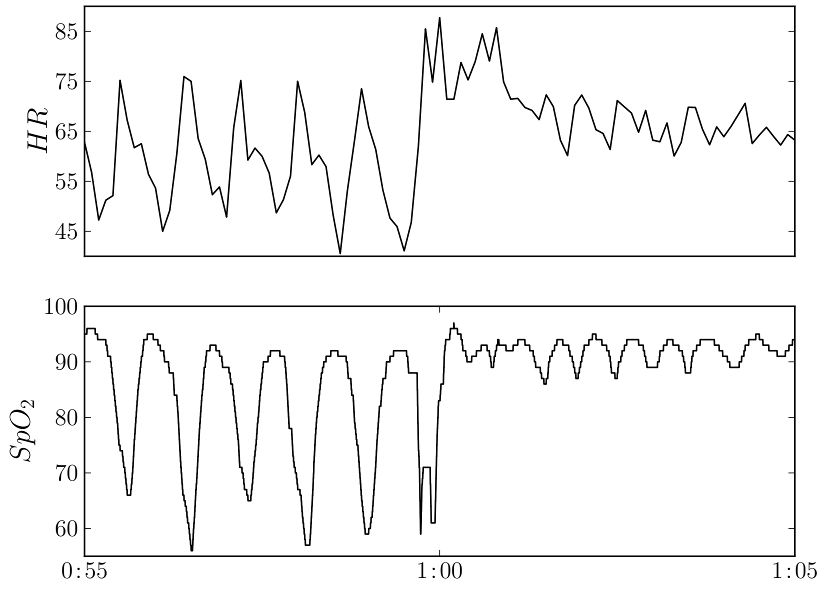


Figure 6.3: A segment of record 03 at the end of an episode of apnea with indications in both the SpO_2 signal and the heart rate *(HR)* signal. The expert marked the time before 1:00 as apnea and the time afterwards as normal. {fig:a03erHR}

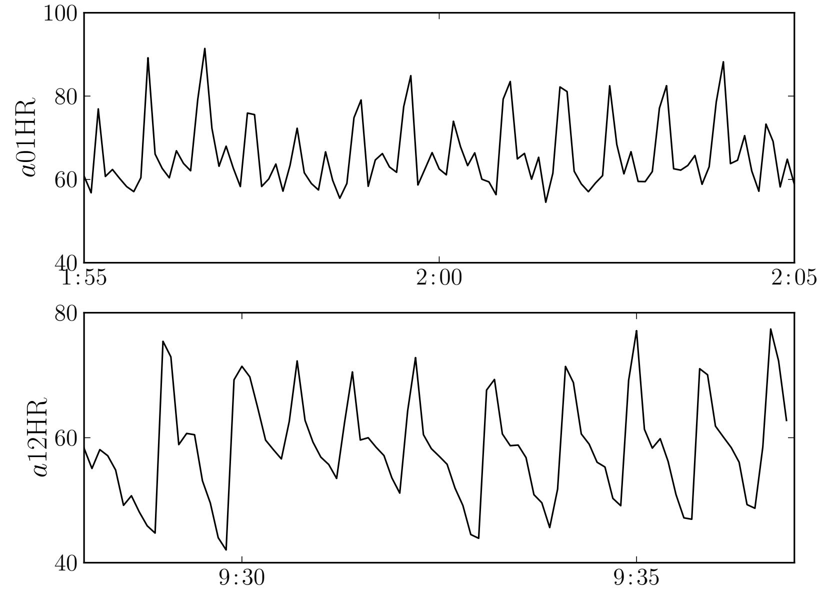


Figure 6.4: Nonlinear effects: The upper plot seems to be a period two oscillation. The lower plot is approximately sawtooth. {fig:ApneaNLD}

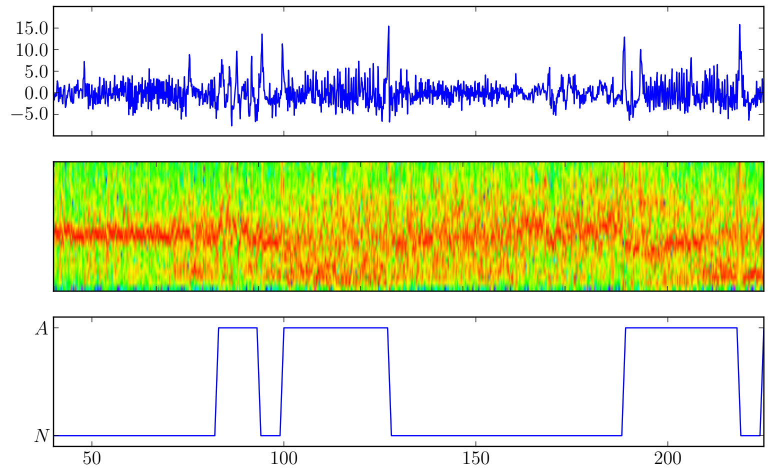


Figure 6.5: Information about respiration in high frequency phase variations. This is the $a11$ record roughly between minutes 40 and 225. The upper plot is heart rate (bandpass filtered 0.09-3.66 cpm), the middle plot is a spectrogram of the phase jitter in the heart rate, and the lower plot is the expert classification. A single band of spectral power between about 10 and 20 cpm without much power below the band in the spectrogram indicates normal respiration. {fig:sgram}

6.3 Decoding Sequences of Classifications

{sec:V4Class}

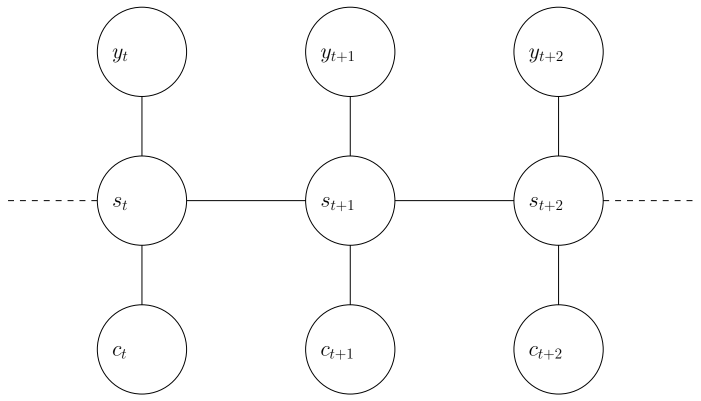
In the book I presented the algorithm described in Fig. 6.6 for finding the *best* classification sequence,

$$\hat{c}_1^T \equiv \underset{c_1^T}{\operatorname{argmax}} P(y_1^T, c_1^T),$$

given an observation sequence y_1^T. I claimed that the number of computations required is a linear function of T. In 2013, as I was developing a test suite for a new version of the software, I discovered a configuration for which the algorithm failed to find a *possible* classification sequence even though it was using the model that generated the test data.

Examining the failure, I discovered that for each of the "best" class histories ending in each of the classes at a particular time t, the conditional probability of a particular state s given that class history was 0. However, at time $t+1$, the only state that could produce $y(t+1)$ required that $S(t) = s$. Thus the algorithm calculated $P(y_1^t, \hat{c}_1^t) = 0$ and died.

I had designed the algorithm thinking that the class sequence was Markov and that future evidence could not change what was the best history leading to any class. The following drawing makes my error obvious.



Blocking out s_{t+1} separates the past from the future, but blocking out c_{t+1} doesn't.

Thus to avoid discarding a class history that may at later times become the first portion of the *best* class sequence, one may need to keep a number of class histories that is exponential in sequence length t. That is too many to retain for most realistic applications.

6.3.1 Finding a *Pretty Good Class* Sequence

{sec:prettygood}

Rather than finding the *best* class sequence given a sequence of observations y_1^T, I now seek a *good* one using the following ideas. The ideas yield performance similar to that described in the book when applied to the apnea problem. At each time t, I keep a limited collection of class histories which I build in the following three steps:

Step 1, Propagate and Sort: I take the histories I've retained at time $t-1$ and use all possible successors to create a list of histories at time t, then I sort those histories by $P(y_1^t, c_1^t)$.

Step 2, Prune: Initially, I limit the collection using the following criteria:

- If there are two histories ${}^a c_1^t$ and ${}^b c_1^t$ with $P(s_i, y_1^t, {}^b c_1^t) \leq P(s_i, y_1^t, {}^a c_1^t) \forall s_i$, then I drop ${}^b c_1^t$.
- The total number of histories is at most $N_{\max}$
- I don't keep histories c_1^t if $\frac{P(y_1^t, c_1^t)}{P(y_1^t, \bar{c}_1^t)} < R_{\min}$ where $\bar{c}_1^t$ is the best class history in the sorted list.

Step 3, Augment: For each state $s(t)$ that is impossible given the histories saved so far, if there are discarded histories for which $s(t)$ is possible, I save the one that comes first in the sorted list.

```
Initialize:
    for each $c$
        $\nu_{\text{next}}(c) = \log\left(\sum_s g(s, C) P_{Y(1),S(1)}(y(1), s)\right)$

Iterate:
    for $t$ from 1 to $T$
        Swap $\nu_{\text{next}} \leftrightarrow \nu_{\text{old}}$
        for each $c_{\text{next}}$

            # Find best predecessor
            $c_{\text{best}} = \text{argmax}_{c_{\text{old}}} \left(\nu_{\text{old}}(c_{\text{old}}) + \log\left(\sum_s g(s, c_{\text{best}}) f(t+1, s, c_{\text{best}})\right)\right)$

            # Update $\nu$
            $\nu_{\text{next}}(c_{\text{next}}) = \nu_{\text{old}}(c_{\text{best}}) + \log\left(\sum_s g(s, c_{\text{best}}) f(t+1, s, c_{\text{best}})\right)$

        # Update predecessor array
        Predecessor[$c_{\text{next}}, t$] = $c_{\text{best}}$

        # Update $\phi$
        for $s$ in $c_{\text{next}}$
            Assign $\phi_{\text{next}}(s, c_{\text{next}})$ using Eqn. 6.7

Backtrack:
    $c_1^t = \hat{c}_1^t(\bar{c})$ , where $\bar{c} = \text{argmax}_c \nu_{\text{next}}(c)$ at $t = T$
```

{fig:viterbiC} Figure 6.6: **This algorithm does not work!** In the book, this figure was entitled *Pseudocode for the Viterbi algorithm for class sequences.*

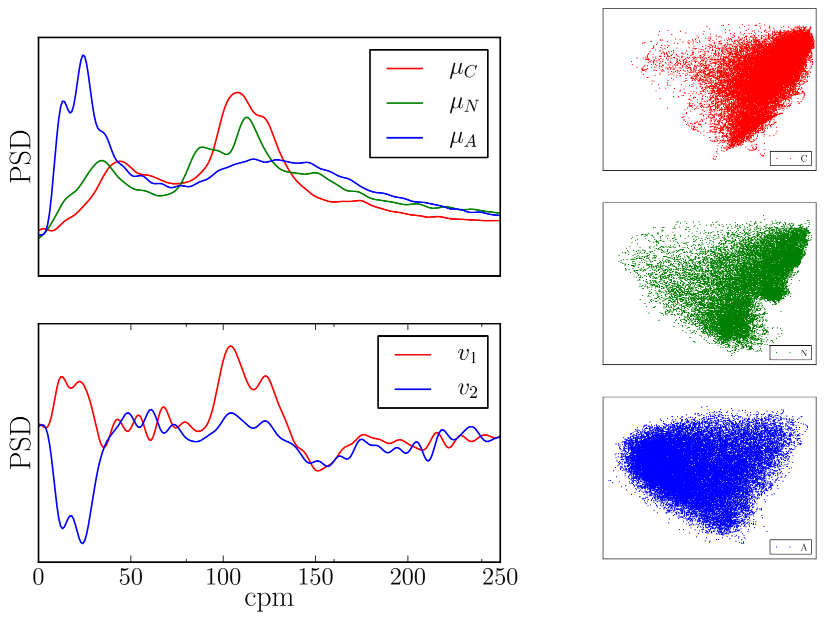


Figure 6.7: Linear discriminant analysis of phase jitter periodograms. The plot in the upper left, shows the following mean periodograms: μ_C, the mean for the c records; μ_N, the mean of the minutes that the expert classified as normal in the a records; and μ_A, the mean of the minutes that the expert classified as apnea in the a records. The lower left plot shows the basis vectors that result from the linear discriminant analysis. Scatter plots of the three classes projected on the basis (v_1, v_2) appear on the right. {fig:LDA}

6.4 Assembling the Pieces

{sec:Pieces}

6.4.1 Extracting a Low Pass Filtered Heart Rate

{sec:LPHR}

6.4.2 Extracting Respiration Information

{sec:RESP}

6.4.3 Classifying Records

{sec:ClassRec}

6.4.4 Model Topology and Training Data

{sec:topology}

Figure 6.9 is a simple xfig drawing.

6.4.5 Tunable Parameters

{sec:tune}

It takes a long time make the data for Fig. 6.10. To let readers avoid that delay, I've included the data in the source package. If you want to build the

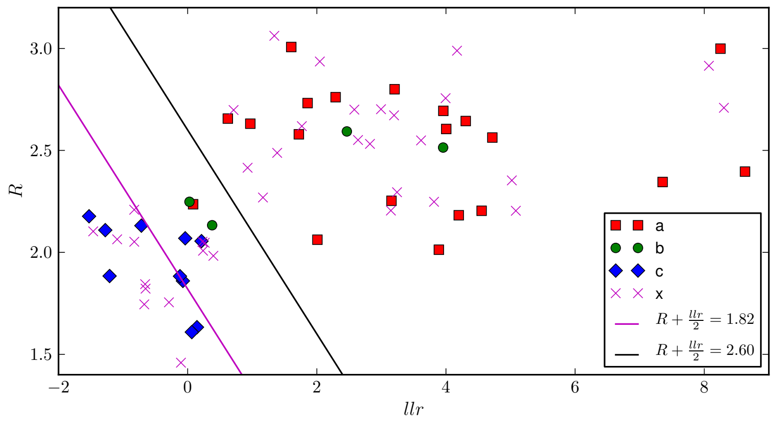


Figure 6.8: The first pass classifier. I've plotted the location of each record using the log likelihood ratio llr and the ratio statistic R. Records to the left of the line $2.39 - \frac{llr}{2}$ are in the L group. Records to the right of the line $2.55 - \frac{llr}{2}$ are in the H group. And those in between are in the M group. {fig:pass1}

data, simply remove *data/PFsurveyH*, make *data/PFsurveyH* and copy it to *data/PFsurvey*.

{sec:results}

6.4.6 Results

The results in Table 6.1 are part of the typed in LaTeXsource. I should write code that creates a file of results and then import the results into the text.

Table 6.2 is a copy of the table in the book. It does not reflect the performance of the new code.

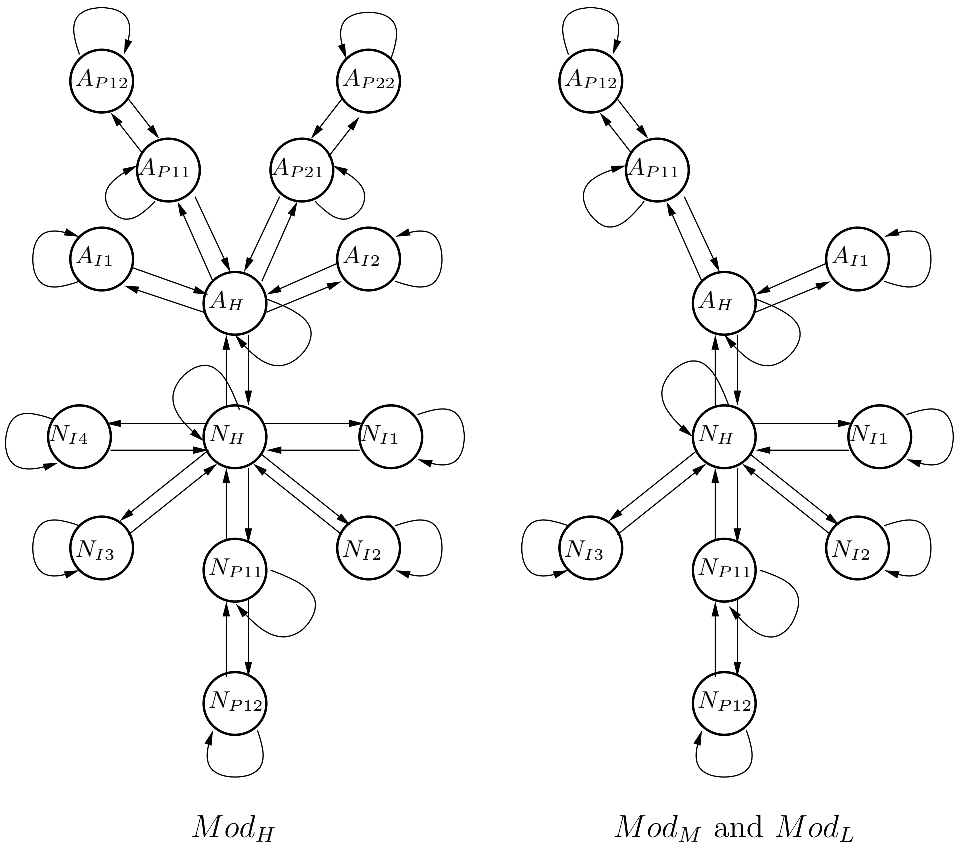


Figure 6.9: Structure of HMMs for minute by minute classification in the second pass of my procedure. I used the structure on the left for those records classified as H on the first pass and the structure on the right for those records classified as M or L on the first pass. {fig:structure}

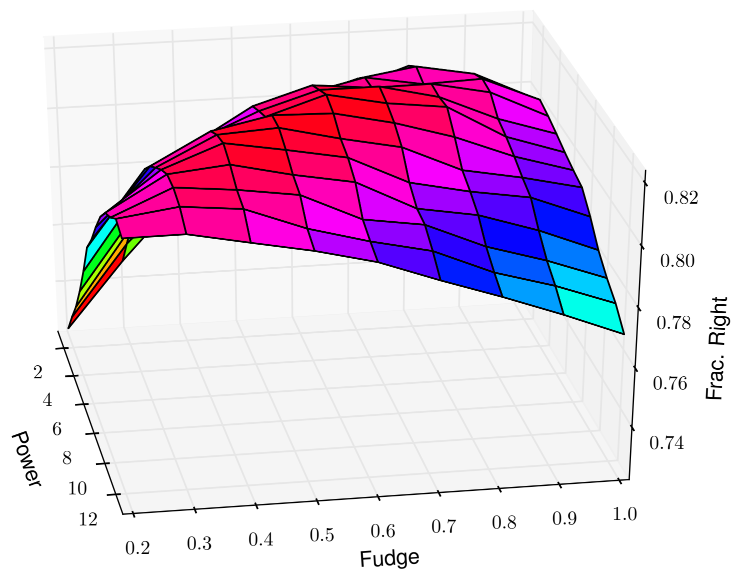


Figure 6.10: The response of classification performance to changes in *Pow* and *Fudge*. I've plotted the performance of Mod_H trained and evaluated on the H group of records. As described in the text, *Pow* governs the relative weighting of the low pass heart rate signal to the respiration characteristics and *Fudge* is a bias for choosing the *normal* classification. The Z axis is the fraction of minutes classified correctly. It takes about 3.75 days to make the data for this plot. {fig:PFsurvey}

Table 6.1: Performance with tuned values of *Fudge* and *Pow* on training records. I've sorted the list in order of how well the code classified each of the minutes in each record. For each record, the number in the column labeled $N \to A$ is the number of minutes labeled as *normal* by the expert that the code labeled as *apnea*. The interpretations of the other columns are similar.

Record	$N \to N$	$N \to A$	$A \to N$	$A \to A$	% Right
a11	198	46	138	84	0.6052
b02	255	169	14	79	0.6460
a06	276	27	140	66	0.6719
a08	197	114	24	165	0.7240
b01	362	105	2	17	0.7798
a07	96	93	12	309	0.7941
a18	37	14	82	356	0.8037
b03	296	71	13	60	0.8091
a03	175	98	0	246	0.8112
a20	184	10	78	237	0.8271
a15	91	50	36	332	0.8310
a05	147	30	44	232	0.8366
a16	140	21	51	269	0.8503
a13	213	38	28	215	0.8664
a09	90	24	39	342	0.8727
a10	404	13	49	50	0.8798
a14	69	57	2	381	0.8841
a17	302	24	32	126	0.8843
a02	72	36	19	401	0.8958
a19	289	8	30	174	0.9242
a12	14	29	3	530	0.9444
b04	418	0	10	0	0.9766
a01	11	8	0	470	0.9836
a04	35	4	2	451	0.9878
c07	424	0	4	0	0.9907
c05	462	0	3	0	0.9935
c09	465	0	2	0	0.9957
c10	429	0	1	0	0.9977
c03	452	1	0	0	0.9978
c06	466	0	1	0	0.9979
c02	500	0	1	0	0.9980
c01	483	0	0	0	1.0000
c04	481	0	0	0	1.0000
c08	513	0	0	0	1.0000
sum	9046	1090	860	5592	0.8824

{tab:result1}

Table 6.2: Here are the scores described in this chapter interspersed with the top scores from the CINC2000 website (`http://www.physionet.org/challenge/2000/top-scores.shtml`).

{tab:cinc2000}

Score	Entrant	Entries
92.62	J McNames, A Fraser, and A Rechtsteiner Portland State University, Portland, OR, USA	4
92.30	B Raymond, R Cayton, R Bates, and M Chappell Birmingham Heartlands Hospital, Birmingham, UK	8
89.36	P de Chazal, C Henehan, E Sheridan, R Reilly, P Nolan, and M O'Malley University College - Dublin, Ireland	15
87.56	M Schrader, C Zywietz, V von Einem, B Widiger, G Joseph Medical School Hannover, Hannover, Germany	9
87.30	MR Jarvis and PP Mitra Caltech, Pasadena, CA, USA	3
86.95	Second entry in this chapter. Adjust *Fudge* to get fraction of apnea minutes in the test records to match the fraction of apnea minutes in the training records	
86.24	First entry in this chapter. Models and parameters tuned to the training records.	
85.63	Z Shinar, A Baharav, and S Akselrod Tel-Aviv University, Ramat-Aviv, Israel	1
85.54	C Maier, M Bauch, and H Dickhaus University of Heidelberg, Heilbronn, Germany	5
84.49	JE Mietus, C-K Peng, and AL Goldberger Beth Israel Deaconess Medical Center, Boston, MA, USA (unofficial entry)	