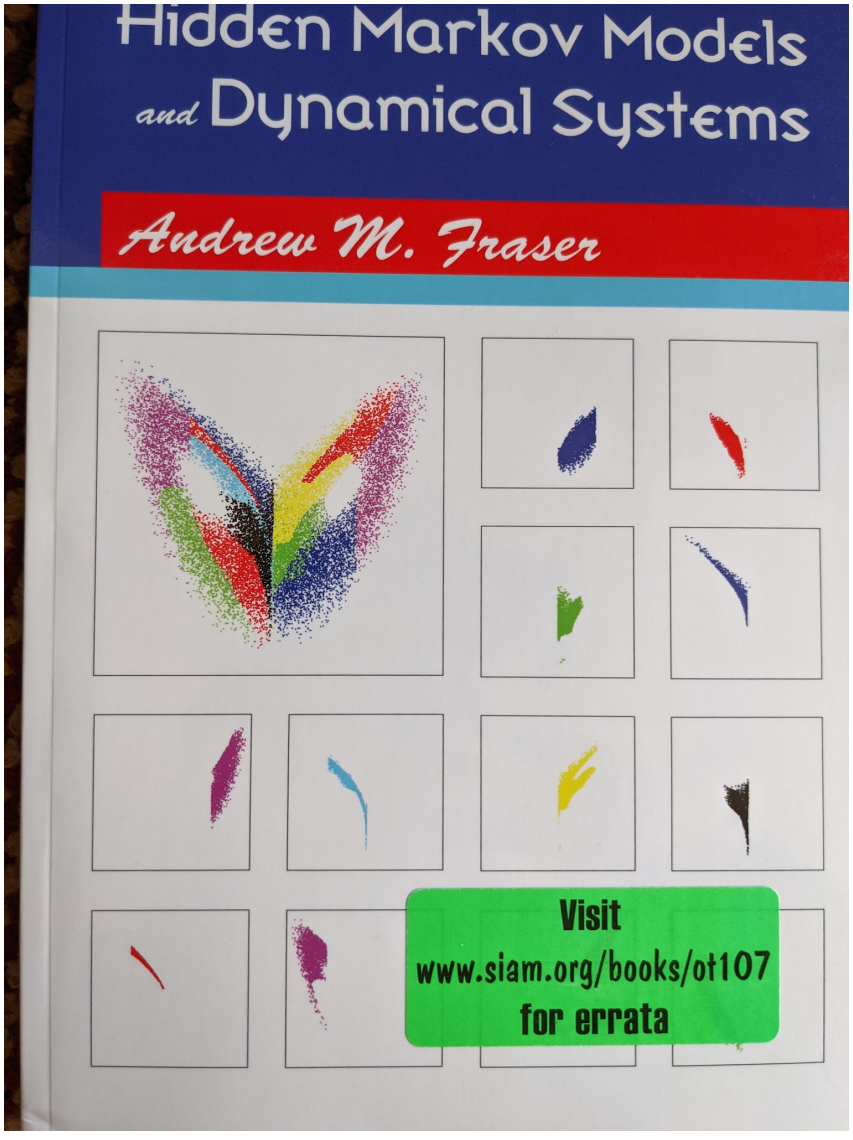
Data Assimilation and Software for Reliability

Andrew M. Fraser

SIAM DS21

2021-5-24

Book from MS on Hidden Markov Models at DS01



Discrete state dynamics

$$P(s[t+1] \mid s[t])$$

Discrete observations

$$P(y[t] \mid s[t])$$

Goals for Book and Software

Reproducible: Fetch source, type *make*, wait $\sim$ 30 hours, view resulting *main.pdf*.

Readable Code: In 2002 I chose Python instead of *C*, *Perl* and or *Octave*.

Shortcomings:

- Unreadable *Makefile*. Love/hate relationship with *make*.
- I focused on appearance of result at expense of readability.
- Code was difficult to read.
- No testing framework.

Best Practices for Scientific Computing

Discovered *Software Carpentry* in 2015. *Best Practices for Scientific Computing* by G. Wilson et al. PLOS 2014 says:

- Write programs for people, not computers. (Tools like black and pylint help.)
- Let the computer do the work. (Use a build tool.)
- Make incremental changes. (Put everything that has been created manually in version control.)
- Don't repeat yourself. (Every piece of data must have a single authoritative representation in the system.)
- Plan for mistakes. (Use an off-the-shelf unit testing library.)
- Optimize software only after it works correctly.
- Document design and purpose, not mechanics.
- Collaborate.

New Text and Software for Book

Goals: Follow best practices

Implemented Testing: Wham! Test of decoding sequences of classes fails.

Conceptual Error: Code assumes classes have Markov property like states (more on this later).

Morals: Follow best practices. Listen to doubters.

New Text and Software for Book (2)

Progress on new code after Los Alamos:

Follow Google Coding Standards:

Built in Documentation:

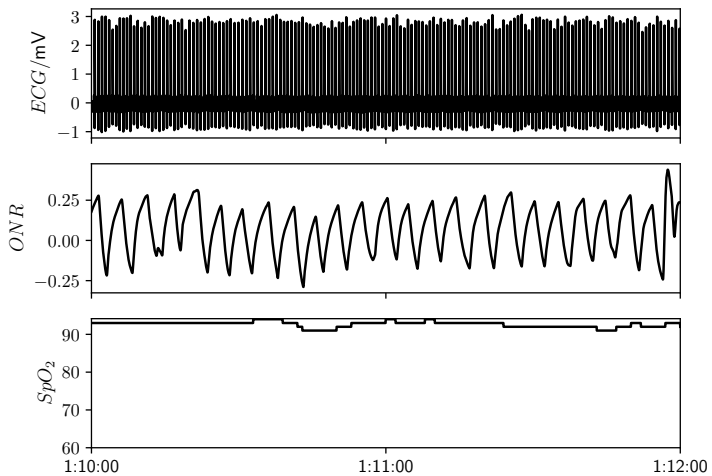
Built in Testing:

Investigated Class Decoding: Complexity is exponential in length. Shortcuts I've tried perform badly.

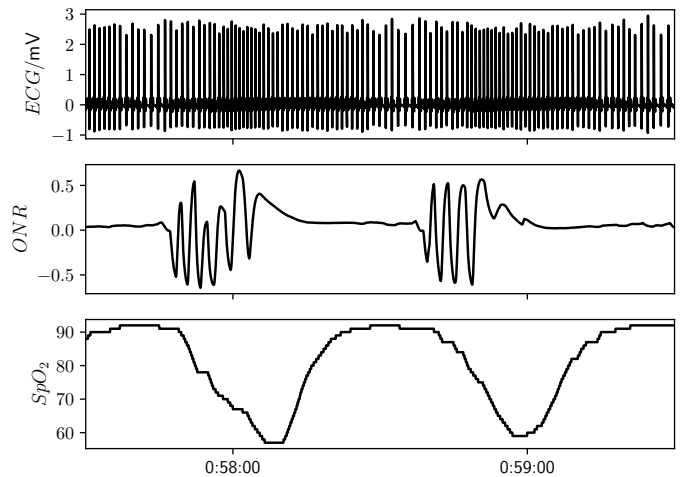
Estimating Class to Detect Apnea

Computers in Cardiology 2000 Challenge: Classify EKG

Normal

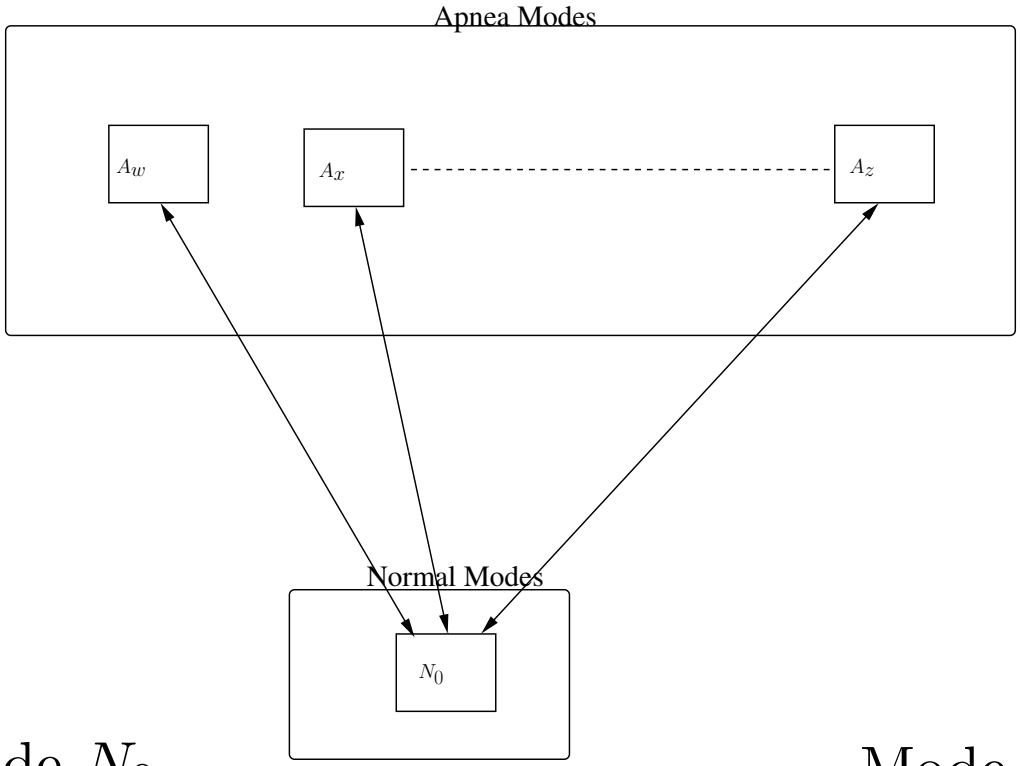


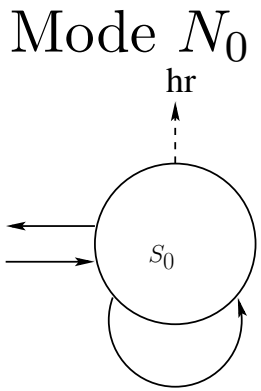
Apnea

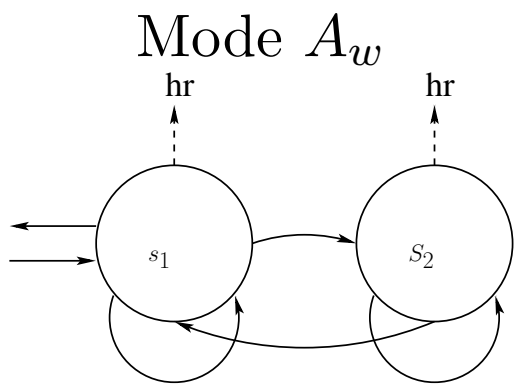


Multiple Apnea Models

Happy sleepers are all alike; every unhappy sleeper is unhappy in its own way.







Estimating Sequences

Viterbi Decoding for States: Computation is linear in T.

$$\hat{s}[0:T] \equiv \underset{\text{state}[0:T]}{\text{argmax}}\, P\left(\text{state}[0:T] \mid \text{heart rate}[0:T]\right)$$

Class Sequence from State Sequence: More apnea modes $\rightarrow$ Less apnea estimated.

$$\hat{c}[t] = C\left(\hat{s}[t]\right)$$

Probability gets spread over many modes.

Max A-posteriori Prob Class Sequence: Exponential in T.

$$\hat{c}[0:T] \equiv \underset{\text{class}[0:T]}{\text{argmax}}\, P\left(\text{class}[0:T] \mid \text{heart rate}[0:T]\right)$$

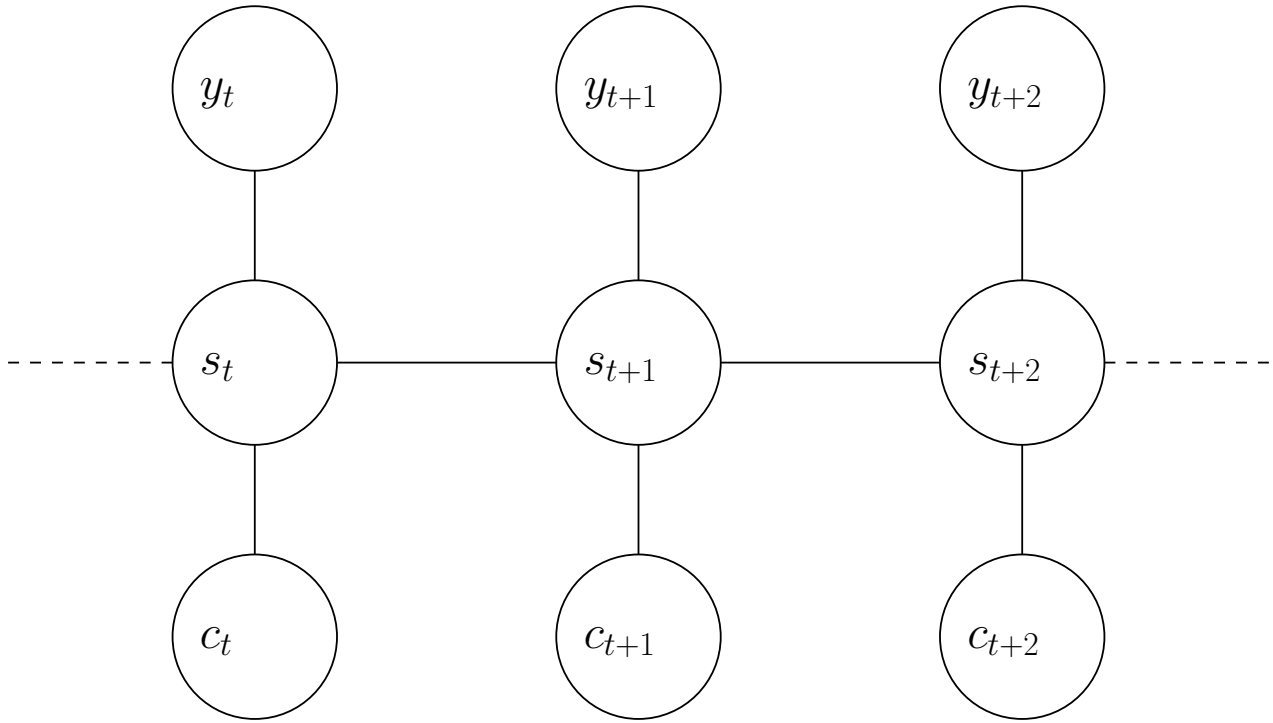
Sequence of MAP Classes: Linear in T, but can yield impossible sequences.

$$\hat{c}[t] = \max_{\text{class}} P\left(\text{class}[t] \mid \text{heart rate}[0:T]\right)$$

Appropriate for 2-class apnea problem.

Graphical Representation of Conditional Independence

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



A bad subsequence of classes may later become a portion of the *best* class sequence. The number of subsequences to calculate and store is exponential in length T.

Conclusions

- Structure work and code for clarity.
- Collaborate and seek peer review.
- Consider advice about good practices.
- Focus on objectives before algorithms.
- Estimating class sequences could be important and interesting.