Bioinformatic Reading

=== Web Pages
Worst web page design I've ever seen: http://www.syntheticsapien.com/MarkovChain/index.html

==========
Computational Methodology in Molecular Biology: 0 444 82875 (80303) 3

Gene finding is about tagging DNA sub-sequences with labels like:
* intergenic DNA
* exon
* intron
* etc.

(p.23)

Dynamic Programming: recursively solving smaller problems and scoring paths (reducing the problem to the solution being the highest scored path from one problem to the next) in order to find the most likely path of states that match the given observable states. This book uses a pretty awesome little cake-baking example, using different scores and ratios between ingredient amounts.

Markov models ≠ Markov chains (but they are associated) (p.24)

A profile HMM uses two chains; one for the hidden states and one for the observable emitted symbols. (Eddy's paper mentions this.)

A Markov Chain is a sequence of events where the outcome (emission) of each state depends on a fixed number of outcomes before it. (It has a fixed-sized memory.) E.g., a first-order Markov Chain has the outcome depend on just the one state before. (p.24)

There doesn't seem to be a standard for where orders start (0 or 1), but most papers define a 1st order model as having a 1 state memory.

Markov Chains can answer questions like "what's the probability that we're looking at a start codon, given that the sequence is CTG" (p.25)

Note that there's a full, simple example on p.25 that also mentions how P(M) can be safely ignored during computation.

Motifs are described with regular expressions. (http://en.wikipedia.org/wiki/Sequence_motif)

Gene hunting would be ideal if it could be done with regular expressions, because it has a "simple 'grammatical structure', like gene finding" (p.45). But the problem is that two proteins that do the same thing are coded differently, and things seem to change in DNA in other ways. (p.46)

"The most popular use of the HMM in molecular biology is as a "probabilistic profile" of a protein family, which is called a profile HMM. (p.45)" (Implying that there's lots of different kinds of HMMs. (Later read that a HMM used to identify profiles is just called a "profile HMM")) This way, a protein (or DNA) family profile can be described so that we can use it to find other members of the family in collections of DNA. One of the

best things about profile HMMs is that they treat gaps in a systematic way. (p.45)

Apart from a profile, we can also use weight matrix methods. (Without gaps or insertions, then all the transition probabilities would be 1(assuming we're matching a left-right pattern), so they can be ignored – with only the emission probabilities left, we can implement the model with just a weight matrix.) This kind of setup results in a simple HMM.

Profiles emerge from multiple alignment.

From page 8 of the pdf:

Developing Profiles; turning alignments

	Sequence	Probability ×100	Log odds
Consensus	A C A C - - A T C	4.7	6.7
Original	A C A - - - A T G	3.3	4.9
sequences	T C A A C T A T C	0.0075	3.0
	A C A C - - A G C	1.2	5.3
	A G A - - - A T C	3.3	4.9
	A C C G - - A T C	0.59	4.6
Exceptional	T G C T - - A G G	0.0023	-0.97

Table 4.1

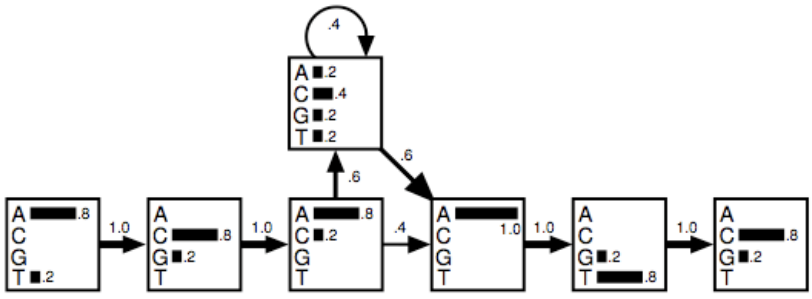

Figure 4.1

into this:

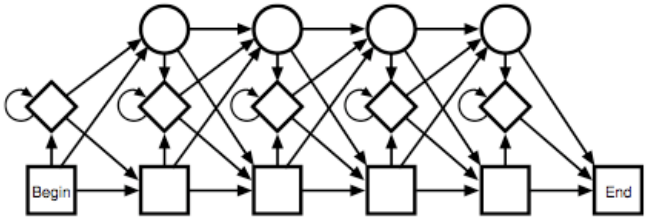

Fig 4.3

Bottom row: main states (model the columns of the alignment)

Emission Distribution: frequency of symbols

Transition Distribution:

prob(begin insertion; moving to the second row) = count(alignments that inserted)/count (alignments)

prob(not making an insertion; continuing along the bottom row) = 1 - prob(begin insertion)

Second row (diamonds): insert states (highly variable regions)

Distribution: Either overall distribution of symbols or a detailed distribution:

Emission probabilities: prob(α) = count(α)/count(all symbols in insertion region)

Transition probabilities:

prob(continue insertion) = count(continued insertion transitions)/count(transitions in insertion region after the first insertion)

prob(exiting transition) = count(non-continued insertion transitions)/count(transitions in insertion region after the first insertion)

= (count(transitions in insertion region after the first insertion) - count(continued insertion transitions))/count(transitions in insertion region after the first insertion)

Third (top) row: delete states

Enable skipping over insertion states (to let non-inserting alignments to "keep up" with inserting alignments).

Building a profile, given a multiple alignment:

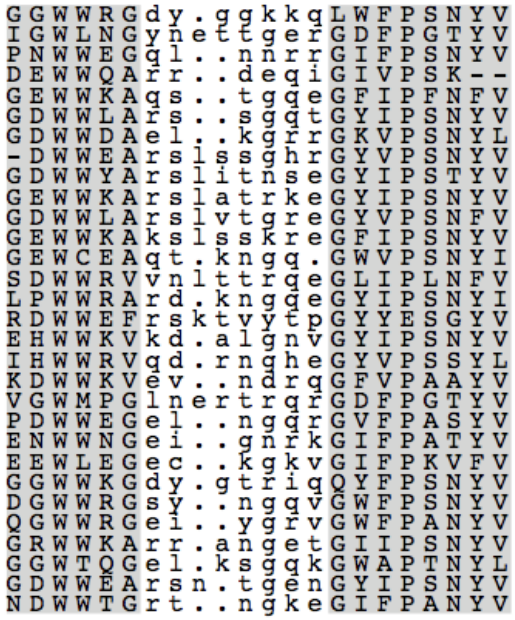

Figure 4.4

The shaded columns are the most conserved, so they become the main states with emission probabilities equal to the amino acid (symbol) count frequencies.

Transition and emission probabilities: see above rules

HMM's usefulness really comes out to shine when we deal with probabilities when we match our regexes. What's a likely match, given that we know the probabilities of a state being a certain way? (See awesome finite state machine diagram on p.47)

On p.47, they're talking about matching DNA motifs, but they're also known as alignments. (Putting two sequences that are assumed to share motifs together (graphically, on top of one another) is called "alignment".) (Not really; pairwise alignment is where two sequences are aligned so that they are the most similar. Searching a database with a profile uses the same idea (and similar dynamic programming techniques) because "if we can find a path through the model where the new sequence fits well in some sense, then we can score the sequence as before". (4.3.2))

What's log-odds about? (p.48) Is the only advantage of using them during scoring is that you can just add up the probabilities instead of having to do multiplication?

No; when only multiplying probabilities together, a shorter sequence (maybe it had some deletes) will have a probability that appears far different than a similar sequence, when it's actually not that different. Using log-odds as well as adding a pseudocount of 1 (or another value) to all counts makes the scores much more clear about what they're saying (i.e. how similar a sequence is to a model (in our case, a profile HMM)). Without this added pseudocount, a count of 0 can cause an entire sequence to appear improbable.

The log-odds score is generated by taking the probability of the sequence divided by the null model (one that treats the sequence as completely random – each emission has the same probability). Note that other null models can be used instead (e.g. the overall frequency probability).

(p.54) Viterbi <- dynamic programming algorithm used to match/align an unknown sequence to a model (HMM).

A_1 A_2 A_3 A_4 A_5
M_1 I_1 I_1 M_2 (delete * 3) M_5

A: sequence
I: insert
M: match
D: delete

(p.54) By using the log-odds score, databases can be searched for members of the same family.

The idea is that we want to match states to each observed symbol in the given sequence.
With a profile HMM, we can do this by aligning its states to the given sequence.

Using the notation above, this means that we'd be matching the top row to the bottom row.

Forward Algorithm: instead of looking for the best score, a sequence can be aligned with the model for *all* possible alignments and the scores from each summed up.

A way of both producing a multiple alignment as well as determining unknown probabilities in a model:

(1) Start with a random model and a bunch of (hopefully similar) sequences.

(2) Align all the sequences to the model with the Viterbi algorithm

(3) From the alignment, find a probability matrix

(4) Re-align with the new probability matrix

(5) If the alignment is different, it means the probability matrix is still "maturing", so start again at (2) with the new matrix.

Note that instead of aligning with Viterbi, it's possible to use the Forward algorithm's summing idea. <-

when used this way, it's known as the *Forward-backward* algorithm, a.k.a. the Baum-Welch algorithm.

However, there are several problems with this:
* How do we choose the model length? "This determines the number of inserts in the final alignment." (4.4)
* "The iterative solution can converge to suboptimal solutions." (4.4)

(p.56) HMMs for Gene Finding

Gene structure has a grammatical structure:
genome := exon intron exon

HMMs that use non-conditional probability can only model regular grammars (i.e. recognizable by finite state automata)

Something that HMMs (using Markov chains of the first order (no look-behind/conditional probabilities)) can't accomplish is to recognize a context-sensitive grammar (usually accomplished with a push-down automata)
Looking into something that does HMM-ing with PDAs would be cool.

How long is a typical gene/gene structure?

p.57 4.1 Signal Sensors

In order to recognize a repeating pattern of known length in a sequence, higher-order (2nd+) models should be used. [Is this a General HMM / semi-HMM?]

p.58 4th order state models (inhomogeneous Markov chain => "a subclass of HMMs"). Ordinary (0th or 1st order) has no conditional probability.

4.3 Combining the Models
By setting up the smaller-focused models in tandem, a larger pattern can be found, just like any other regular expression.

HMMs <=> regex with probability

Two recent methods use so-called generalized HMMs. Genie [37, 38, 39] combines neural networks into an HMM-like model, whereas GENSCAN [40] is more similar to HMMgene, but uses a different model type for splice site. Also, the generalized HMM can explicitly use exon length distributions, which is not possible in a standard HMM. Web pointers to gene finding can be found at http://www.cbs.dtu.dk/krogh/genefinding.html.

==========

Bioinformatics Second Editions Baldi and Brunak (Black book)
See HMM definition on p. 166

Big three questions:

Likelihood:
Given a sequence, how probable is it?
Solved by calculating log-odds-ratios and adding

Decoding:
Given a sequence, what's the most probable path (transitions and emissions)?
Solved by Viterbi or Forward

Learning (alignment and construction):
Given some related sequences, what are their transition and emission parameters (probabilities?) (i.e, how do you build a profile HMM?)
Solved by Viterbi or Baum-Welch (expectation maximization)

p.179 Gradient descent and generalized expectation maximization (GEM) mentioned.

==========

Computational Molecular Biology: An Algorithmic Approach Pavel A. Pevzner (Red/Yellow Russian book)

p.148 profile HMMs
p.144 8.7 "Fair-bet casino"
8.8 HMM

==========

Bioinformatics Computing Bryan Bergeron

==========

An Introduction to Bioinformatics Algorithms Neil C. Jones and Pavel Pevzner

==========

What is a hidden Markov model?
Sean Eddy - Washington University School of Medicine

Concise summary of HMM used in labelling

==========

Profile hidden Markov models, Eddy 14 (9) 755, Bioinformatics

The name 'hidden Markov model' comes from the fact that the state sequence is a first-order Markov chain, but only the symbol sequence is directly observed.

Once an HMM is drawn, regardless of its complexity, the same standard dynamic programming algorithms can be used for aligning and scoring sequences with the model (Durbin et al., 1998). These algorithms, called Forward (for scoring) and Viterbi (for alignment), have a worst-case algorithmic complexity of O(NM2) in time and O(NM) in space for a sequence of length N and an HMM of M states. For profile HMMs that have a constant number of state transitions per state rather than the vector of M transitions per state in fully connected HMMs, both algorithms run in O(NM) time and O(NM) space—not coincidentally, identical to other sequence alignment dynamic programming algo-

rithms. For a modest (constant) penalty in time, very memory-efficient O(M) and O(M1.5) versions of Viterbi and Forward can also be implemented (Hughey and Krogh, 1996; Tarnas and Hughey, 1998).

p.3 "Profile HMMs are strongly linear, left-right models, unlike the general HMM case" (Does that imply that general HMMs are ergodic (cyclic, connected)?)

==========

Other Notes:

Table of contents with subheading abstracts (1 or 2 sentences) of each part

e.g.

Likelihood

Given a sequence, how probable is it? (Solved with log-odds-ratio)

Check into BioRuby and ask on their mailing list if I can edit (fix up) their wiki. http://bioruby.org/

Check out HMMER: http://hmmer.janelia.org/

See also: http://selab.janelia.org/publications/cupbook.html

McGill Centre for Bioinformatics: http://www.mcb.mcgill.ca/

Grad program in Bioinformatics: http://bioinformatics.ubc.ca/bgp/

CREAD: http://rulai.cshl.edu/cread/

What does "consensus" keep referring to? http://en.wikipedia.org/wiki/Consensus_sequence
Sequence alignment: http://en.wikipedia.org/wiki/Sequence_alignment

If a sequence motif [http://en.wikipedia.org/wiki/Sequence_motif] (nucleotide pattern that's widespread/conjectured to have biological (functional / structural) significance) is highly conserved (kept the same) across libraries / lineages.

Homology/Orthology: http://en.wikipedia.org/wiki/Orthology

BAC stuff: http://www.scq.ubc.ca/?p=266

Meetings with Nelly:

Thu Oct 12 15:01:03 EDT 2006:

A Markov Chain is the same as a Markov Model. (No; see Eddy's paper or line 18)

A Markov Chain/Model of order n is a non-deterministic finite state machine (which in turn is a Regular Expression) with the n being the number of states "backwards" the probability of outcome depends on.

A HMM differs from a Markov Model/Chain in the way that the transition path is unknown (so as soon as there's a choice between at least two paths, you don't know in which state the machine is).

Notes from black notebook

A Hidden Markov Model is essentially a regular expression with probability.
(Notes were later just stuck in the right sections above.)

==========

From Sean Eddy's userguide to HMMER

HMM profiles are "statistical descriptions of the consensus of a multiple sequence alignment. "
Position specific scores for symbols and position-specific penalties for opening and extending an insertion or deletion.

BLAST, FASTA, or Smith/Waterman (all traditional pairwise alignment algorithms) use position-independent scoring parameters.

For building profiles from unaligned sequences, look at CLUSTALW or hmmt from HMMER 1:
fpt.genetics.wustl.edu/pub/eddy/hmmer/hemmer-1.8.4.tar.Z