Figure 5.4 A hidden Markov model derived from the small alignment shown in Figure 5.3 using Laplace’s rule. Emission probabilities are shown as bars opposite the different amino acids for each match state, and transition probabilities are indicated by the thickness of the lines. The $I \rightarrow I$ transition probabilities times 100 are shown in the insert states. (Figure generated automatically using the SAM package.)
Uses of probabilistic sequence models/HMMs

- Segmentation
- Multiple alignment using profile HMMs
- Prediction of sequence function (gene family models)
- ** Gene finding **
Review

• **Gene**
  – A sequence of nucleotides that are translated into proteins

• **Gene prediction**
  – Given the model of a gene above, determine the beginning and end positions of all genes in a genome.
Central dogma of molecular biology

• Information is stored in DNA

• Genome is processed into messenger RNA molecules (transcription)

• RNA molecules are processed to form proteins (translation)
Open reading frames

• Generally defined as regions in genes between a start (ATG) and stop (eg. TGA) codon.

• Size is a multiple of 3

• Six possibilities given any DNA sequence
  – 0 offset, + strand; 1 offset, + strand, 2 offset, + strand
  – 0 offset, - strand; 1 offset, - strand, 2 offset, - strand
Long ORFs

• At random, we’d expect a stop codon every 64 nucleotides.

• Many bacteria genes are much longer than this.

• These can be used to train a statistical model.
But wait!

- Finding ORFs alone isn’t sufficient. *E. coli* has up to 6500 ORFs but only 1100 “real” genes

- Complicated by the fact ORFs can overlap on different strands and be correct (figure in class)
Homework to the rescue (or at least part of the way there)

• We know codons are triplets.

• We also know from last class codons are not equal.

• A simple gene finder is no different from your Markov assignment except now we’ll need to compute two probabilities
Simple gene finding

• Score every ORF using all seven models

• Normalize the scores such that they represent the probability of coding

• Choose the highest
One fix is to use a third order model
IMM

- Interpolated Markov Models (IMMs) overcome the training problem by generating models of variable order.

- Bias is put towards higher models if and only if there is enough training data.

- Achieved via a linear combination of probabilities based on varied lengths.
Simple linear interpolation

\[ P_{\text{IMM}}(x_i \mid x_{i-n}, \ldots, x_{i-1}) = \lambda_0 P(x_i) + \lambda_1 P(x_i \mid x_{i-1}) + \ldots + \lambda_n P(x_i \mid x_{i-n}, \ldots, x_{i-1}) \]

• where \( \sum_i \lambda_i = 1 \)
GLIMMER

• Addressed the fundamental training problem of Markov models

• As mentioned before, we want the highest order model possible

• However, a $k^{th}$ order model requires $4^{k+1}$ probabilities to be estimated
  – Impractical for small genomes
Only 1 of the 37 genes missed by GLIMMER was found by the 5th order model

On the other hand, it found 107 more true genes

<table>
<thead>
<tr>
<th>Model</th>
<th>Genes found</th>
<th>Genes missed</th>
<th>Additional genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLIMMER IMM</td>
<td>1680 (97.8%)</td>
<td>37</td>
<td>209</td>
</tr>
<tr>
<td>5th-Order Markov</td>
<td>1574 (91.7%)</td>
<td>143</td>
<td>104</td>
</tr>
</tbody>
</table>

The first column indicates how many of the 1717 annotated genes in *H.influenzae* were found by each algorithm. The ‘additional genes’ column shows how many extra genes, not included in the 1717 annotated entries, were called genes by each method.
Pipmaker

- [http://www.bx.psu.edu/miller_lab/](http://www.bx.psu.edu/miller_lab/)

- Visualization of BLASTZ alignments

- Although pips are compact and informative, they do not show alignment information for the second sequence.
  - Dotplots are used to see relevant differences
Applications of genome alignment

• Comparing different genome assemblies
• Locating genome duplications and conserved segments
• Gene finding through comparative genomics
• Analyzing pathogenic bacteria against their harmless close relatives
Two different most parsimonious scenarios that transform the order of the 11 synteny blocks on the mouse X chromosome into the order on the human X chromosome.

Pevzner P., Tesler G. PNAS 2003;100:7672-7677
Homology map

We multiply align these blocks together
1. find multi-MUMs in sequences
2. calculate pairwise distances
3. run neighbor-joining to get guide tree;
Eukaryote-to-eukaryote gene transfer events revealed by the genome sequence of the wine yeast Saccharomyces cerevisiae EC1118

Maïté Novo¹, Frédéric Bigey¹, Emmanuelle Beyne¹, Virginie Galeote¹, Frédéric Gavory², Sandrine Mallet³, Brigitte Cambon¹, Jean-Luc Legras⁴, Patrick Wincker², Serge Casaregola³ and Sylvie Dequin¹,²

*Author Affiliations

Edited by Jasper Rine, University of California, Berkeley, CA, and approved August 3, 2009

¹²M.N. and F.B. contributed equally to this work. (received for review May 5, 2009)
Methods

• “Basic alignments were routinely performed with BLAST (10). Genomes were aligned, using MUMmer 3.0 (11).”

• “Gene prediction for the EC1118 genome was carried out with GlimmerHMM (12), trained on S288c ORFs from the Saccharomyces Genome Database (version 2008-10-10).”
Chromosomal distribution of the 3 unique EC1118 regions.

Region A
- VIII
- VI
- 38 Kb

Region B
- XIV
- 17 Kb

Region C
- XV
- 65 Kb

EC1118 unique chromosomal regions

Novo M et al. PNAS 2009;106:16333-16338
Suffix arrays require even less space than a suffix tree

- Very simply, it is a sorted list of suffixes
  - Example in the Aluru chapter posted as a resource
FIGURE 1.1: Suffix tree, suffix array and Lcp array of the string *mississippi*. The suffix links in the tree are given by $x \rightarrow z \rightarrow y \rightarrow u \rightarrow r$, $v \rightarrow r$, and $w \rightarrow r$. 
Linear time of suffix arrays

• There were three papers in 2002 that solved the old problem of constructing suffix arrays in linear time.

• These were:
  – Ko and Aluru – very interesting, but hard to understand
  – Kim et al. – was based on older parallel suffix tree algorithms
  – Karakkanen and Sanders is the simplest and most elegant.