Gene Finding & Review

Michael Schatz

Bioinformatics Lecture 4
Quantitative Biology 2010

DNA Sequencing

- Illumina
  - http://www.youtube.com/watch?v=77r5p8IBwJk

- Pacific Biosciences
  - http://www.pacificbiosciences.com/video_lg.html

Outline

1. 'Semantic' Sequence Analysis
    1. Prokaryotic Gene Finding
    2. Eukaryotic Gene Finding
    3. Phylogenetic Trees (Supplemental)

2. Review
    1. Exact Matching
    2. Sequence Alignment
    3. Genome Assembly
    4. Gene Finding

Gene Prediction: Computational Challenge

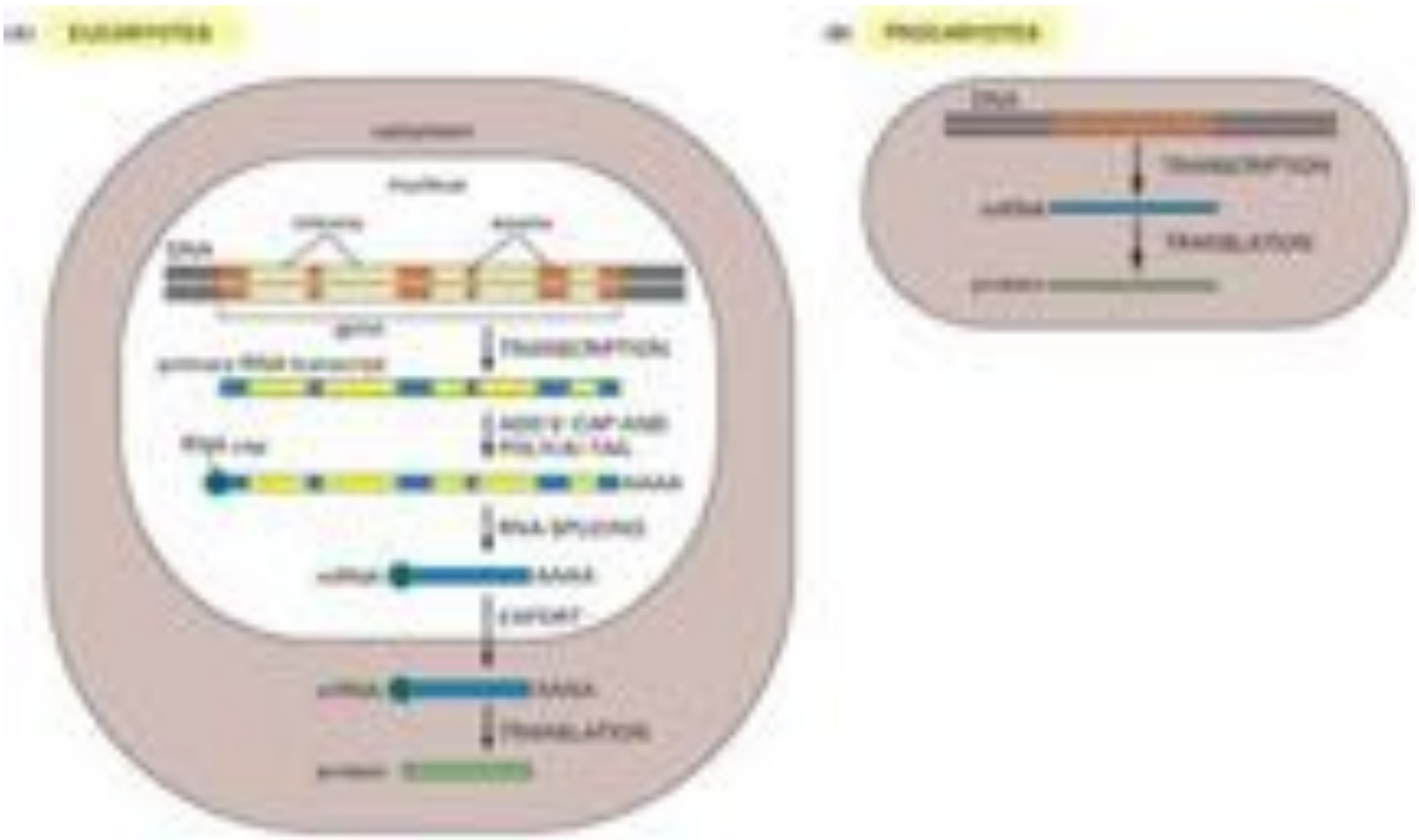
aatgcatgcggctatgctaatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaa
tgcatgcggctatgcaagctgggatccgatgactatgctaagctgggatccgatgacaatgcatgcggctatgc
taatgaatggtcttgggatttaccttggaatgctaagctgggatccgatgacaatgcatgcggctatgctaatga
atggtcttgggatttaccttggaatatgctaatgcatgcggctatgctaagctgggatccgatgacaatgcatgc
ggctatgctaatgcatgcggctatgcaagctgggatccgatgactatgctaagctgcggctatgctaatgcatg
cggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatc
ctgcggctatgctaatgaatggtcttgggatttaccttggaatgctaagctgggatccgatgacaatgcatgcgg
ctatgctaatgaatggtcttgggatttaccttggaatatgctaatgcatgcggctatgctaagctgggaatgcatg
cggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatc
cgatgactatgctaagctgcggctatgctaatgcatgcggctatgctaagctcatgcggctatgctaagctggg
aatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaag
ctgggatccgatgactatgctaagctgcggctatgctaatgcatgcggctatgctaagctcggctatgctaatga
atggtcttgggatttaccttggaatgctaagctgggatccgatgacaatgcatgcggctatgctaatgaatggtc
ttgggatttaccttggaatatgctaatgcatgcggctatgctaagctgggaatgcatgcggctatgctaagctgg
gatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatccgatgactatgctaagc
tgcggctatgctaatgcatgcggctatgctaagctcatgcgg

Gene Prediction: Computational Challenge

aatgcatgcggctatgctaatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaa
tgcatgcggctatgcaagctgggatccgatgactatgctaagctgggatccgatgacaatgcatgcggctatgc
taatgaatggtcttgggatttaccttggaatgctaagctgggatccgatgacaatgcatgcggctatgctaatga
atggtcttgggatttaccttggaatatgctaatgcatgcggctatgctaagctgggatccgatgacaatgcatgc
ggctatgctaatgcatgcggctatgcaagctgggatccgatgactatgctaagctgcggctatgctaatgcatg
cggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatc
ctgcggctatgctaatgaatggtcttgggatttaccttggaatgctaagctgggatccgatgacaatgcatgcgg
ctatgctaatgaatggtcttggga cggctatgctaagctgggaatgcatg
cggctatgctaagctgggatccg gcatgcggctatgcaagctgggatc
cgatgactatgctaagctgcggctatgctaatgcatgcggctatgctaagctcatgcggctatgctaagctggg
aatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaag
ctgggatccgatgactatgctaagctgcggctatgctaatgcatgcggctatgctaagctcggctatgctaatga
atggtcttgggatttaccttggaatgctaagctgggatccgatgacaatgcatgcggctatgctaatgaatggtc
ttgggatttaccttggaatatgctaatgcatgcggctatgctaagctgggaatgcatgcggctatgctaagctgg
gatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatccgatgactatgctaagc
tgcggctatgctaatgcatgcggctatgctaagctcatgcgg

Gene!

Genes to Proteins



Bacterial Gene Finding and Glimmer (also Archaeal and viral gene finding)

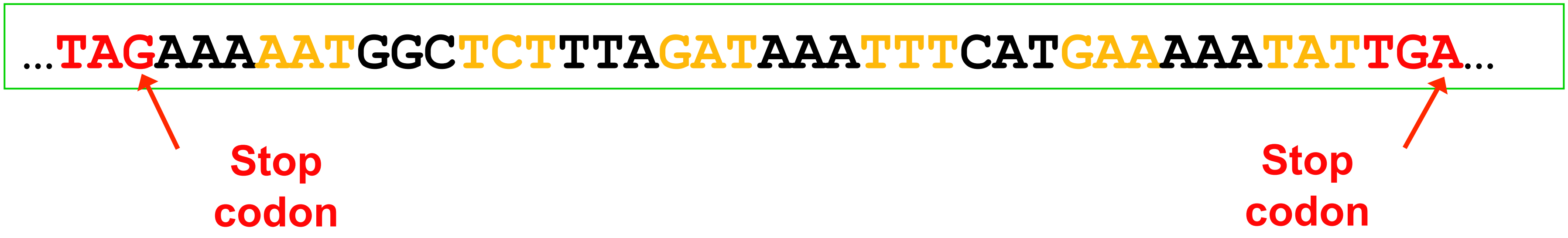
Arthur L. Delcher and Steven Salzberg
Center for Bioinformatics and Computational Biology
University of Maryland

Outline

- A (very) brief overview of microbial gene-finding
- Glimmer1 & 2
  - Interpolated Markov Model (IMM)
  - Interpolated Context Model (ICM)
- Glimmer3
  - Reducing false positives
  - Improving coding initiation site predictions
  - Running Glimmer3

Step One

- Find open reading frames (ORFs).



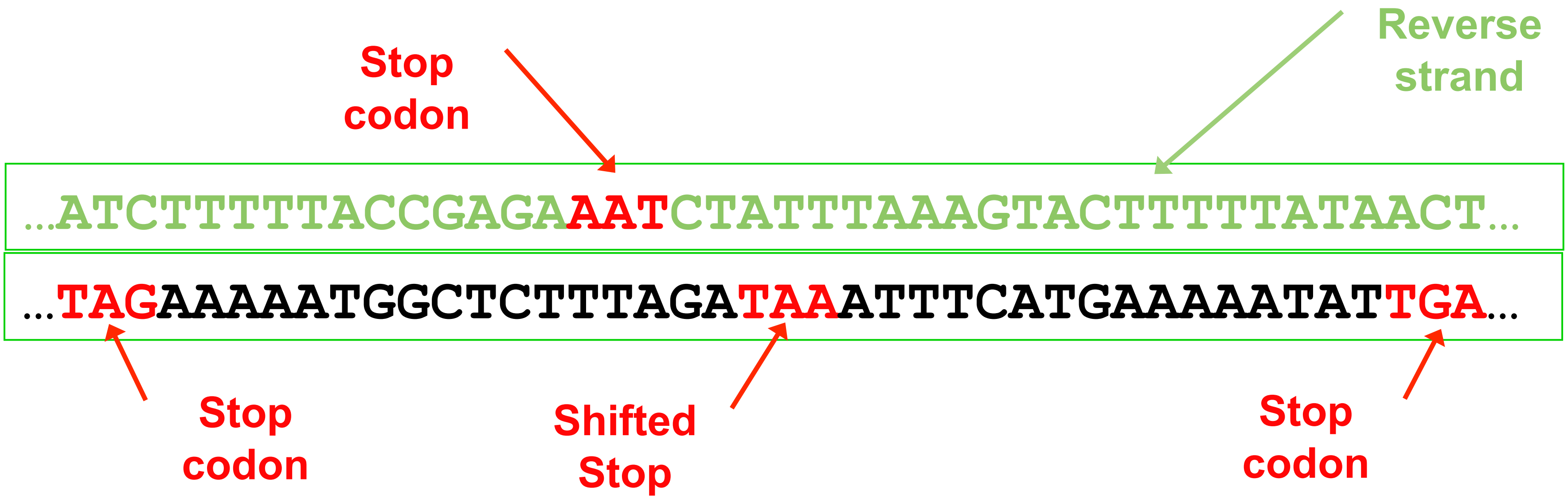
...TAGAAAAATGGCTCTTTAGATAAATTTCATGAAAAATATTGA...

Stop codon

Stop codon

Step One

- Find open reading frames (ORFs).



Stop codon

Reverse strand

...ATCTTTTTACCGAGA**AAT**CTATTTAAAGTACTTTTTATAACT...

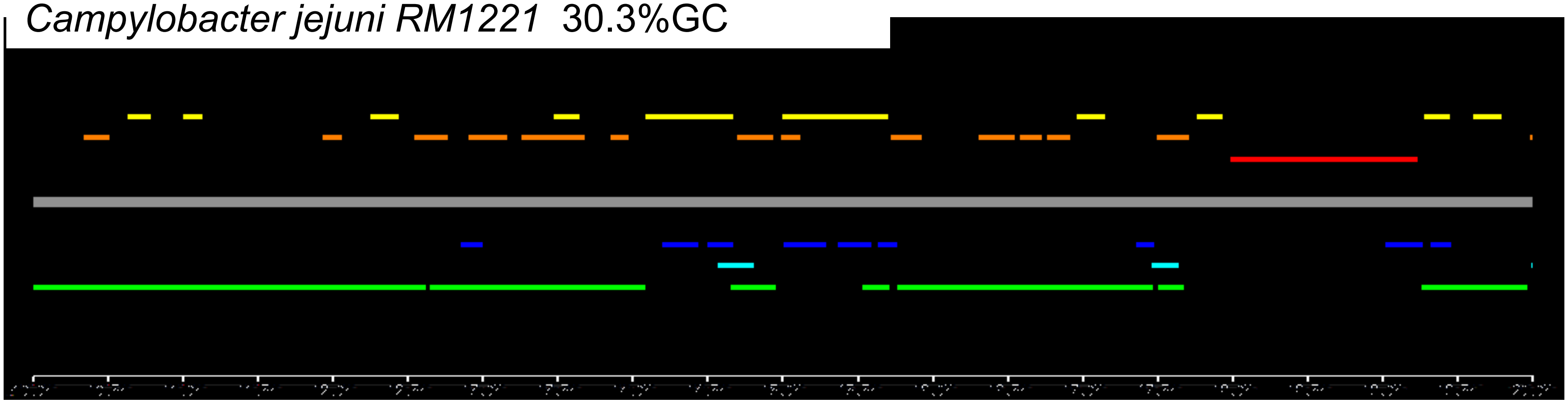
...**TAG**AAAAATGGCTCTTTAGA**TAA**ATTTCATGAAAAATAT**TGA**...

Stop codon

Shifted Stop

Stop codon

- But ORFs generally overlap …



Campylobacter jejuni RM1221 30.3%GC

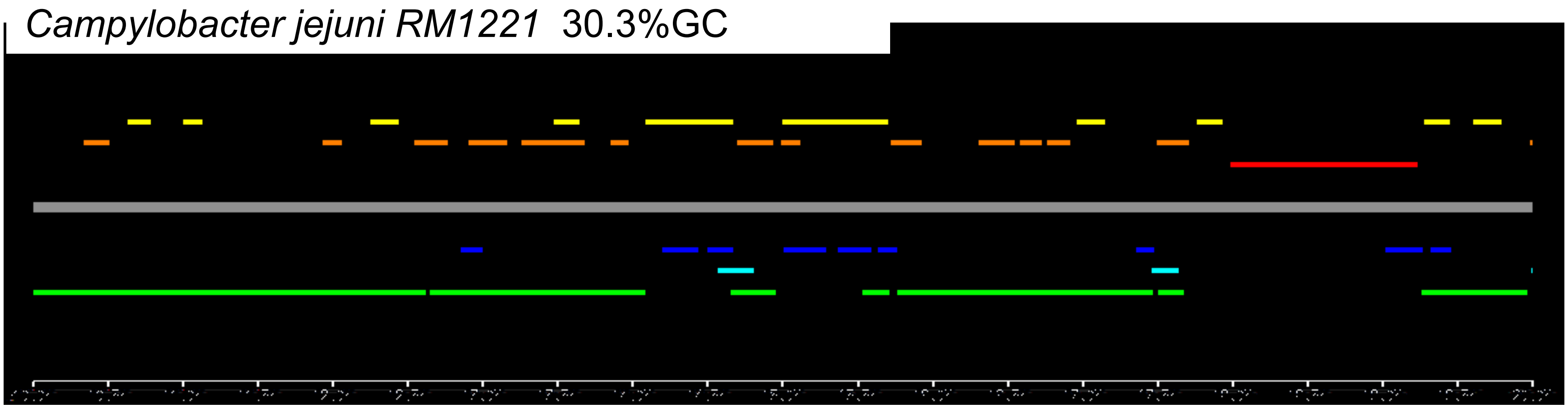
All ORFs on both strands shown

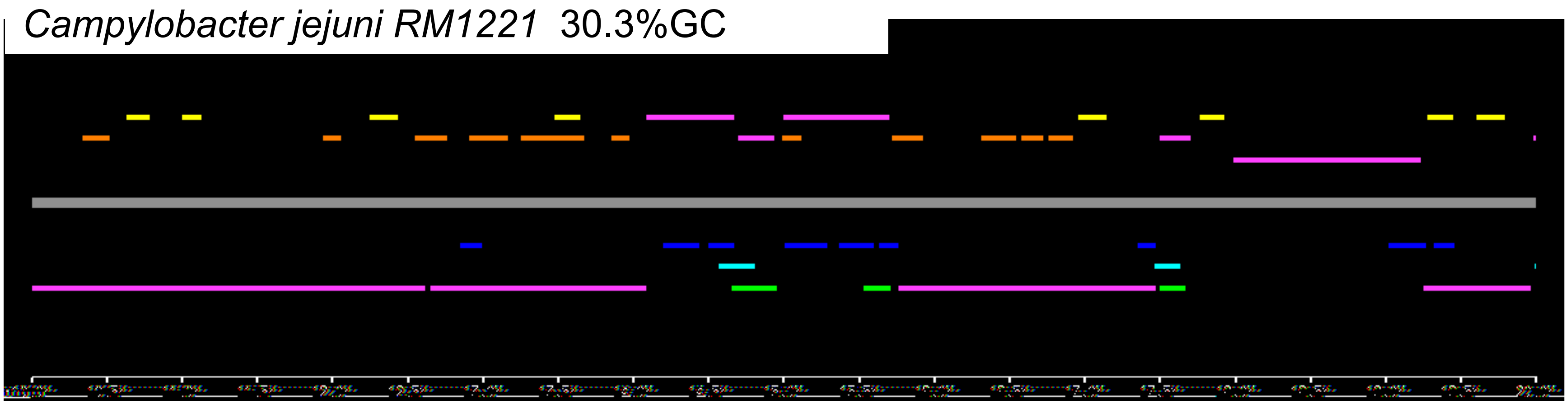
- color indicates reading frame

Longest ORFs likely to be protein-coding genes

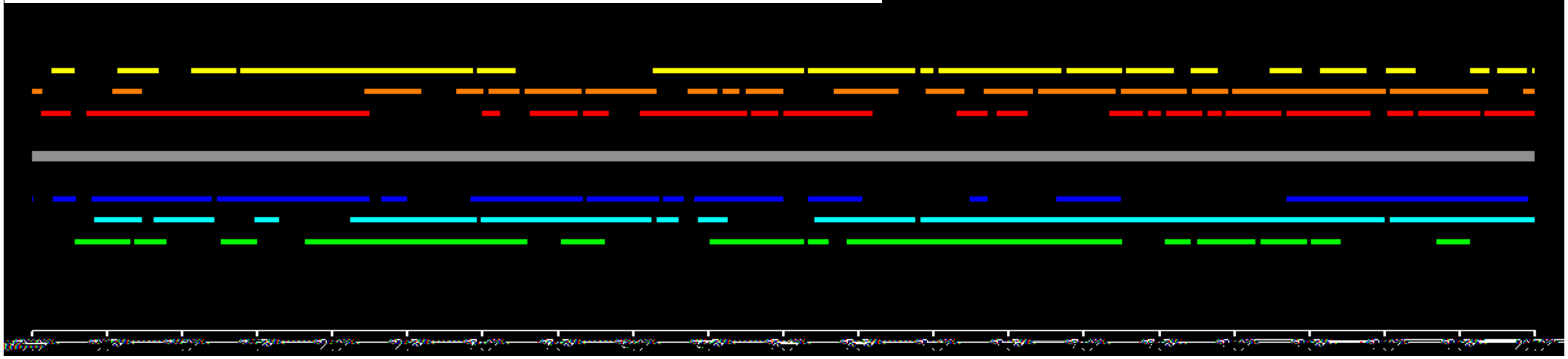
Note the low GC content

All genes are ORFs but not all ORFs are genes


Campylobacter jejuni RM1221 30.3%GC

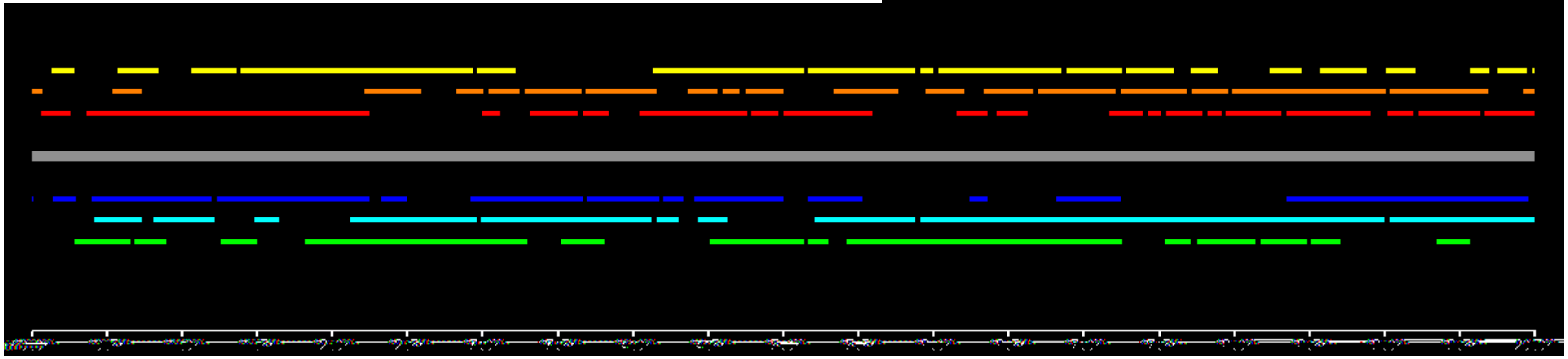

Campylobacter jejuni RM1221 30.3%GC

Mycobacterium smegmatis MC2 67.4%GC

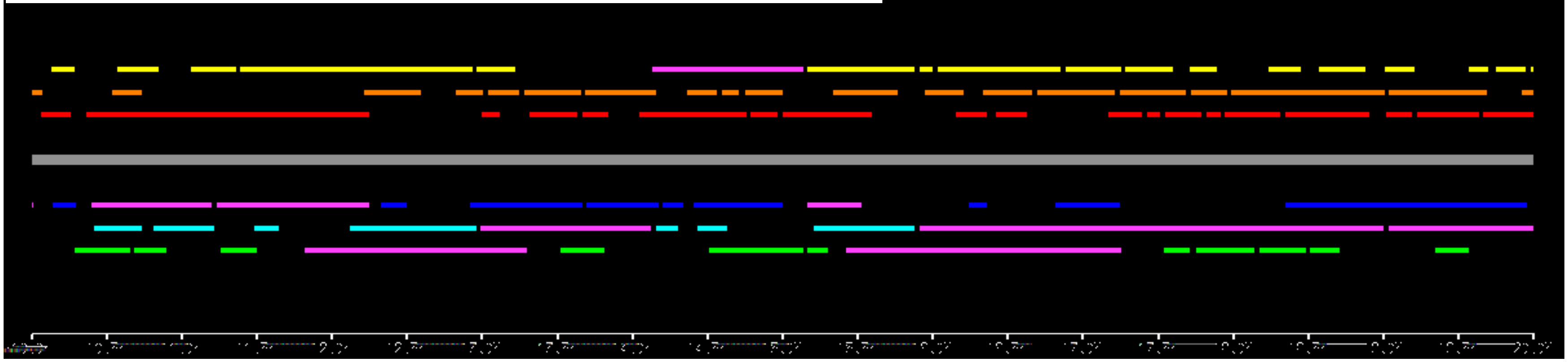


Note what happens in a high-GC genome

Mycobacterium smegmatis MC2 67.4%GC



Mycobacterium smegmatis MC2 67.4%GC



The Problem

- Need to decide which orfs are genes.
  - Then figure out the coding start sites

- Can do homology searches but that won't find novel genes
  - Besides, there are errors in the databases

- Generally can assume that there are some known genes to use as training set.
  - Or just find the obvious ones

Probabilistic Methods

- Create models that have a probability of generating any given sequence.

- Train the models using examples of the types of sequences to generate.

- The "score" of an orf is the probability of the model generating it.
  - Can also use a negative model (*i.e.*, a model of non-orfs) and make the score be the ratio of the probabilities (*i.e.*, the odds) of the two models.
  - Use logs to avoid underflow

Fixed-Order Markov Models

- *k* th-order Markov model bases the probability of an event on the preceding *k* events.
- Example: With a 3rd-order model the probability of this sequence:

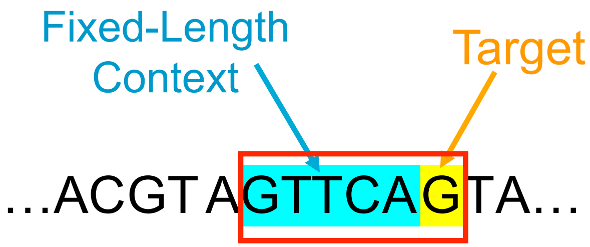
$$\cdots \mathrm{C}\overbrace{\mathrm{TAG}}^{\text{Context}}\overset{\text{Target}}{\mathrm{A}}\mathrm{T}\cdots$$

would be:

$$\cdots P(\mathrm{G}\,|\,\mathrm{CTA})\cdot P(\underset{\text{Target}}{\mathrm{A}}\,|\,\underbrace{\mathrm{TAG}}_{\text{Context}})\cdot P(\mathrm{T}\,|\,\mathrm{AGA})\cdots$$

Fixed-Order Markov Models

- Advantages:
  - Easy to train. Count frequencies of (*k*+1)-mers in training data.
  - Easy to compute probability of sequence.
- Disadvantages:
  - Many (*k*+1)-mers may be undersampled in training data.
  - Models data as fixed-length chunks.



Interpolated Markov Models (IMM)

- Introduced in Glimmer 1.0

  Salzberg, Delcher, Kasif & White, *NAR* 26, 1998.

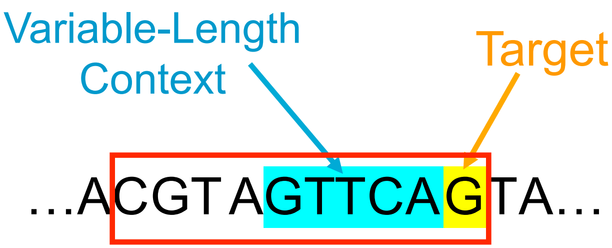
- Probability of the target position depends on a *variable* number of previous positions (sometimes 2 bases, sometimes 3, 4, etc.)

- How many is determined by the specific context.
  - *E.g.*, for context ggtta the next position might depend on previous 3 bases tta.
  - But for context catta all 5 bases might be used.

IMMs *vs* Fixed-Order Models

- Performance
  - IMM generally should do at least as well as a fixed-order model.
  - Some risk of overtraining.

- IMM result can be stored and used like a fixed-order model.
  - IMM will be somewhat slower to train and will use more memory.



Variable-Length Context

Target

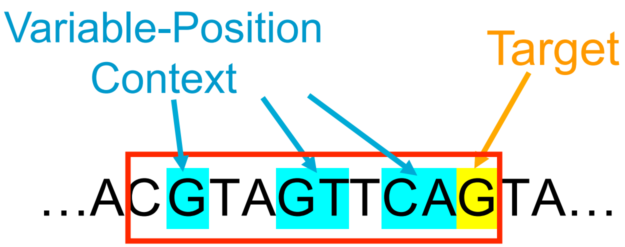
…ACGTAGTTCAGTA…

Interpolated Context Model (ICM)

- Introduced in Glimmer 2.0

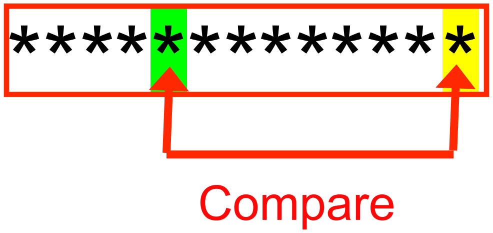
  Delcher, Harmon, *et al.*, *Nucl. Acids Res*. 27, 1999.

- Doesn’t require adjacent bases in the window preceding the target position.
- Choose set of positions that are most informative about the target position.



ICM

- For all windows compare distribution at each context position with target position

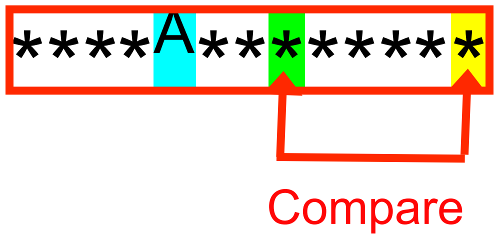


- Choose position with max mutual information

$$I(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}$$

ICM

- Continue for windows with fixed base at chosen positions



- Recurse until too few training windows
  - Result is a tree—depth is # of context positions used
- Then same interpolation as IMM, between node and parent in tree

Overlapping Orfs

- Glimmer1 & 2 used rules.
- For overlapping orfs *A* and *B*, the overlap region *AB* is scored separately:
  - If *AB* scores higher in *A*'s reading frame, and *A* is longer than *B*, then reject *B*.
  - If *AB* scores higher in *B*'s reading frame, and *B* is longer than *A*, then reject *A*.
  - Otherwise, output both *A* and *B* with a "suspicious" tag.
    - Also try to move start site to eliminate overlaps.
- Leads to high false-positive rate, especially in high-GC genomes.

Glimmer3

- Uses a dynamic programming algorithm to choose the highest-scoring set of orfs and start sites.
- Not quite an HMM
  - Allows small overlaps between genes
    - “small” is user-defined
  - Scores of genes are not necessarily probabilities.
  - Score includes component for likelihood of start site

Reverse Scoring

- In Glimmer3 orfs are scored from 3' end to 5' end, *i.e.*, from stop codon back toward start codon.
- Helps find the start site.
  - The start should appear near the peak of the cumulative score in this direction.
  - Keeps the context part of the model entirely in the coding portion of gene, which it was trained on.

Reverse Scoring

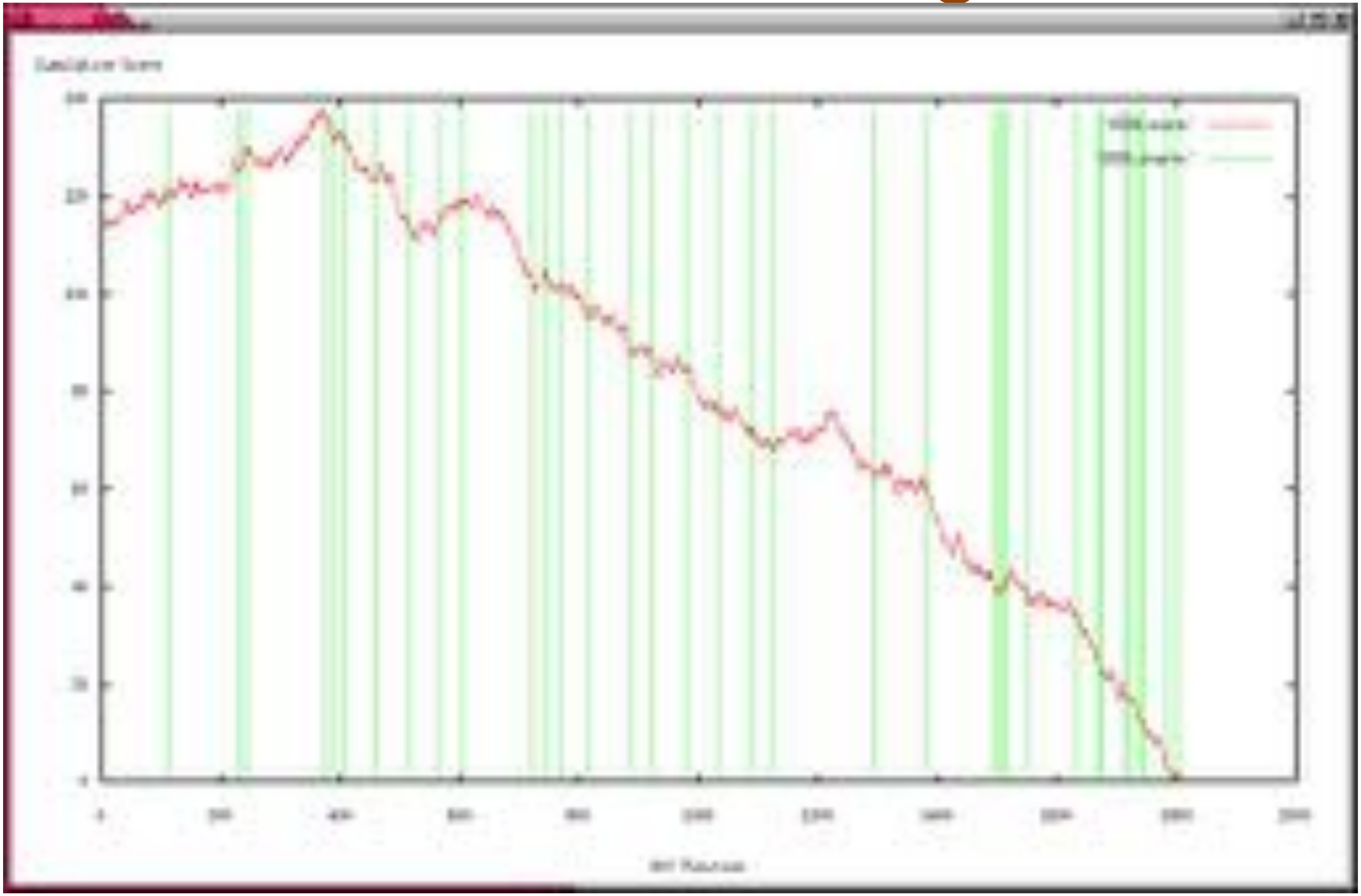


Table 6. Glimmer3 prediction accuracy compared to other gene-finding systems. Glimmer3 predictions are as in the preceding table and each entry is the Glimmer3 value minus the corresponding value for the other gene finder. GeneMark.hmm results were taken from the .GeneMarkHMM files downloaded from NCBI. EasyGene 1.2 results were downloaded from http://servers.binf.ku.dk/cgi-bin/easygene/search. GeneMarkS results were obtained from the server at http://exon.gatech.edu/GeneMark/genemarks.cgi. None of these systems had results for U. parvum, which uses a non-standard translation code. NA entries indicate values that were not available for download.

Genome		vs. GeneMark.hmm			vs. EasyGene 1.2			vs. GeneMarkS		
Organism	# Genes	3' Match	5' & 3'	Extra	3' Match	5' & 3'	Extra	3' Match	5' & 3'	Extra
A. fulgidus	1195	+1	31	91	+5	16	+108	4	-3	71
B. [illegible]	1132	-7	-48	-134	+33	45	+175	-1	-402	-142
B. subtilis	1476	+7	+394	+97	+98	16	+94	4	99	+194
C. jejuni	1592	-1	+31	+19	+80	+8	+302	-1	34	-29
C. perfringens	1944	-2	+177	124	-2	8	-21	-1	44	129
E. coli	3608	24	+18	+188	+60	+80	+407	24	29	+106
G. sulfurreducens	2330	-13	-225	+34	+5	-2	+48	-14	+41	-66
H. pylori	919	-1	-4	-34	+4	6	+140	-1	8	-41
P. fluorescens	4535	-17	-388	+59	NA	NA	NA	+17	-479	+46
R. solanacearum	2817	-7	+183	+223	+31	+48	+250	3	+140	+106
S. epidermidis	1438	+3	32	44	NA	NA	NA	+4	-244	14
T. pallidum	579	+1	-9	+94	+8	-6	+174	-1	-26	-90
Averages		+2	-89	+23	+19	2	+198	-2	-98	-29

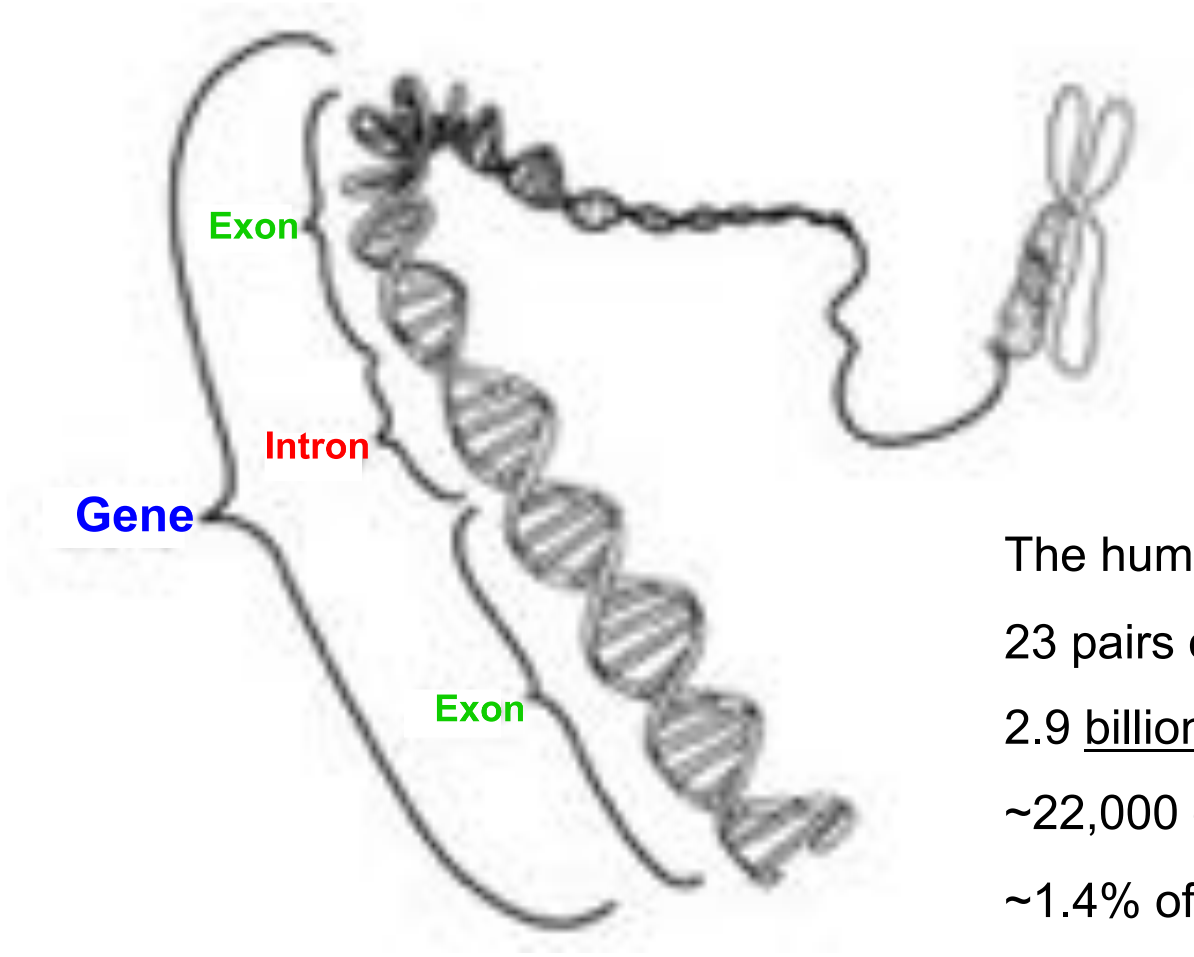
Overview of Eukaryotic Gene Prediction

CBB 231 / COMPSCI 261

W.H. Majoros

Duke University

Exons, Introns, and Genes



The human genome:

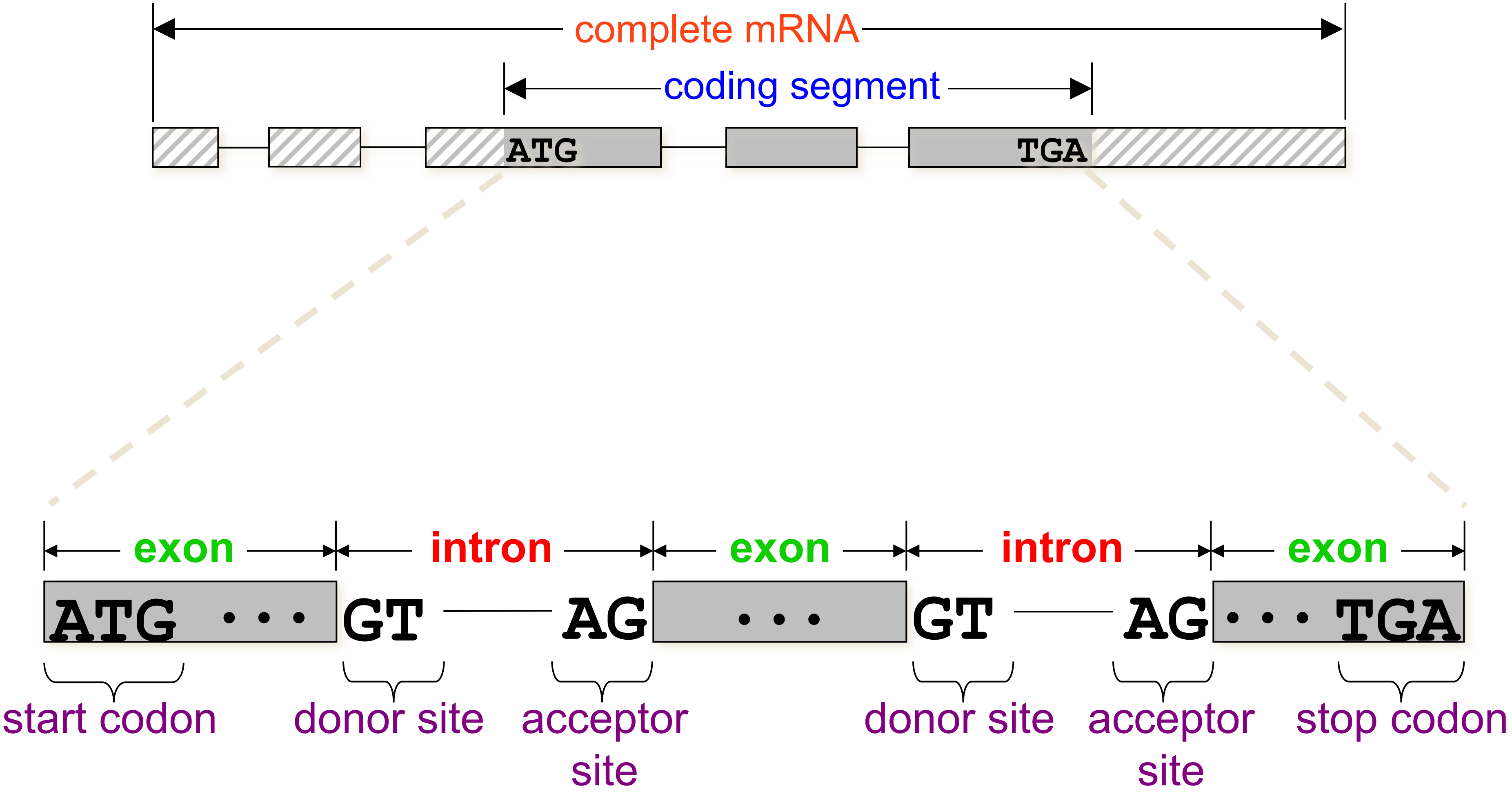
23 pairs of chromosomes

2.9 billion A’s, T’s, C’s, G’s

~22,000 genes (?)

~1.4% of genome is coding

Eukaryotic Gene Syntax



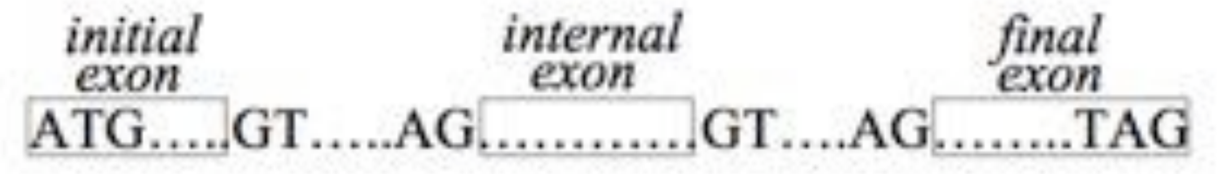
Regions of the gene outside of the CDS are called *UTR*'s (*untranslated regions*), and are mostly ignored by gene finders, though they are important for regulatory functions.

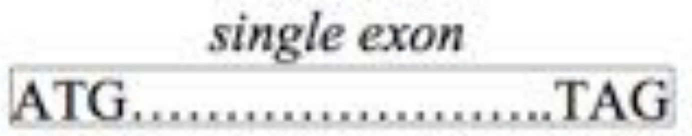
Duke UNIVERSITY

Types of Exons

Three types of exons are defined, for convenience:

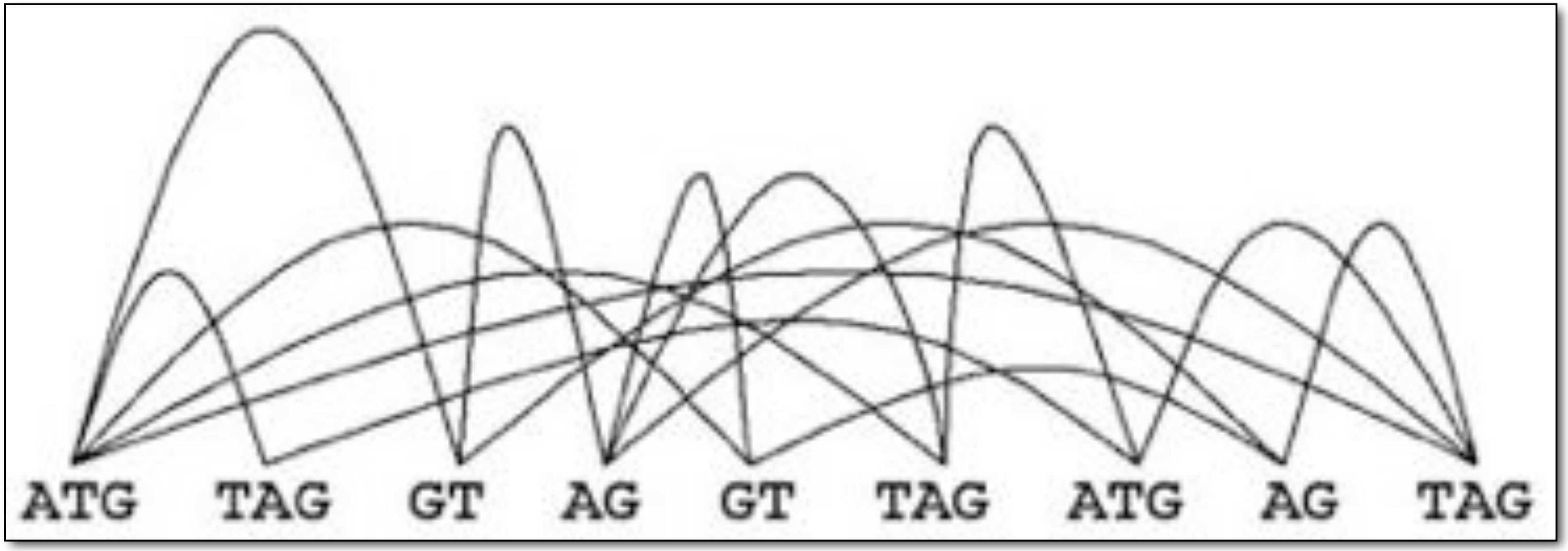
- *initial exons* extend from a start codon to the first donor site;
- *internal exons* extend from one acceptor site to the next donor site;
- *final exons* extend from the last acceptor site to the stop codon;
- *single exons* (which occur only in *intronless genes*) extend from the start codon to the stop codon:





Representing Gene Syntax with ORF Graphs

After identifying the most promising (i.e., highest-scoring) signals in an input sequence, we can apply the gene syntax rules to connect these into an *ORF graph*:

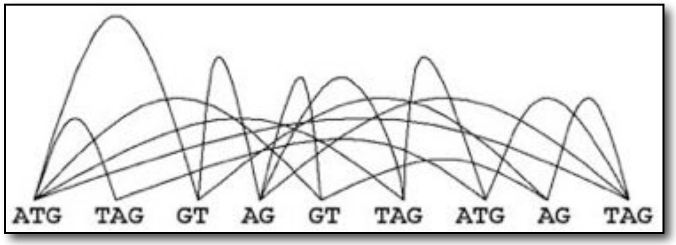


An ORF graph represents all possible *gene parses* (and their scores) for a given set of putative signals. A *path* through the graph represents a single gene parse.

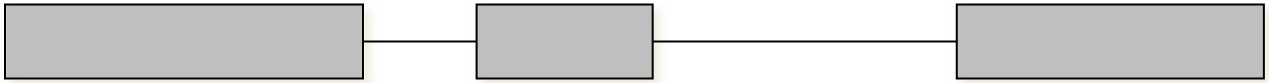
Conceptual Gene-finding Framework

```
TATTCCGATCGATCGATCTCTCTAGCGTCTACG
CTATCATCGCTCTCTATTATCGCGCGATCGTCG
ATCGCGCGAGAGTATGCTACGTCGATCGAATTG
```

identify most promising signals, score signals and content regions between them; induce an ORF graph on the signals

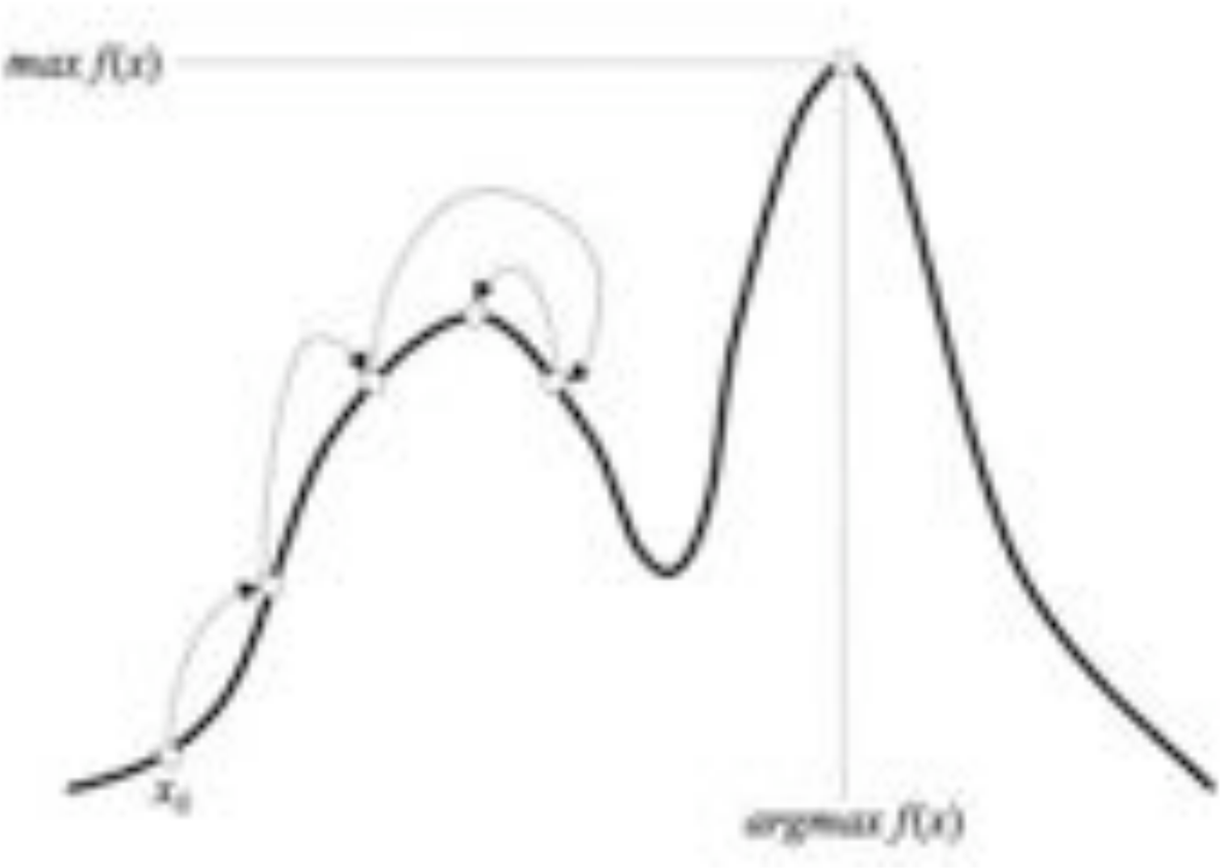


find highest-scoring path through ORF graph; interpret path as a gene parse = gene structure



The Notion of an Optimal Gene Structure

If we could enumerate all putative gene structures along the x-axis and graph their scores according to some function f(x), then the highest-scoring parse would be denoted argmax f(x), and its score would be denoted max f(x). A gene finder will often find the *local maximum* rather than the *global maximum*.

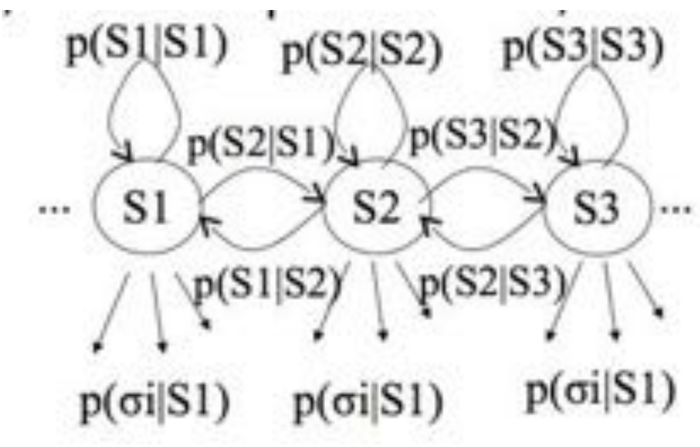


Hidden Markov Models (HMMs)

Steven Salzberg
CMSC 828N, Univ. of Maryland

What is an HMM?

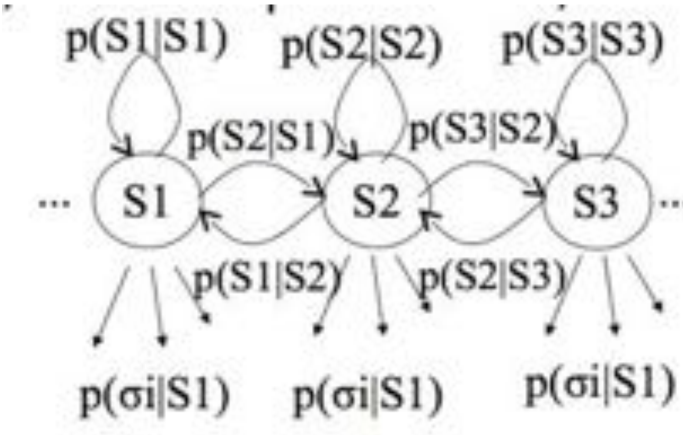
- Dynamic Bayesian Network
  - A set of states
    - {Fair, Biased} for coin tossing
    - {Gene, Not Gene} for Bacterial Gene
    - {Intergenic, Exon, Intron} for Eukaryotic Gene



  - A set of emission characters
    - E={H,T} for coin tossing
    - E={1,2,3,4,5,6} for dice tossing
    - E={A,C,G,T} = for DNA
  - State-specific emission probabilities
    - P(H | Fair) = .5, P(T | Fair) = .5, P(H | Biased) = .9, P(T | Biased) = .1
    - P(A | Gene) = .9, P(A | Not Gene) = .1 …
  - A probability of taking a transition
    - $P(s_i=Fair|s_{i-1}=Fair) = .9$, $P(s_i=Bias|s_{i-1} = Fair)$ .1
    - $P(s_i=Exon \mid s_{i-1}=Intergenic)$, …

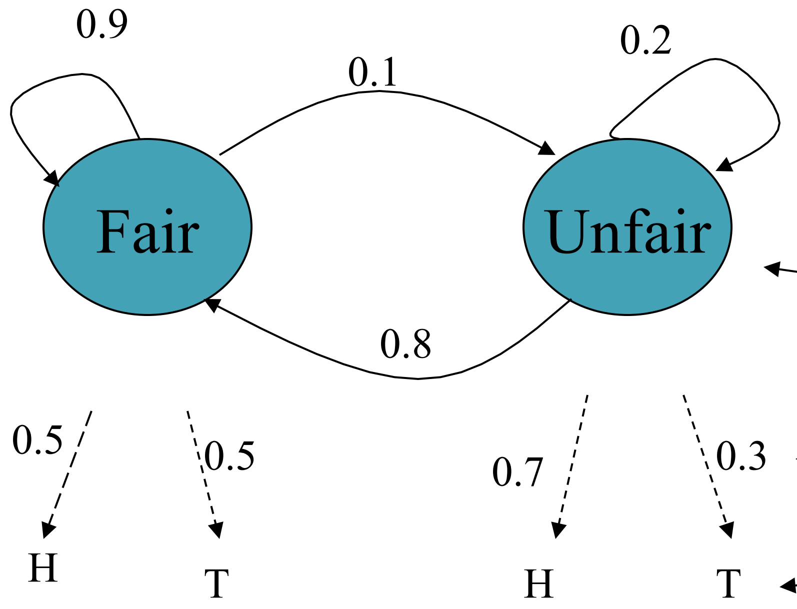
Why Hidden?

- Observers can see the emitted symbols of an HMM but have no ability to know which state the HMM is currently in.
  - But we can ***infer*** the most likely hidden states of an HMM based on the given sequence of emitted symbols.



HTHHTTHHHTHTHTHHTHHHHHHTHTHH

HMM Example - Casino Coin



State transition probs.

States

Symbol emission probs.

Observation Symbols

Observation Sequence

State Sequence

HTHHTTHHHTHTHTHHTHHHHHHTHTHH

FFFFFFUUUFFFFFFUUUUUUUFFFFFF

Motivation: Given a sequence of H & Ts, can you tell at what times the casino cheated?

Slide credit: Fatih Gelgi, Arizona State U.

Three classic HMM problems

1. **Evaluation**: given a model and an output sequence, what is the probability that the model generated that output?

- To answer this, we consider all possible paths through the model

- Example: we might have a set of HMMs representing protein families -> pick the model with the best score

Three classic HMM problems

2. **Decoding**: given a model and an output sequence, what is the most likely state sequence through the model that generated the output?

- A solution to this problem gives us a way to match up an observed sequence and the states in the model.

AAAGC ATG CAT TTA ACG AGA GCA CAA GGG CTC TAA TGCCG

The sequence of states is an annotation of the generated string – each nucleotide is generated in intergenic, start/stop, coding state

Three classic HMM problems

3. **Learning**: given a model and a set of observed sequences, how do we set the model's parameters so that it has a high probability of generating those sequences?

- This is perhaps the most important, and most difficult problem.

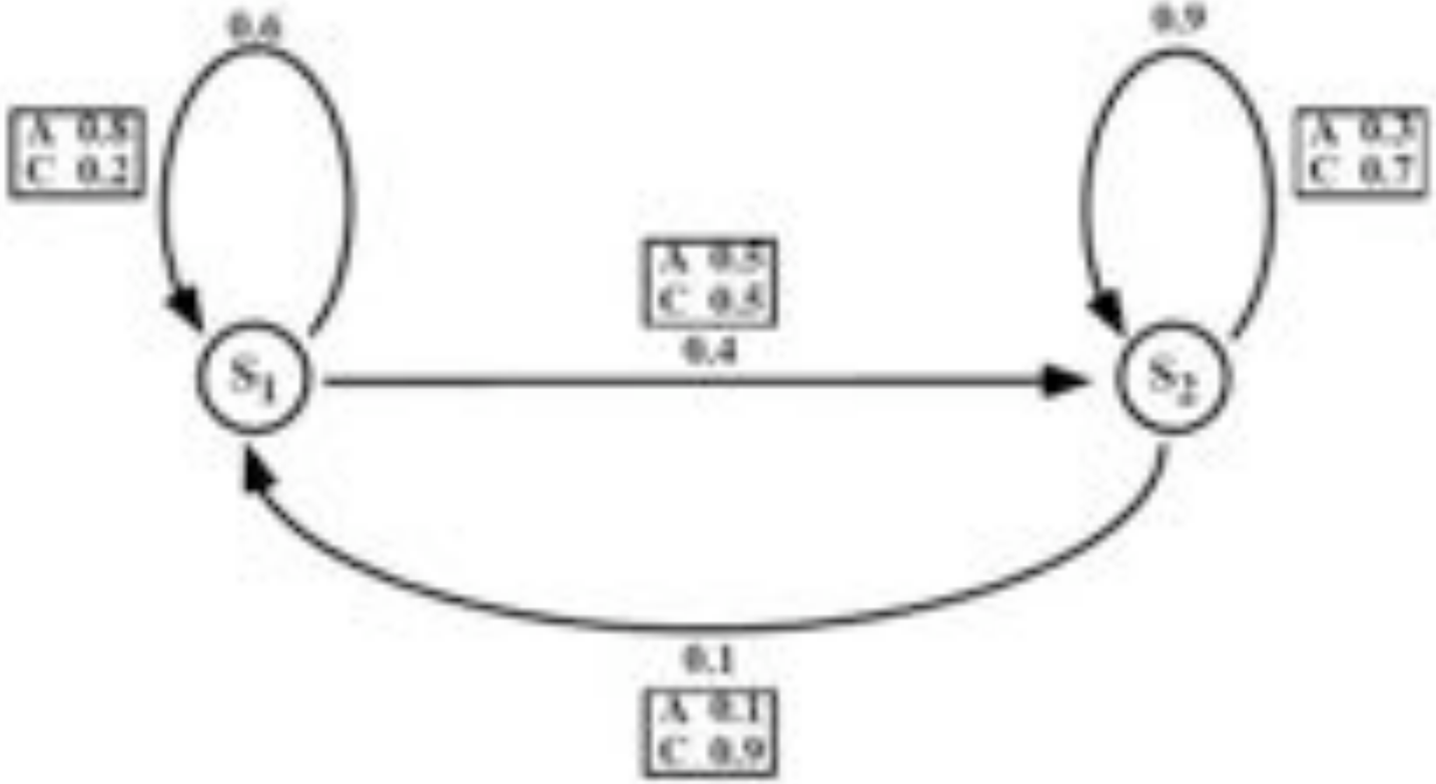
- A solution to this problem allows us to determine all the probabilities in an HMMs by using an ensemble of training data

Solving the Evaluation problem: The Forward algorithm

- To solve the Evaluation problem (probability that the model generated the sequence), we use the HMM and the data to build a *trellis*

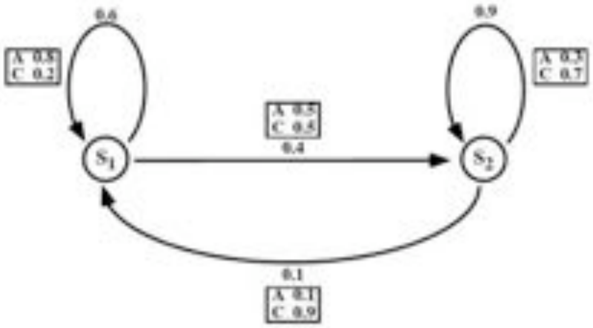
- Filling in the trellis will give tell us the probability that the HMM generated the data by finding all possible paths that could do it

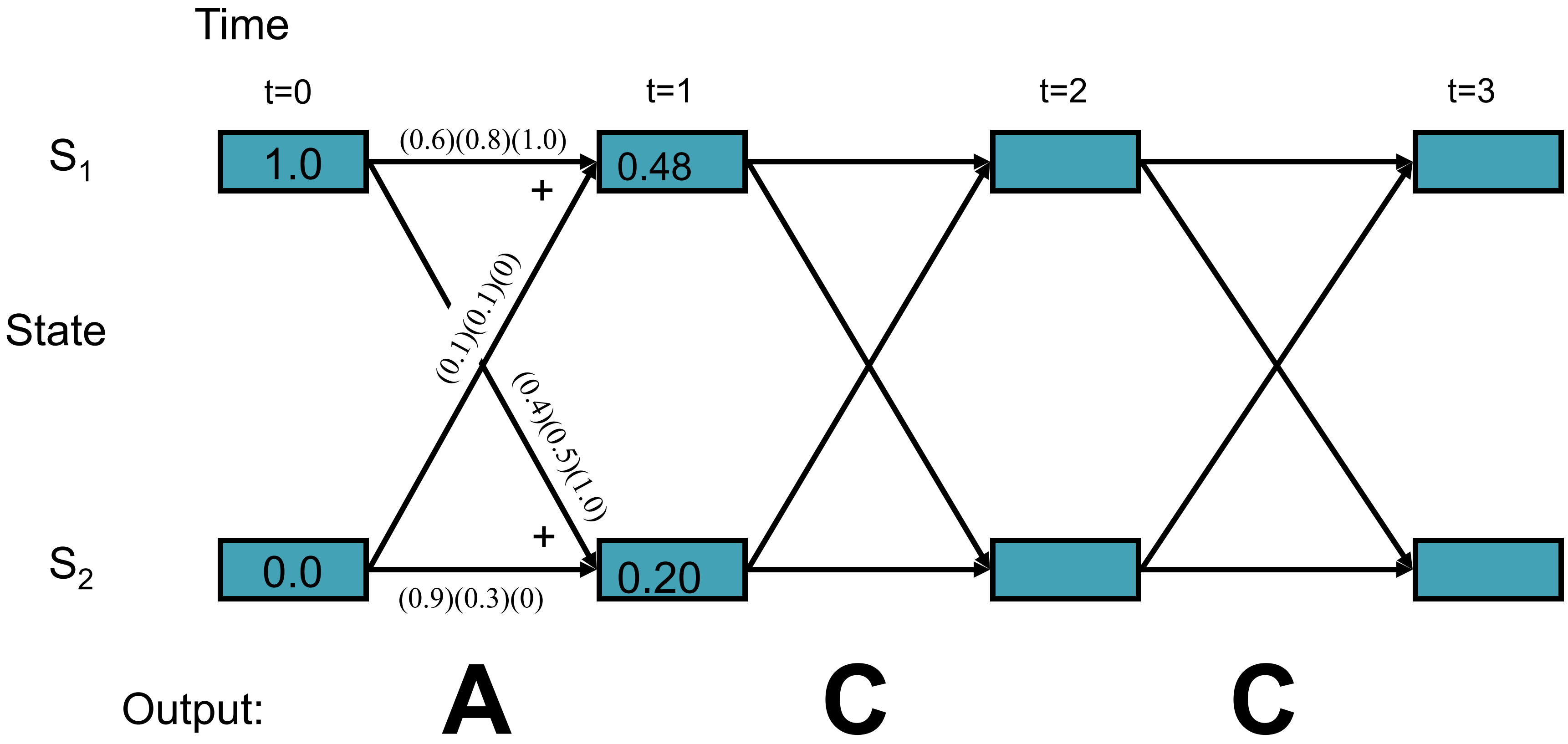
Our sample HMM



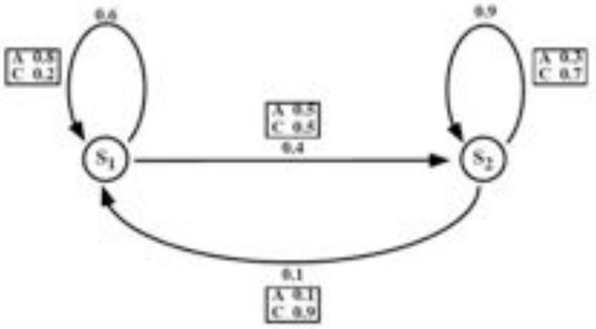
Let S_1 be initial state, S_2 be final state

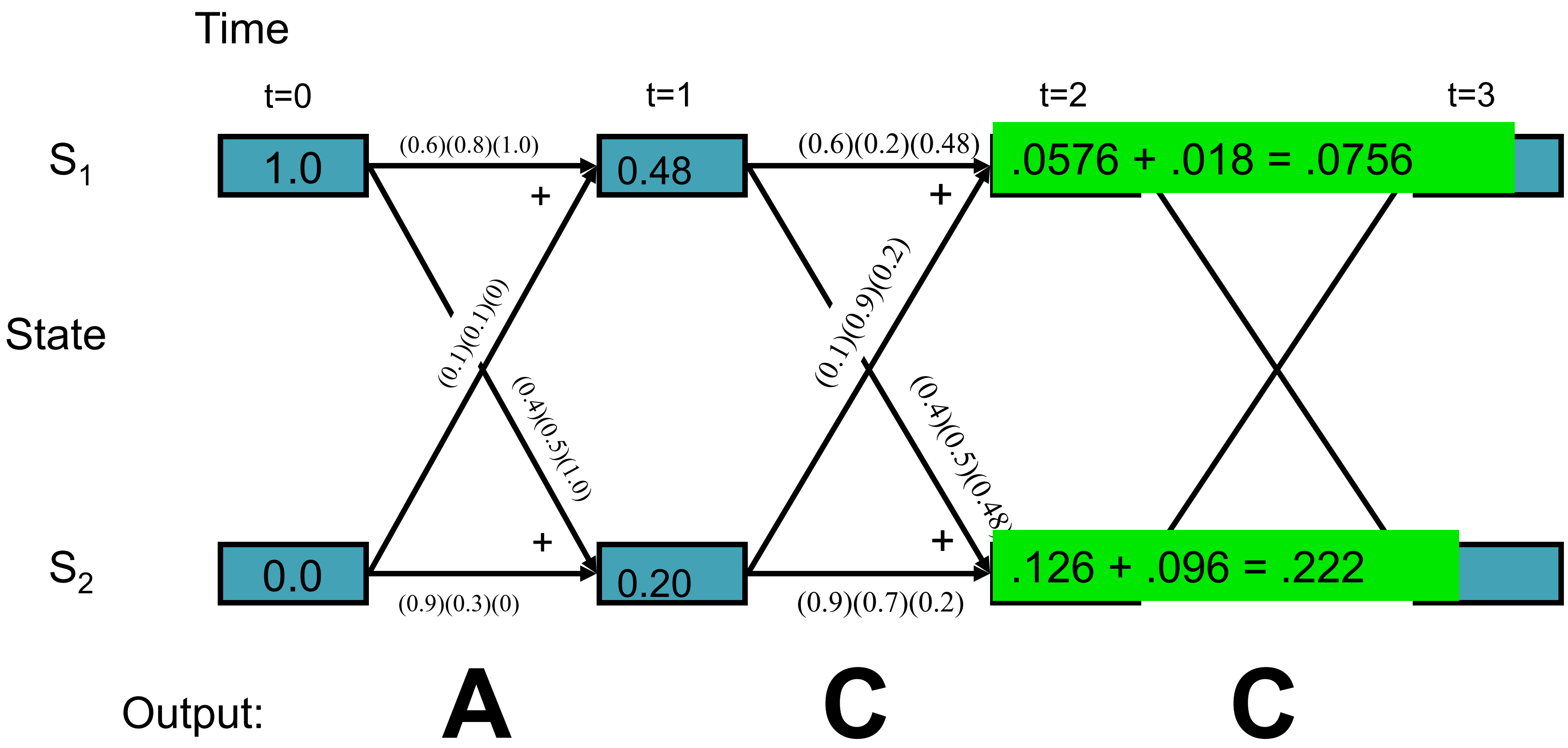
A trellis for the Forward Algorithm



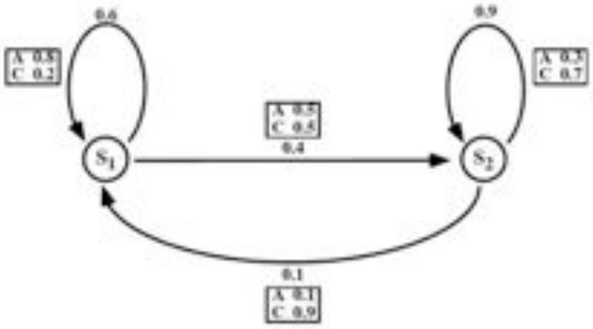


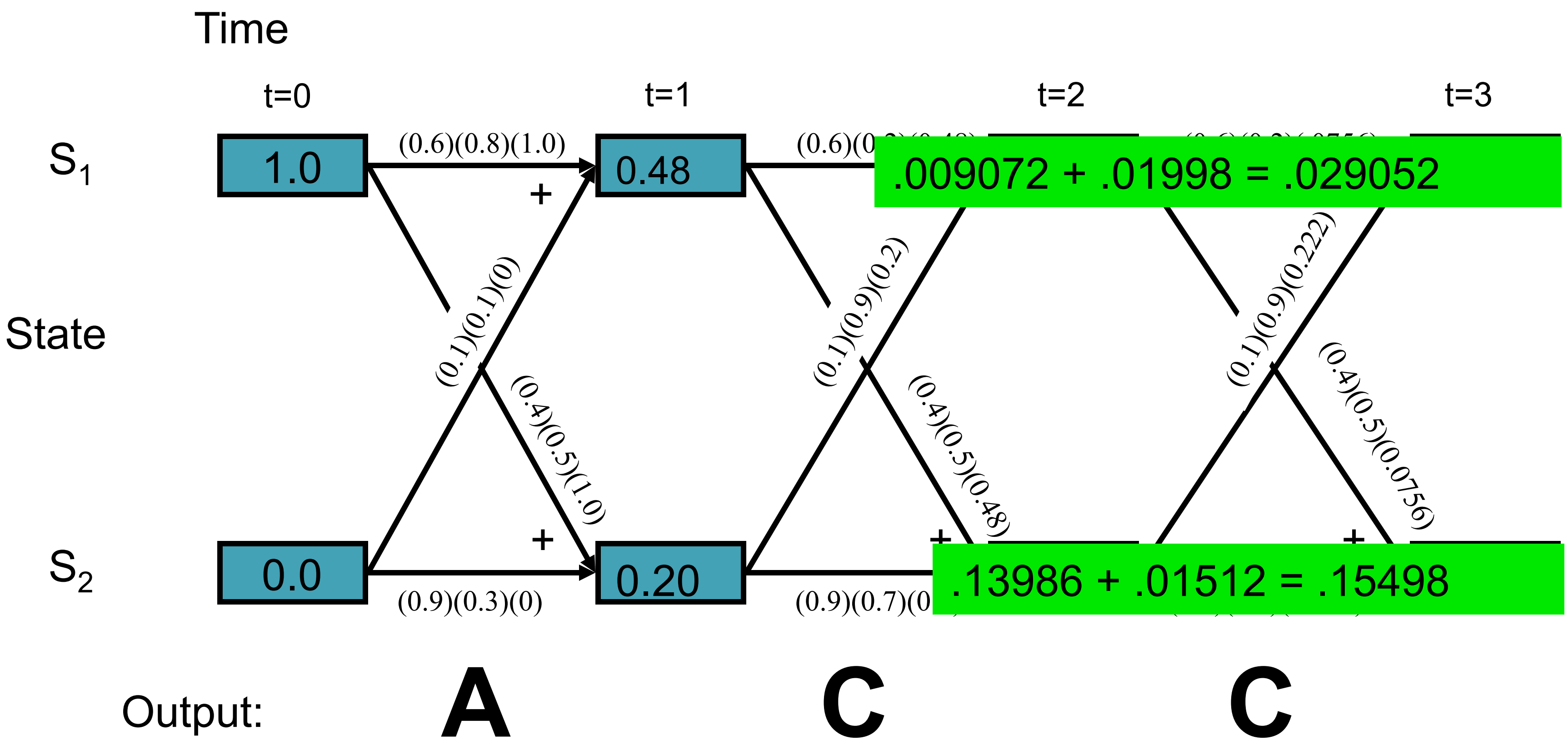
A trellis for the Forward Algorithm





A trellis for the Forward Algorithm





.009072 + .01998 = .029052

.13986 + .01512 = .15498

Probability of the model

- The Forward algorithm computes *P(y|M)*
- If we are comparing two or more models, we want the likelihood that each model generated the data: *P(M|y)*
- Use Bayes' law:

$$P(M \mid y) = \frac{P(y \mid M)P(M)}{P(y)}$$

- Since P(y) is constant for a given input, we just need to maximize *P(y|M)P(M)*

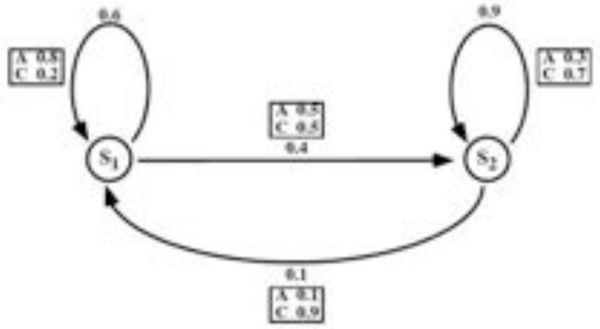
Solving the Decoding Problem: The Viterbi algorithm

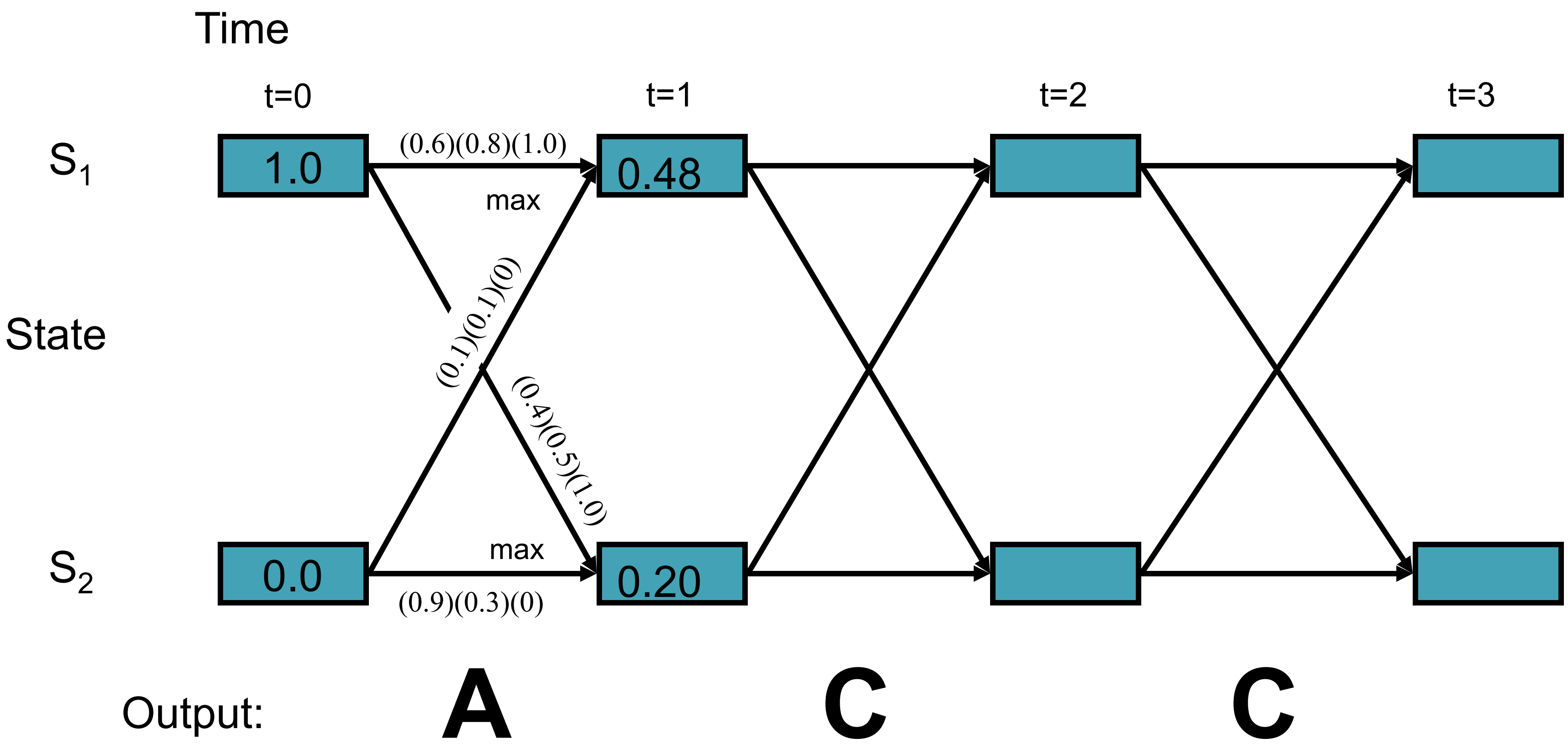
- To solve the decoding problem (find the most likely sequence of states), we evaluate the Viterbi algorithm

$$V_i(t) = \begin{cases} 0 & : \quad t = 0 \wedge i \neq S_I \\ 1 & : \quad t = 0 \wedge i = S_I \\ \max V_j(t-1) a_{ji} b_{ji}(y) & : \quad t > 0 \end{cases}$$

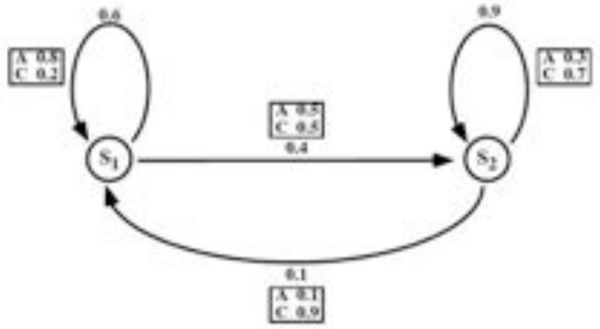
Where $V_i(t)$ is the probability that the HMM is in state *i* after generating the sequence $y_1, y_2, \ldots, y_t$, following the *most probable path* in the HMM

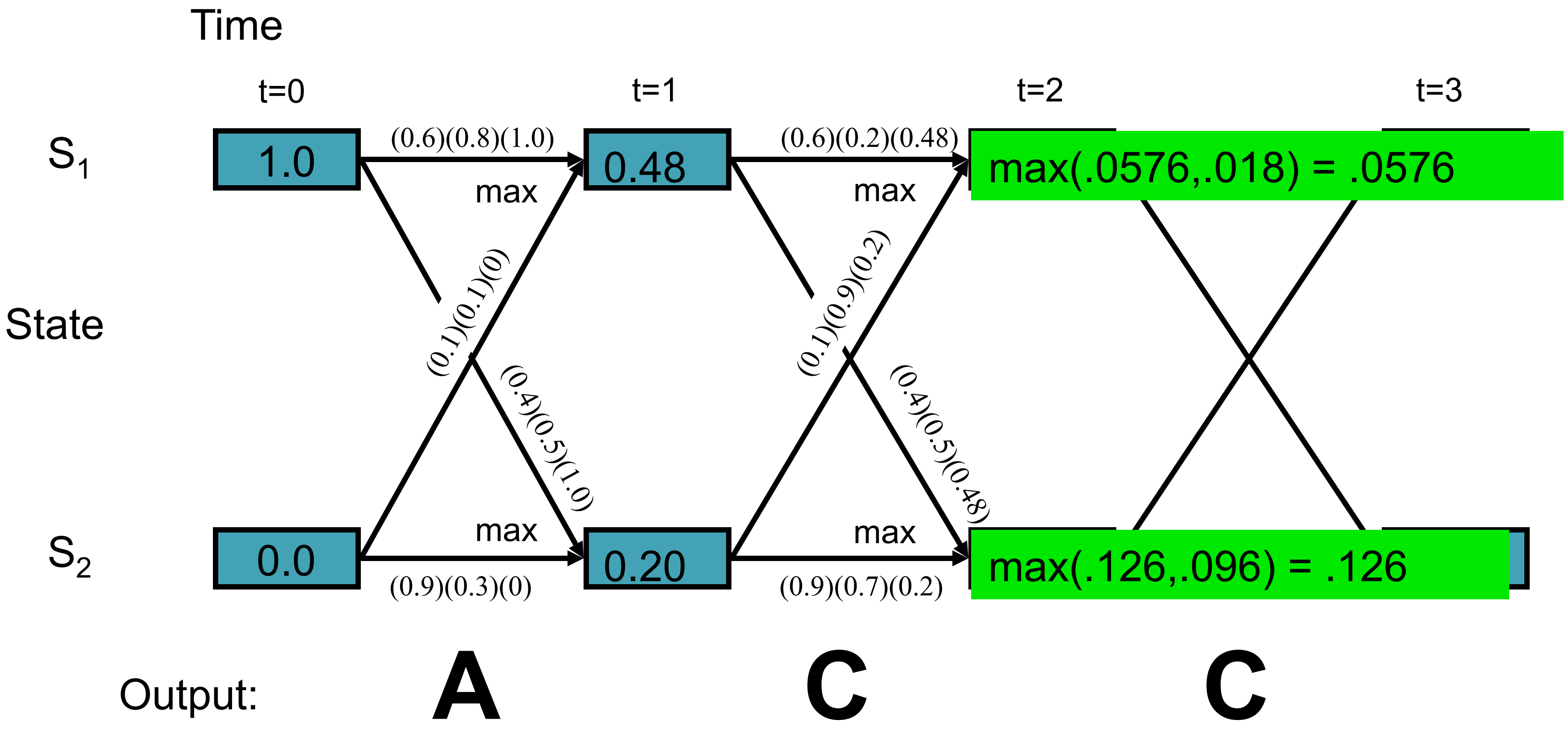
A trellis for the Viterbi Algorithm





A trellis for the Viterbi Algorithm





Time

t=0

t=1

t=2

t=3

S_1

State

S_2

1.0

(0.6)(0.8)(1.0)

max

0.48

(0.6)(0.2)(0.48)

max

max(.0576,.018) = .0576

(0.1)(0.1)(0)

(0.4)(0.5)(1.0)

(0.1)(0.9)(0.2)

(0.4)(0.5)(0.48)

0.0

max

(0.9)(0.3)(0)

0.20

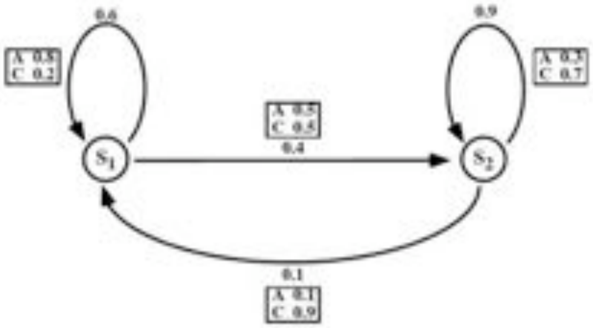
max

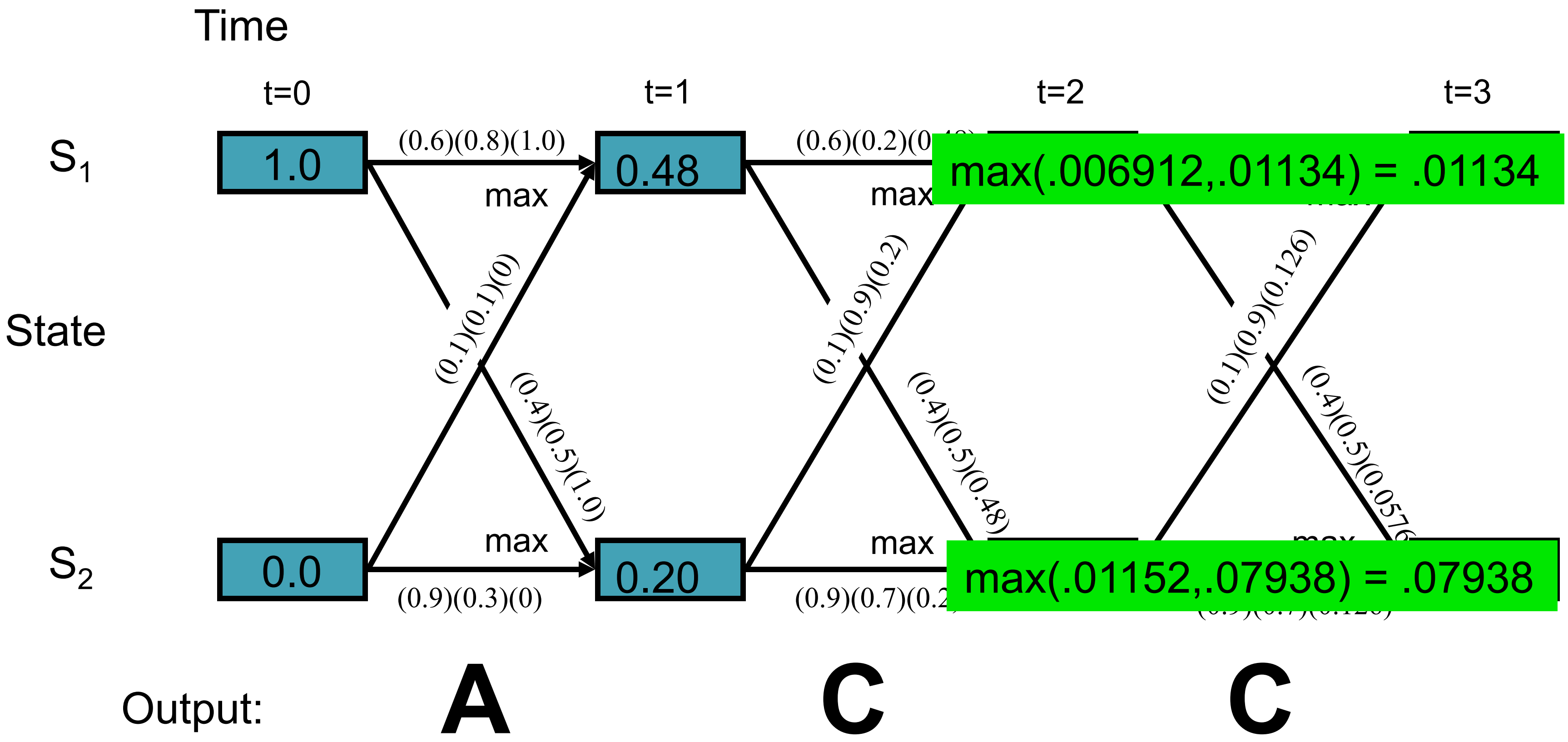
(0.9)(0.7)(0.2)

max(.126,.096) = .126

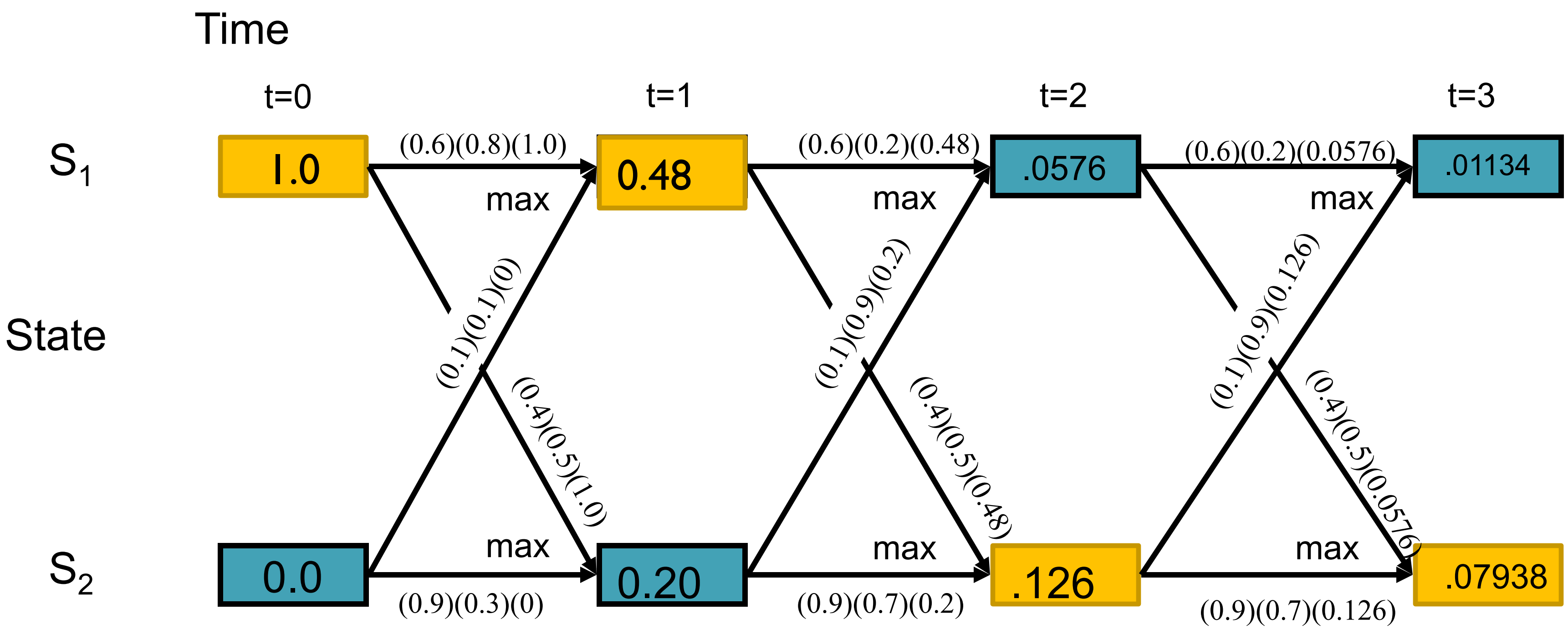
Output: A C C

A trellis for the Viterbi Algorithm





A trellis for the Viterbi Algorithm



Output: **A** **C** **C**

Parse: $\mathbf{S_1}$ $\mathbf{S_2}$ $\mathbf{S_2}$

Learning in HMMs: The E-M algorithm

- In order to learn the parameters in an "empty" HMM, we need:
  - The topology of the HMM
  - Data - the more the better

- The learning algorithm is called "Expectation-Maximization" or E-M
  - Also called the Forward-Backward algorithm
  - Also called the Baum-Welch algorithm

- See Supplemental Slides

Eukaryotic Gene Finding with GlimmerHMM

Mihaela Pertea

Assistant Research Scientist

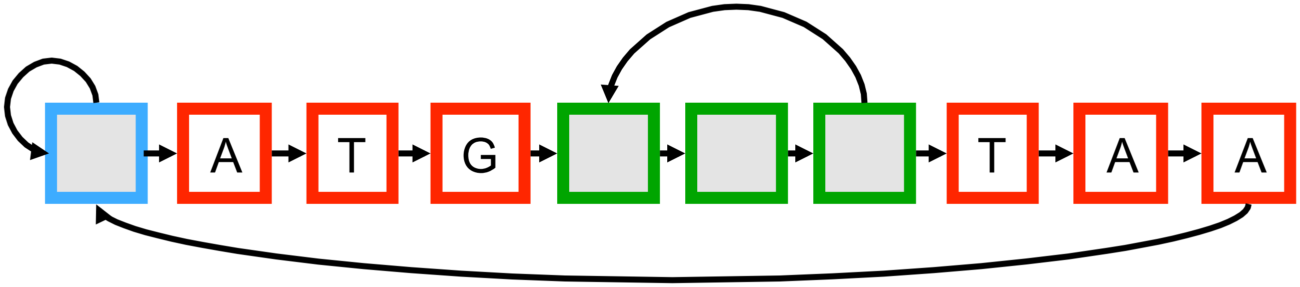
CBCB

HMMs and Gene Structure

- Nucleotides {*A*,*C*,*G*,*T*} are the observables

- Different states generate nucleotides at different frequencies

A simple HMM for unspliced genes:

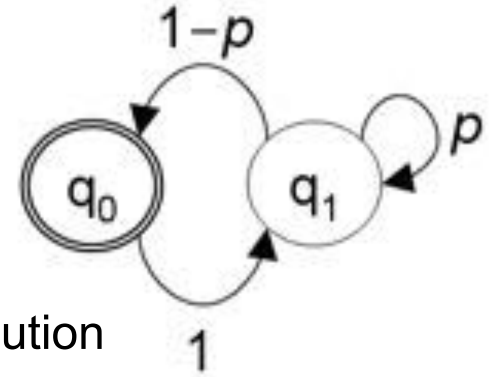


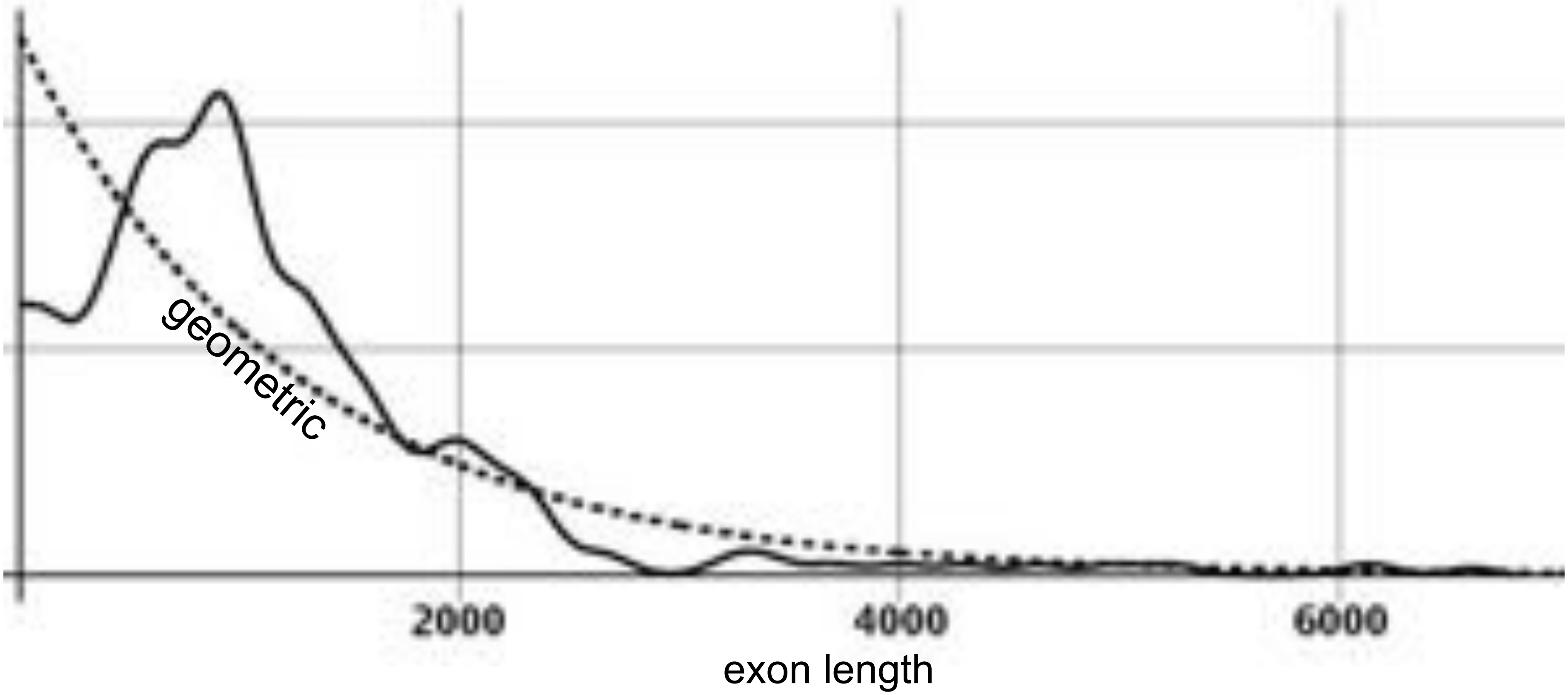
AAAGC ATG CAT TTA ACG AGA GCA CAA GGG CTC TAA TGCCG

- The sequence of states is an annotation of the generated string – each nucleotide is generated in intergenic, start/stop, coding state

HMMs & Geometric Feature Lengths

$$P(x_0 \ldots x_{d-1} \mid \theta) = \left(\prod_{i=0}^{d-1} P_e(x_i \mid \theta) \right) \underbrace{p^{d-1}(1-p)}_{\text{geometric distribution}}$$





Generalized HMMs Summary

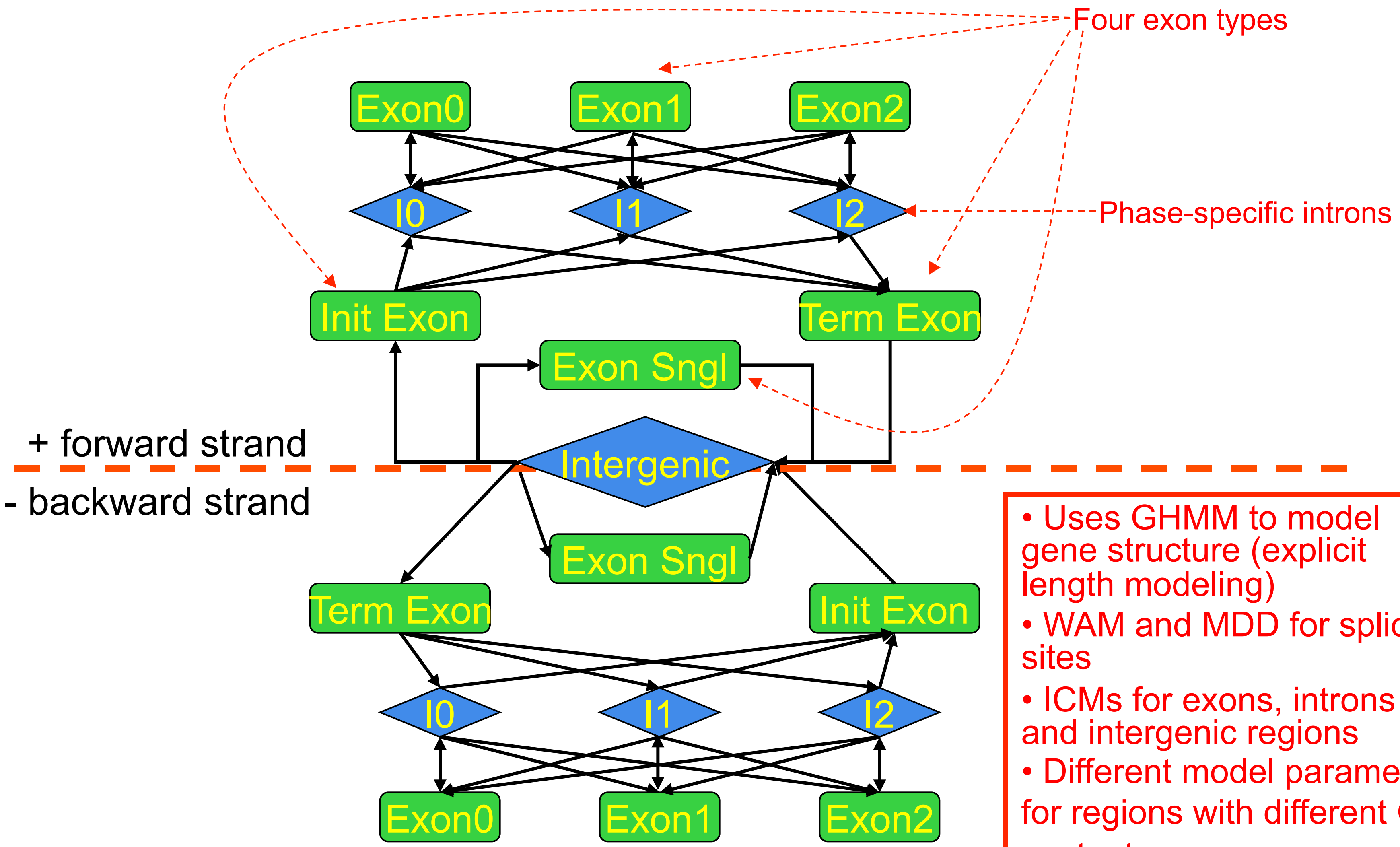
- GHMMs generalize HMMs by allowing each state to emit a subsequence rather than just a single symbol

- Whereas HMMs model all feature lengths using a geometric distribution, coding features can be modeled using an arbitrary length distribution in a GHMM

- Emission models within a GHMM can be any arbitrary probabilistic model ("submodel abstraction"), such as a neural network or decision tree

- GHMMs tend to have many fewer states => simplicity & modularity

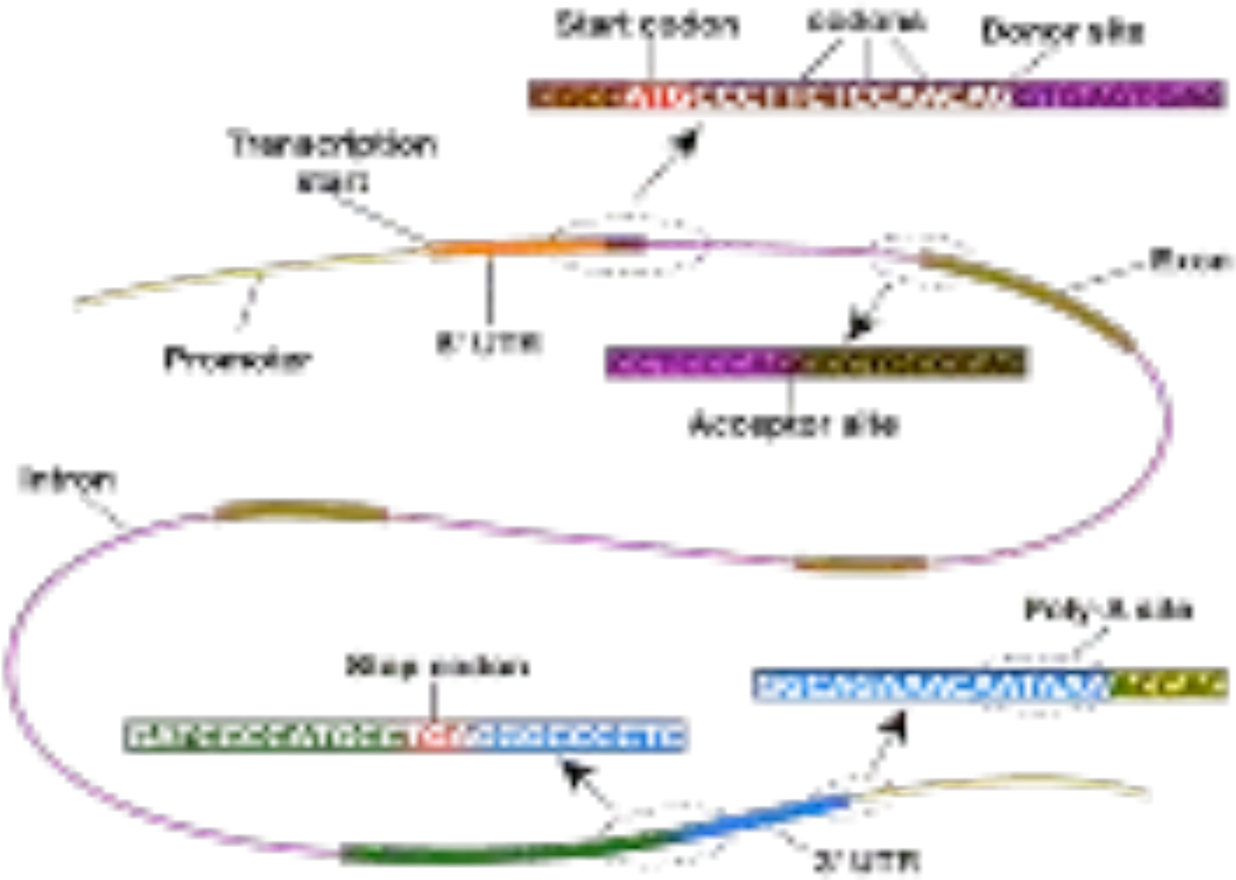
GlimmerHMM architecture



- Uses GHMM to model gene structure (explicit length modeling)
- WAM and MDD for splice sites
- ICMs for exons, introns and intergenic regions
- Different model parameters for regions with different GC content
- Can emit a graph of high-scoring ORFS

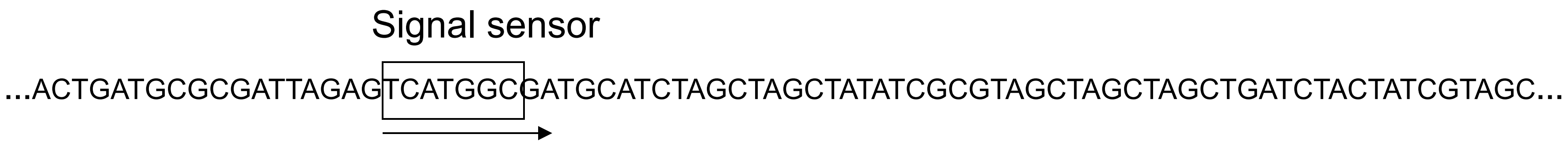
Signal Sensors

Signals – short sequence patterns in the genomic DNA that are recognized by the cellular machinery.



Identifying Signals In DNA

We slide a fixed-length model or “window” along the DNA and evaluate `score (signal)` at each point:



Signal sensor

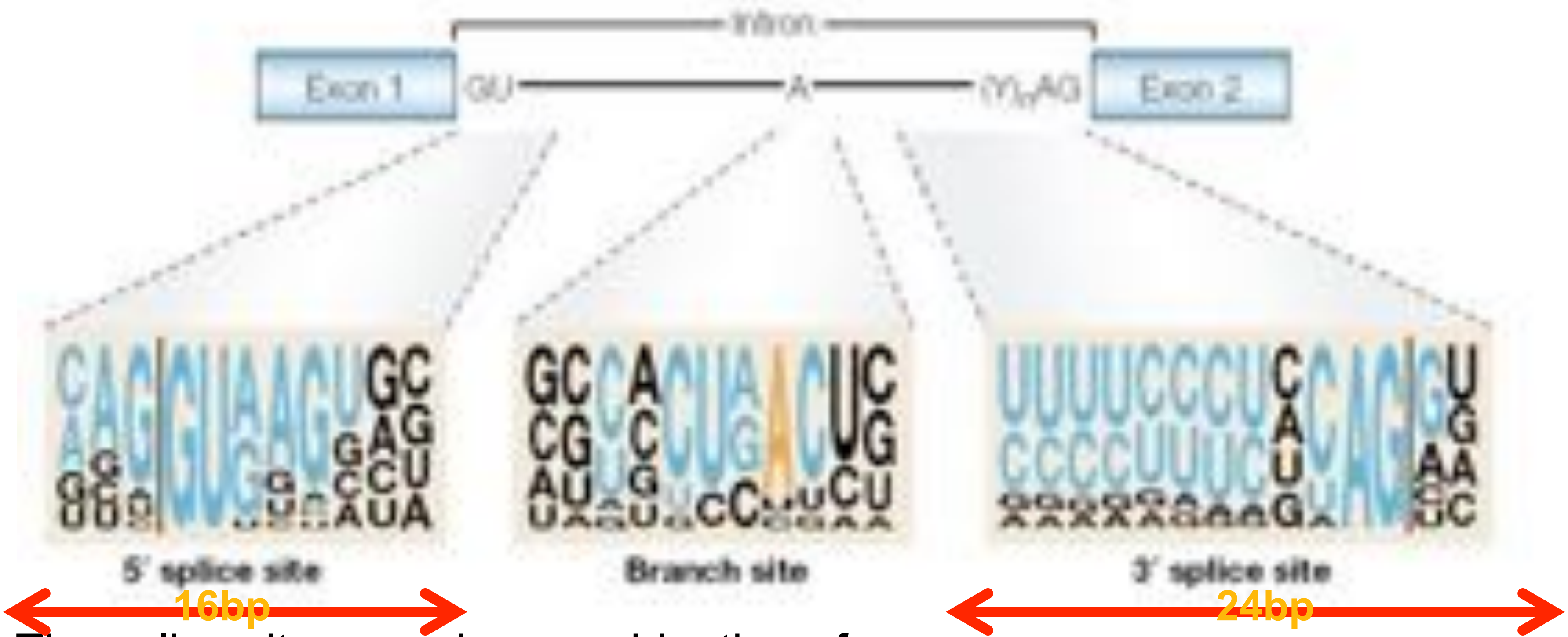
...ACTGATGCGCGATTAGAGTCATGGCGATGCATCTAGCTAGCTATATCGCGTAGCTAGCTAGCTGATCTACTATCGTAGC...

When the `score` is greater than some threshold (determined empirically to result in a desired sensitivity), we remember this position as being the potential site of a signal.

The most common signal sensor is the Weight Matrix:

A = 31% T = 28% C = 21% G = 20%	A = 18% T = 32% C = 24% G = 26%	**A** 100%	**T** 100%	**G** 100%	A = 19% T = 20% C = 29% G = 32%	A = 24% T = 18% C = 26% G = 32%

Splice site prediction

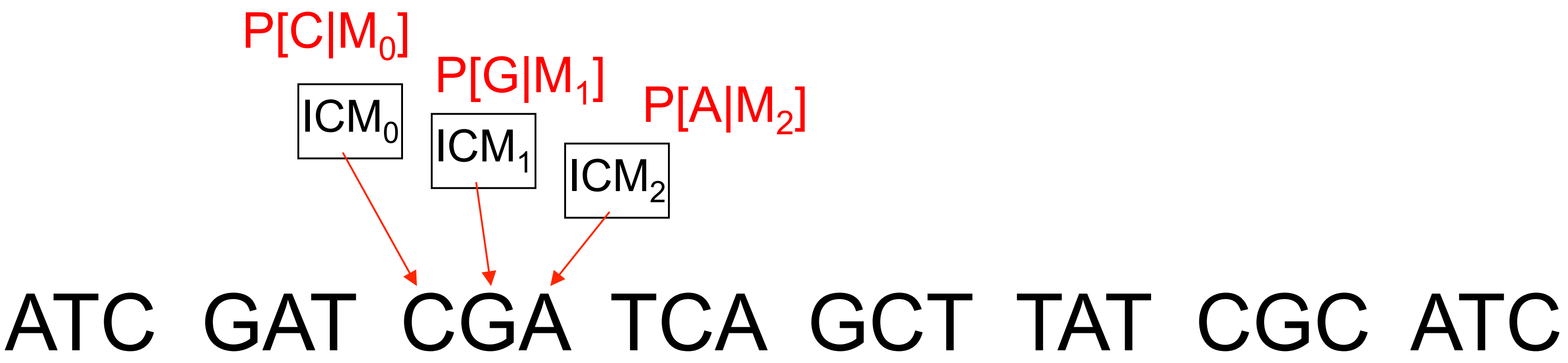


The splice site score is a combination of:

- first or second order inhomogeneous Markov models on windows around the acceptor and donor sites
- MDD decision trees
- longer Markov models to capture difference between coding and non-coding on opposite sides of site (optional)
- maximal splice site score within 60 bp (optional)

Coding vs Non-coding

A three-periodic ICM uses three ICMs in succession to evaluate the different codon positions, which have different statistics:



P[C|M_0] P[G|M_1] P[A|M_2]

ICM_0 ICM_1 ICM_2

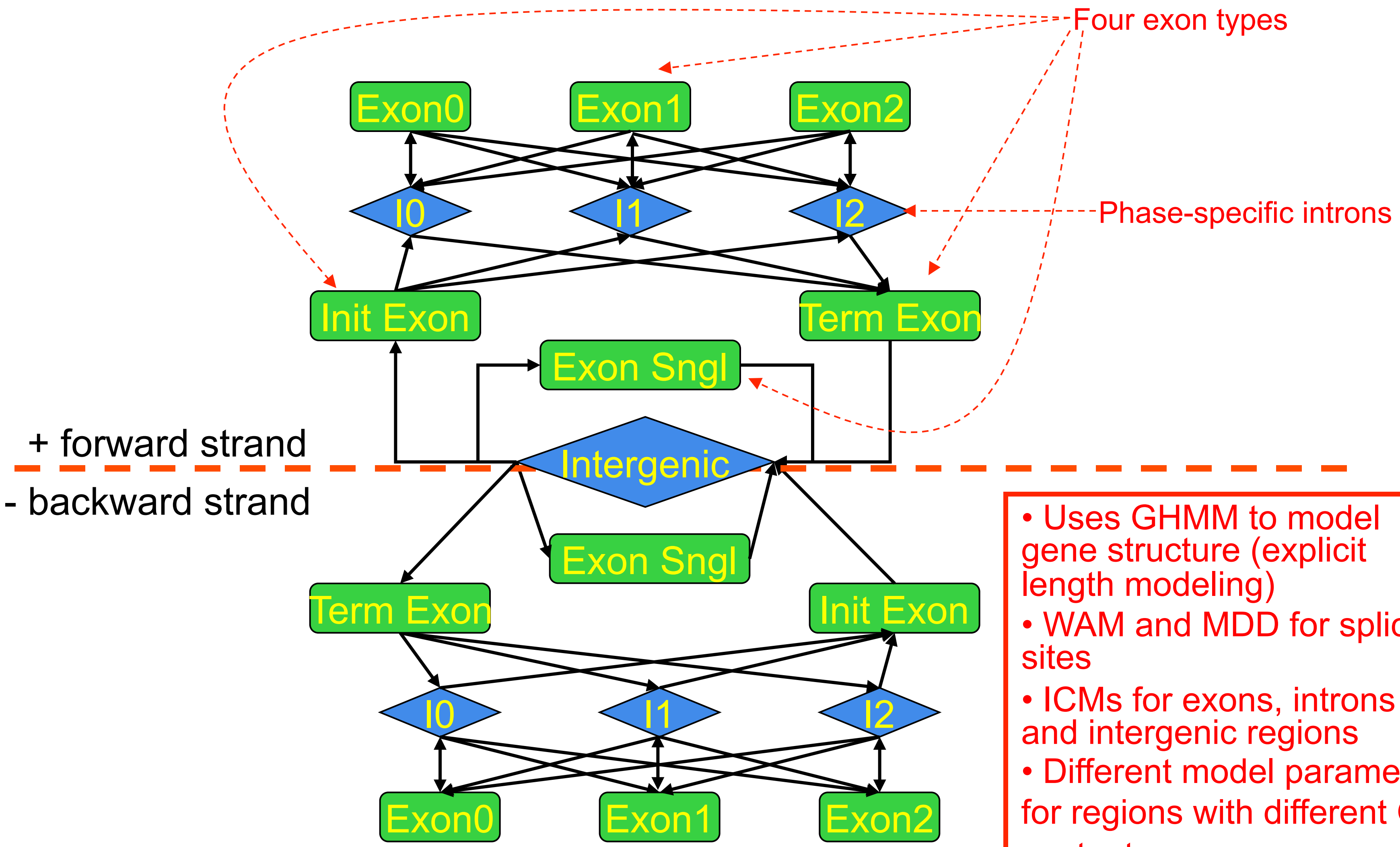
ATC GAT CGA TCA GCT TAT CGC ATC

The three ICMs correspond to the three phases. Every base is evaluated in every phase, and the score for a given stretch of (putative) coding DNA is obtained by multiplying the phase-specific probabilities in a mod 3 fashion:

$$\prod_{i=0}^{L-1} P_{(f+i)(\mathrm{mod}3)}(x_i)$$

GlimmerHMM uses 3-periodic ICMs for coding and homogeneous (non-periodic) ICMs for noncoding DNA.

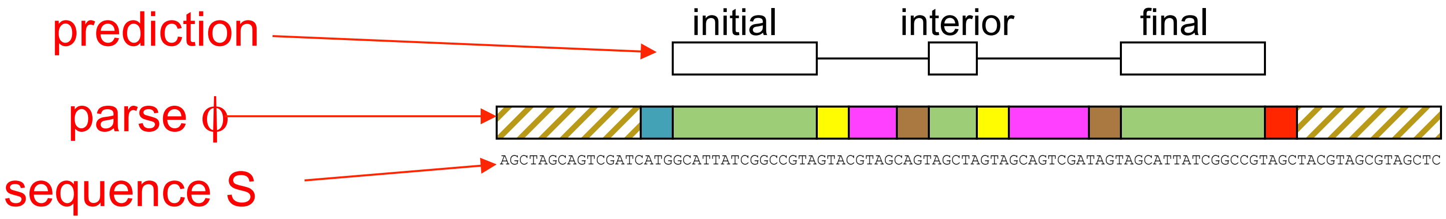
GlimmerHMM architecture



- Uses GHMM to model gene structure (explicit length modeling)
- WAM and MDD for splice sites
- ICMs for exons, introns and intergenic regions
- Different model parameters for regions with different GC content
- Can emit a graph of high-scoring ORFS

Gene Prediction with a GHMM

Given a sequence S, we would like to determine the parse ϕ of that sequence which segments the DNA into the most likely exon/intron structure:

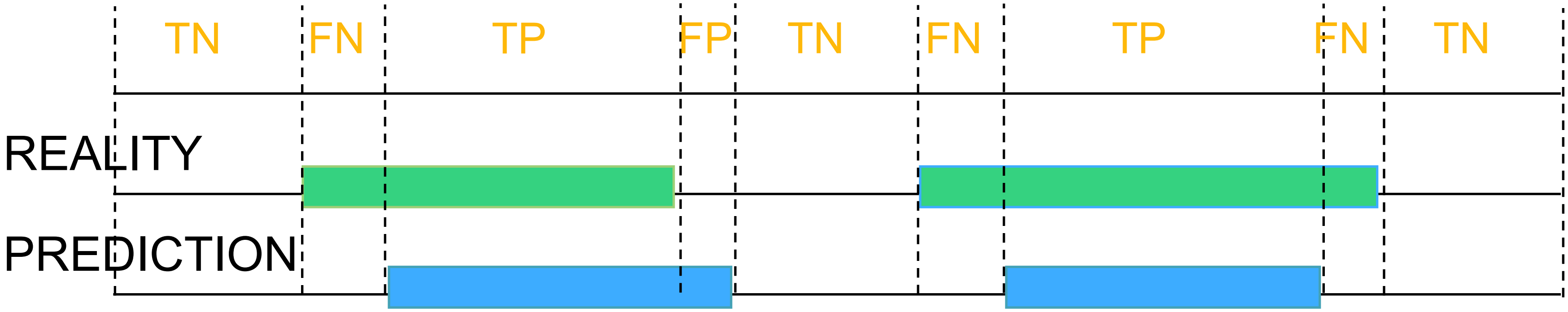

prediction
parse ϕ
sequence S
initial
interior
final
AGCTAGCAGTCGATCATGGCATTATCGGCCGTAGTACGTAGCAGTAGCTAGTAGCAGTCGATAGTAGCATTATCGGCCGTAGCTACGTAGCGTAGCTC

The parse ϕ consists of the coordinates of the predicted exons, and corresponds to the precise sequence of states during the operation of the GHMM (and their duration, which equals the number of symbols each state emits).

This is the same as in an HMM except that in the HMM each state emits bases with fixed probability, whereas in the GHMM each state emits an entire feature such as an exon or intron.

Evaluation of Gene Finding Programs

Nucleotide level accuracy

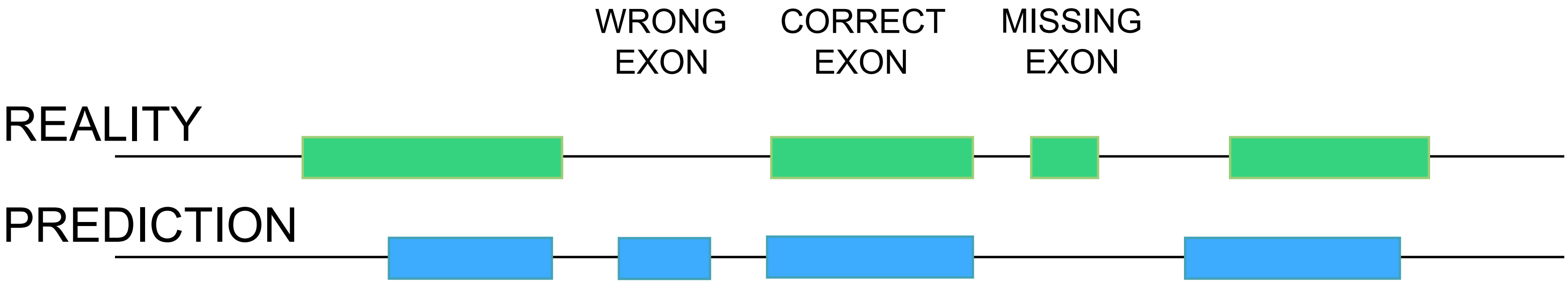


Sensitivity: $$Sn = \frac{TP}{TP + FN}$$

Specificity: $$Sp = \frac{TP}{TP + FP}$$

More Measures of Prediction Accuracy

Exon level accuracy



$$ExonSn = \frac{TE}{AE} = \frac{\text{number of correct exons}}{\text{number of actual exons}}$$

$$ExonSp = \frac{TE}{PE} = \frac{\text{number of correct exons}}{\text{number of predicted exons}}$$

GlimmerHMM on human data

	Nuc Sens	*Nuc Spec*	*Nuc Acc*	*Exon Sens*	*Exon Spec*	*Exon Acc*	*Exact Genes*
GlimmerHMM	86%	72%	79%	72%	62%	67%	17%
Genscan	86%	68%	77%	69%	60%	65%	13%

GlimmerHMM’s performace compared to Genscan on 963 human RefSeq genes selected randomly from all 24 chromosomes, non-overlapping with the training set. The test set contains 1000 bp of untranslated sequence on either side (5' or 3') of the coding portion of each gene.

GlimmerHMM is a high-performance ab initio gene finder

Arabidopsis thaliana test results

	Nucleotide			Exon			Gene		
	Sn	Sp	**Acc**	Sn	Sp	**Acc**	Sn	Sp	**Acc**
GlimmerHMM	97	99	**98**	84	89	**86.5**	60	61	**60.5**
SNAP	96	99	**97.5**	83	85	**84**	60	57	**58.5**
Genscan+	93	99	**96**	74	81	**77.5**	35	35	**35**

•All three programs were tested on a test data set of 809 genes, which did not overlap with the training data set of GlimmerHMM.

•All genes were confirmed by full-length Arabidopsis cDNAs and carefully inspected to remove homologues.

Summary

- Prokaryotic gene finding distinguishes between genes and random ORFs
  - Prokaryotic genes have simple structure and are largely homogenous, making it relatively easy to recognize their sequence composition
- Eukaryotic gene finding identifies the genome-wide most probable gene models (set of exons)
  - GHMM to enforce overall gene structure, separate models to score splicing/transcription signals
  - Accuracy depends to a large extent on the quality of the training data
    - All future genome projects will be accompanied by mRNAseq

Break

READING THE ENTIRE GENOME CODE...
Listen to this part: "AGGCTAATC CGCATAACTG"
Wow!

Review

Exact Matching

- Explain the Brute Force search algorithm (algorithm sketch, running time, space requirement)

1. Suffix Arrays

2. Suffix Trees

3. Hash Tables

4. How many times do we expected GATTACA or GATTACA*2 or GATTACA*3 to be in the genome?

Sequence Alignment

1. What is a good scoring scheme for aligning:

   English words? Illumina Reads? Gene Sequences? Genomes?

2. Explain Dynamic Programming for Sequence Alignment

3. BLAST

4. MUMmer

5. Bowtie

Graphs and Assembly

1. How do I compute the shortest path between 2 nodes and how long does it take?

2. Mark connected components in a graph?

3. Shortest path visiting all nodes?

4. Describe Genome Assembly

5. How do we detect mis-assemblies?

Gene Finding

1. Describe Prokaryotic Gene Finding

2. Describe Eukaryotic gene finding

3. What is an Markov Chain?
   - IMM? ICM? HMM? GHMM?

4. What do the Forward and Viterbi Algorithms Compute

CS Fundamentals

1. Order these running times

   $O(\lg n)$, $O(2^n)$, $O(n^{100})$, $O(n^2)$, $O(n!)$ $O(n\lg n)$, $O(n(\lg n)(\lg n))$, $O(1)$, $O(1.5^n)$

2. Describe Selection Sort
3. QuickSort
4. Bucket Sort

5. Describe Recursion
6. Dynamic Programming
7. Branch-and-Bound
8. Greedy Algorithm

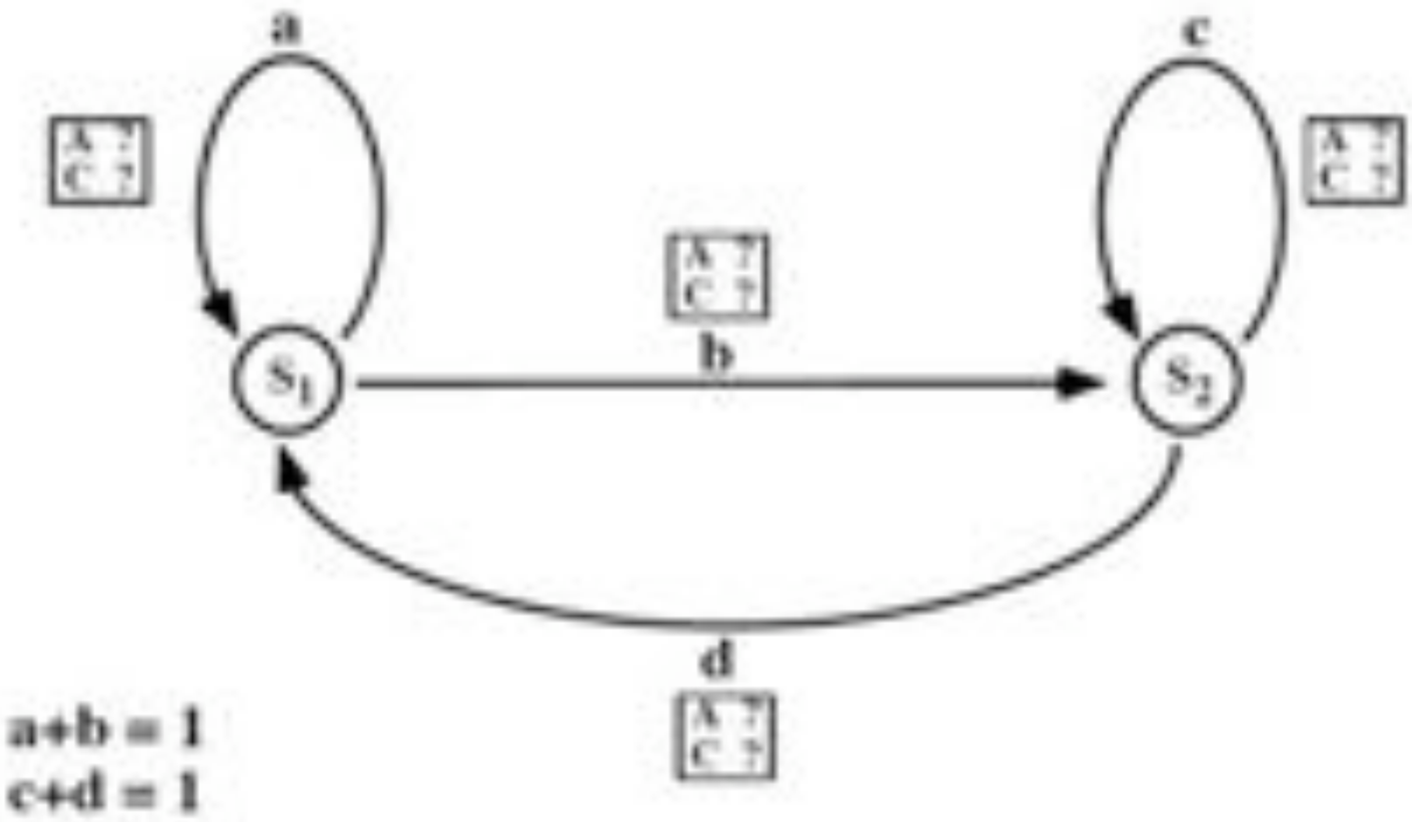
9. Describe an NP-complete problem

Supplemental

Learning in HMMs: The E-M algorithm

- In order to learn the parameters in an "empty" HMM, we need:
  - The topology of the HMM
  - Data - the more the better
- The learning algorithm is called "Estimate-Maximize" or E-M
  - Also called the Forward-Backward algorithm
  - Also called the Baum-Welch algorithm

An untrained HMM



Some HMM training data

- CACAACAAAACCCCCCACAA
- ACAACACACACACACACCAAAC
- CAACACACAAACCCC
- CAACCACCACACACACACCCCA
- CCCAAAACCCCAAAAACCC
- ACACAAAAAACCCAACACACAACA
- ACACAACCCCAAAACCACCAAAAA

Step 1: Guess all the probabilities

- We can start with random probabilities, the learning algorithm will adjust them
- If we can make good guesses, the results will generally be better

Step 2: the Forward algorithm

- Reminder: each box in the trellis contains a value $\alpha_i(t)$

$\alpha_i(t)$ is the probability that our HMM has generated the sequence y_1, y_2, …, y_t and has ended up in state i.

Reminder: notations

- sequence of length T:

y_1^T

- all sequences of length T:

Y_1^T

- Path of length T+1 generates Y:

x_1^{T+1}

- All paths:

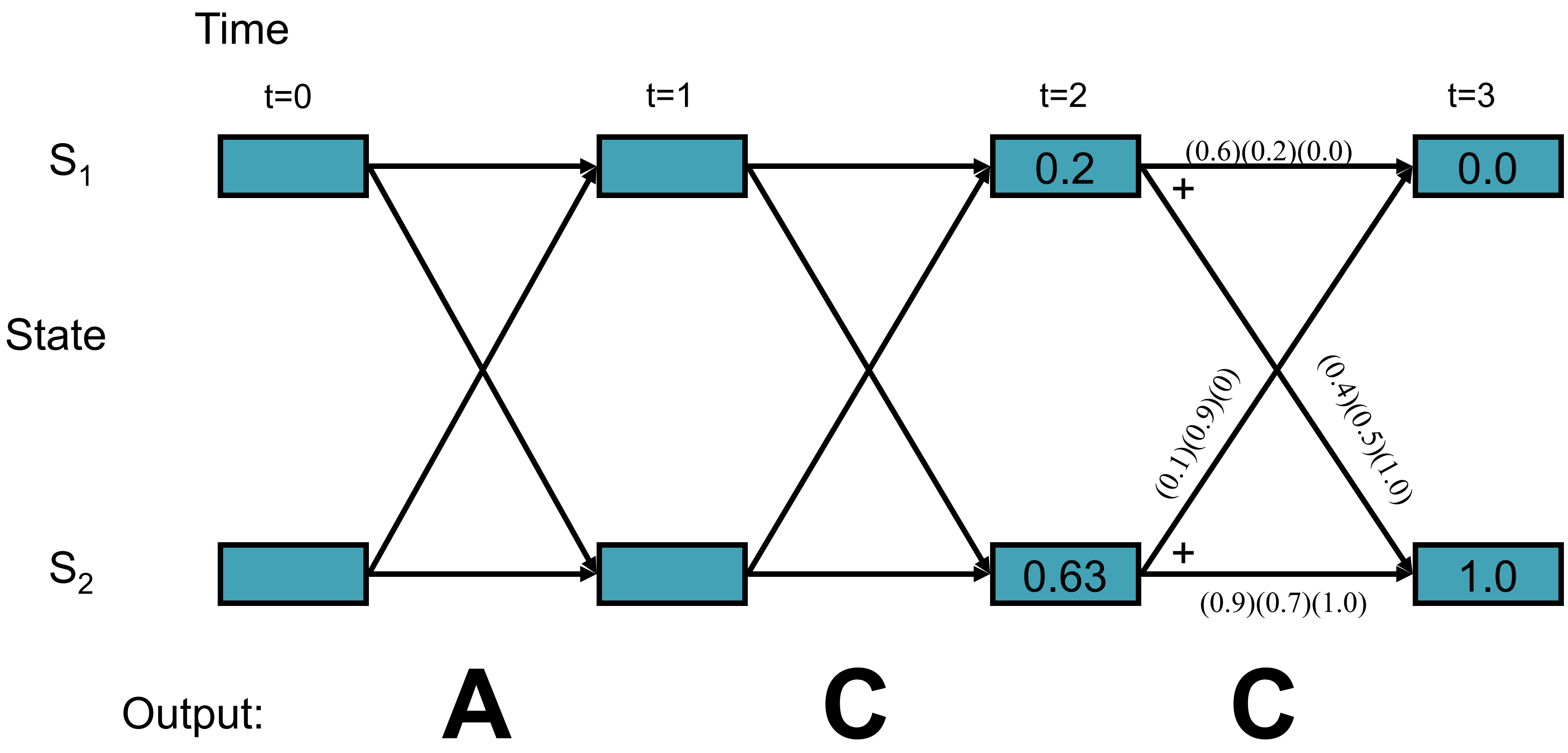
X_1^{T+1}

Step 3: the Backward algorithm

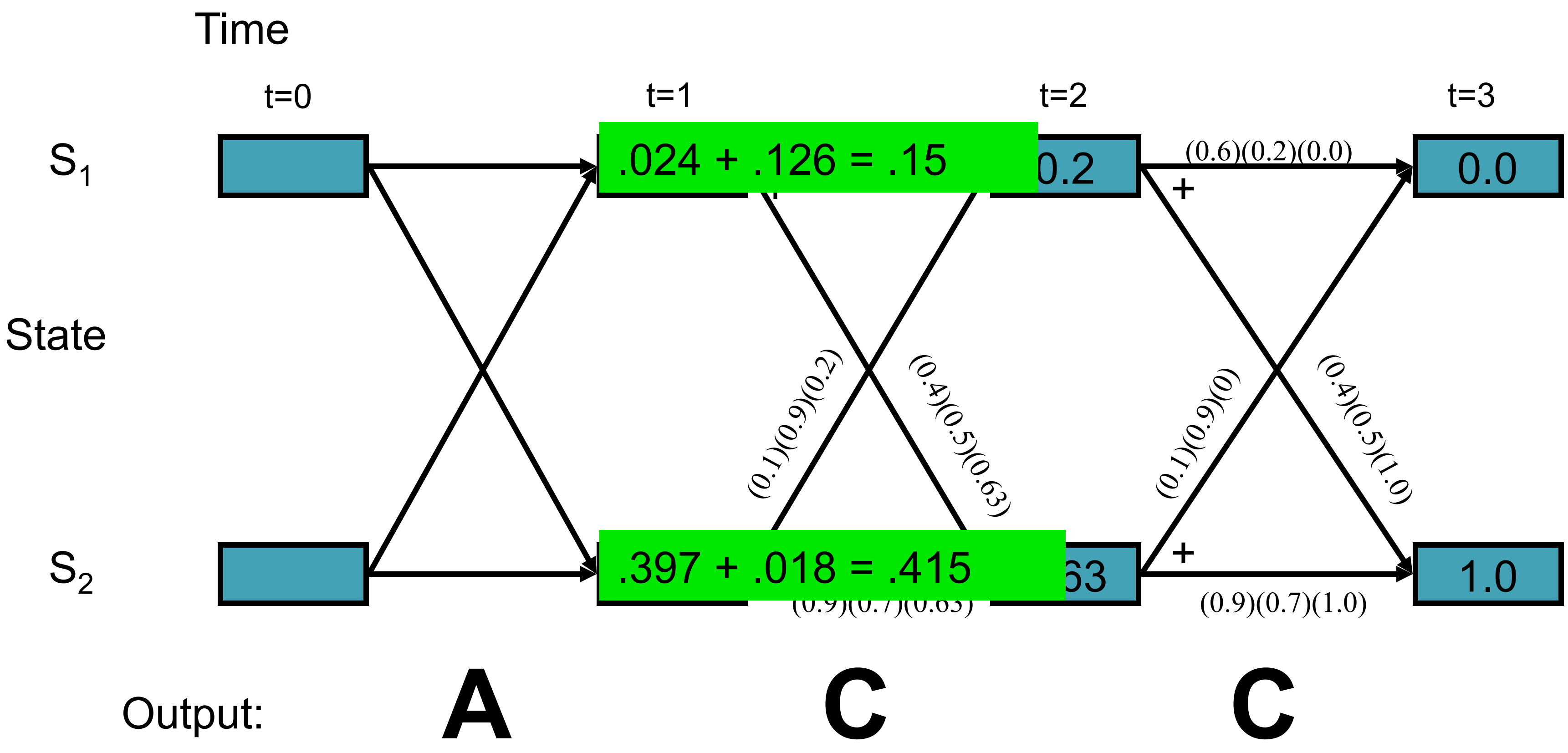
- Next we need to compute $\beta_i(t)$ using a Backward computation

$\beta_i(t)$ *is the probability that our HMM will generate the rest of the sequence* $y_{t+1}, y_{t+2}, \ldots, y_T$ *beginning in state i*

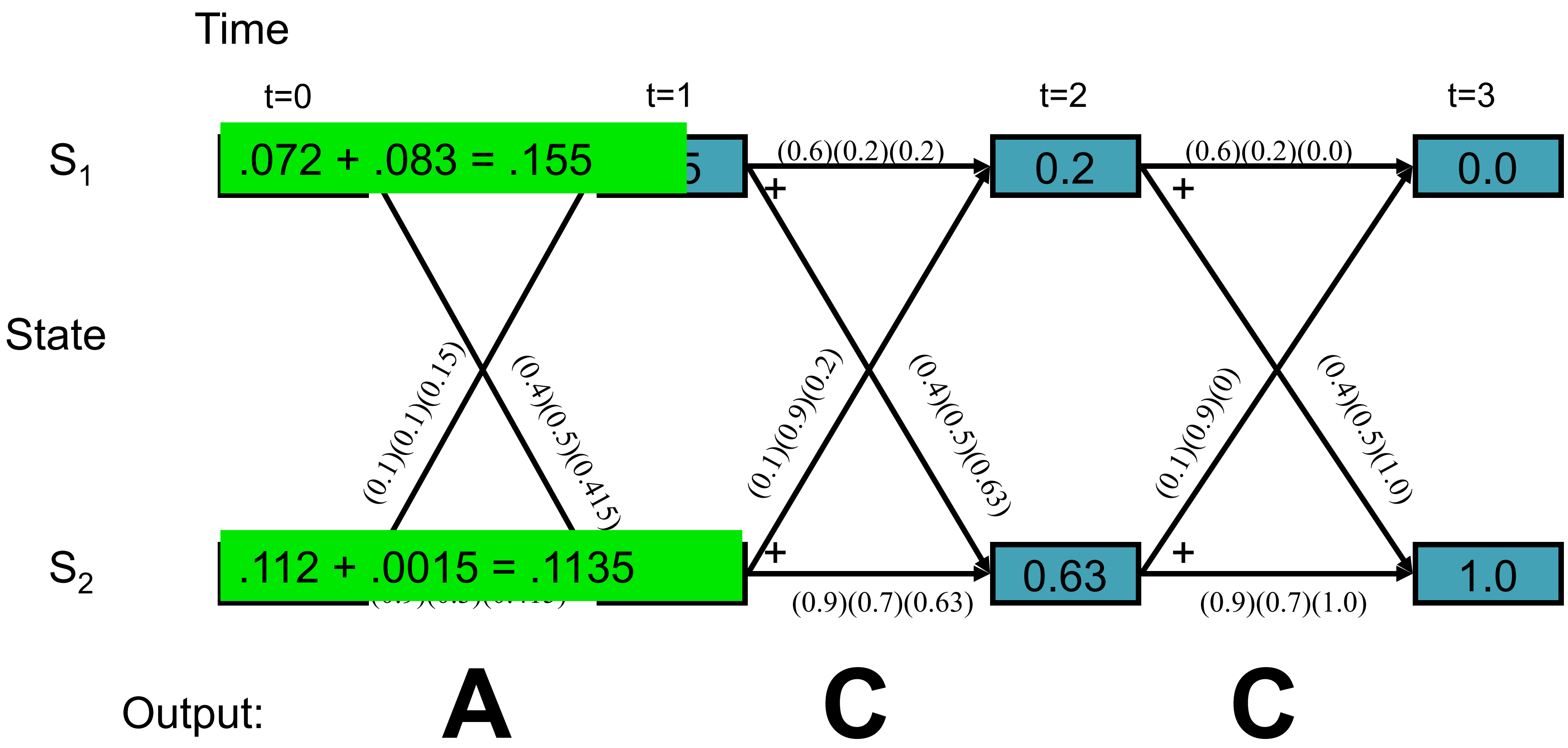
A trellis for the Backward Algorithm



A trellis for the Backward Algorithm (2)



A trellis for the Backward Algorithm (3)



Step 4: Re-estimate the probabilities

- After running the Forward and Backward algorithms once, we can re-estimate all the probabilities in the HMM
- α_{SF} is the prob. that the HMM generated the entire sequence
- Nice property of E-M: the value of α_{SF} never decreases; it converges to a local maximum
- We can read off α and β values from Forward and Backward trellises

Compute new transition probabilities

- γ is the probability of making transition i-j at time t, given the observed output
  - γ is dependent on data, plus it only applies for one time step; otherwise it is just like $a_{ij}(t)$

$$\gamma_{ij}(t) = P(X_t = i, X_{t+1} = j \mid y_1^T)$$

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1)a_{ij}b_{ij}(y_t)\beta_j(t)}{\alpha_{S_F}}$$

What is gamma?

- Sum γ over all time steps, then we get the expected number of times that the transition i-j was made while generating the sequence Y:

$$C_1 = \sum_{t=1}^{T} \gamma_{ij}(t)$$

How many times did we leave i?

- Sum γ over all time steps and all states that can follow i, then we get the expected number of times that the transition i-x as made for any state x:

$$C_2 = \sum_{t=1}^{T} \sum_{k} \gamma_{ik}(t)$$

Recompute transition probability

$$a_{ij} = \frac{C_1}{C_2}$$

In other words, probability of going from state i to j is estimated by counting how often we took it for our data (C1), and dividing that by how often we went from i to other states (C2)

Recompute output probabilities

- Originally these were $b_{ij}(k)$ values
- We need:
  - expected number of times that we made the transition i-j and emitted the symbol k
  - The expected number of times that we made the transition i-j

New estimate of b$_{ij}$(k)

$$b_{ij}(k) = \frac{\sum\limits_{t:y_t=k} \gamma_{ij}(t)}{\sum\limits_{t=1}^{T} \gamma_{ij}(t)}$$

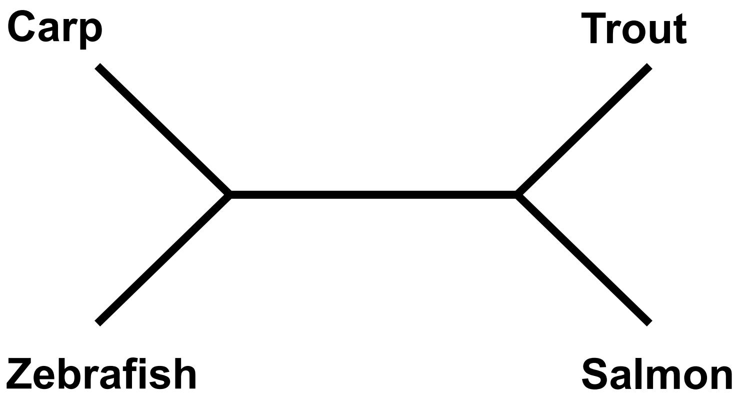
Step 5: Go to step 2

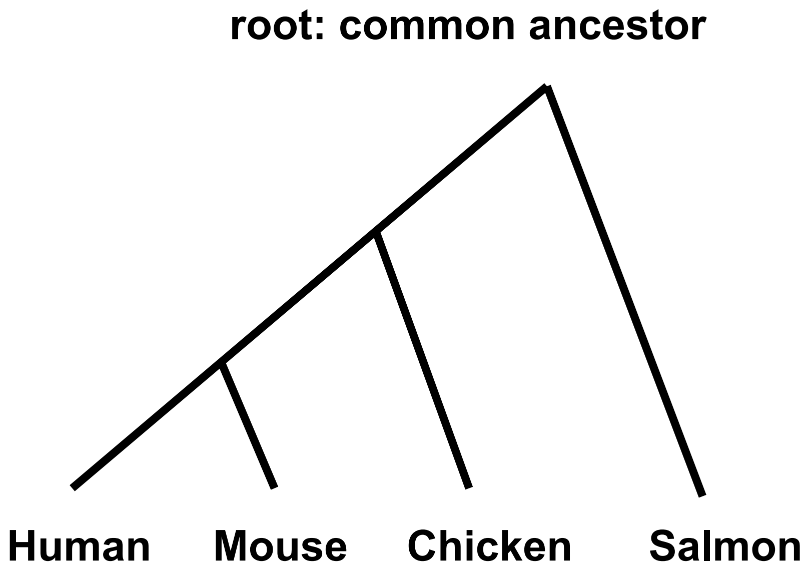
- Step 2 is Forward Algorithm
- Repeat entire process until the probabilities converge
  - Usually this is rapid, 10-15 iterations
- "Estimate-Maximize" because the algorithm first estimates probabilities, then maximizes them based on the data
- "Forward-Backward" refers to the two computationally intensive steps in the algorithm

Multiple Alignment
&
Phylogenetic Trees

bioalgorithms.info

Rooted and Unrooted Trees





Unrooted trees give no information about the order of events

Unrooted vs. Rooted Trees

- An unrooted tree gives information about the relationships between taxa.
  - $(2k-5)!/2^{k-3}(k-3)!$ trees with *k* leaves

- A rooted gene tree gives information about the order of events.
  - $(2k-3)!/2^{k-2}(k-2)!$ trees with *k* leaves

Multiple Alignment versus Pairwise Alignment

- **Up until now we have only tried to align two sequences.**

Multiple Alignment versus Pairwise Alignment

- **Up until now we have only tried to align two sequences.**
- **What about more than two? And what for?**

Multiple Alignment versus Pairwise Alignment

- Up until now we have only tried to align two sequences.
- What about more than two? And what for?
- A faint similarity between two sequences becomes significant if present in many
- Multiple alignments can reveal subtle similarities that pairwise alignments do not reveal

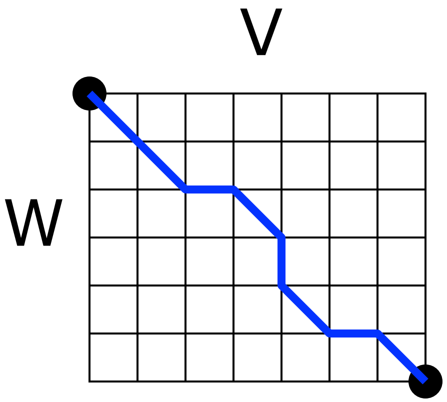
Generalizing the Notion of Pairwise Alignment

- Alignment of 2 sequences is represented as a 2-row matrix
- In a similar way, we represent alignment of 3 sequences as a 3-row matrix

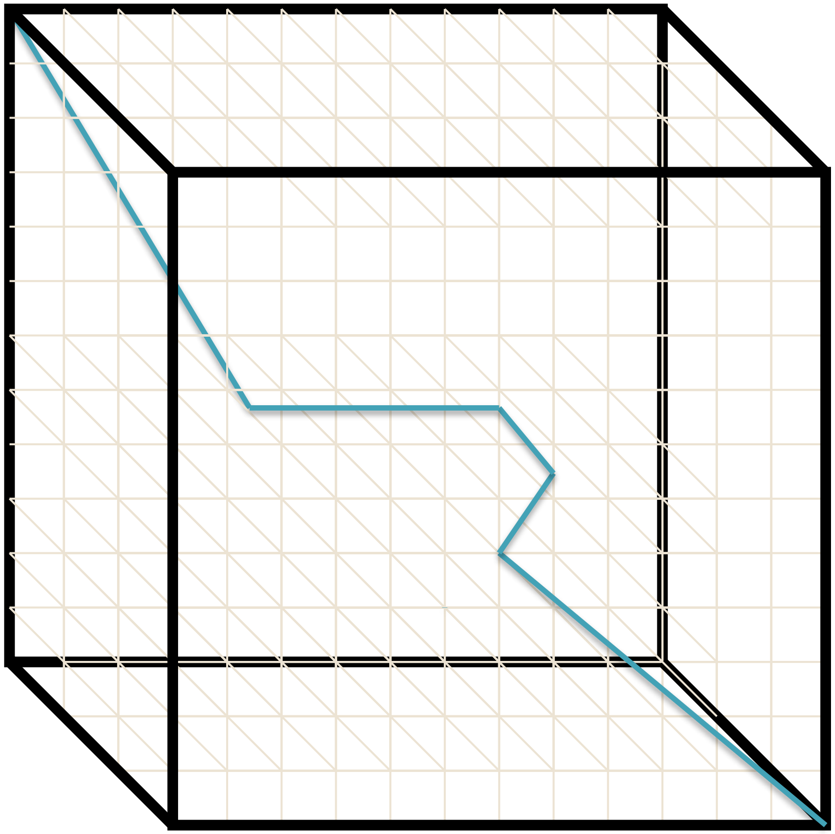
```
A T _ G C G _
A _ C G T _ A
A T C A C _ A
```

- Score: more conserved columns, better alignment

2-D vs 3-D Alignment Grid



2-D edit graph



3-D edit graph

Multiple Alignment: Dynamic Programming

- $$s_{i,j,k} = \max \begin{cases} s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) \\ s_{i-1,j-1,k} + \delta(v_i, w_j, \_) \\ s_{i-1,j,k-1} + \delta(v_i, \_, u_k) \\ s_{i,j-1,k-1} + \delta(\_, w_j, u_k) \\ s_{i-1,j,k} + \delta(v_i, \_, \_) \\ s_{i,j-1,k} + \delta(\_, w_j, \_) \\ s_{i,j,k-1} + \delta(\_, \_, u_k) \end{cases}$$

  cube diagonal: no indels

  face diagonal: one indel

  edge diagonal: two indels

- $\delta(x, y, z)$ is an entry in the 3-D scoring matrix

Multiple Alignment: Running Time

- For 3 sequences of length ***n***, the run time is $7n^3$; $O(n^3)$

- For ***k*** sequences, build a ***k***-dimensional Manhattan, with run time $(2^k-1)(n^k)$; $O(2^k n^k)$

- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to ***k*** sequences but it is impractical due to exponential running time
  - Apply heuristics to efficiently compute approx. solutions

ClustalW

- Popular multiple alignment tool today

- ‘W’ stands for ‘weighted’ (different parts of alignment are weighted differently).

- Three-step process

  1.) Construct pairwise alignments

  2.) Build Guide Tree

  3.) Progressive Alignment guided by the tree

Step 1: Pairwise Alignment

- Aligns each sequence again each other giving a similarity matrix
- Similarity = exact matches / sequence length (percent identity)

$$\begin{array}{c|cccc}
 & v_1 & v_2 & v_3 & v_4 \\
\hline
v_1 & - & & & \\
v_2 & .17 & - & & \\
v_3 & .87 & .28 & - & \\
v_4 & .59 & .33 & .62 & -
\end{array}$$

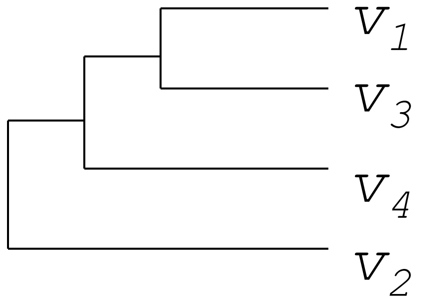
(.17 means 17 % identical)

Step 2: Guide Tree

- Create Guide Tree using the similarity matrix
  - ClustalW uses the neighbor-joining method
    - Iteratively merge the pair that are close to one another, but far from others
  - Guide tree roughly reflects evolutionary relations

Step 2: Guide Tree (cont'd)

	v_1	v_2	v_3	v_4
v_1	–			
v_2	.17	–		
v_3	.87	.28	–	
v_4	.59	.33	.62	–



Calculate:

$v_{1,3}$ = alignment (v_1, v_3)

$v_{1,3,4}$ = alignment$((v_{1,3}),v_4)$

$v_{1,2,3,4}$ = alignment$((v_{1,3,4}),v_2)$

Step 3: Progressive Alignment

- Start by aligning the two most similar sequences
- Following the guide tree, add in the next sequences, aligning to the existing alignment
- Insert gaps as necessary

```
FOS_RAT         PEEMSVTS-LDLTGGLPEATTPESEEAFTLPLLNDPEPK-PSLEPVKNISNMELKAEPFD
FOS_MOUSE       PEEMSVAS-LDLTGGLPEASTPESEEAFTLPLLNDPEPK-PSLEPVKSISNVELKAEPFD
FOS_CHICK       SEELAAATALDLG----APSPAAAEEAFALPLMTEAPPAVPPKEPSG--SGLELKAEPFD
FOSB_MOUSE      PGPGPLAEVRDLPG-----STSAKEDGFGWLLPPPPPPP-----------------LPFQ
FOSB_HUMAN      PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPPP-----------------LPFQ
                .   . :   ** .     :..  *:.*   *   . *                   **:
```

Dots and stars show how well-conserved a column is.