

# Gene Finding & HMMs

Michael Schatz

Bioinformatics Lecture 4  
Quantitative Biology 2014





# Outline

## 1. 'Semantic' Sequence Analysis

1. Prokaryotic Gene Finding
2. Eukaryotic Gene Finding

## 2. Review

1. Indexing & Exact Match
2. Sequence Alignment & Dynamic Programming
3. Graphs & Genome Assembly
4. Gene Finding & HMMs
5. Data Structures and Algorithms

# Gene Prediction: Computational Challenge

aatgcatgcggctatgctaataatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaa  
tgcatgcggctatgcaagctgggatccgatgactatgctaagctgggatccgatgacaatgcatgcggctatgc  
taatgaatggtcttgggatttaccttgggaatgctaagctgggatccgatgacaatgcatgcggctatgctaata  
atggtcttgggatttaccttgggaatgctaataatgcatgcggctatgctaagctgggatccgatgacaatgcatgc  
ggctatgctaataatgcatgcggctatgcaagctgggatccgatgactatgctaagctgcggctatgctaataatgcatg  
cggctatgctaagctgggatccgatgacaatgcatgcggctatgctaataatgcatgcggctatgcaagctgggatc  
ctgcggctatgctaataatggtcttgggatttaccttgggaatgctaagctgggatccgatgacaatgcatgcgg  
ctatgctaataatggtcttgggatttaccttgggaatgctaataatgcatgcggctatgctaagctgggaatgcatg  
cggctatgctaagctgggatccgatgacaatgcatgcggctatgctaataatgcatgcggctatgcaagctgggatc  
cgatgactatgctaagctgcggctatgctaataatgcatgcggctatgctaagctcatgcggctatgctaagctggg  
aatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaataatgcatgcggctatgcaag  
ctgggatccgatgactatgctaagctgcggctatgctaataatgcatgcggctatgctaagctcggctatgctaata  
atggtcttgggatttaccttgggaatgctaagctgggatccgatgacaatgcatgcggctatgctaataatggtc  
ttgggatttaccttgggaatgctaataatgcatgcggctatgctaagctgggaatgcatgcggctatgctaagctgg  
gatccgatgacaatgcatgcggctatgctaataatgcatgcggctatgcaagctgggatccgatgactatgctaagc  
tgcggctatgctaataatgcatgcggctatgctaagctcatgcgg

# Gene Prediction: Computational Challenge

aatgcatgaggctatgctaagcatgaggctatgctaagctgggatccgatgacaatgcatgaggctatgcta  
tgcatgaggctatgcaagctgggatccgatgactatgctaagctgggatccgatgacaatgcatgaggctatgc  
taatgaatggtcttgggatttaccttgggaatgctaagctgggatccgatgacaatgcatgaggctatgcta  
atggtcttgggatttaccttgggaatgctaagcatgaggctatgctaagctgggatccgatgacaatgcatgc  
ggctatgctaagcatgaggctatgcaagctgggatccgatgactatgctaagctgaggctatgctaagcatg  
aggctatgctaagctgggatccgatgacaatgcatgaggctatgctaagcatgaggctatgcaagctgggatc  
ctgaggctatgctaagcatggtcttgggatttaccttgggaatgctaagctgggatccgatgacaatgcatgagg  
ctatgctaagcatggtcttgggaaggctatgctaagctgggaatgcatg  
aggctatgctaagctgggatccgatgcatgaggctatgcaagctgggatc  
cgatgactatgctaagctgaggctatgctaagcatgaggctatgctaagctcatgaggctatgctaagctggg  
aatgcatgaggctatgctaagctgggatccgatgacaatgcatgaggctatgctaagcatgaggctatgcaag  
ctgggatccgatgactatgctaagctgaggctatgctaagcatgaggctatgctaagctcaggctatgctaag  
atggtcttgggatttaccttgggaatgctaagctgggatccgatgacaatgcatgaggctatgctaagcatggtc  
ttgggatttaccttgggaatgctaagcatgaggctatgctaagctgggaatgcatgaggctatgctaagctgg  
gatccgatgacaatgcatgaggctatgctaagcatgaggctatgcaagctgggatccgatgactatgctaagc  
tgaggctatgctaagcatgaggctatgctaagctcatgagg

Gene!



# Bacterial Gene Finding and Glimmer

(also Archaeal and viral gene finding)

Arthur L. Delcher and Steven Salzberg  
Center for Bioinformatics and Computational Biology  
Johns Hopkins University School of Medicine

# Outline

- A (very) brief overview of microbial gene-finding
- Evolution of Glimmer
  - Glimmer 1
    - Interpolated Markov Model (IMM)
  - Glimmer 2
    - Interpolated Context Model (ICM)
  - Glimmer 3
    - Reducing false positives with a DP alg for selection
    - Improving coding initiation site predictions

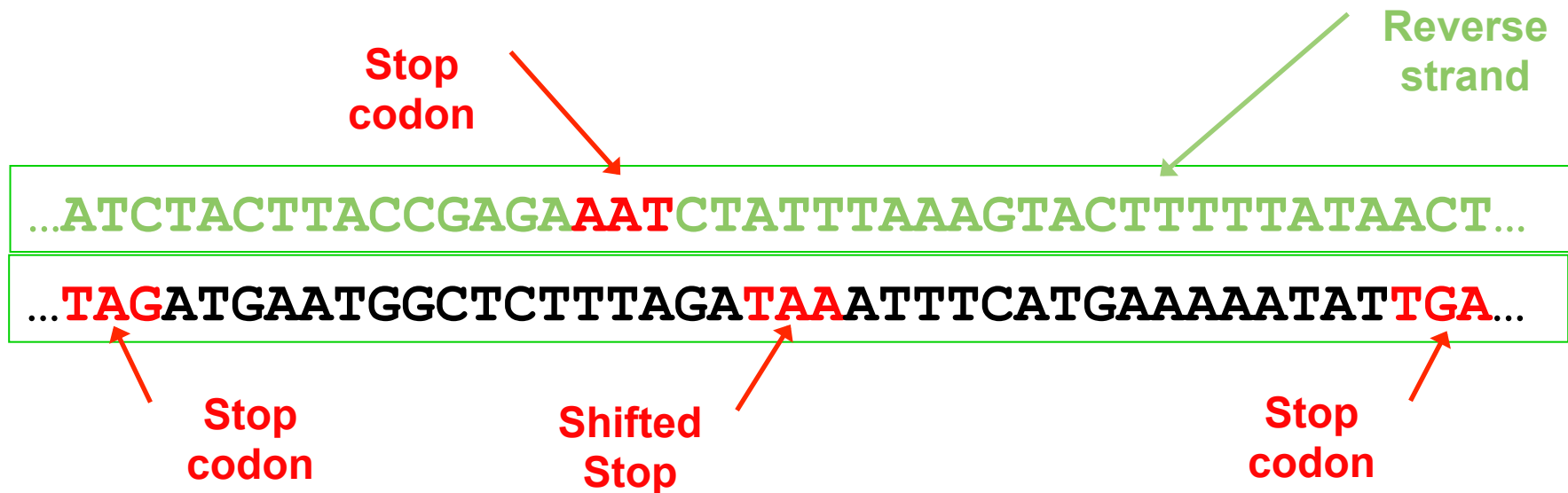
# Step One

- Find open reading frames (ORFs).



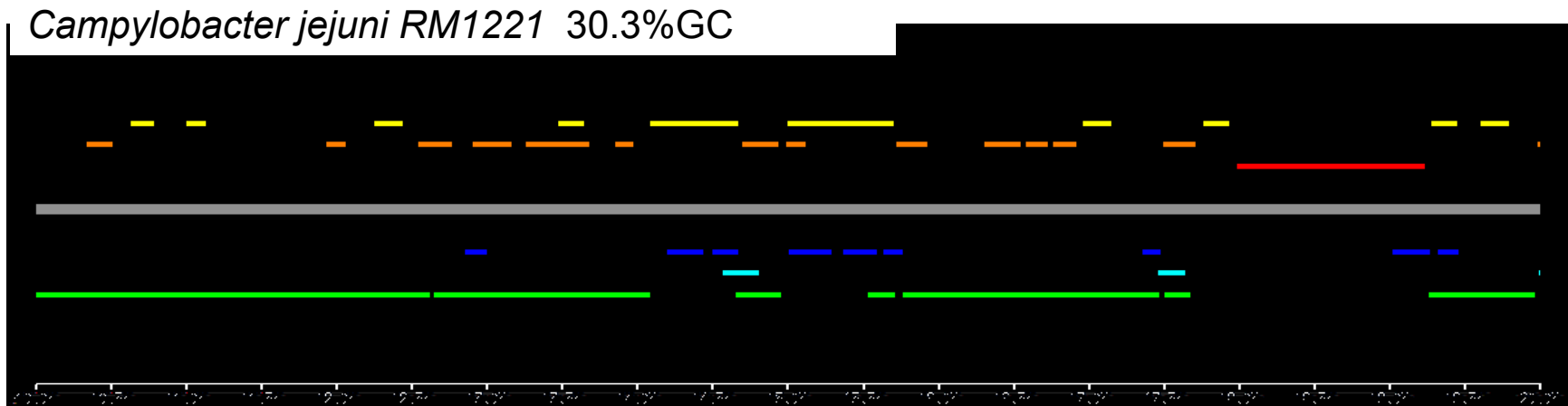
# Step One

- Find open reading frames (ORFs).



- But ORFs generally overlap ...



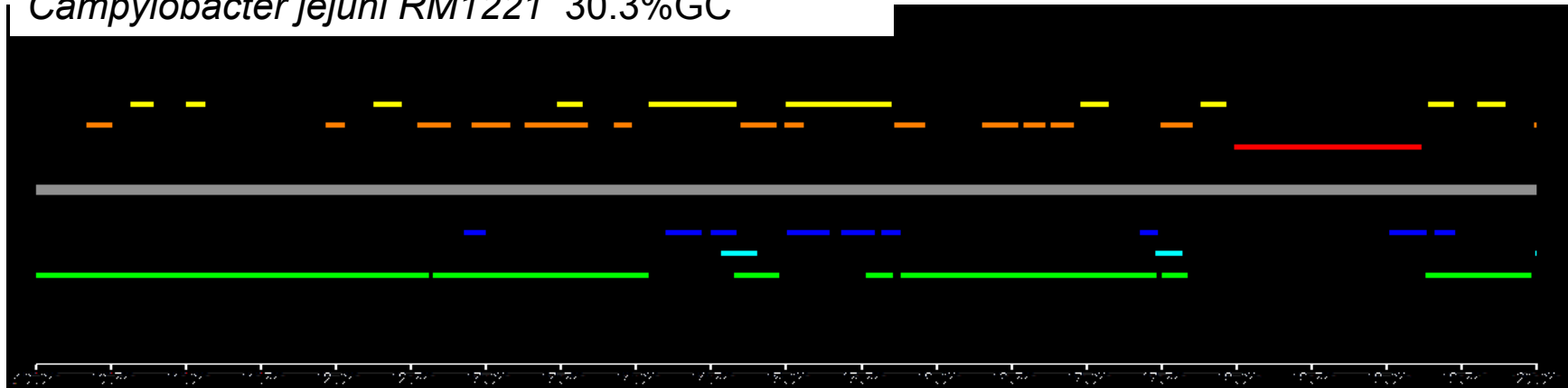


All ORFs longer than 100bp on both strands shown  
- color indicates reading frame  
Longest ORFs likely to be protein-coding genes

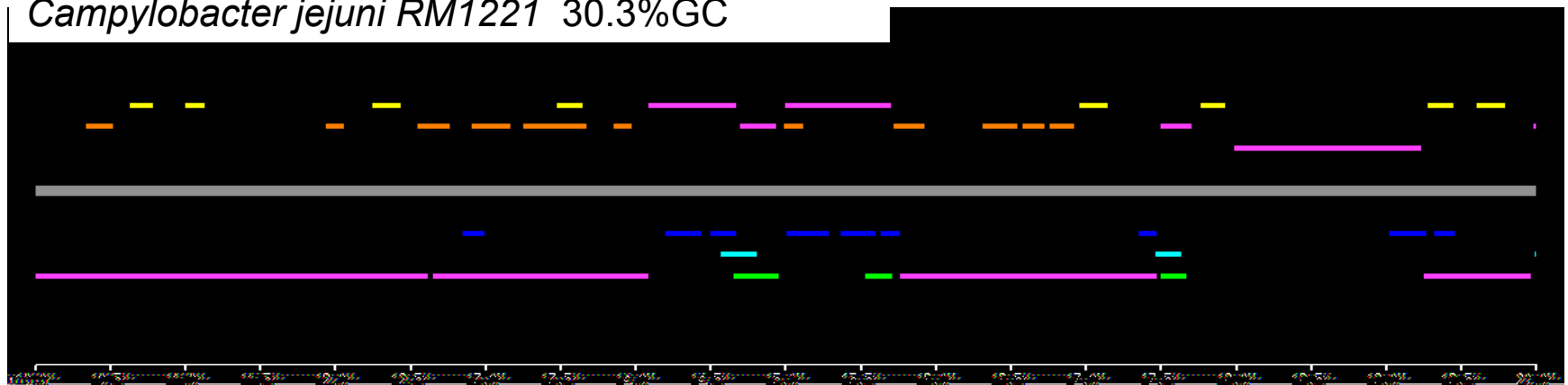
Note the low GC content

All genes are ORFs but not all ORFs are genes

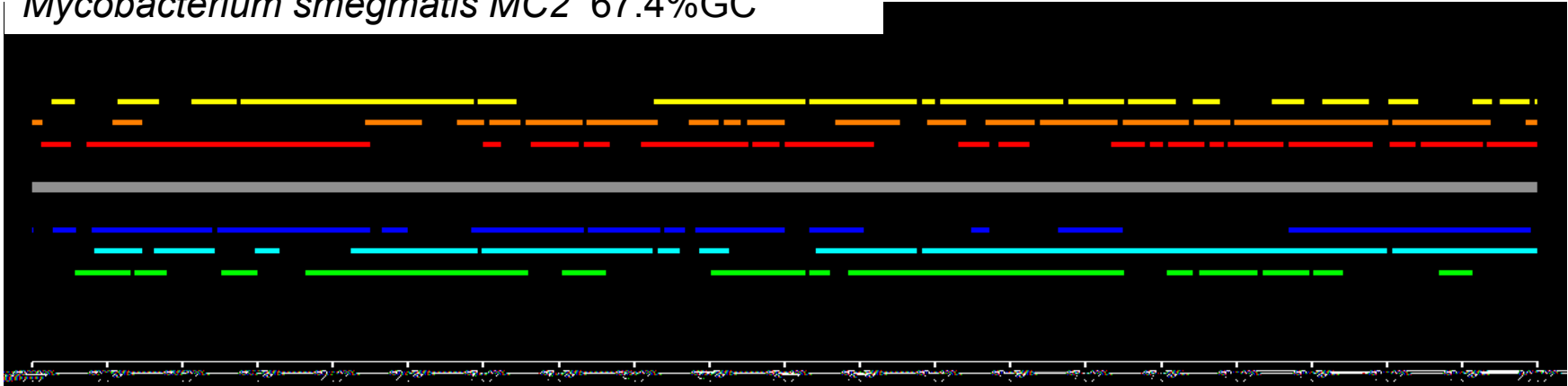
*Campylobacter jejuni* RM1221 30.3%GC



*Campylobacter jejuni* RM1221 30.3%GC

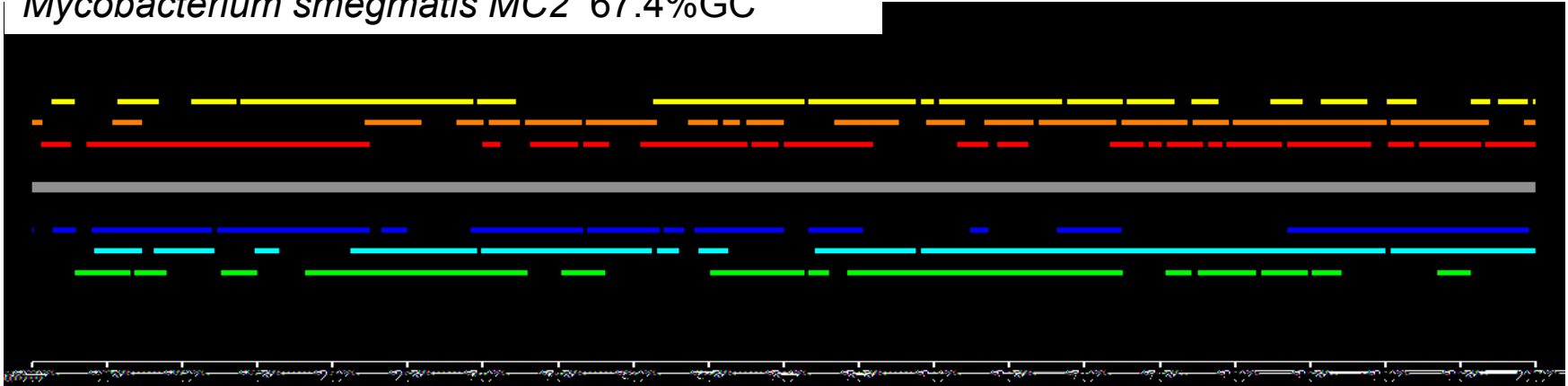


*Mycobacterium smegmatis* MC2 67.4%GC

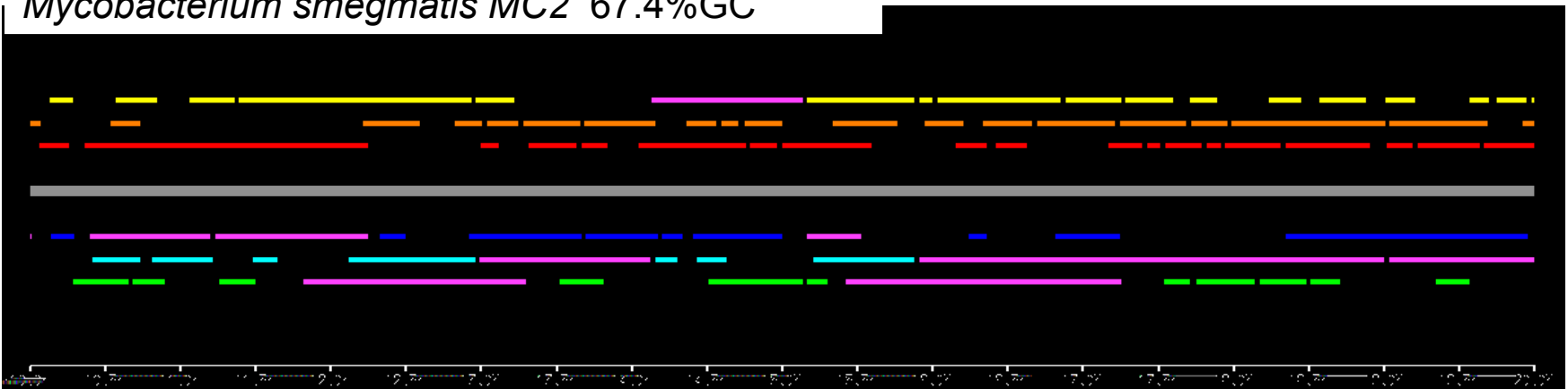


Note what happens in a high-GC genome

*Mycobacterium smegmatis* MC2 67.4%GC



*Mycobacterium smegmatis* MC2 67.4%GC



# Probabilistic Methods

- Create models that have a probability of generating any given sequence.
  - Evaluate gene/non-genome models against a sequence
- Train the models using examples of the types of sequences to generate.
  - Use RNA sequencing, homology, or “obvious” genes
- 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

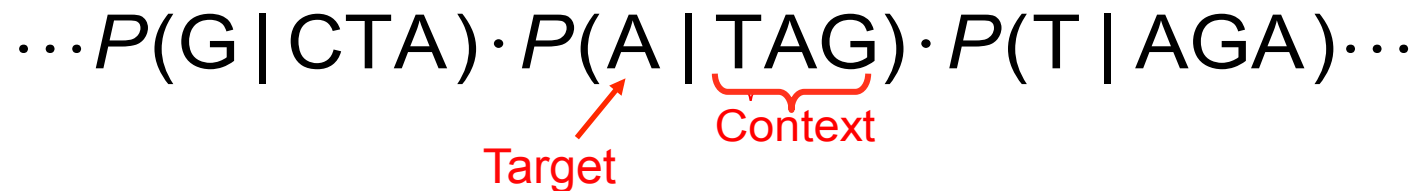
- $k^{\text{th}}$ -order Markov model bases the probability of an event on the preceding  $k$  events.
- Example: With a 3<sup>rd</sup>-order model the probability of this sequence:

...CTAGAT...



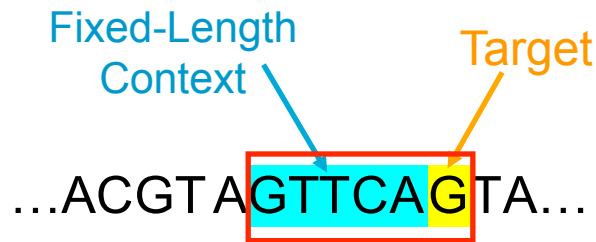
- would be:

... $P(G | CTA) \cdot P(A | TAG) \cdot P(T | AGA) \dots$



# 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

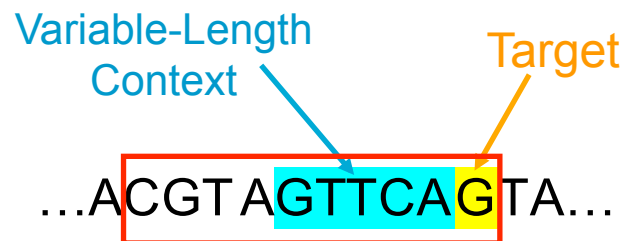
Salzberg, Delcher, Kasif & White, *NAR* 26, 1998.

- 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 **ggta** the next position might depend on previous 3 bases **ta**.
  - But for context **catta** all 5 bases might be used.



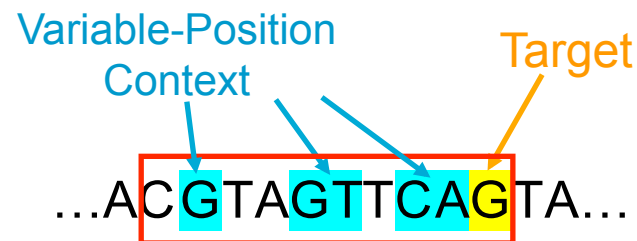
# IMMs vs Fixed-Order Models

- 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.



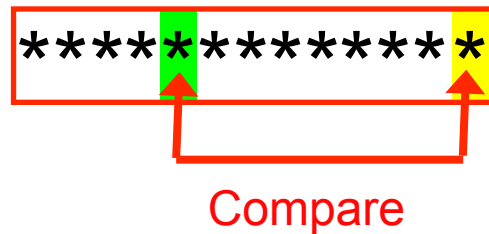
# Interpolated Context Model (ICM)

- 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

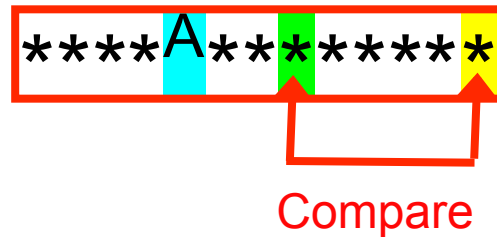


- Choose position with max mutual information

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

# 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.

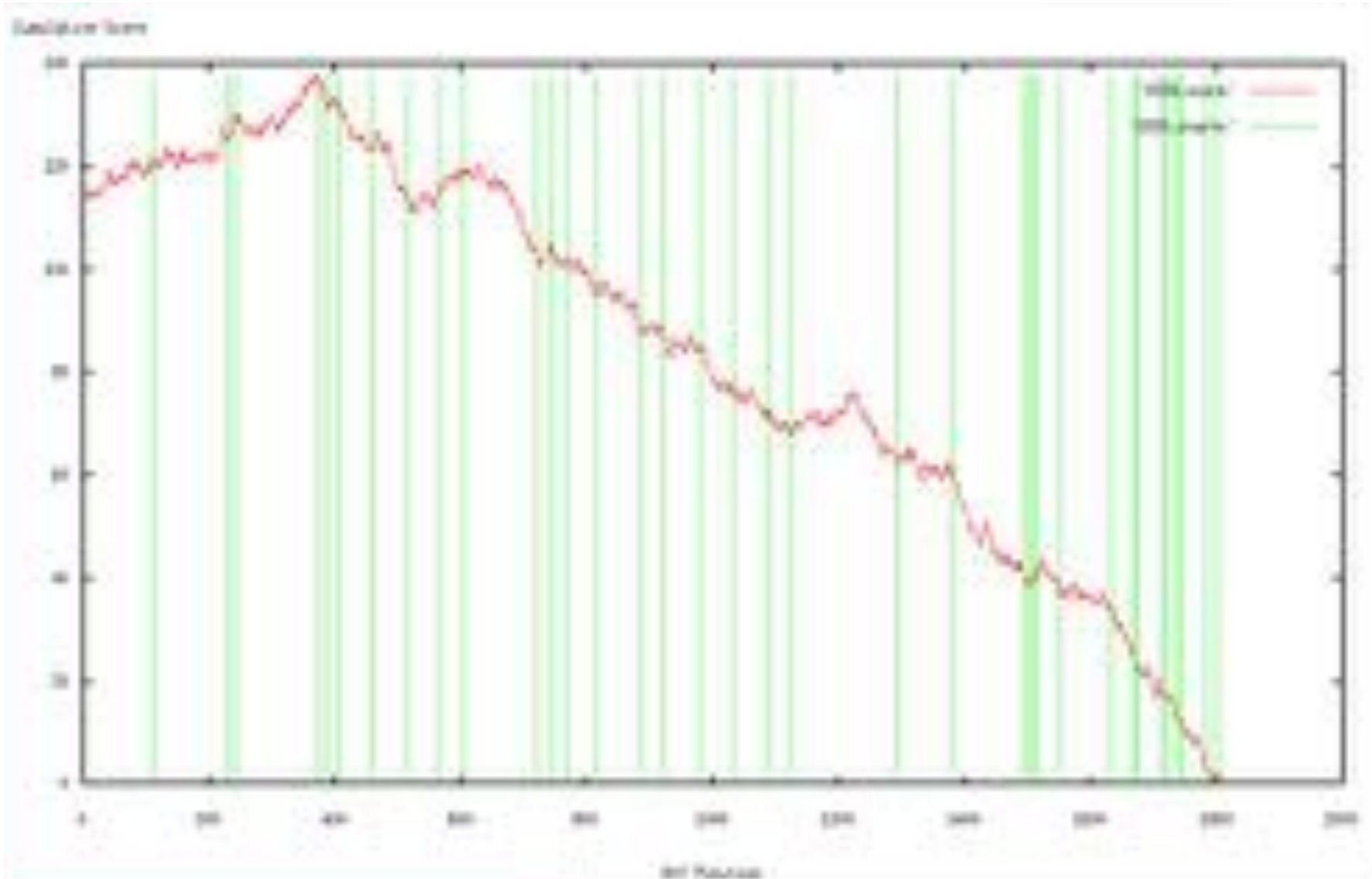
# Glimmer3

- Uses a dynamic programming algorithm to choose the highest-scoring set of orfs and start sites.
  - Similar to the longest increasing subsequence problem we saw before
- 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 8.** *Chrom3* prediction accuracy compared to other gene-finding systems. Chrom3 predictions are in the preceding table and each entry is the Chrom3 value across the corresponding value for the other gene finder. Genefinder systems were taken from the *Genefinder* (1999) files downloaded from NCBI. EzyGene 1.2 results were downloaded from <http://servers.biol.uzh.ch/genefinder/ezygene1.2/>. Genefinder results were obtained from the server at <http://www.genefinder.org/Genefinder/genefinder.cgi>. None of these systems had results for 11 proteins, which were a non-standard translation code. NA entries indicate results that were not available for download.

| Genome               |         | v1, Genefinder |      |       | v2, EzyGene 1.2 |     |       | v3, Genefinder |      |       |
|----------------------|---------|----------------|------|-------|-----------------|-----|-------|----------------|------|-------|
| Organism             | F-Genes | F-Match        | V&F  | Score | F-Match         | V&F | Score | F-Match        | V&F  | Score |
| <i>A. nidulans</i>   | 1103    | +1             | -31  | -81   | +5              | -16 | +108  | +1             | -31  | -71   |
| <i>A. nidulans</i>   | 1102    | -2             | -48  | -134  | +10             | -40 | +172  | +1             | -402 | -142  |
| <i>A. nidulans</i>   | 1175    | +1             | +181 | +27   | +10             | -18 | +188  | +1             | -39  | +181  |
| <i>C. elegans</i>    | 1292    | +1             | +11  | +19   | +10             | -18 | +180  | +1             | -34  | +29   |
| <i>C. elegans</i>    | 1294    | -2             | +177 | +20   | -2              | -8  | -20   | -1             | -44  | +18   |
| <i>E. coli</i>       | 300     | -31            | +11  | +188  | +10             | +18 | +187  | -31            | -39  | +180  |
| <i>G. gallus</i>     | 1201    | +12            | +103 | +14   | +2              | -2  | +48   | +14            | +41  | +46   |
| <i>H. sapiens</i>    | 913     | -1             | -4   | -31   | +4              | -8  | +148  | -1             | -8   | -41   |
| <i>P. falciparum</i> | 889     | +17            | +188 | +29   | NA              | NA  | NA    | +17            | +179 | +48   |
| <i>S. cerevisiae</i> | 1112    | +7             | +181 | +103  | +10             | +48 | +180  | -1             | +140 | +180  |
| <i>T. gondii</i>     | 1001    | +1             | -31  | -81   | NA              | NA  | NA    | +1             | +141 | -81   |
| <i>T. pallidum</i>   | 479     | +1             | -8   | +18   | +18             | -8  | +174  | -1             | -38  | +18   |
| Average              |         | +1             | +89  | +12   | +10             | -2  | +148  | +1             | +48  | +29   |

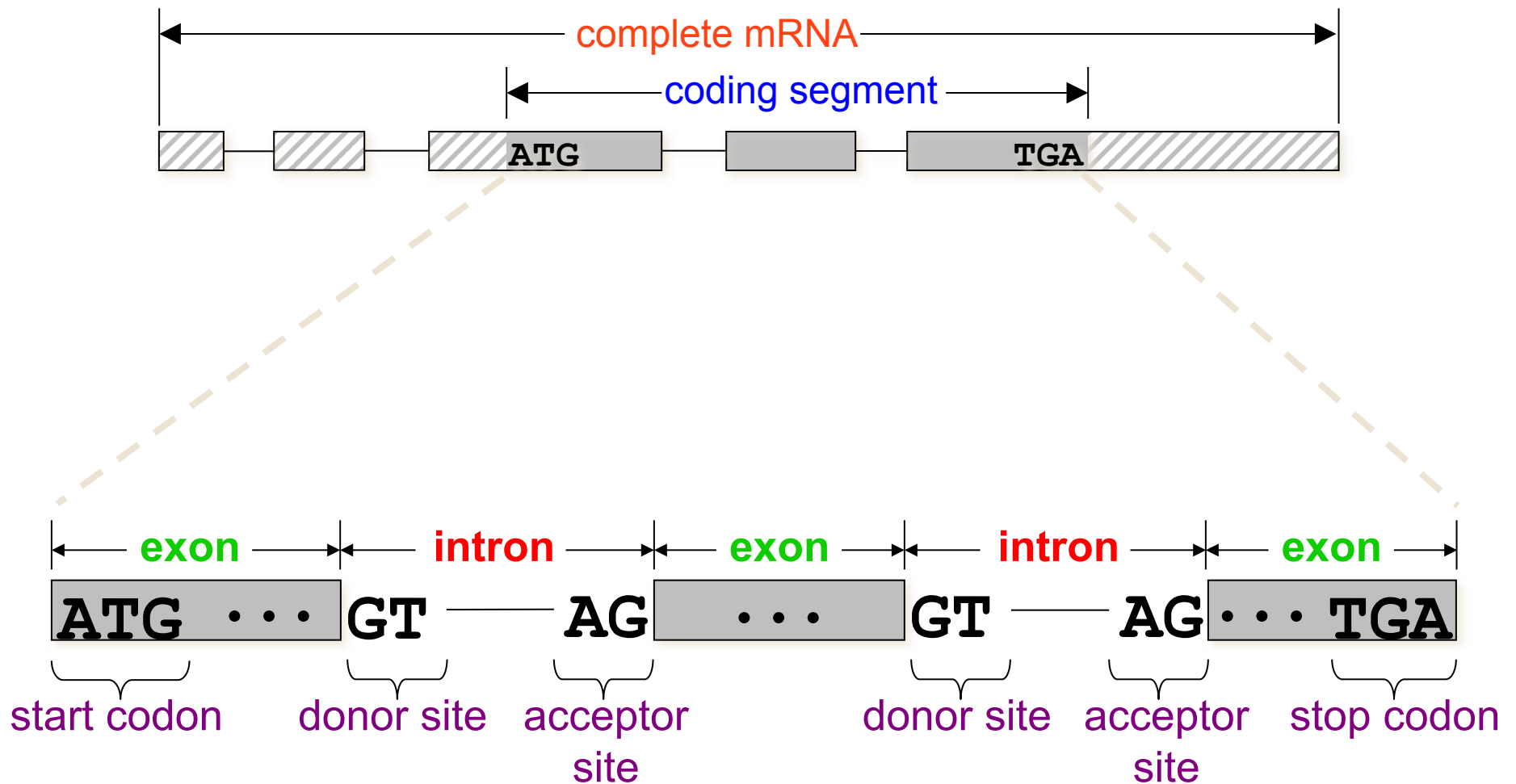


# Overview of Eukaryotic Gene Prediction

CBB 231 / COMPSCI 261

*W.H. Majoros*

# Eukaryotic Gene Syntax

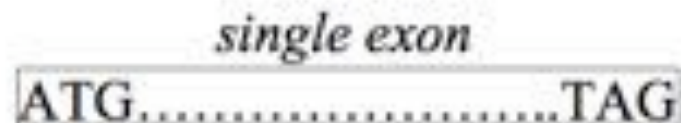
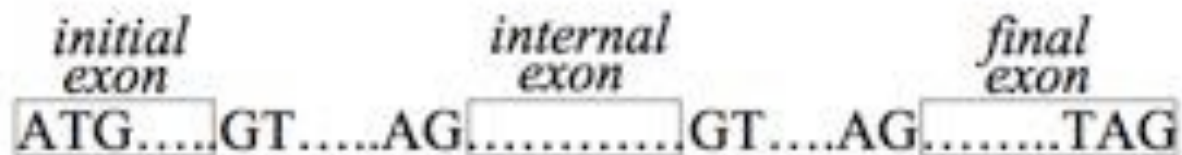


Regions of the gene outside of the CDS are called **UTR**'s (*untranslated regions*), and are mostly ignored by gene finders, though they are important for regulatory functions.

# Types of Exons

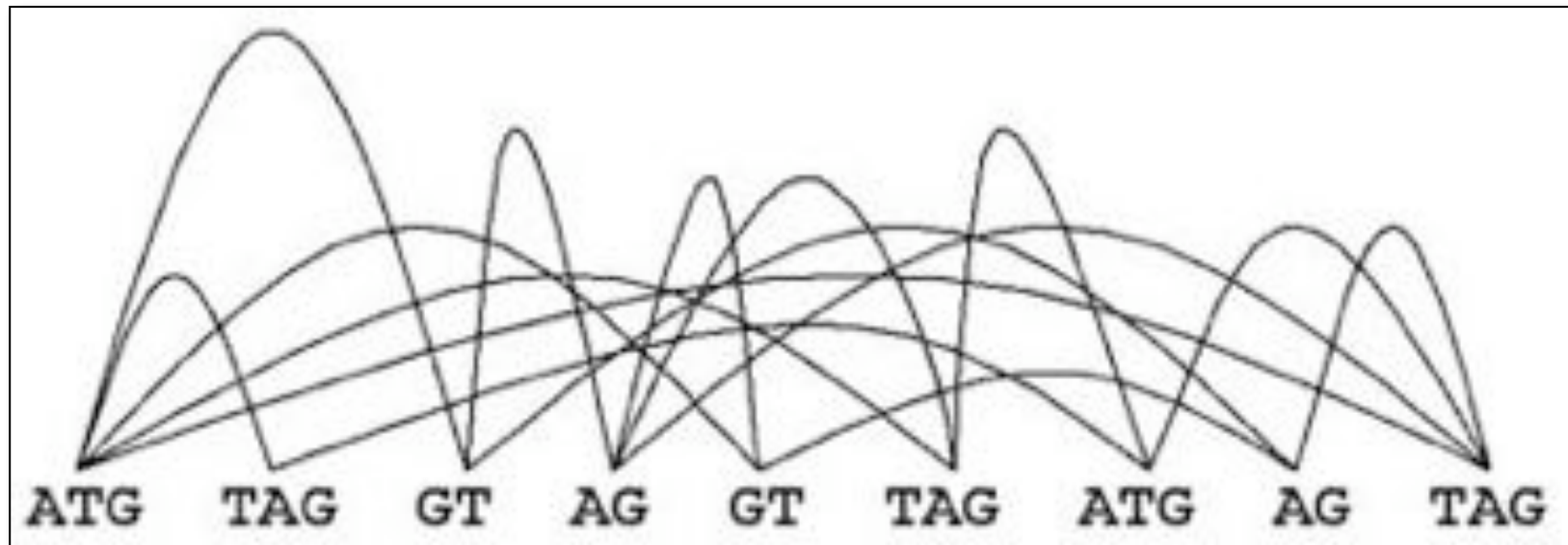
Three types of exons are defined, for convenience:

- *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

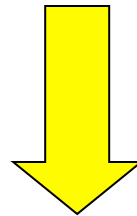
After identifying the most promising (i.e., highest-scoring) signals in an input sequence, we can apply the gene syntax rules to connect these into an *ORF graph*:



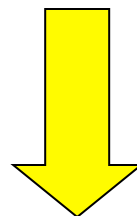
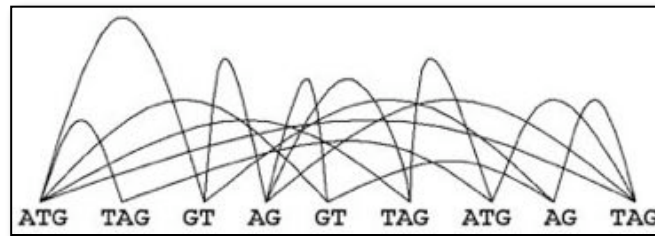
An ORF graph represents all possible *gene parses* (and their scores) for a given set of putative signals. A *path* through the graph represents a single gene parse.

# 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



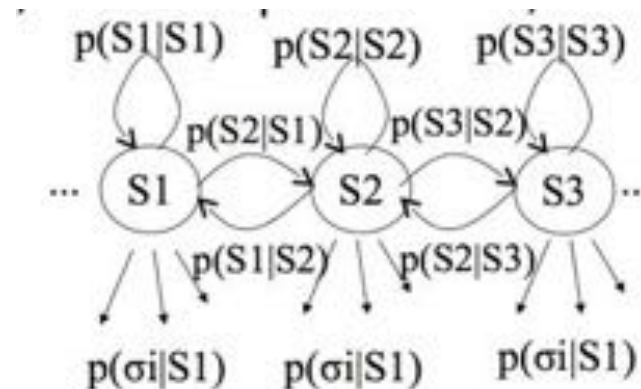


# Hidden Markov Models (HMMs)

Steven Salzberg  
JHU

# Why Hidden?

- Similar to Markov models used for prokaryotic gene finding, but system may transition between multiple models called states (gene/non-gene, intergenic/exon/intron)
- Observers can see the emitted symbols of an HMM (i.e., nucleotides) 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.



AAAGCATGCATTTAACGTGAGCACAAATAGATTACA



# 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

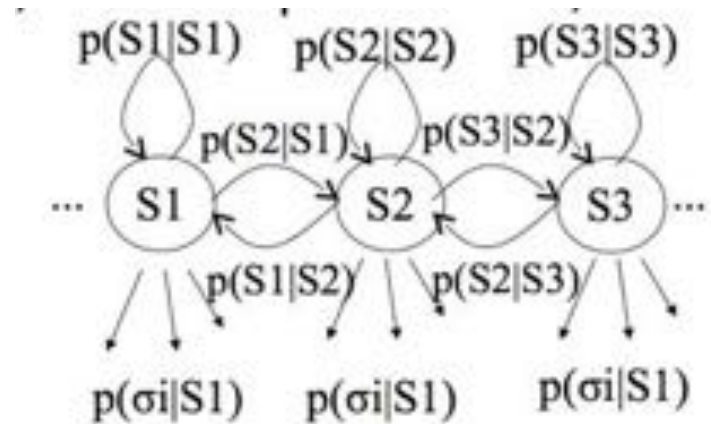
- $E=\{H,T\}$  for coin tossing
    - $E=\{1,2,3,4,5,6\}$  for dice tossing
    - $E=\{A,C,G,T\}$  for DNA

- State-specific emission probabilities

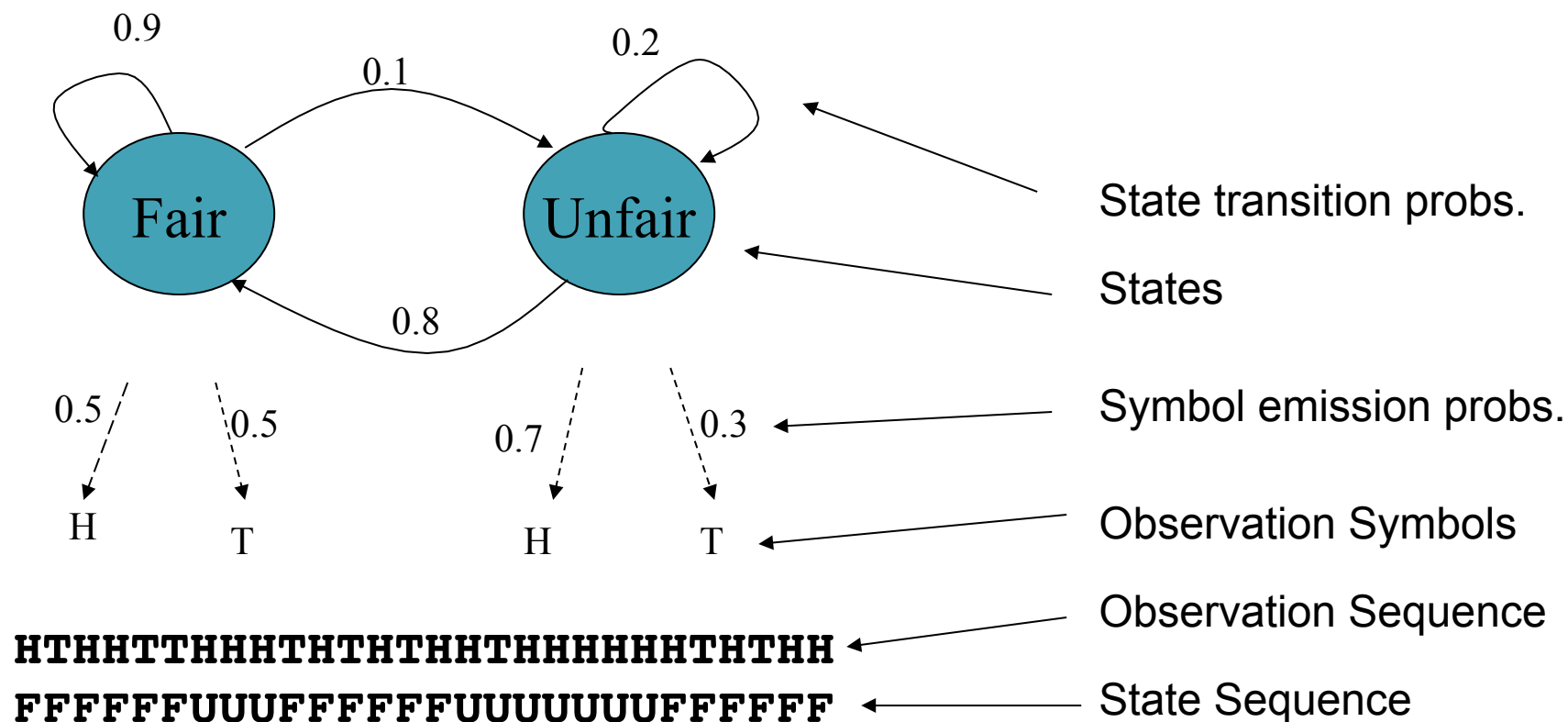
- $P(H \mid \text{Fair}) = .5, P(T \mid \text{Fair}) = .5, P(H \mid \text{Biased}) = .9, P(T \mid \text{Biased}) = .1$
    - $P(A \mid \text{Gene}) = .9, P(A \mid \text{Not Gene}) = .1 \dots$

- A probability of taking a transition

- $P(s_i=\text{Fair} \mid s_{i-1}=\text{Fair}) = .9, P(s_i=\text{Bias} \mid s_{i-1} = \text{Fair}) = .1$
    - $P(s_i=\text{Exon} \mid s_{i-1}=\text{Intergenic}), \dots$



# HMM Example - Casino Coin



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

# Three classic HMM problems

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

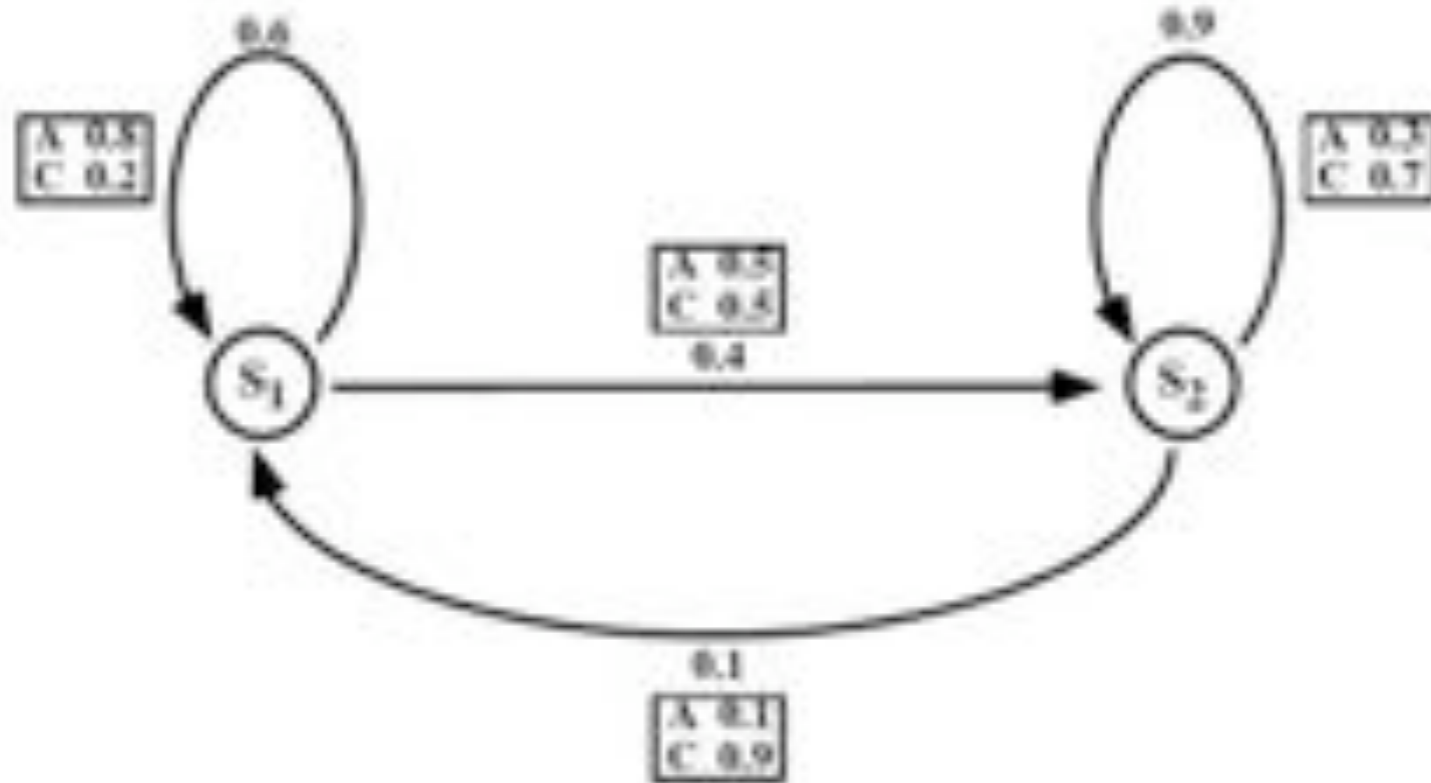
# Three classic HMM problems

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

# Solving the Evaluation problem: The Forward algorithm

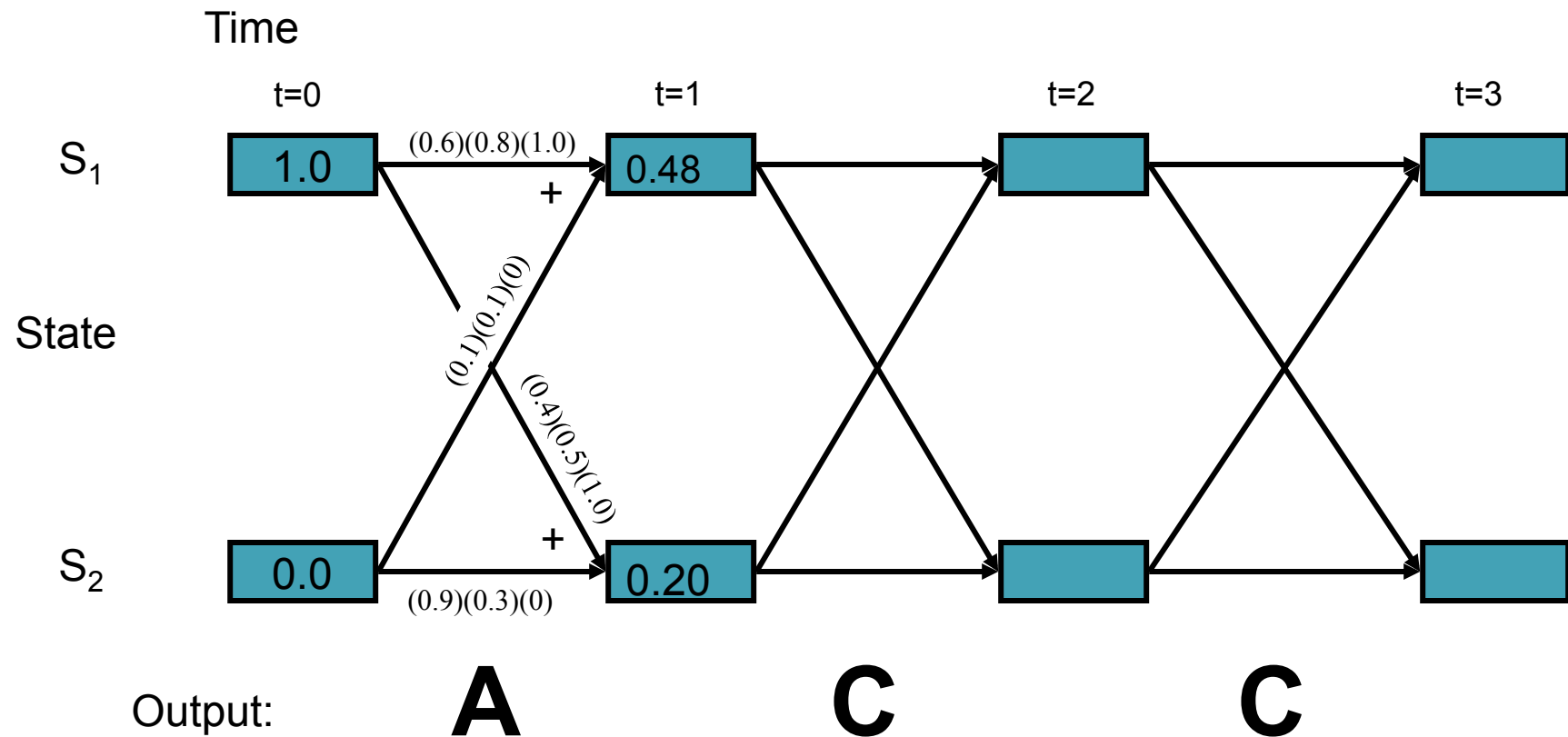
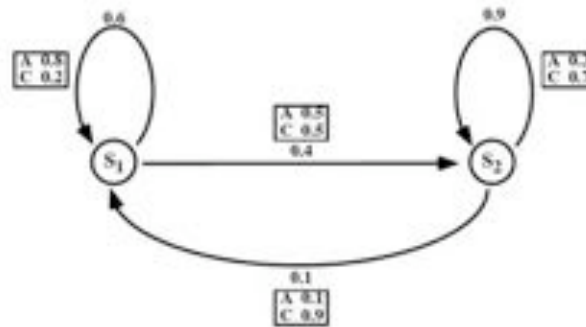
- To solve the Evaluation problem (probability that the model generated the sequence), we use the HMM and the data to build a *trellis*
- Filling in the trellis will give tell us the probability that the HMM generated the data by finding all possible paths that could do it
  - Especially useful to evaluate from which models, a given sequence is most likely to have originated

# Our sample HMM

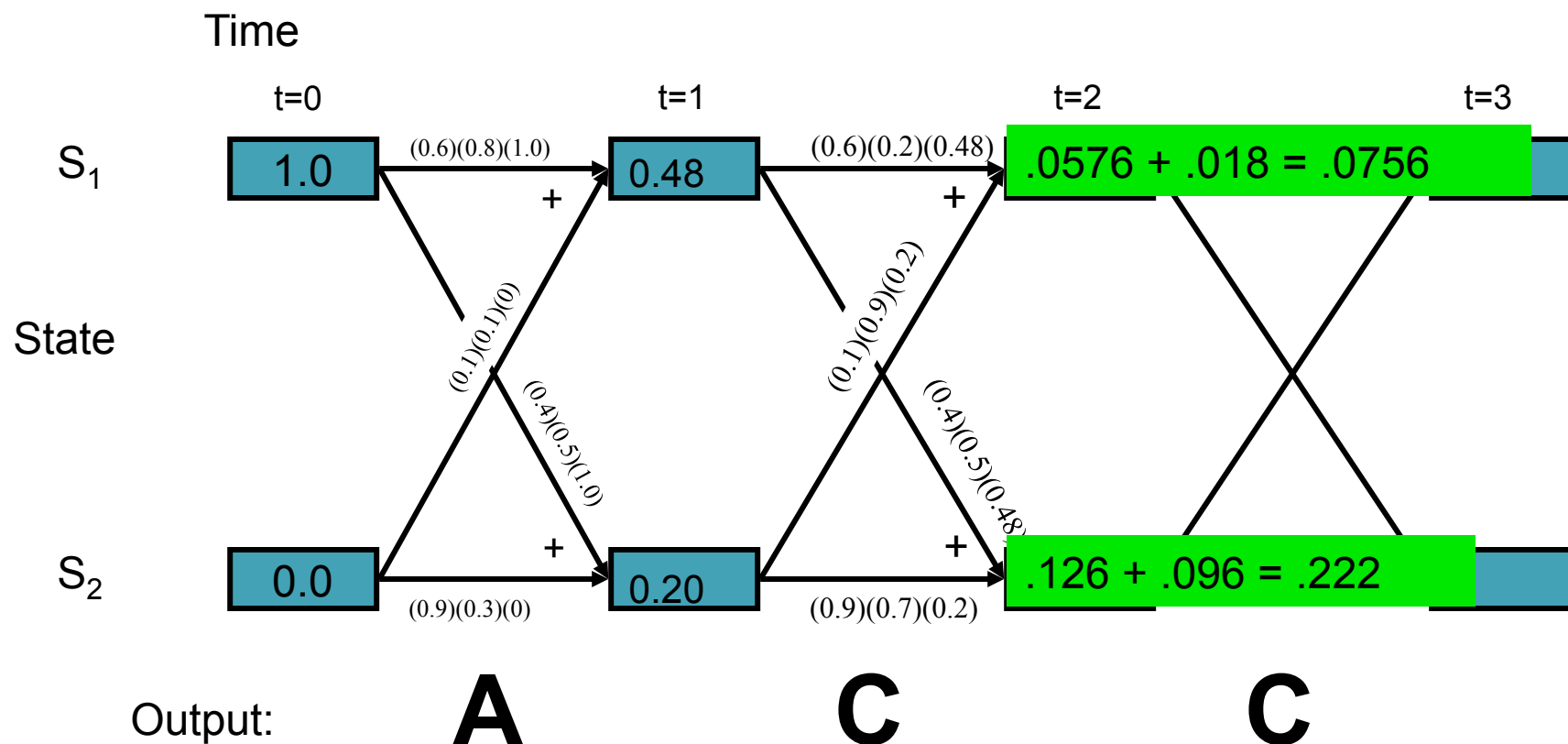
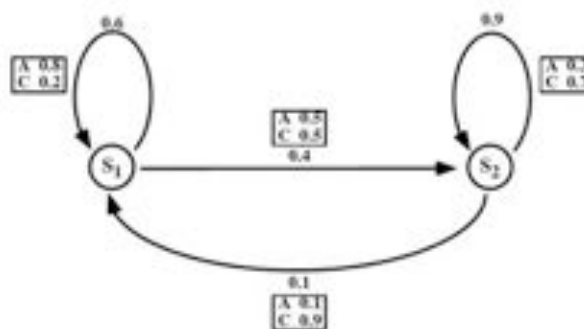


Let  $S_1$  be initial state,  $S_2$  be final state

# A trellis for the Forward Algorithm

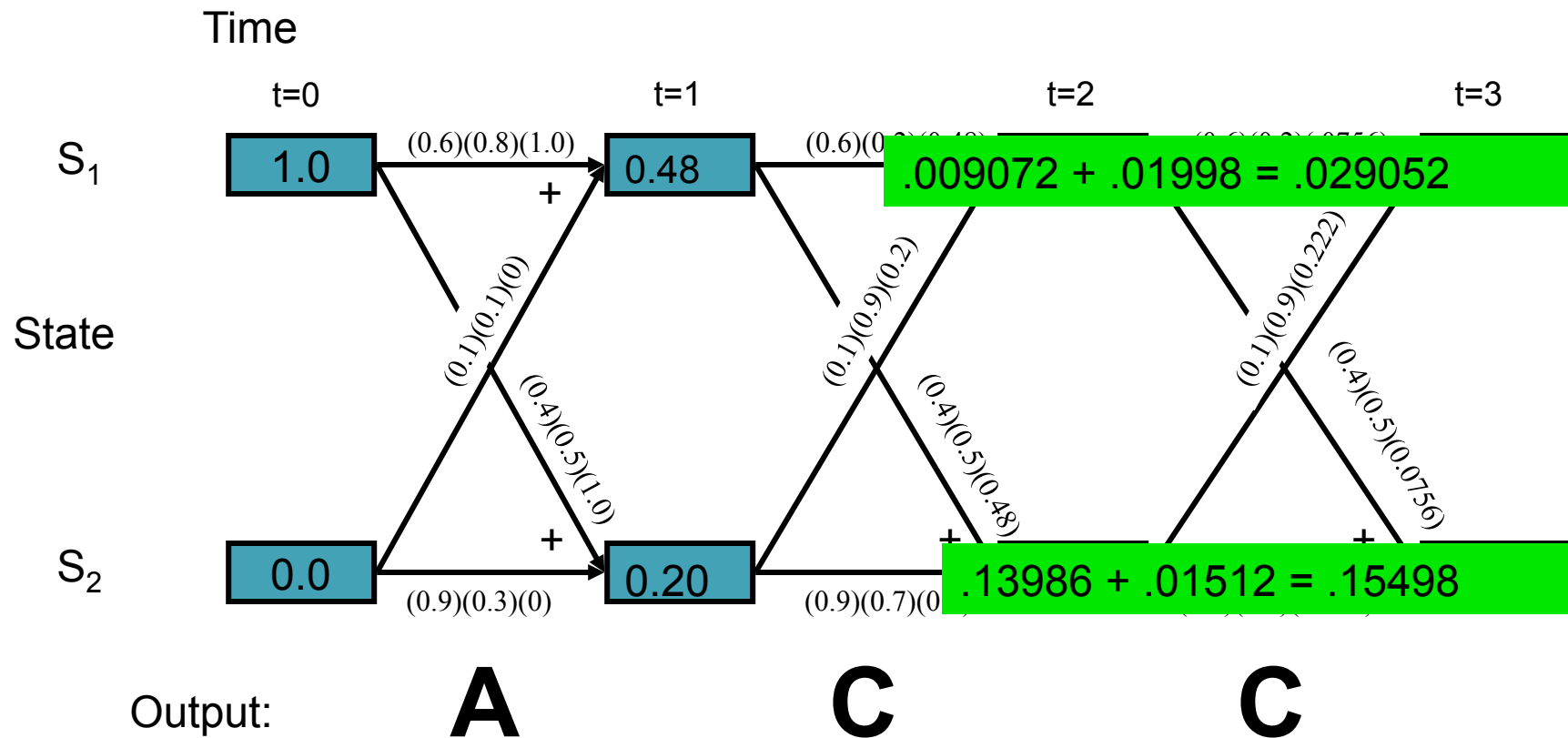
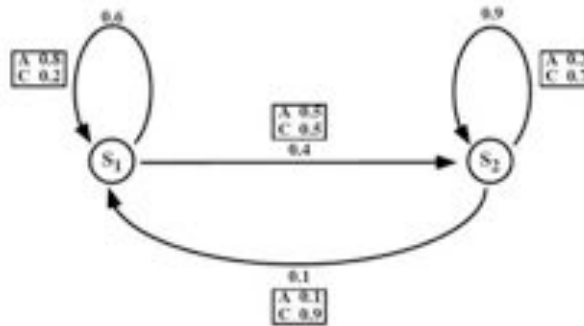


# A trellis for the Forward Algorithm





# A trellis for the Forward Algorithm



# 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: 
$$P(M | y) = \frac{P(y | M)P(M)}{P(y)}$$

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

# 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.

AAAGCATGCATTTAACGAGAGCACAAAGGGCTCTAATGCCG

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

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

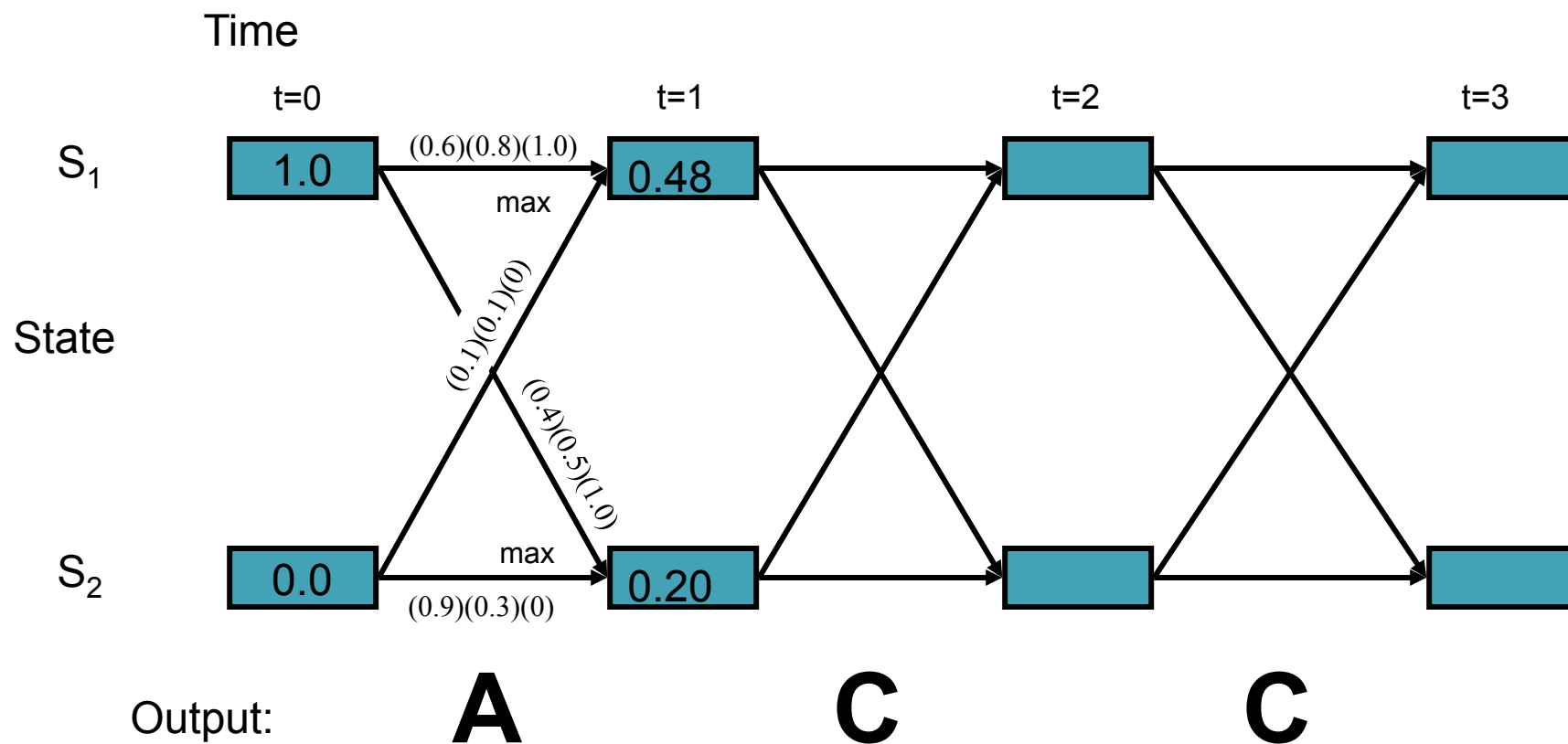
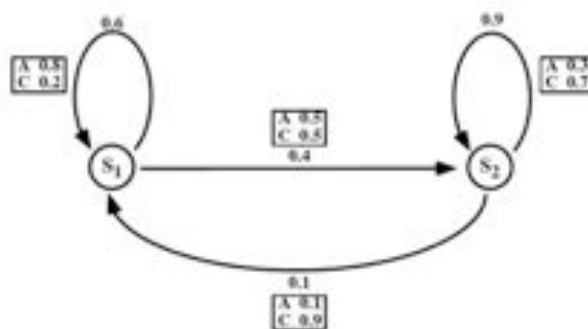
# Solving the Decoding Problem: The Viterbi algorithm

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

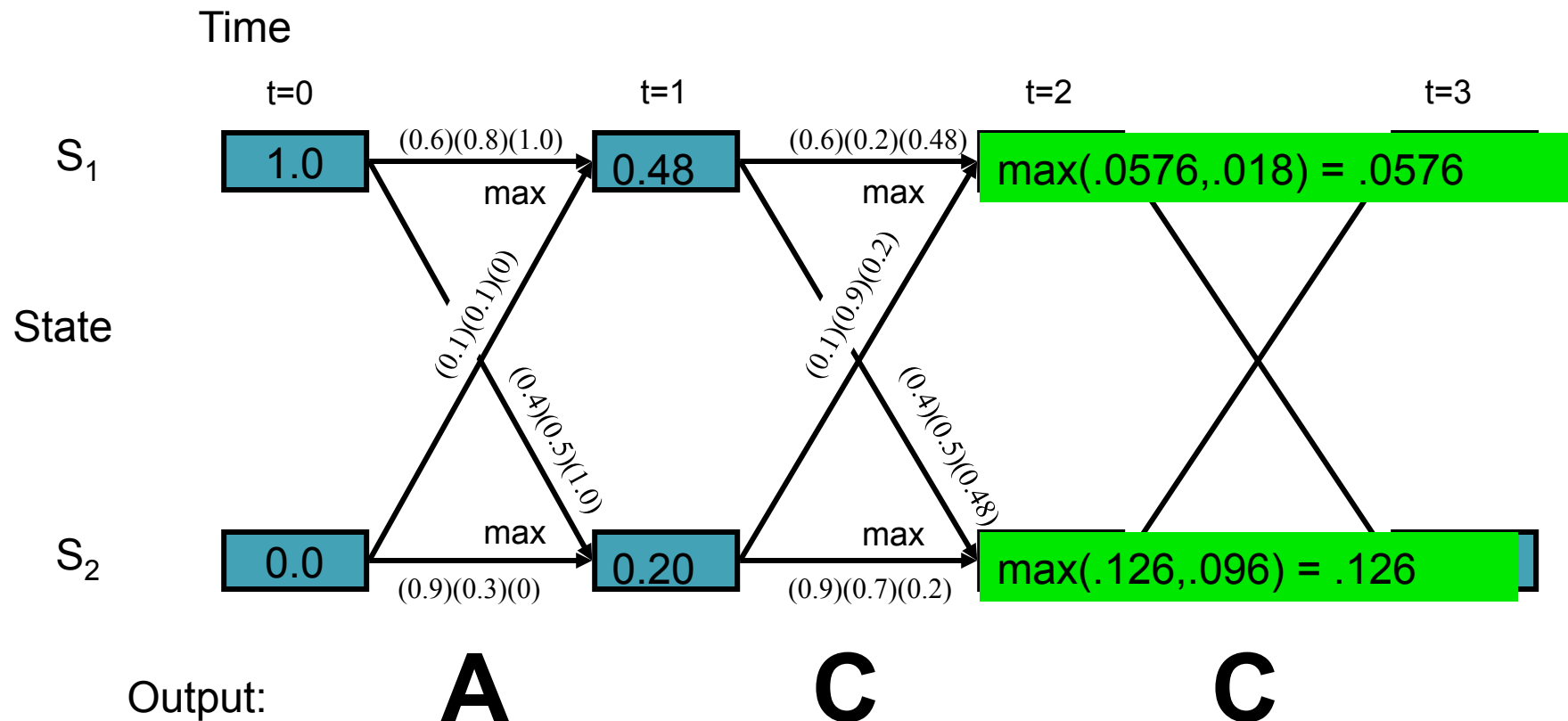
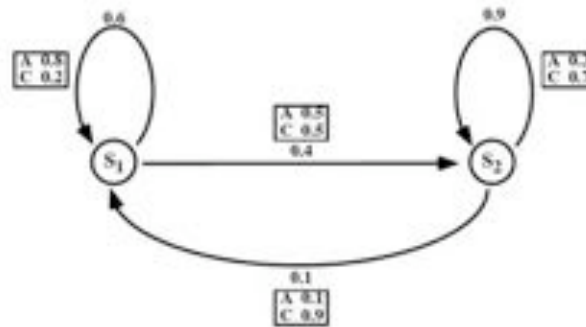
$$V_i(t) = \begin{cases} 0 & : t = 0 \wedge i \neq S_I \\ 1 & : t = 0 \wedge i = S_I \\ \max_j V_j(t-1) a_{ji} b_{ji}(y) & : t > 0 \end{cases}$$

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

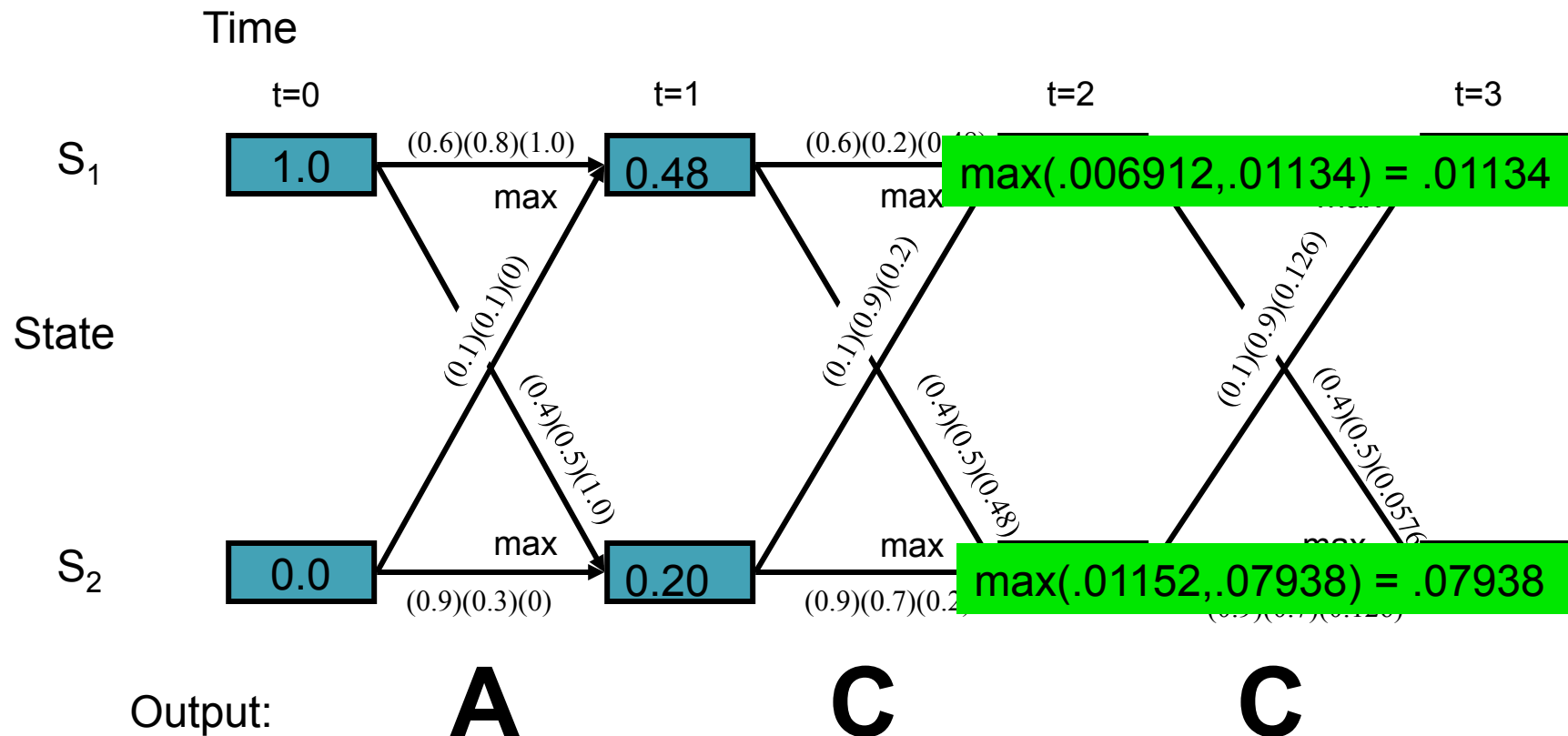
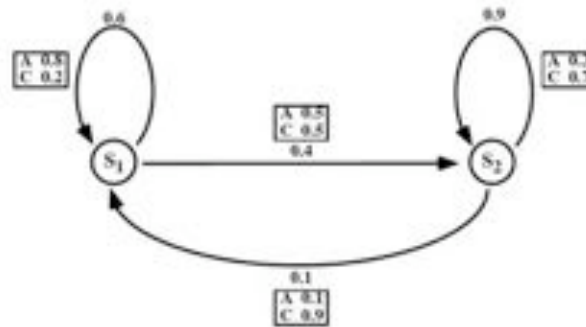
# A trellis for the Viterbi Algorithm



# A trellis for the Viterbi Algorithm

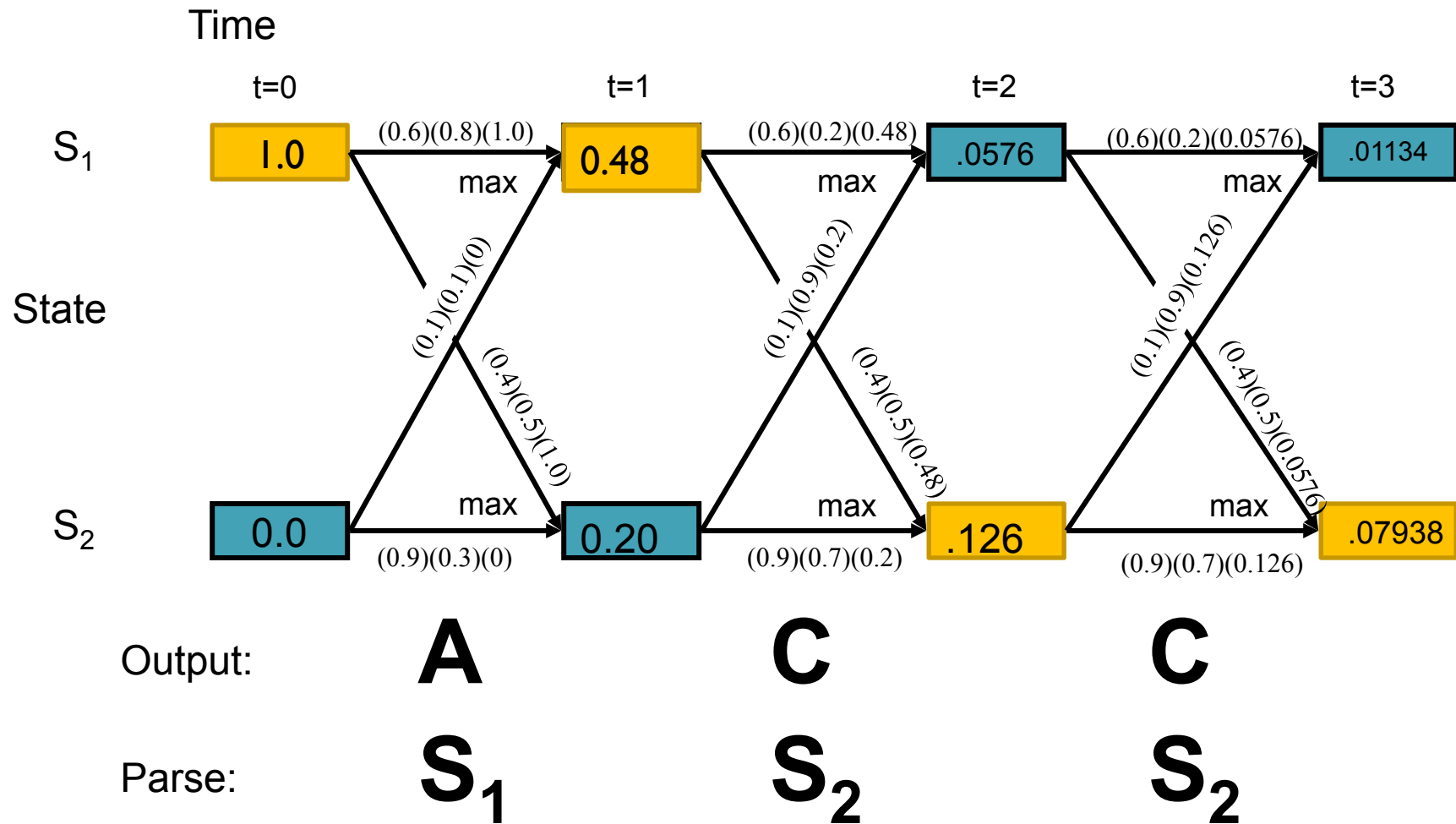


# A trellis for the Viterbi Algorithm





# A trellis for the Viterbi Algorithm



# 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

# Learning in HMMs: The E-M algorithm

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

➔ Great topic for QB2?



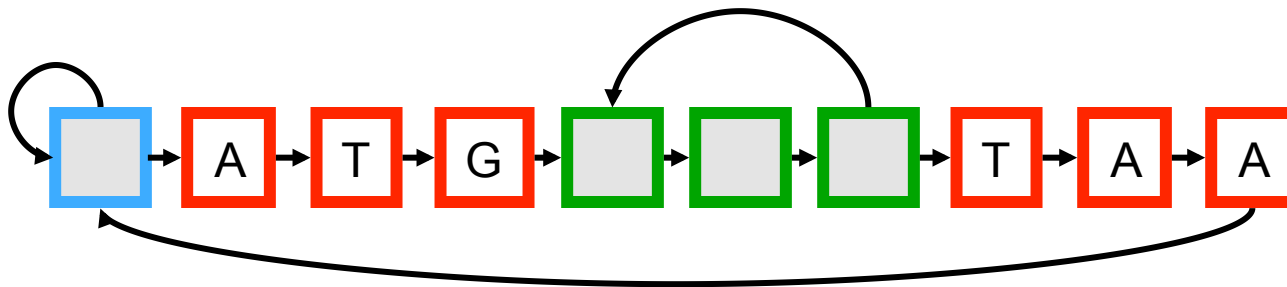
# **Eukaryotic Gene Finding with GlimmerHMM**

Mihaela Pertea  
Assistant Professor  
JHU

# 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:

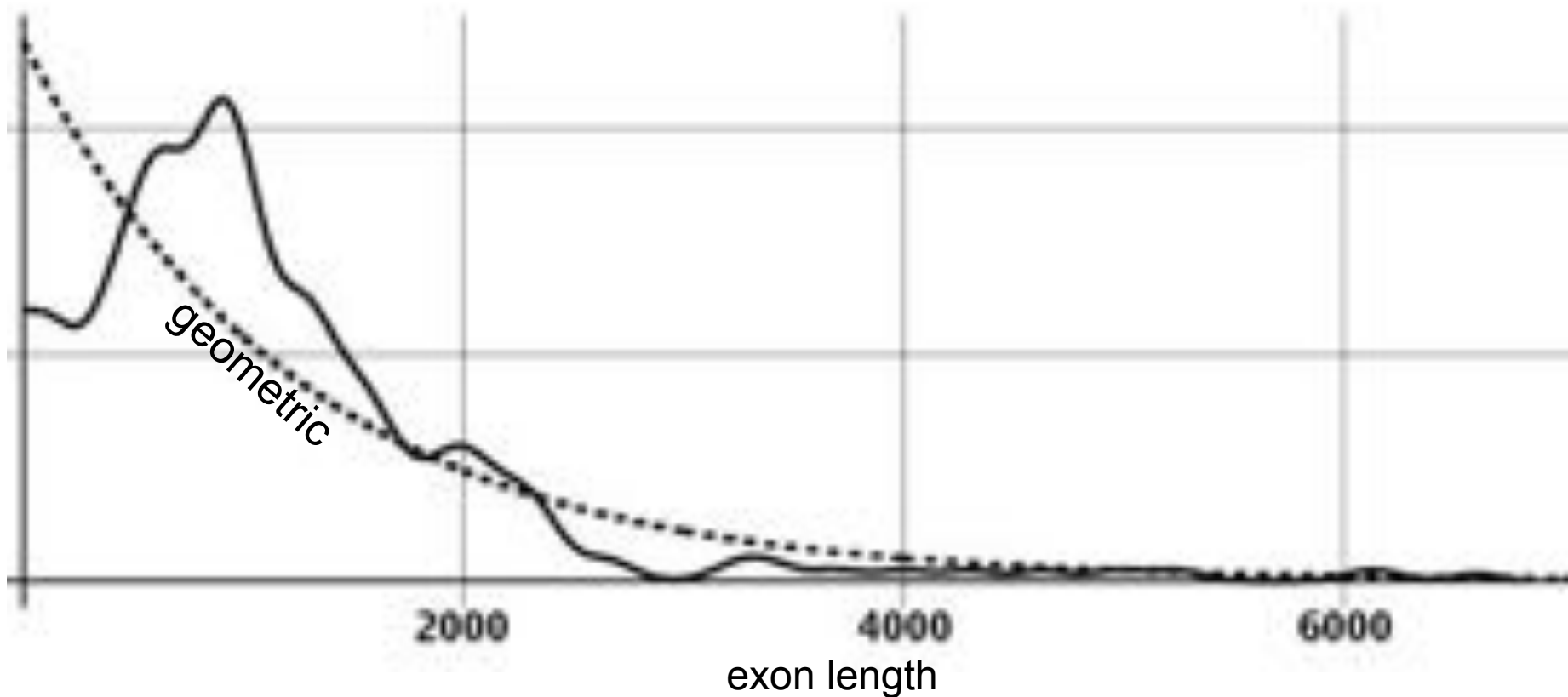
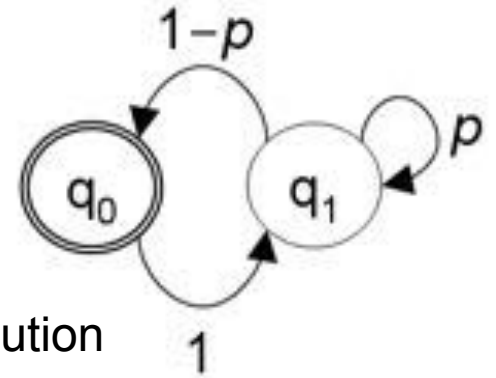


AAAGC ATG CAT TTA ACG AGA GCA CAA GGG CTC TAA TGCCG

- 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

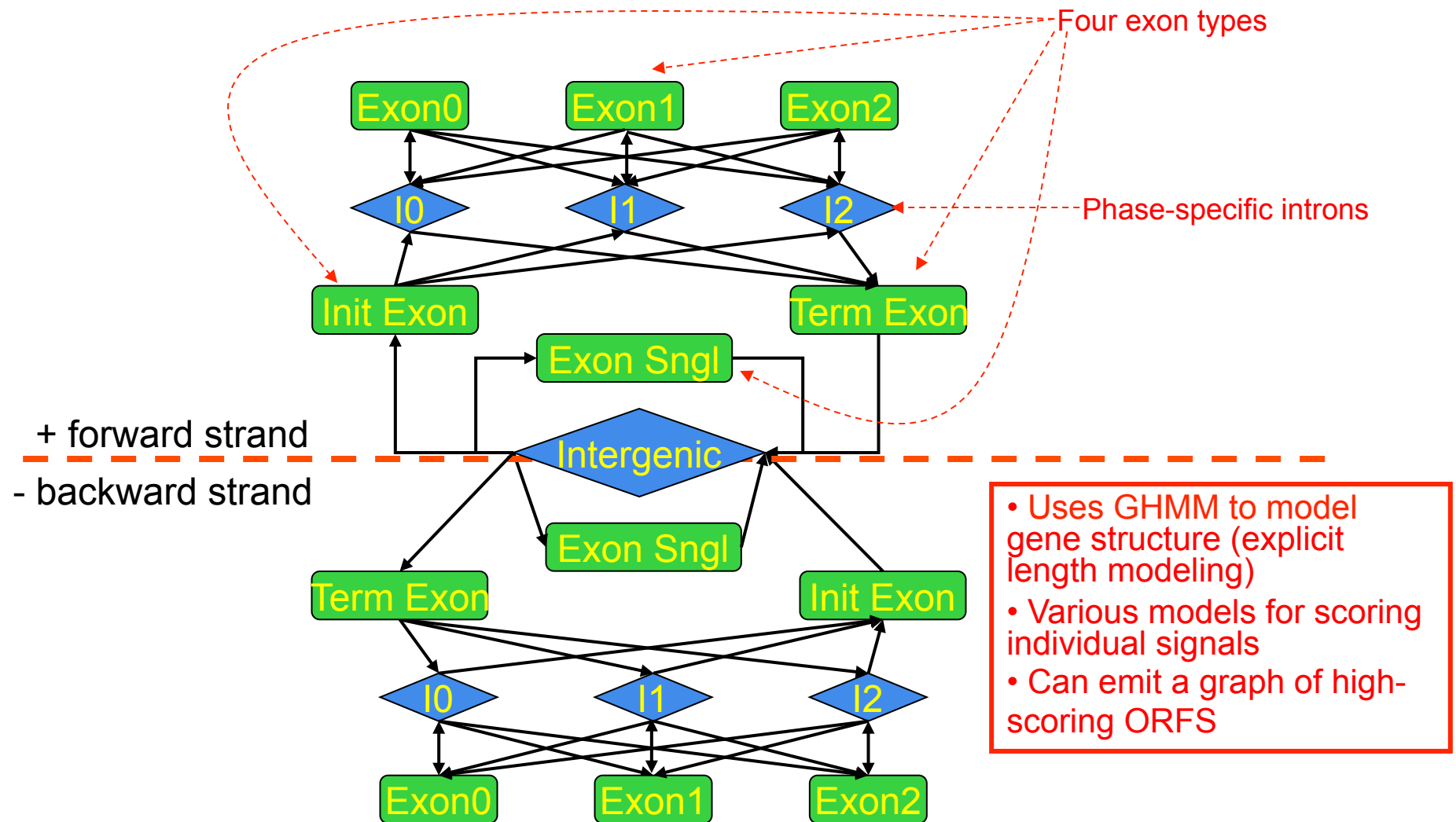
$$P(x_0 \dots x_{d-1} \mid \theta) = \left( \prod_{i=0}^{d-1} P_e(x_i \mid \theta) \right) \underbrace{p^{d-1} (1-p)}_{\text{geometric distribution}}$$



# 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

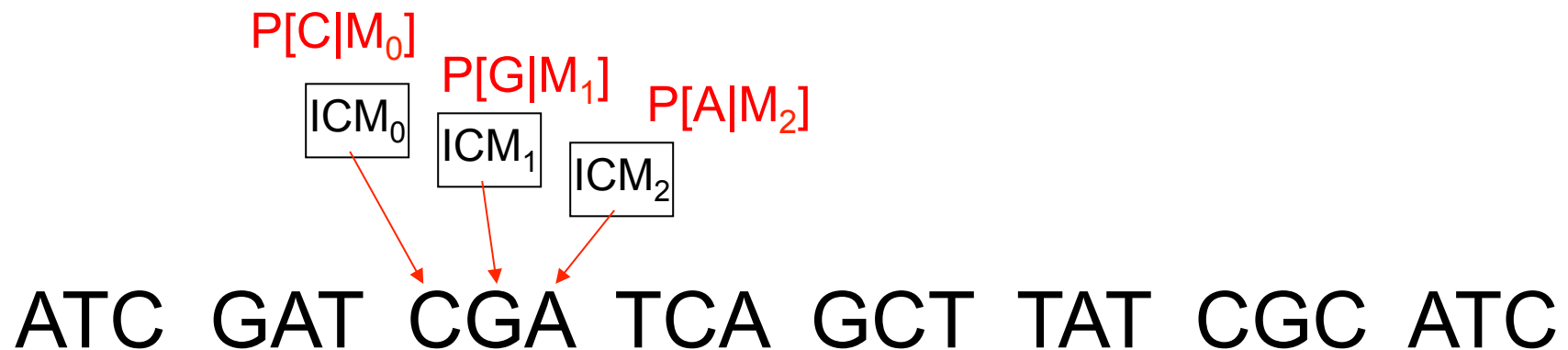
# GlimmerHMM architecture





# Coding vs Non-coding

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



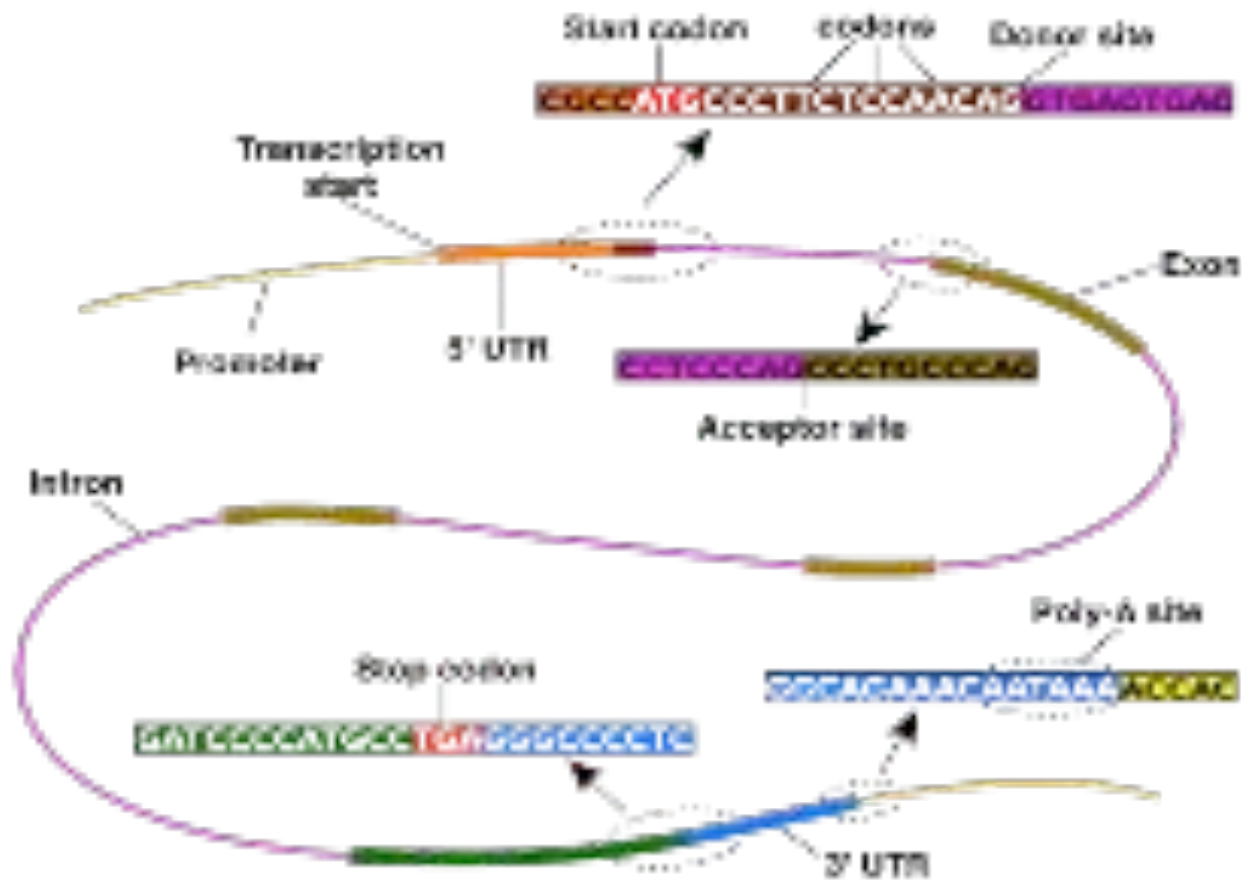
The three ICMs correspond to the three phases. Every base is evaluated in every phase, and the score for a given stretch of (putative) coding DNA is obtained by multiplying the phase-specific probabilities in a mod 3 fashion:

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

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

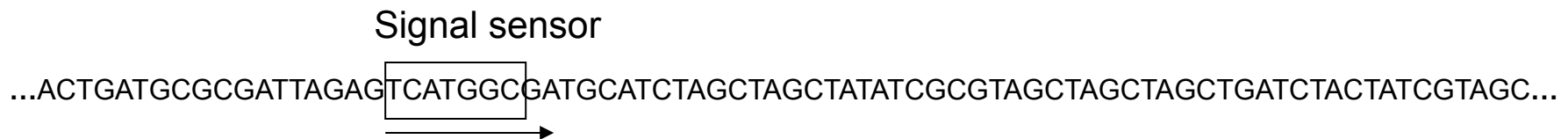
# Signal Sensors

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:

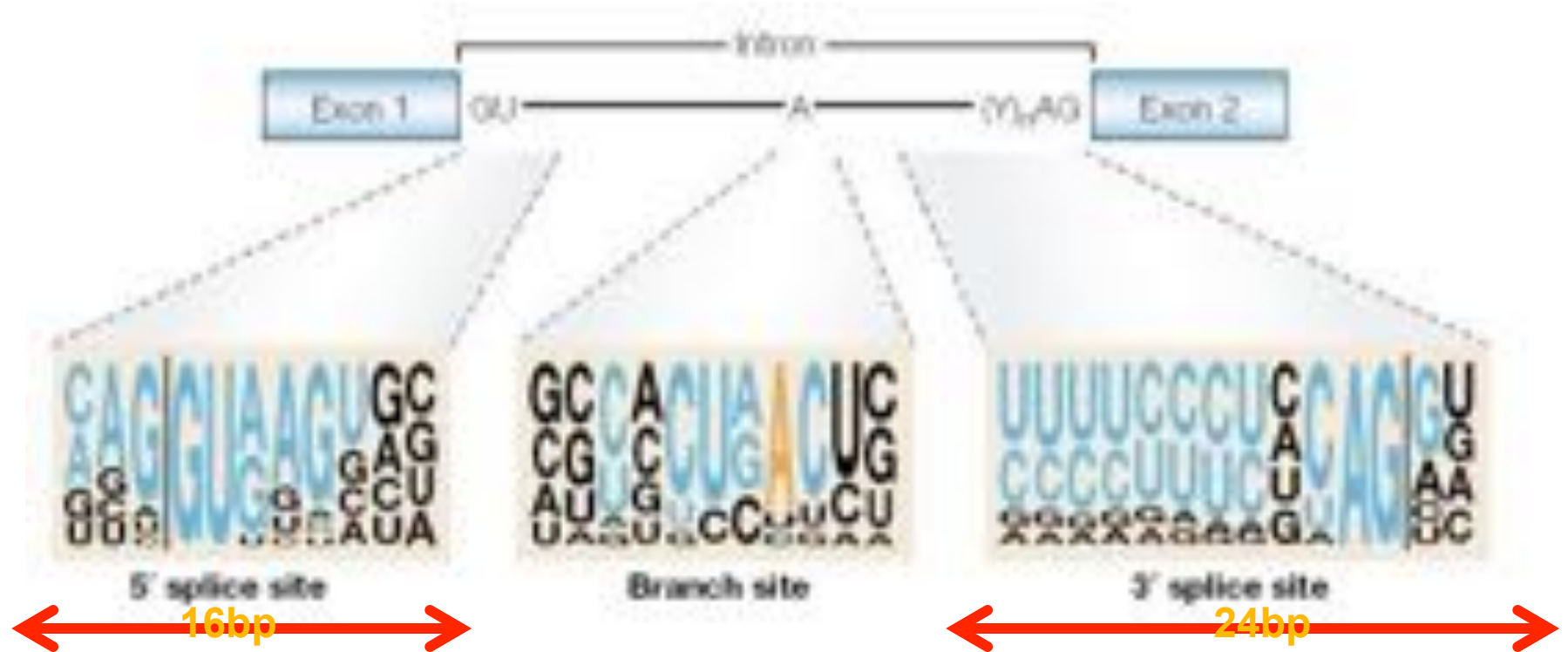


When the `score` is greater than some threshold (determined empirically to result in a desired sensitivity), we remember this position as being the potential site of a signal.

The most common signal sensor is the Weight Matrix:

|  |  |                  |                  |                  |  |  |
|--|--|------------------|------------------|------------------|--|--|
| A = 31%<br>T = 28%<br>C = 21%<br>G = 20% | A = 18%<br>T = 32%<br>C = 24%<br>G = 26% | <b>A</b><br>100% | <b>T</b><br>100% | <b>G</b><br>100% | A = 19%<br>T = 20%<br>C = 29%<br>G = 32% | A = 24%<br>T = 18%<br>C = 26%<br>G = 32% |
|--|--|------------------|------------------|------------------|--|--|

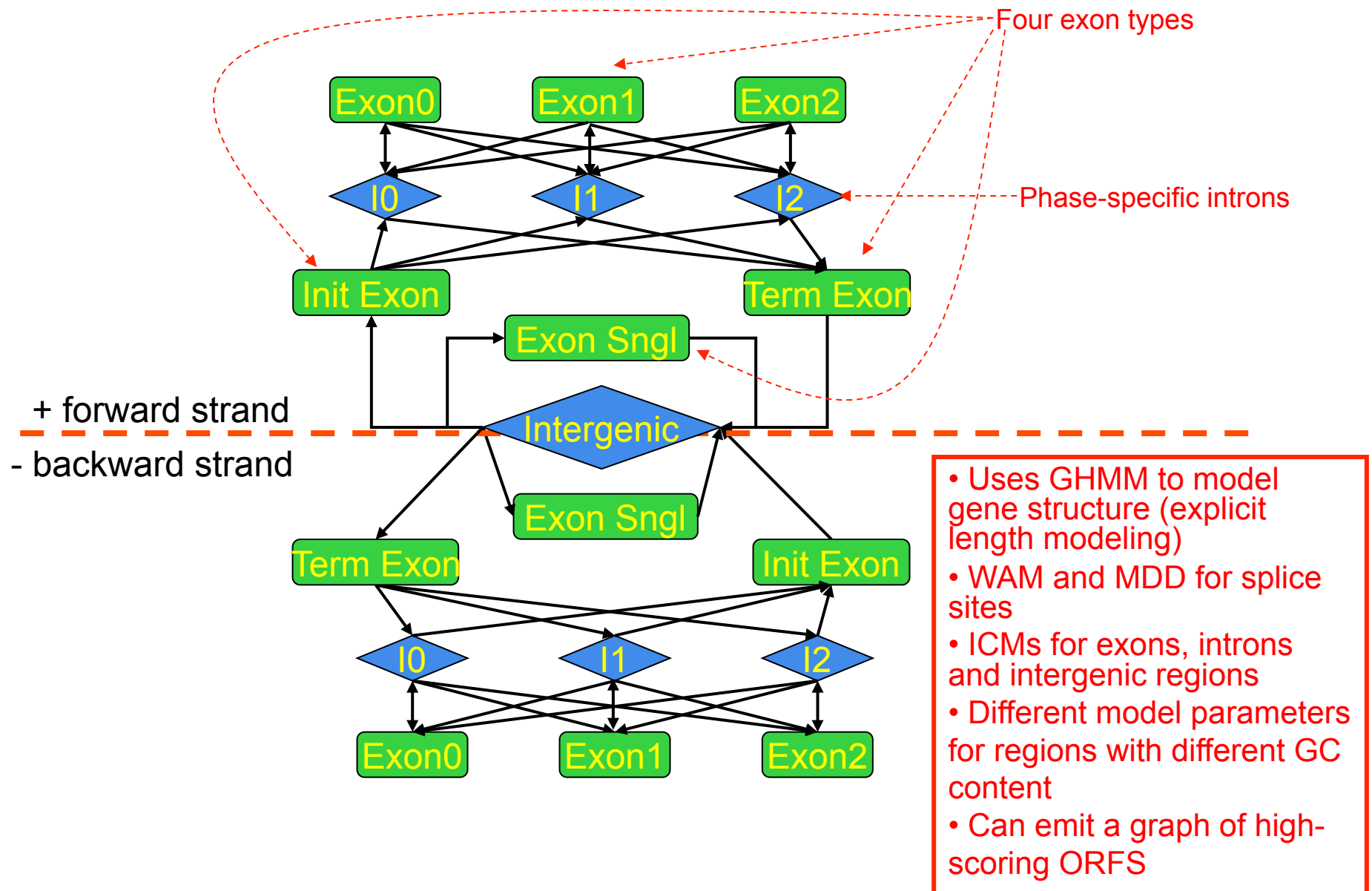
# Splice site prediction



The splice site score is a combination of:

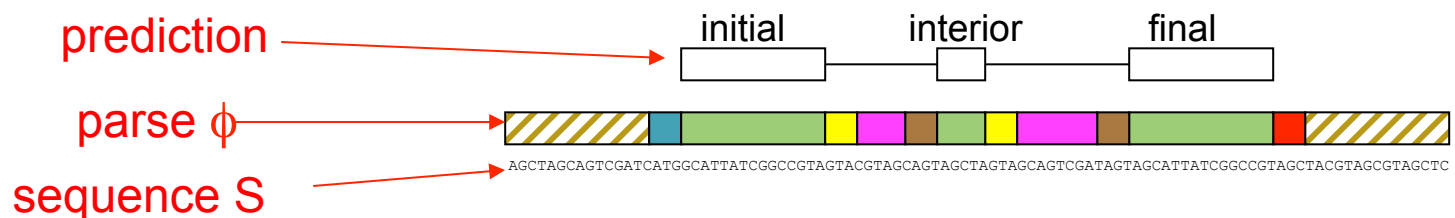
- first or second order inhomogeneous Markov models on windows around the acceptor and donor sites
- Maximal dependence decomposition (MDD) decision trees
- longer Markov models to capture difference between coding and non-coding on opposite sides of site (optional)
- maximal splice site score within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

# GlimmerHMM architecture



# Gene Prediction with a GHMM

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

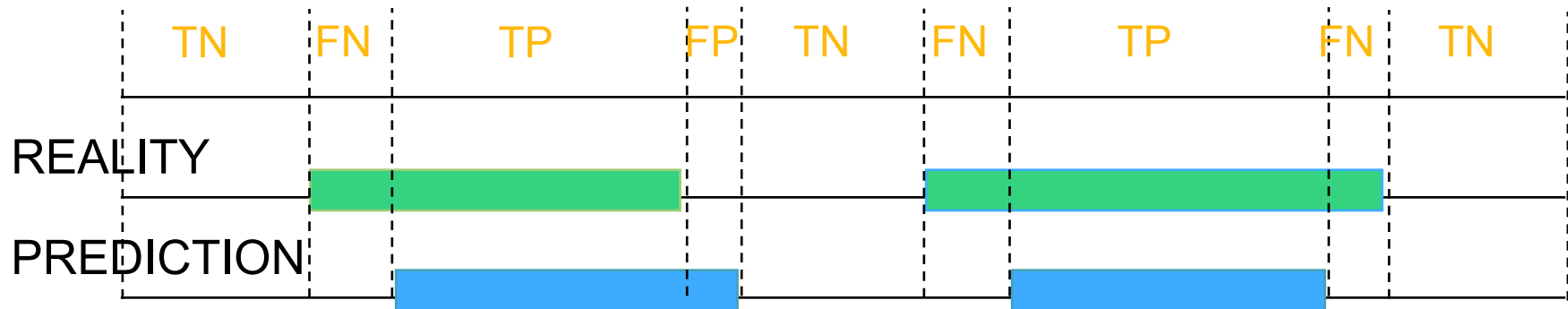


The parse  $\phi$  consists of the coordinates of the predicted exons, and corresponds to the precise sequence of states during the operation of the GHMM (and their duration, which equals the number of symbols each state emits).

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: 
$$S_n = \frac{TP}{TP + FN}$$

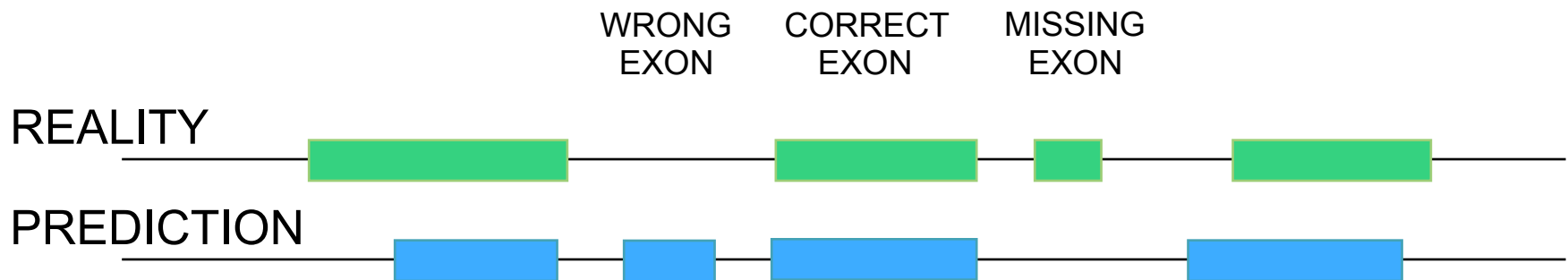
What fraction of reality did you predict?

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

What fraction of your predictions are real?

# 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 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   |

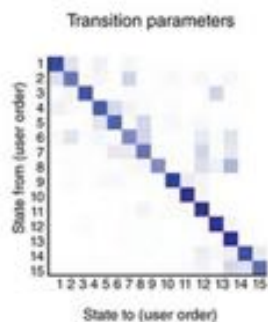
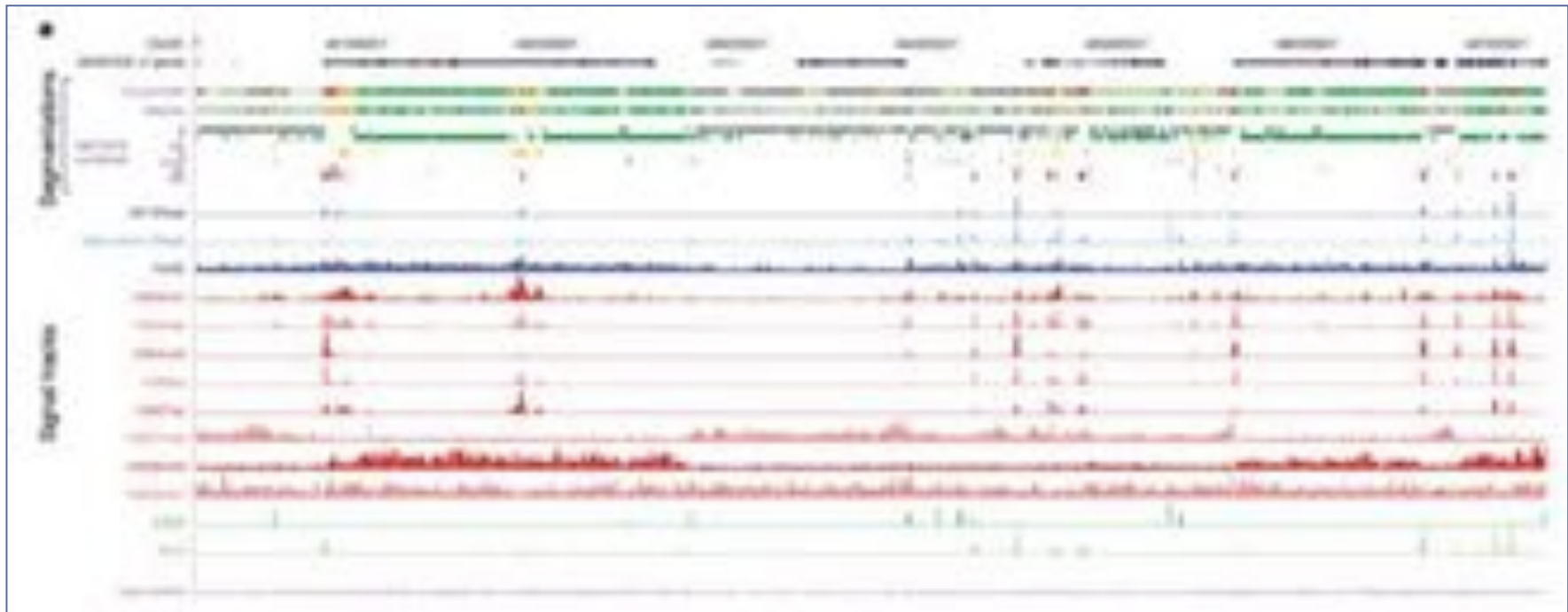
- All three programs were tested on a test data set of 809 genes, which did not overlap with the training data set of GlimmerHMM.
- All genes were confirmed by full-length Arabidopsis cDNAs and carefully inspected to remove homologues.

# GlimmerHMM on human data

|                   | <i>Nuc<br/>Sens</i> | <i>Nuc<br/>Spec</i> | <i>Nuc<br/>Acc</i> | <i>Exon<br/>Sens</i> | <i>Exon<br/>Spec</i> | <i>Exon<br/>Acc</i> | <i>Exact<br/>Genes</i> |
|-------------------|---------------------|---------------------|--------------------|----------------------|----------------------|---------------------|------------------------|
| <i>GlimmerHMM</i> | 86%                 | 72%                 | 79%                | 72%                  | 62%                  | 67%                 | 17%                    |
| <i>Genscan</i>    | 86%                 | 68%                 | 77%                | 69%                  | 60%                  | 65%                 | 13%                    |

GlimmerHMM's performance 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.

# ChromHMM: Signal Integration



| Label | Description   | Signal   | Color      |
|-------|---|--|------------|
| CTCF  | CTCF-annotated element  | Sites of CTCF signal lacking histone modifications, often associated with open chromatin. Many probably have a function in insulator assays, but because of the multifunctional nature of CTCF, we are conservative in our description. Also enriched for the cohesin components RAD21 and SMC3. CTCF is known to recruit the cohesin complex.   | Teal/green |
| E     | Predicted enhancer  | Regions of open chromatin associated with H3K4me1 signal. Enriched for other enhancer-associated marks, including transcription factors known for all of enhancers. In enhancer assays, many of these (>50%) function as enhancers. A more conservative alternative would be to regulatory regions. Enriched for sites for the proteins encoded by EP300, KID, KID1, GATA2, HNF4A, JUNB, JUN, MYC, SMARCA4, SMARCB1, SMRT, and TAL1 genes in K562 cells. Have nuclear and whole-cell RNA signal, particularly poly(A) <sup>+</sup> fraction. | Orange     |
| RE    | Predicted repressed or low-activity region                      | Regions that generally surround TSS segments (see below). This is an emergent state that includes H3K27me3 polycomb-enriched regions, along with regions that are silent in terms of observed signal for the input assays to the segmentations (low or no signal). They may have other signals (for example, RNA, not in the segmentation input data). Enriched for sites for the proteins encoded by REST and some other factors (for example, proteins encoded by REST, C/EBP, MAFK, TBM6B, ZNF274 and ZNF275 genes in K562 cells).        | Light red  |
| TSS   | Predicted promoter region including TSS                         | Round close to or overlapping GENCODE TSS sites. High precision/coverage for TSSs. Enriched for H3K4me3. Sites of open chromatin. Enriched for transcription factors known to act close to promoters and polymerases Pol II and Pol III. Short RNAs are most enriched in these segments.   | Bright red |
| T     | Predicted transcribed region                                    | Overlaid gene bodies with H3K36me3 transcriptional elongation signal. Enriched for phosphorylated Ser4 of Pol II signal (promoting polymerase II elongation). RNA, especially subgenomic.  | Dark green |
| WE    | Predicted weak enhancer or open chromatin co-regulatory element | Similar to the E data, but weaker signals and weaker enrichments.  | Yellow     |

- Summarize the individual assays into 7 functional/ regulatory states using an HMM across the genome

**ChromHMM: automating chromatin-state discovery and characterization**

Ernst & Kellis (2012) *Nature Methods*. doi:10.1038/nmeth.1906

# Summary

- Prokaryotic gene finding distinguishes real 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
    - Lots of active research incorporating other \*-seq data into model

# Break



# Review

# Exact Matching

- Explain the Brute Force search algorithm (algorithm sketch, running time, space requirement)
  1. Suffix Arrays
  2. BWT
  3. How many times do we expect GATTACA to be in the human genome (3Gbp), barley (6GB) or pine (24GB)?

# Sequence Alignment

1. What is a good scoring scheme for aligning:  
English words? Illumina Reads? Gene Sequences? Genomes?
2. Explain Dynamic Programming for computing edit distance
3. BLAST
4. 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 align genomes?

# 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

# Thank you!

- We have covered an amazing amount of material in a short amount of time
  - We tried to select the most important topics for you 😊
  - So many topics left unexplored 😞
- Additional Resources:
  - Google, <http://schatzlab.cshl.edu/teaching/>
  - Rotations, Research Projects
  - Textbooks, Coursera, Khan Academy
  - Papers, conferences, meetings
  - Practice, practice, practice and never stop learning

Email 24/7 for homework questions & exam prep

Let us know how we did – we take your input very seriously!