GENE FINDING

The Computational Problem

We are given a sequence of DNA and we wish to know which subsequence or concatenation of subsequences constitutes a gene.

The Computational Problem Confounding Realities:

There is a key difference between prokaryotes and eukaryotes in the context of recognizing genes in sequences:

- Over 85% of the prokaryotic genome is coding
- Only 3% of the eukaryotic genome is coding

The Computational Problem Restated

Parse genomic DNA to identify a complete gene

Eurkaryotic Genes

In order to find a eukaryotic gene, at the very least we must identify 4 signals:

1. Start codon
2. Stop codon
3. Beginning of intron (donor site)
4. End of intron (acceptor site)

It helps to find other signals outside the gene such as promotors and ribosomal binding sites

Start Codon

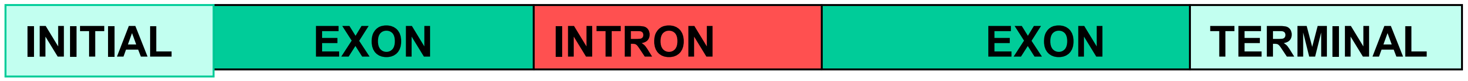
- Eukaryote ATG
- Prokaryote ATG,GTG,TTG

- Can occur by chance $1/4^3$ each
- There is usually something characteristic upstream (a promotor), but its location and sequence are not always consistent among genes or species

Stop Codon

- TAA. TAG. TGA
- Nothing characteristic upstream
- Also a $1/4^3$ chance of random occurrence for each

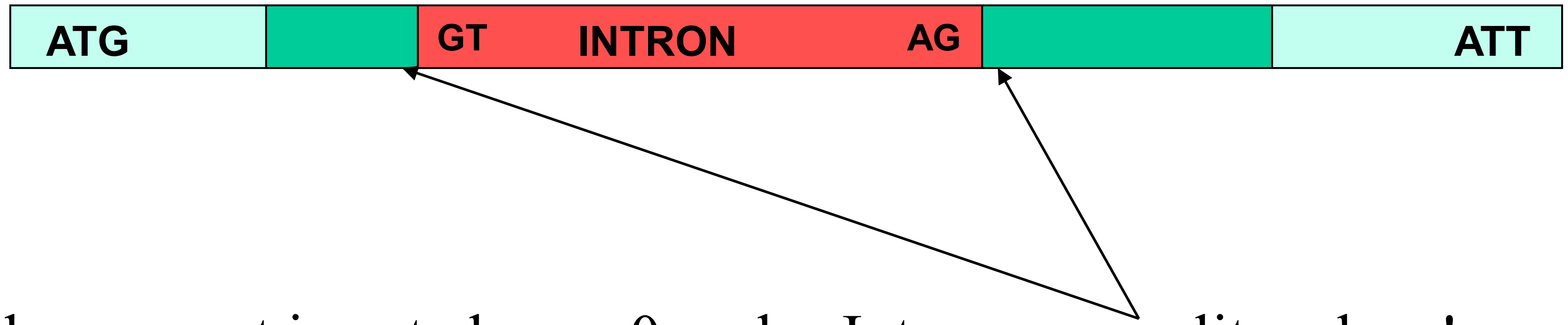
Exons in Eukaryotes



- Initial Exon
  - Begins with START codon
- Terminal Exon
  - Ends with STOP codon

Splice Sites

- Donor almost always GT
- Acceptor almost always AG
- Certain consistencies around the splice sites.



Exon base count is not always 0mod_3: Introns can split codons!

The split is not necessarily in-frame

Open Reading Frames (ORFs)

Begins with START codon, ends with STOP codon.

Usually it is the longest sequence without a stop codon.

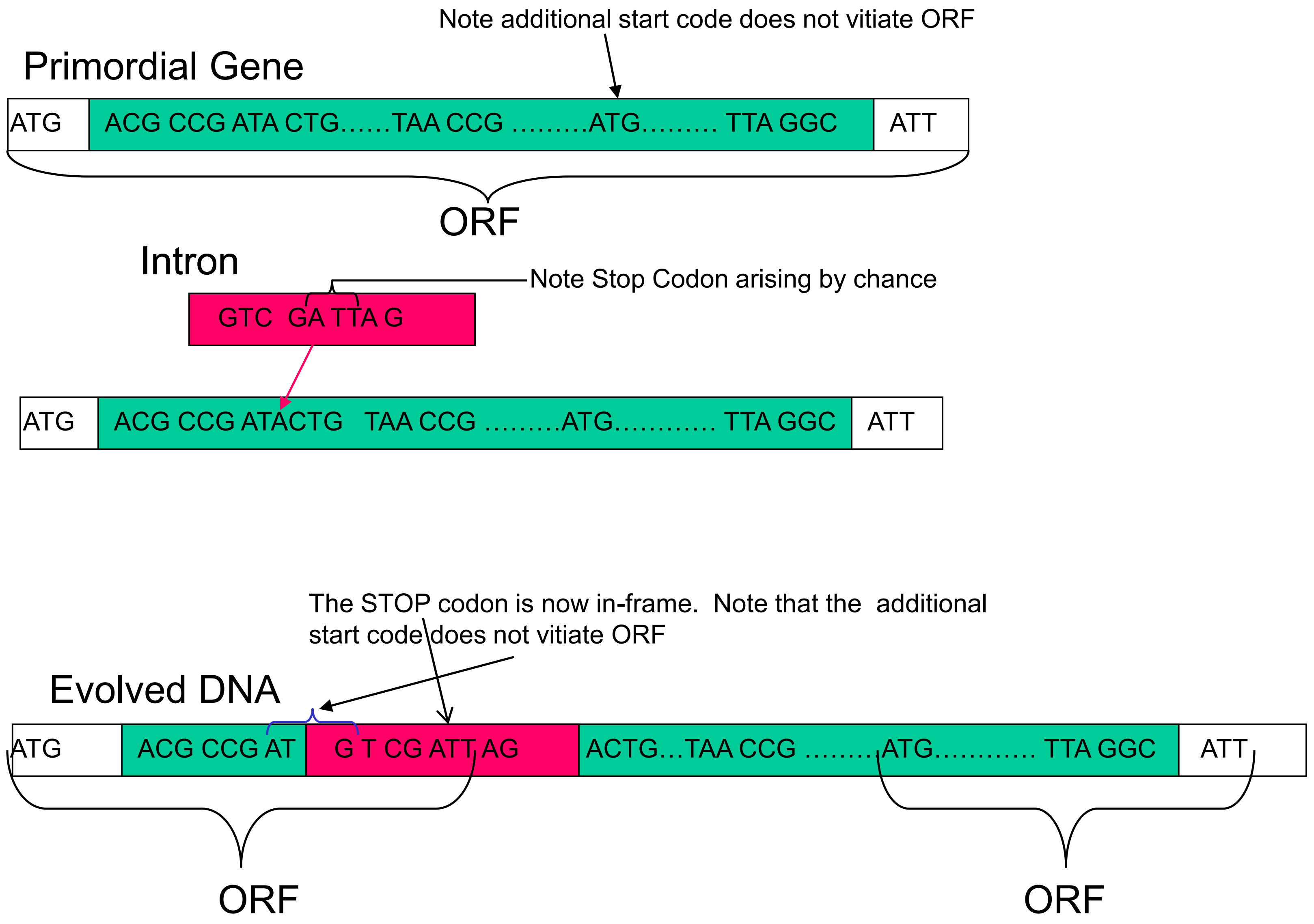
Should be a piece of cake: Theoretically, an ORF is a Gene. Indeed, in a prokaryote, it probably *is* a gene

The problem:

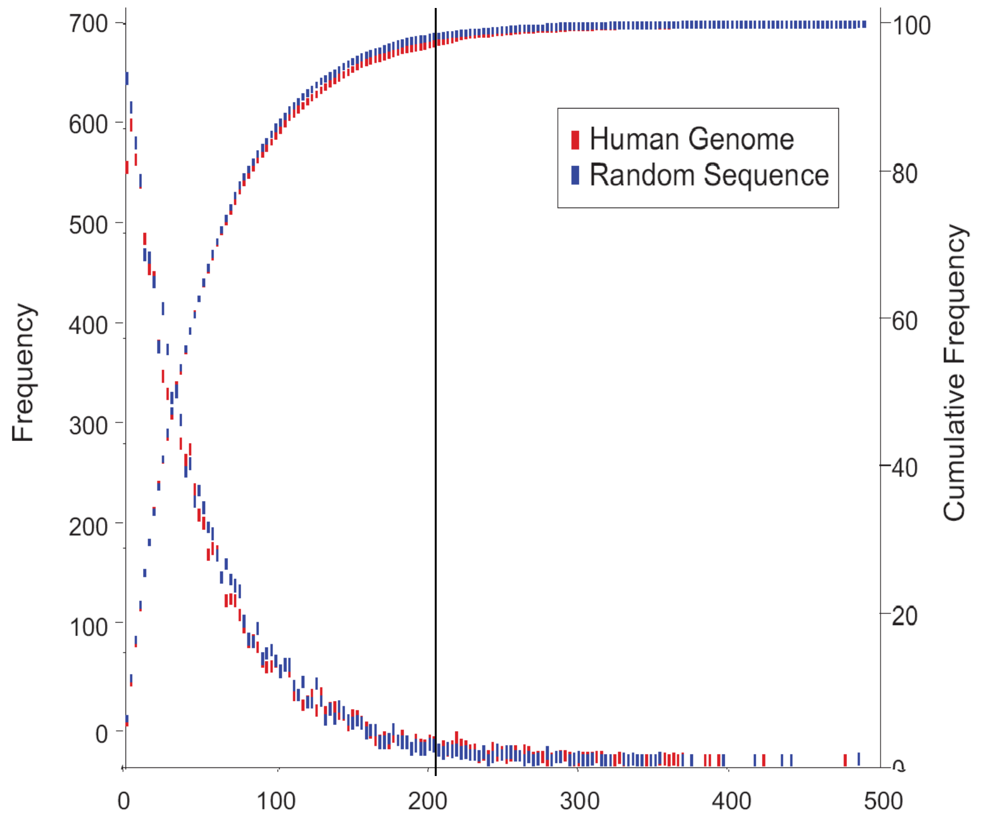
An ORF is ended by a stop codon. How do you know that the STOP codon is the ‘real’ stop, or instead is a stop codon sequence embedded randomly in a non-coding area (1/64 chance)?

Further, this STOP codon could come from any of 6 reading frames.

(Sense and anti-sense, 3 frames each)


Note additional start code does not vitiate ORF
Primordial Gene
ATG
ACG CCG ATA CTG……TAA CCG ………ATG……… TTA GGC
ATT
ORF
Intron
Note Stop Codon arising by chance
GTC GA TTA G
ATG
ACG CCG ATACTG TAA CCG ………ATG………… TTA GGC
ATT
The STOP codon is now in-frame. Note that the additional start code does not vitiate ORF
Evolved DNA
ATG
ACG CCG AT
G T CG ATT AG
ACTG…TAA CCG ………ATG………… TTA GGC
ATT
ORF
ORF

ORFs have lengths upwardly bounded at 600, with 99% of length <200, exponentially distributed. One theory is that they originated from random DNA as evidenced by this random generation model



Length of ORFs (bases)

Origin of the Split Structure of Spliceosomal Genes from Open Reading-Frame Length Constraints in Primordial Random DNA Sequences
Rahul Regulapati, Chandan Kumar Singh, Ashwini Bhasi, Periannan Senapathy manuscript in preparation 2006- personal communication

So, in summary, and as perhaps an oversimplification:

- Prokaryotes
    - Find the ORF

- Eukaryotes
    - Need to identify more, including, at the very least, splice sites

Gene Finding Strategies

- Brute-force Signal ID (match, consensus, motif)
- Content scoring methods-analyze large segments for statistical patterns
  - Codon frequency. Differs in
    - Coding part of exons
    - Noncoding part of exons
    - Introns
    - Intergenic regions
  - Di Codon frequency
  - N-mer freqencies
- Entropy (varies among the 4 regions)
- Periodic frequencies

Gene Finding Models

- Weight Matrix Models
- Probabilistic
  - Hidden Markov Models
  - Bayesian
- Decision Trees
- Neural Nets
- Combinations

Machine Learning Issue:

Supervised *vs* Unsupervised

Gene Finding Targets

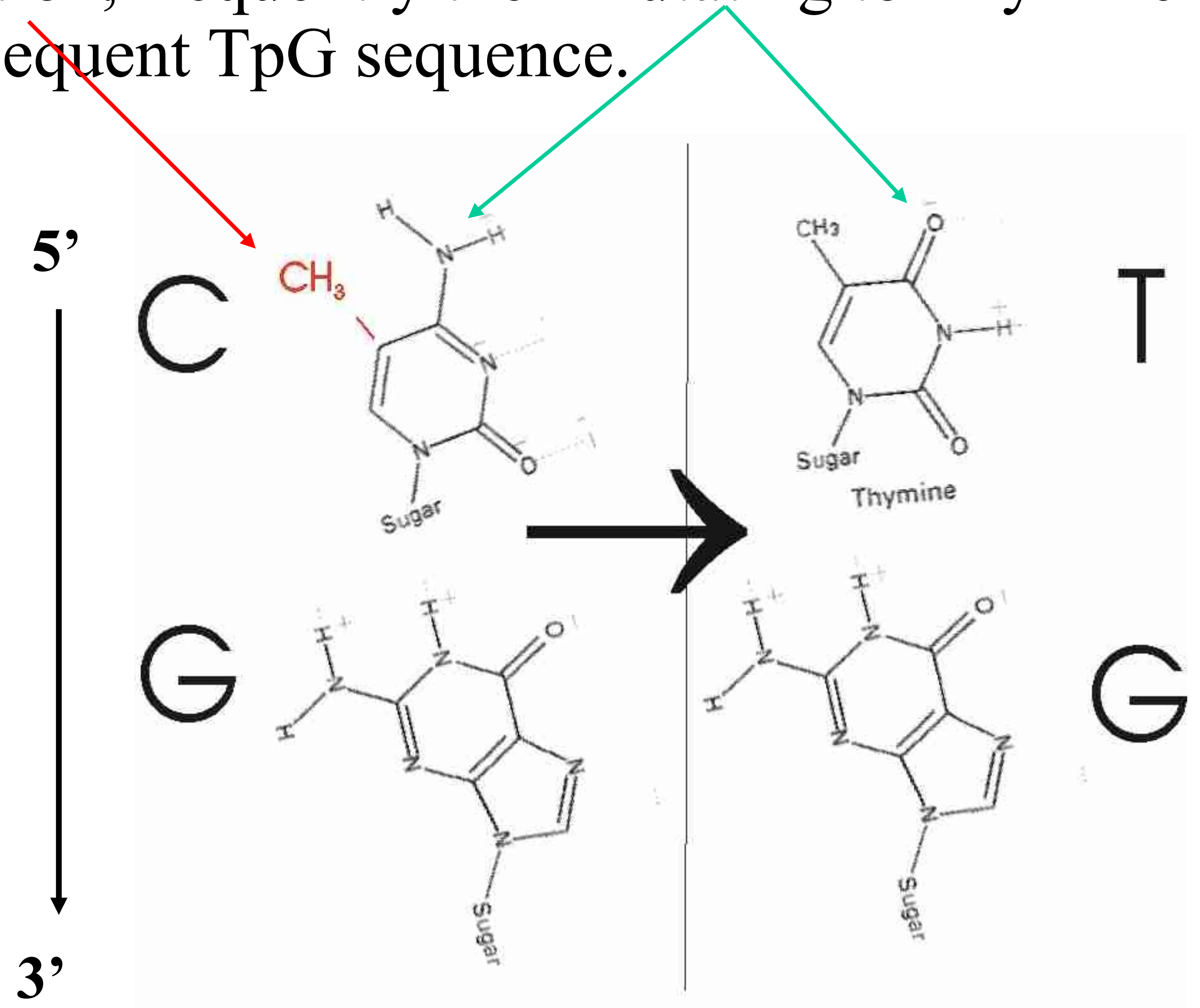
- Gene content
- Intragenic content
- Splice Sites
- Promotor region signals
  - CpG Islands in vertebrates
- Promotor binding sites
- Ribosome binding site
  - Prokaryotes:
    - Shine-Delgarno sequences **AGGAGG** at -35, -8
    - Pribnow-Schaller Box **TATAAT** at -10
  - Eukaryotes:
    - Kozack consensus **A/GAACCCATGG** at -8
    - TATA Box **TATAAA** at -25
    - CAAT Box **GGCCAATCT** 75-80 upstream from other promoters
- Transcription Start site
- Gene start and stop signals

A Weight-Matrix Example

Signals: CpG Islands

CpG Islands

Cytosine before Guanine becomes methylated on its 1-position, frequently then mutating to Thymine with a consequent TpG sequence.



CpG Islands

As a consequence, the frequencies of C followed by G occur much more seldom than would be random.

Given this predilection for eliminating CpG, the event of an actual occurrence of CpG, then, might be a **SIGNAL** because it has now become ‘unexpected’ in this new context

CpG Islands-Signals

Indeed, it is a signal:

- In vertebrates, near start codons or near promoters, the methylation/substitution does not occur for some reason
- As a consequence, there are extended lengths of DNA in which CpG dinucleotide sequences exist at a much higher frequency than elsewhere. These are called ***CpG Islands.*** These islands are located in sections from 200-500 bp long, near promotors. ~40% of mammals have them, and nearly twice that in humans

Brute Force: A Classical Approach

Employ a variant of codon frequency, using dinucleotide (or transition) frequency and run a window along the string of DNA

Does a short stretch of DNA come from a CpG island?

- Train on CpG islands
  - Compute the probability of each possible transition

$$\frac{\sum_{\forall i, \forall j} transition_{island} i \rightarrow j}{number\ of\ all\ possible\ transitions_{island}}$$

- Train on CpG oceans
  - Compute the probability of each possible transition

$$\frac{\sum_{\forall i, \forall j} transition_{ocean} i \rightarrow j}{number\ of\ all\ possible\ transitions_{ocean}}$$

From Durbin,Eddy *et al* 1998

Table of Transition Probabilities for CpG Islands

Model +	A	C	G	T	
A	0.180	0.274	0.426	0.120	= 1
C	0.171	0.368	0.274	0.188	= 1
G	0.161	0.339	0.375	0.125	= 1
T	0.079	0.355	0.384	0.182	= 1

Table of Transition Probabilities for Regions with no CpG Islands

Model -	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

From Durbin,Eddy *et al* 1998

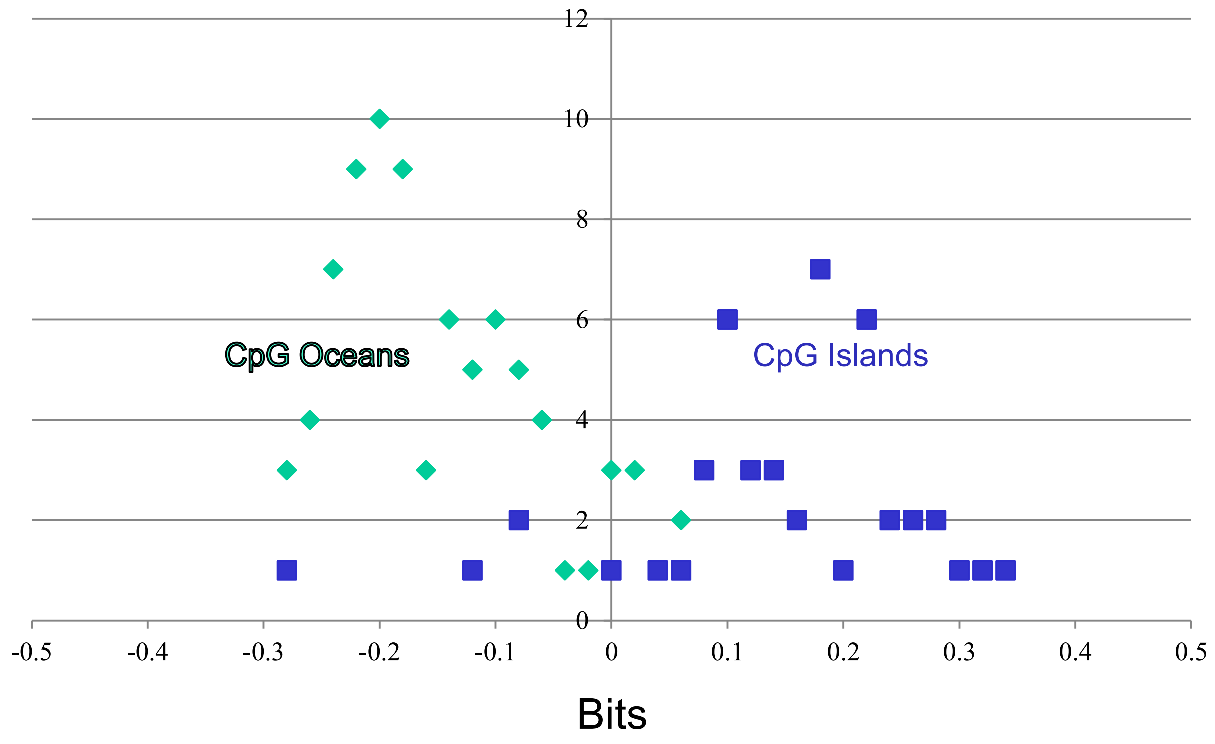
$$Log\ Odds\ Ratio = \log \frac{P(x \mid Island)}{P(X \mid Ocean)} = \sum_{i=1}^{L} \log \frac{I_{x_{i-1}x_i}}{O_{x_{i-1}x_i}}$$

Gives an overview of the window contents rather than one specific transition

	A	C	G	T
A	$-.740$	$.419$	$.580$	$-.803$
C	$-.913$	$.302$	1.812	$-.685$
G	$-.624$	$.461$	$.331$	$-.730$
T	-1.169	$.573$	$.393$	$-.679$

Log_2 likelihood length-normalized scores for many sequences

There are 48 islands and 81 oceans shown here



Modified from
Durbin,Eddy *et al* 1998

The Never-Ending Story

In order to train the model, you must know *up front* whether the training data came from an island or an ocean

Other issues:

- Window length
  - May engulf an ocean embedded between 2 islands (islands are long)
- Sensitivity in crossing shoreline
  - Hysteresis nullifies important info

GENE FINDING ANOTHER APPROACH

The Hidden Markov Model

The Markov Assumption and Finite Automata

In a nondeterministic finite automaton, the probability of transition from one state B to the next C does NOT depend on the probability of arriving at the first state B from its preceding state A.

This is the Markov assumption on states B and C.

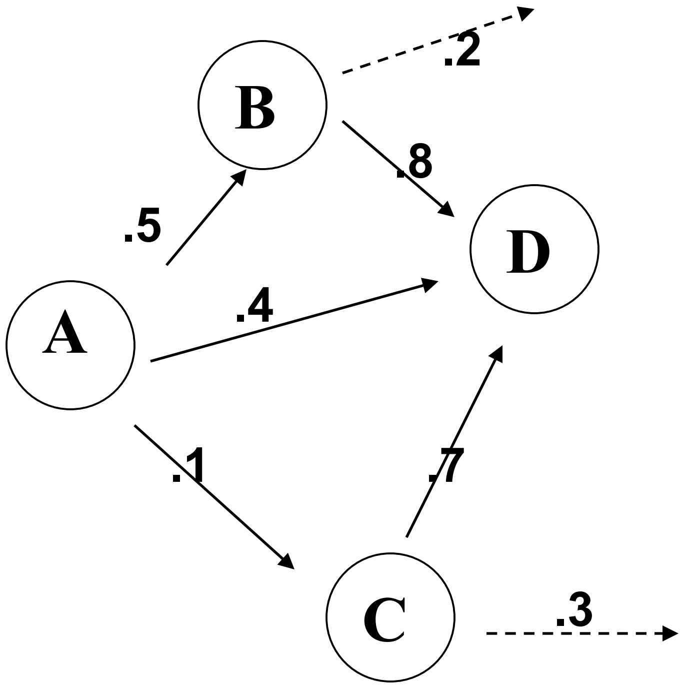
Seeking the probability p of transition from one state (A) to another (B) in a nondeterministic finite automaton p(A→B) is the same thing as asking

What is the probability of B *given* A?

or

What is p(B|A)?

NDFA



P(D|A)=.4

Another way to think of this:

1. The Markov assumption lets us use a nondeterministic finite automaton to model the chain processing. Keep in mind that a transition probability out of an automaton state is a joint probability conditioned only on the state we are in - this is the Markov condition.
2. In that case, we can parse the states with a *regular expression*

The Markov Assumption

The Markov assumption enabled us to get rid of the extended joint probabilities

$$p(x) = p(x_{n,} x_{n-1,} x_{n-2} \ldots\ldots x_1)$$

$$= p(x_{n,} \mid x_{n-1,} x_{n-2} \ldots\ldots x_1)\, p(x_{n-1,} x_{n-2} \ldots\ldots x_1)$$

Replacing them by a much simpler expression

$$p(x) = p(x_n \mid x_{n-1})\, p(x_{n-2} \mid x_{n-3}) \ldots .p(x_1) = p(x_n) \prod_{n=2}^{N} a_{x_{n-1}} a_{x_n}$$

Hidden Markov Model

- A set of states (*eg* CpG island, CpG ocean)
- A set of symbols to be emitted (*eg* {A,C,T,G})
- A set of emission probabilities for each symbol from each state
- An index (*eg* time, next nucleotide)
- A transition probability between successive states

Hidden Markov Models

•There is a probability of <u>transitions</u> out of each state

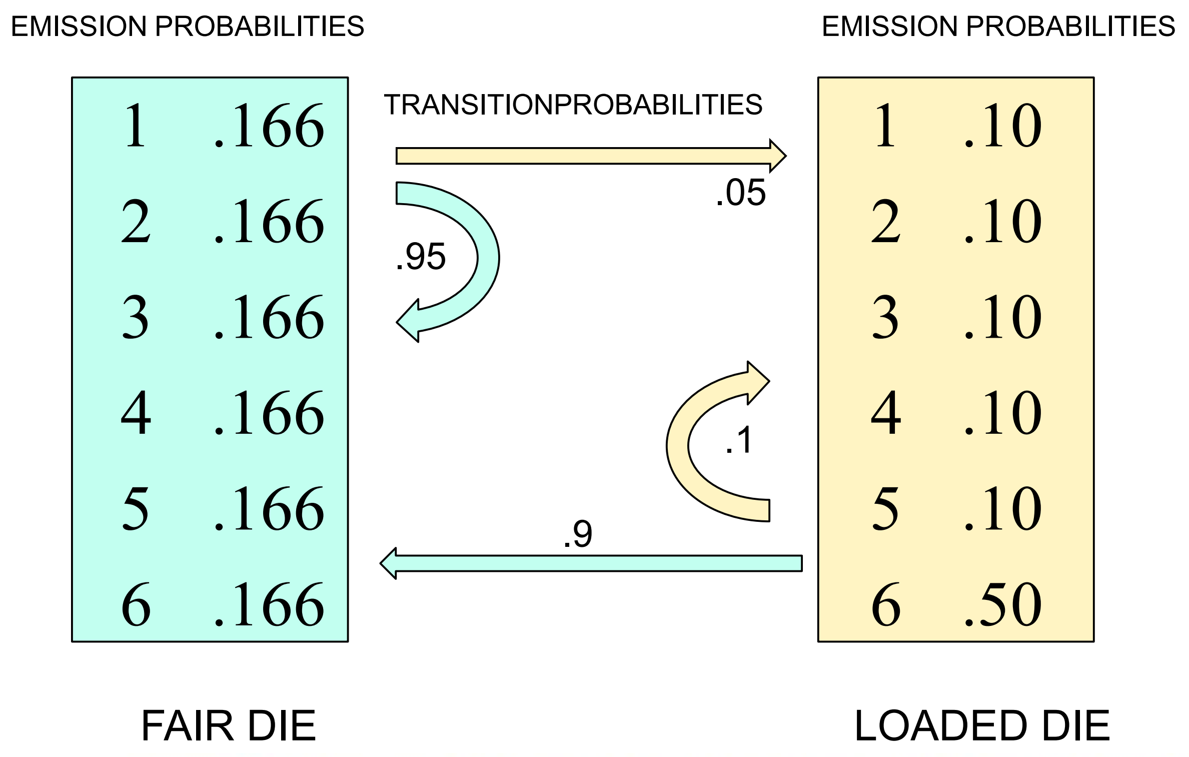
  This is the ocean→island, island →ocean concept

•There is a probability of <u>emissions</u> within each state

  This is the transition matrix within an island (or ocean) concept

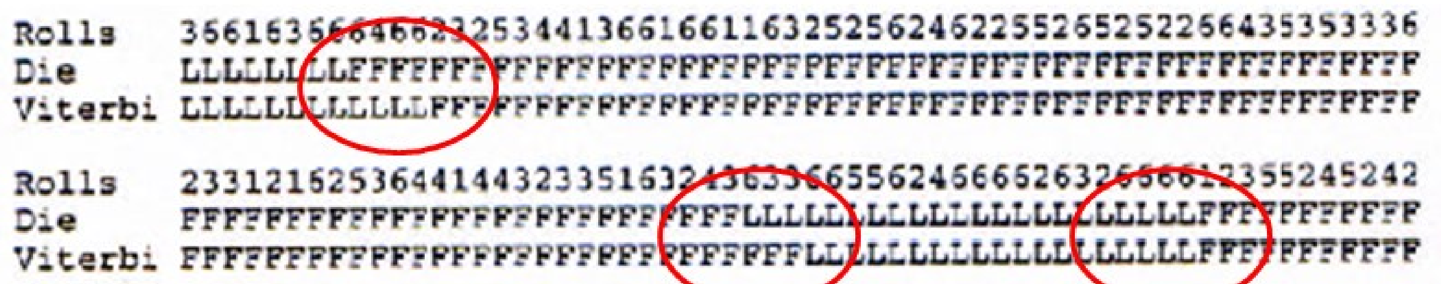
HMM Example – the Dishonest Casino

The casino uses two dice, one fair and another loaded with a 50% chance of rolling a six. In one of 20 rolls of a single die, the dealer will slip in the loaded die, but keep it in play only one in ten rolls. Can we detect this mischief?



1	.166
2	.166
3	.166
4	.166
5	.166
6	.166

1	.10
2	.10
3	.10
4	.10
5	.10
6	.50



```
Rolls   366163666466232534413661661163252562462255265252266435353336
Die     LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi LLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls   233121625364414432335163243633665562466652632666612355245242
Die     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLFFFFFFFFF
Viterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLFFFFFFFF
```

From Durbin,Eddy *et al* 1998

Hidden Markov Models (HMMs)

- Why Markov?
  - Because reducing the problem to transitions between states requires the Markov assumption for mathematical tractability.
- Why hidden?
  - Because ***only the emission is observed***. It is to be deduced what state (hidden from us) generated the emission.

BUILDING THE MODEL Definitions/Notation

Sequence of symbols x_i	The ordered list of emissions observed
The path π	The sequence of states which generated the observed symbols
Transition probability a_{ij}	Probability of moving from one path position to the next
Emission probability $\varepsilon_i(b)$	Probability of the given state i emitting symbol b

USING THE MODEL

Sometimes we are interested in the probability of all paths that will result in the sequence of emissions that we observe. To do this, we use the Forward Algorithm

But very often we are interested in the most probable path through the data that will generate the observed sequence. To do this, we use the Viterbi Algorithm

Because of the Markov assumption, the joint probability of an observed sequence of symbols, **x** , and the sequence of states, **π**, can easily be expressed by

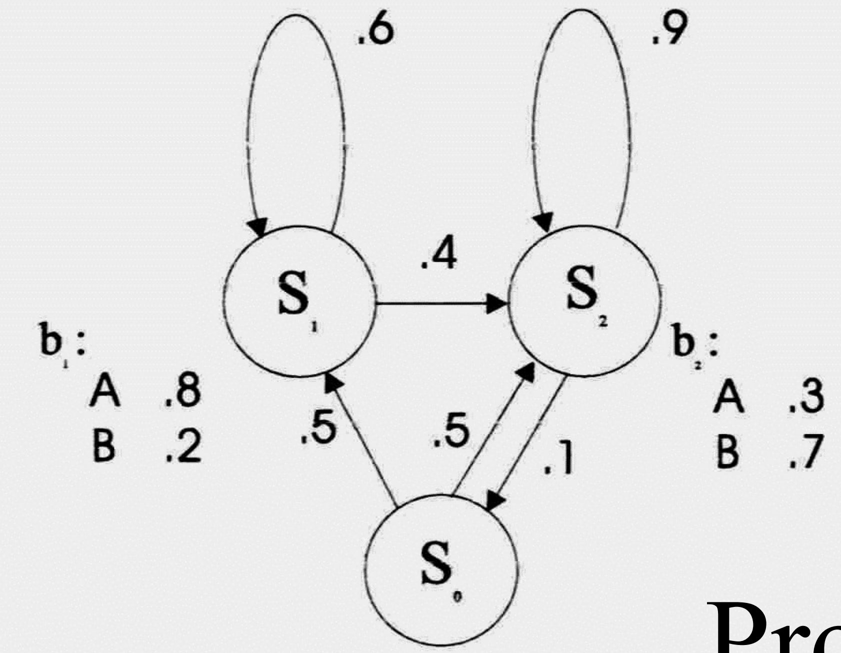
$$p(x,\pi) = \underbrace{a_{0\pi_1}}_{\text{Transition probability from start state to state } \pi_1} \prod_{i=1}^{Len\,of\,x} \underbrace{e_{\pi_i}(x_i)}_{\text{Emission probability of symbol } x_i \text{ in state } \pi_i} \underbrace{a_{\pi_i\pi_{i+1}}}_{\text{Transition probability from state } \pi_i \text{ to state } \pi_{i+1}}$$

where $x_i \in \mathbf{x}$, $\pi_i \in \boldsymbol{\pi}$

We may wish to find the total probability $p(\mathbf{x})$ that we could observe **x**

or

we may wish to find the most likely path that will emit **x**, $\underset{\pi}{\text{argMax}}\ p(x,\pi)$

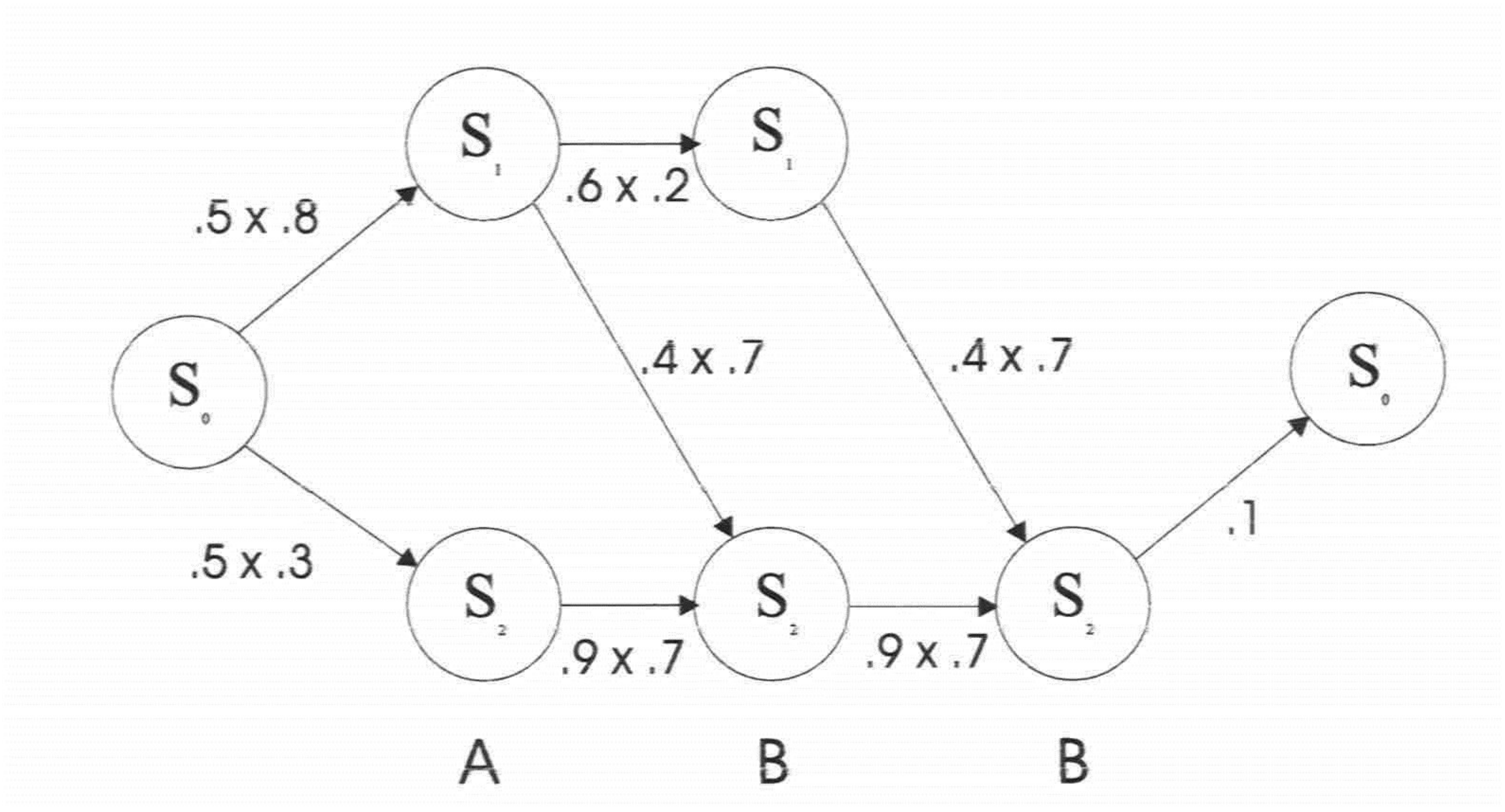


K → e ↓	S1	S2
A	.8	.3
B	.2	.7

$$\{a_{i,j}\} = \begin{array}{c|ccc} & s_0 & s_1 & s_2 \\ s_0 & 0 & .5 & .5 \\ s_1 & 0 & .6 & .4 \\ s_2 & .1 & 0 & .9 \end{array}$$

Prokaryotic Gene Recognition A very simple example

HMM path probabilities when observing the sequence of symbols A-B-B



The Probability of All Paths Emitting an Observed Sequence
The Forward Algorithm

Define a function relating all joint probabilities up to the observation of interest,x_i

$$f_k(i) = P(x_1 x_2 \ldots\ldots x_{i,} \pi = k)$$

Then recurse

Forward variable for the *l*th state at time i+1

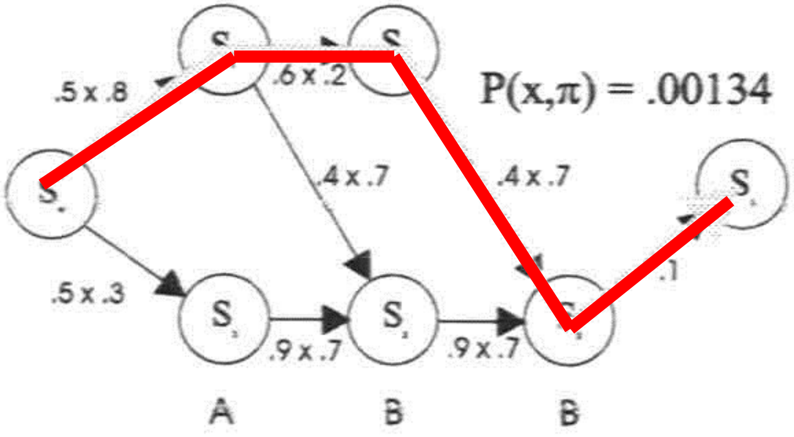
Emission of symbol x_i from the lth state

Forward variable for the *k*th state at time i

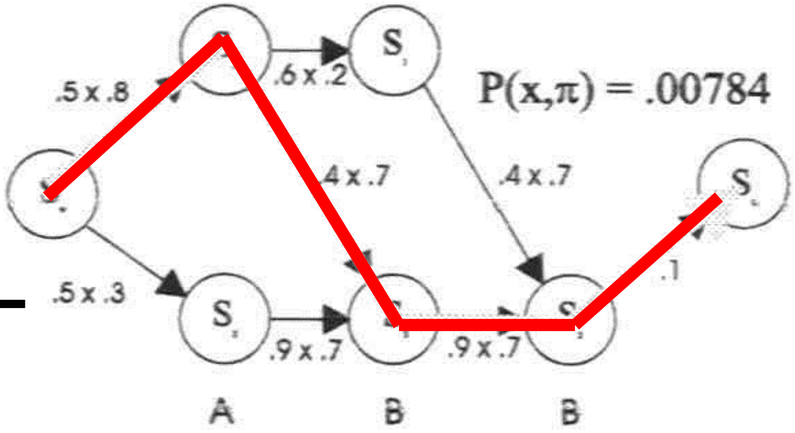
transition from kth to lth state

$$f_l(i+1) = e_l(x_i)\sum_k f_k(i)a_{kl}$$

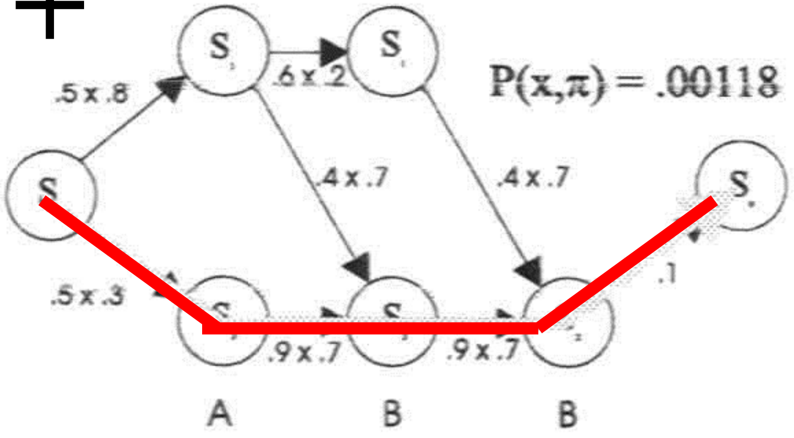
HMM –Forward Algorithm does this:



+



+



= .01036

Most Probable Path

In many problems we seek the most probable path, Certainly in the Dishonest Casino problem, the most probable path is the solution to what letters to expect

We seek

$$\pi^* = \underset{\pi}{\mathrm{argMax}}\ p(x,\pi)$$

But this is a hard problem…

Viterbi Algorithm

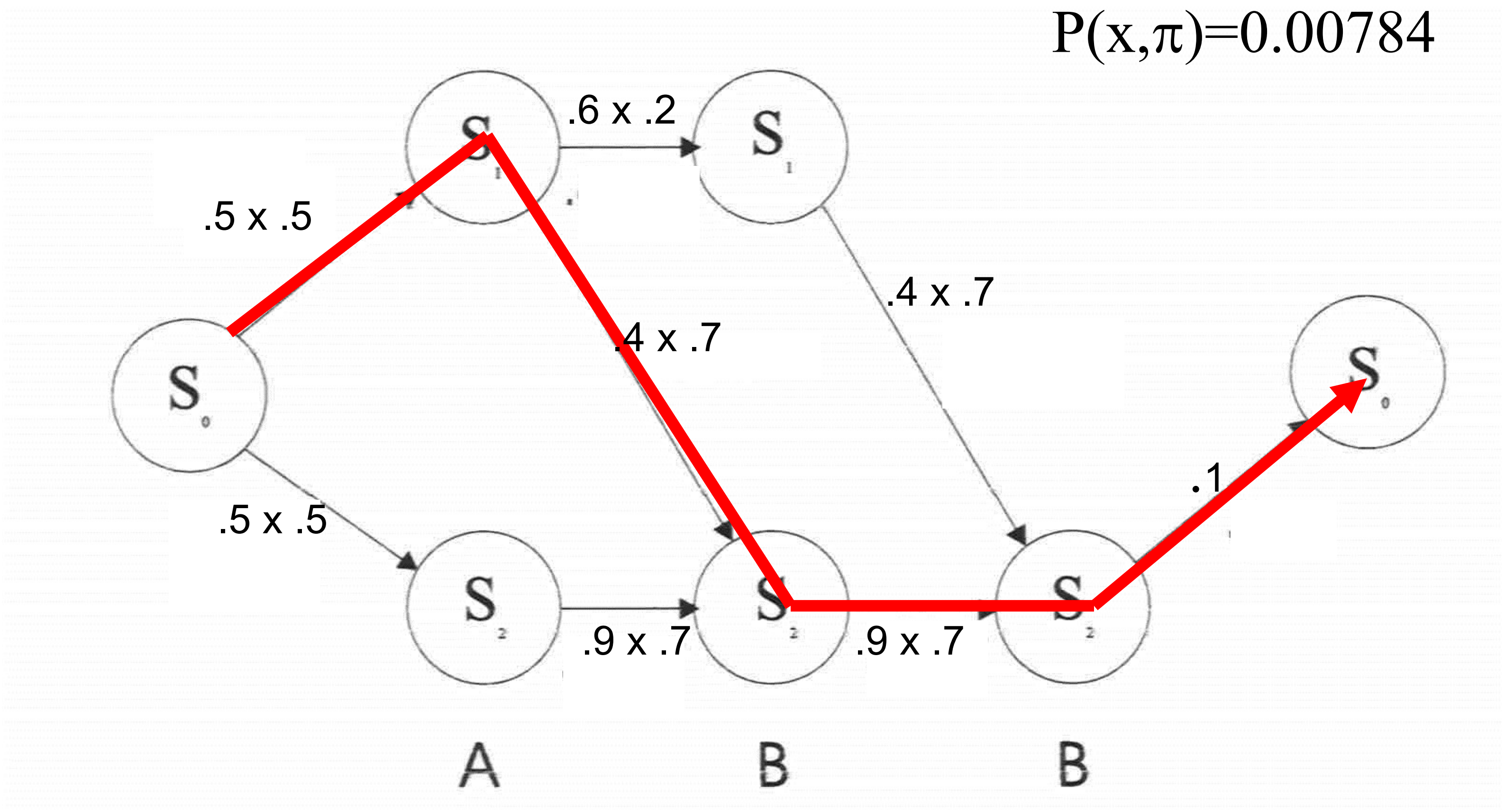
$\pi_k(i)$ is the most probable path ending in state k with observation (emission) i

Assume $\pi_k(i)$ is known for <u>all</u> the <u>states</u> k

Then, recursing,

$$\pi_l(i+1) = e_l(x_{i+1}) \max(\pi_k(i) a_{kl})$$

HMM –Viterbi Algorithm does this:



With this dynamic programming approach, the problem is now tractable (of polynomial complexity)

Missing Parameters

- Sometimes we don’t know the emission or transition probabilities
- Sometimes we don’t even know the path

We are given <u>only</u> the emissions and we must infer the parameters

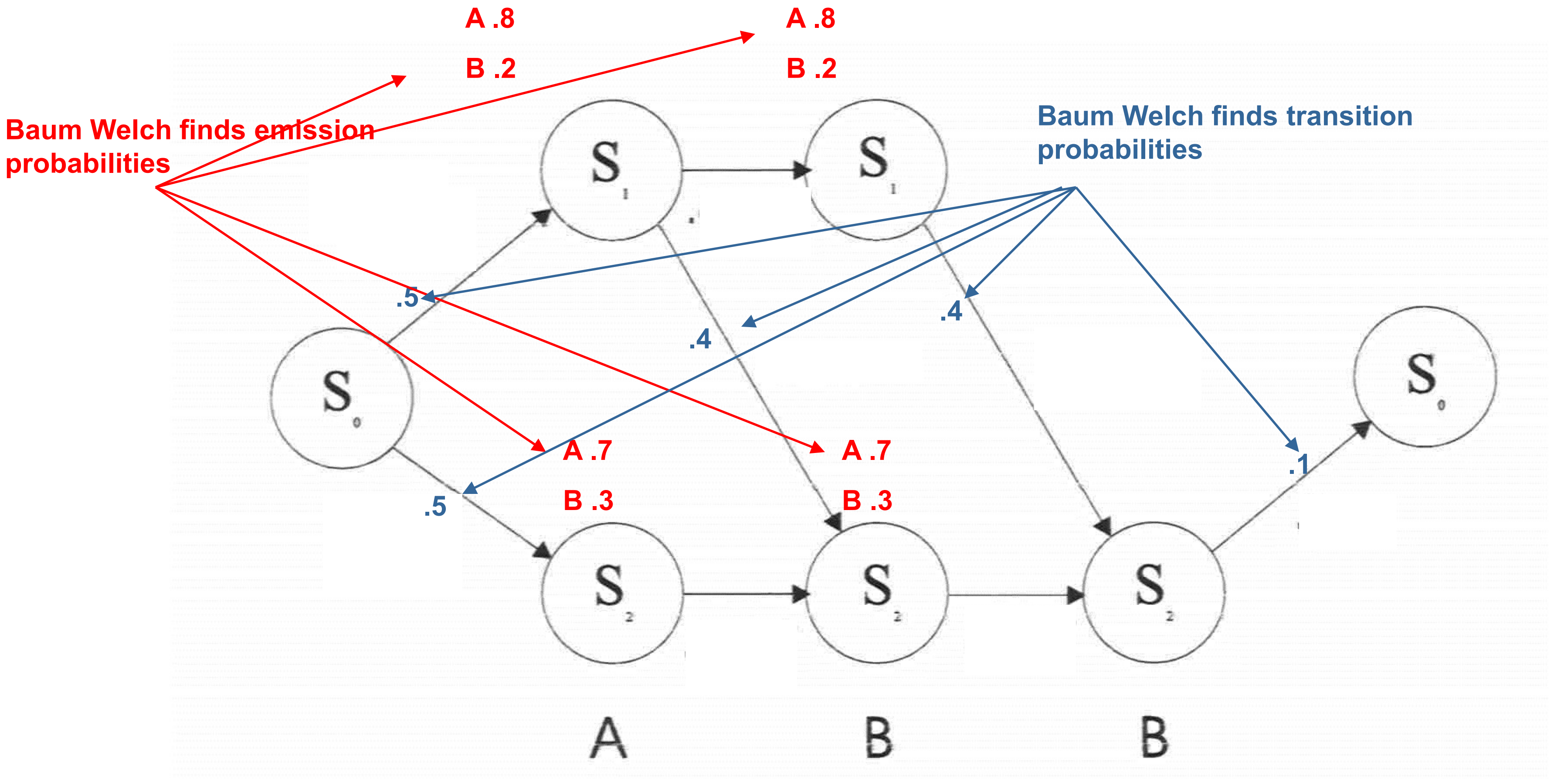
E-M

As we have done in other cases of hidden parameters, we call upon the Expectation-Maximization algorithm to find the missing parameters

Like all E-M

- the solution will be local
- the solution will be sensitive to pseudocounts
- the solution will be sensitive to initial guesses

The Baum-Welch algorithm is a special E-M algorithm that is suited specifically for the HMM model. B-W computes the requisite parameters.


A .8
B .2
A .8
B .2
Baum Welch finds emission probabilities
Baum Welch finds transition probabilities
S₁
S₁
S₀
S₀
S₂
S₂
S₂
.5
.4
.4
.5
.1
A .7
B .3
A .7
B .3
A
B
B

- Forward Algorithm gives probability of being in a given state at time t ($0<t<T$) given the sequence data thus far
- Backward Algorithm gives *a posteriori* probability that an observation came from a given state within the observed sequence when the entire emitted sequence is known

The Backward Algorithm Implements Posterior Decoding

- Recurse backward from the end of the sequence

$$b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$$

for i = L –1 ….1

The Posterior Probability of an Observed Sequence

Write product of forward and backward variables. This product is the forward probability of arriving in the current state and the backward probability of generating the final state, given the current state

$$P(x, \pi_i = k) = f_k(i) b_k(i)$$

Posterior Probability Re-written

We can re-write

$$P(x, \pi_i = k)$$

as the conditional probability

$$P(\pi_i = k \mid x) = \frac{f_k(i) b_k(i)}{P(x)}$$

This is the ‘guts’ of the BW Algorithm

#1

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

$$e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

#2

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(1+1)$$

#3

$$E_k(b) = \sum_j \frac{1}{P(x^j)} \sum_{\{i|x_i^j=b\}} f_k^j(i) b_k^j(i)$$

Baum Welch

- Use multiple training sequences
- Set parameters to a guess
- Set A and E variables to pseudo-count

EXPECTATION

$\forall$ sequence j=1…n

- Calculate $f_k(i)$ for sequence j using the forward algorithm
- Calculate $b_k(i)$ for sequence j using the backward algorithm

MAXIMIZATION

- Add the contribution of sequence j to A (#2) and E (#3)
- Calculate new model parameters using #1

- Stop when no change

A Strategy for Gene Finding Using an HMM

1. Build a Model
   - Understand the gene structure
     - Donor and acceptor sites, if applicable
     - Intergenic regions
     - UTRs
   - Decide the order of the Markov model
   - Decide whether to go with base, dinucleotide, codon, n-mer

2. Train the model on known genes
   - Learn with the BW Algorithm
   - Transition probabilities
   - Emission probabilities

3. Parse (decode) unknown genes using the model

Profiles

- Often we seek a signal that can vary from gene to gene within a species; we may have simply the consensus sequence for the signal.
- A matching algorithm won’t work, we need to detect an instance of the consensus
- PSSMs are well suited for this task

PSSMs

The HMM can handle the concept of a PSSM. Stormo* gives the example of several promotors from the -10 region that have one of two consensus sequences: TATAAT or TATRNT. The PSSM below encodes the sequence TATAAT.

A	–38	19	1	12	10	–48
C	–15	–38	–8	–10	–3	–32
G	–13	–49	–6	–7	–10	–48
T	17	–32	8	–6	–6	19

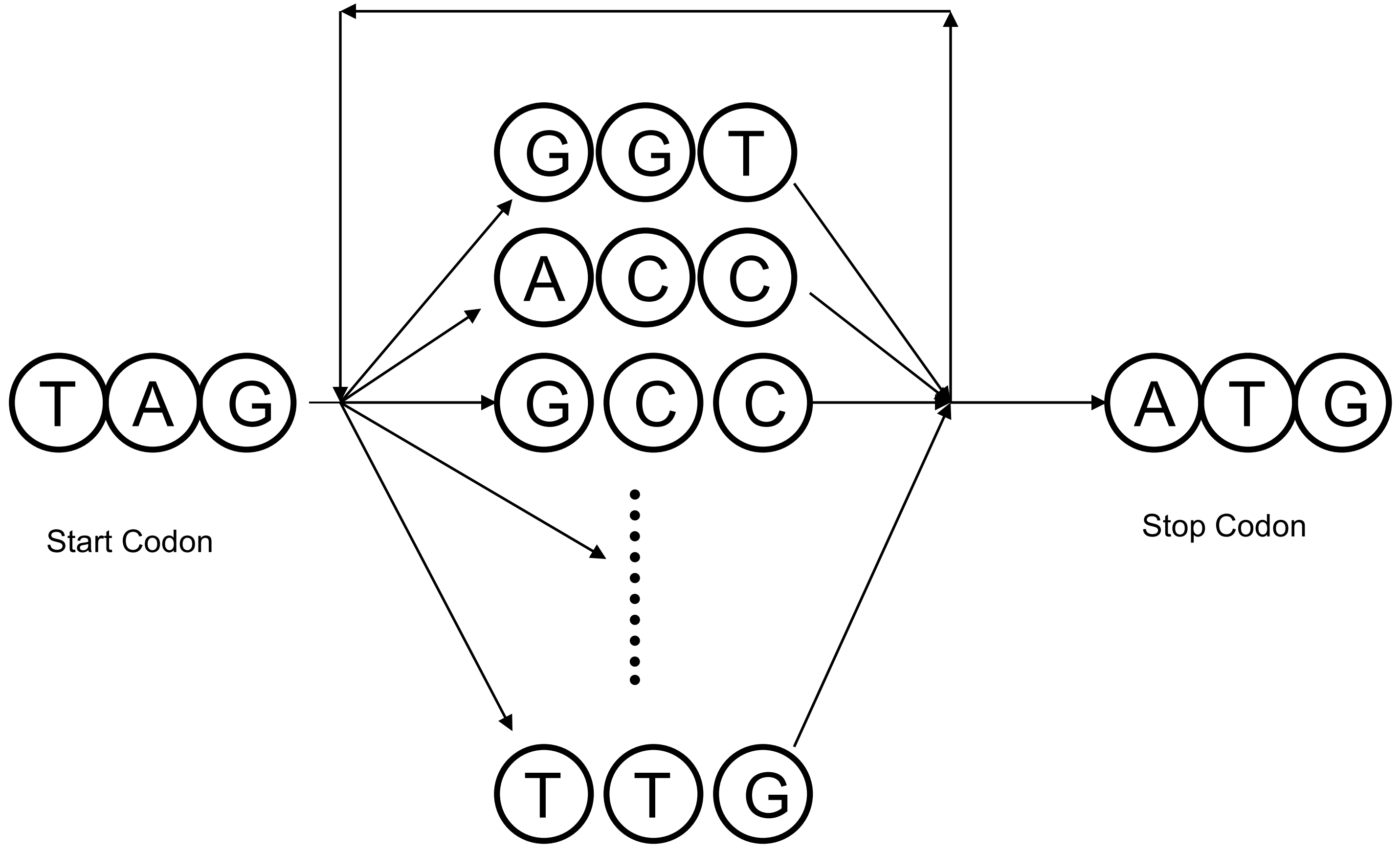
Stormo, Gary D DNA Binding Sites: Representation and Discovery *Bioinformatics* 16(no 1) 2000, *pp16-23*

HMM as a Parser

Krough:

- Introns and exons are words in a regular language
  - A sentence always begins and ends with an exon
  - Introns alternate with exons- never two exons or two introns in succession
- The HMM (a complicated FSM) can parse this language

Single gene model using codon elements rather than single base elements



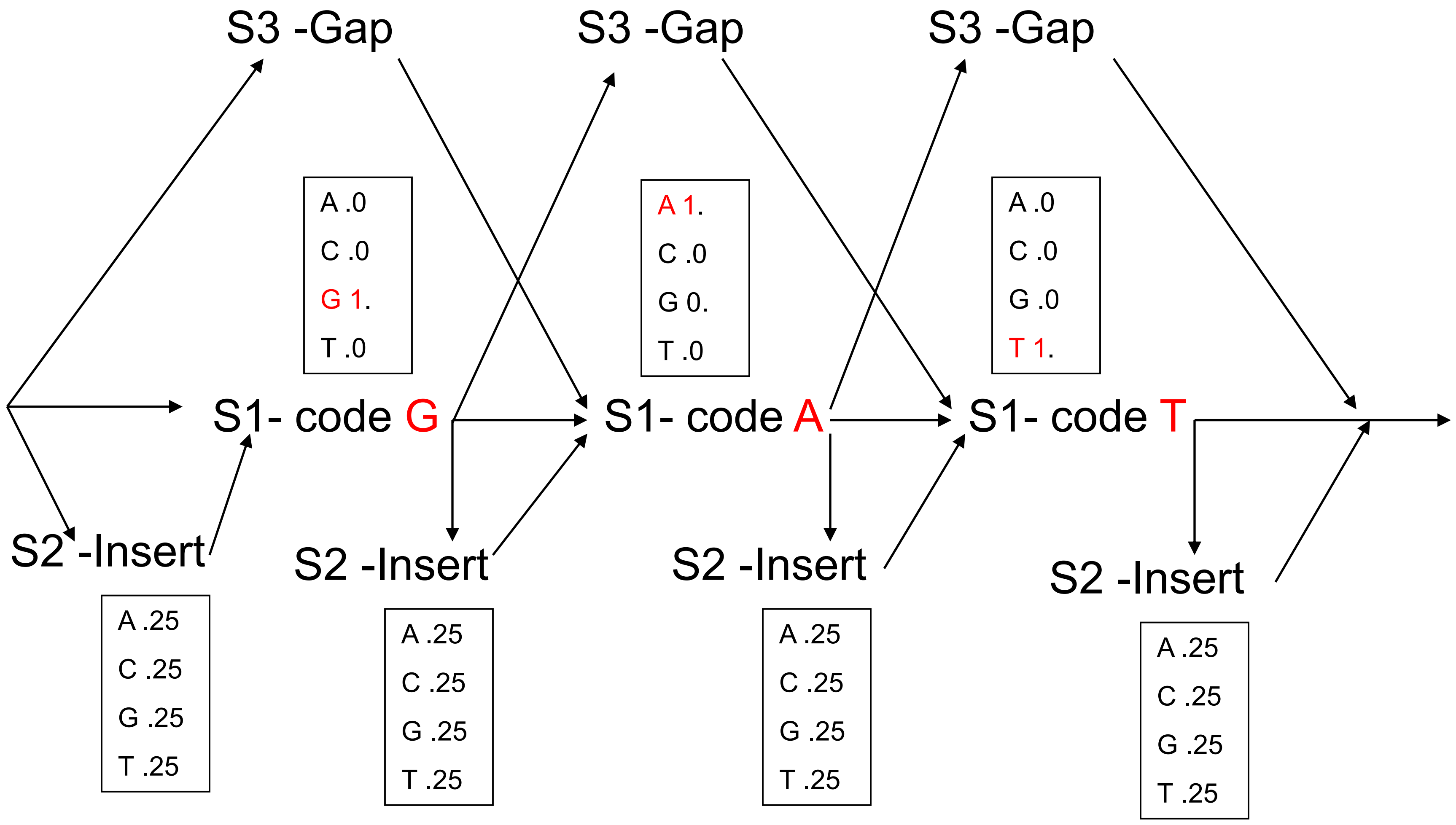
Start Codon

Stop Codon

61 possible Codons

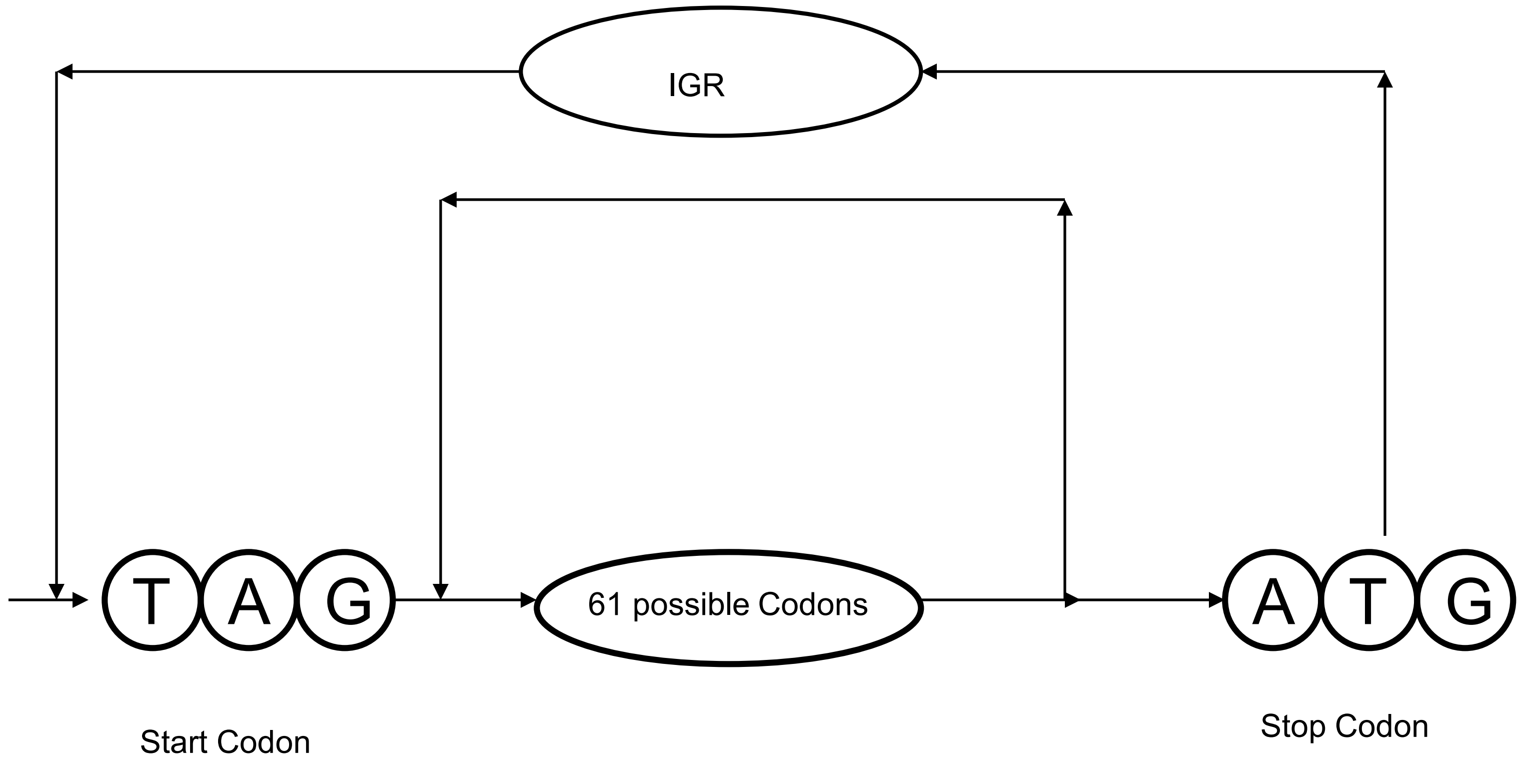
After Krogh 1994

Aspartamine (GAT) codon model



After Krogh 1994

A prokaryotic genome model



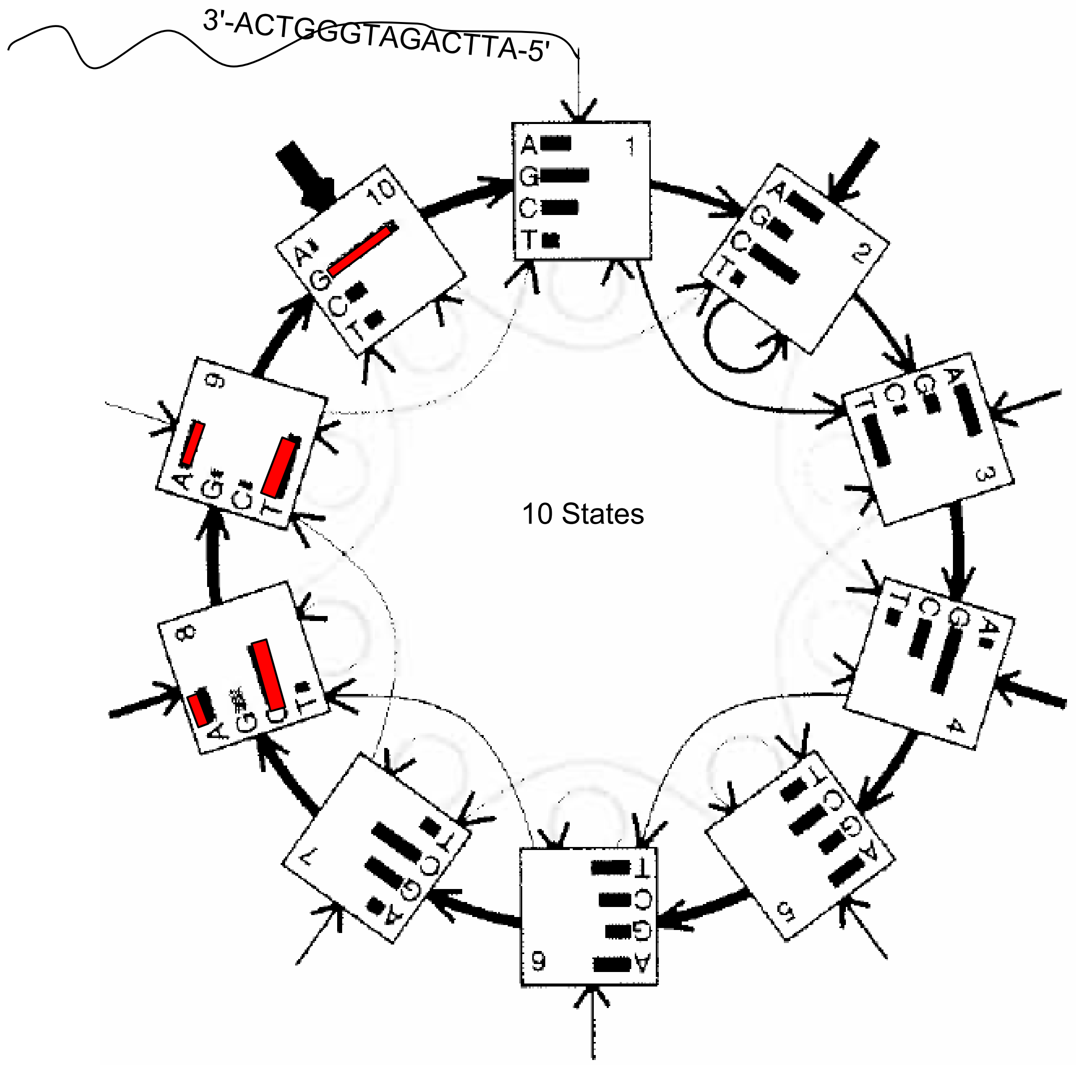
After Krogh 1994

Special Problems with Eukaryotic Gene Finding

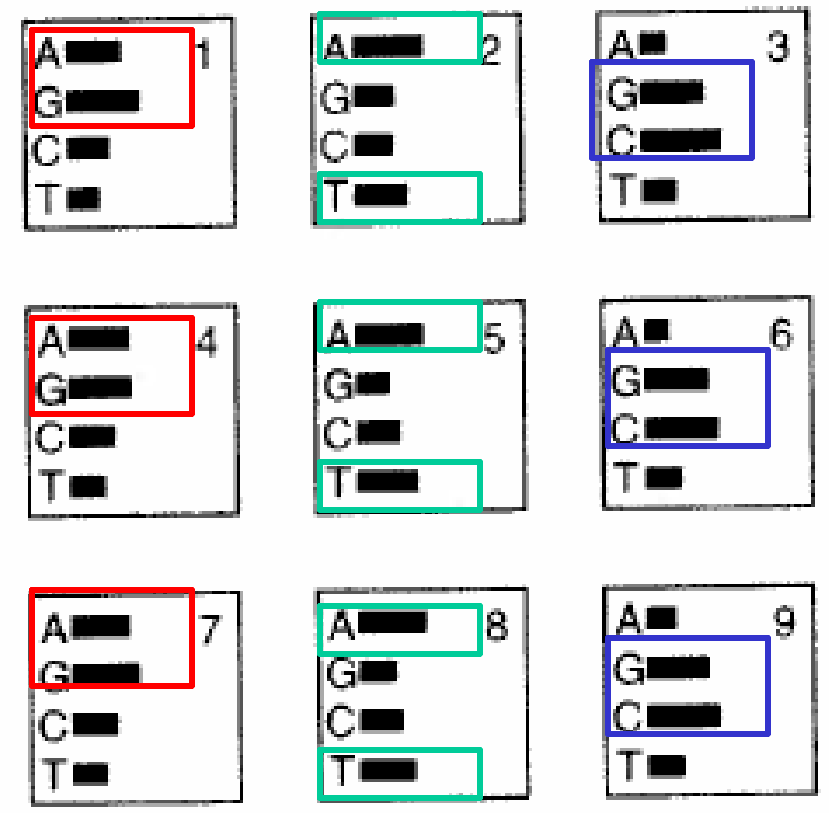
- Must train on known intron-exon and exon-intron junctions
- Must find promoter motifs
- Must have ORF awareness

Here is a circular HMM of 10 symbols. As the coding DNA string is 'threaded' into the model, a recurrence of nucleotide content is evidenced at 8,9,10.

The model length is not a multiple of 3, so that codon context is not detected, although codon content (*vis à vis* random DNA) is, in fact, detected



Here are the 9 sets of emissions from a 9 position circular HMM. Notice the recurrence of the content mod 3, implying synchrony with the codons, and a more nearly accurate representation of coding DNA codon content



From Bioinformatics The Machine Learning Approach Baldi, P and Brunak, S MIT Press Cambridge 1998

Higher Order HMMs

- For detection of certain isolated patterns, particularly in a promotor, higher order HMMs are appropriate. In a second order HMM, for example, instead of the transition C|B, we would have C|B,A

- The higher the order, the more it looks like an exact string match

- Consider A|T,A,T as an application of this idea in a 3rd order HMM transition

Classical HMM application

- •Speech Recognition

HMM applications in Bioinformatics

- •Gene Finding
- •Protein secondary structure ID
- •Protein tertiary structure ID
- •RNA structure prediction
- •Sequence alignment (particularly MSA)
- •Tree construction in phylogeny

Popular Gene Finders

- HMM (1ST ORDER) : Genemark
- HMM (5TH ORDER) : Glimmer
- ANN-HMM: Grail
- Dynamic Programming: GeneParser

The ANN and HMM Hybrid Model for Gene Finding

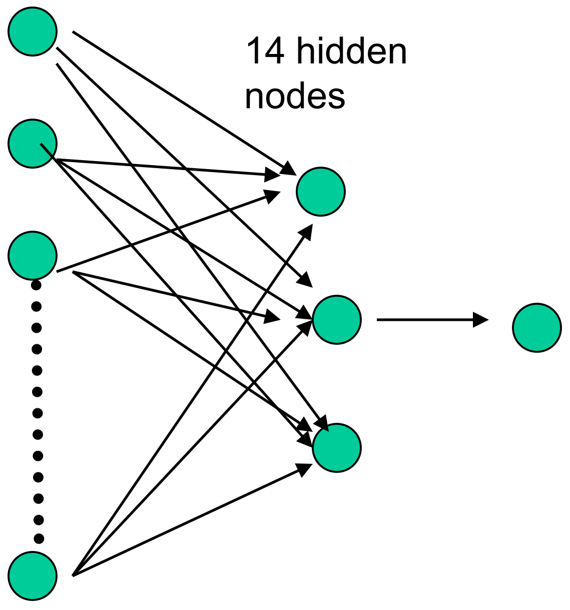
- Combining technologies such as ANNs and HMMs can lead to complex, sophisticated, and highly accurate gene predictors.
- Sometimes the ANNs and the HMMs are trained separately (on different tasks) then combined
- A true hybrid will inter-couple the ANN output, as a gate, to the HMM emission, trained as a single unit

ANNs: GRAIL INPUTS

- Frame Bias Matrix (frequency of each base in each frame)
- 3-periodicity of nuleotides (Fickett feature)
- Coding sextuple word preferences
  - Word preferences in frame 1
  - Word preferences in frame 2
  - Word preferences in frame 3
- Maximum word preferences in frames
- Sextuple word communality
- Repetitive sextuple word

Grail
ANN Inputs

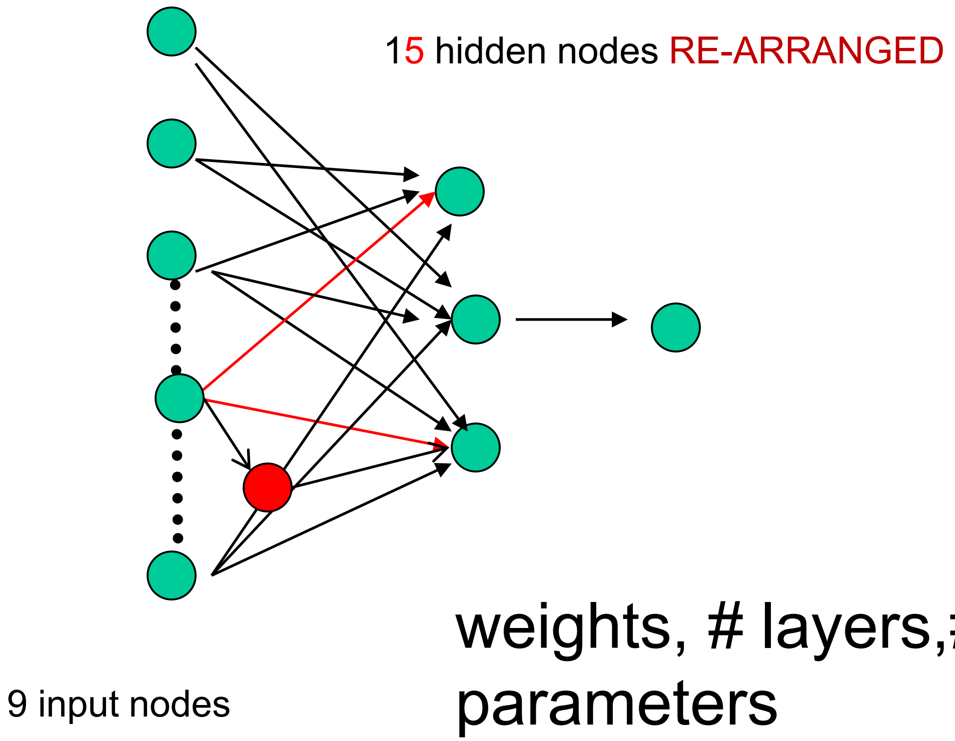
GRAIL inputs



9 input nodes

after Uberbacher and Mural, 1991

Genetic Algorithm modification



weights, # layers,# nodes,
parameters
all modified by GA