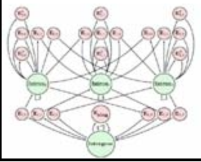
Computational Genomics

10-810/02-710, Spring 2009

Gene Finding and HMM

Eric Xing

Lecture 5, January 28, 2009



Reading: Durbin Chap 3, class assignment

The HMM Algorithms

Questions:

- **Decoding**: What is the most likely DNA parsing? Viterbi
- **Evaluation**: What is the probability of the observed sequence? Forward
- **Decoding**: What is the probability that the state of the 3rd position is Bk or gene, given the observed sequence? Forward-Backward
- **Learning**: Under what parameterization are the observed sequences most probable? Baum-Welch (EM)

The likelihood of a sequence

- We want to calculate $P(\mathbf{x})$, the likelihood of $\mathbf{x}$, given the HMM
  - Sum over all possible ways of generating $\mathbf{x}$:

$$p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x},\mathbf{y}) = \sum_{y_1}\sum_{y_2}\cdots\sum_{y_N} \pi_{y_1}\prod_{t=2}^{T} a_{y_{t-1},y_t}\prod_{t=1}^{T} p(x_t \mid y_t)$$

- Complexity?
- Why useful?

The Forward Algorithm

- We want to calculate $P(\mathbf{x})$, the likelihood of $\mathbf{x}$, given the HMM
  - Sum over all possible ways of generating $\mathbf{x}$:

$$p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x},\mathbf{y}) = \sum_{y_1}\sum_{y_2}\cdots\sum_{y_N} \pi_{y_1}\prod_{t=2}^{T} a_{y_{t-1},y_t}\prod_{t=1}^{T} p(x_t \mid y_t)$$

  - To avoid summing over an exponential number of paths $\mathbf{y}$, define

$$\alpha(y_t^k = 1) = \alpha_t^k \overset{\text{def}}{=} P(x_1, \ldots, x_t, y_t^k = 1)$$

(the forward probability)

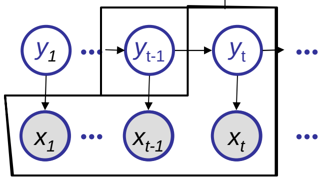
  - The recursion:

$$\alpha_t^k = p(x_t \mid y_t^k = 1)\sum_i \alpha_{t-1}^i a_{i,k}$$

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Forward Algorithm – derivation

- Compute the forward probability:



$$\begin{aligned}
\alpha_t^k &= P(x_1,\ldots,x_{t-1},x_t,y_t^k=1)\\
&= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},x_t,y_{t-1},y_t^k=1)\\
&= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1})P(y_t^k=1\mid y_{t-1},x_1,\ldots,x_{t-1})P(x_t\mid y_t^k=1,x_1,\ldots,x_{t-1},y_{t-1})\\
&= \sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1})P(y_t^k=1\mid y_{t-1})P(x_t\mid y_t^k=1)\\
&= P(x_t\mid y_t^k=1)\sum_i P(x_1,\ldots,x_{t-1},y_{t-1}^i=1)P(y_t^k=1\mid y_{t-1}^i=1)\\
&= P(x_t\mid y_t^k=1)\sum_i \alpha_{t-1}^i a_{i,k}
\end{aligned}$$

Chain rule: $P(A,B,C)=P(A)P(B\mid C)P(C\mid A,B)$

The Forward Algorithm

- We can compute α_t^k for all k, t, using dynamic programming!

Initialization:

$$\alpha_1^k = P(x_1\mid y_1^k=1)\pi_k$$

$$\begin{aligned}
\alpha_1^k &= P(x_1,y_1^k=1)\\
&= P(x_1\mid y_1^k=1)P(y_1^k=1)\\
&= P(x_1\mid y_1^k=1)\pi_k
\end{aligned}$$

Iteration:

$$\alpha_t^k = P(x_t\mid y_t^k=1)\sum_i \alpha_{t-1}^i a_{i,k}$$

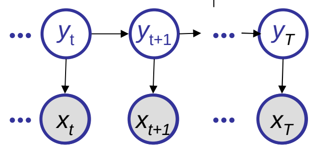
Termination:

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Backward Algorithm

- We want to compute $P(y_t^k = 1 \mid \mathbf{x})$,

  the posterior probability distribution on the t th position, given $\mathbf{x}$



  - We start by computing

$$
\begin{aligned}
P(y_t^k = 1, \mathbf{x}) &= P(x_1, \ldots, x_t, y_t^k = 1, x_{t+1}, \ldots, x_T) \\
&= P(x_1, \ldots, x_t, y_t^k = 1) P(x_{t+1}, \ldots, x_T \mid x_1, \ldots, x_t, y_t^k = 1) \\
&= P(x_1 \ldots x_t, y_t^k = 1) P(x_{t+1} \ldots x_T \mid y_t^k = 1)
\end{aligned}
$$

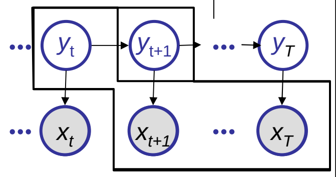
Forward, α_t^k Backward, $\beta_t^k = P(x_{t+1}, \ldots, x_T \mid y_t^k = 1)$

  - The recursion:

$$\beta_t^k = \sum_i a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

The Backward Algorithm – derivation

- Define the backward probability:



$$
\begin{aligned}
\beta_t^k &= P(x_{t+1}, \ldots, x_T \mid y_t^k = 1) \\
&= \sum_{y_{t+1}} P(x_{t+1}, \ldots, x_T, y_{t+1} \mid y_t^k = 1) \\
&= \sum_i P(y_{t+1}^i = 1 \mid y_t^k = 1) p(x_{t+1} \mid y_{t+1}^i = 1, y_t^k = 1) P(x_{t+2}, \ldots, x_T \mid x_{t+1}, y_{t+1}^i = 1, y_t^k = 1) \\
&= \sum_i P(y_{t+1}^i = 1 \mid y_t^k = 1) p(x_{t+1} \mid y_{t+1}^i = 1) P(x_{t+2}, \ldots, x_T \mid y_{t+1}^i = 1) \\
&= \sum_i a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i
\end{aligned}
$$

Chain rule: $P(A, B, C \mid \alpha) = P(A, \alpha) P(B \mid C, \alpha) P(C \mid A, B, \alpha)$

The Backward Algorithm

- We can compute β_t^k for all k, t, using dynamic programming!

<u>**Initialization:**</u>

$$\beta_T^k = 1, \ \forall k$$

<u>**Iteration:**</u>

$$\beta_t^k = \sum_i a_{k,i} P(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

<u>**Termination:**</u>

$$P(\mathbf{x}) = \sum_k \alpha_1^k \beta_1^k$$

Example:

Posterior decoding

- We can now calculate

$$P(y_t^k = 1 \mid \mathbf{x}) = \frac{P(y_t^k = 1, \mathbf{x})}{P(\mathbf{x})} = \frac{\alpha_t^k \beta_t^k}{P(\mathbf{x})}$$

- Then, we can ask
  - What is the most likely state at position *t* of sequence $\mathbf{x}$:

$$k_t^* = \arg\max_k P(y_t^k = 1 \mid \mathbf{x})$$

  - Note that this is an MPA of a single hidden state, what if we want to a MPA of a whole hidden state sequence?
  - Posterior Decoding: $\left\{ y_t^{k_t^*} = 1 : t = 1 \cdots T \right\}$
  - This is different from MPA of a whole sequence of hidden states
  - This can be understood as *bit error rate* vs. *word error rate*

Example:
MPA of *X* ?
MPA of *(X, Y)* ?

x	*y*	*P(x,y)*
0	*0*	*0.35*
0	*1*	*0.05*
1	*0*	*0.3*
1	*1*	*0.3*

Computational Complexity and implementation details

- What is the running time, and space required, for Forward, and Backward?

$$\alpha_t^k = p(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

$$V_t^k = p(x_t \mid y_t^k = 1) \max_i a_{i,k} V_{t-1}^i$$

Time: $O(K^2 N)$; Space: $O(KN)$.

- Useful implementation technique to avoid underflows
  - Viterbi: sum of logs
  - Forward/Backward: rescaling at each position by multiplying by a constant

Learning HMM: two scenarios

- **Supervised learning**: estimation when the “right answer” is known
  - **Examples:**

    GIVEN: a genomic region x = $x_1 \ldots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands

    GIVEN: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

- **Unsupervised learning**: estimation when the “right answer” is unknown
  - **Examples:**

    GIVEN: the porcupine genome; we don’t know how frequent are the CpG islands there, neither do we know their composition

    GIVEN: 10,000 rolls of the casino player, but we don’t see when he changes dice

- **QUESTION:** Update the parameters θ of the model to maximize $P(x|\theta)$ --- Maximal likelihood (ML) estimation

Supervised ML estimation

- Given $x = x_1 \ldots x_N$ for which the true state path $y = y_1 \ldots y_N$ is known,
  - **Define:**

    A_{ij} = # times state transition $i \rightarrow j$ occurs in $\mathbf{y}$

    B_{ik} = # times state i in $\mathbf{y}$ emits k in $\mathbf{x}$

  - **We can show that the maximum likelihood parameters θ are:**

$$a_{ij}^{ML} = \frac{\#(i \rightarrow j)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=2}^T y_{n,t-1}^i y_{n,t}^j}{\sum_n \sum_{t=2}^T y_{n,t-1}^i} = \frac{A_{ij}}{\sum_{j'} A_{ij'}}$$

$$b_{ik}^{ML} = \frac{\#(i \rightarrow k)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=1}^T y_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^T y_{n,t}^i} = \frac{B_{ik}}{\sum_{k'} B_{ik'}}$$

  - **What if y is continuous? We can treat $\{(x_{n,t}, y_{n,t}) : t = 1:T, n = 1:N\}$ as $N \times T$ observations of, e.g., a Gaussian, and apply learning rules for Gaussian …**

Supervised ML estimation, ctd.

- **Intuition:**
  - When we know the underlying states, the best estimate of θ is the average frequency of transitions & emissions that occur in the training data

- **Drawback:**
  - Given little data, there may be **overfitting**:
    - P(x|θ) is maximized, but θ is unreasonable

      0 probabilities – VERY BAD

- **Example:**
  - Given 10 casino rolls, we observe

```
x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3
y = F, F, F, F, F, F, F, F, F, F
```

  - Then: $a_{FF} = 1$; $a_{FL} = 0$

    $b_{F1} = b_{F3} = .2$;

    $b_{F2} = .3$; $b_{F4} = 0$; $b_{F5} = b_{F6} = .1$

Pseudocounts

- Solution for small training sets:
  - Add pseudocounts

    A_{ij} = # times state transition $i \rightarrow j$ occurs in $\mathbf{y}$ + R_{ij}

    B_{ik} = # times state i in $\mathbf{y}$ emits k in $\mathbf{x}$ + S_{ik}

  - R_{ij}, S_{ij} are pseudocounts representing our prior belief
  - Total pseudocounts: $R_i = \Sigma_j R_{ij}$, $S_i = \Sigma_k S_{ik}$,
    - --- "strength" of prior belief,
    - --- total number of imaginary instances in the prior

- Larger total pseudocounts $\Rightarrow$ strong prior belief

- Small total pseudocounts: just to avoid 0 probabilities --- smoothing

Unsupervised ML estimation

- Given $x = x_1 \ldots x_N$ for which the true state path $y = y_1 \ldots y_N$ is **unknown**,
  - **EXPECTATION MAXIMIZATION**

0. Starting with our best guess of a model M, parameters θ:
1. Estimate A_{ij} , B_{ik} in the training data
   - How? $A_{ij} = \sum_{n,t} \langle y^i_{n,t-1} y^j_{n,t} \rangle \quad B_{ik} = \sum_{n,t} \langle y^i_{n,t} \rangle x^k_{n,t}$,
2. Update θ according to A_{ij} , B_{ik}
   - Now a "supervised learning" problem
3. Repeat 1 & 2, until convergence

This is called the Baum-Welch Algorithm

We can get to a provably more (or equally) likely parameter set θ each iteration

How to compute expected count?

$$B_{ik} = \sum_{n,t} \langle y^i_{n,t} \rangle x^k_{n,t}$$

$$\langle y^i_{n,t} \rangle = P(Y^i_{n,t} = 1 | \mathbf{x}_n) = \frac{\alpha^i_{n,t} \beta^i_{n,t}}{P(\mathbf{x}_n)}$$

$$A_{ij} = \sum_{n,t} \langle y^i_{n,t-1} y^j_{n,t} \rangle$$

$$\langle y^i_{n,t-1} y^j_{n,t} \rangle = P(Y^i_{n,t-1} = 1, Y^j_{n,t} = 1 | \mathbf{x}_n) = \frac{\alpha^i_{n,t-1} a_{i,j} x^j_{n,t} \beta^j_{n,t}}{P(\mathbf{x}_n)}$$

The Baum Welch algorithm

- The complete log likelihood

$$\ell_c(\boldsymbol{\theta}; \mathbf{x}, \mathbf{y}) = \log p(\mathbf{x}, \mathbf{y}) = \log \prod_n \left(p(y_{n,1}) \prod_{t=2}^{T} p(y_{n,t} \mid y_{n,t-1}) \prod_{t=1}^{T} p(x_{n,t} \mid x_{n,t}) \right)$$

- The expected complete log likelihood

$$\langle \ell_c(\boldsymbol{\theta}; \mathbf{x}, \mathbf{y}) \rangle = \sum_n \left(\langle y_{n,1}^i \rangle_{p(y_{n,1}|\mathbf{x}_n)} \log \pi_i \right) + \sum_n \sum_{t=2}^{T} \left(\langle y_{n,t-1}^i y_{n,t}^j \rangle_{p(y_{n,t-1}, y_{n,t}|\mathbf{x}_n)} \log a_{i,j} \right) + \sum_n \sum_{t=1}^{T} \left(x_{n,t}^k \langle y_{n,t}^i \rangle_{p(y_{n,t}|\mathbf{x}_n)} \log b_{i,k} \right)$$

- EM
  - The E step

$$\gamma_{n,t}^i = \langle y_{n,t}^i \rangle = p(y_{n,t}^i = 1 \mid \mathbf{x}_n)$$

$$\xi_{n,t}^{i,j} = \langle y_{n,t-1}^i y_{n,t}^j \rangle = p(y_{n,t-1}^i = 1, y_{n,t}^j = 1 \mid \mathbf{x}_n)$$

  - The M step ("symbolically" identical to MLE)

$$\pi_i^{ML} = \frac{\sum_n \gamma_{n,1}^i}{N} \qquad a_{ij}^{ML} = \frac{\sum_n \sum_{t=2}^{T} \xi_{n,t}^{i,j}}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i} \qquad b_{ik}^{ML} = \frac{\sum_n \sum_{t=1}^{T} \gamma_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i}$$

The Baum-Welch algorithm -- comments

Time Complexity:

# iterations × O(K^{2}N)

- Guaranteed to increase the log likelihood of the model
- Not guaranteed to find globally best parameters
- Converges to local optimum, depending on initial conditions
- Too many parameters / too large model: Overt-fitting

Higher-order HMMs

- **The Genetic Code**
  - 3 nucleotides make 1 amino acid
  - Statistical dependencies in triplets
- **Question:**
  - Recognize protein-coding segments with an HMM

	U	C	A	G
U	UUU phe UUC UUA leu UUG	UCU UCC ser UCA UCG	UAU tyr UAC UAA *Stop* UAG *Stop*	UGU cys UGC UGA *Stop* UGG *Stop*
C	CUU CUC leu CUA CUG	CCU CCC pro CCA CCG	CAU his CAC CAA gln CAG	CGU CGC arg CGA CGG
A	AUU AUC ile AUA AUG met	ACU ACC thr ACA ACG	AAU asn AAC AAA lys AAG	AGU ser AGC AGA arg AGG
G	GUU GUC val GUA GUG	GCU GCC ala GCA GCG	GAU asp GAC GAA glu GAG	GGU GGC gly GGA GGG

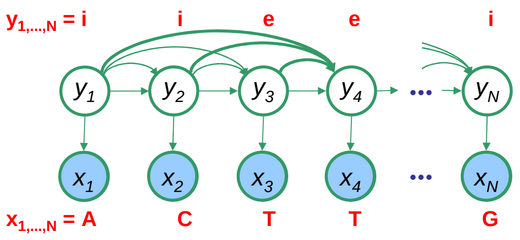
Higher-order HMMs

- Every state of the HMM emits 1 nucleotide
- Transition probabilities:

  Probability of a state at one position, given those of 3 previous positions (triplets):

  $P(y_i \mid y_{i-1}, y_{i-2}, y_{i-3})$
- Emission probabilities:

  $P(x_i \mid y_i)$
- Algorithms extend with small modifications

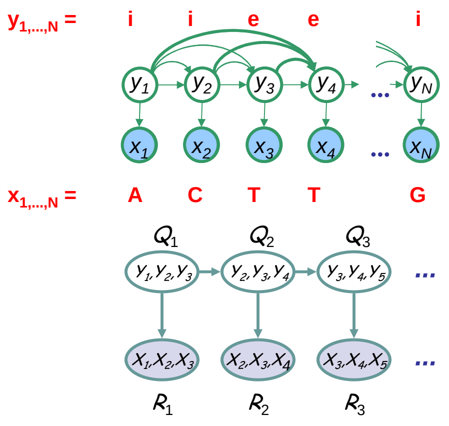


Inference on Higher-order HMMs

- Building 1st-order HMM on "mega" state
- Use FB algorithm as usual
  - $P(Q_2|R)$

  $\rightarrow P(y_2, y_3, y_4|X)$

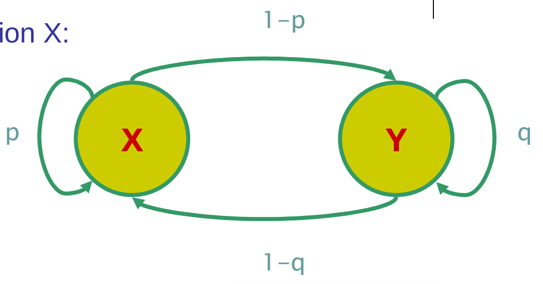
  $\rightarrow P(y_3|X)=\Sigma_{y_2,y_4} P(y_2, y_3, y_4|X)$


$y_{1,\dots,N}$ = i i e e i
y_1 y_2 y_3 y_4 ... y_N
x_1 x_2 x_3 x_4 ... x_N
$x_{1,\dots,N}$ = A C T T G
Q_1 Q_2 Q_3
y_1,y_2,y_3 y_2,y_3,y_4 y_3,y_4,y_5 ...
x_1,x_2,x_3 x_2,x_3,x_4 x_3,x_4,x_5 ...
R_1 R_2 R_3

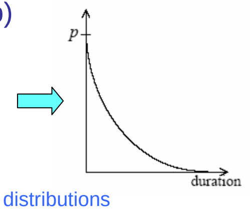
Modeling the Duration of States

- Length distribution of region X:

  $E[l_X] = 1/(1-p)$

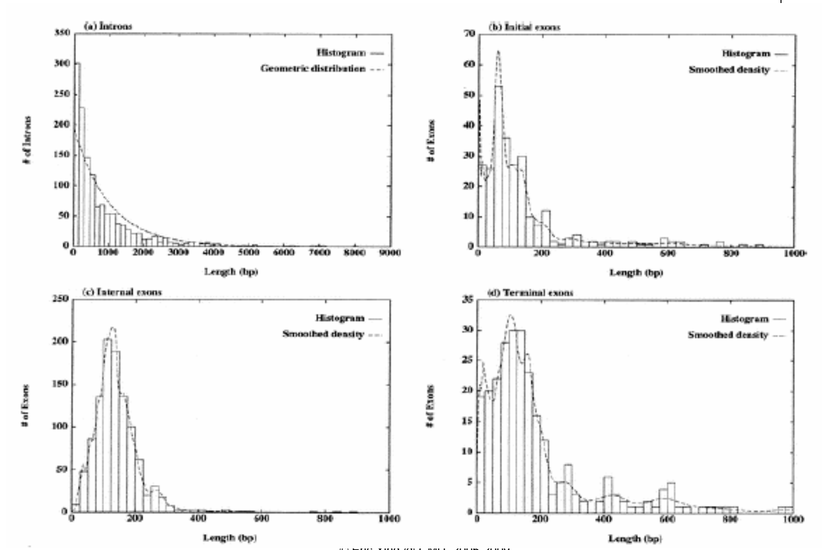


- Geometric distribution, with mean 1/(1-p)
  - (homework: derive this)



  - This is a significant disadvantage of HMMs
  - Several solutions exist for modeling different length distributions

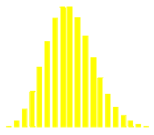
Observed Duration Time



Poisson Point Process

- A counting process that represents the total number of occurrences of discrete events during a temporal/spatial interval
  - the number of occurrences in any internal of length τ is Poisson distributed with parameter $\lambda\tau$:

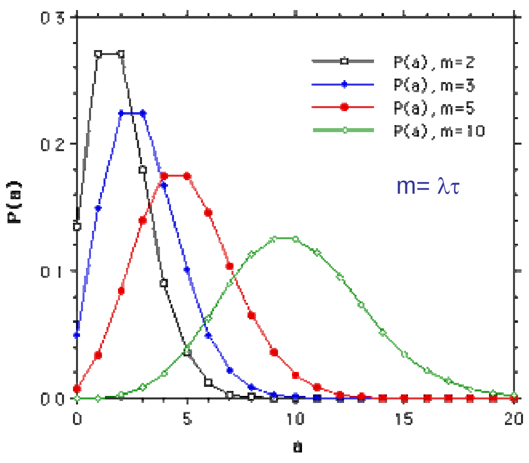
$$p(A(t+\tau) - A(n) = n) = e^{-\lambda\tau}\frac{(\lambda\tau)^n}{n!}$$



  - the number of occurrences in disjoint intervals are independent
  - the duration of the interval between two consecutive occurrences has the following distribution:

$$p(\tau < s) = 1 - e^{-\lambda s}$$

Poisson point process

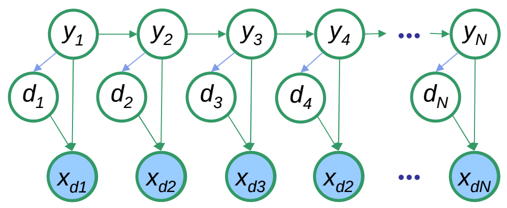


Truncation is needed at both ends!

Generalized HMM

Upon entering a state:

1. Choose duration d, according to probability distribution
2. Generate d letters according to emission probs
3. Take a transition to next state according to transition probs

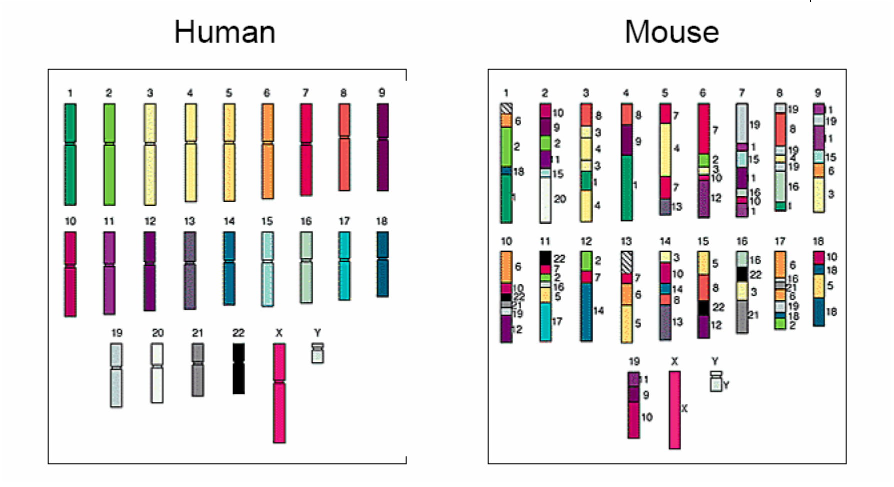

y_1 y_2 y_3 y_4 … y_N
d_1 d_2 d_3 d_4 d_N
x_{d1} x_{d2} x_{d3} x_{d2} … x_{dN}

Disadvantage: Increase in complexity:

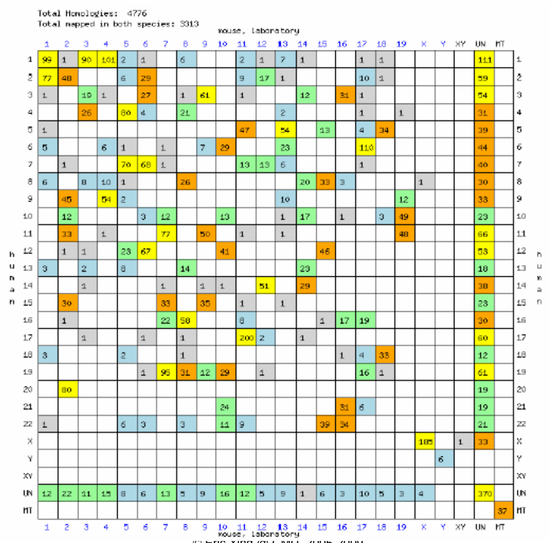
Time: $O(D^2)$
Space: $O(D)$

where D = maximum duration of state

Comparative Genomics



A pairwise comparison between human and mouse genome



Total Homologies: 4776
Total mapped in both species: 3313

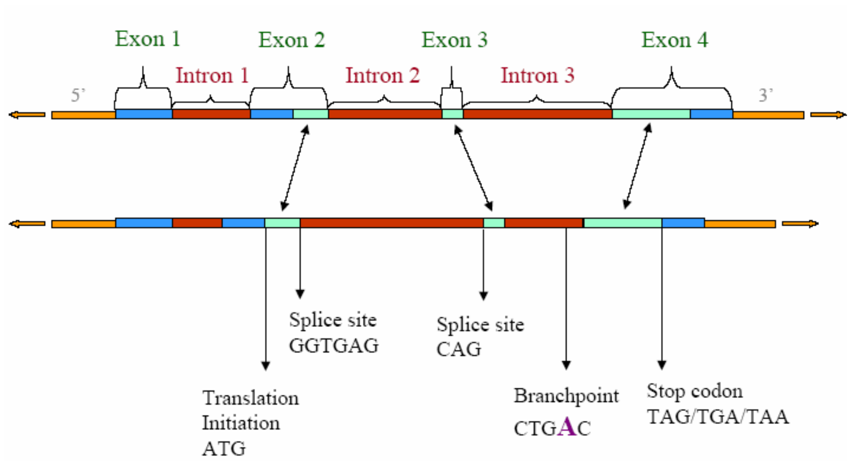
mouse, laboratory

human

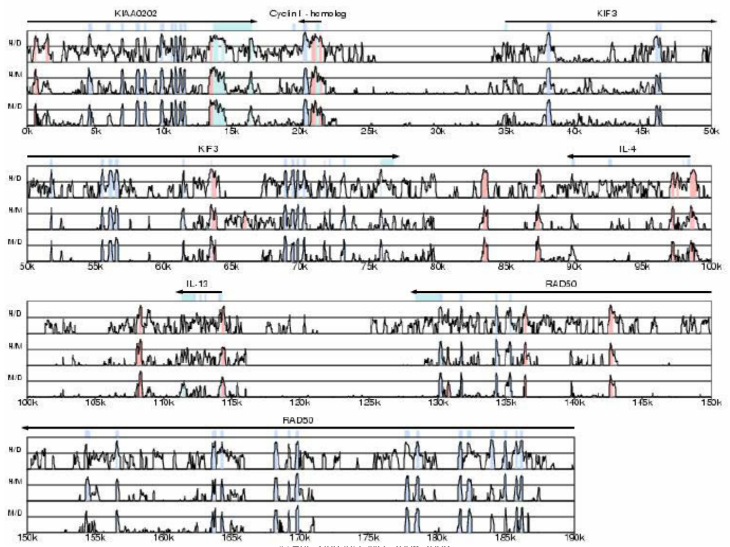
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	X	Y	XY	UN	MT
1	99	1	90	101	2	1		6			2	1	7	1			1	1					111	
2	77	48			6	29					9	17	1				10	1					59	
3	1		19	1		27		1	61		1			12		31	1						54	
4			26		80	4		21					2				1		1				31	
5	1										47		54		13		4	34					39	
6	5			6	1		1		7	29			23				110						44	
7		1			70	68	1				13	13	6				1						40	
8	6		8	10	1			26						20	33	3				1			30	
9		45		54	2								10						12				33	
10		12				3	12			13			1	17		1		3	49				23	
11		33		1			77		50		1		1						48				66	
12		1	1		23	67				41					46								53	
13	3		2		8			14						23									18	
14			1				1		1	1		51		29									38	
15		30					33		35		1		1										23	
16		1					22	58			8				1	17	19						30	
17			1			1		1			200	2		1									60	
18	3				2			1								1	4	33					12	
19						1	95	31	12	29		1					16	1					61	
20		80																					19	
21										24						31	6						19	
22	1				6	3		3		11	9				39	34							21	
X																				185		1	33	
Y																					6			
XY																								
UN	12	22	11	15	8	6	13	5	9	16	12	5	9	1	6	3	10	5	3	4			370	
MT																								37

mouse, laboratory

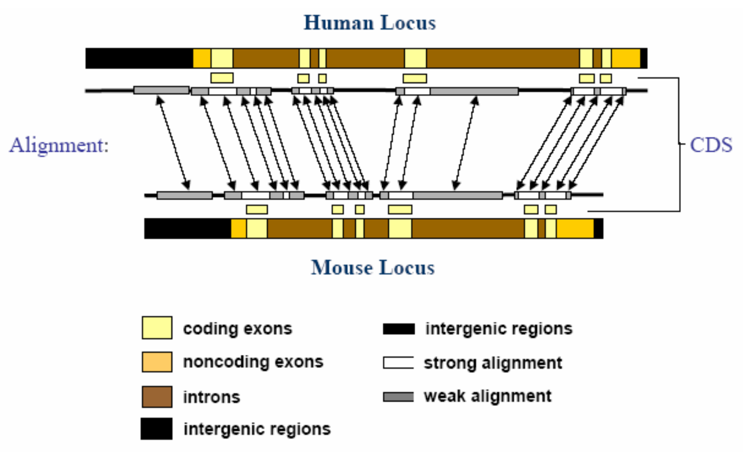
Aligning One Locus



Three Pairwise Alignments

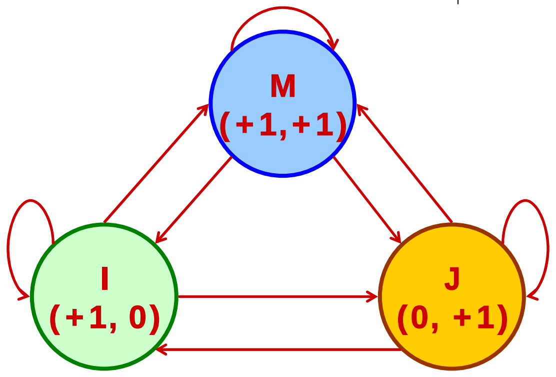


Example: a human/mouse ortholog



Paired HMM

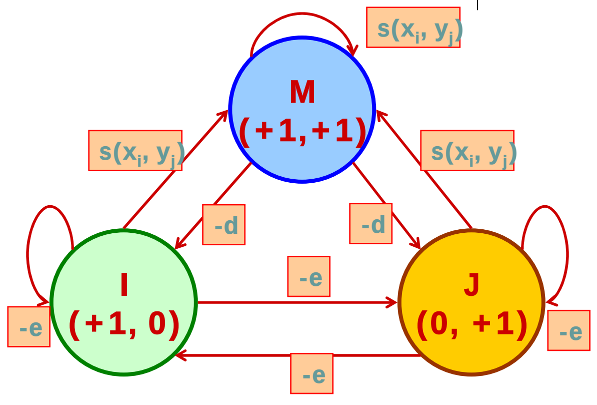
Alignments correspond
1-to-1 with sequences
of states M, I, J



```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC-GGTCGATTTGCCCGACC
IMMJMMMMMMMJJMMMMMMJMMMMMMMIIMMMMMIII
```

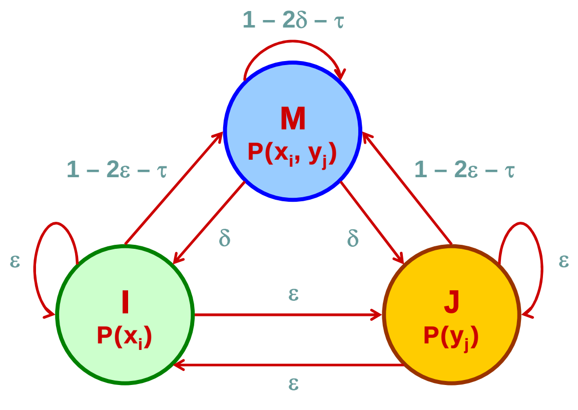
Let’s score the transitions

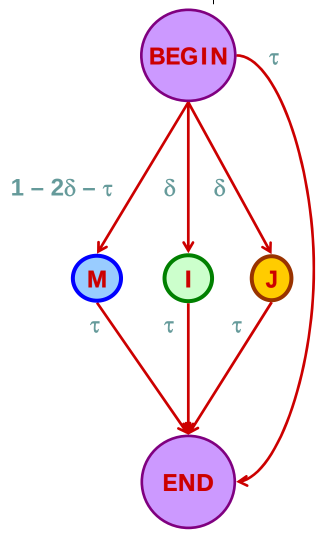
Alignments correspond 1-to-1 with sequences of states M, I, J



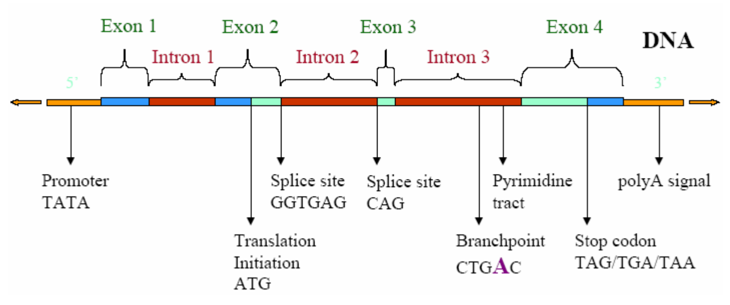
```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC-GGTCGATTTGCCCGACC
IMMJMMMMMMMJJMMMMMMJMMMMMMMIIMMMMMIII
```

A Pair HMM for alignments



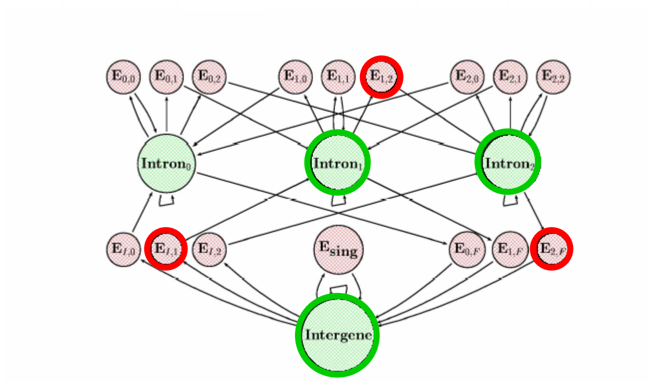


Gene Finding

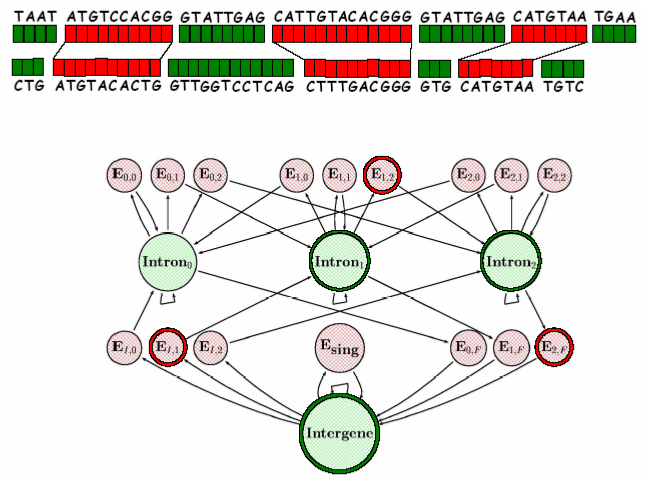


Generalized HMM Gene finder

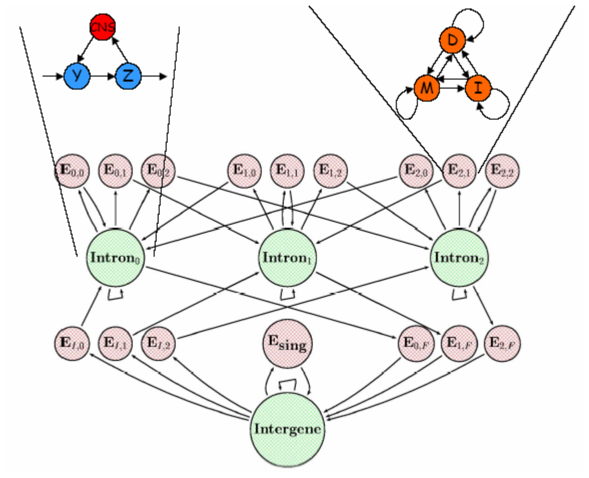
TAAT ATGTCCACGG GTATTGAG CATTGTACACGGG GTATTGAG CATGTAA TGAA



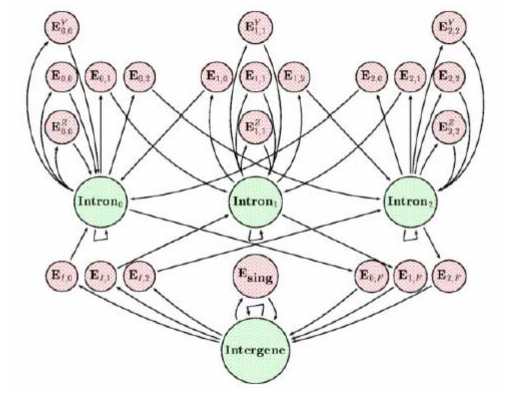
Generalized Pair-HMM gene finder


TAAT ATGTCCACGG GTATTGAG CATTGTACACGGG GTATTGAG CATGTAA TGAA
CTG ATGTACACTG GTTGGTCCTCAG CTTTGACGGG GTG CATGTAA TGTC
$E_{0,0}$ $E_{0,1}$ $E_{0,2}$ $E_{1,0}$ $E_{1,1}$ $E_{1,2}$ $E_{2,0}$ $E_{2,1}$ $E_{2,2}$
Intron_0 Intron_1 Intron_2
$E_{I,0}$ $E_{I,1}$ $E_{I,2}$ E_{sing} $E_{0,F}$ $E_{1,F}$ $E_{2,F}$
Intergene

Hierarchical state transition in pHMM


CNS
Y
Z
D
M
I
$E_{0,0}$ $E_{0,1}$ $E_{0,2}$ $E_{1,0}$ $E_{1,1}$ $E_{1,2}$ $E_{2,0}$ $E_{2,1}$ $E_{2,2}$
Intron_0 Intron_1 Intron_2
$E_{I,0}$ $E_{I,1}$ $E_{I,2}$ E_{sing} $E_{0,F}$ $E_{1,F}$ $E_{2,F}$
Intergene

Allowing for inserted exons



Acknowledgments

- **Serafim Batzoglou**: for some of the slides adapted or modified from his lecture slides at Stanford University
- **Lior Pachter'**: for some of the slides modified from his lectures at UC Berkeley