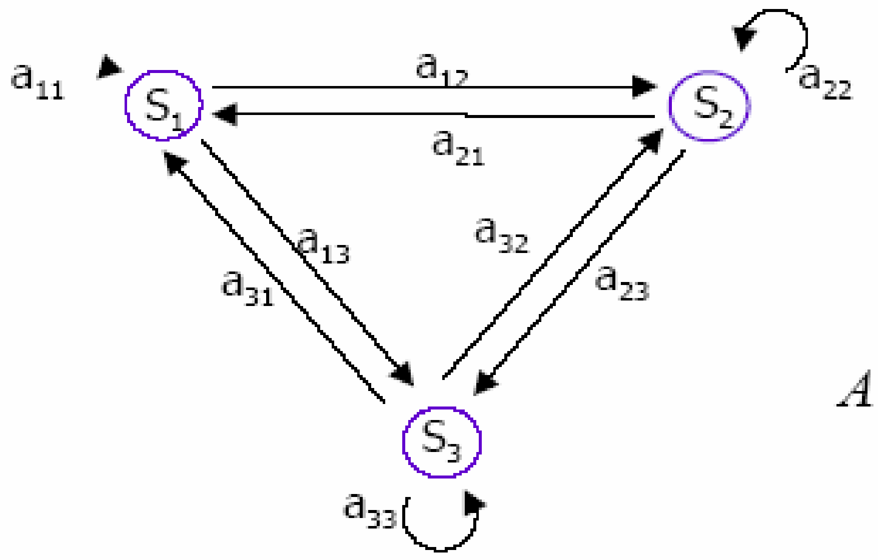
Hidden Markov models

Markov Model

- Collection of states: $\{S_1, S_2, \ldots, S_N\}$



Transition Probability

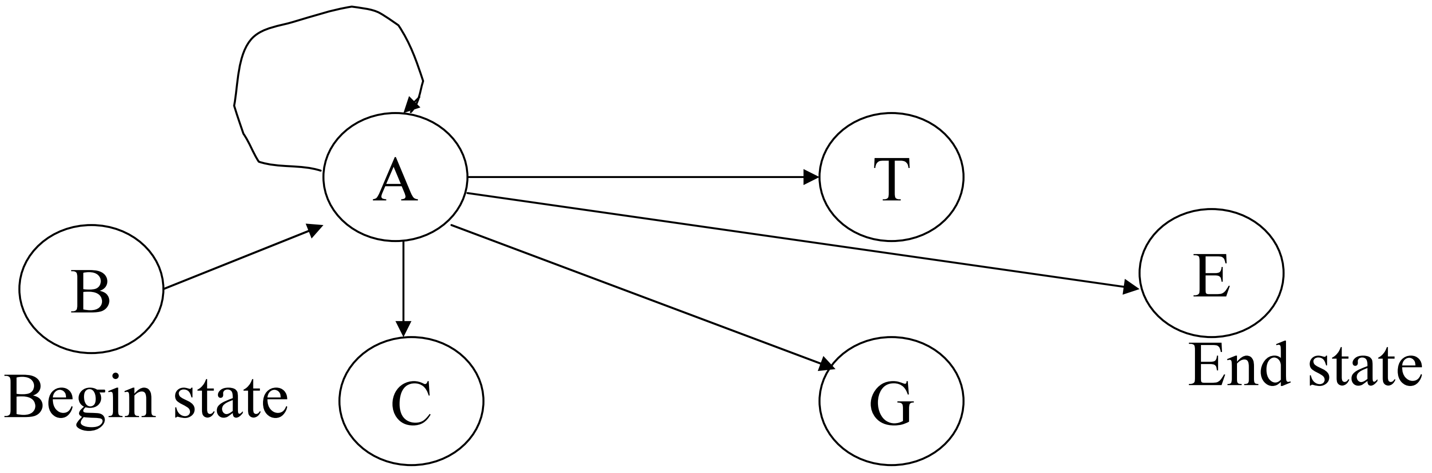
$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Markov Model

- Collection of states: $\{S_1, S_2, \ldots, S_N\}$
- Markov condition: transition probabilities: $A_{ij}^t = P(S_i^{t+1} \mid S_j^t)$
- Initial state
- Equilibrium (stationary) distribution
- Model parameter estimated assuming equilibrium distribution.

Markov Chain

- Models a sequence $S=x_1x_2x_3\ldots x_n$ in which probability of a symbol depends on its previous symbol(s). Only the transitions from and to the state A is shown. Each transistion has an associated probability.



A Markov Chain in DNA alphabet A, T , C and G

Probabilities

$$P(x) = P(x_1 x_2 .....x_n) = P(x_1)\prod_{i=2}^{i=n} P(x_i \mid x_{i-1})$$

$$P(x_1 = s) = P(s \mid B)$$

$$P(E \mid x_n = t) = P(E \mid t)$$

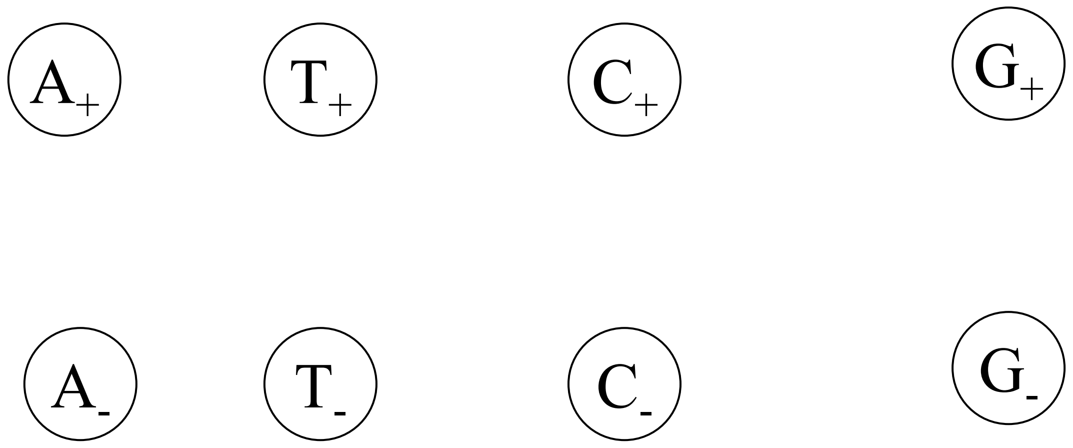
Where both *s* and *t* could be any one of the states A, T, C and G

CpG Island Example

- Question 1
- Given a short sequence of DNA , does it belong to CpG island family
- Question 2
- Given a long sequence, does this contain a CpG island sequence.
- We answered the first question by developing a discriminant model

Hidden Markov Model

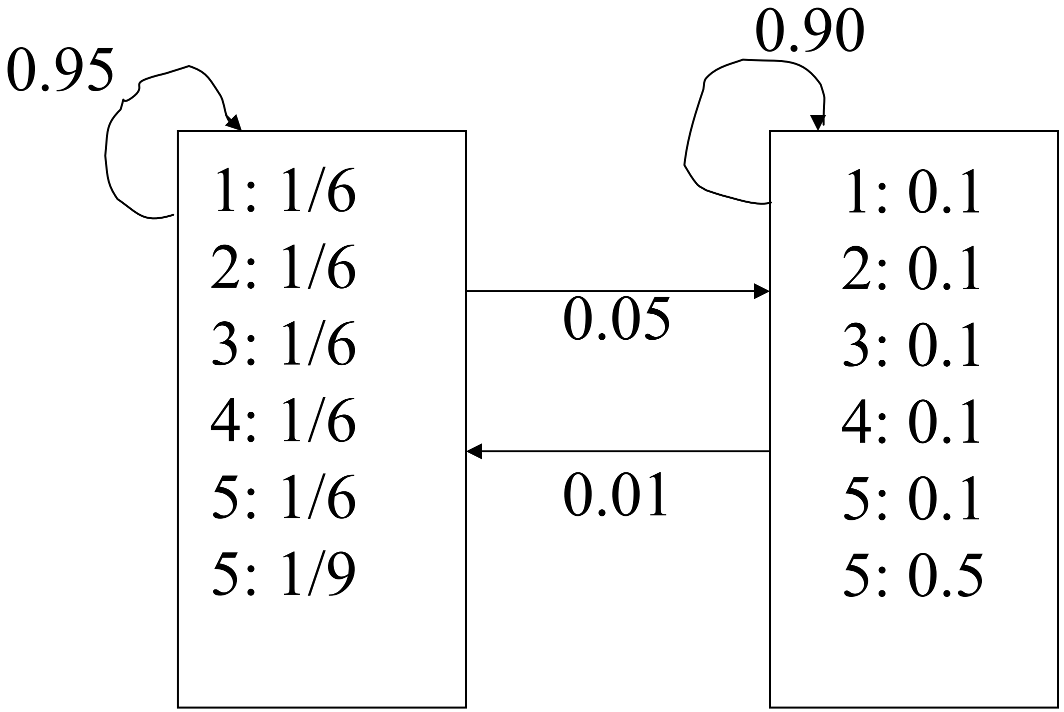
- To answer the second question, we introduced the notion of hidden Markov model (HMM). Transitions from every state to any other state are not shown.



HMM Definitio

- In Markov Chain, there is a one-to-one correspondence between the state and the symbols being generated.
- In HMM, there is no one-to-one correspondence between the state and the symbol. Given an output sequence of states, there is no way to tell what state sequence the HMM traveled.

Occasionally Dishonest Casino



Definition of HMM

- We have to differentiate between a sequence of output symbol generated with the sequence of states in HMM .The sequence of states is called a **path** π
- The i-th state in the path is denoted π_i
- The chain is characterized by the probabilities $P(l/k)= P(\pi_i = l \mid \pi_{i-1} = \mathrm{k})$ and
- $P(k \mid 0)$= The probability that the HMM starts at state k as the begin state 0.

HMM Definition (cont.)

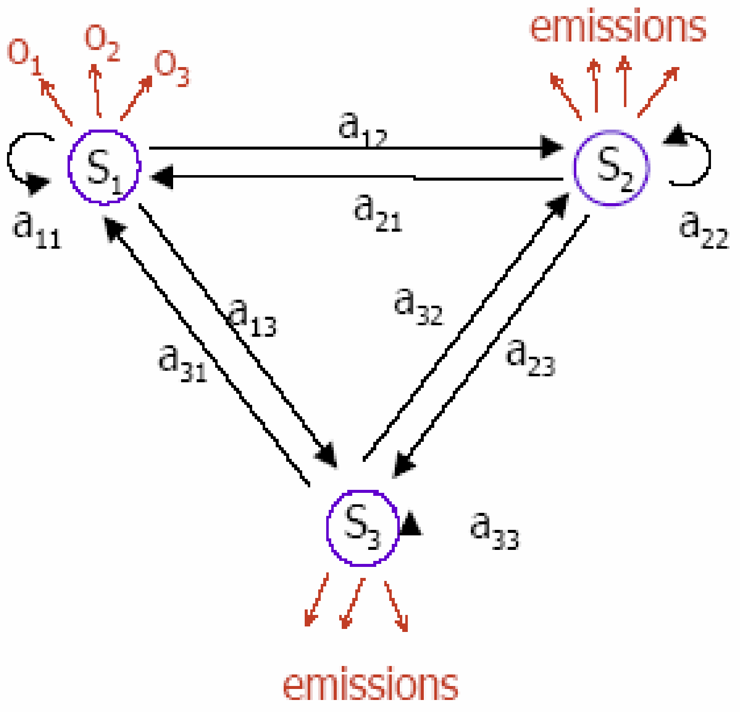
- Since the symbols are decoupled from the states, we need to define what is called

emission probabilities as

$$e_k(b) = P(x_i = b \mid \pi_i = k)$$

HMM = Markov Model + Position Specific Score Matrix

- Markov model: transition between states
- Each state emits an observation:
- Emission probability = PSSM



Position Specific Score Matrix
(Positional Weight Matrix, Profile)

Occurrence frequency of every letter at each position

TACGAT
TATAAT
TATAAT
GATACT
TATGAT
TATGTT

$\rightarrow$

A	0	6	0	3	4	0
C	0	0	1	0	1	0
G	1	0	0	3	0	0
T	5	0	5	0	1	6

$\rightarrow$

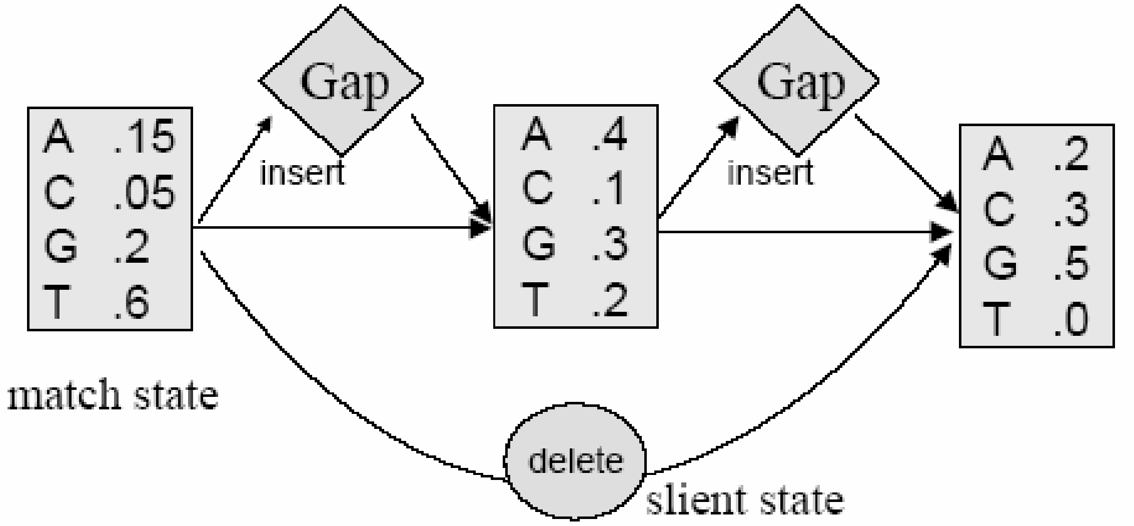
A	0	1	0	.5	.67	0
C	0	0	.16	0	.16	0
G	.16	0	0	.5	0	0
T	.83	0	.83	0	.16	1

Probability distribution at each site (profile)

Adjacent sites assumed independent

HMMs in Sequence Representation

- DNA sequence has insertion/deletion



Probability of an Observed Sequence

- The joint probability of an observed sequence *x* and a state sequence is:

$$P(x,\pi) = P(0 \mid \pi_1)\prod_{i-1}^{n} e_{\pi_1}(x_i)P(\pi_i \mid \pi_{i-1})$$

where we require that $\pi_{n+1} = 0$

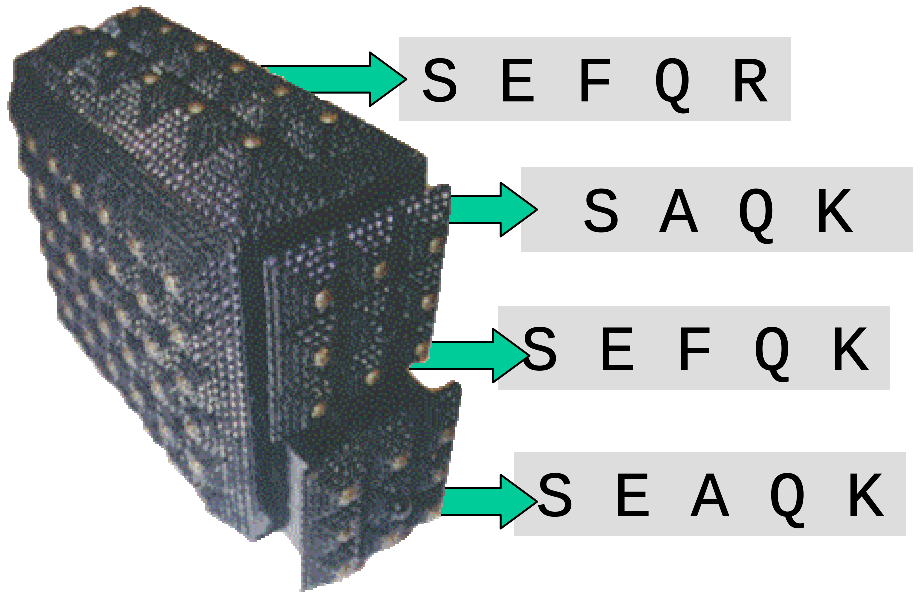
This formula is not very useful unless we know the path.
We could find the most likely path or the path using
a posteriori distribution over states. The algorithm to do that is
called the Viterbi Algorithm.

The Sequence Family Black Box

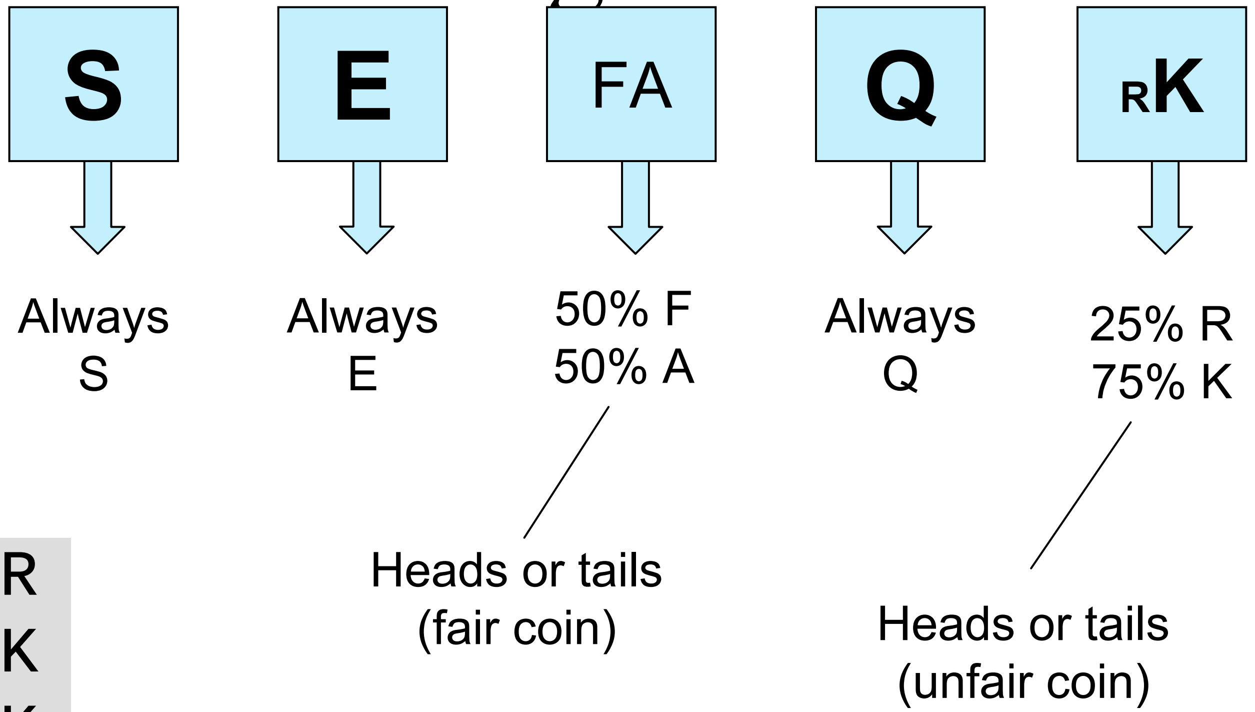
Given an alignment...

S	E	F	Q	R
S	-	A	Q	K
S	E	F	Q	K
S	E	A	Q	K

...design a black box that generates similar sequences

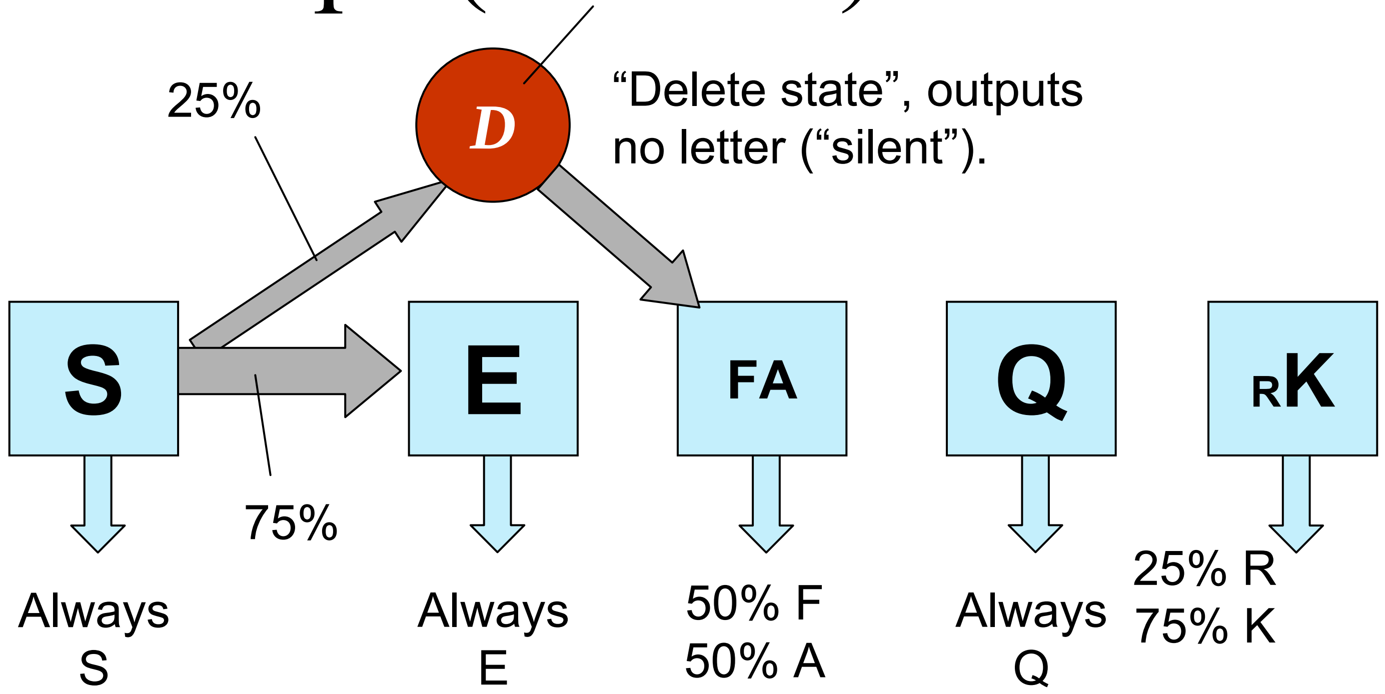


Random generator



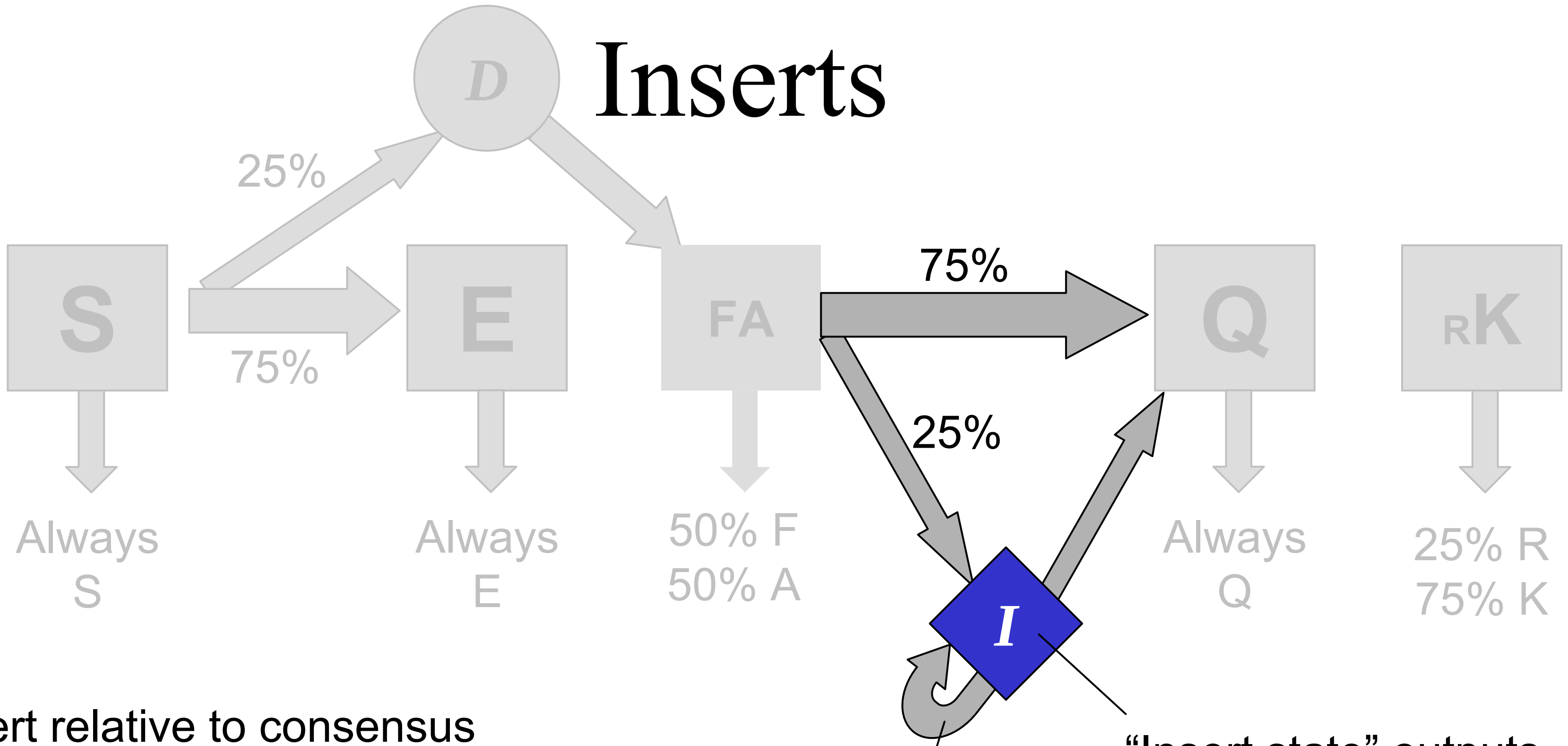
S	E	F	Q	R
S	-	A	Q	K
S	E	F	Q	K
S	E	A	Q	K

Gaps (Deletes)



S	E	F	Q	R
S	-	A	Q	K
S	E	F	Q	K
S	E	A	Q	K

Inserts



“Self-loop” allows inserts of any length

“Insert state” outputs any letter at random

Insert relative to consensus

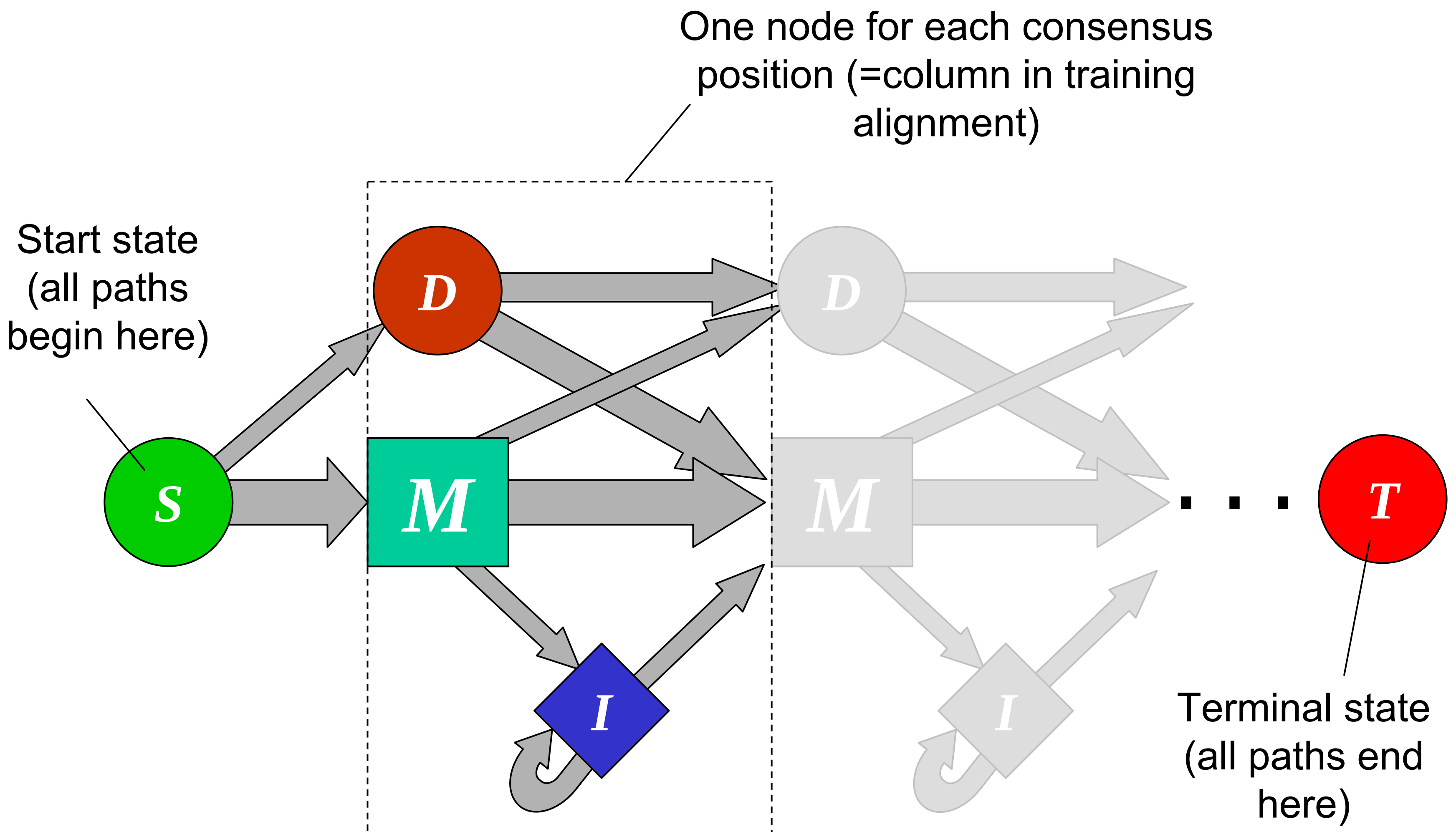
```
S E F - - - Q R
S - A V W I Q K
S E F - - - Q K
S E A - - - Q K
```

Generalize

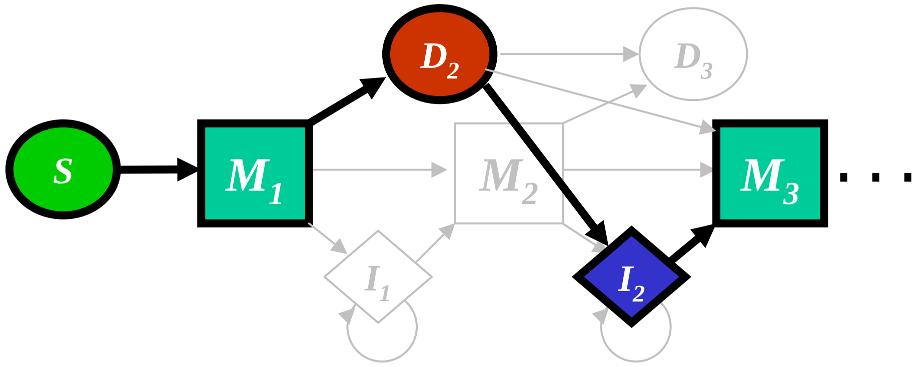
- So far, model is too specific
- Generated sequences all very similar to training set
- Won’t generate more distant homologs
- Generalize: at each position, allow non-zero probability for:

  1. any letter
  2. gap (delete)
  3. inserts

Profile HMM


One node for each consensus position (=column in training alignment)
Start state (all paths begin here)
D
S
M
I
D
M
I
. . .
T
Terminal state (all paths end here)

Profile HMM: graphical model



- Emitter states: M_k and I_k
  - generate one letter on each visit
  - letter emitted following a probability distribution over the alphabet
- Silent state: D_k
- Transitions between states
  - each state has a probability distribution over next state to visit
- Special silent states: Start and Terminal
- Begin in Start state
- Repeat until in Terminal state:
  1. Output letter according to emission distribution (unless silent).
  2. Choose next state according to transition distribution.

Hidden Markov Model

- Set of M states $\{S_i \; i = 1 .. M\}$
- Alphabet $\{a_k, k = 1 .. K\}$ (K=4 for DNA, K=20 for amino acids)
- M x K emission probabilities e_{ik}
  - e_{ik} = probability of emitting letter A_k from state S_i
- M x M transition matrix $t_{ij} = P(S_j \mid S_i)$
  - *n*th order Markov model: probs depend on *n* predecessor states
  - Profile HMM: only 3 transitions for each state, transition matrix is sparse
- Model normalization, sum of probabilities = 1
  - For all states, emissions sum to one and transitions sum to one

    $\sum_k e_{ik} = 1, \forall i$

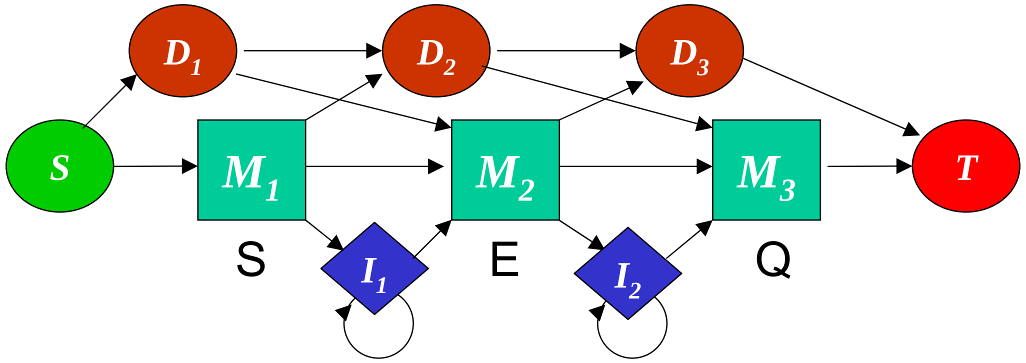
    $\sum_j t_{ij} = 1, \forall i$
  - Special case: silent state S_i, $e_{ik} = 0$ for all letters k.

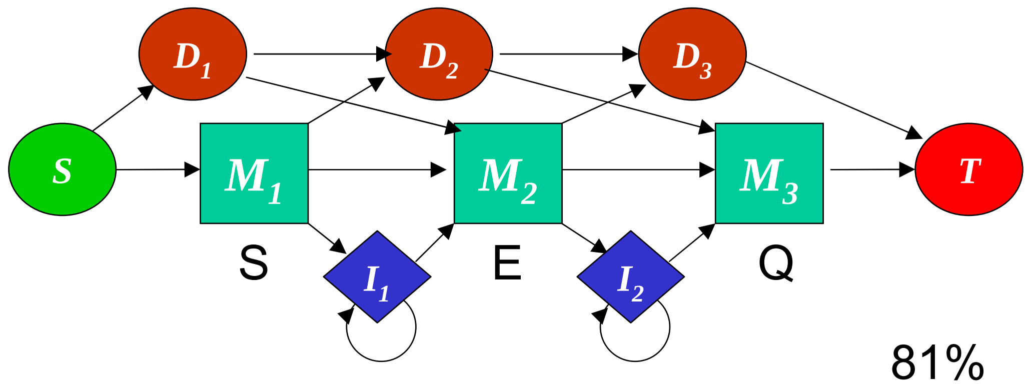
“Profile” HMM

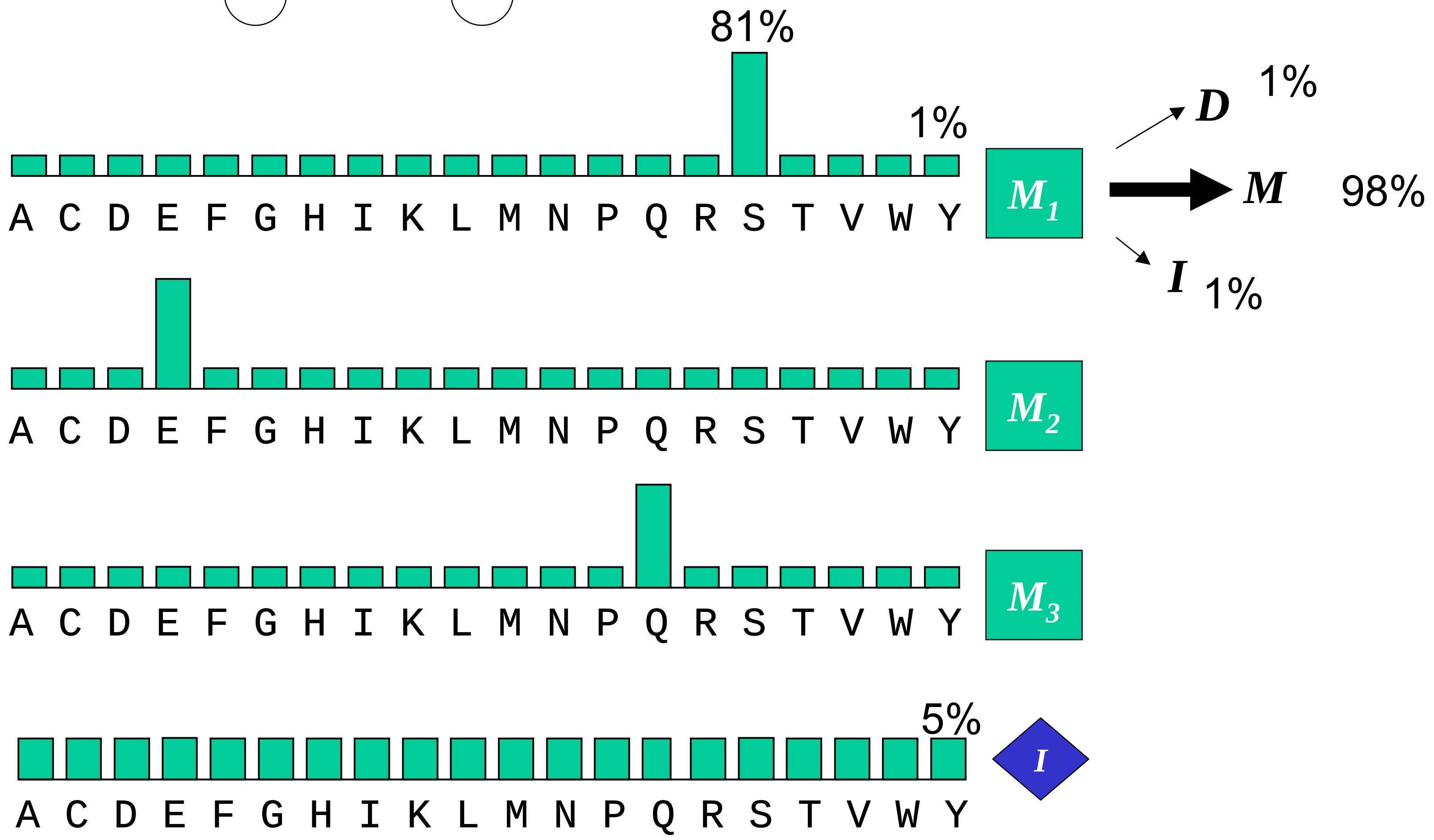
- Profile = statistical model of sequence family
- Other types of HMMs, e.g. genefinders, are not profile HMMs
- From now on, HMM = profile HMM

Given sequence & HMM, path "hidden"

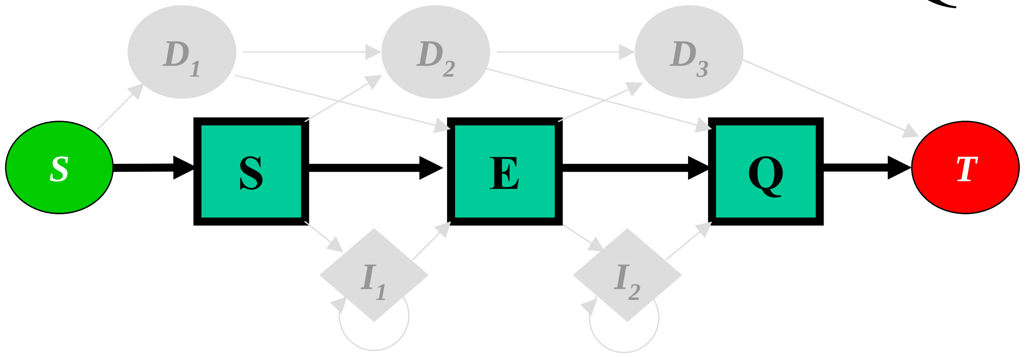
- Any (generalized) HMM can generate all possible sequences
- Many ways to generate given sequence from given HMM
- Example: model of motif "SEQ"



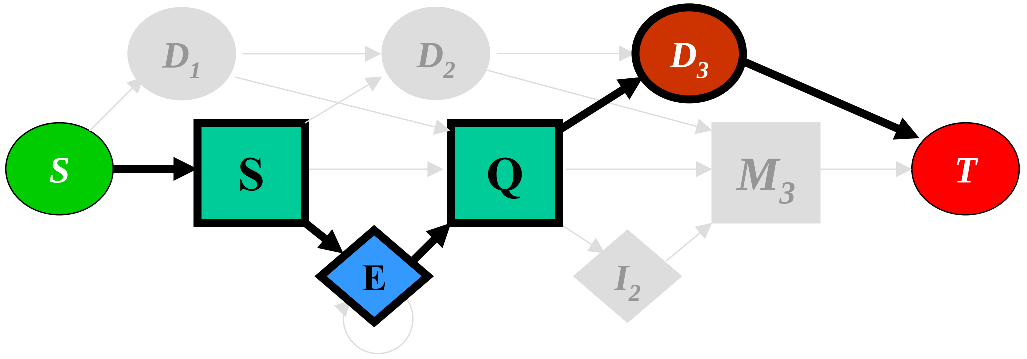

D_1
D_2
D_3
S
M_1
M_2
M_3
T
I_1
I_2
S
E
Q


81%
1%
A C D E F G H I K L M N P Q R S T V W Y
M_1
D 1%
M 98%
I 1%
M_2
M_3
5%
I

More than one way to generate "SEQ"



P = 50.02%



P = 0.0002%

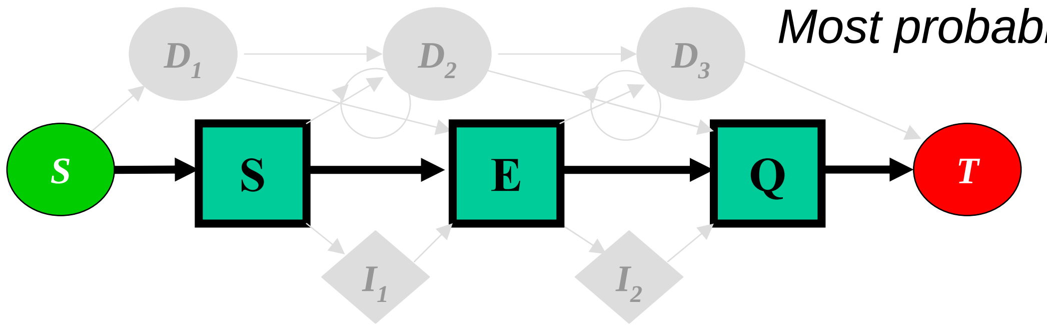
“Hidden” Markov Model

- Given string and model, path cannot be determined
- May different paths may generate same string
- Path is “hidden”
- Any (generalized) model can generate all possible strings
- Proof: consider a path like this:

  S→D→D→I→I→I ... (visit Insert state once for each letter) ... →I→D→D→T

HMMs for alignment

- Find most probable path that generates the string
- Path equivalent to assigning each letter to an emitter state
- Letter assigned to match state is aligned to consensus position (column) in training alignment



Most probable path

Alignment:

$M_1 \leftrightarrow S$
$M_2 \leftrightarrow E$
$M_3 \leftrightarrow Q$

P(sequence | HMM)

- Longer paths always less probable...
- ...each transition & emission multiplies by probability < 1
- More general model tends to give lower probability

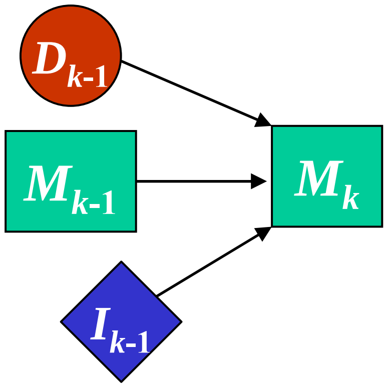
Viterbi algorithm

- Finds a most probable path (MPP) through HMM given a sequence
- There may be more than one MPP
- Dynamic programming algorithm
- Closely related to standard pair-wise alignment with affine gap penalties
  - HMM has position-specific gap penalties
  - Typically fixed in other methods, though ClustalW has position-specific heuristics
  - Gap penalties correspond to transition scores
  - $M \rightarrow D$ or $M \rightarrow I$ = gap open
  - $D \rightarrow D$ or $I \rightarrow I$ = gap extend

Key definition

- V(i, Q) = Probability of a most probable sub-path (MPSP)
- that (a) emits the first *i* letters of the sequence, and
  (b) ends in state Q
- Informally, this is probability of best match of prefix of model to prefix of sequence
- Recursively compute these probabilities (dynamic programming)
- Reduces complexity to O(ML) time and space
  - M=model length, L=sequence length
- This assumes fixed number of transitions into each state
  - 3 for a profile HMM
- For general first-order HMM with K states, is order $O(K^2L)$

Recursion for Match state M_k



MPSP(i, Q) = most probable sub-path that (a) emits first i letters in sequence S and (b) ends in state Q.

V(i, Q) = probability of MPSP(i, Q)

Three possibilities for MPSP(i, M_k):

$\text{MPSP}(i-1, M_{k-1}) + M_{k-1} \rightarrow M_k$,

$\text{MPSP}(i-1, I_{k-1}) + I_{k-1} \rightarrow M_k$, or

$\text{MPSP}(i, D_{k-1}) + D_{k-1} \rightarrow M_k$

Hence:

$$V(i, M_k) = \max \{ \\ V(i-1, M_{k-1})\, P(M_{k-1} \rightarrow M_k) \\ V(i-1, I_{k-1})\, P(I_{k-1} \rightarrow M_k) \\ V(i, I_{k-1})\, P(D_{k-1} \rightarrow M_k) \}$$

$V(i, M_k)$

Probability of an edge $P(Q \rightarrow R)$ is transition probability x emission probability (unless silent).

Define:

$t(Q \rightarrow R)$ = transition probability $P(R \mid Q)$

$e(Q, a)$ = emission probability of letter a in state Q.

Then:

$P(M_{k-1} \rightarrow M_k) = t(M_{k-1} \rightarrow M_k)\, e(M_k, S_i)$

$P(I_{k-1} \rightarrow M_k) = t(M_{k-1} \rightarrow M_k)\, e(I_k, S_i)$

$P(D_{k-1} \rightarrow M_k) = t(D_{k-1} \rightarrow M_k)$

Finally: $V(i, M_k) = \max \{$

$V(i - 1, M_{k-1})\, t(M_{k-1} \rightarrow M_k)\, e(M_k, S_i)$

May be two or three that have max value, so may be > 1 overall MPP.

$V(i - 1, I_{k-1})\, t(M_{k-1} \rightarrow M_k)\, e(I_k, S_i)$

$V(i , I_{k-1})\, t(D_{k-1} \rightarrow M_k) \quad \}$

General case V(i, Q)

In general:

$V(i, Q) = \max R$ *(R ranges over all states in HMM)*

```
        {
        V(i – 1, R) t(R→Q) e(Q, S_i)  (if R is emitter state)

        V(i, R) t(R→Q) (if R is silent state)
        }
```

Probability of MPP = V(L, T) *(L=length of sequence, T=terminal state)*.

Edges of the MPP can be found by storing max case for each i,Q, or by trace-back.

Note that in a profile HMM, $t(R \rightarrow Q)$ is zero for most Q,R pairs, this is exploited to make a more efficient implementation.

Forward / backward algorithm

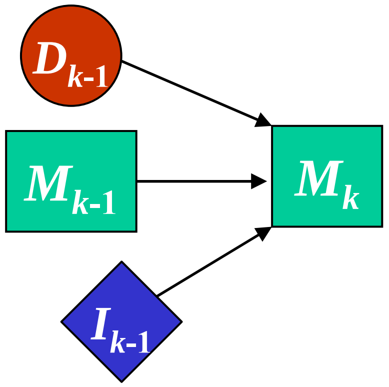
- **Computes probability that sequence is generated by the HMM**

  P(sequence | HMM)
- **Considers all ways the sequence may be generated, not just the most probable (as in Viterbi)**
- **Computes probability that a given position in the sequence output by a given emitter state**

  P(i ↔ Q | sequence, HMM)

  (↔ means "aligned to" or "emitted by")
- **Used to construct a "posterior decoding" alignment**
  - **Allegedly more accurate than Viterbi**
  - **See**

Forward recursion for Match state M_k



F(i, Q) = Probability that a sub-path (a) emits first i letters in sequence S and (b) ends in state Q.

= Sum of probability over all sub-paths that satisfy (a) and (b)

Three possibilities for final edge.

Hence:

sum vs. max in Viterbi

$$F(i, M_k) = F(i-1, M_{k-1})\, P(M_{k-1}\rightarrow M_k) +$$
$$F(i-1, I_{k-1})\, P(I_{k-1}\rightarrow M_k) +$$
$$F(i, I_{k-1})\, P(D_{k-1}\rightarrow M_k)$$

General case F(i, Q)

In general:

F(i, Q) = ∑ R *(R ranges over all states in HMM)*

{

F(i – 1, R) t(R→Q) e(Q, S_i) *(if R is emitter state)*

F(i, R) t(R→Q) *(if R is silent state)*

}

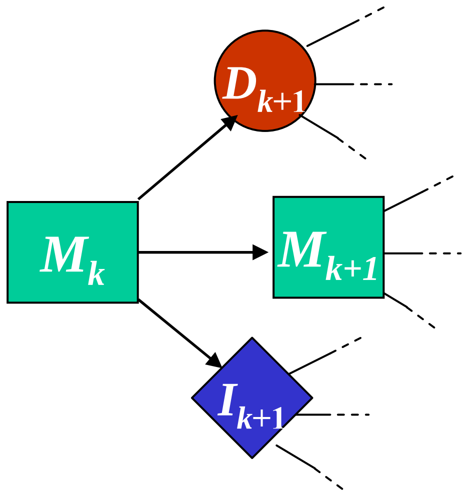
P(sequence | HMM) = F(L, T) *(L=length of sequence, T=terminal state).*

Note that in a profile HMM, t(R→Q) is zero for most Q,R pairs, this is exploited to make a more efficient implementation.

Backward algorithm

- B(i, Q) = Probability that a sub-path Q ---> End (a) emits LAST L – i letters in sequence S, given that (b) sub-path up to state Q emitted FIRST i letters.
- Informally, is probability that SUFFIX of model matches SUFFIX of sequence.

Backward recursion for Match state M_k



B(i, Q) = Probability that a sub-path Q ---> End
(a) emits LAST L – i letters in sequence S given that
(b) sub-path up to state Q emitted FIRST i letters.

= Sum of probability over all sub-paths that satisfy (a) and (b)

Three ways to get from M_k to the End state.

$B(i, M_k) = P(M_k \rightarrow M_{k+1})\, B(i + 1, M_{k+1}) +$

$P(M_k \rightarrow I_{k+1})\, B(i + 1, I_{k+1}) +$

$P(M_{k-1} \rightarrow D_{k+1})\, B(i + 1, I_{k+1})$

General case B(i, Q)

If Q is an emitter state:

B(i, Q) = ∑ R *(R ranges over all states in HMM)*

```
        {
        t(Q→R) e(Q, $S_i$) B(i + 1, R)
        }
```

If Q is a silent state:

B(i, Q) = ∑ R *(R ranges over all states in HMM)*

```
        {
        t(Q→R) B(i, R)
        }
```

P(sequence | HMM) = B(0,S) = F(L,T) ***(S=Start, T=Terminal, L=seq length)***

$P(i \leftrightarrow Q \mid \text{sequence, HMM})$

- Probability that position i in sequence is emitted by state Q

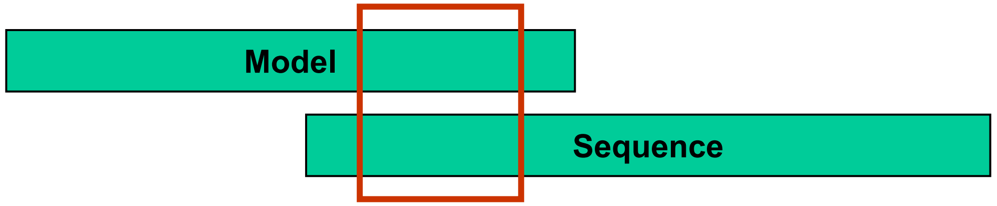
$P(i \leftrightarrow Q \mid \text{sequence, HMM})$

= (probability any sub-path reaches Q and emits up to i) x

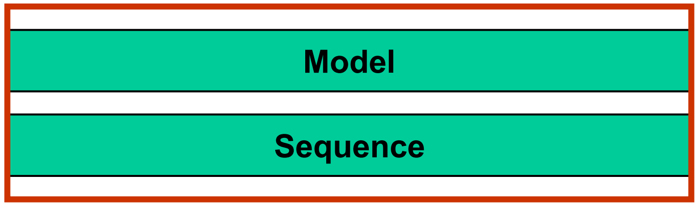
(probability any sub-path starts at Q and emits rest)

Alignment “styles” (boundary conds.)

- Local or global to model or sequence

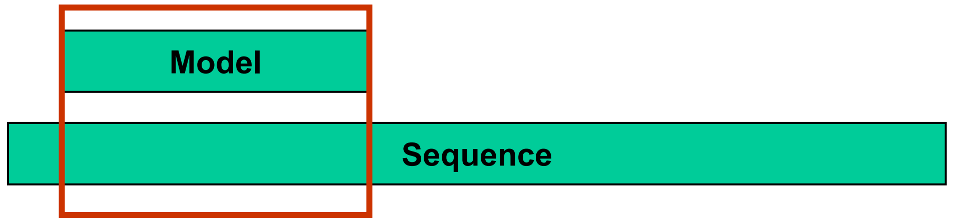


Local-local
(like BLAST)



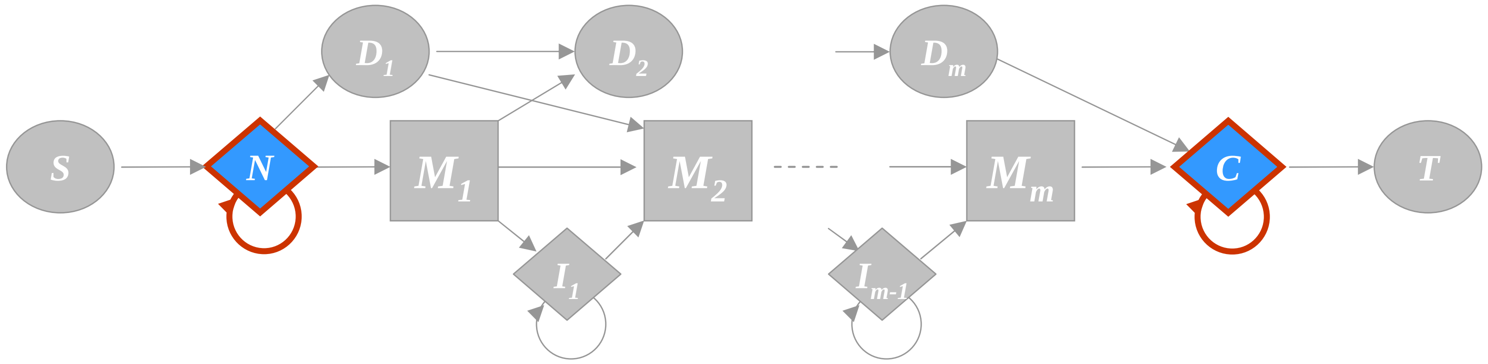
Global-global
(like ClustalW)

Semi-global



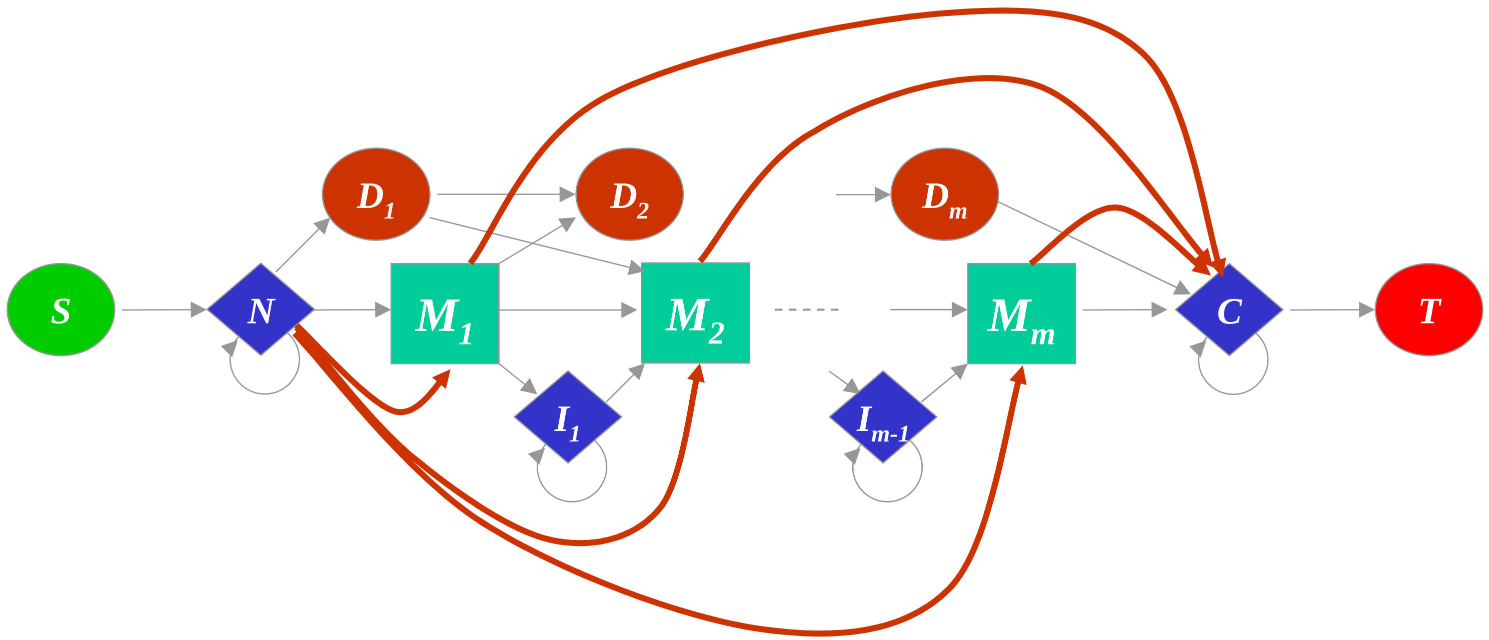
- Global to model, local to sequence (“glocal”)
- Typically used for finding domains or motifs, e.g. PFAM
- Global-global more appropriate for modeling whole proteins

Local to sequence



- Add N and C terminal insert states
- Emit zero or more letters before / after main model
- Special rule: N and C emit only on self-loop, not on first visit

Local to model



- Add “entry” and “exit” transitions
- Alignment can begin and end at any match state

HMM software packages

- HMMER ("Hammer")
  - Sean Eddy, UWash St. Louis
- SAM (Sequence Analysis and Modeling)
  - UC Santa Cruz

HMMER

- Free download
- Source code provided (“C” language)
- Runs on Linux, Unix, Windows, Mac, Sun
- Nice manual
- Relatively easy to install and use
- Most widely used in the community
- http://hmmer.wustl.edu/

SAM

- License required
- No source code available
- Harder to use -- more parameters, not well explained
- Includes more algorithms and parameters than HMMER
  - *buildmodel*
  - posterior decoding alignments
  - SAM-Txx homolog recognition & alignment (like PSI-BLAST, but better)
  - Txx probably best in class

Implementation issues

- Underflow
  - Probability 0.1
  - Model length 100
  - $0.1^{100} = 10^{-100}$, underflows floating point on many CPUs
- min *float* in Microsoft C = 10^{-39}
- Solution: convert to $\log_2$
- Multiplying probabilities becomes adding log-probabilities
- HMMER uses $\lfloor 1000 \log_2 P/P_{NULL} \rfloor$
  - Minus infinity = -100000
- Because integer arithmetic faster
  - But not much faster these days, probably not worth it today
- But risks rounding error, integer under / overflow

Whole-genome alignment

- Sequence length very large
- Cannot use $O(L^2)$ algorithms
- Solution: use fast methods to find “seeds”
  - also called “anchors”
- Extend seeds by dynamic programming
- (optional) combine local alignments into global alignment or synteny graph

Whole-genome alignment

- MUMMER
  - Delcher, A.L., Phillippy, A., Carlton, J. and Salzberg, S.L. (2002) Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res* **30**(11): 2478-83.
- AVID and MAVID
  - Bray, N., Dubchak, I. and Pachter, L. (2003) AVID: A global alignment program. *Genome Res* **13**(1): 97-102.
  - Bray, N. and Pachter, L. (2004) MAVID: Constrained Ancestral Alignment of Multiple Sequences. *Genome Res* **14**(4): 693-9.
- LAGAN and Multi-LAGAN
  - Brudno, M., Do, C.B., Cooper, G.M., Kim, M.F., Davydov, E., Green, E.D., Sidow, A. and Batzoglou, S. (2003) LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res* **13**(4): 721-31.

Textbooks

- Introduction to computational molecular biology, Setubal, J. and Meidanis, J.
  - Introduction to biological sequences and fundamental sequence analysis algorithms, many of which are based on dynamic programming. Gives pseudo-code for many algorithms. Probably the most accessible textbook for programmers who are not experts in computer science or biology.
- Biological sequence analysis, Durbin, R., Eddy, S., Krogh, A., Mitchison, G.
  - Graduate text. Emphasizes probabilistic models, especially Bayesian methods and graphical models (e.g., profile HMMs). Skimpy on biological background, motivation and limitations of their algorithmic approaches, and assumes strong math skills.
- Algorithms on strings, trees and sequences, Gusfield, D.
  - Graduate / advanced undergraduate text. Not much on trees. Very much a computer science perspective, again skimpy on the biology. Comprehensive coverage of dynamic programming algorithms on sequences; also other approaches such as suffix trees.