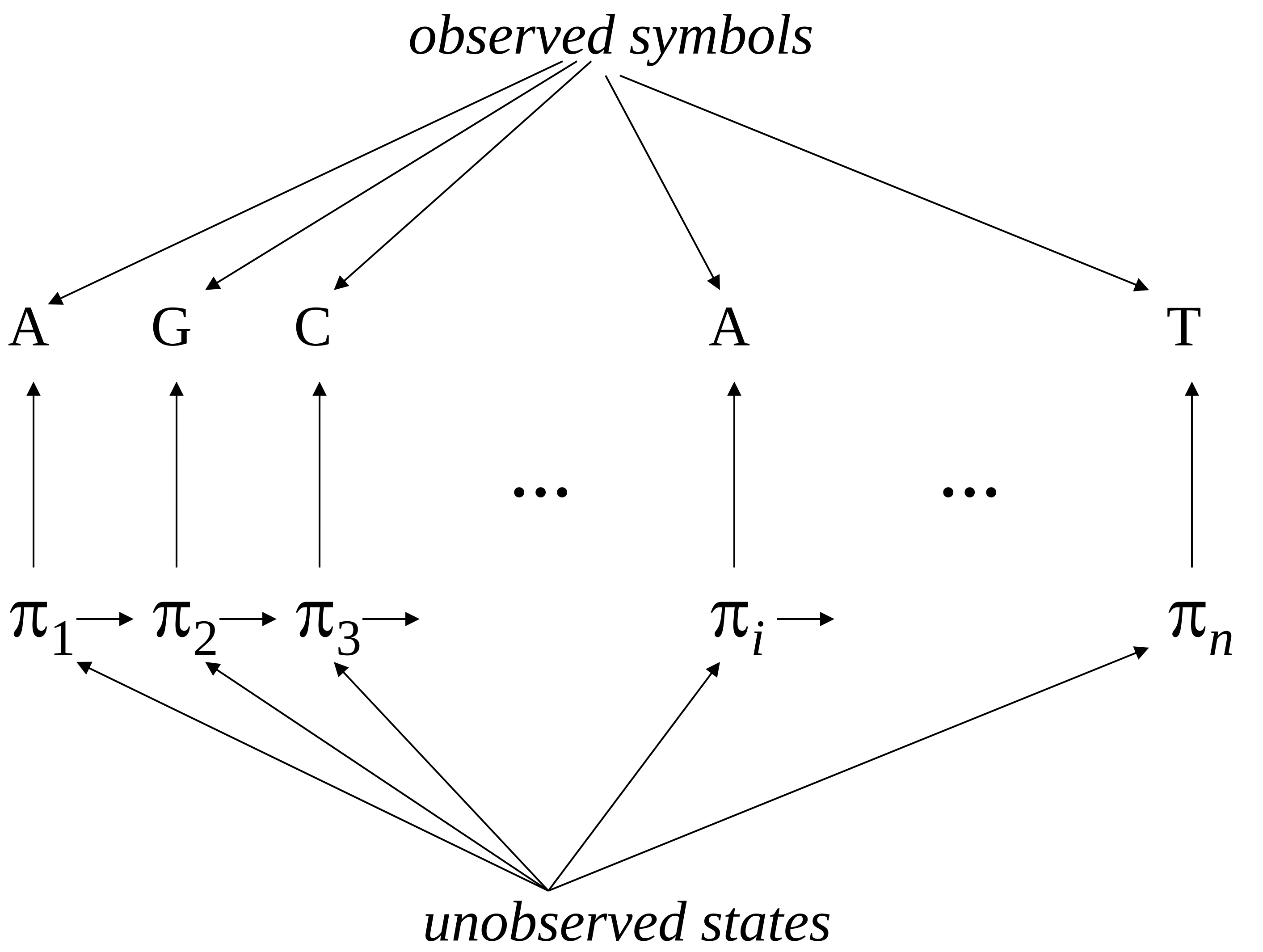
Today's Lecture: HMMs

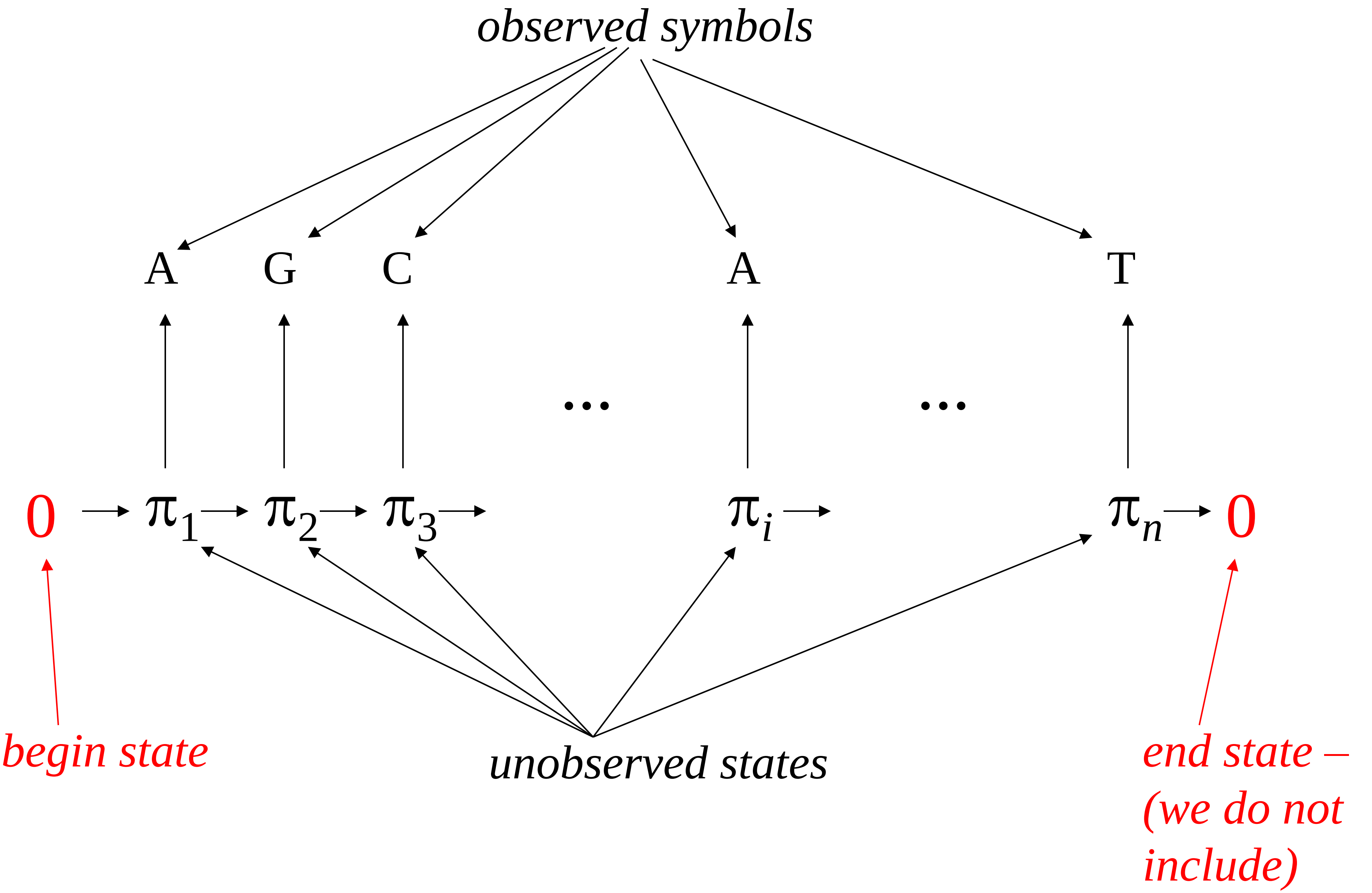
- Definitions
- Examples
- Probability calculations
  - WDAG
  - Dynamic programming algorithms:
    - Forward
    - Viterbi
- Parameter estimation
  - Viterbi training

Hidden Markov Models

- Probability models for sequences of *observed symbols*, e.g.
  - nucleotide or amino acid residues
  - aligned pairs of residues
  - aligned set of residues corresponding to leaves of an underlying evolutionary tree
  - angles in protein chain (structure modelling)
  - sounds (speech recognition)

- Assume a sequence of “*hidden*” (unobserved) *states* underlies each observed symbol sequence
- Each state “*emits*” symbols (one symbol at a time)
- States may correspond to underlying “reality” we are trying to infer, e.g.
  - unobserved biological feature:
    - (positions within) site,
    - coding region of gene
  - rate of evolution
  - protein structural element
  - speech phoneme


observed symbols
A
G
C
...
A
...
T
π_1
π_2
π_3
π_i
π_n
unobserved states


observed symbols
A
G
C
A
T
...
...
0
π_1
π_2
π_3
π_i
π_n
0
begin state
unobserved states
end state –
(we do not
include)

Advantages of HMMs

- Flexible –gives reasonably good models in wide variety of situations
- Computationally efficient
- Often interpretable:
  - hidden states can correspond to biological features.
  - can find most probable sequence of hidden states

    = biological "parsing" of residue sequence.

HMMs: Formal Definition

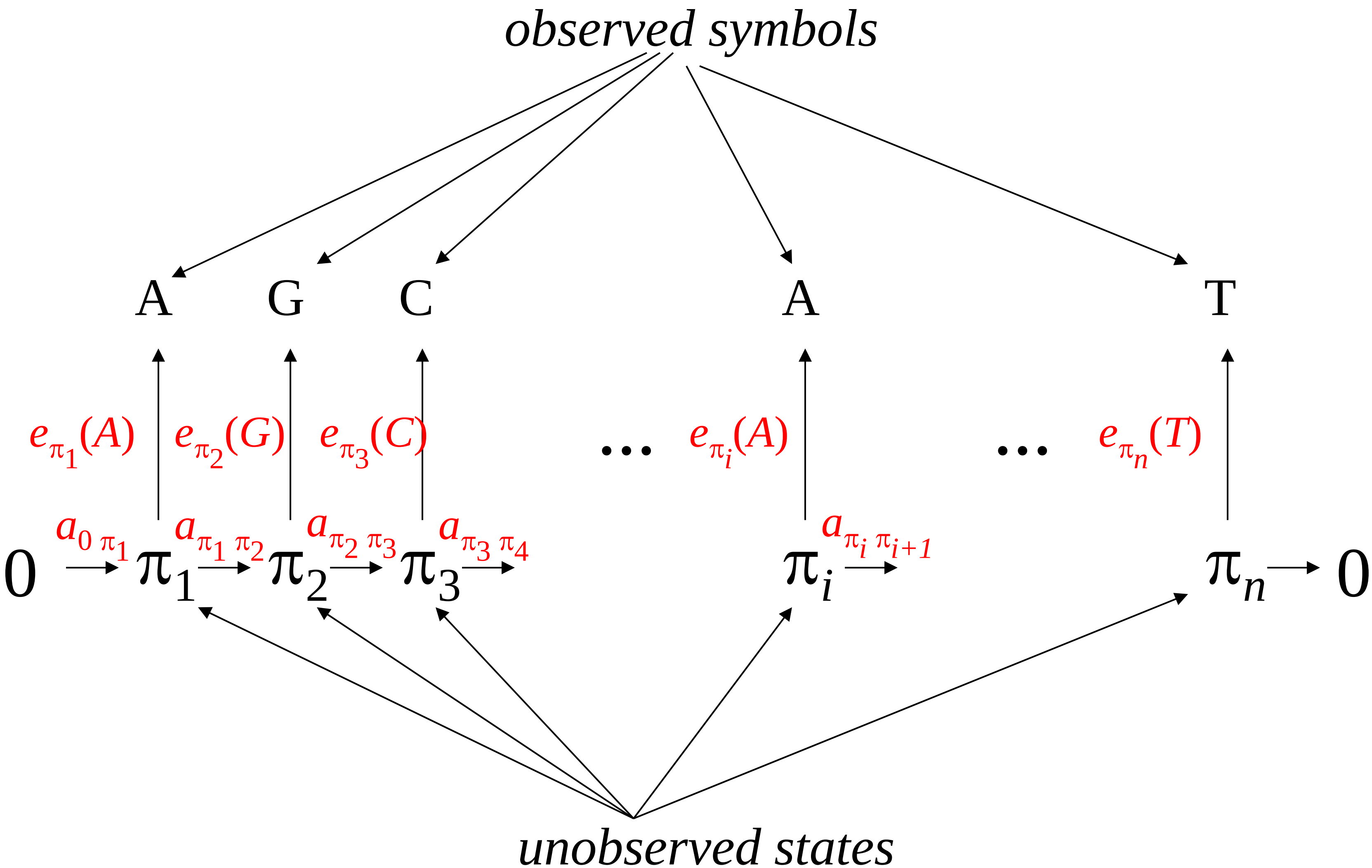
- Alphabet B = {b} of *observed symbols*
- Set S = {k} of *hidden states* (usually k = 0,1, 2 ...,m; 0 is reserved for "begin" state, and sometimes also an "end" state)
- (Markov chain property): prob of state occurring at given position depends only on immediately preceding state, and is given by

  transition probabilities (a_{kl}): a_{kl} = Prob(next state is l | curr state is k)

  $\sum_l a_{kl} = 1$, for each k.

  - Usually, many transition probabilities are set to 0.
  - Model *topology* is the # of states, and *allowed* (i.e. $a_{kl} \neq 0$) transitions.

  Sometimes omit begin state, in which case need *initiation probabilities* (p_k) for sequence starting in a given state



observed symbols
A
G
C
A
T
$e_{\pi_1}(A)$
$e_{\pi_2}(G)$
$e_{\pi_3}(C)$
...
$e_{\pi_i}(A)$
...
$e_{\pi_n}(T)$
$a_{0\,\pi_1}$
$a_{\pi_1\,\pi_2}$
$a_{\pi_2\,\pi_3}$
$a_{\pi_3\,\pi_4}$
$a_{\pi_i\,\pi_{i+1}}$
0
π_1
π_2
π_3
π_i
π_n
0
unobserved states

- Prob that symbol occurs at given sequence position depends only on hidden state at that position, and is given by

  emission probabilities:

  $e_k(b)$ = Prob(observed symbol is b | curr state is k)

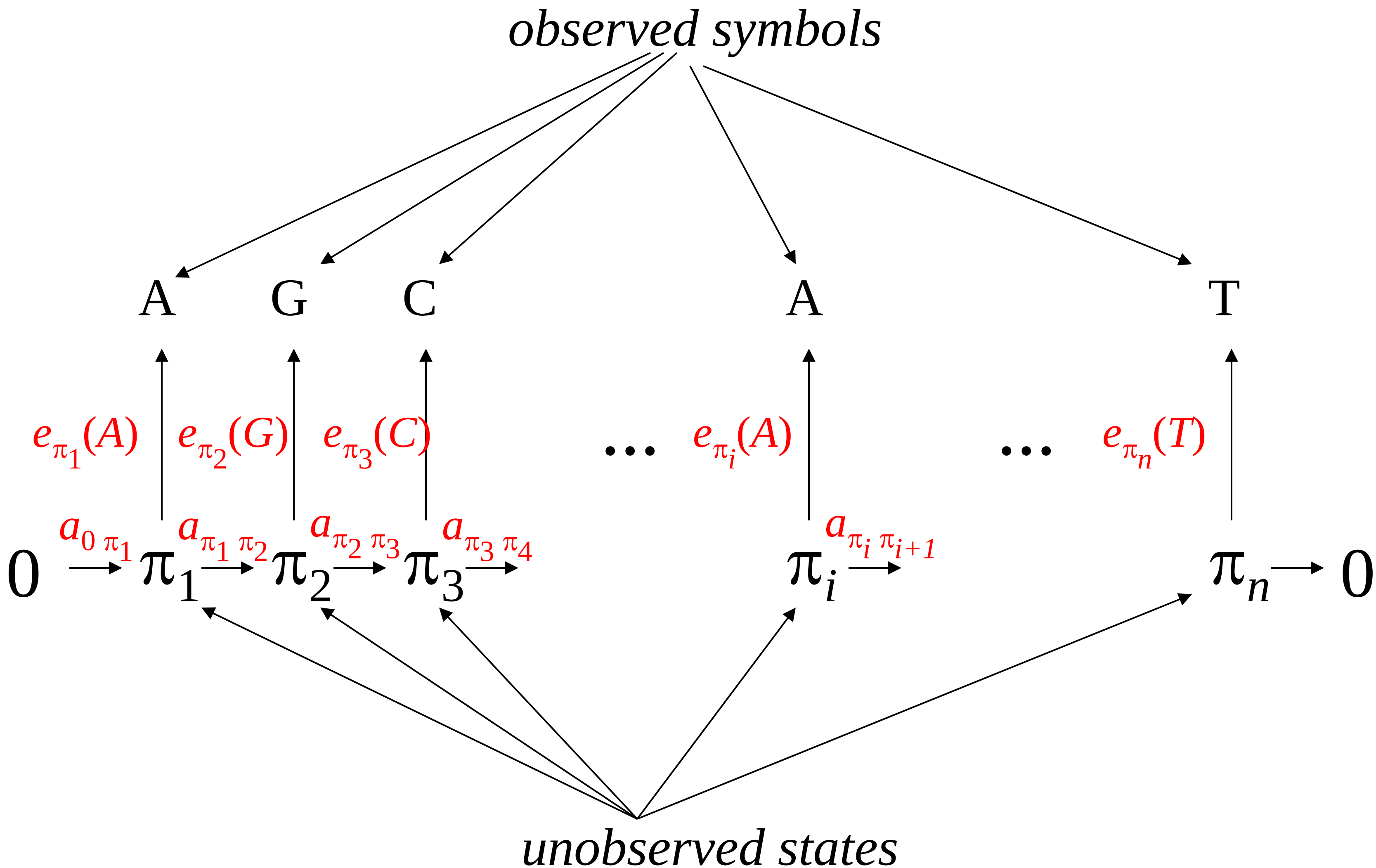
  (begin and end states do not emit symbols)

- Note that
  - there are no *direct* dependencies between observed symbols in the sequence, however
  - there are *indirect* dependencies implied by state dependencies

Where do the parameters come from?

- Can either
  - *define* parameter values *a priori*, or
  - *estimate* them from training data (observed sequences of the type to be modelled).
- Usually one does a mixture of both –
  - model topology is defined (some transitions set to 0), but
  - remaining parameters estimated

Hidden Markov Model



HMM Examples

- Site models:
  - "states" correspond to positions (columns in the tables). state i transitions only to state $i+1$:
    - $a_{i,i+1} = 1$ for all i;
    - all other a_{ij} are 0
  - emission probabilities are position-specific frequencies: values in frequency table columns

Topology for Site HMM: ‘allowed’ transitions (transits with non-zero prob – all are 1)

0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12 →

HMM for *C. elegans* 3’ Splice Sites

3’ ss

Intron | Exon

A	3276	3516	2313	476	67	757	240	8192	0	3359	2401	2514
C	970	648	664	236	129	1109	6830	0	0	1277	1533	1847
G	593	575	516	144	39	595	12	0	8192	2539	1301	1567
T	3353	3453	4699	7336	7957	5731	1110	0	0	1017	2957	2264
CONSENSUS	W	W	W	T	T	t	C	A	G	r	w	w

Emission probabilities

A	0.400	0.429	0.282	0.058	0.008	0.092	0.029	1.000	0.000	0.410	0.293	0.307
C	0.118	0.079	0.081	0.029	0.016	0.135	0.834	0.000	0.000	0.156	0.187	0.225
G	0.072	0.070	0.063	0.018	0.005	0.073	0.001	0.000	1.000	0.310	0.159	0.191
T	0.409	0.422	0.574	0.896	0.971	0.700	0.135	0.000	0.000	0.124	0.361	0.276

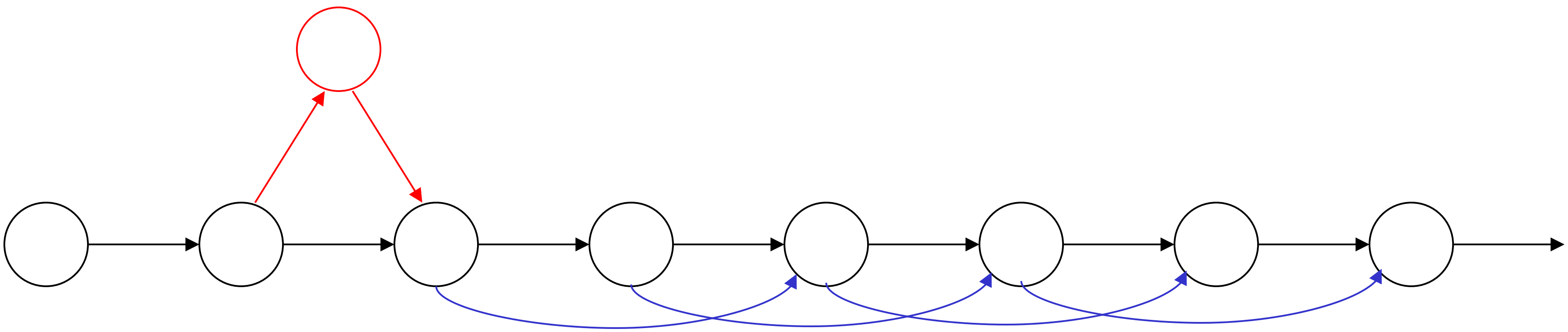
0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12

‘hidden’ states

- Can expand model to allow omission of nuc at some positions by including other (downstream) transitions (or via “silent states”)
- Can allow insertions by including additional states.
- transition probabilities no longer necessarily 1 or 0

Insertions & Deletions in Site Model

insertion state



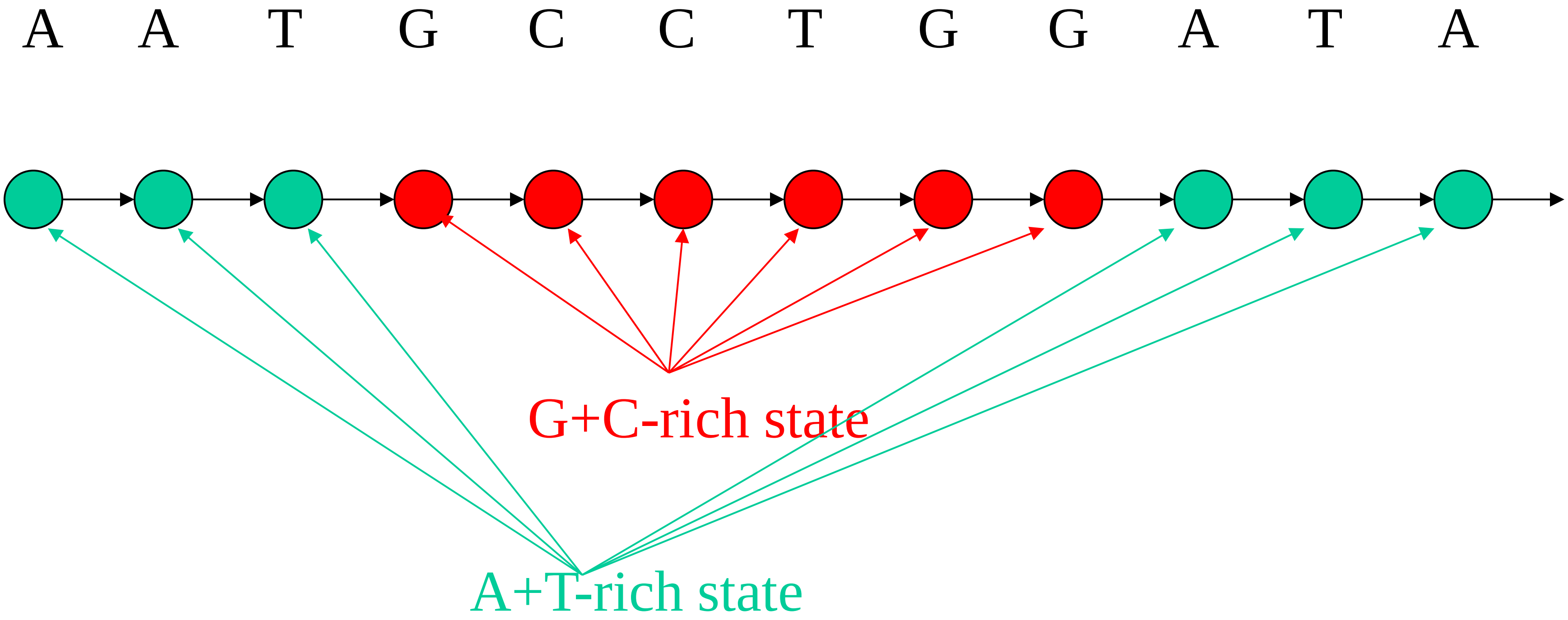
other transitions correspond
to deletions

Examples (cont'd) – 1-state HMMs

- single state, emitting residues with specified freqs:
  - = 'background' model

Examples (cont'd) – 2-state HMMs

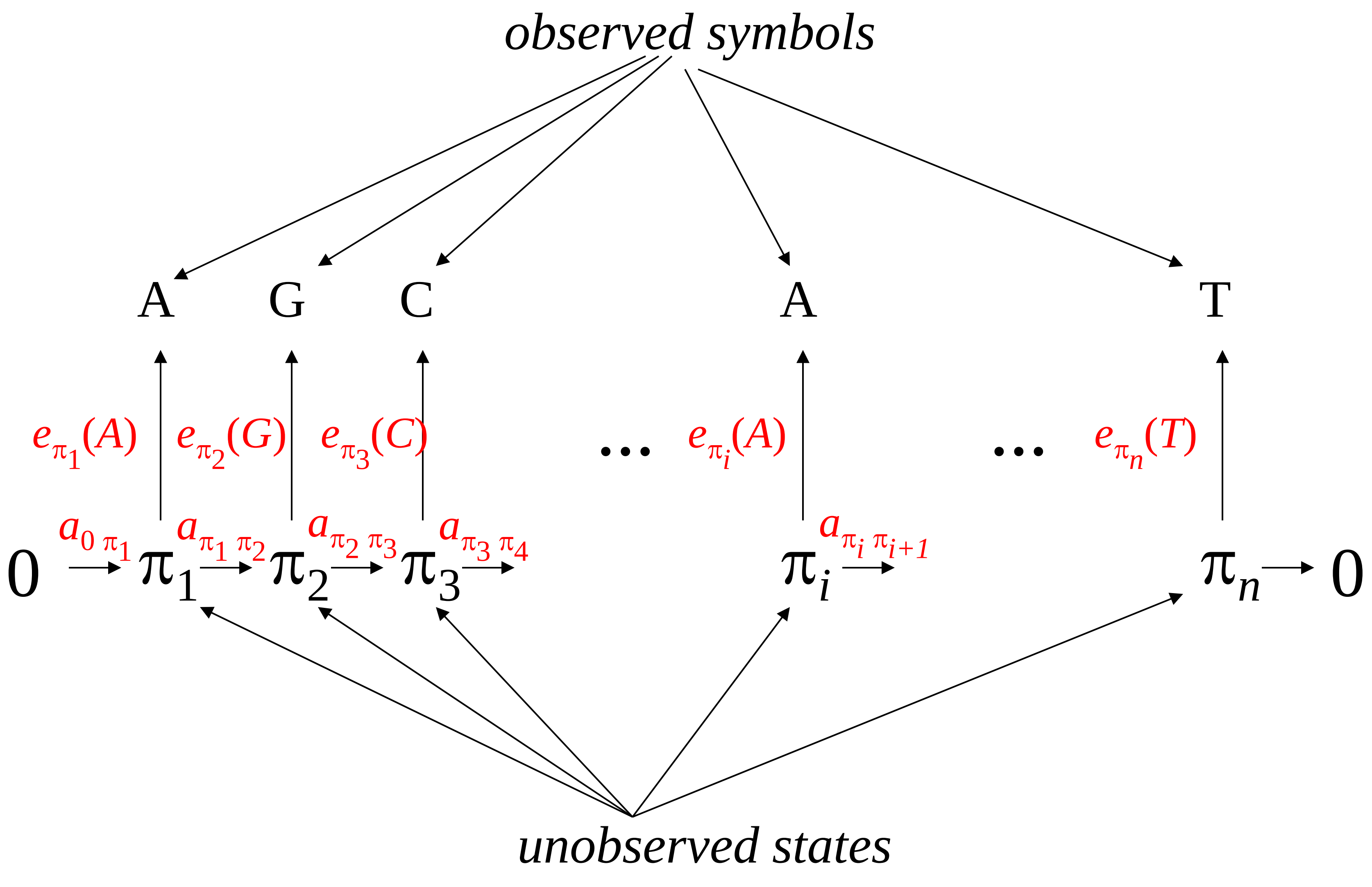
- if a_{11} and a_{22} are small (close to 0), and
  a_{12} and a_{21} are large (close to 1),
  then get (nearly) periodic model with period 2; e.g.
  - dinucleotide repeat in DNA, or
  - (some) beta strands in proteins.
- if a_{11} and a_{22} large, and
  a_{12} and a_{21} small,
  then get models of alternating regions of different compositions (specified by emission probabilities), e.g.
  - higher vs. lower G+C content regions (RNA genes in thermophilic bacteria); or
  - hydrophobic vs. hydrophilic regions of proteins (e.g. transmembrane domains).


A A T G C C T G G A T A
G+C-rich state
A+T-rich state

2-state HMMs

- Can find most probable state decomposition (‘Viterbi path’) consistent with observed sequence
- Advantages over linked-list dynamic programming method (lecture 3) for finding high-scoring segments:
  - That method assumes you *know* appropriate parameters to find targeted regions; HMM method can *estimate* parameters.
  - HMM (easily) finds multiple segments
  - HMM can attach *probabilities* to alternative decompositions
  - HMM generalization to > 2 *types* of segments is easy – just allow more states!
- Disadvantage:
  - Markov assumption on state transitions implies geometric distribution for lengths of regions -- may not be appropriate

Hidden Markov Model



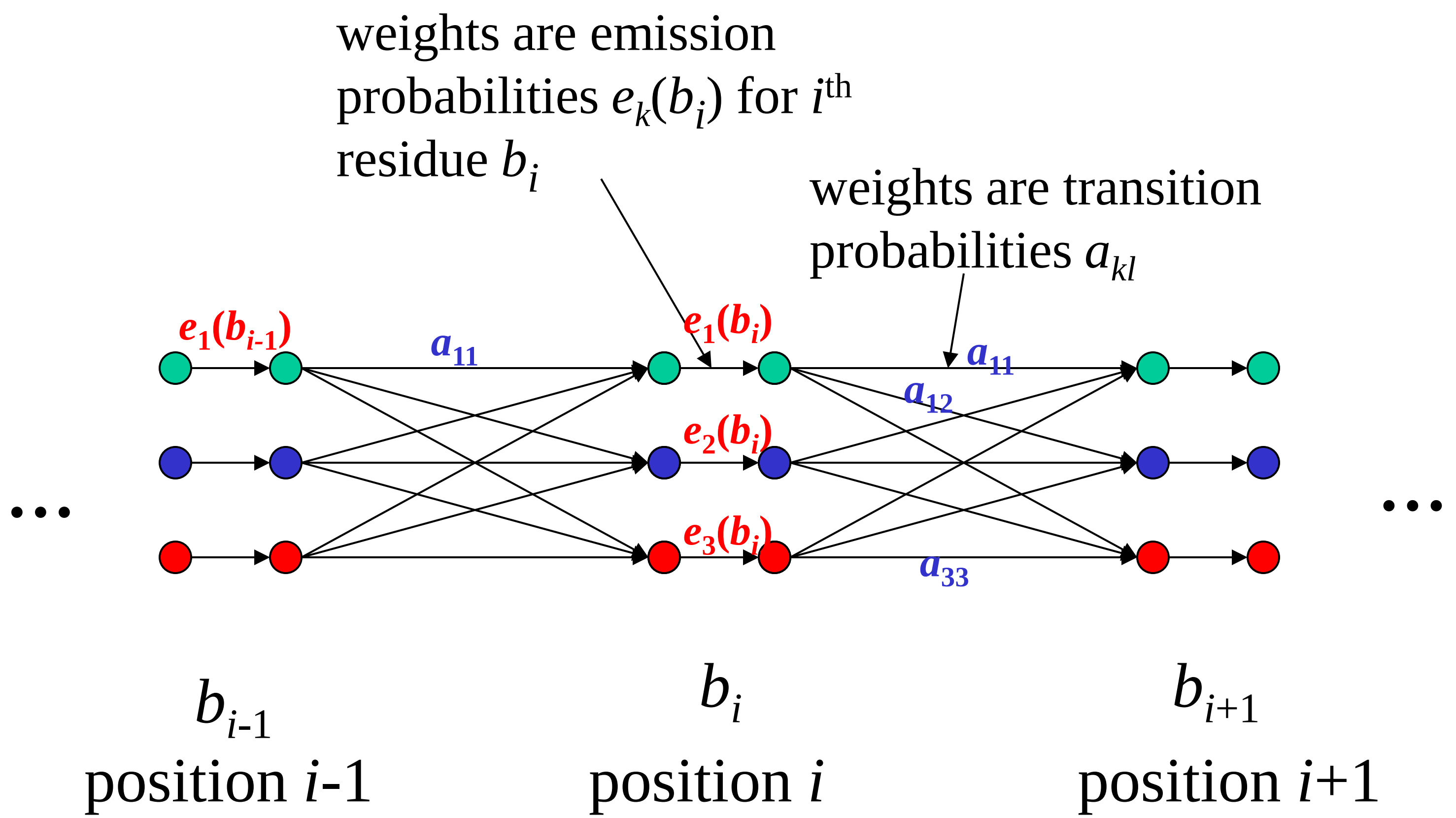
HMM Probabilities of Sequences

- Prob of sequence of states $\pi_1\pi_2\pi_3 \ldots \pi_n$ is $a_{0\pi_1}a_{\pi_1\pi_2}a_{\pi_2\pi_3}a_{\pi_3\pi_4} \ldots a_{\pi_{n-1}\pi_n}$.
- Prob of seq of observed symbols $b_1b_2b_3 \ldots b_n$, conditional on state sequence is $e_{\pi_1}(b_1)e_{\pi_2}(b_2)\, e_{\pi_3}(b_3) \ldots e_{\pi_n}(b_n)$
- Joint probability $= a_{0\pi_1}\prod^{n}_{i=1} a_{\pi_i\pi_{i+1}}\, e_{\pi_i}(b_i)$ (define $a_{\pi_n\pi_{n+1}}$ to be 1)
- (Unconditional) prob of observed sequence = sum (of joint probs) over all possible state paths
  - not practical to compute directly, by 'brute force'! We will use dynamic programming.

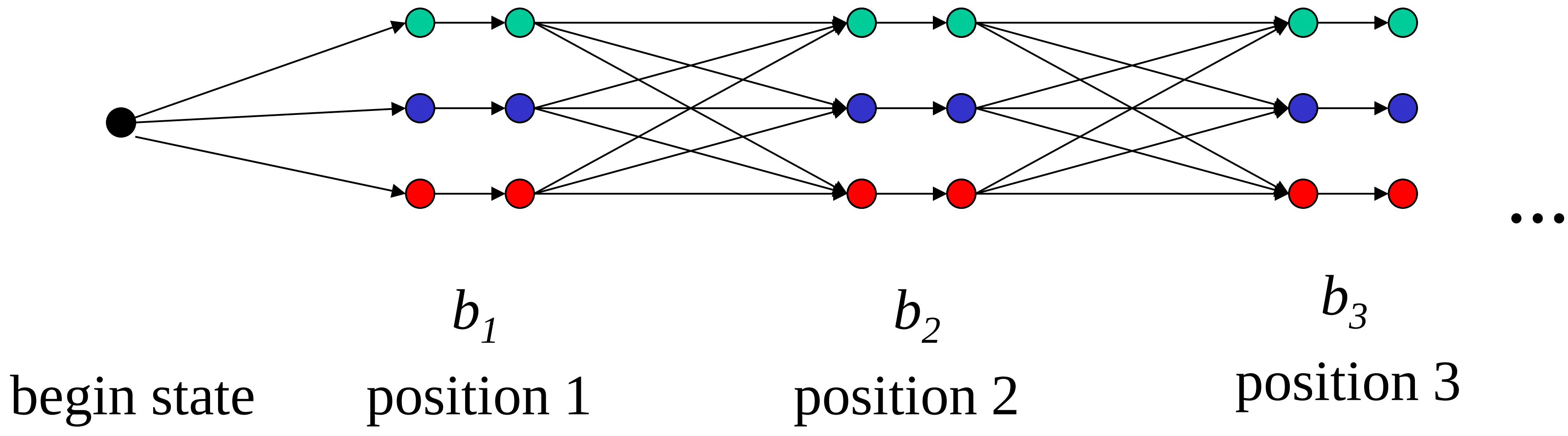
Computing HMM Probabilities

- WDAG structure for sequence HMMs:
  - for i^{th} position in seq (i = 1, ... n), have 2 nodes for each state:
    - total # nodes = $2ns + 1$, where n = seq length, s = # states
  - Pair of nodes for a given state at i^{th} position is connected by an *emission* edge
    - Weight is the emission prob for i^{th} observed residue.
    - Can omit node pair if emission prob = 0.
  - Have *transition* edges connecting (right-hand) state nodes at position i with (left-hand) state nodes at position $i+1$
    - Weights are transition probs
    - Can omit edges with transition prob = 0.

WDAG for 3-state HMM, length *n* sequence



Beginning of Graph



- ***Paths*** through graph from begin node to end node correspond to *sequences of states*
- ***Product weight*** along path

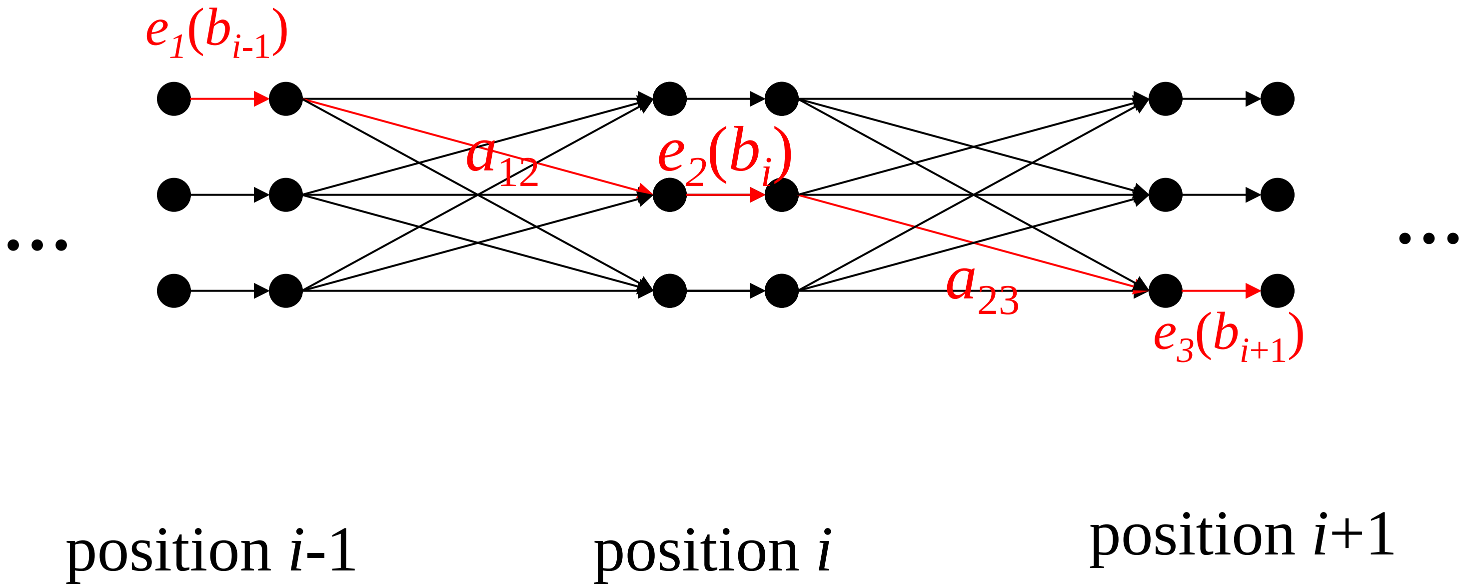
  = *joint probability* of state sequence & observed symbol sequence
- Sum of (product) path weights, over all paths,

  = *probability of observed sequence*
- Sum of (product) path weights over
  - all paths going through a particular node, or
  - all paths that include a particular edge,

  divided by prob of observed sequence,

  = *posterior probability* of that edge or node
- ***Highest-weight path*** = *highest probability state sequence*

Path Weights



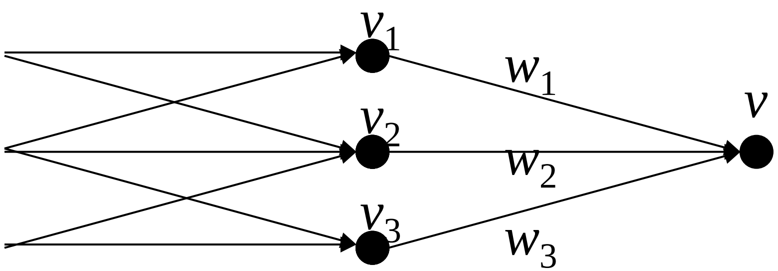
- By general results on WDAGs, can use dynamic programming to find
  - sum of all product path weights
    - = "forward algorithm" for probability of observed sequence
  - highest weight path
    - = "Viterbi algorithm" to find highest probability path
  - sum of all product path weights through particular node or particular edge
    - = "forwards/backwards algorithm" to find posterior probabilities

- In each case,
  - compute successively for each node (by increasing depth: left to right)
    - the sum (for forward & forward/backward algorithm), or
    - maximum (for Viterbi algorithm),

    of the weights of all paths ending at that node.
  - In forwards/backwards approach, work through all nodes twice, once in each direction.
- (N.B. paths are constrained to begin at the begin node!)

For each vertex v, let $f(v) = \sum_{\text{paths } p \text{ ending at } v} \text{weight}(p)$, where weight($p$) = *product* of edge weights in p. Only consider paths starting at 'begin' node.

Compute $f(v)$ by dynam. prog: $f(v) = \sum_i w_i\, f(v_i)$, where
v_i ranges over the parents of v, and
w_i = weight of the edge from v_i to v.



Similarly for $m(v) = \max_{p \text{ ending at } v} \text{weight}(p)$

and for $b(v) = \sum_{p \text{ beginning at } v} \text{weight}(p)$

(the paths *beginning* at v are the ones *ending* at v in the *reverse graph*).

The Viterbi path is

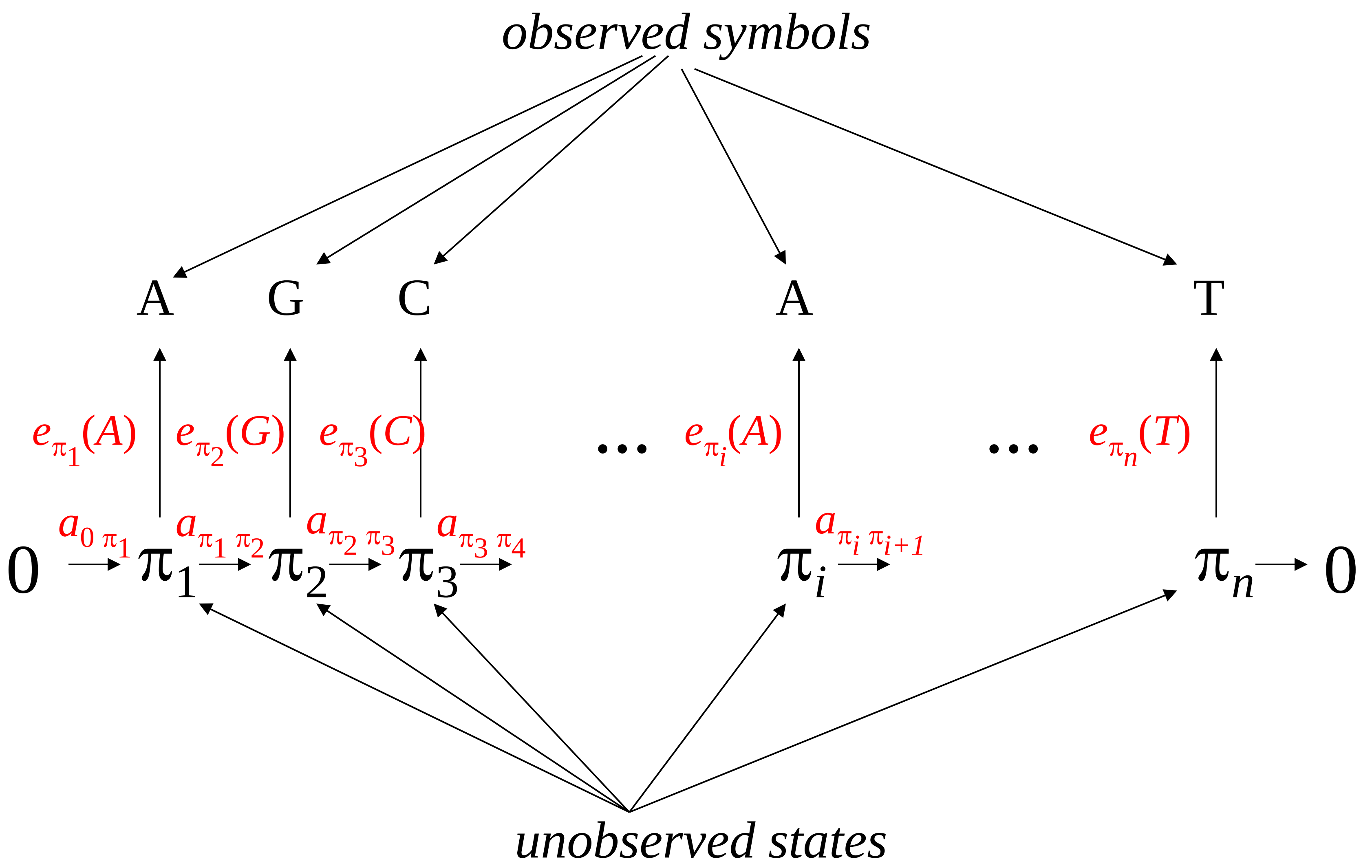
the *most probable parse*!

- Numerical issues: multiplying many small values can cause underflow. Remedies:
  - *Scale* weights to be close to 1 (affects all paths by same constant factor – which can be multiplied back later); or
  - (where possible) use log weights, so can add instead of multiplying.
  - see Rabiner & Tobias Mann links on web page
    - & will discuss further in discussion section
- Complexity: $O(|V|+|E|)$, total # vertices and edges.
  - # nodes = $2ns + 2$ where n = sequence length, s = # states.
  - # edges = $(n - 1)s^2 + ns + 2s$
  - So overall complexity is $O(ns^2)$
    - (actually s^2 can be reduced to # 'allowed' transitions between states – depends on model topology).

HMM Parameter Estimation

- Suppose parameter values (transition & emission probs) unknown
- Need to estimate from set of training sequences
- *Maximum likelihood* (ML) estimation (= choice of param vals to maximize prob of data) is preferred
  - – optimality properties of ML estimates discussed in Ewens & Grant

Hidden Markov Model



Parameter estimation when state sequence is known

- When underlying state sequence for each training sequence is *known*,
  - e.g.: weight matrix; Markov model

then ML estimates are given by:

  - emission probabilities:

    $e_k(b)$^ = (# times symbol *b* emitted by state *k*) / (# times state *k* occurs) .
  - transition probabilities:

    a_{kl} ^ = (# times state *k* followed by state *l*) / (# times state *k* occurs)
  - in denominator above, *omit occurrence at last position of sequence* (for transition probabilities)
    - But include it for emission probs
  - can include pseudocounts, to incorporate prior expectations/avoid small sample overfitting (Bayesian justification)

Parameter estimation when state sequences *unknown*

- *Viterbi training*

1. choose starting parameter values
2. find highest weight paths (Viterbi) for each sequence
3. estimate new emission and transition probs as above, *assuming* Viterbi state sequence is true
4. iterate steps 2 and 3 until convergence
   - not guaranteed to occur – but nearly always does
5. does *not* necessarily give ML estimates, but often are reasonably good