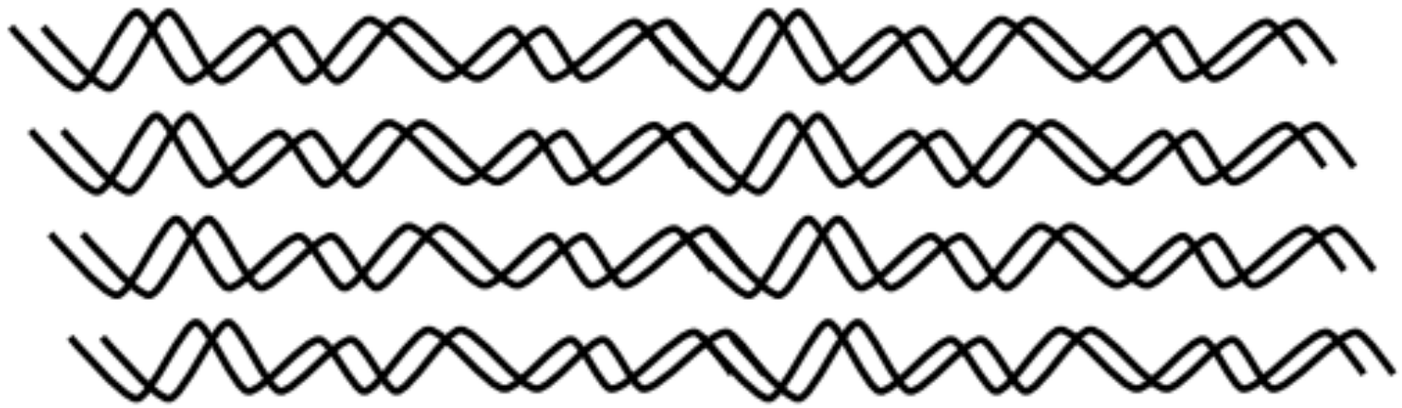
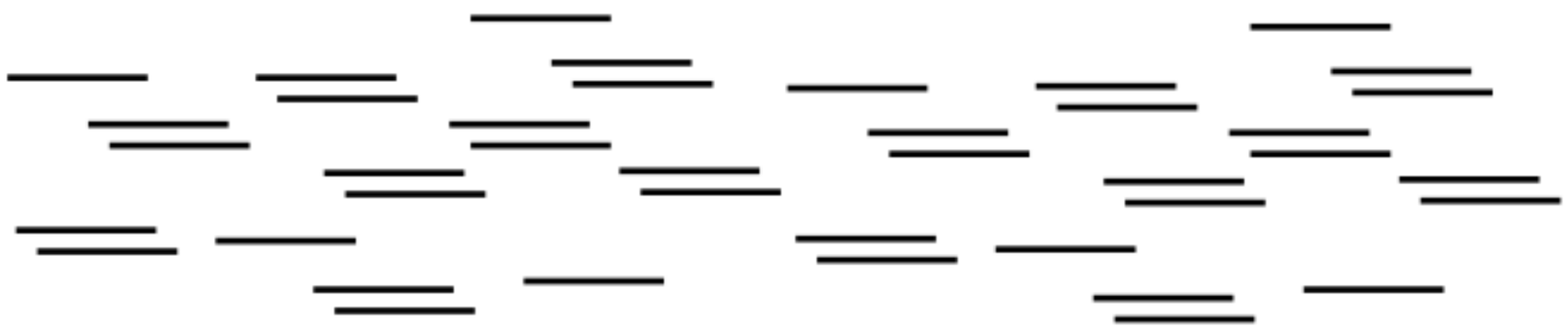
Assembly

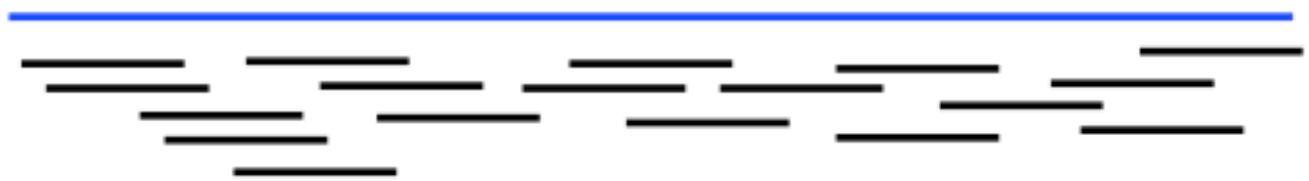
Many copies of the DNA



Shear it, randomly breaking them into many small pieces, read ends of each:



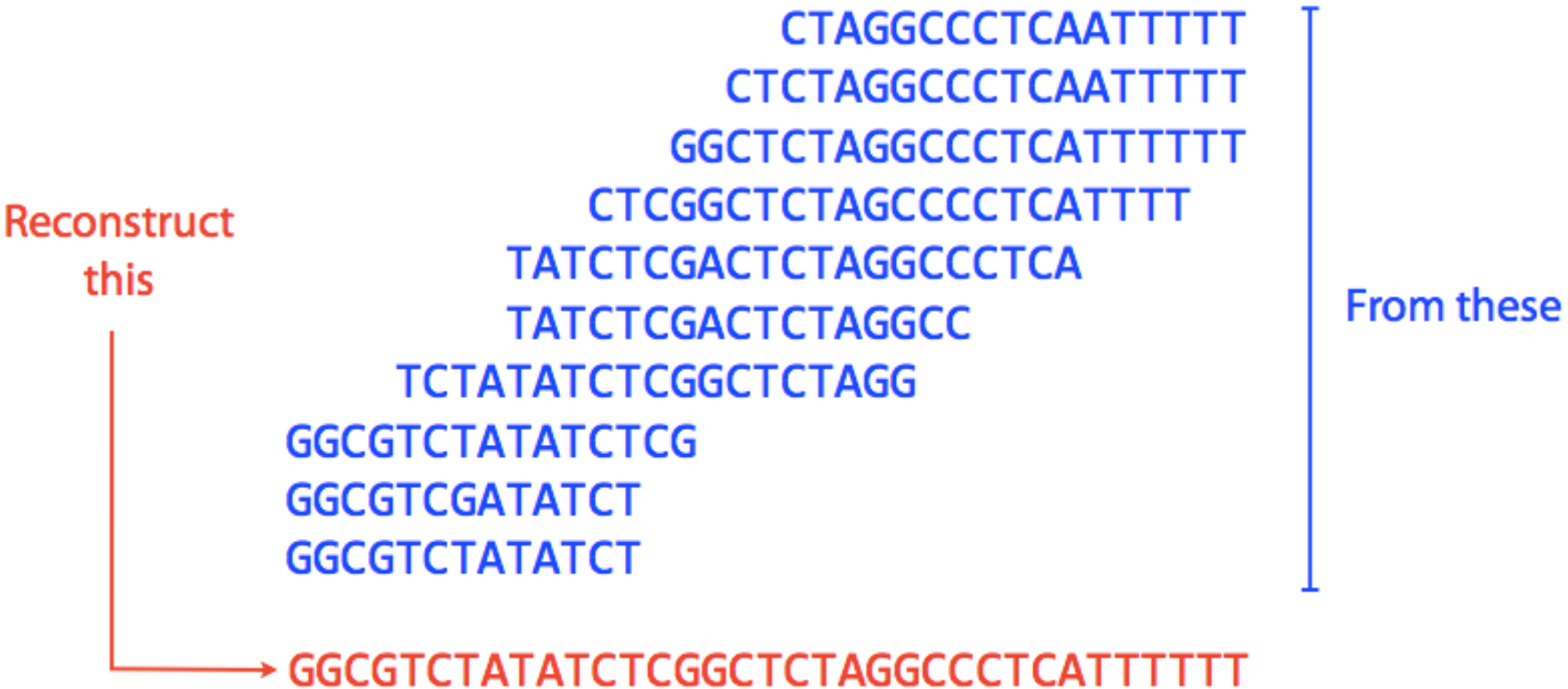
Assemble into original genome:



* Slide from Carl Kingsford

Assembly

<u>Computational Challenge</u>: assemble individual short fragments (reads) into a single genomic sequence ("superstring")



Shortest common superstring

Problem: Given a set of strings, find a shortest string that contains all of them

Input: Strings $s_1, s_2, \ldots, s_n$

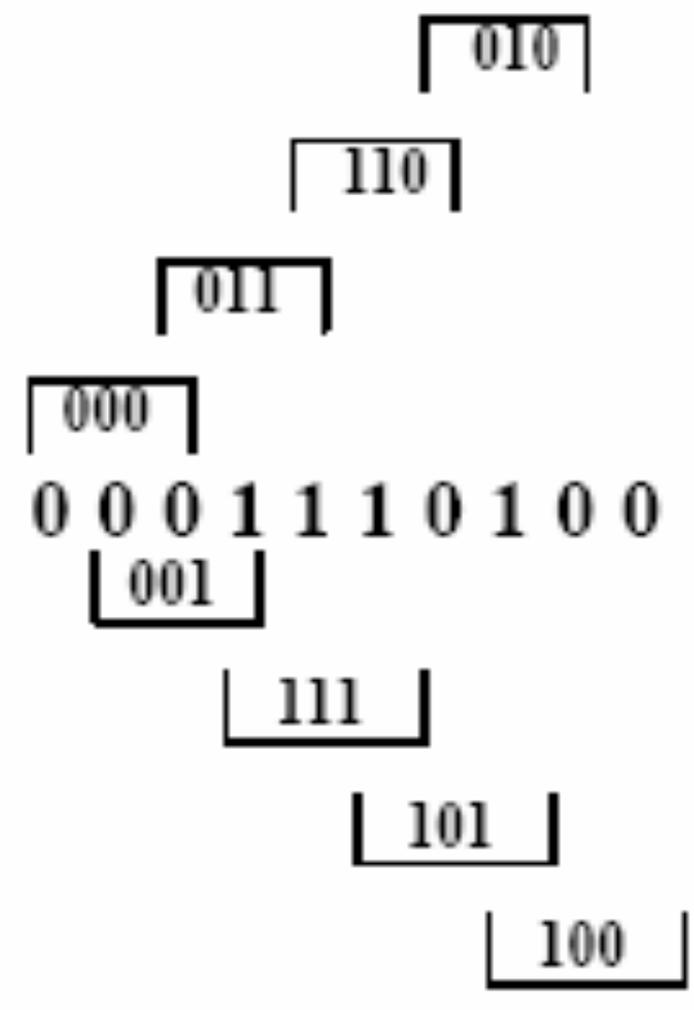
Output: A string s that contains all strings $s_1, s_2, \ldots, s_n$ as substrings, such that the length of s is minimized

Shortest common superstring

Set of strings: {000, 001, 010, 011, 100, 101, 110, 111}

Concatenation Superstring 000 001 010 011 100 101 110 111

Shortest superstring



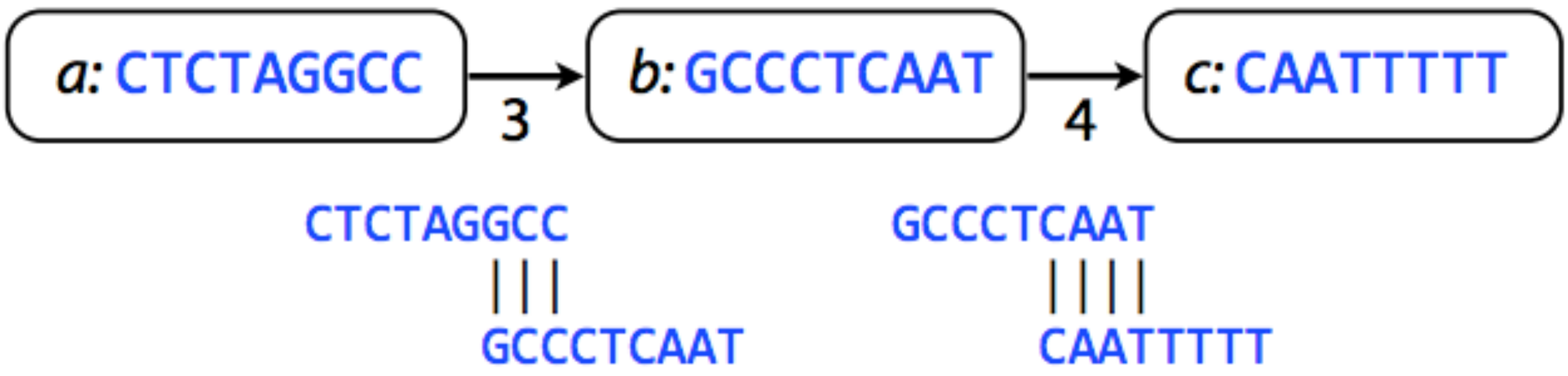
Overlap Graph

Below: overlap graph, where an overlap is a suffix/prefix match of at least 3 characters

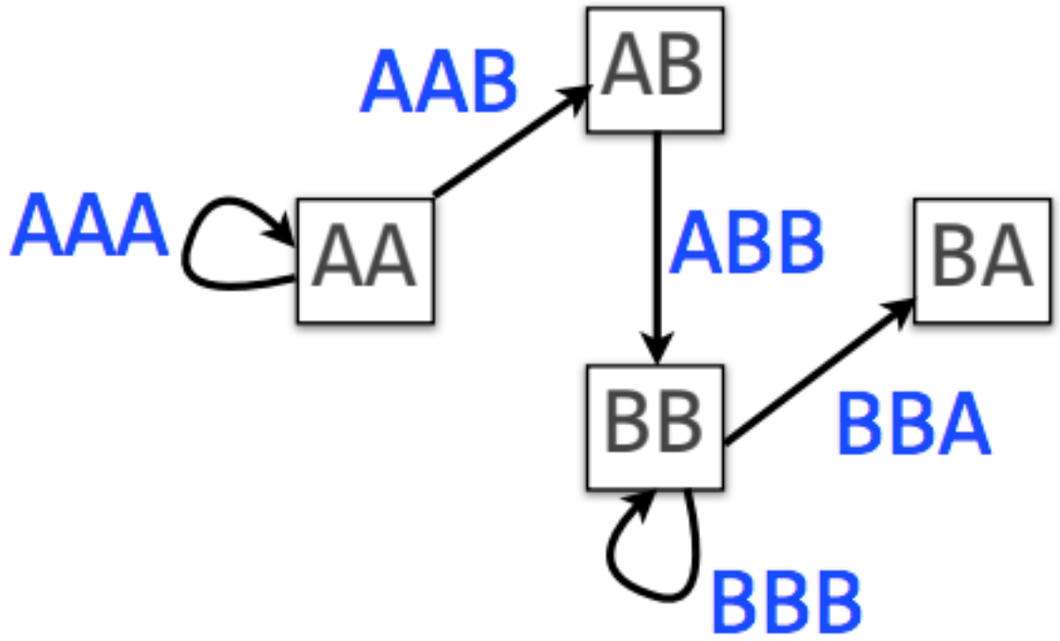
A vertex is a read, a directed edge is an overlap between suffix of source and prefix of sink

Vertices (reads): { *a:* CTCTAGGCC, *b:* GCCCTCAAT, *c:* CAATTTTT }

Edges (overlaps): { (*a*, *b*), (*b*, *c*) }



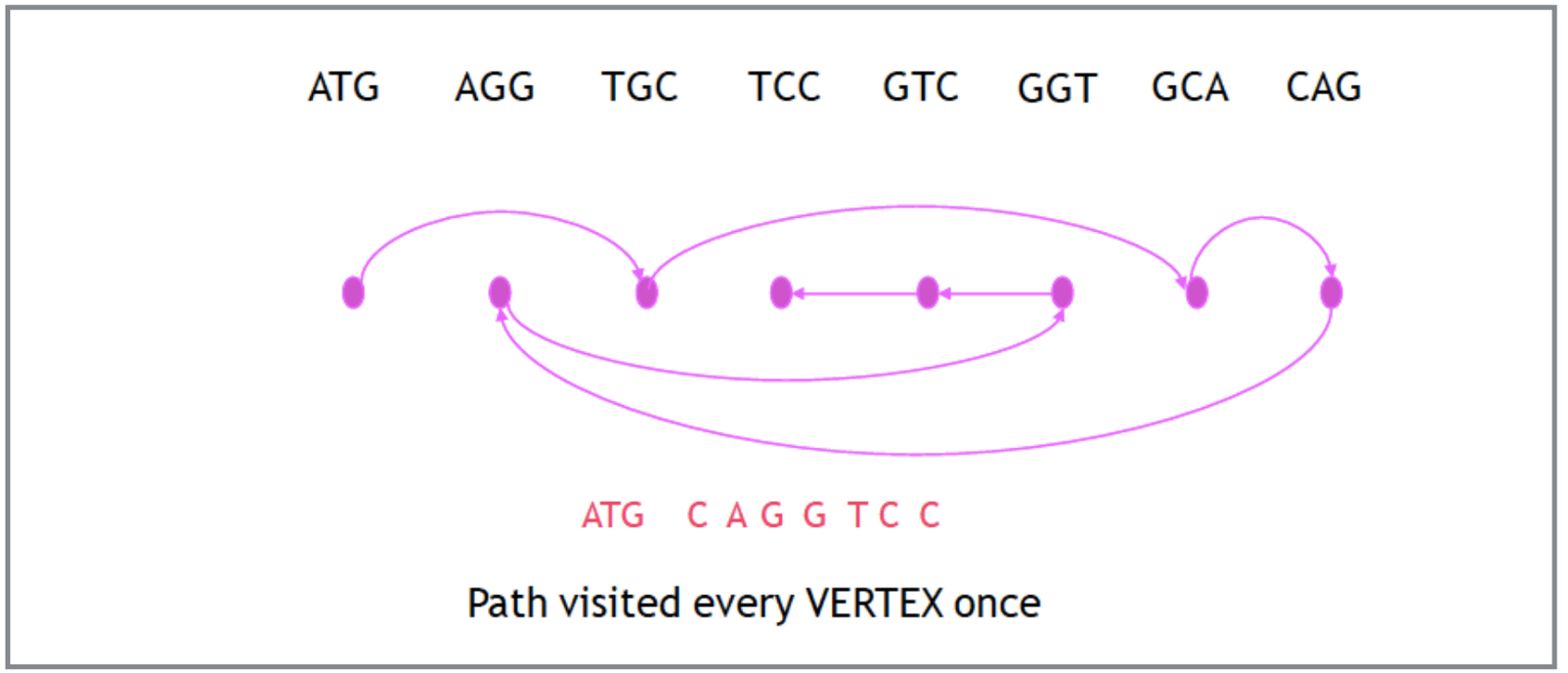
De Bruijn Graph

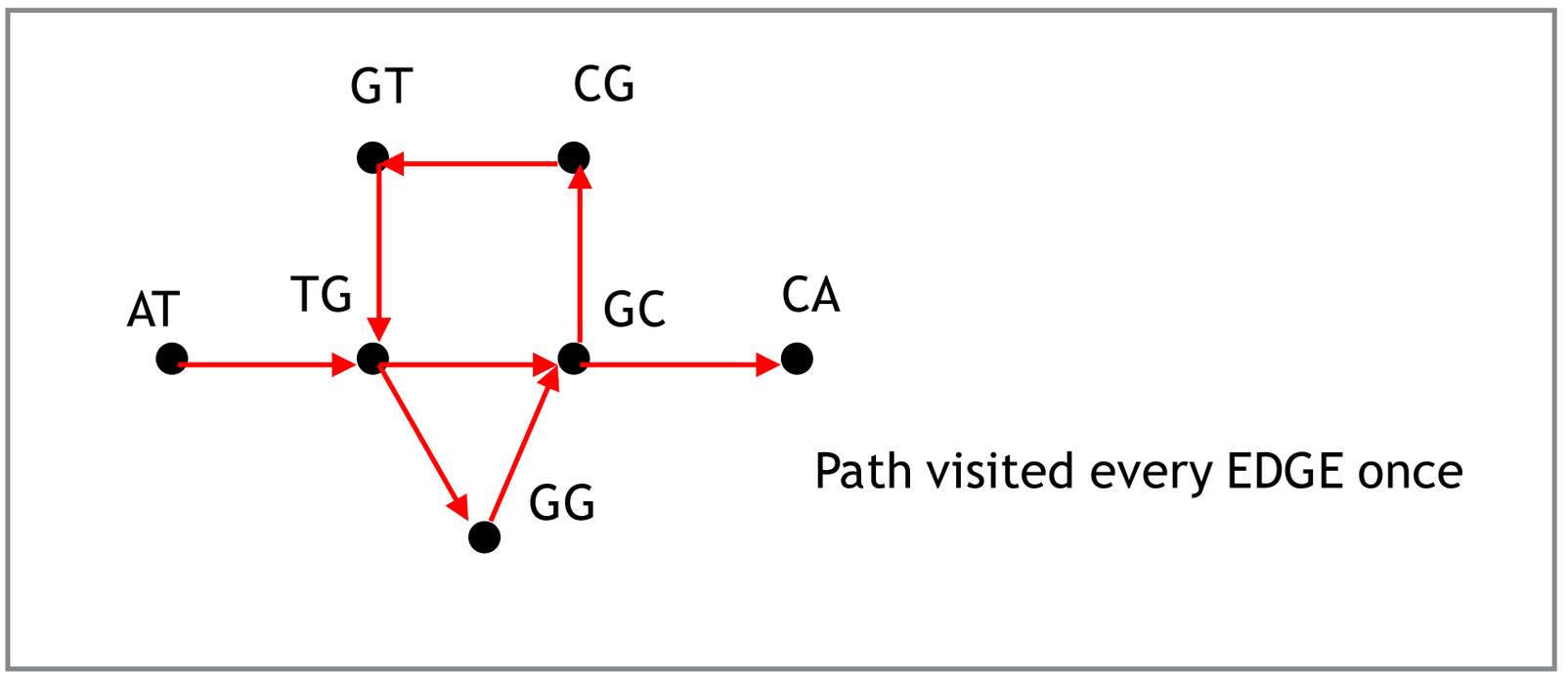


An edge corresponds to an overlap (of length *k*-2) between two *k*-1 mers.
More precisely, it corresponds to a *k*-mer from the input.

Overlap graph vs De Bruijn graph

$S = \{ ATG \;\; AGG \;\; TGC \;\; TCC \;\; GTC \;\; GGT \;\; GCA \;\; CAG \}$





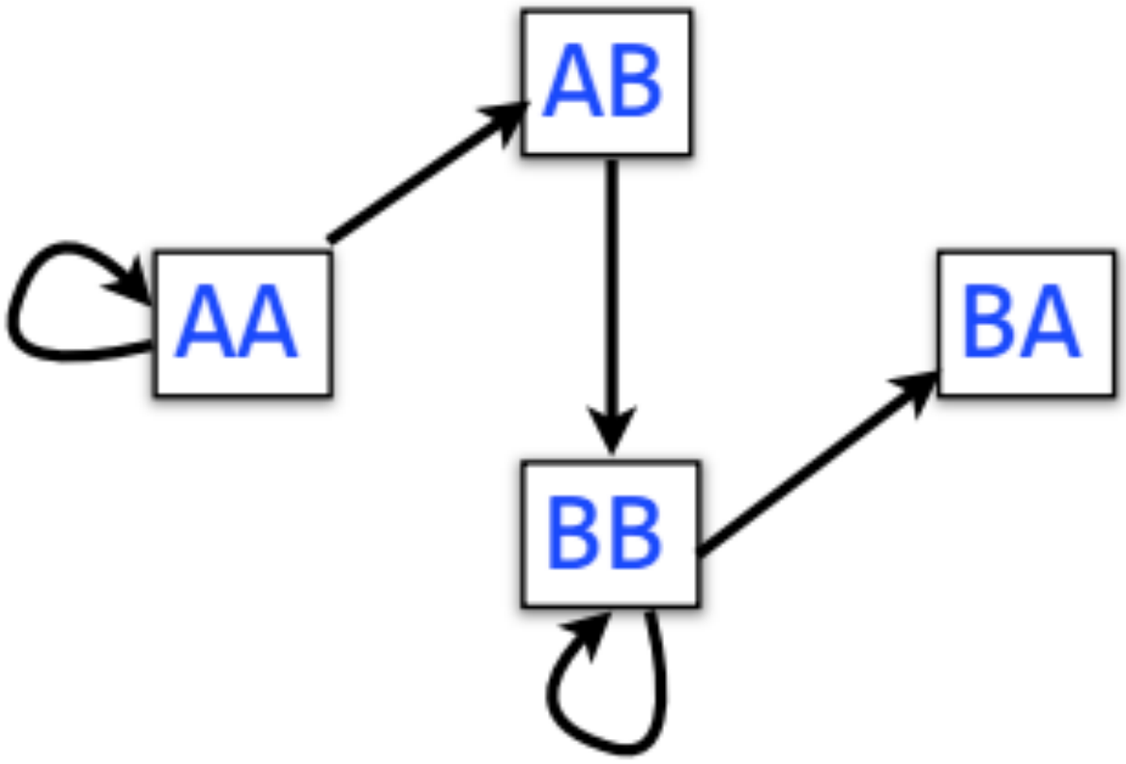
Some definitions

Node is *balanced* if indegree equals outdegree

Node is *semi-balanced* if indegree differs from outdegree by 1

Graph is *connected* if each node can be reached by some other node

Eulerian walk visits each edge exactly once

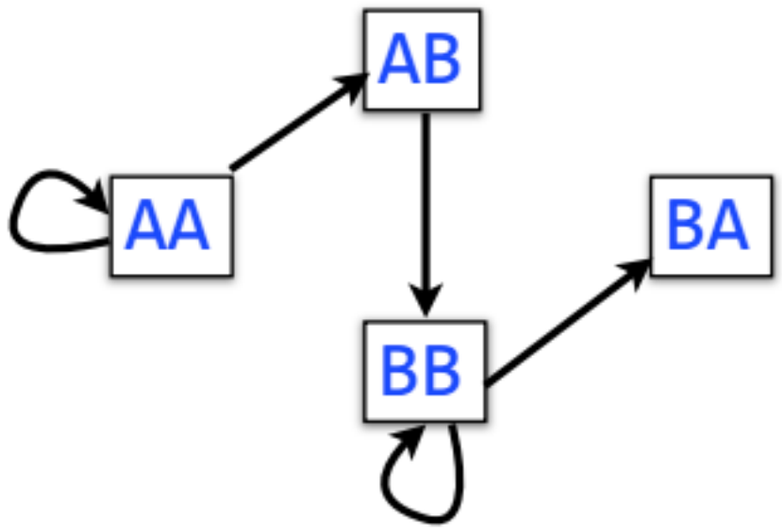


Eulerian walk/path

Eulerian walk visits each edge exactly once

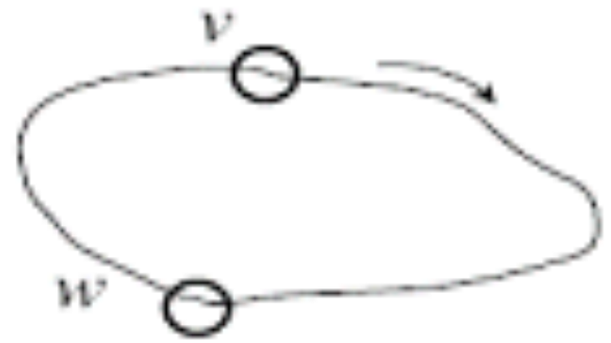
Not all graphs have Eulerian walks. Graphs that do are *Eulerian*.
(For simplicity, we won't distinguish Eulerian from semi-Eulerian.)

A directed, connected graph is Eulerian if and only if it has **zero or** 2 semi-balanced nodes and all other nodes are balanced



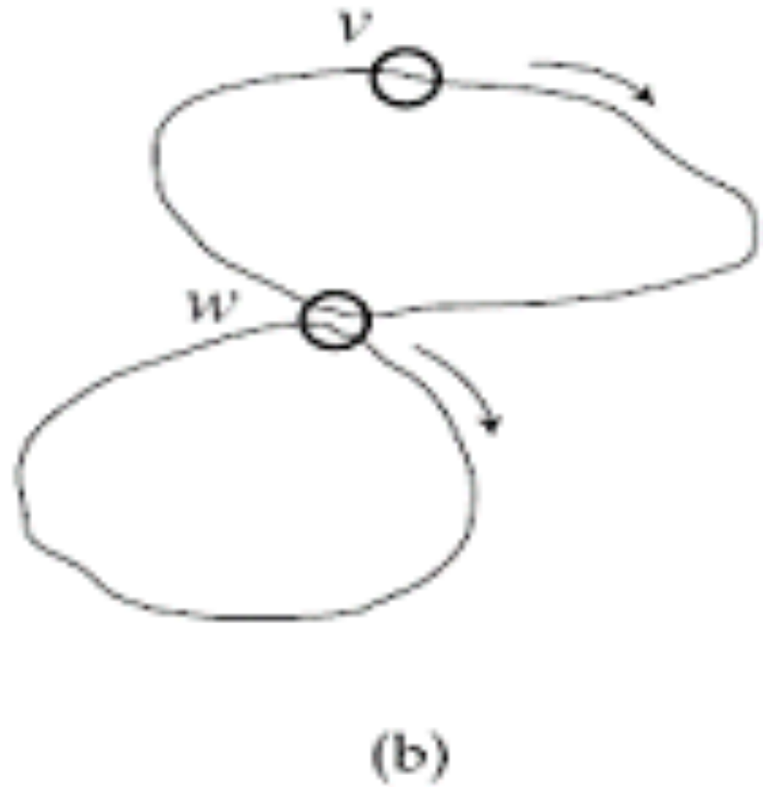
Assume all nodes are balanced

a. Start with an arbitrary vertex *v* and form an arbitrary cycle with unused edges until a dead end is reached. Since the graph is Eulerian this dead end is necessarily the starting point, i.e., vertex *v*.



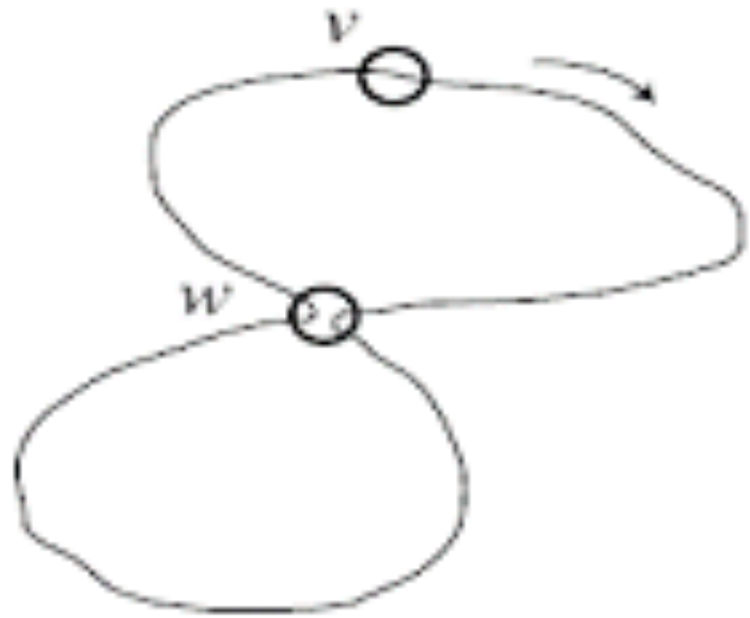
(a)

b. If cycle from (a) is not an Eulerian cycle, it must contain a vertex *w*, which has untraversed edges. Perform step (a) again, using vertex *w* as the starting point. Once again, we will end up in the starting vertex *w*.



(b)

c. Combine the cycles from (a) and (b) into a single cycle and iterate step (b).



(c)

Eulerian path

- A vertex *v* is "semibalanced" if
  | in-degree(*v*) - out-degree(*v*)| = 1

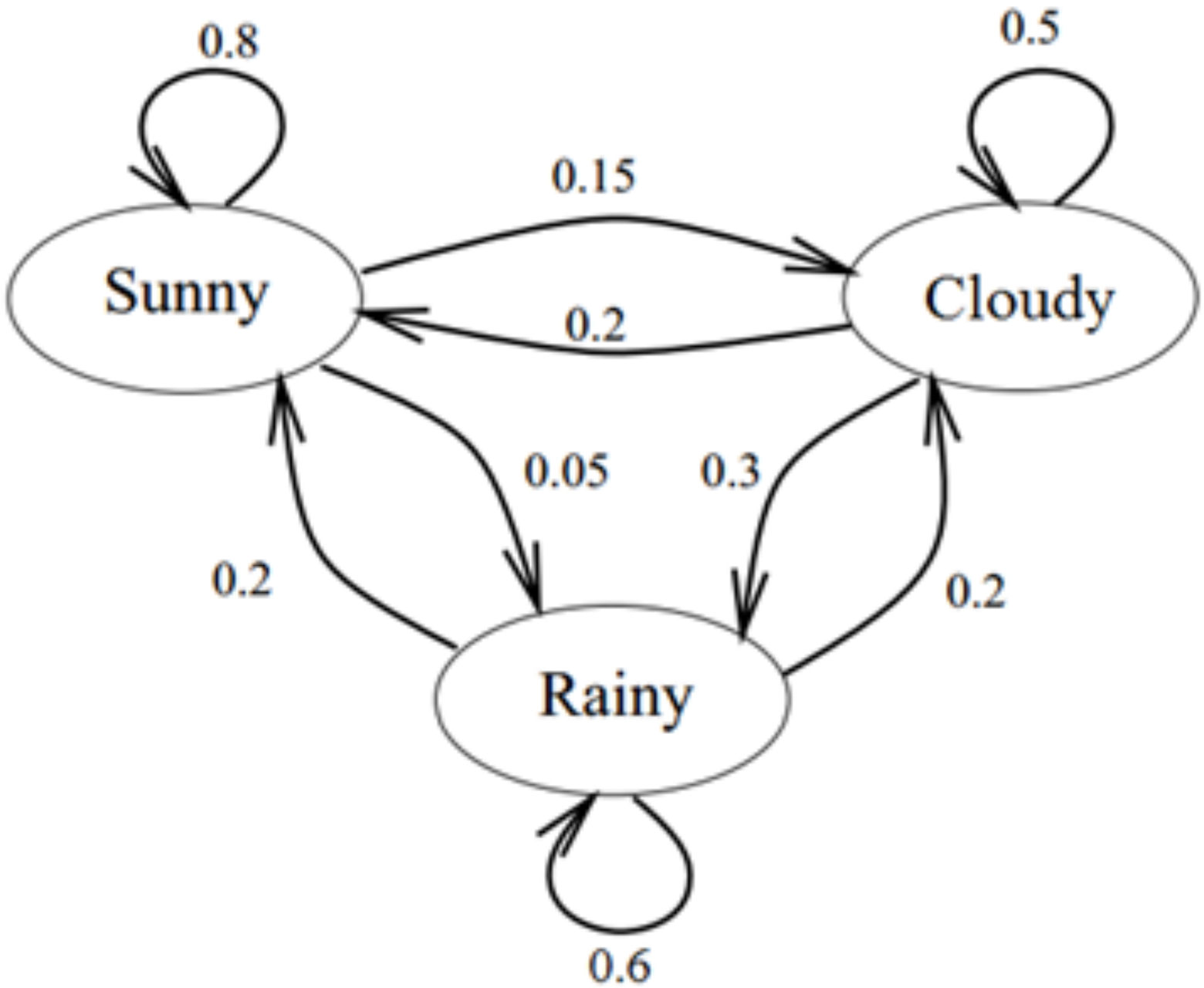
- If a graph has an Eulerian *path* starting from *s* and ending at *t*, then all its vertices are balanced *with the possible exception of s and t*

- Add an edge between two semibalanced vertices: now all vertices should be balanced (assuming there was an Eulerian path to begin with). Find the Eulerian cycle, and remove the edge you had added. You now have the Eulerian path you wanted.

Complexity?

Hidden Markov Models

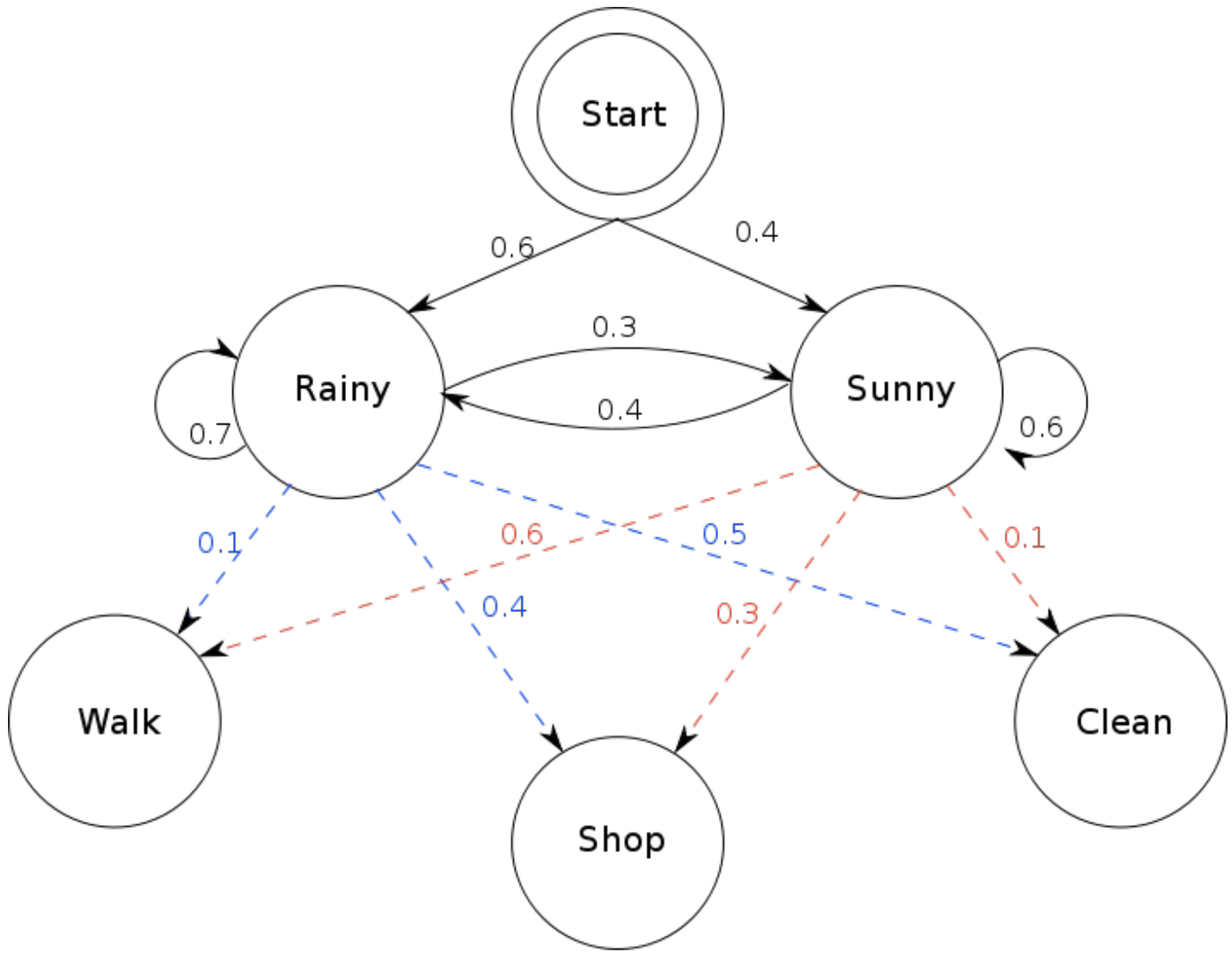
Markov Model (Finite State Machine with Probs)



Modeling a sequence of weather observations

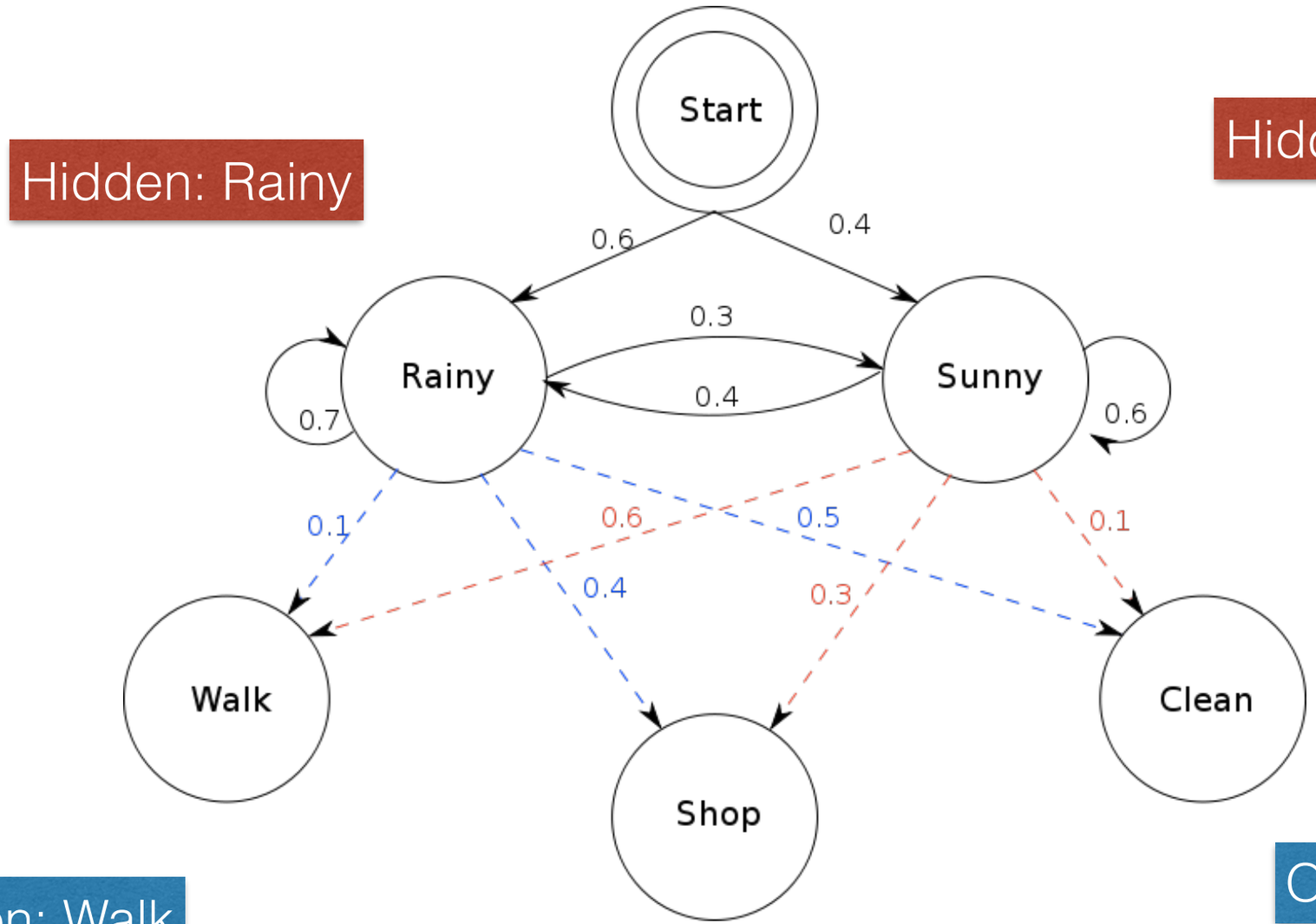
Hidden Markov Models

Assume the states in the machine are not observed and we can observe some output at certain states.



Hidden Markov Models

Assume the states in the machine are not observed and we can observe some output at certain states.



Hidden: Rainy

Hidden: Sunny

Observation: Walk

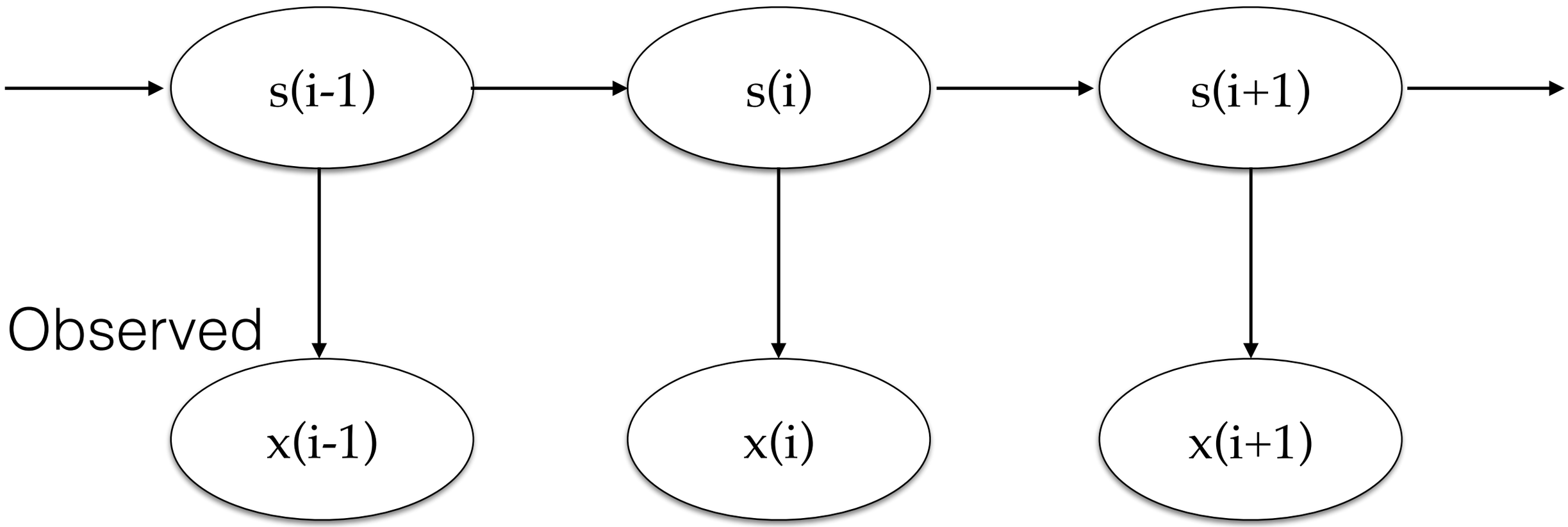
Observation: Shop

Observation: Clean

Generate a sequence from a HMM

$p(s(i)|s(i-1))$ $p(s(i+1)|s(i))$

Hidden



$p(x(i-1)|s(i-1))$ $p(x(i)|s(i))$ $p(x(i+1)|s(i+1))$

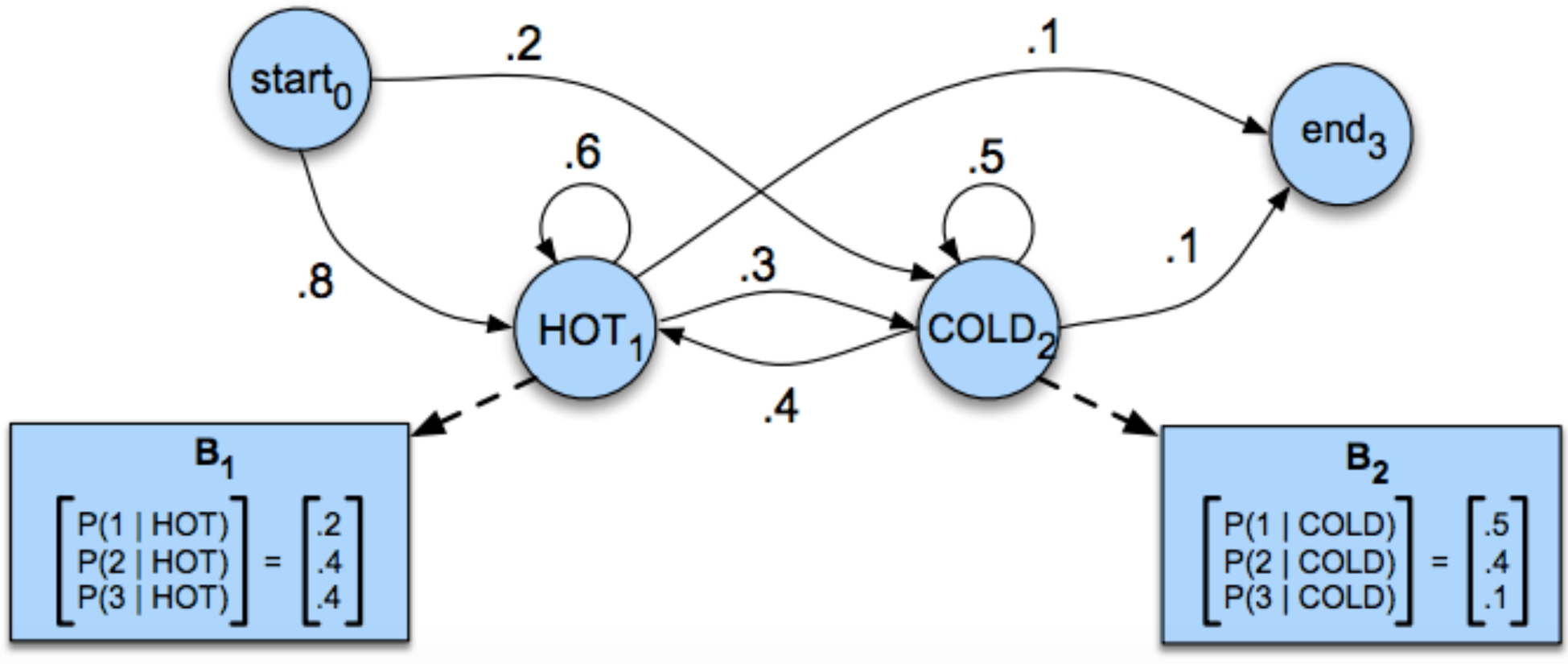
Generate a sequence from a HMM

Hidden: temperature

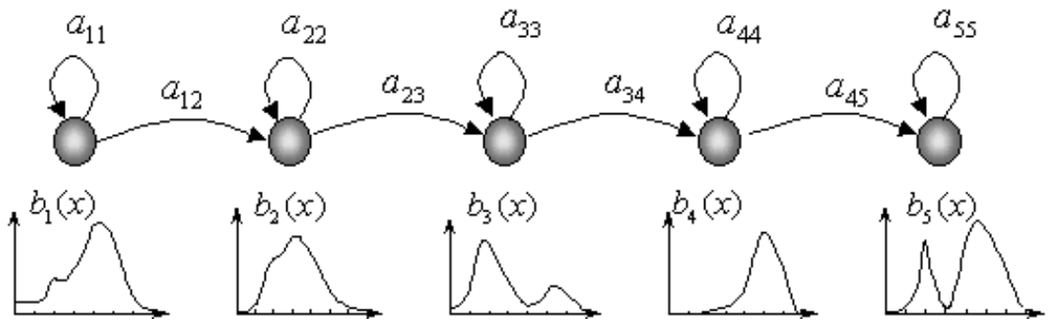
Observed: number of ice creams

HHHHHHCCCCCCCHHHHHH

33233321111111233332

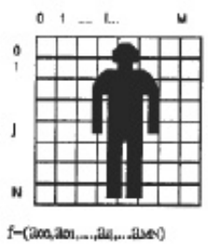


Hidden Markov Models: Applications



Speech recognition

[Yamato et al. CVPR 1992]: Tennis plays



Symbol sequence	60 61 61 62 62 62 63 63 64 64 65 66 66 66 67 68 68 69 69 70 70 70 71 71

Action recognition

Motif Finding

Problem:

Find frequent motifs with length L in a sequence dataset

ATCGCGCGGCGCGGAATCGDTATCGCGCGCC**CAGGTAAGT**

GCGCGCG**CAGGTAAGG**TATTATGCGAGACGATGTGCTATT

GTAGGCTGATGTGGGGGG**AAGGTAAGT**CGAGGAGTGCATG

CTAGGGAAACCGCGCGCGCGCGAT**AAGGTGAGT**GGGAAAG

Assumption: the motifs are very similar to each other but look very different from the rest part of sequences

Motif: a first approximation

Assumption 1: lengths of motifs are fixed to L

Assumption 2: states on different positions on the sequence are independently distributed

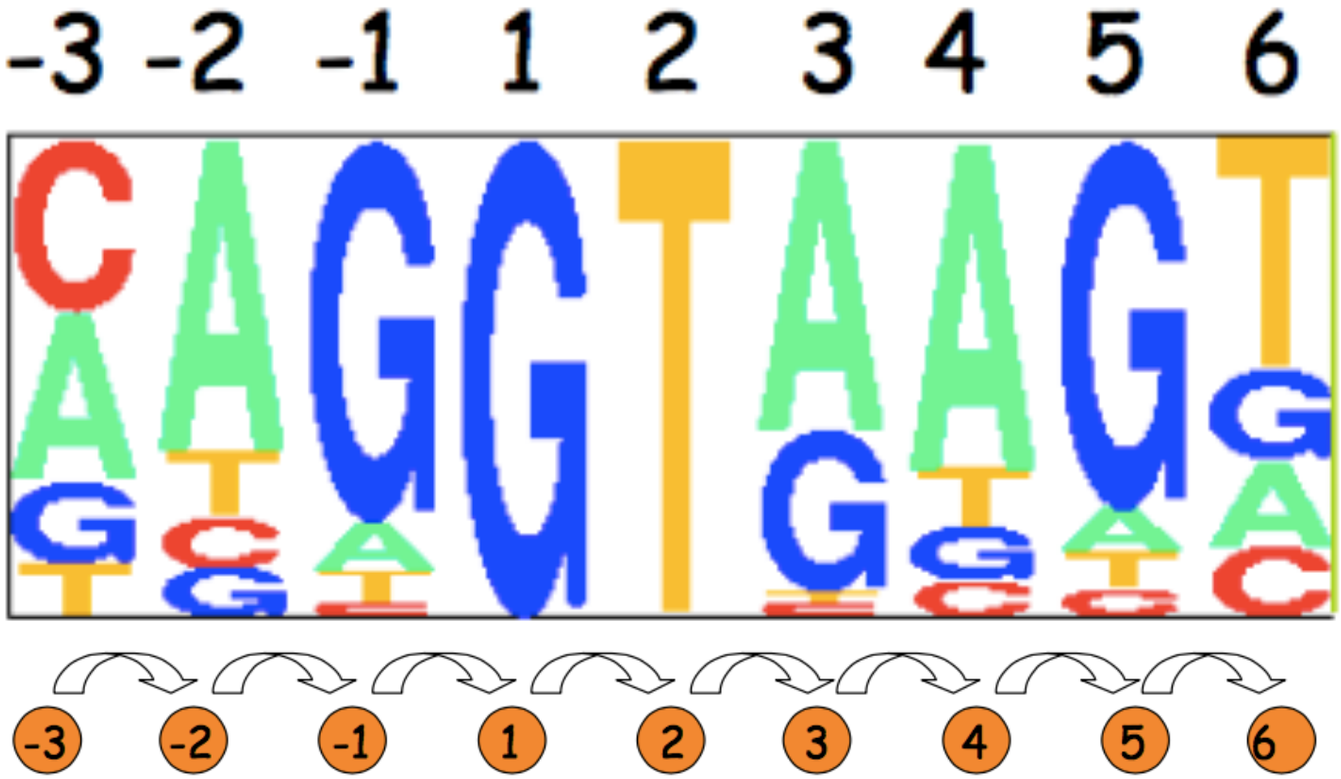
$$p_i(A) = \frac{N_i(A)}{N_i(A) + N_i(T) + N_i(G) + N_i(C)}$$

$$p(x) = \prod_{i=1}^{L} p_i(x(i))$$

Motif: (Hidden) Markov models

Assumption 1: lengths of motifs are fixed to L

Assumption 2: future letters depend only on the present letter



$$p_i(A|G) = \frac{N_{i-1,i}(G,A)}{N_{i-1}(G)}$$

$$p(x) = p_1(x(1)) \prod_{i=2}^{L} p_i(x(i)|x(i-1))$$

Motif Finding

Problem:

We don't know the exact locations of motifs in the sequence dataset

ATCGCGCGGCGCGGAATCGDTATCGCGCGCC**CAGGTAAGT**

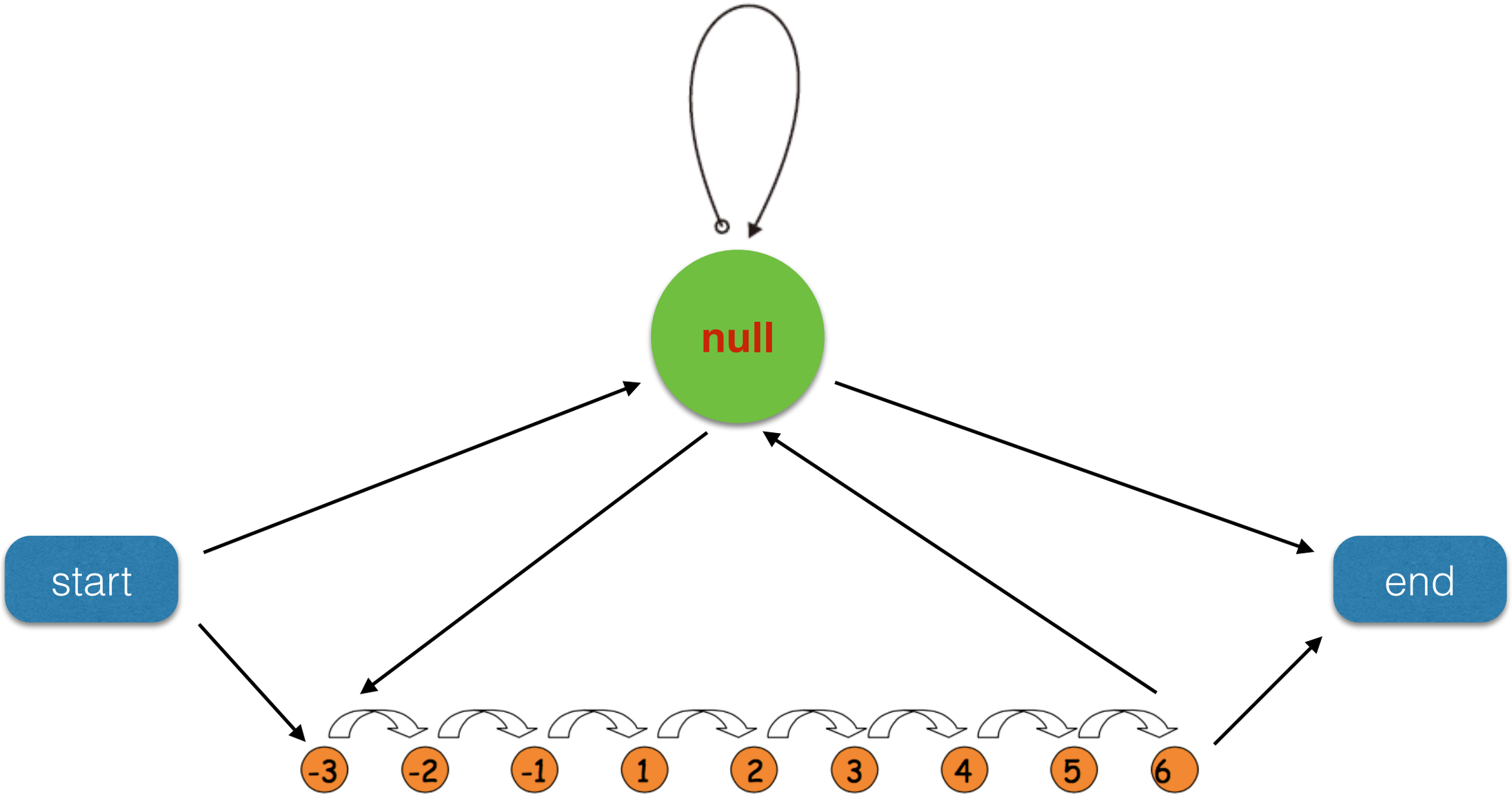
GCGCGCG**CAGGTAAGG**TATTATGCGAGACGATGTGCTATT

GTAGGCTGATGTGGGGGG**AAGGTAAGT**CGAGGAGTGCATG

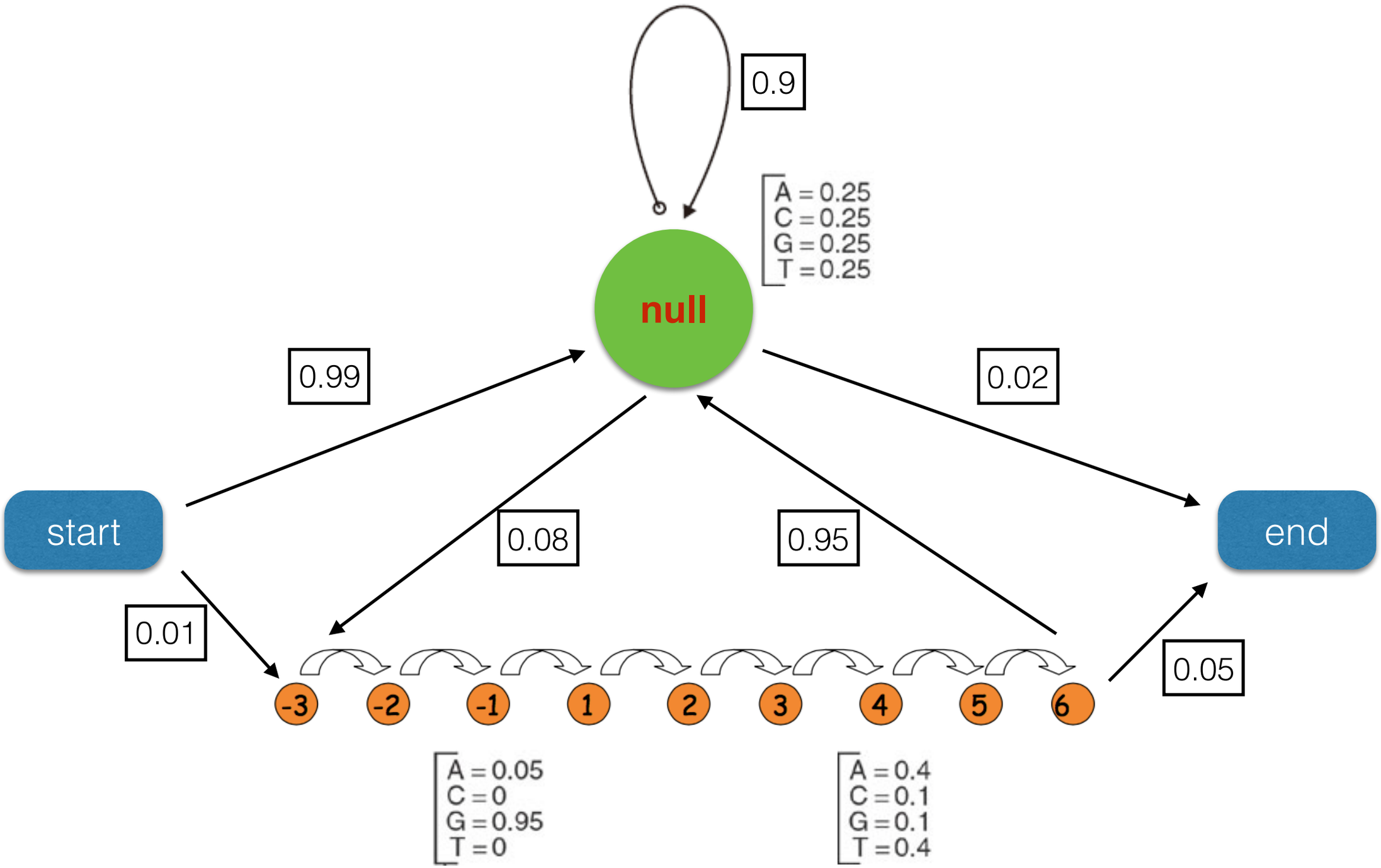
CTAGGGAAACCGCGCGCGCGCGAT**AAGGTGAGT**GGGAAAG

Assumption: the motifs are very similar to each other but look very different from the rest part of sequences

Hidden state space



Hidden Markov Model (HMM)



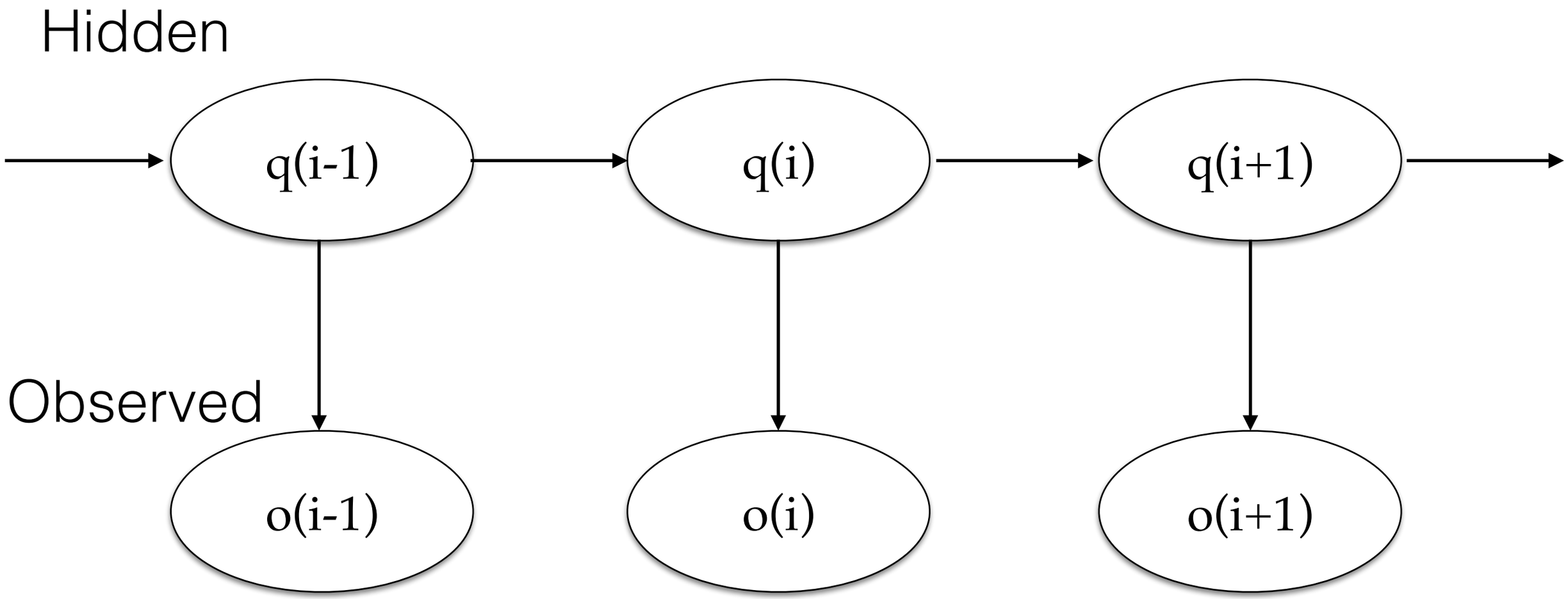
How to build HMMs?

Computational problems in HMMs

Hidden Markov Models

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state

Hidden Markov Model

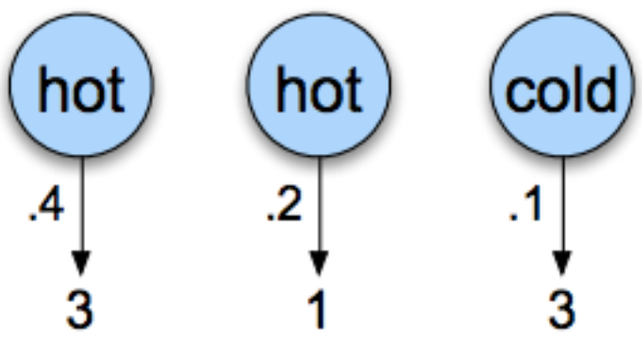


Conditional Probability of Observations

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

Example:

$$P(3\ 1\ 3|\text{hot hot cold}) = P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$

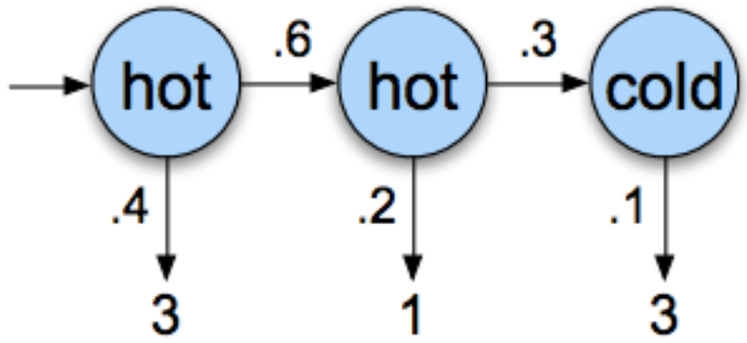


Joint and marginal probabilities

Joint:

$$P(O,Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{n} P(o_i|q_i) \times \prod_{i=1}^{n} P(q_i|q_{i-1})$$

$$\begin{aligned} P(3\ 1\ 3, \text{hot hot cold}) &= P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot}) \\ &\quad \times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold}) \end{aligned}$$



Marginal:

$$P(O) = \sum_{Q} P(O,Q) = \sum_{Q} P(O|Q)P(Q)$$

$$P(3\ 1\ 3) = P(3\ 1\ 3, \text{cold cold cold}) + P(3\ 1\ 3, \text{cold cold hot}) + P(3\ 1\ 3, \text{hot hot cold}) + \ldots$$

How to compute the probability of observations

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

$$P(O) = \sum_{Q} P(O, Q) = \sum_{Q} P(O|Q)P(Q)$$

For an HMM with N hidden states and an observation sequence of T observations, there are N^T possible hidden sequences. For real tasks, where N and T are both large, N^T is a very large number, so we cannot compute the total observation likelihood by computing a separate observation likelihood for each hidden state sequence and then summing them.

$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\lambda)$ represents the probability of being in state j after seeing the first t observations, given the automaton λ. The value of each cell $\alpha_t(j)$ is computed by summing over the probabilities of every path that could lead us to this cell.

Here, $q_t = j$ means "the tth state in the sequence of states is state j". We compute this probability $\alpha_t(j)$ by summing over the extensions of all the paths that lead to the current cell. For a given state q_j at time t, the value $\alpha_t(j)$ is computed as

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

Forward algorithm

$\alpha_{t-1}(i)$	the **previous forward path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

1. Initialization:

$$\alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$

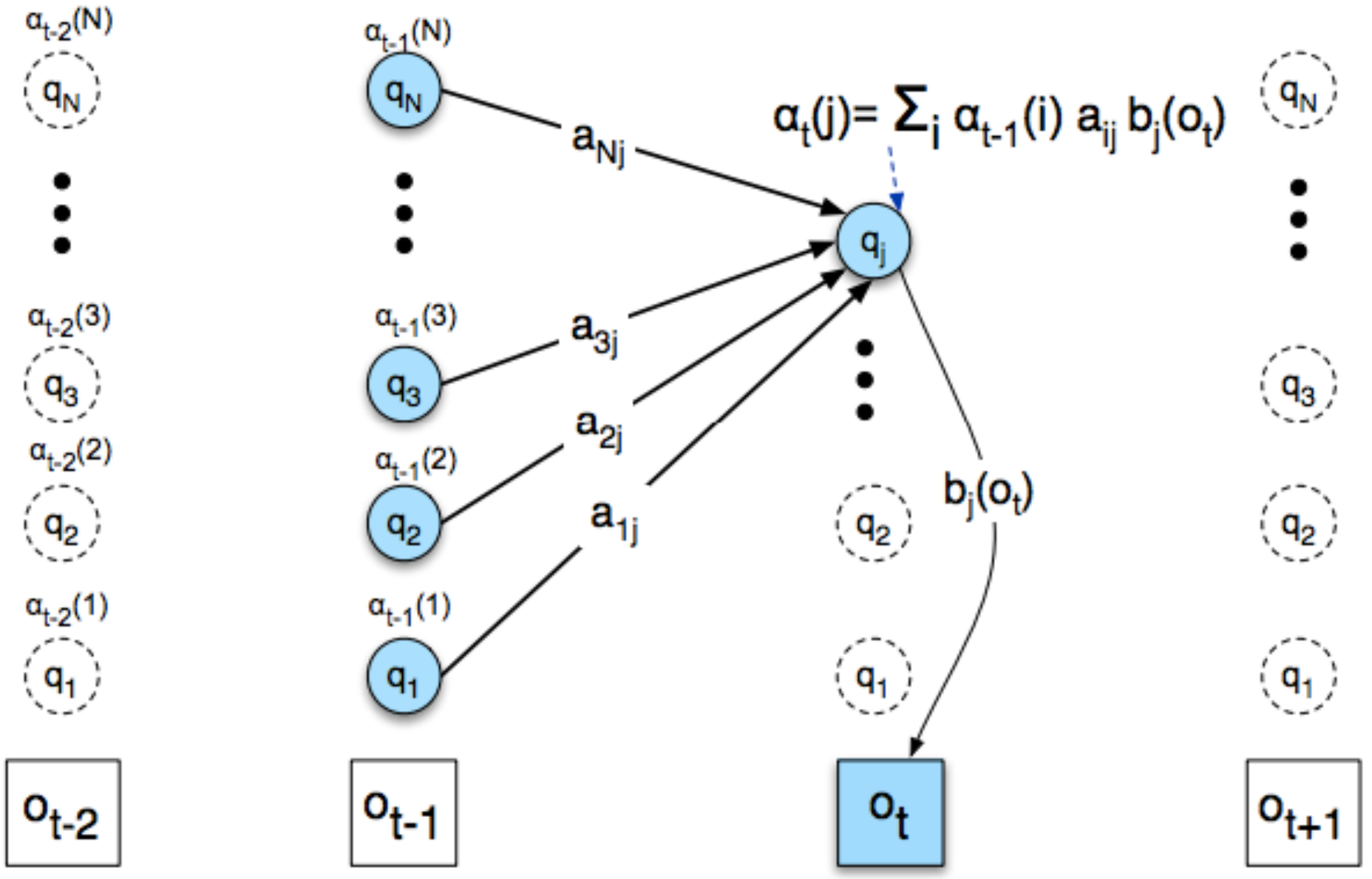
2. Recursion (since states 0 and F are non-emitting):

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

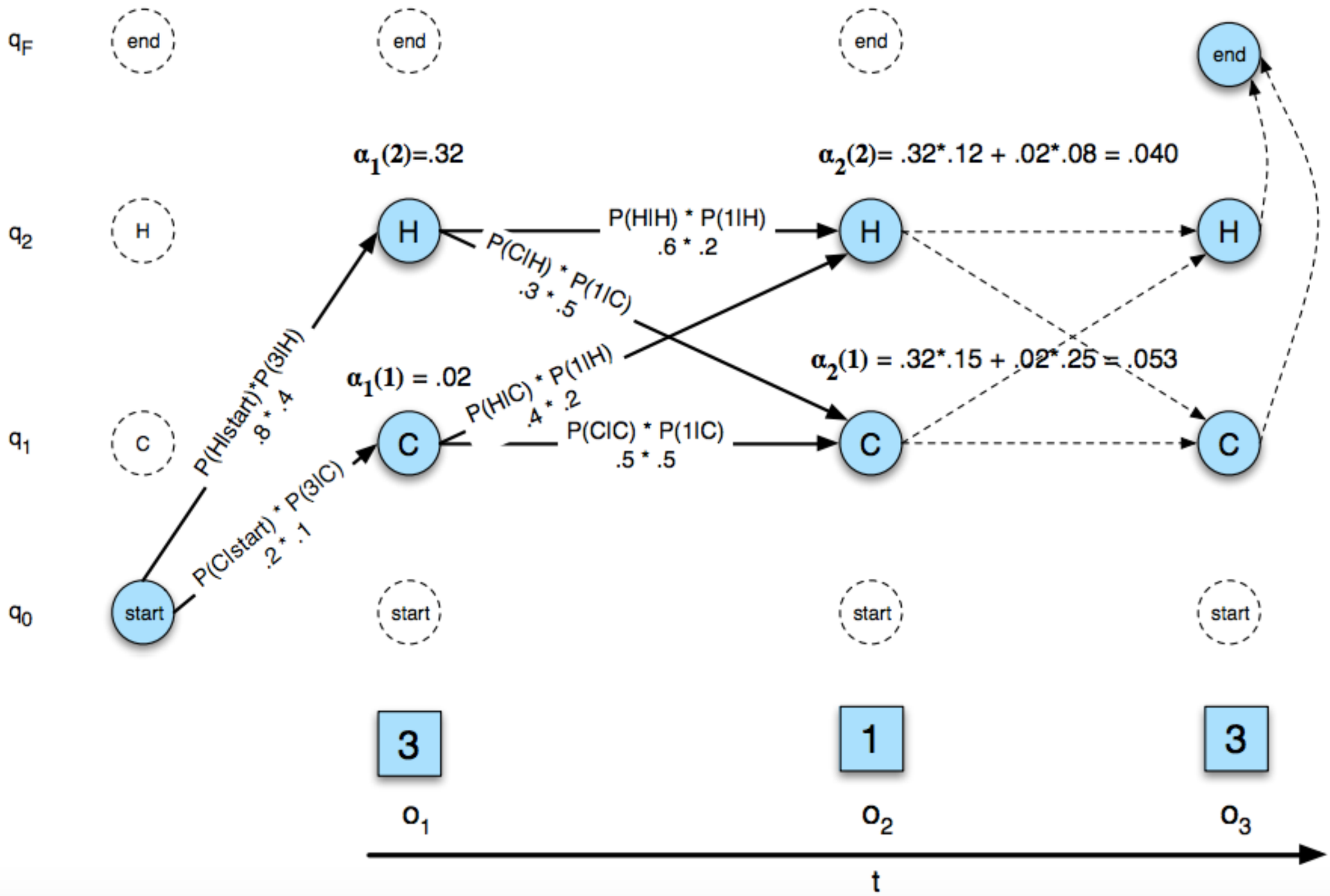
3. Termination:

$$P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i)\, a_{iF}$$

Forward algorithm



Forward algorithm



Decoding: finding the most probable states

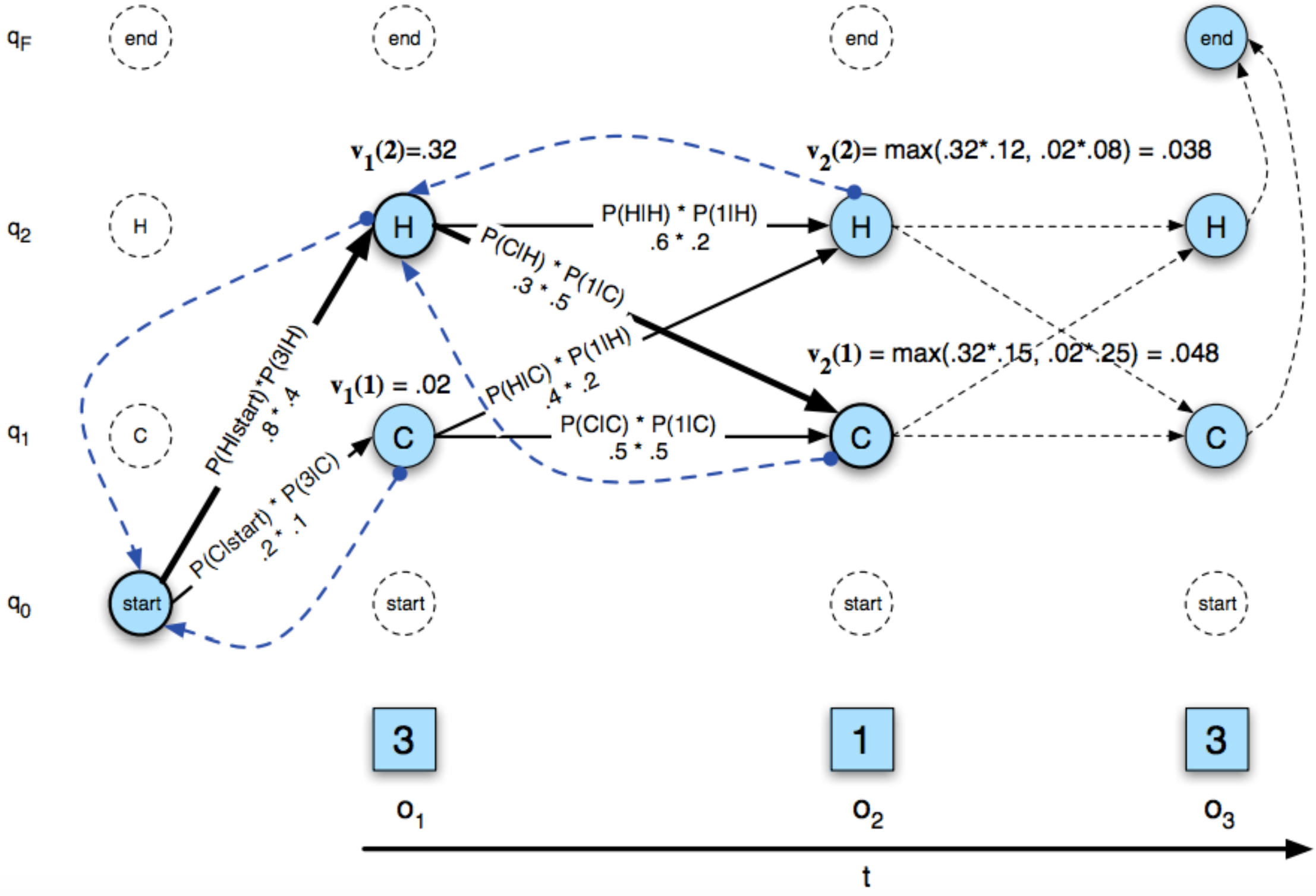
Decoding: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

Similar to the forward algorithm, we can define the following value:

$$v_t(j) = \max_{q_0, q_1, \cdots, q_{t-1}} P(q_0, q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = j | \lambda)$$

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j


q_F
q_2
q_1
q_0
end
end
end
end
H
C
start
start
start
start
$v_1(2)$=.32
$v_2(2)$= max(.32*.12, .02*.08) = .038
$v_1(1)$ = .02
$v_2(1)$ = max(.32*.15, .02*.25) = .048
P(H|H) * P(1|H)
.6 * .2
P(C|H) * P(1|C)
.3 * .5
P(H|C) * P(1|H)
.4 * .2
P(C|C) * P(1|C)
.5 * .5
P(H|start)*P(3|H)
.8 * .4
P(C|start) * P(3|C)
.2 * .1
3
1
3
o_1
o_2
o_3
t

Viterbi algorithm

1. **Initialization:**

$$\begin{aligned} v_1(j) &= a_{0j} b_j(o_1) \quad 1 \le j \le N \\ bt_1(j) &= 0 \end{aligned}$$

2. **Recursion** (recall that states 0 and q_F are non-emitting):

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

$$bt_t(j) = \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

3. **Termination:**

$$\text{The best score:} \quad P* = v_T(q_F) = \max_{i=1}^{N} v_T(i) * a_{iF}$$

$$\text{The start of backtrace:} \quad q_T* = bt_T(q_F) = \operatorname*{argmax}_{i=1}^{N} v_T(i) * a_{iF}$$