Fall 2010 Graduate Course on
Dynamic Learning

Chapter 7: Hidden Markov Models

October 25, 2010

Byoung-Tak Zhang

School of Computer Science and Engineering &
Cognitive Science and Brain Science Programs
Seoul National University

http://bi.snu.ac.kr/~btzhang/

Overview

- Motivating Applications
  - Computational Linguistics
  - Computational Biology
- HMM: A Graphical Description
- HMM: A More Formal Description
  - Elements of HMM
  - Three Central Problems
- Algorithms
  - Evaluation: Forward-Backward Algorithm
  - Decoding: Viterbi Algorithm
  - Learning: Baum-Welch Algorithm
- Extensions

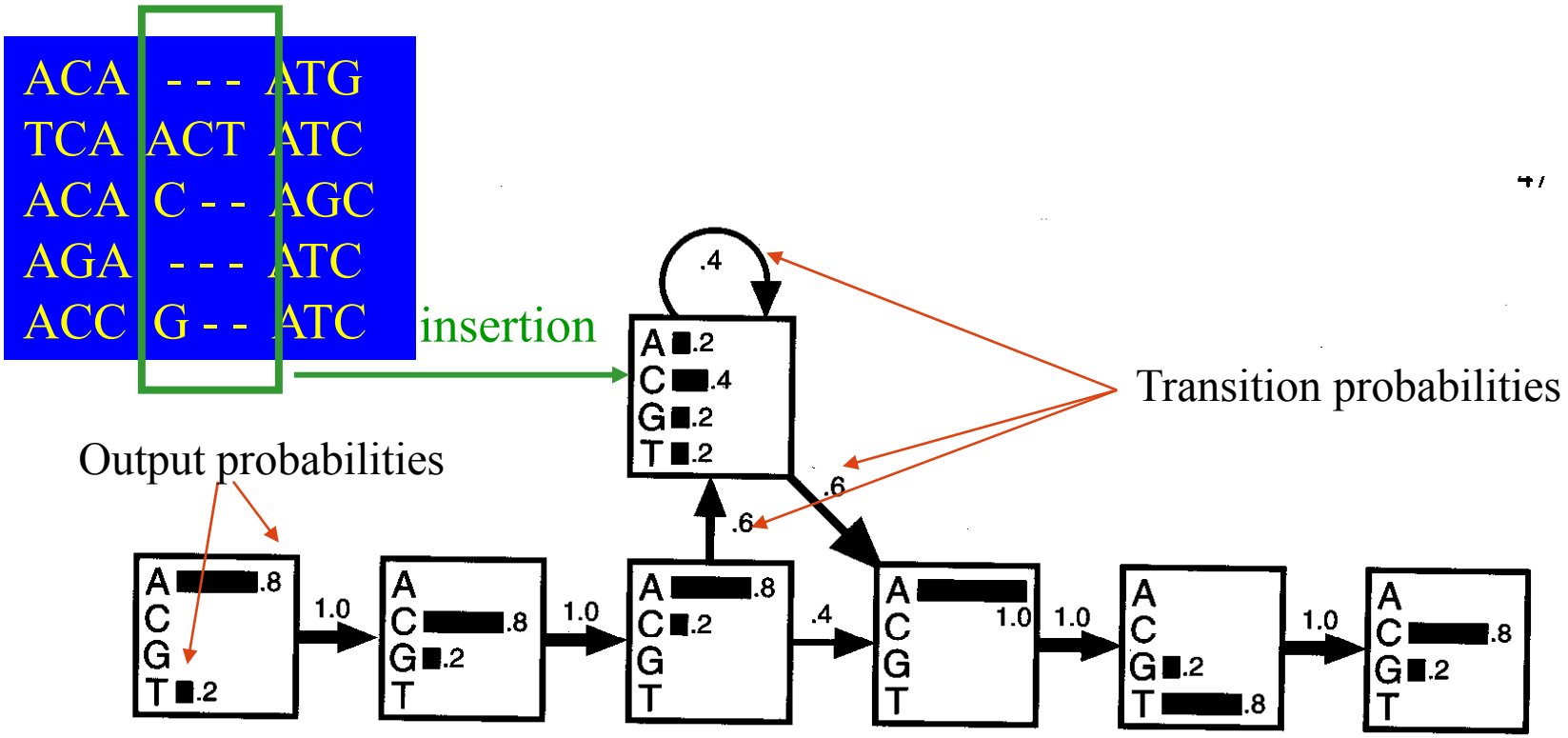
Markov Random Processes

- A random sequence has the Markov property if its dis tribution is determined solely by its current state. Any random process having this property is called a *Mark ov random process*.
- For observable state sequences (state is known from d ata), this leads to a *Markov chain* model.
- For non-observable states, this leads to a *Hidden Mar kov Model* (HMM).

HMMs in Computational Linguistics

- Speech recognition (observed: acoustic signal, hidden: words)
- Handwriting recognition (observed: image, hidden: words)
- Part-of-speech tagging (observed: words, hidden: part-of-speech tags)
- Machine translation (observed: foreign words, hidden: words in target language)

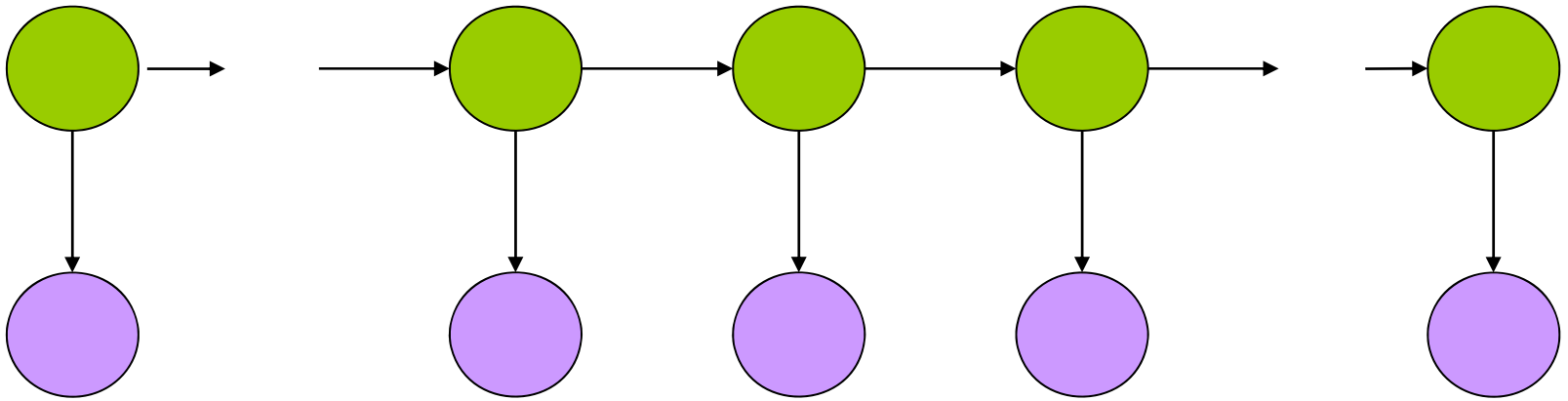
HMMs in Computational Biology



A HMM model for a DNA motif alignments. The transitions are shown with arrows whose thickness indicate their probability. In each state, the histogram shows the probabilities of the four bases.

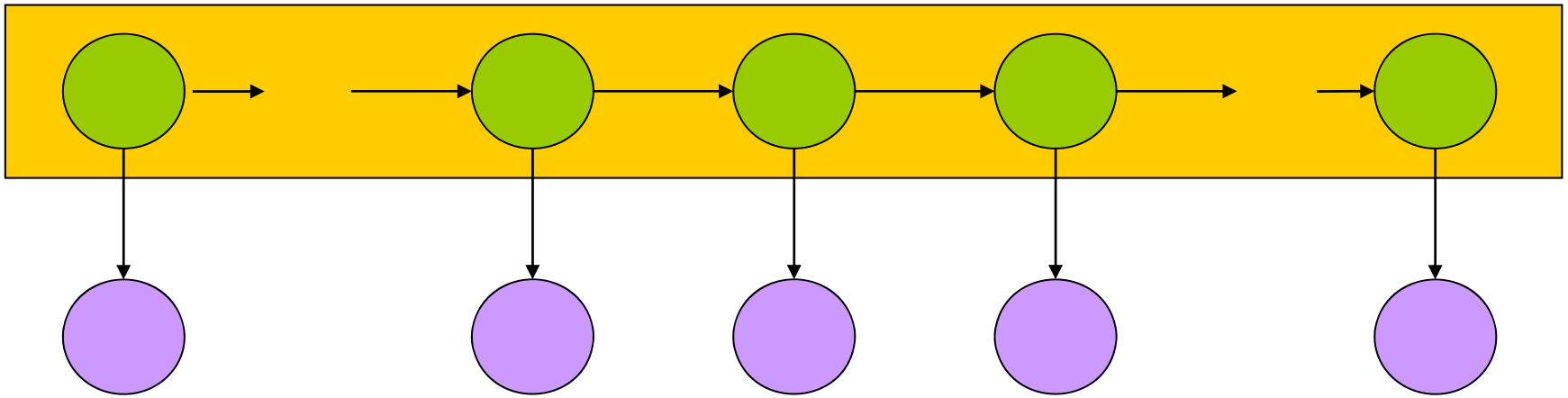
HMM:
A Graphical Description

Formally: What is an HMM?



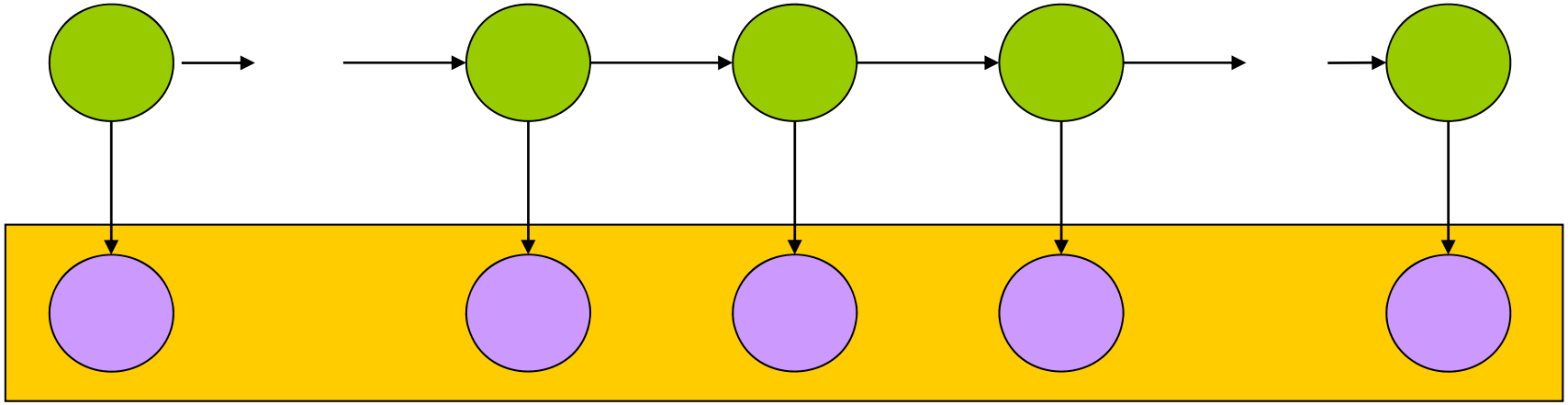
- Graphical model
- Circles indicate states
- Arrows indicate probabilistic dependencies between states

What is an HMM?



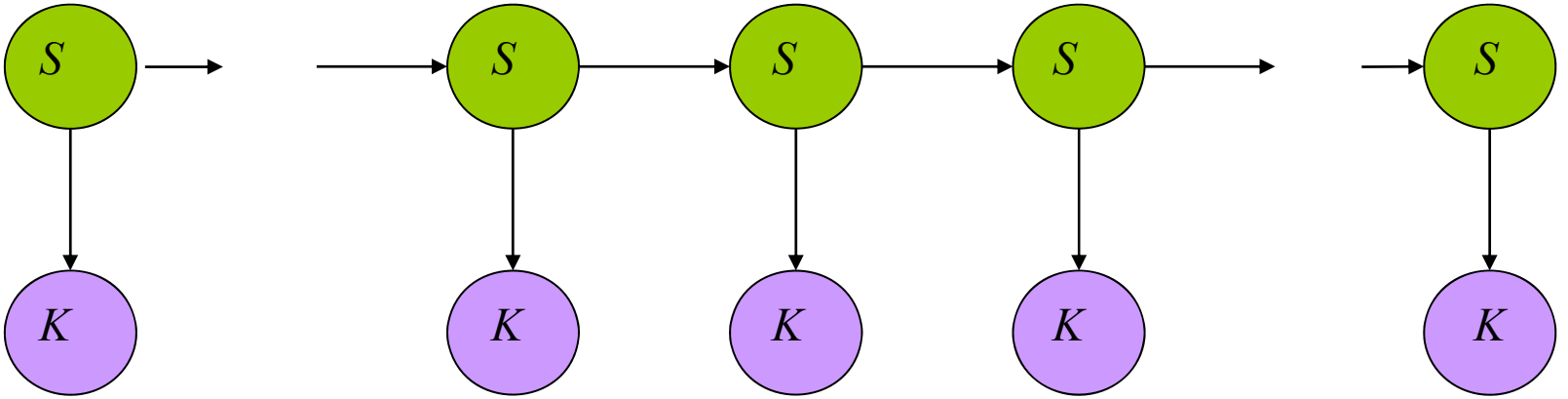
- Green circles are ***hidden states***
- Dependent only on the previous state
- "The past is independent of the future given the present."

What is an HMM?



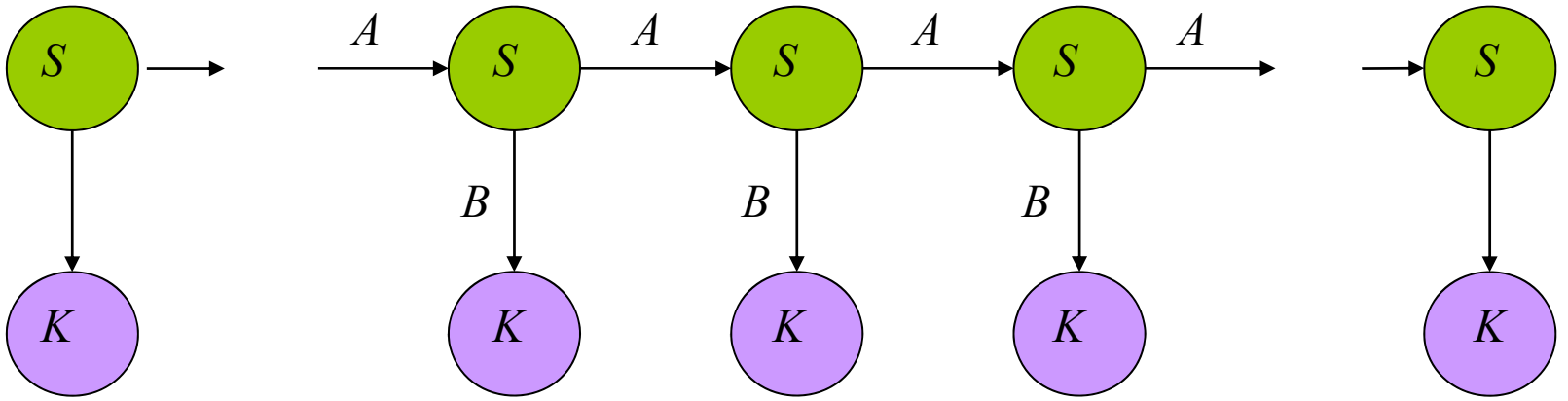
- Purple nodes are ***observed states***
- Dependent only on their corresponding hidden state

HMM Formalism



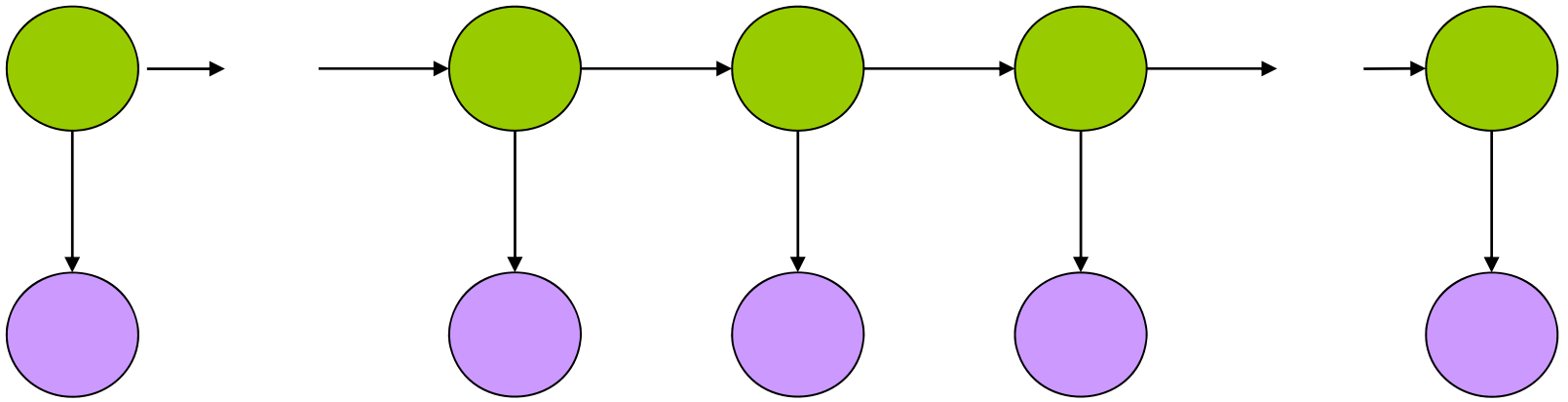
- $\{S, K, P, A, B\}$
- S : $\{s_1 \ldots s_N\}$ are the values for the hidden states
- K : $\{k_1 \ldots k_M\}$ are the values for the observations

HMM Formalism



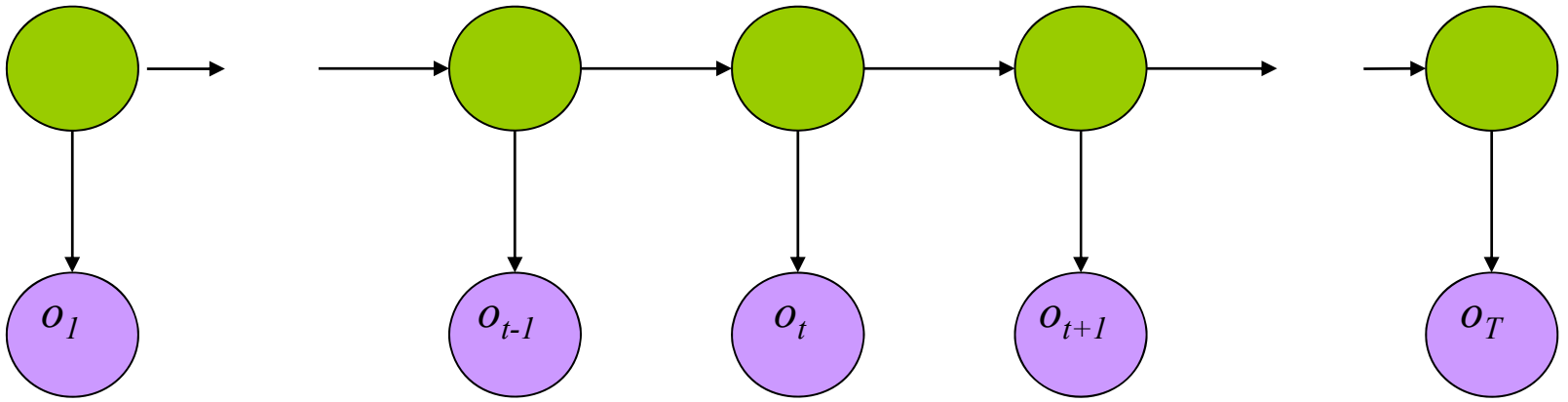
- $\{S, K, P, A, B\}$
- $P = \{p_i\}$ are the initial state probabilities
- $A = \{a_{ij}\}$ are the state transition probabilities
- $B = \{b_{ik}\}$ are the observation state probabilities

Inference in an HMM



- Compute the probability of a given observation sequence (Evaluation)
- Given an observation sequence, compute the most likely hidden state sequence (Decoding)
- Given an observation sequence and set of possible models, find the model most closely fits the data (Learning)

Evaluation

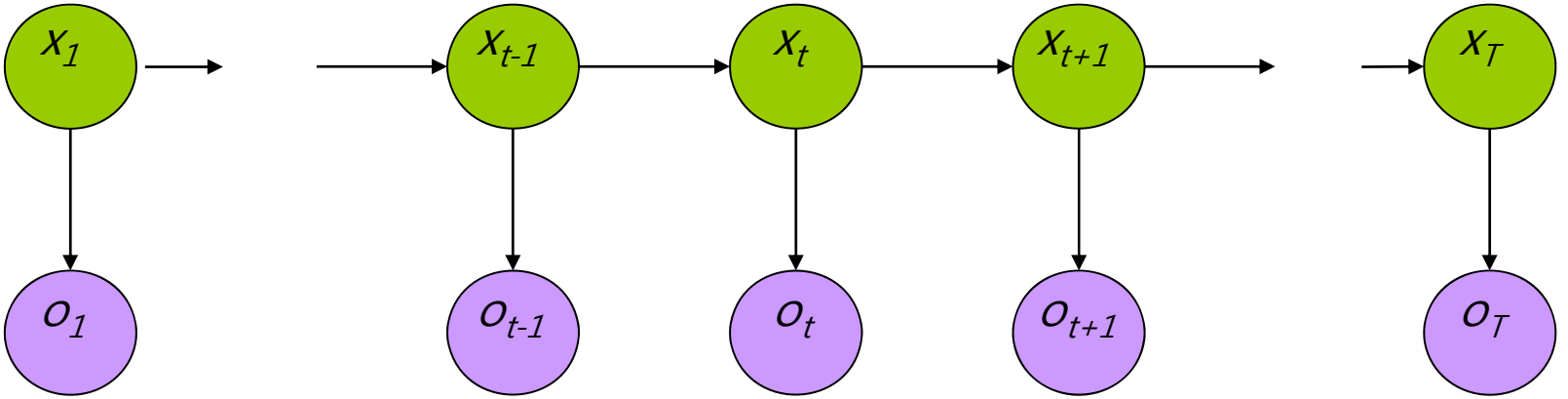


Given an observation sequence and a model,
compute the probability of the observation sequence

$$O = (o_1 \ldots o_T),\ \mu = (A, B, \Pi)$$

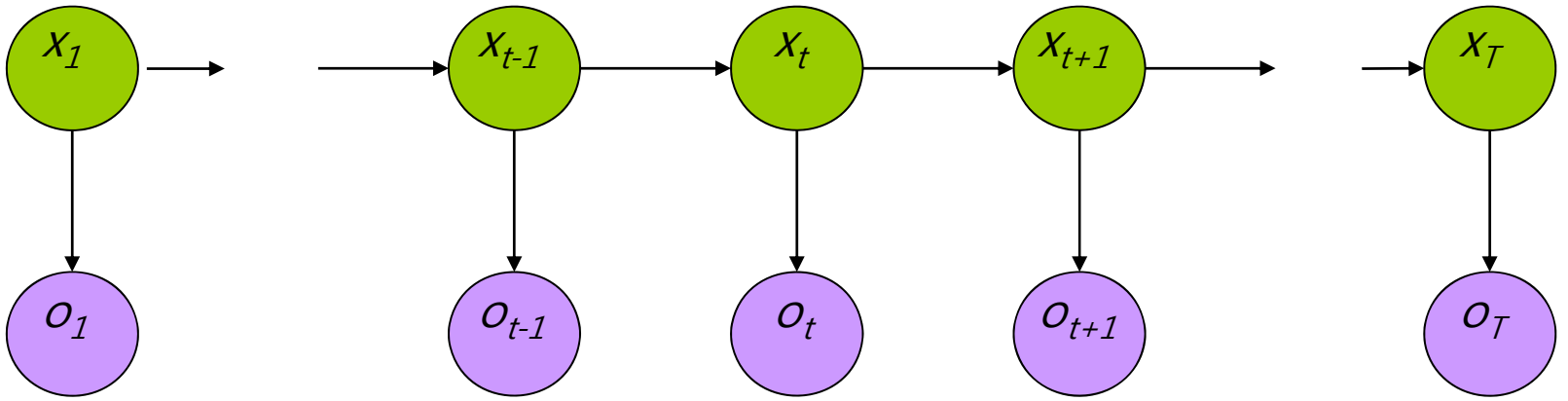
Compute $P(O \mid \mu)$

Evaluation



$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

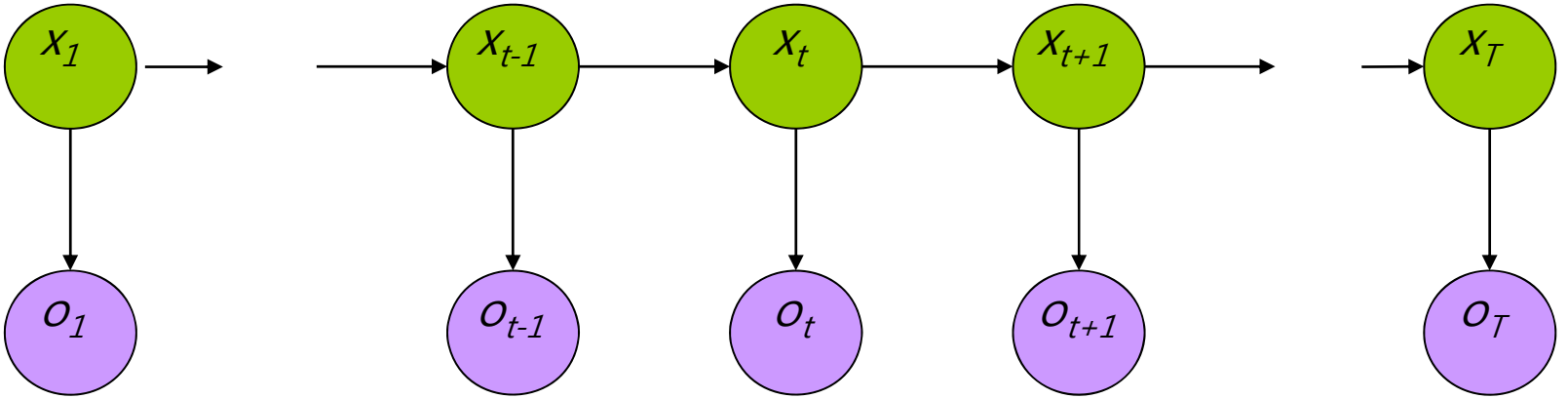
Evaluation



$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

$$P(X \mid \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \ldots a_{x_{T-1} x_T}$$

Evaluation

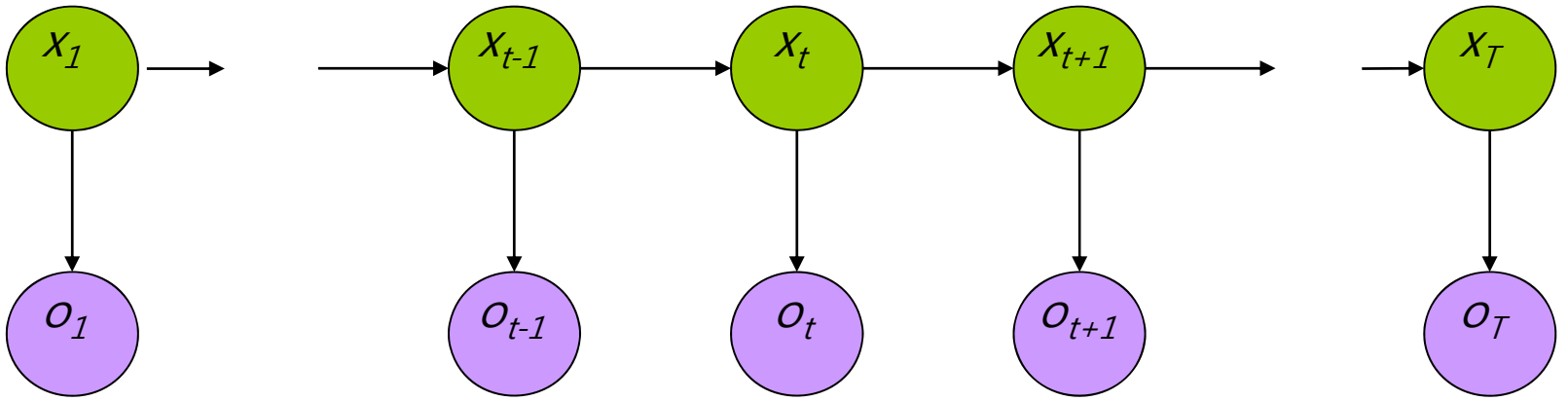


$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

$$P(X \mid \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \ldots a_{x_{T-1} x_T}$$

$$\boxed{P(O, X \mid \mu) = P(O \mid X, \mu) P(X \mid \mu)}$$

Evaluation



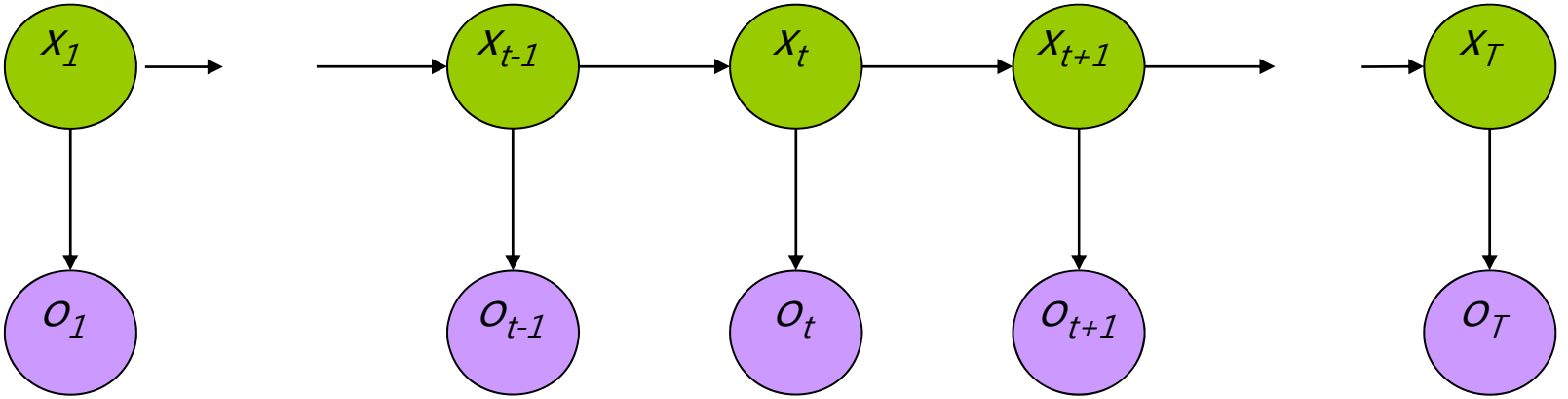
$$P(O \mid X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \ldots b_{x_T o_T}$$

$$P(X \mid \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \ldots a_{x_{T-1} x_T}$$

$$P(O, X \mid \mu) = P(O \mid X, \mu) P(X \mid \mu)$$

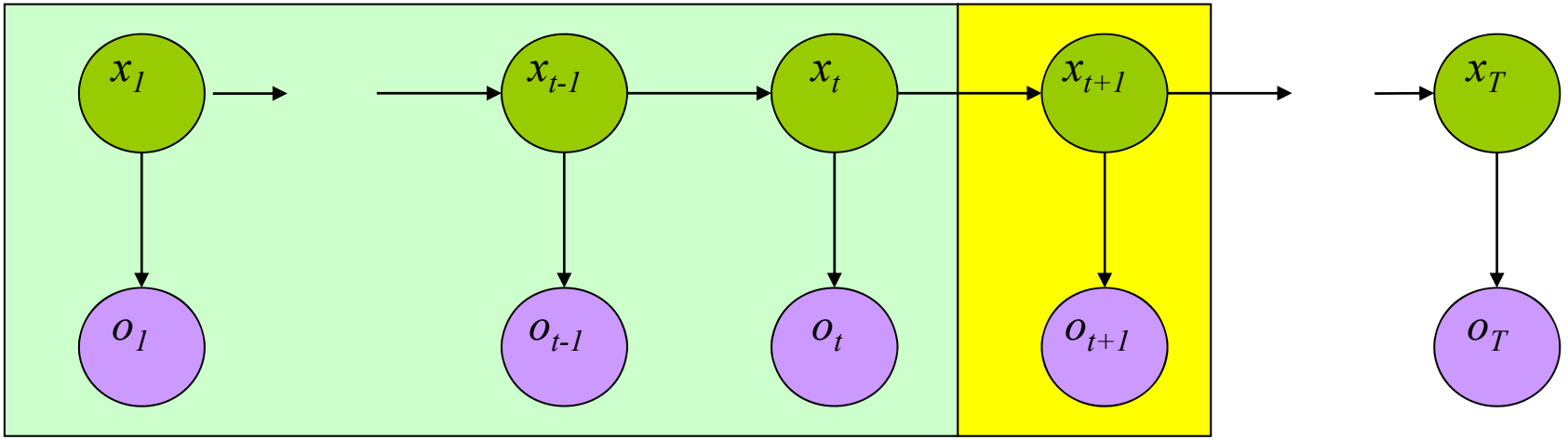
$$P(O \mid \mu) = \sum_X P(O \mid X, \mu) P(X \mid \mu)$$

Evaluation



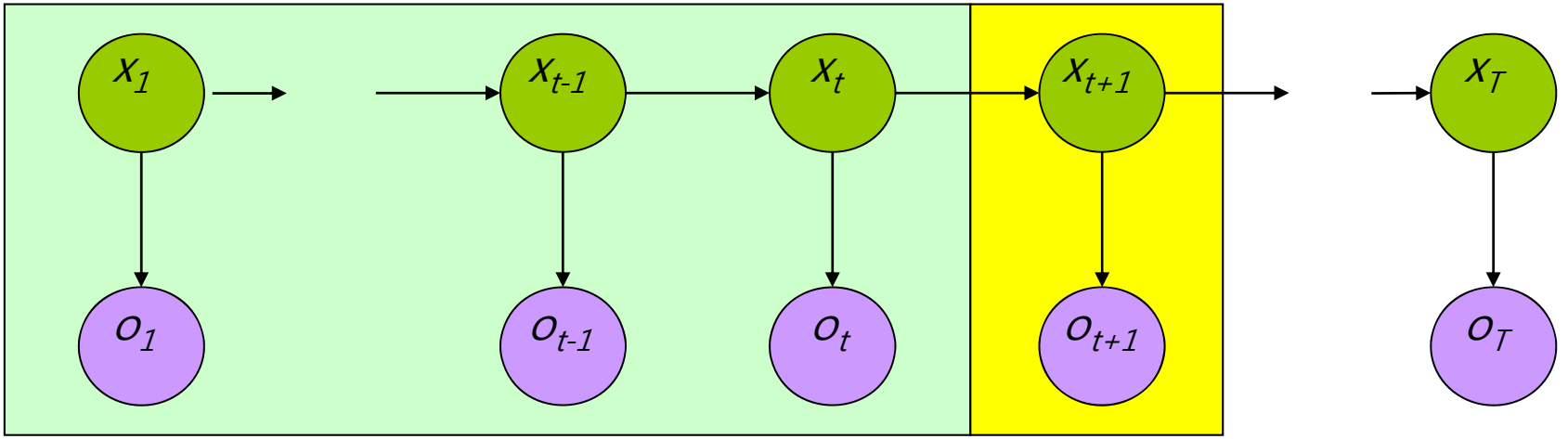
$$P(O \mid \mu) = \sum_{\{x_1 \ldots x_T\}} \pi_{x_1} b_{x_1 o_1} \prod_{t=1}^{T-1} a_{x_t x_{t+1}} b_{x_{t+1} o_{t+1}}$$

Forward Procedure



- Special structure gives us an efficient solution using *dynamic programming*.
- **Intuition**: Probability of the first *t* observations is the same for all possible *t*+1 length state sequences.
- **Define:** $\alpha_i(t) = P(o_1 \ldots o_t, x_t = i \mid \mu)$

Forward Procedure



$$\alpha_j(t+1)$$

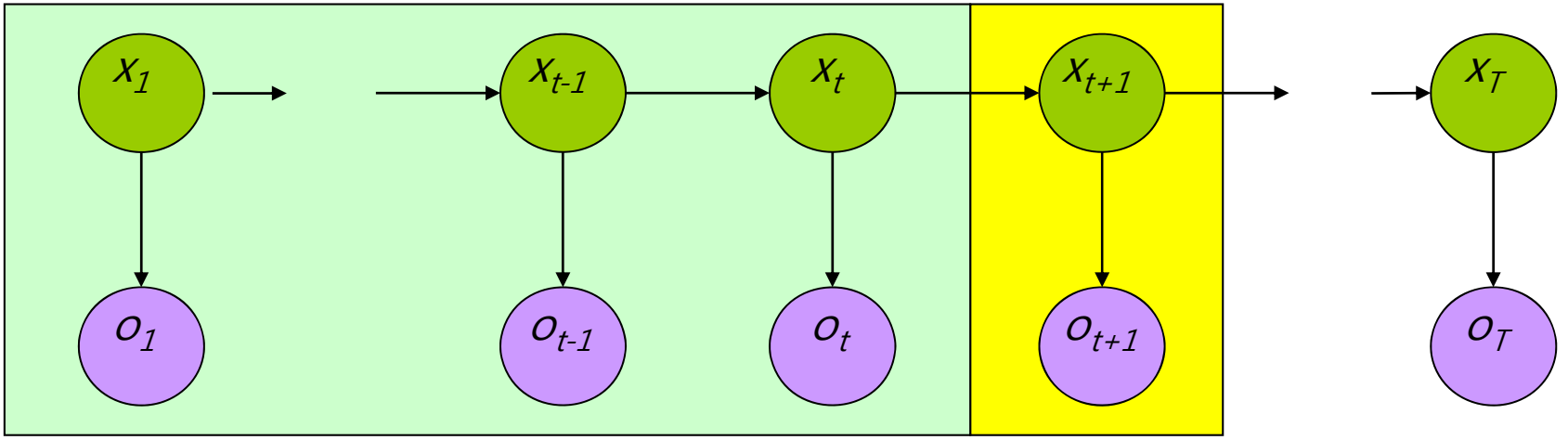
$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



$$\alpha_j(t+1)$$

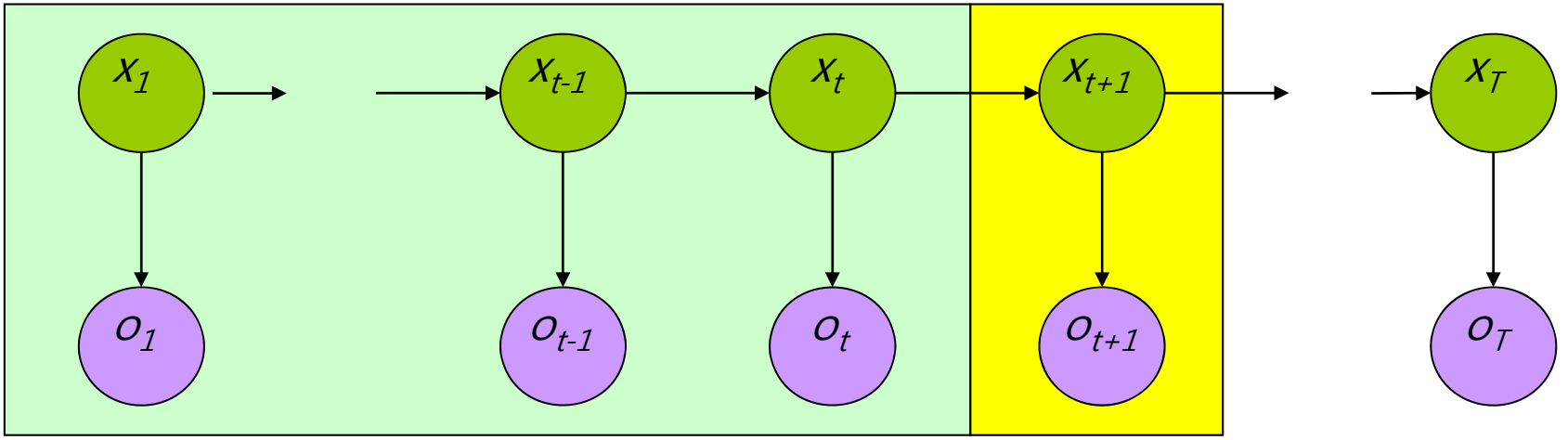
$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



$$\alpha_j(t+1)$$

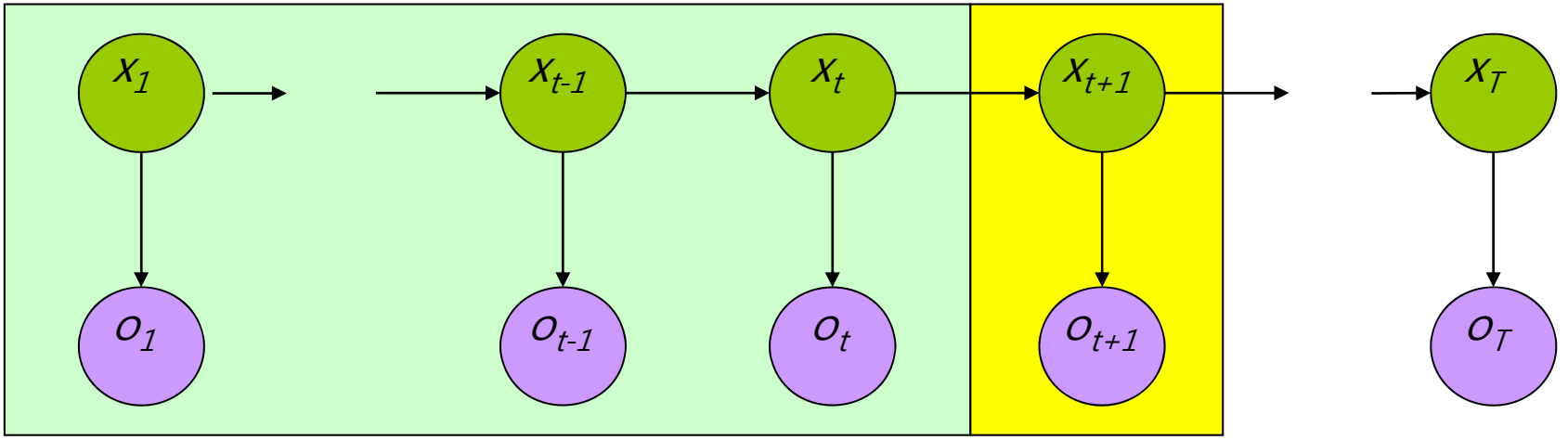
$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



$$\alpha_j(t+1)$$

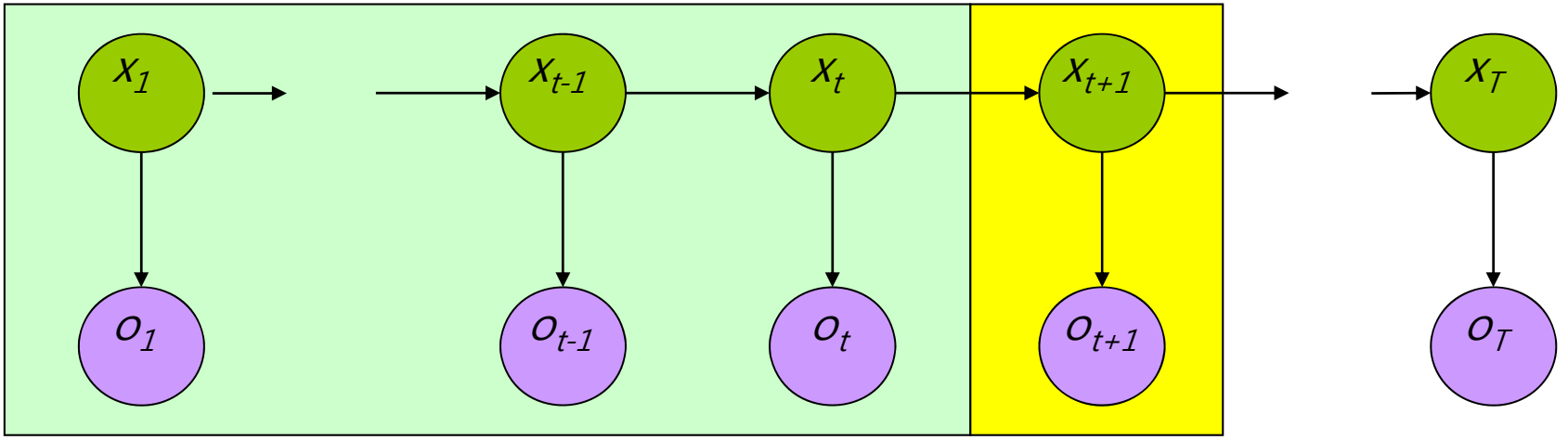
$$= P(o_1 \ldots o_{t+1}, x_{t+1} = j)$$

$$= P(o_1 \ldots o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t \mid x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j) P(x_{t+1} = j)$$

$$= P(o_1 \ldots o_t, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

Forward Procedure



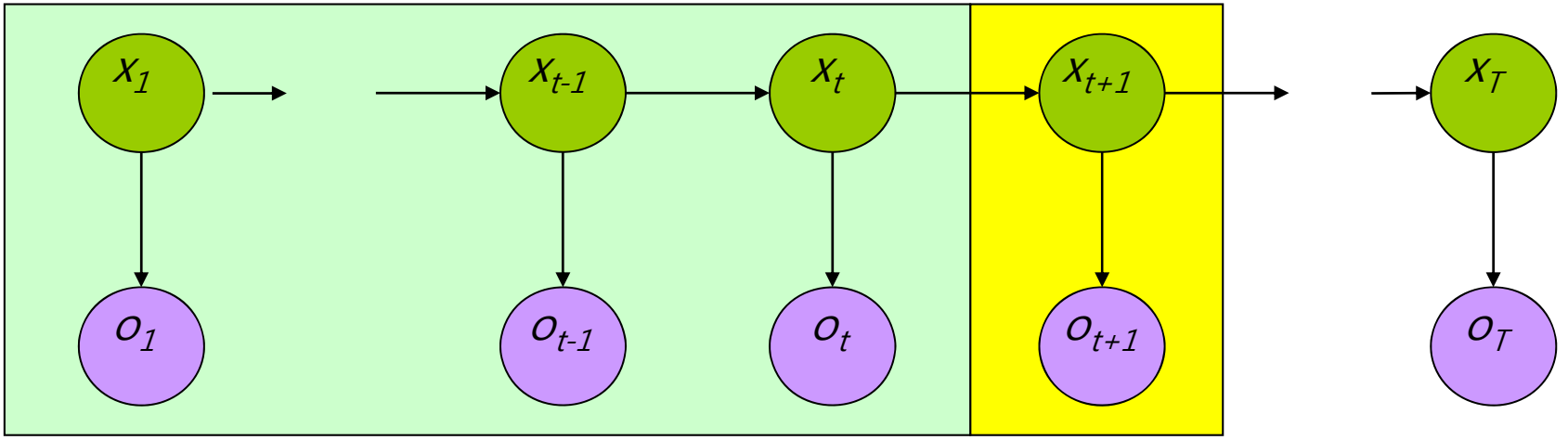
$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Forward Procedure



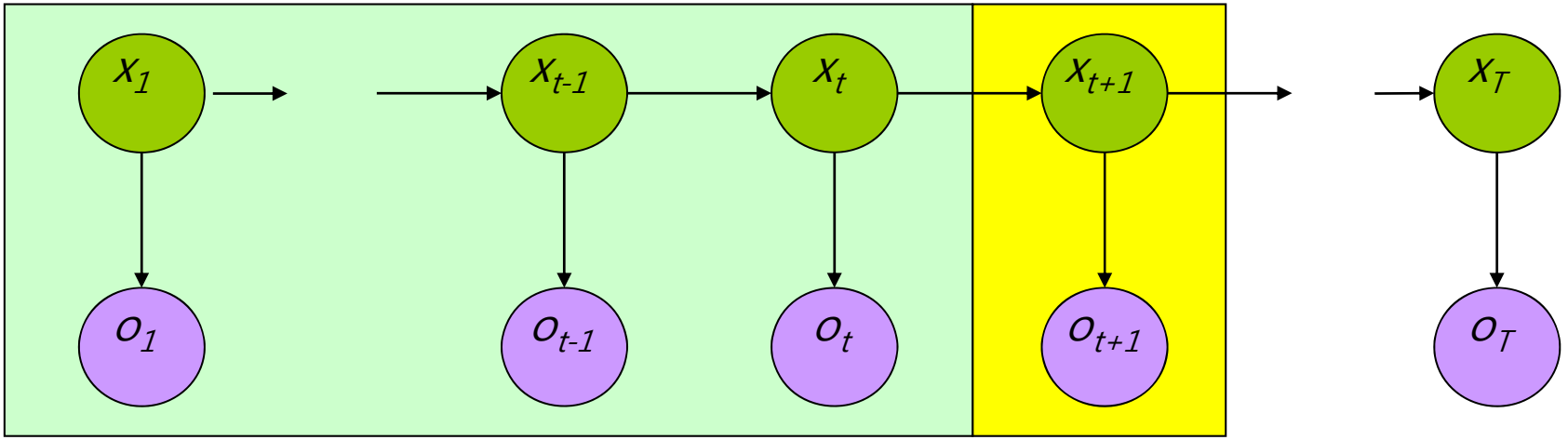
$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Forward Procedure



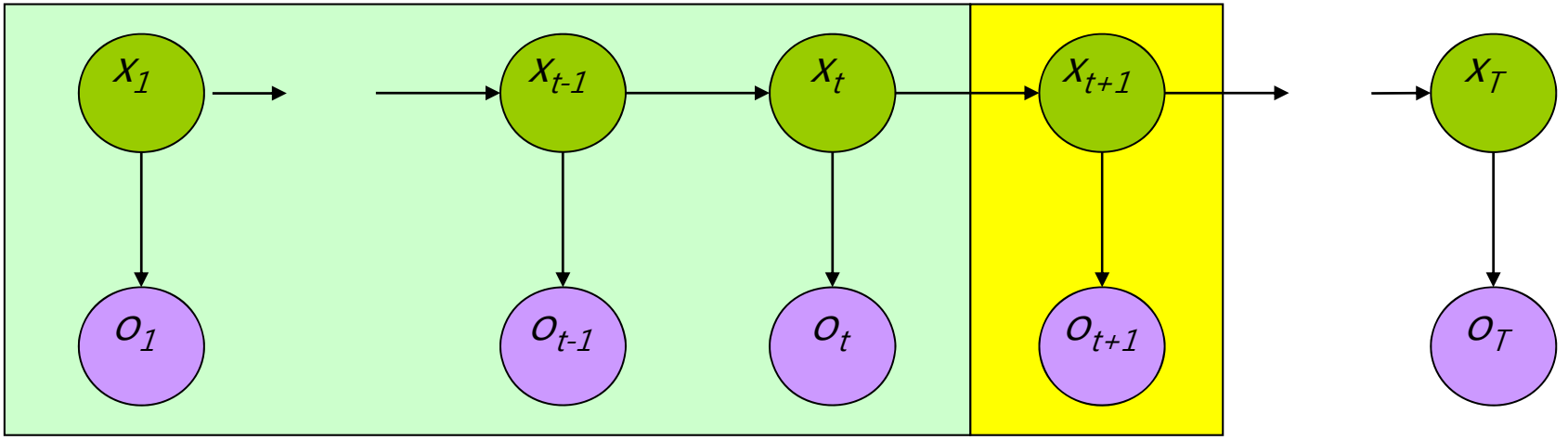
$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Forward Procedure



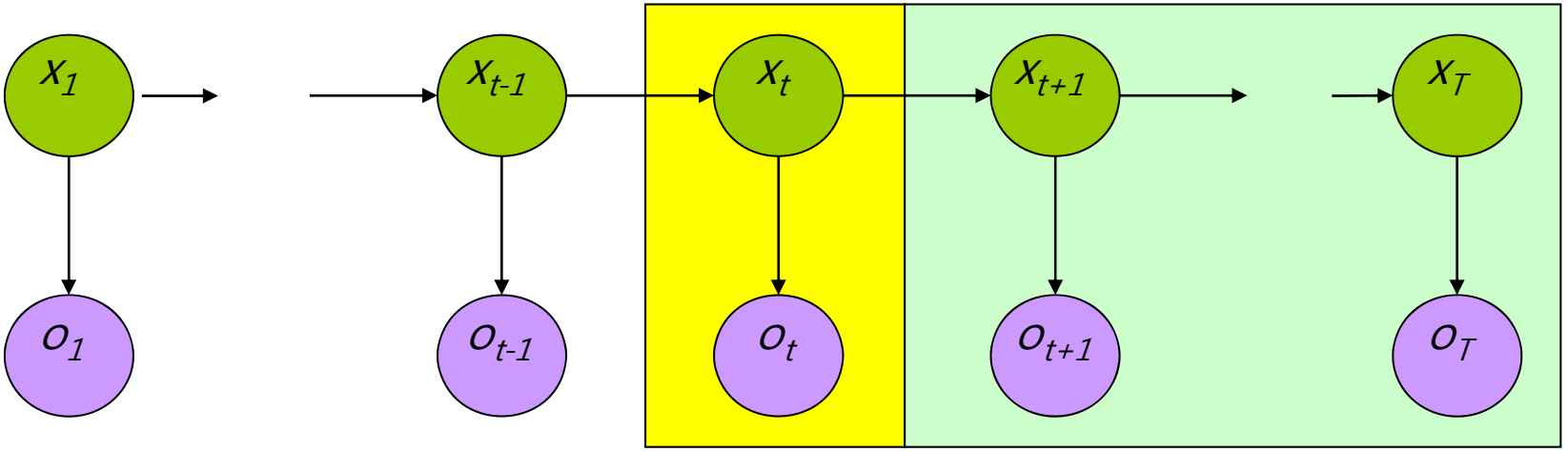
$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_{t+1} = j \mid x_t = i) P(x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} P(o_1 \ldots o_t, x_t = i) P(x_{t+1} = j \mid x_t = i) P(o_{t+1} \mid x_{t+1} = j)$$

$$= \sum_{i=1\ldots N} \alpha_i(t) a_{ij} b_{jo_{t+1}}$$

Backward Procedure



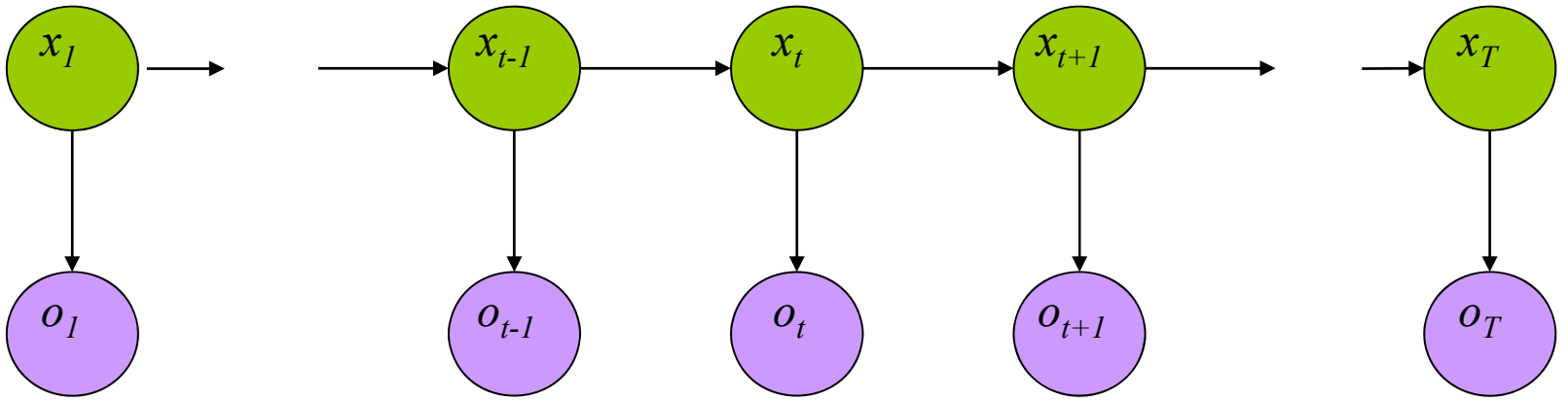
$$\beta_i(T+1) = 1$$

$$\beta_i(t) = P(o_t \ldots o_T \mid x_t = i)$$

$$\beta_i(t) = \sum_{j=1 \ldots N} a_{ij} b_{io_t} \beta_j(t+1)$$

Probability of the rest of the states given the first state

Evaluation: Three Solutions

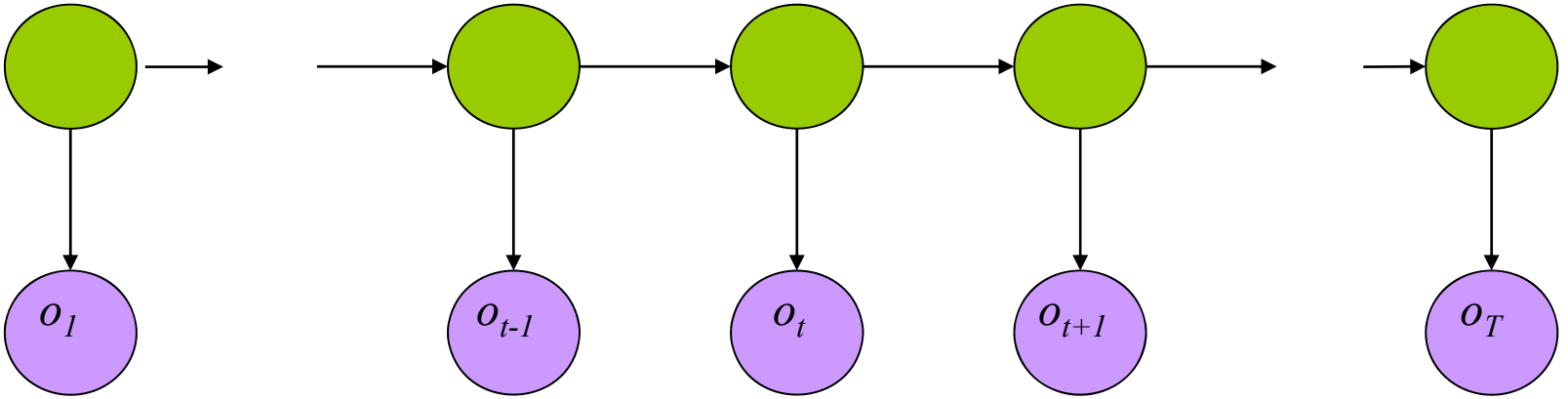


$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(T)$$ **Forward Procedure**

$$P(O \mid \mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$$ **Backward Procedure**

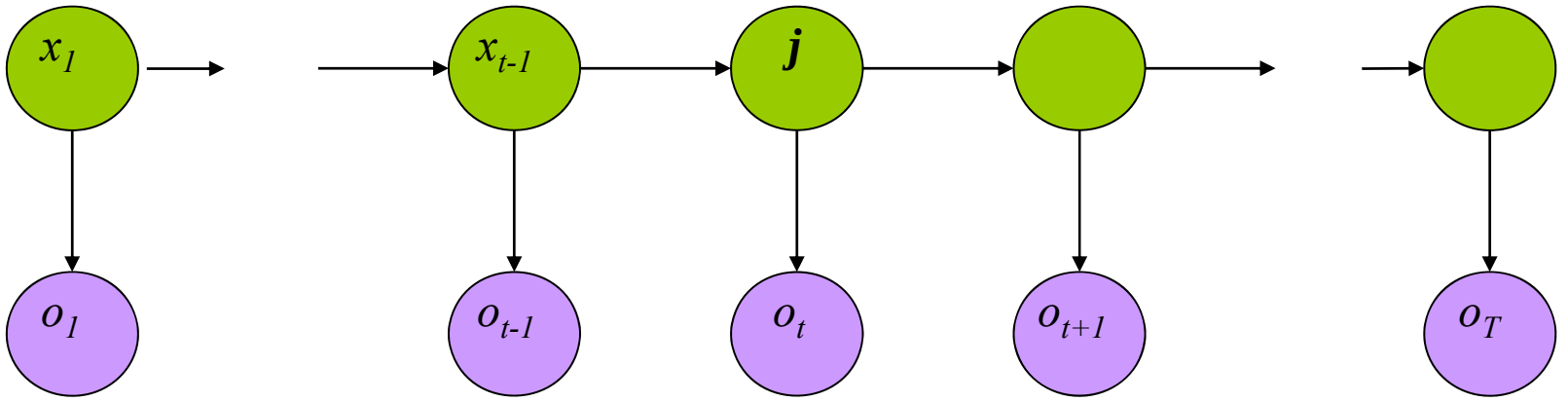
$$P(O \mid \mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$ **Combination**

Decoding: Best State Sequence



- Find the state sequence that best explains the observations
- **Viterbi** algorithm
- $\arg\max_X P(X \mid O)$

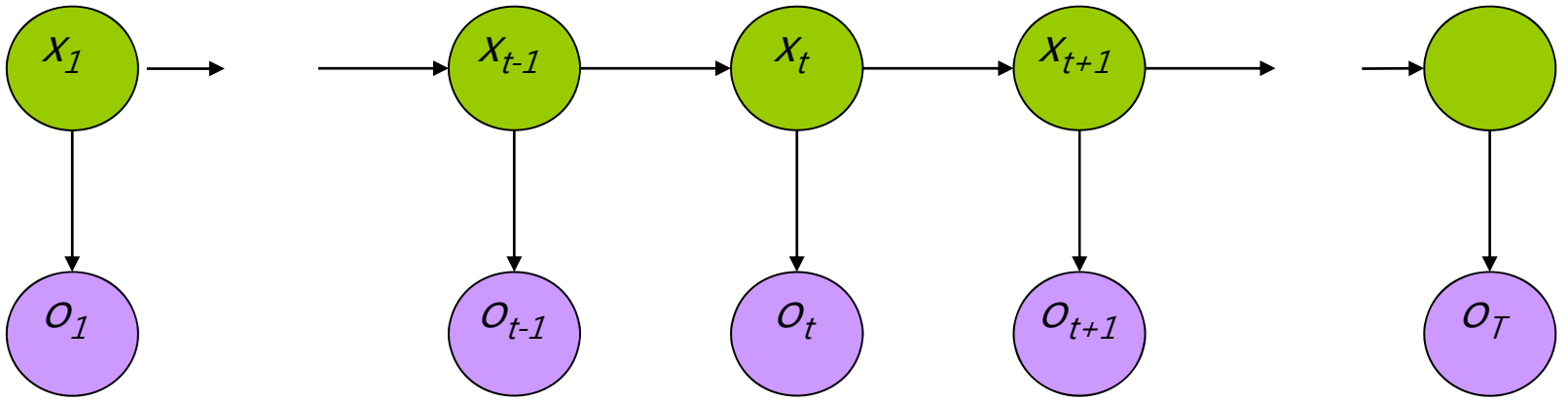
Viterbi Algorithm



$$\delta_j(t) = \max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_{t-1}, o_1 \ldots o_{t-1}, x_t = j, o_t)$$

The state sequence which maximizes the probability of seeing the observations to time *t-1*, landing in state *j*, and seeing the observation at time *t*

Viterbi Algorithm



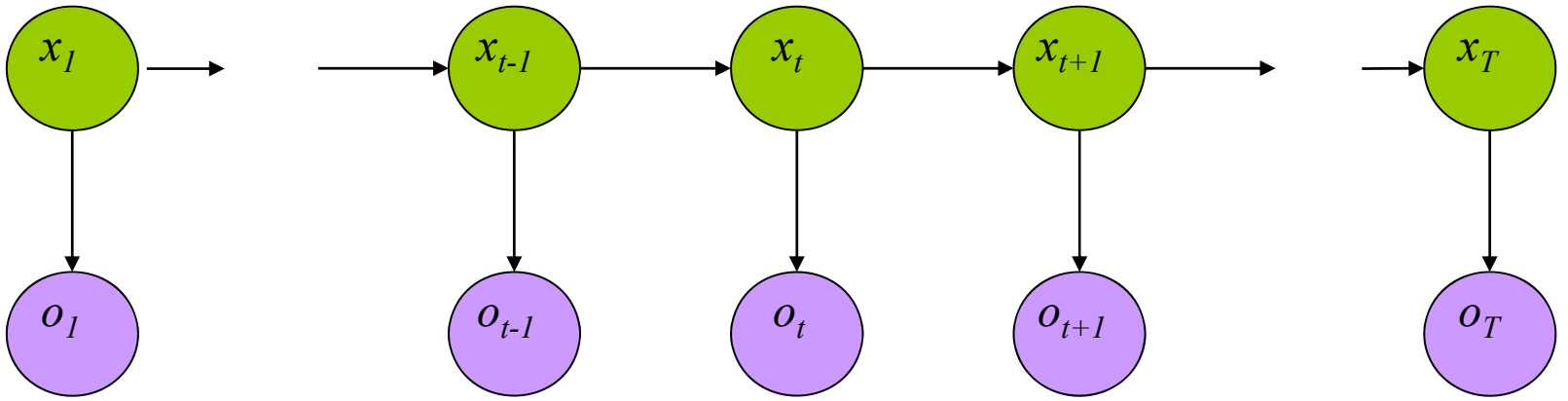
$$\delta_j(t) = \max_{x_1 \ldots x_{t-1}} P(x_1 \ldots x_{t-1}, o_1 \ldots o_{t-1}, x_t = j, o_t)$$

$$\delta_j(t+1) = \max_i \delta_i(t) a_{ij} b_{jo_{t+1}}$$

$$\psi_j(t+1) = \arg\max_i \delta_i(t) a_{ij} b_{jo_{t+1}}$$

Recursive Computation

Viterbi Algorithm



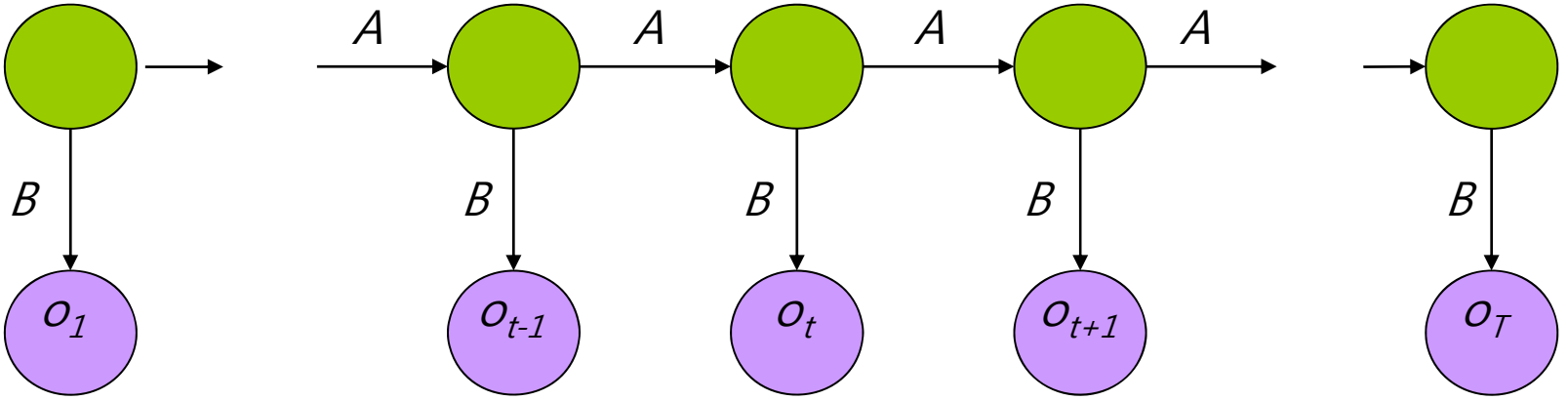
$$\hat{X}_T = \arg\max_i \delta_i(T)$$

$$\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$

$$P(\hat{X}) = \arg\max_i \delta_i(T)$$

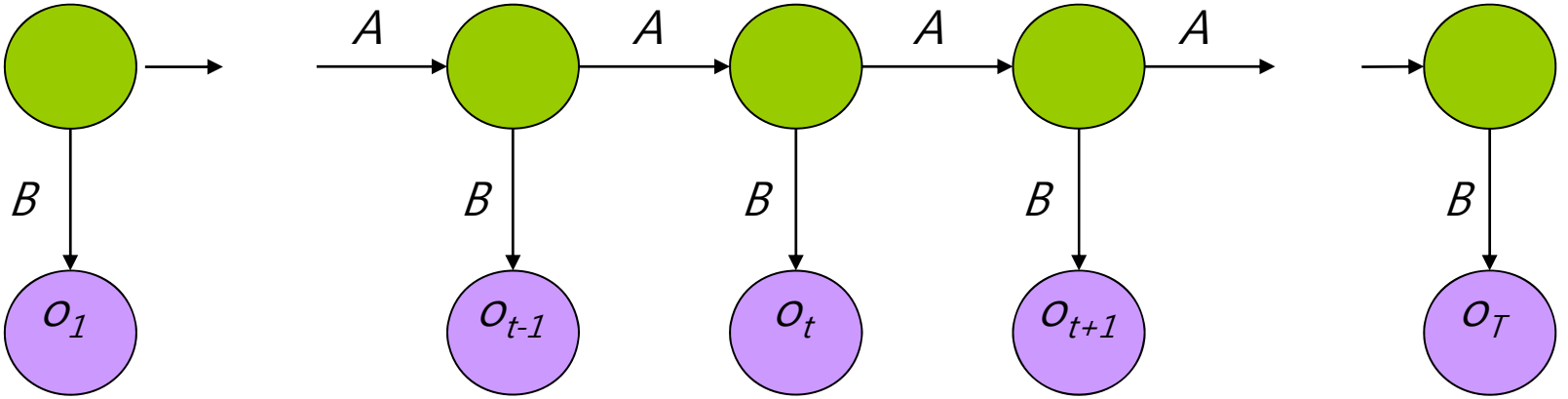
Compute the most likely state sequence by working backwards

Learning: Parameter Estimation



- Given an observation sequence, find the model that is most likely to produce that sequence.
- No analytic method
- Given a model and observation sequence, update the model parameters to better fit the observations.

Parameter Estimation



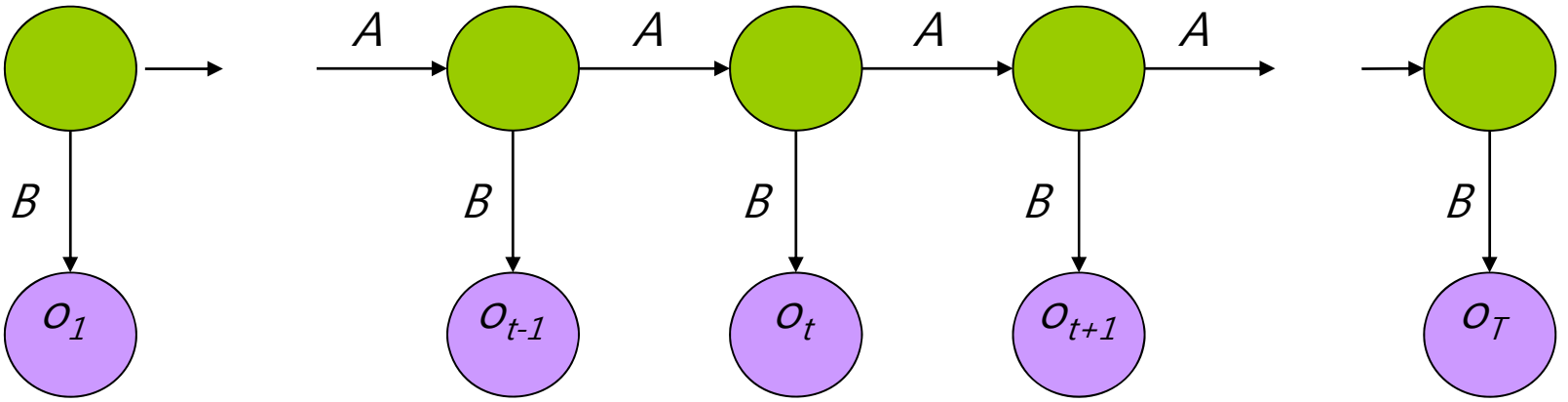
$$p_t(i,j) = \frac{\alpha_i(t) a_{ij} b_{jo_{t+1}} \beta_j(t+1)}{\sum_{m=1\ldots N} \alpha_m(t)\beta_m(t)}$$

Probability of traversing an arc

$$\gamma_i(t) = \sum_{j=1\ldots N} p_t(i,j)$$

Probability of being in state *i*

Parameter Estimation



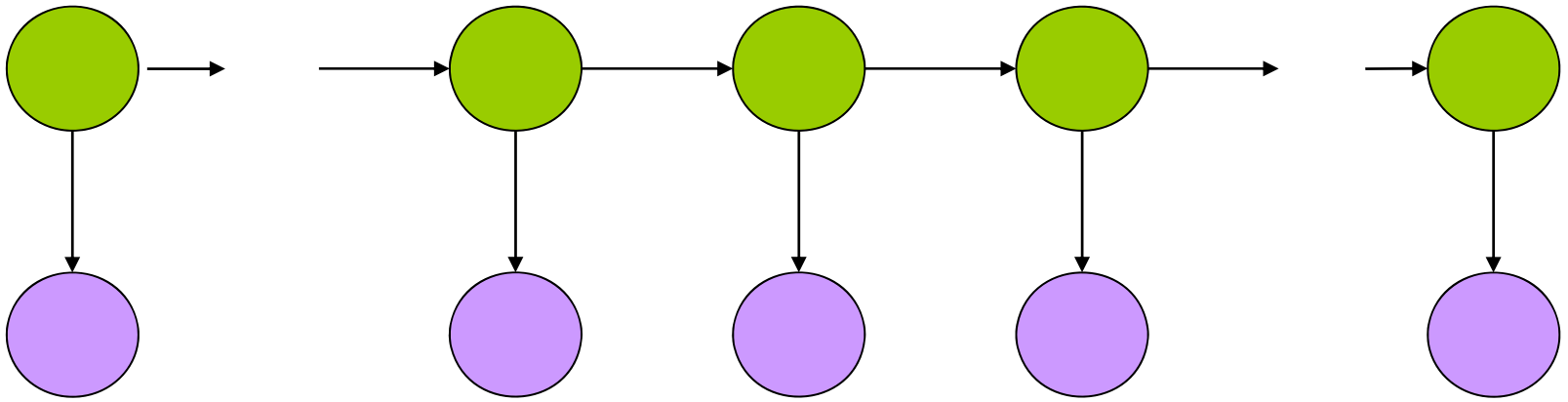
$$\hat{\pi}_i = \gamma_i(1)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)}$$

$$\hat{b}_{ik} = \frac{\sum_{\{t:o_t=k\}} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_i(t)}$$

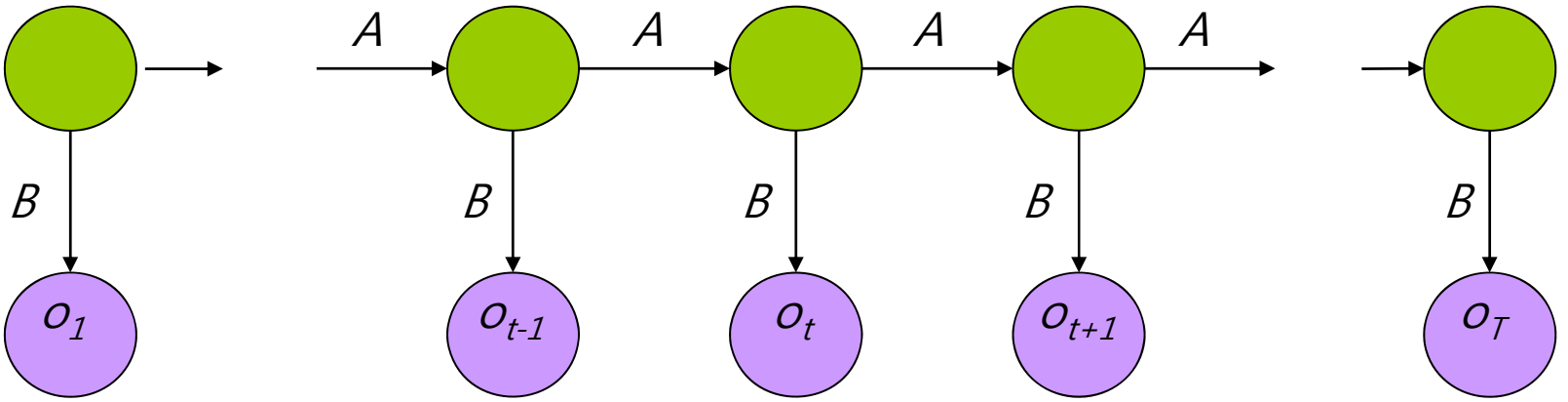
Now we can compute the new estimates of the model parameters.

HMM Applications



- Generating parameters for n-gram models
- Tagging speech
- Speech recognition

The Most Important Thing



We can use the special structure of this model to do a lot of neat math and solve problems that are otherwise not solvable.

HMM:
A More Formal Description

Specification of an HMM

An HMM is specified as $\lambda = (A,B,\pi)$

A : state transition probability matrix

$a_{ij} = P(q_{t+1} = j \mid q_t = i)$

B : observation probability distribution

$b_j(k) = P(o_t = k \mid q_t = j) \quad i \le k \le M$

π : initial state distribution

$Q = (q_1, q_2, \ldots, q_T)$: sequence of states

$O = (o_1, o_2, \ldots, o_T)$: sequence of observed symbols

N : number of states

M : number of symbols (observables)

Central Problems in HMM

- Problem 1: Evaluation
  - Compute $P(O|\lambda)$
    - Probability of occurrence of a particular observation sequence, $O = (o_1,\ldots,o_k)$, given the model
    - Complicated due to many possible hidden states Q
    - Sequence classification
- Problem 2: Decoding
  - Find state sequence Q, such that $P(O,Q|\lambda)$ is maximal
    - Optimal state sequence to produce a given sequence of observations, $O = (o_1,\ldots,o_k)$, given model
    - Optimality criterion
    - Pattern recognition
- Problem 3: Learning
  - Find λ, such that $P(O|\lambda)$ is maximal
    - Determine optimum model, given a training set of observation sequences

Problem 1: Naïve Solution

- State sequence $q = (q_1,\ldots q_T)$
- Assume independent observations:

$$P(O \mid q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) = b_{q_1}(o_1) b_{q_2}(o_2) \ldots b_{q_T}(o_T)$$

Observations are mutually independent, given the hidden states. (Joint distribution of independent variables factorises into marginal distributions of the independent variables.)

- Observe that:

$$P(q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T}$$

- Thus:

$$P(O \mid \lambda) = \sum_{q} P(O \mid q, \lambda) P(q \mid \lambda)$$

- The sum is over all state paths q
- There are N^T states paths,
- Each path costs $O(T)$ calculations, leading to $O(TN^T)$

Problem 1: Naïve Solution (cont.)

- Finally get:

$$P(O|\lambda) = \sum_q P(O|q,\lambda)P(q|\lambda)$$

$$= \sum_q b_{q_1}(o_1)b_{q_2}(o_2)...b_{q_T}(o_T)\pi_{q_1}a_{q_1q_2}a_{q_2q_3}...a_{q_{T-1}q_T}$$

$$= \sum_q \pi_{q_1}b_{q_1}(o_1)a_{q_1q_2}b_{q_2}(o_2)a_{q_2q_3}...a_{q_{T-1}q_T}b_{q_T}(o_T)$$

$$= \sum_q \pi_{q_1}b_{q_1}(o_1)\prod_{t=2}^{T} a_{q_{t-1}q_t}b_{q_t}(o_t)$$

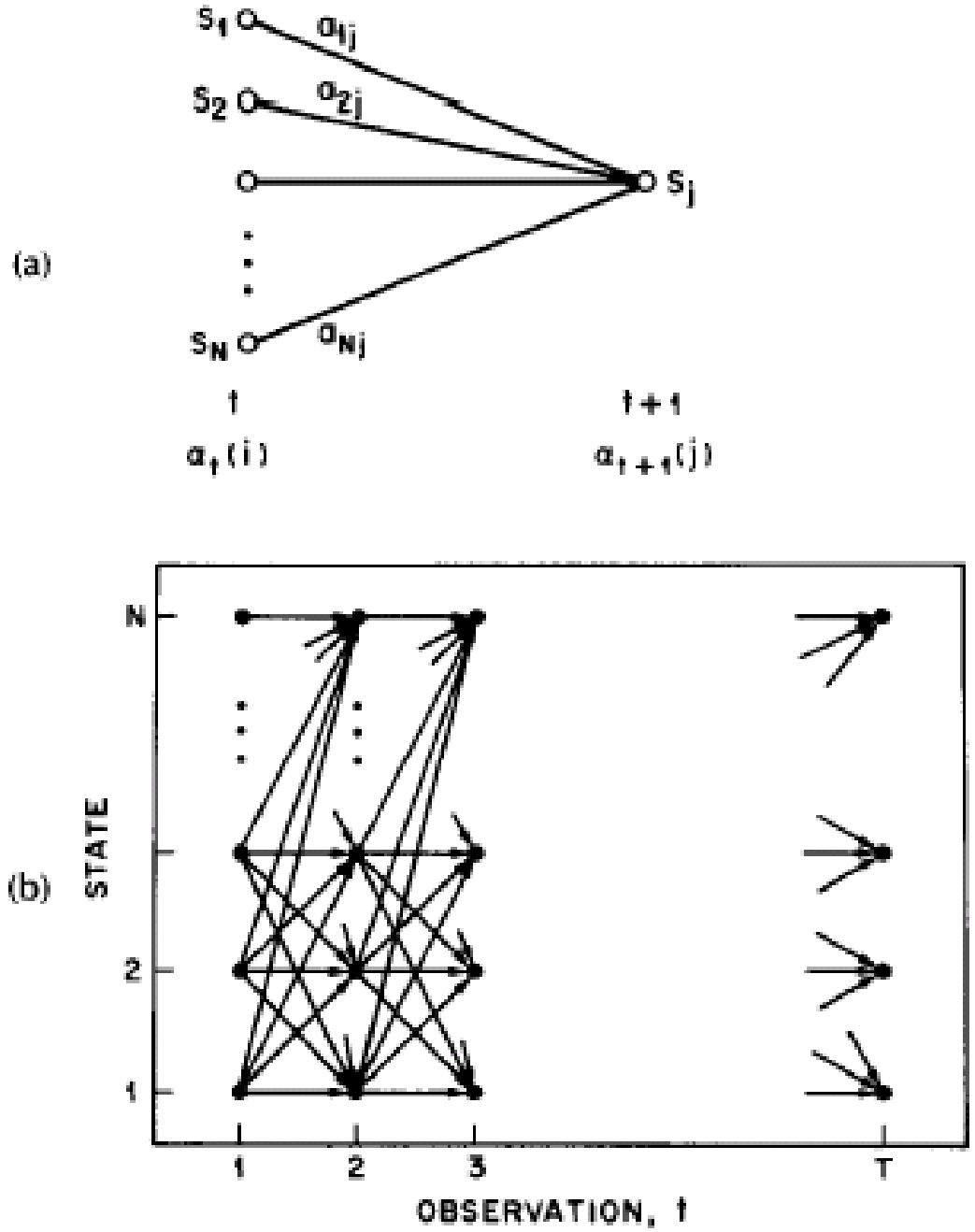
Problem 1: Efficient Solution

Forward algorithm:

- Define auxiliary forward variable α:

$$\alpha_t(i) = P(o_1,\ldots,o_t \mid q_t = i,\lambda)$$

$\alpha_t(i)$ is the probability of observing a partial sequence of observables $o_1,\ldots o_t$ such that at time t, state $q_t=i$


(a)
S_1
S_2
S_N
a_{1j}
a_{2j}
a_{Nj}
S_j
t
t+1
$\alpha_t(i)$
$\alpha_{t+1}(j)$
(b)
STATE
N
2
1
1
2
3
T
OBSERVATION, t

Problem 1: Efficient solution

- Recursive algorithm:
  - Initialise:

$$\alpha_1(i) = \pi_i b_i(o_1)$$

(Partial obs seq to t AND state i at t) x (transition to j at $t+1$) x (sensor)

  - Calculate:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(o_{t+1})$$

Sum, as can reach j from any preceding state

α incorporates partial obs seq to t

  - Obtain:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Sum of different ways of getting obs seq

Complexity is $O(N^2 T)$

Problem 1: Alternative Solution

Backward algorithm:

- Define auxiliary forward variable β:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T \mid q_t = i, \lambda)$$

$\beta_t(i)$ is the probability of observing a sequence of observables $o_{t+1}, \ldots, o_T$ given state $q_t = i$ at time t, and λ

Problem 1: Alternative Solution

- Recursive algorithm:
  - Initialise:

$$\beta_T(j) = 1$$

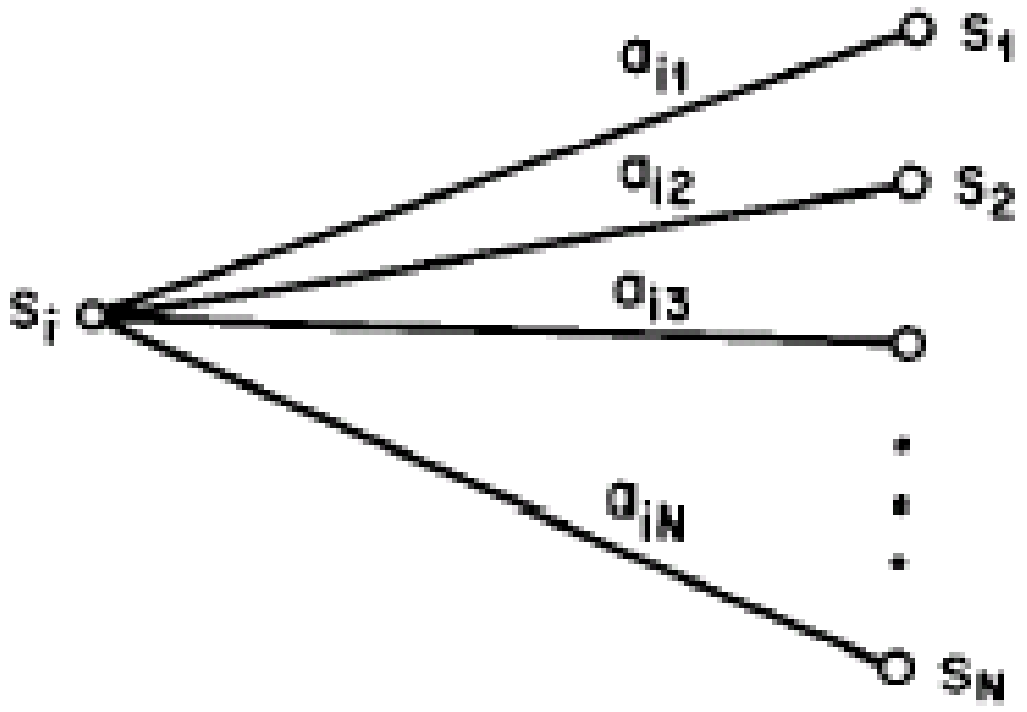
  - Calculate:

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1})$$

  - Terminate:

$$p(O \mid \lambda) = \sum_{i=1}^{N} \beta_1(i) \qquad t = T-1,\ldots,1$$

Complexity is $O(N^2T)$



t

$\beta_t(i)$

$t+1$

$\beta_{t+1}(j)$

Problem 2: Decoding

- Choose state sequence to maximise probability of observation sequence
- Viterbi algorithm - inductive algorithm that keeps the best state sequence at each instance

Problem 2: Decoding

Viterbi algorithm:

- Find state sequence to maximize $P(O,Q|\lambda)$:

$$P(q_1, q_2, ..., q_T \mid O, \lambda)$$

- Define auxiliary variable δ:

$$\delta_t(i) = \max_q P(q_1, q_2, ..., q_t = i, o_1, o_2, ..., o_t \mid \lambda)$$

$\delta_t(i)$ is the probability of the most probable path ending in state $q_t=i$

Problem 2: Decoding

- Recurrent property:

To get state seq, need to keep track of argument to maximise this, for each t and j. Done via the array $\psi_t(j)$.

$$\delta_{t+1}(j) = \max_i(\delta_t(i)a_{ij})b_j(o_{t+1})$$

- Algorithm:
  1. Initialize:

$$\delta_1(i) = \pi_i b_i(o_1) \qquad 1 \le i \le N$$

$$\psi_1(i) = 0$$

Problem 2: Decoding

2. Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} (\delta_{t-1}(i) a_{ij}) b_j(o_t)$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} (\delta_{t-1}(i) a_{ij}) \qquad 2 \le t \le T, 1 \le j \le N$$

3. Terminate:

$$P^* = \max_{1 \le i \le N} \delta_T(i)$$

*P** gives the state-optimized probability

$$q_T^* = \arg\max_{1 \le i \le N} \delta_T(i)$$

*Q** is the optimal state sequence
(*Q** = {q1*,q2*,…,qT*})

Problem 2: Decoding

4. Backtrack state sequence:

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t+T-1, T-2, \ldots, 1$$

$O(N^2T)$ time complexity

Problem 3: Learning

- Training HMM to encode obs seq such that HMM should identify a similar obs seq in future
- Find $\lambda=(A,B,\pi)$, maximising $P(O|\lambda)$
- General algorithm:
  - Initialise: λ_0
  - Compute new model λ, using λ_0 and observed sequence O
  - Then $\lambda_o \leftarrow \lambda$
  - Repeat steps 2 and 3 until:

$$\log P(O \mid \lambda) - \log P(O \mid \lambda_0) < d$$

Problem 3: Learning

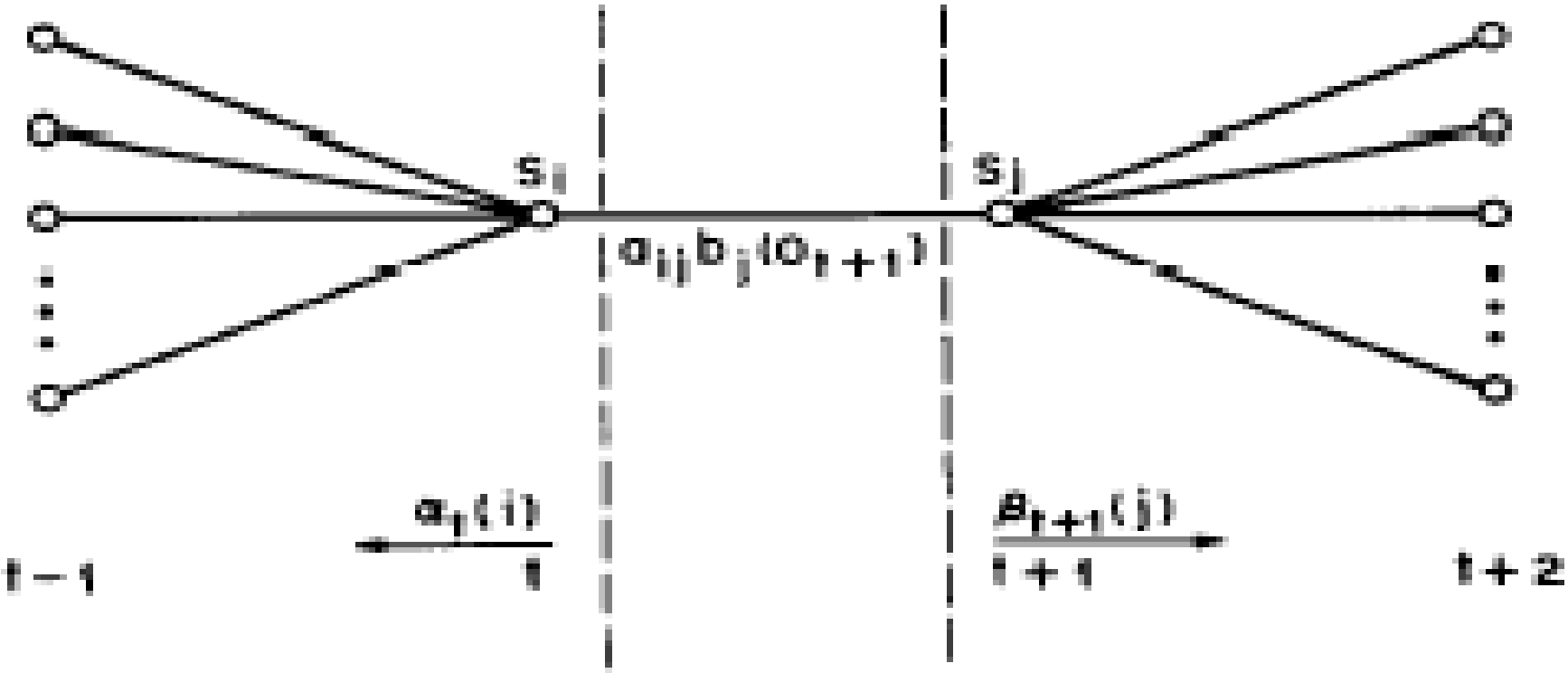
Step 1 of Baum-Welch algorithm:

- Let $\xi(i,j)$ be a probability of being in state i at time t and at state j at time $t+1$, given λ and O seq

$$\xi(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$$

Problem 3: Learning



Operations required for the computation of the joint event that t he system is in state s_i and time t and state s_j at time t+1

Problem 3: Learning

- Let $\gamma_t(i)$ be a probability of being in state i at time t, given O

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

$\sum_{t=1}^{T-1} \gamma_t(i)$ - expected no. of transitions from state i

$\sum_{t=1}^{T-1} \xi_t(i,j)$ - expected no. of transitions $i \to j$

Problem 3: Learning

Step 2 of Baum-Welch algorithm:

- $\hat{\pi} = \gamma_1(i)$ the expected frequency of state *i* at time *t=1*

$$\hat{a}_{ij} = \frac{\sum \xi_t(i,j)}{\sum \gamma_t(i)}$$

ratio of expected no. of transitions from state *i* to *j* over expected no. of transitions from state *i*

$$\hat{b}_j(k) = \frac{\sum_{t,o_t=k} \gamma_t(j)}{\sum \gamma_t(j)}$$

ratio of expected no. of times in state *j* observing symbol *k* over expected no. of times in state *j*

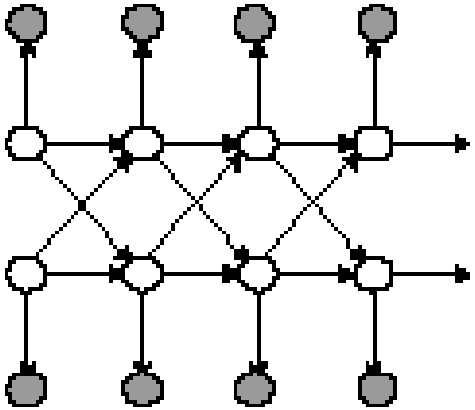
Problem 3: Learning

- Baum-Welch algorithm uses the forward and backward algorithms to calculate the auxiliary variables α,β
- B-W algorithm is a special case of the EM algorithm:
  - E-step: calculation of ξ and γ
  - M-step: iterative calculation of $\hat{\pi}$, $\hat{a}_{ij}$, $\hat{b}_j(k)$
- Practical issues:
  - Can get stuck in local maxima
  - Numerical problems – log and scaling

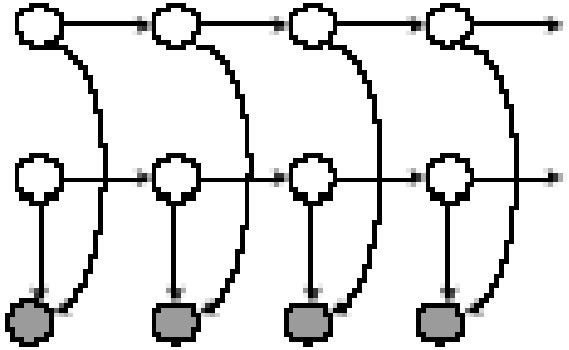
Extensions

- Problem-specific:
  - Left to right HMM (speech recognition)
  - Profile HMM (bioinformatics)
- General machine learning:
  - Factorial HMM
  - Coupled HMM
  - Hierarchical HMM
  - Input-output HMM
  - Switching state systems
  - Hybrid HMM (HMM +NN)
  - Special case of graphical models
    - Bayesian nets
    - Dynamical Bayesian nets

Examples



Coupled HMM



Factorial HMM

HMMs – Sleep Staging

- Flexer, Sykacek, Rezek, and Dorffner (2000)
- Observation sequence: EEG data
- Fit model to data according to 3 sleep stages to produce continuous probabilities: *P*(wake), *P*(deep), and *P*(REM)
- Hidden states correspond with recognised sleep stages. 3 continuous probability plots, giving *P* of each at every second