Statistical Machine Learning Methods for Bioinformatics

I. Hidden Markov Model Theory

Jianlin Cheng, PhD

Informatics Institute, Department of Computer Science

University of Missouri

2009

What’s is Machine Learning?

- As a broad subfield of artificial intelligence, machine learning is concerned with the design and development of algorithms and techniques that allow computers to “learn”.
- The major focus is to extract information from data automatically, by computational and statistical methods. Closely related to data mining, statistics, and theoretical computer science.

Why Machine Learning?

Bill Gates: If you design a machine that can learn, it is worth 10 Microsoft.

Applications of Machine Learning

- Natural language processing, syntactic pattern recognition, search engine
- Bioinformatics, cheminformatics, medical diagnosis
- Detecting credit card fraud, stock market analysis, speech and handwriting recognition, object recognition
- Game playing and robotics

Common Algorithms Types

- **Statistical Modeling** (HMM, Bayesian networks)
- **Supervised Learning** (classification)
- Unsupervised Learning (clustering)

Why Bioinformatics?

For the first time, we computer / information scientists can make direct impact on health and medical sciences. Working with biologists, we will revolutionize the research of life science, decode the secrete of the life, and cure diseases threatening human being.

Don’t Quit Because It is Hard

- **Interviewer**: Do you view yourself a billionaire, CEO, or the president of Microsoft?
- **Bill Gates**: I view myself a software engineer. Business is not complicated, but science is complicated.
- **JFK:** we want to go to moon, not because it is easy, but because it is hard.

Statistical Machine Learning Methods for Bioinformatics

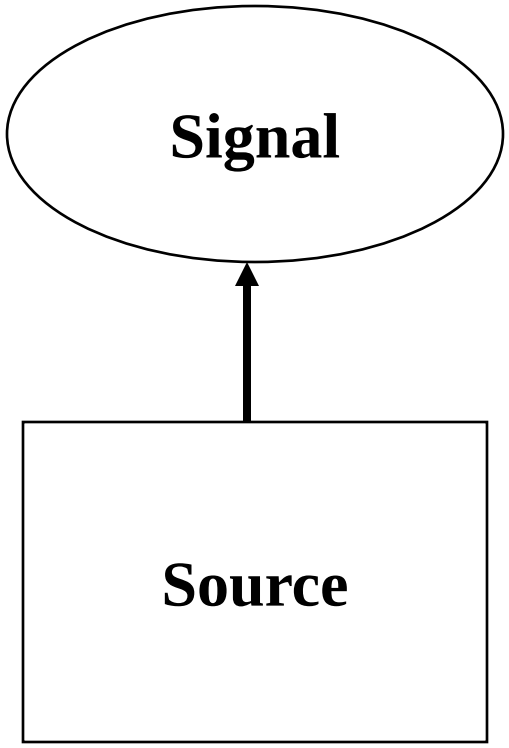
1. Hidden Markov Model and its Application in Bioinformatics (e.g. sequence and profile alignment)
2. Support Vector Machine and its Application in Bioinformatics (e.g. protein fold recognition)
3. EM Algorithm, Gibbs Sampling, and Bayesian Networks and their Applications in Bioinformatics (e.g. gene regulatory network)

Possible Projects

- Multiple sequence alignment using HMM
- Profile-profile alignment using HMM
- Protein secondary structure prediction using neural networks
- Protein fold recognition using support vector machine
- Or your own projects upon my approval

Work alone or in a group up to 3 persons. 40% presentation and 60% report.

Real World Process



Discrete signals: characters, nucleotides,…

Continuous signals: speech samples, music temperature measurements, music,…

Signal can be pure (from a single source) or be corrupted from other sources (e.g Noise)

Stationary source: its statistical properties does not vary.

Nonstationary source: the signal properties vary over time.

Fundamental Problem: Characterizing Real-World Signals in Terms of Signal Models

- A signal model can provide basis for a theoretical description of a signal processing system which can be used to process signals to improve outputs. (remove noise from speech signals).
- Signal models let us learn a great deal about the signal source without having the source available. (simulate source and learn via simulation).
- Signal models work pretty well in practical system – prediction systems, recognition systems, identification systems).

Types of Signal Models

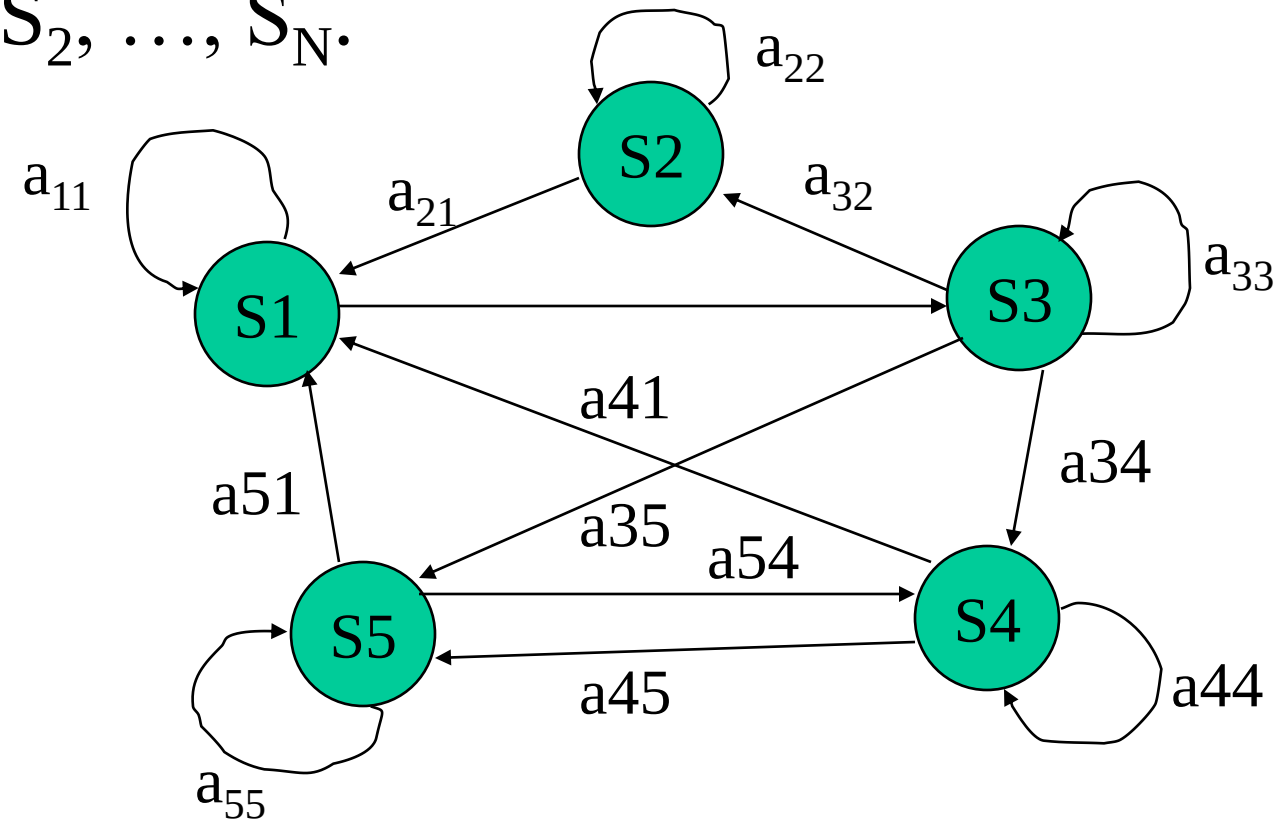
- **Deterministic models**: exploit known properties of signals. (sine wave, sum of exponentials). Easy, only need to estimate parameters (amplitude, frequency, …)
- **Statistical models**: try to characterize only the statistical properties of models (Gaussian processes, Poisson processes, Markov processes, Hidden Markov process…)
- **Assumption of statistical models**: signals can be well characterized as a parametric random process, and the parameters of the stochastic processes can be estimated in a well defined manner.

History of HMM

- Introduced in the late 1960s and early 1970s (L.E. Baum and others)
- Widely used in speech recognition in 1980s. (Baker, Jelinek, Rabiner and others)
- Widely used in biological sequence analysis in 1990s till now (Churchill, Haussler, Krog, Baldi, Durbin, Eddy, Kaplus, Karlin, Burges, Hughey, Snonhammer, Sjolander, Edgar, Soeding, and **many others**)

Discrete Markov Process

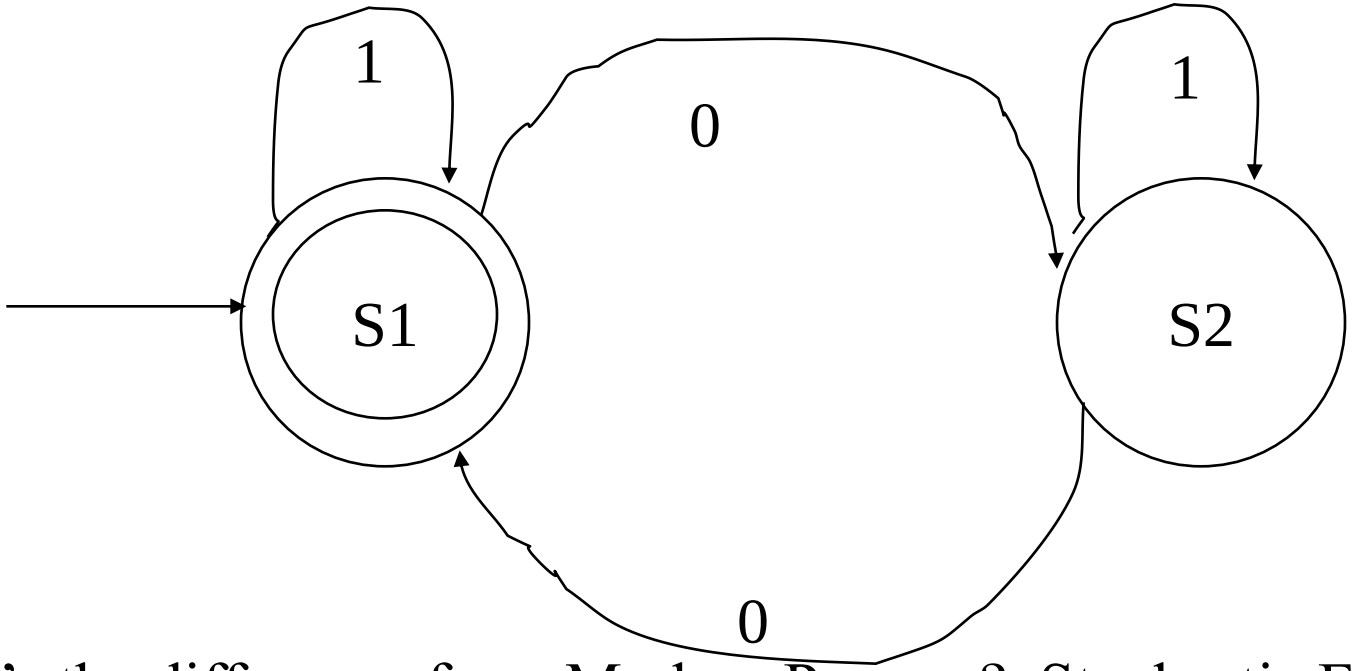
- Consider a system described at any time as being in one of a set of N distinct states, S_1, S_2, …, S_N.



At regularly spaced discrete times, the system undergoes a change of state according to a set of probabilities.

Finite State Machine

- A class of Automata
- Example: deterministic finite automaton (state transition according to input) to determines if the binary input contains an even number of 0s.



What's the difference from Markov Process? Stochastic Finite Automaton

Definition of Variables

- Time instants associated with state changes as t = 1, 2, ….,T
- Observation: $O = o_1 o_2 \ldots o_T$
- Actual state at time t as q_t.
- A full probabilistic description of the system requires the specification of the current state, as well as all the predecessor states.
- The first order Makov chain (truncation):

  $P[q_t = S_j \mid q_{t-1} = S_i, q_{t-2} = S_k, \ldots] = P[q_t = S_j \mid q_{t-1} = S_i]$.
- Further simplification: state transition is independent of time. $a_{ij} = P[q_t = S_j \mid q_{t-1} = S_i]$, 1 <= i, j <= N, subject to constraints:

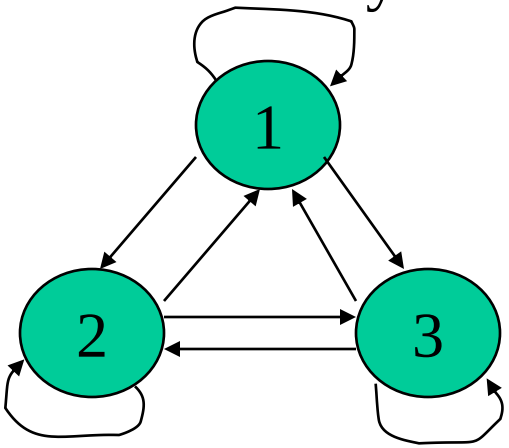
  $a_{ij} >= 0$ and $\sum_{j=1}^{N} a_{ij} = 1$

Observable Markov Model

- The stochastic process is called an observable model if the output of the process is the set of states at each instant of time, which each state corresponds to a physical (observable) event.

Example: Markov Model of Whether

State 1: rain or snow
State 2: cloudy
State 3: sunny



Matrix **A** of state transition probabilities:

$\mathbf{A} = \{a_{ij}\} =$

i \ j		
0.4	0.3	0.3
0.2	0.6	0.2
0.1	0.1	0.8

Observable Markov Model

Example: Markov Model of Whether

State 1: rain or snow
State 2: cloudy
State 3: sunny

Matrix A of state transition probabilities:

$A = \{a_{ij}\} =$

i \ **j**	1	2	3
1	0.4	0.3	0.3
2	0.2	0.6	0.2
3	0.1	0.1	0.8

Question:

Given the weather on day 1 (t=1) is sunny (state 3), what is the probability of for the next 7 days will be "sun – sun – rain – rain – sun – cloudy – sun?

Formalization

- Define the observation sequence O = {S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3} corresponding to t = 1, 2, .., 8.
- P (O | M) = P[S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3 | Model]

  = P[S_3] * P[S_3|S_3] * P[S_3|S_3] * P[S_1|S_3] *

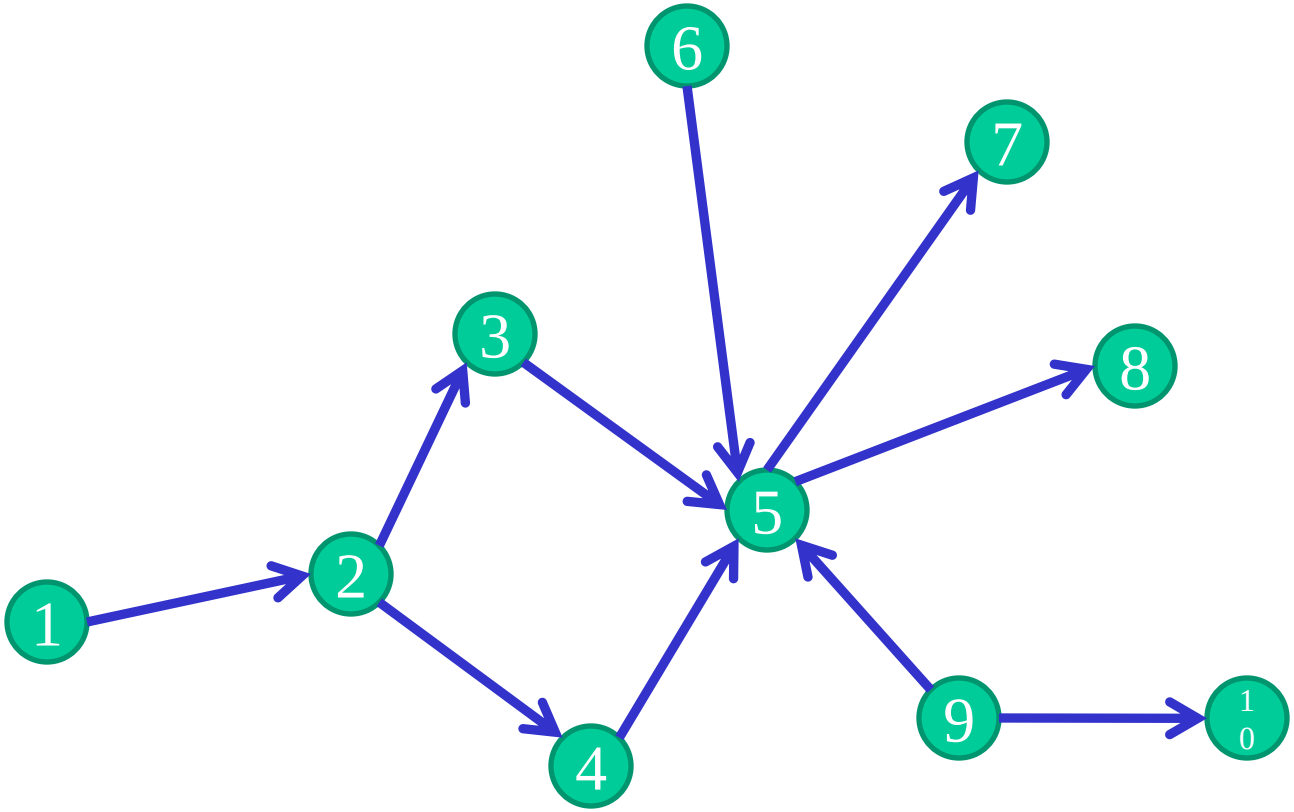
  P[S_1|S_1] * P[S_3|S_1] * P[S_2|S_3] * P[S_3|S_2]

  = π3 * 0.8 * 0.8 * 0.1 * 0.4 * 0.3 * 0.1 * 0.2

  = $1.536 * 10^{-4}$

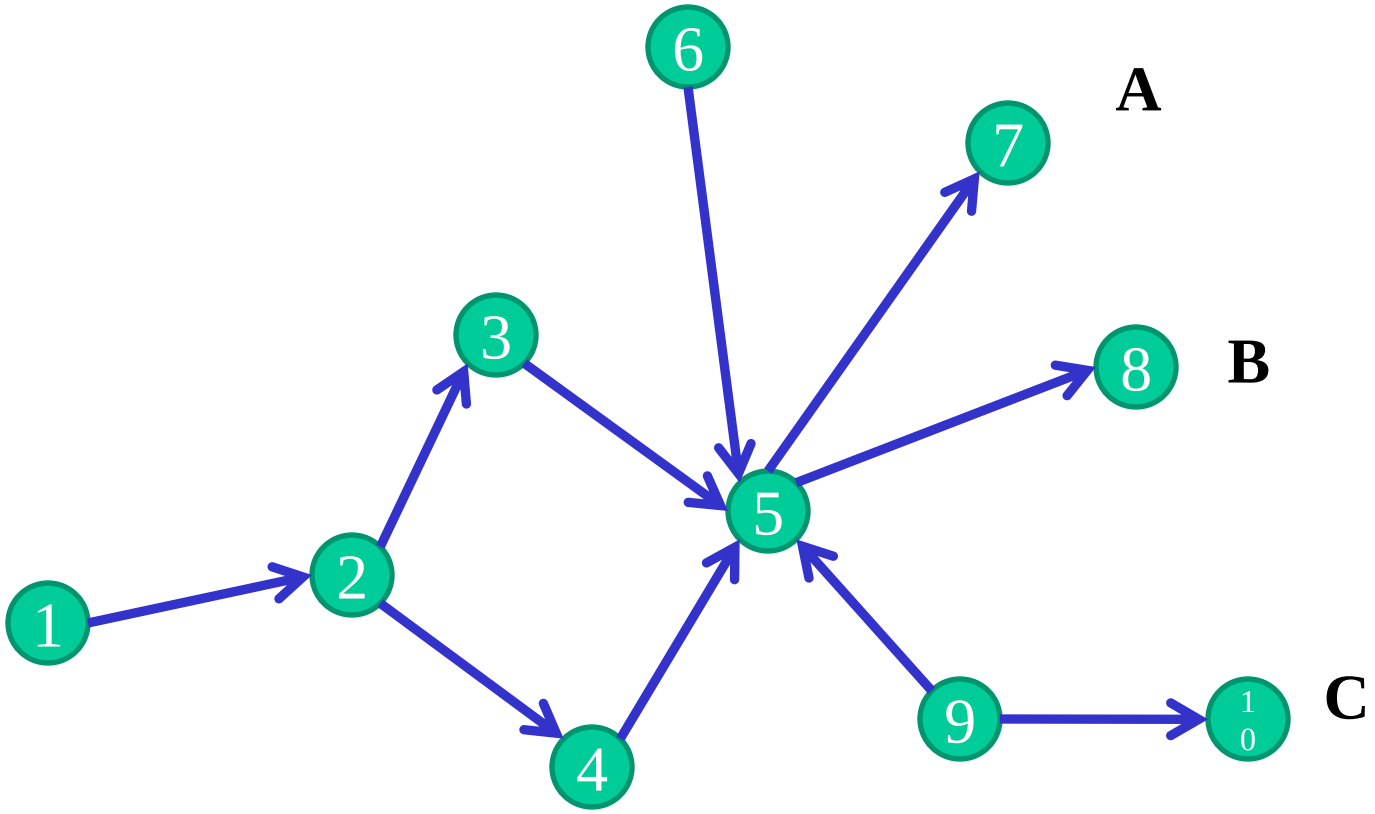
Denote π_i = P[q_1 = S_i], 1 <= i <= N.

Web as a Markov Model



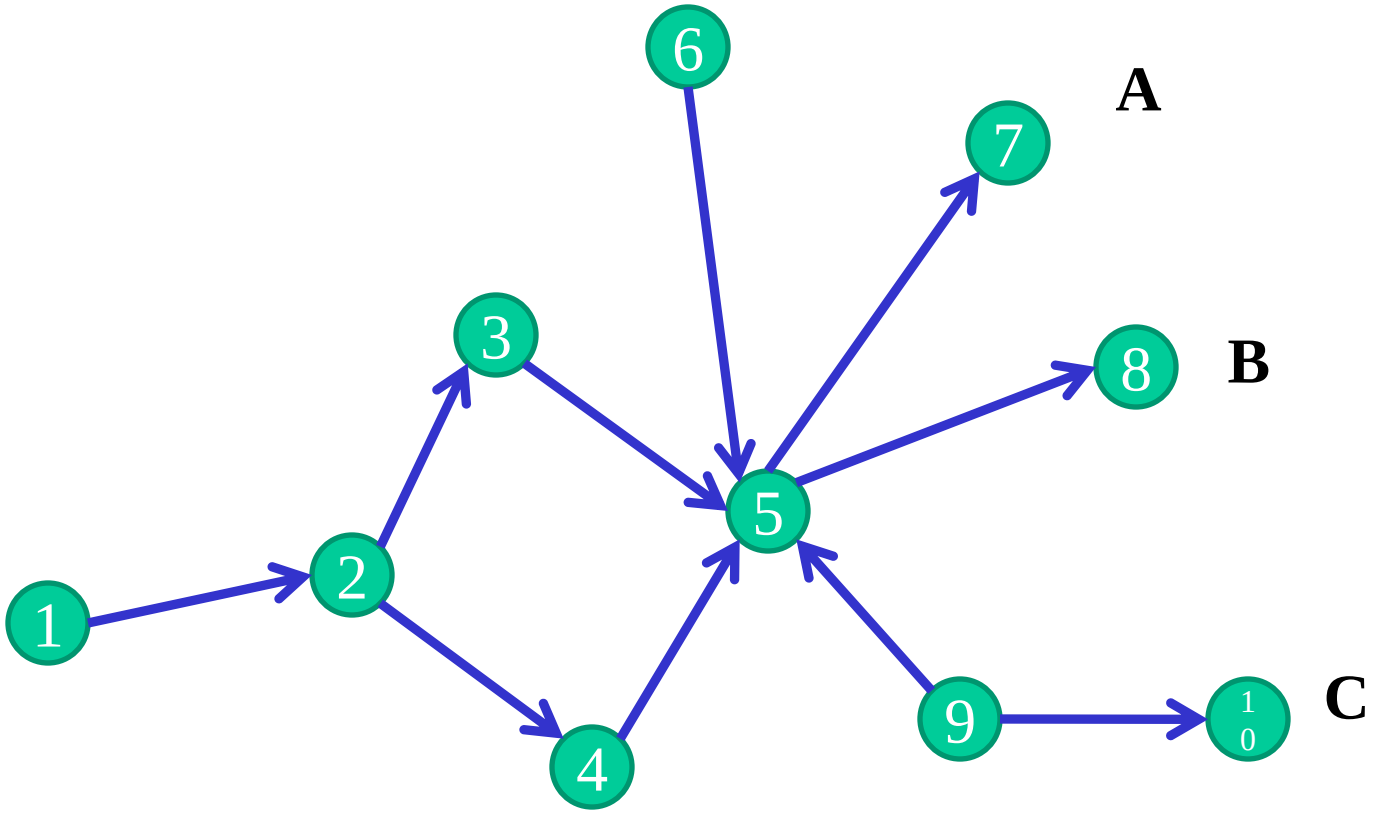
Which web page is most likely to be visited? (most popular?)
Which web pages are least likely to be visited?

Web as a Markov Model



Which one (A or C) is more likely to be visited?
How about A and B?

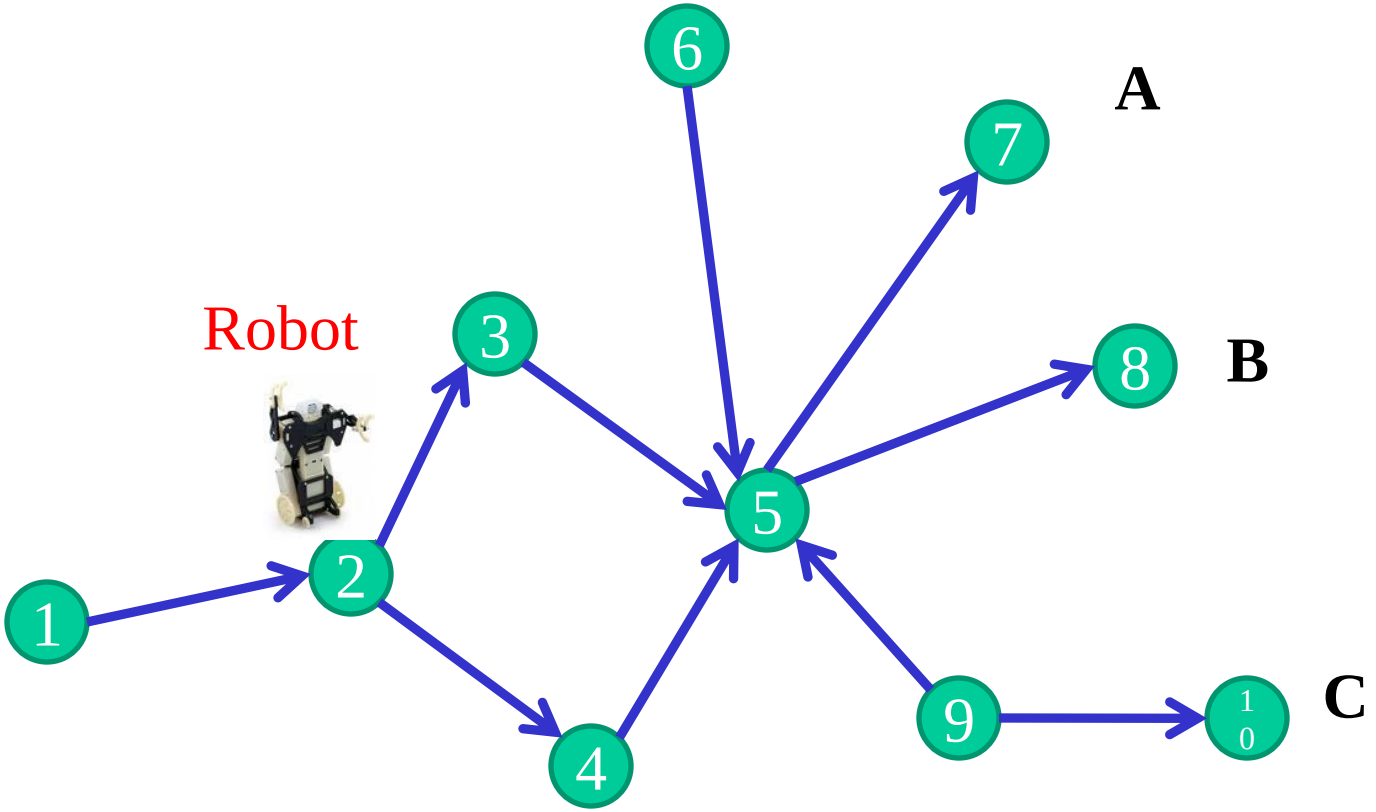
Random Walk – Markov Model



Which one (A or C) is more likely to be visited?
How about A and B?

What two things are important to the popularity of a page?

Random Walk – Markov Model



Randomly select an out-link to visit next page
The probability of taking an out-link is equally distributedly among all links

Page Rank Calculation

Web Link Matrix

	1	2	3	4	5	6	7	8	9	10
1	1									
2	1	1								
3		1/2	1							
4		1/2		1						
5			1	1	1	1			1/2	
6						1				
7					1/2		1			
8					1/2			1		
9									1	
10									1/2	1

Rank Vector

	0.1		0.1
	0.1		0.2
	0.1		0.15
	0.1		0.15
X	0.1	=	0.45
	0.1		0.1
	0.1		0.15
	0.1		0.15
	0.1		0.1
	0.1		0.15

Each row is the in-link of the page

Repeat until it converges.

Hidden Markov Models

- Observation is a probabilistic function of the state.
- Doubly embedded process: Underlying stochastic process that is not observable (hidden), but only be observed through another set of stochastic processes that produce the sequence of observations.

Coin Tossing Models

- A person is tossing multiple coins in other room. He tells you the result of each coin flip.
- A sequence of hidden coin tossing experiments is performed, with observation sequence consisting of a series of heads and tails.

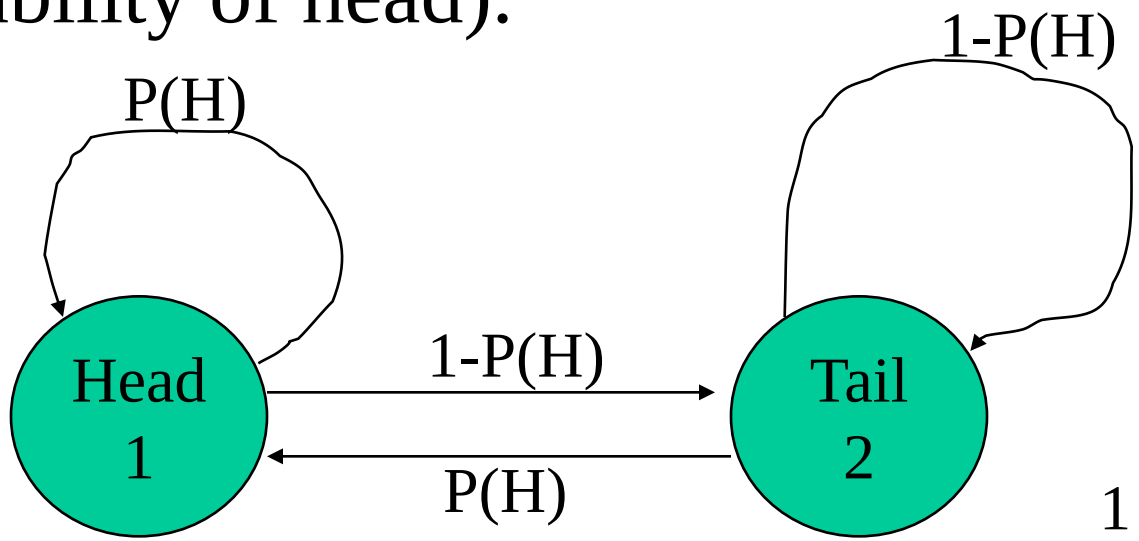
$O = O_1 O_2 O_3 \ldots O_T = HHTTTHTTH \ldots H$

How to Build a HMM to Model to Explain the Observed Sequence?

- What do the states in the model correspond to?
- How many states should be in the model?

Observable Markov Model for Coin Tossing

- Assume a single biased coin is tossed.
- States: Head and Tail.
- The Markov model is observable, the only issue for complete specification of the model is to decide the best value of bias (the probability of head).

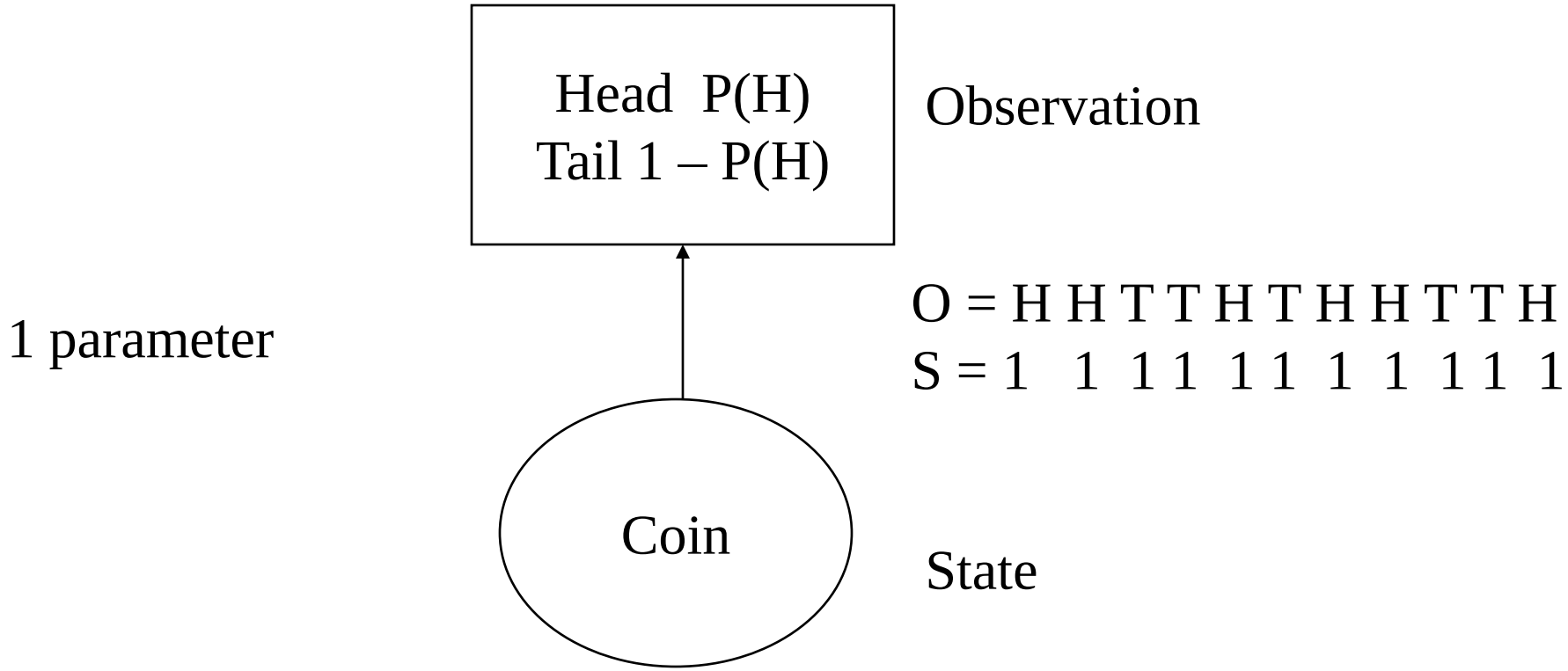


1 parameter

O = H H T T H T H H T T H

S = 1 1 2 2 1 2 1 1 2 2 1

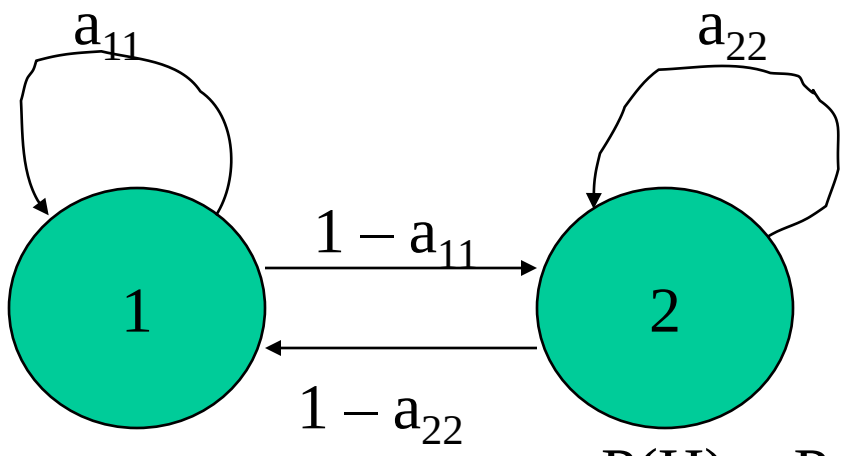
One State HMM Model



Unknown parameter is bias: P(H). Degenerate HMM

Two-State HMM

- 2 states corresponding to two different, biased coins being tossed.
- Each state is characterized by a probability distribution of heads and tails.
- Transition between states are characterized by a state transition matrix.



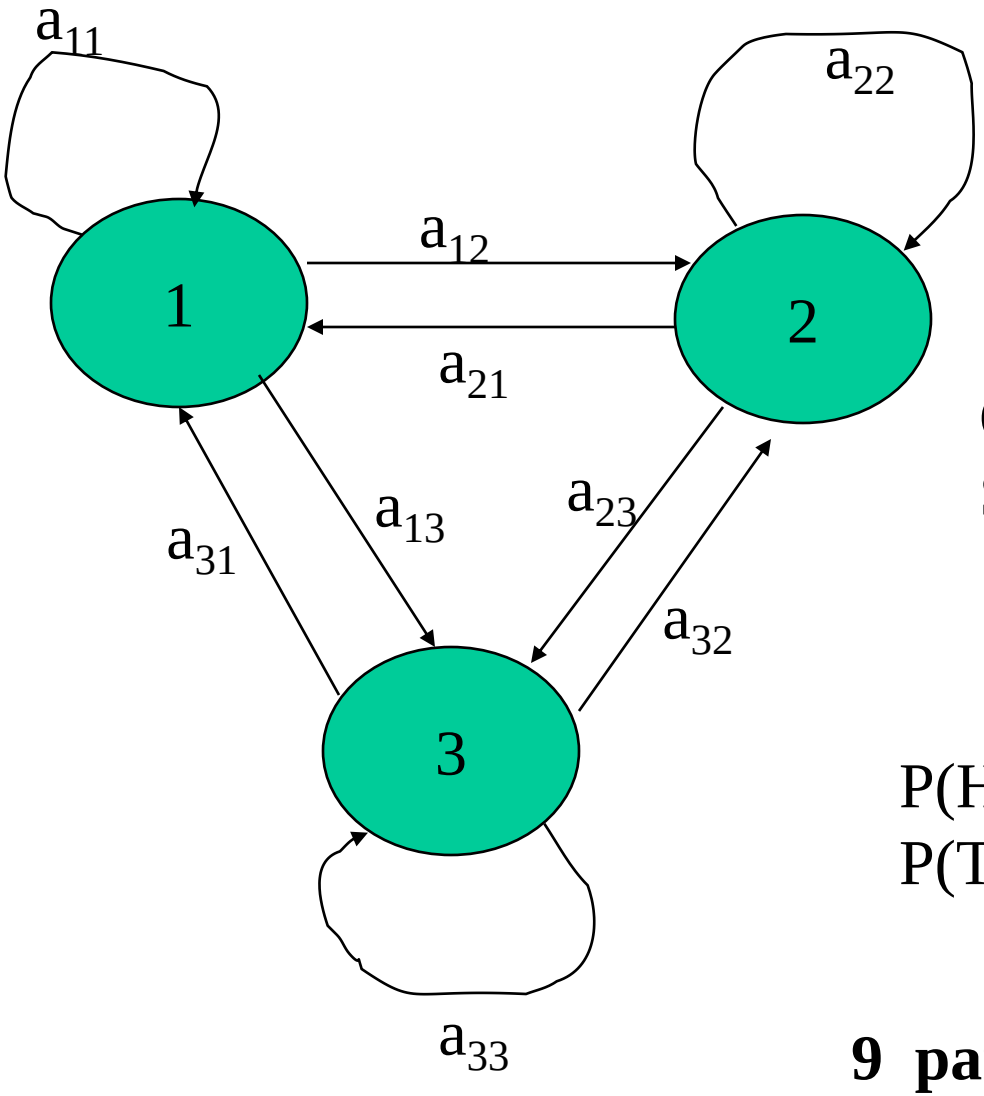
$P(H) = P_1$
$P(T) = 1 - P_1$

$P(H) = P_2$
$P(T) = 1 - P_2$

O = H H T T H T H H T T H
S = 2 1 1 2 2 2 1 2 2 1 2

4 Parameters

Three-State HMM



O = H H T T H T H H T T H

S = 3 1 2 3 3 1 1 2 3 1 3

	1	2	3
P(H)	P_1	P_2	P_3
P(T)	$1-P_1$	$1-P_2$	$1-P_3$

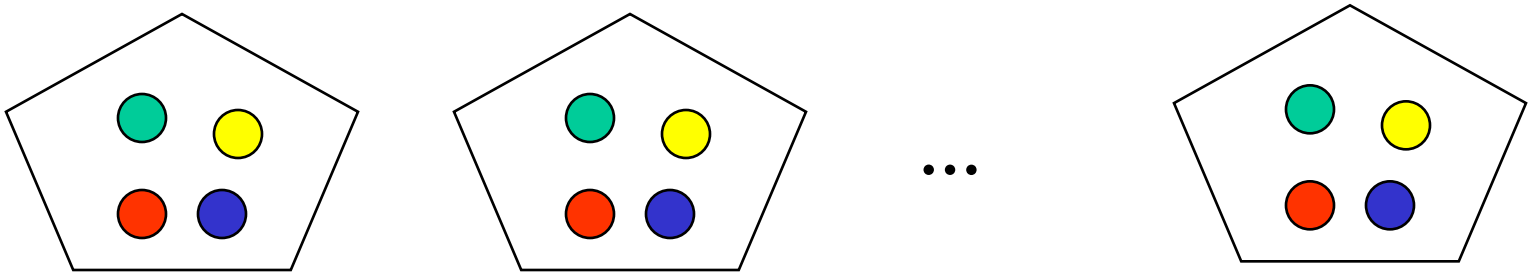
9 parameters

Model Selection

- Which model best match the observations? P(O|M)
- 1-State HMM and Markov Model has one parameter.
- 2-State HMM has four parameters.
- 3-State HMM has nine parameters.
- Practical considerations impose strong limitation on model size. (validation, prediction performance)
- Constrained by underlying physical process.

N-State M-Symbol HMM

- N Urns. Each Urn has a number of colored balls. M distinct colors of the balls.



Urn 1

$P(red) = b_1(1)$
$P(blue) = b_1(2)$
$P(green) = b_1(3)$
…
$P(Yellow) = b_1(M)$

Urn 2

$P(red) = b_2(1)$
$P(blue) = b_2(2)$
$P(green) = b_2(3)$
…
$P(Yellow) = b_2(M)$

Urn N

$P(red) = b_N(1)$
$P(blue) = b_N(2)$
$P(green) = b_N(3)$
…
$P(Yellow) = b_N(M)$

O = {green, green, blue, red, yellow, red, …., blue}

Elements of HMM

- N, the number of states in the model. $S = \{S_1, S_2, \ldots, S_N\}$, and the state at time t as q_t.
- M, the number of distinct observation symbols per state, i.e., the discrete alphabet size. The observation symbols corresponds to the physical output of the system. V = {v1, v2, …, V_M}
- A, the state transition probability matrix. $A = \{a_{ij}\}$. $a_{ij} = P[q_{t+1} = j \mid q_t = i]$, $1 <= i, j <= N$. For the special case where any state can reach any other state in a single step ($a_{ij}>0$) for all i, j.
- The observation symbol (emission) probability distribution in state j, $B=\{b_j(k)\}$, where $b_j(k) = P[v_k \text{ at } t \mid q_t = s_j]$, $1<= j <= N$, $1<= k <= M$.
- The initial state distribution $\pi = \{\pi_i\}$ where $\pi_i = P[q_1=S_i]$, $1<= i <=N$.
- Complete specification: N, M, A, B, and π. $\lambda = \{A, B, \pi\}$

Use HMM to Generate Observations

Given appropriate values of N, M, A, B, and π, HMM can be used as a generator to give an observation sequence: $O = O_1\ O_2\ \ldots\ O_T$. Each O_t is one of symbols from V, and T is the number of observations in the sequence as follows:

1. Choose an initial state $q_1 = S_i$ according to π.
2. Set t = 1
3. Choose $O_t = v_k$ according to the symbol probability distribution in state S_i, i.e., $b_i(k)$.
4. Transit to a new state $q_{t+1} = S_j$ according to the state transition probability distribution for state S_i, i.e., a_{ij}.
5. Set t = t + 1; return to step 3 if t < T; otherwise terminate the procedure.

Three Main Problems of HMM

1. Given the observation $O = O_1 O_2 \ldots O_T$ and a model $\lambda = (A, B, \pi)$, how to efficiently compute $P(O|\lambda)$?
2. Given the observation $O = O_1 O_2 \ldots O_T$ and the model λ, how to we choose a corresponding state sequence (path) $Q = q_1 q_2 \ldots q_T$ which is optimal in some meaningful sense? (best explain the observations)
3. How do we adjust the model parameters λ to maximize $P(O|\lambda)$?

Insights

- Problem 1: **Evaluation and scoring** (choose the model which best matches the observations)
- Problem 2: **Decoding. Uncover the hidden states**. We usually use an optimality criterion to solve this problem as best as possible.
- Problem 3: **Learning**. We attempt to optimize the model parameters so as to best describe how a given observation sequence comes out. The observation sequence used to adjust the model parameters is called a training sequence. Key to create best models for real phenomena.

Word Recognition Example

- For each word of in a word vocabulary, we design a N-state HMM.
- The speech signal (observations) of a given word is a time sequence of coded spectral vectors (spectral code, or ***phoneme***). (M unique spectral vectors). Each observation is the index of the spectral vector closest to the original speech signal.
- **Task 1**: **Build individual word models** by using solution to Problem 3 to estimate model parameters for each model.
- **Task 2**: **Understand the physical meaning of model states** using the solution to problem 2 to segment each of word training sequences into states and then study the properties of the spectral vectors that lead to the observation in each state. Refine model.
- **Task 3**: **Use solution to Problem 1 to score each word model** based upon the given test observation sequence and select the word whose score is highest.

Solution to Problem 1 (scoring)

- Problem: calculate the probability of the observation sequence, $O=O_1O_2\ldots O_T$, given the model λ, i.e., $P(O|\lambda)$.
- Brute force: enumerate every possible state sequence of length T. Consider one such fixed state sequence $Q = q_1q_2\ldots q_T$.
  $P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t \mid q_t, \lambda)$, assuming statistical independence of observations. We get,
  $P(O|Q, \lambda) = b_{q1}(O_1) * b_{q2}(O_2) \ldots bq_T(O_T)$.

- $P(Q|\lambda) = \pi_{q1} a_{q1q2} a_{q2q3} \ldots a_{qT\text{-}1qT}$
- Joint probability: $P(O,Q|\lambda) = P(O|Q,\lambda)P(Q|\lambda)$.
- The probability of O (given the model)

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

$$= \sum_{q_1 q_2 \ldots q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

Time complexity: $O(T * N^{T})$

Forward-Backward Algorithm

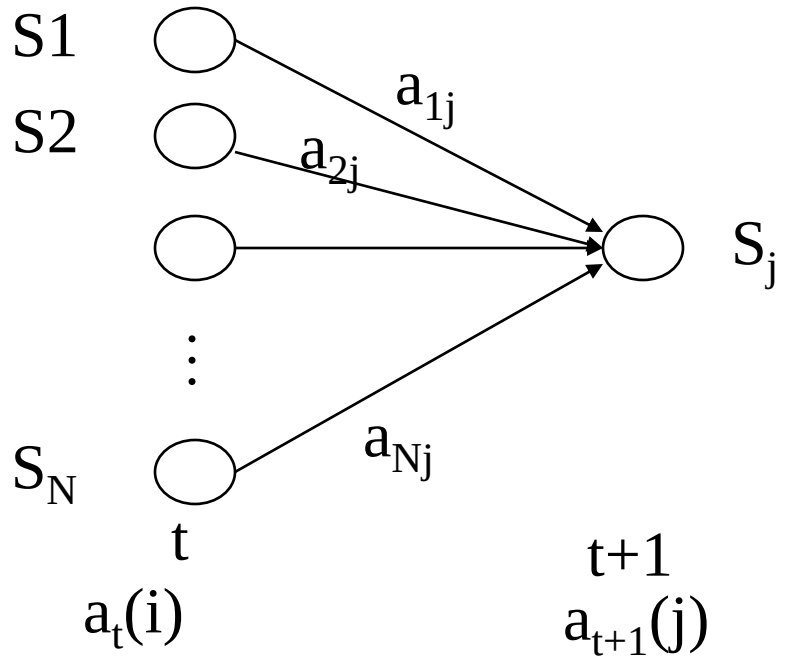
Definition: Forward variable $a_t(i) = P(O_1,O_2 \ldots O_t, q_t=S_i|\lambda)$, i.e., the joint probability of the partial observation $O_1O_2 \ldots O_t$ and state S_i at time t.

Algorithm (inductive or recursive):

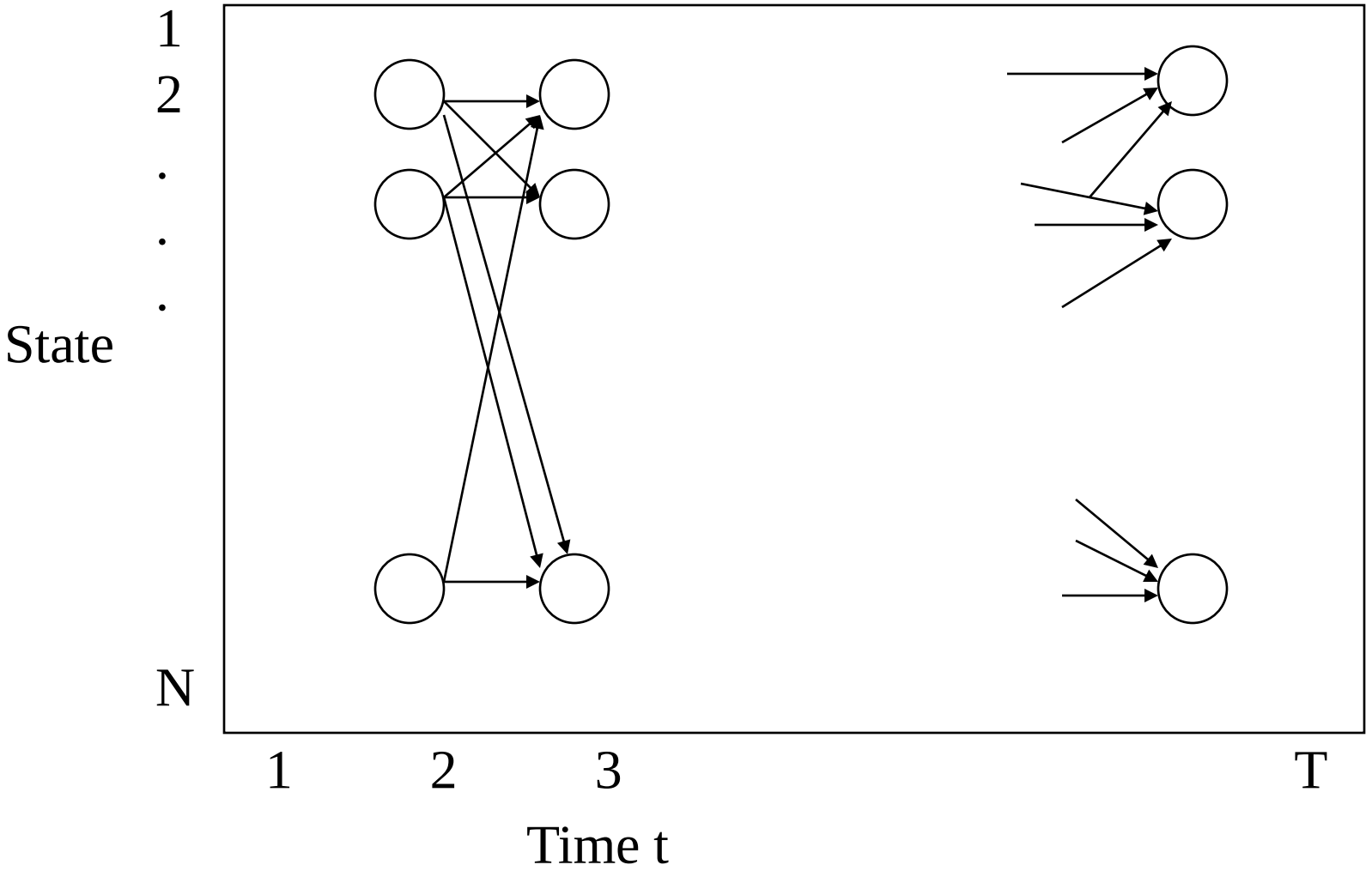
1. Initialization: $a_1(i) = \pi_i b_i(O_1)$, $1 <= i <= N$
2. Induction: $a_{t+1}(j) = (\sum_{i=1}^{N} a_t(i)a_{ij})b_j(O_{t+1})$, $1<= t <= T-1$, $1 <= j <= N$.
3. Termination: $P(O|\lambda) = \sum_{i=1}^{N} a_T(i)$

Insights

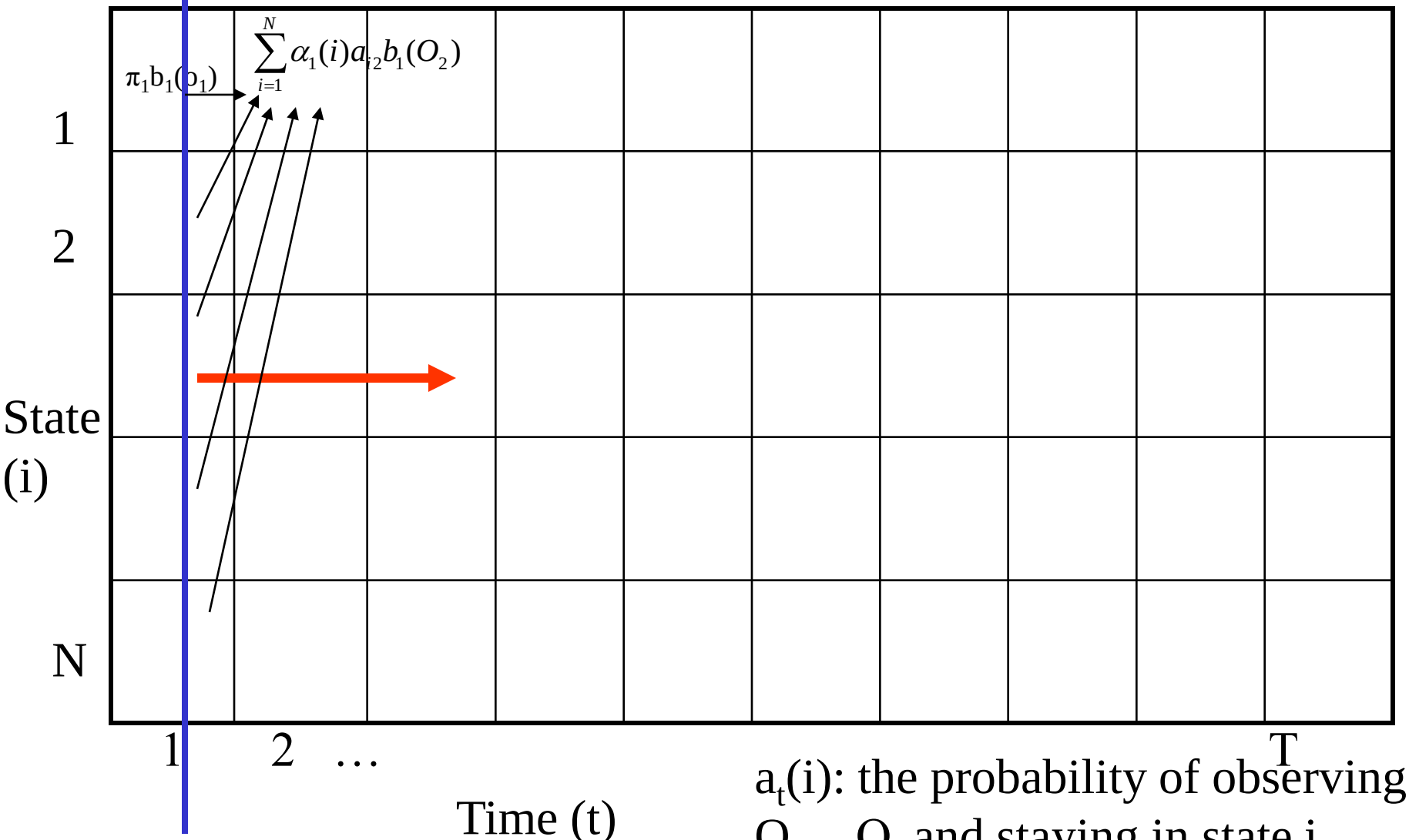
- Step 1: initialize the forward probabilities as the joint probability of state S_i and initial observation O_1.
- Step 2 (induction):



Lattice View



Matrix View (Implementation)



$a_t(i)$: the probability of observing $O_1 \ldots O_t$ and staying in state i.

Fill the matrix column by column. What other bioinformatics algorithm does this algorithm look like?

Time Complexity

- The computation is performed for all states j, for a given time t; the computation is then iterated for t=1,2,…T-1. Finally the desired is the sum of the terminal forward variable $a_T(i)$. This is the case since $a_T(i) = P(O_1O_2…O_T, q_T = S_i|\lambda)$.
- Time complexity: N^2T.

Insights

- Key: there is only N states at each time slot in the lattice, all the possible state sequences will merge into these N nodes, no matter how long the observation sequence.
- Similar to Dynamic Programming (DP trick).

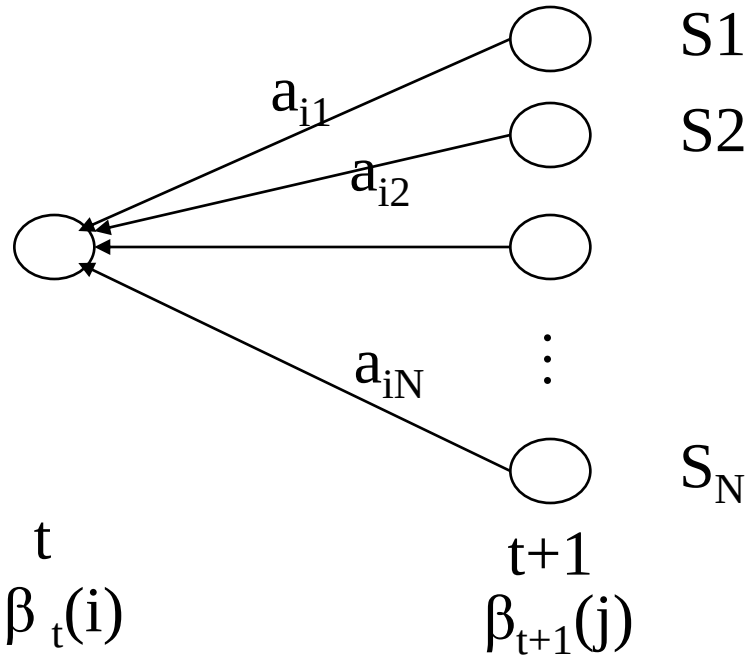
Backward Algorithm

- Definition: backward variable $\beta_t(i)$ = $P(O_{t+1}O_{t+2}\ldots O_T|q_t=S_i,\lambda)$.
- Initialization: $\beta_T(i) = 1$, $1 <= i <= N$. ($P(O_{T+1}|q_T=S_i, \lambda) = 1$. O_{T+1}, a dummy).
- Induction: $\beta_t(i) = \sum \alpha_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$

  $t = T-1, T-2, \ldots, 1$, $1 <= j <= N$.

The initialization step 1 arbitrarily defines $\beta_T(i)$ to be 1 for all i.

Backward Algorithm



Time Complexity: N^2T

Solution to Problem 2 (decoding)

- Find the "optimal" states associated with the given observation sequence. There are different optimization criterion.
- One optimality criterion is to choose the states q_t which are individually most likely. This optimality criterion maximizes the expected number of correct individual states.
- Definition: $\gamma_t(i) = P(q_t=S_i|O, \lambda)$, i.e., the probability of being in state S_i at time t, given the observation sequence O, and the model λ.

Gamma Variable is Expressed by Forward and Backward Variables

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$

The normalization factor $P(O|\lambda)$ makes $\gamma_t(i)$ a probability measure so that its sum over all states is 1.

Solve the Individually Most Likely State q_t

Algorithm: for each column (t) of gamma matrix, select element with maximum value

$$q_t = \arg\max_{1 \le i \le N} [\gamma_t(i)]$$

$$1 <= t <= T$$

This maximizes the expected number of correct states by choosing the most likely state for each time t. A simple join of individually most likely states won't yield an optimal path.

Problem with resulting state sequence (path) : low probability (a_{ij} is low) or even not valid ($a_{ij} = 0$).

Reason: determines the most likely state at every instant, without regard to the probability of occurrence of sequences of states.

Find the Best State Sequence (Viterbi Algorithm)

- Dynamic programming
- Definition: $\delta_t(i) = \max_{q_1 q_2 \ldots q_t} P[q_1 q_2 \ldots q_t = i, o_1 o_2 \ldots o_t \mid \lambda]$

  the best score (highest probability) along a single path at time t, which accounts for the first t observations and ends in state S_i.
- Induction: $\delta_{t+1}(j) = [\max_i \delta_t(i)\alpha_{ij}]b_j(O_{t+1})$
- To retrieve the state sequence, we need to keep track of the chosen state for each t and j. We use a array (matrix) $\psi_t(j)$ to record them.

Viterbi Algorithm

- Initialization:

  $\delta_1(i)=\pi_i b_i(O_1), \quad 1 <= i <= N$

  $\Psi_1(i) = 0.$

- Recursion $\delta_t(j) = \max_{1\le i\le N}[\delta_{t-1}(i)\alpha_{ij}]b_j(O_t), 2\le t\le T, 1\le j\le N$

  $\psi_t(j) = \arg\max_{1\le i\le N}[\delta_{t-1}(i)\alpha_{ij}], 2\le t\le T, 1\le j\le N.$

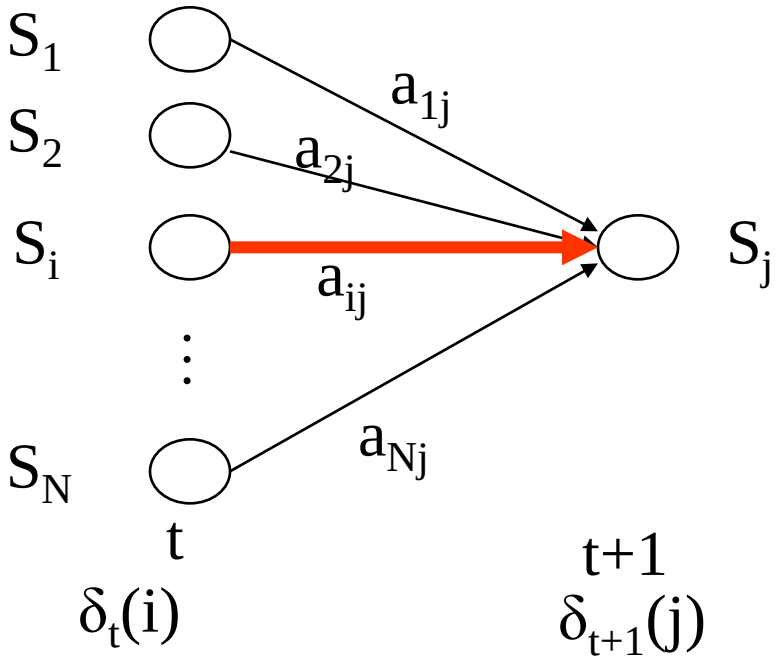
- Termination

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

- Path (state) backtracking

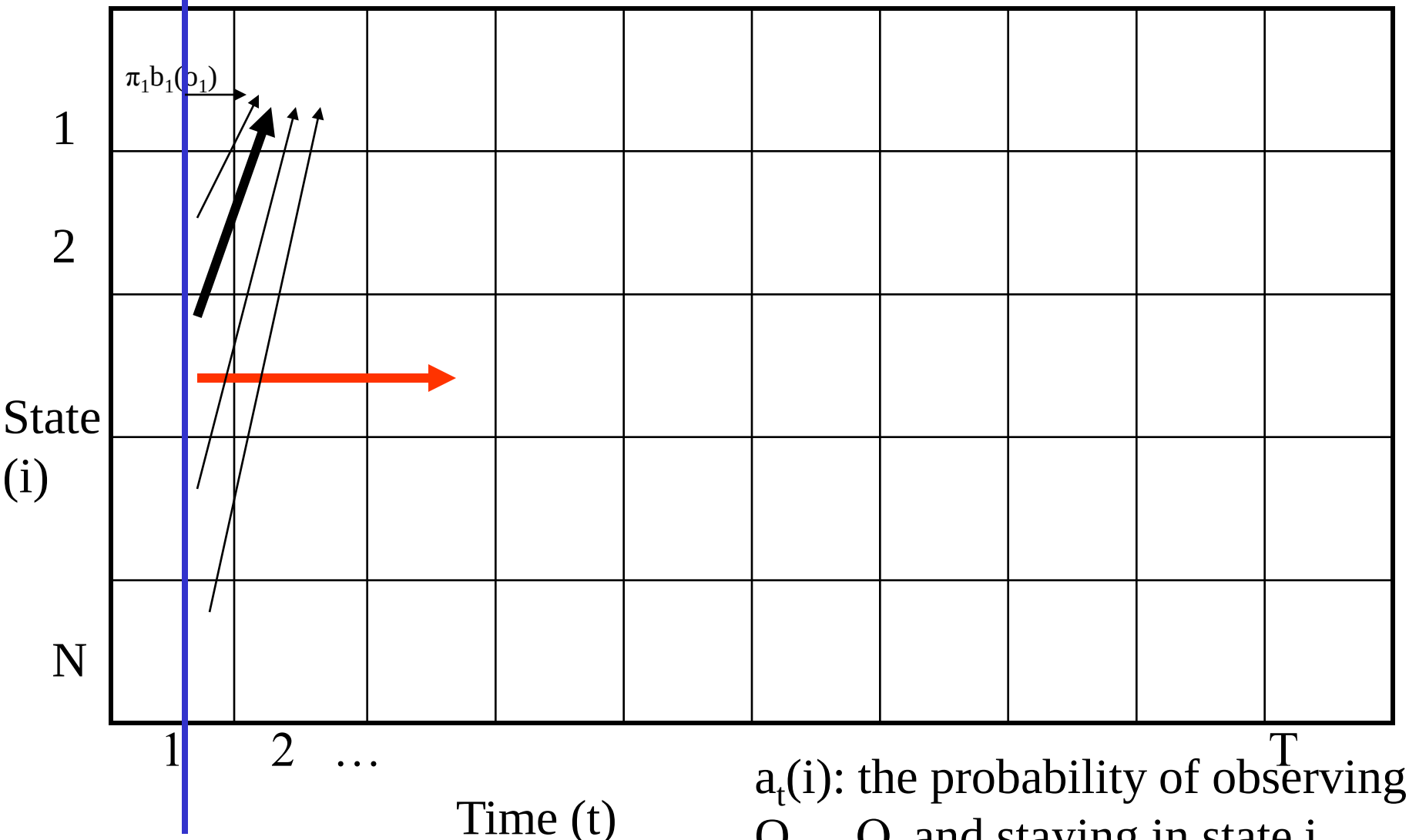
  $q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, ..., 1$

Induction



Choose the transition step yielding the maximum probability.

Matrix View (Implementation)



$a_t(i)$: the probability of observing $O_1 \ldots O_t$ and staying in state i.

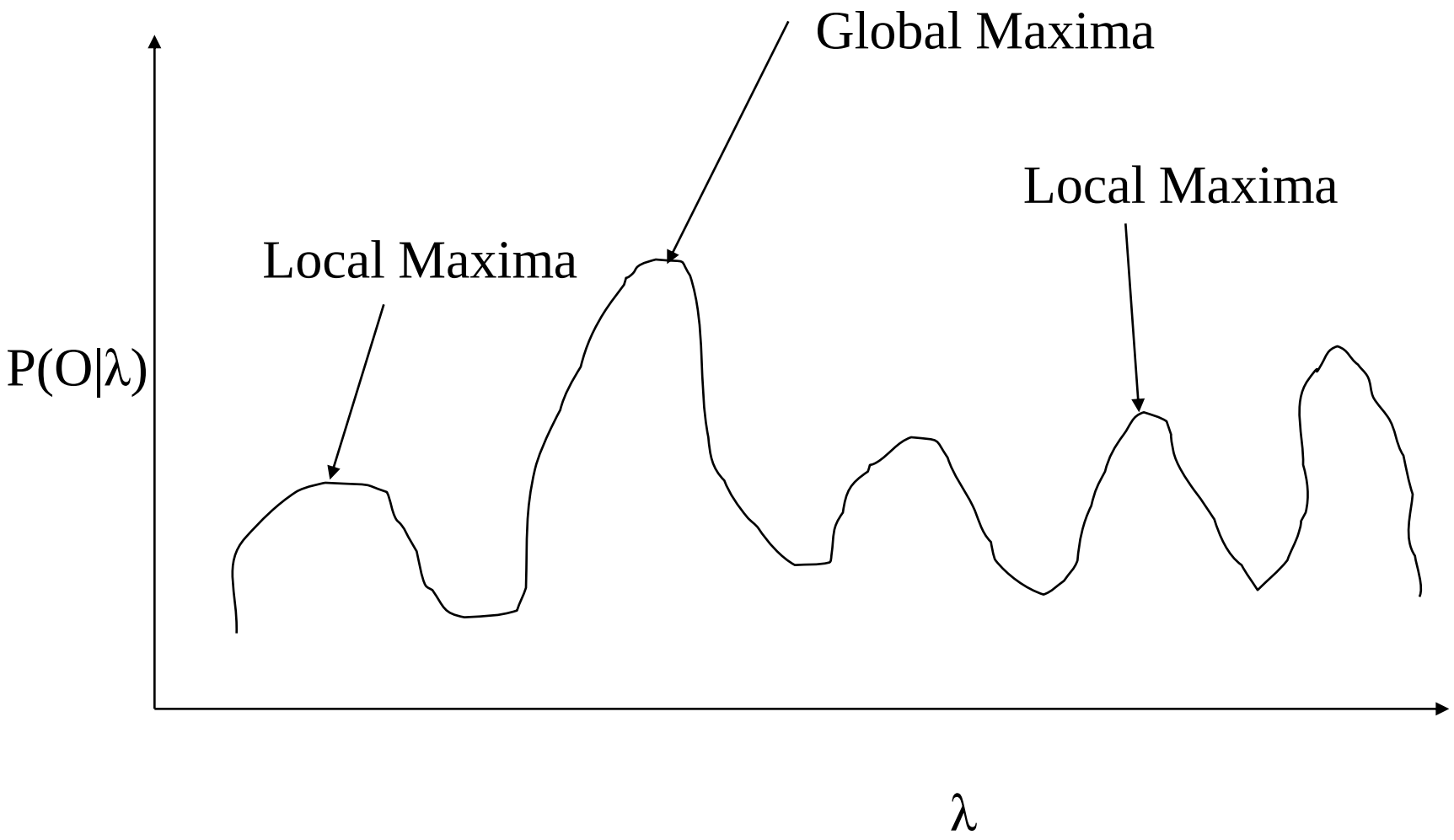
Fill the matrix column by column. What other bioinformatics algorithm does this algorithm look like? (sequence alignment)

Insights

- Viterbi algorithm is similar (except for the backtracking step) in implementation to the forward calculation.
- The major difference is the maximization over previous states instead of the summing procedure used in the forward algorithm.
- The same lattice / matrix structure efficiently implements the Viterbi algorithm.
- Viterbi algorithm is similar as global sequence alignment algorithm. (both use dynamic programming)

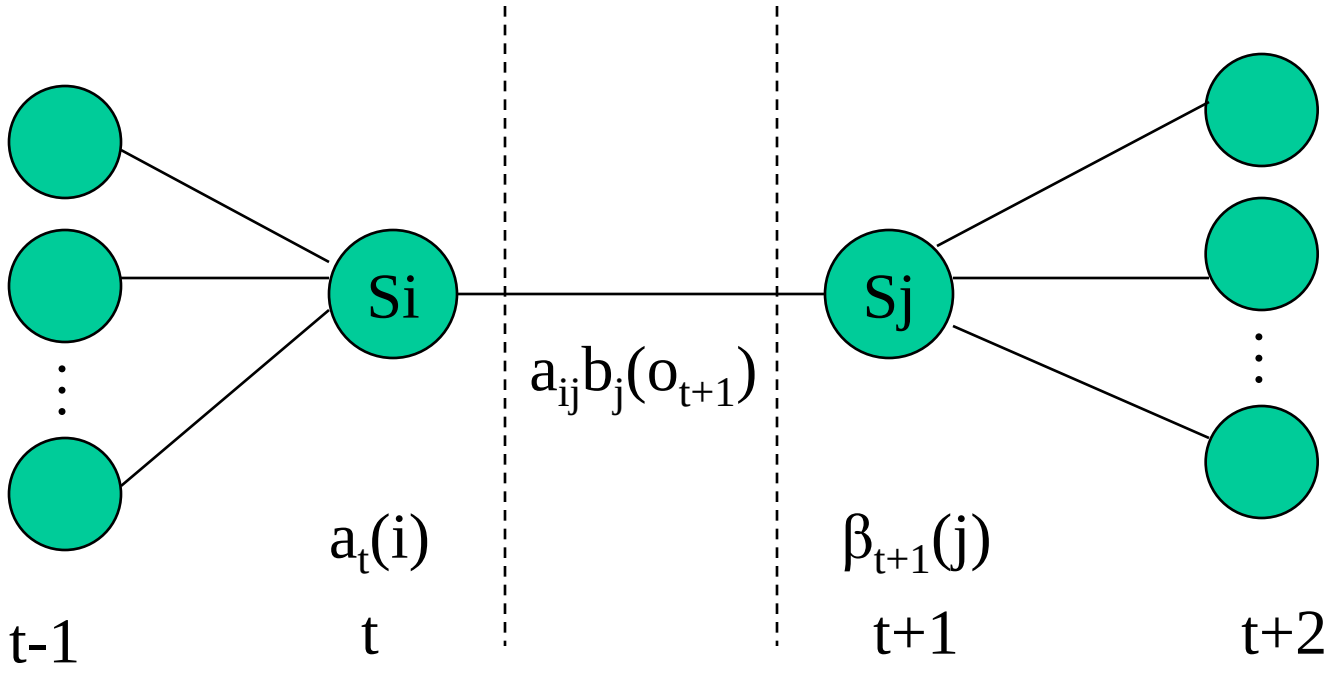
Solution to Problem 3: Learning

- **Goal**: adjust the model parameters $\lambda=(A,B,\pi)$ to maximize the probability ($P(O|\lambda)$) of the observation sequence given the model. (called maximum likelihood)
- **Bad news**: No optimal way of estimating the model parameters to find global maxima.
- **Good news**: We can locally maximize $P(O|\lambda)$ using iterative procedure such as Baum-Welch algorithm, EM algorithm, and gradient techniques.


Global Maxima
Local Maxima
Local Maxima
P(O|λ)
λ

Baum-Welch Algorithm

- Definition: $\xi_t(i,j)$, the probability of being in state S_i at time t and state S_j at time t+1, given the model and observation sequence, i.e. $\xi_t(i,j) = P(q_t=S_i, q_{t+1}=S_j|O,\lambda)$



From the definitions of the forward and backward variables, we can write $\xi_t(i,j)$ in the form:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)}$$

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

The numerator term is just $P(q_t=S_i, q_{t+1}=S_j|O,\lambda)$ and the division by $P(O|\lambda)$ gives the desired probability measure.

Important Quantities

- $\gamma_t(i)$ is the probability of being in state S_i at time t, given the observation and the model, hence we can relate $\gamma_t(i)$ to $\xi_t(i,j)$ by summing over j, giving

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

$\sum_{1}^{T-1} \gamma_t(i)$ = expected number of transitions from S_i.

$\sum_{t=1}^{T-1} \xi_t(i, j)$ = expected number of transitions from S_i to S_j.

Re-estimation of HMM parameters

A set of reasonable re-estimation formulas for π, A, and B are:

$\overline{\pi_i}$ = expected frequency (number of times) in state S_i at time (t=1) = $\gamma_1(i)$

$$\overline{a_{ij}} = \frac{\text{Expected number of transitions from state } S_i \text{ to state } S_j}{\text{Expected number of transitions from state } S_i.} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\overline{b_j}(k) = \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state j}} = \frac{\sum_{t=1, s.t. O_t = V_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Baum-Welch Algorithm

- **Initialize** model $\lambda = (A,B,\pi)$
- **Repeat**

E-step: Use forward/backward algorithm to expected frequencies, given the current model λ and O.

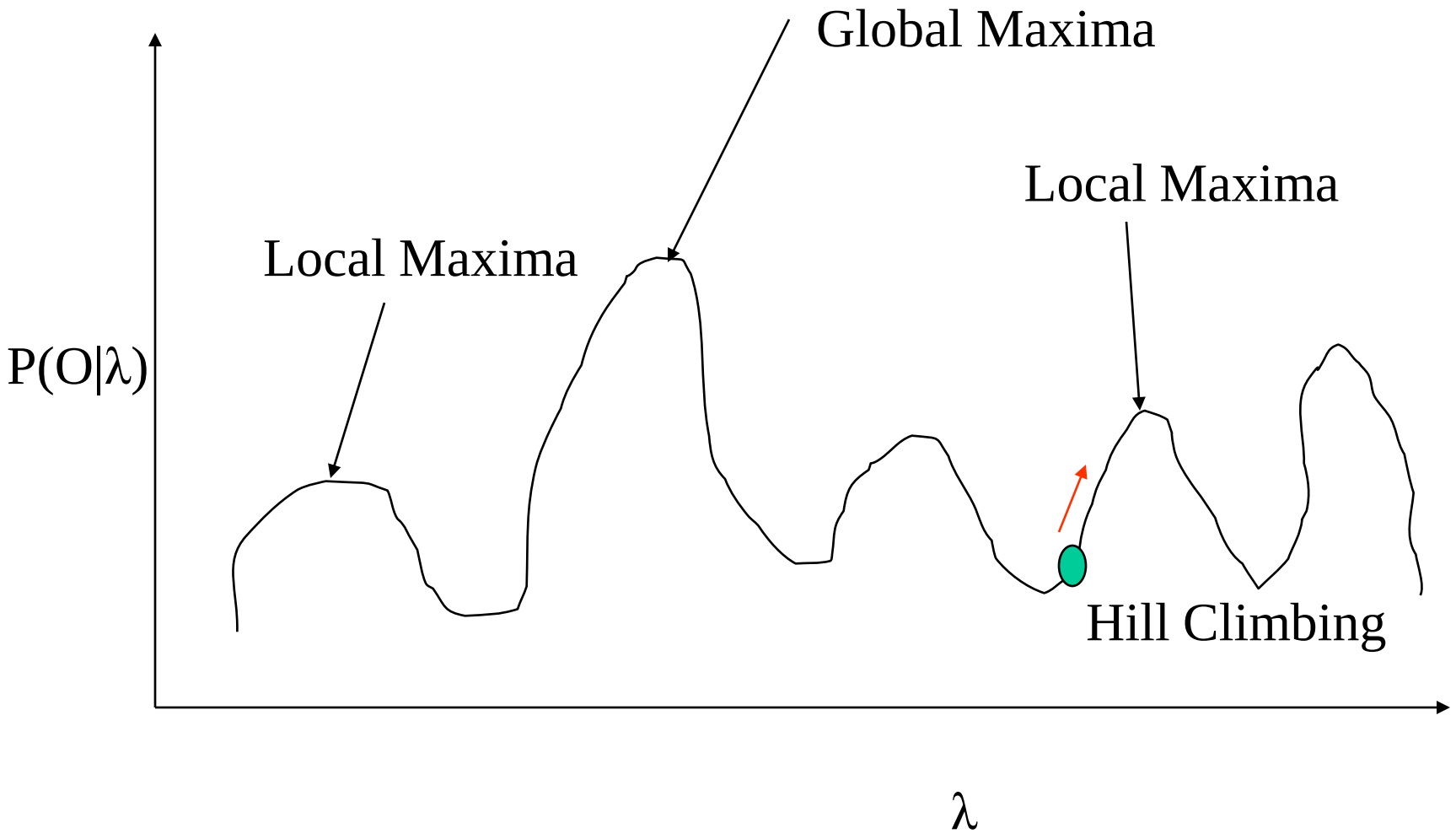
M-step: Use expected frequencies compute the new model $\overline{\lambda}$.

If $(\overline{\lambda} = \lambda)$, stops, otherwise, set $\lambda = \overline{\lambda}$ and go to repeat.

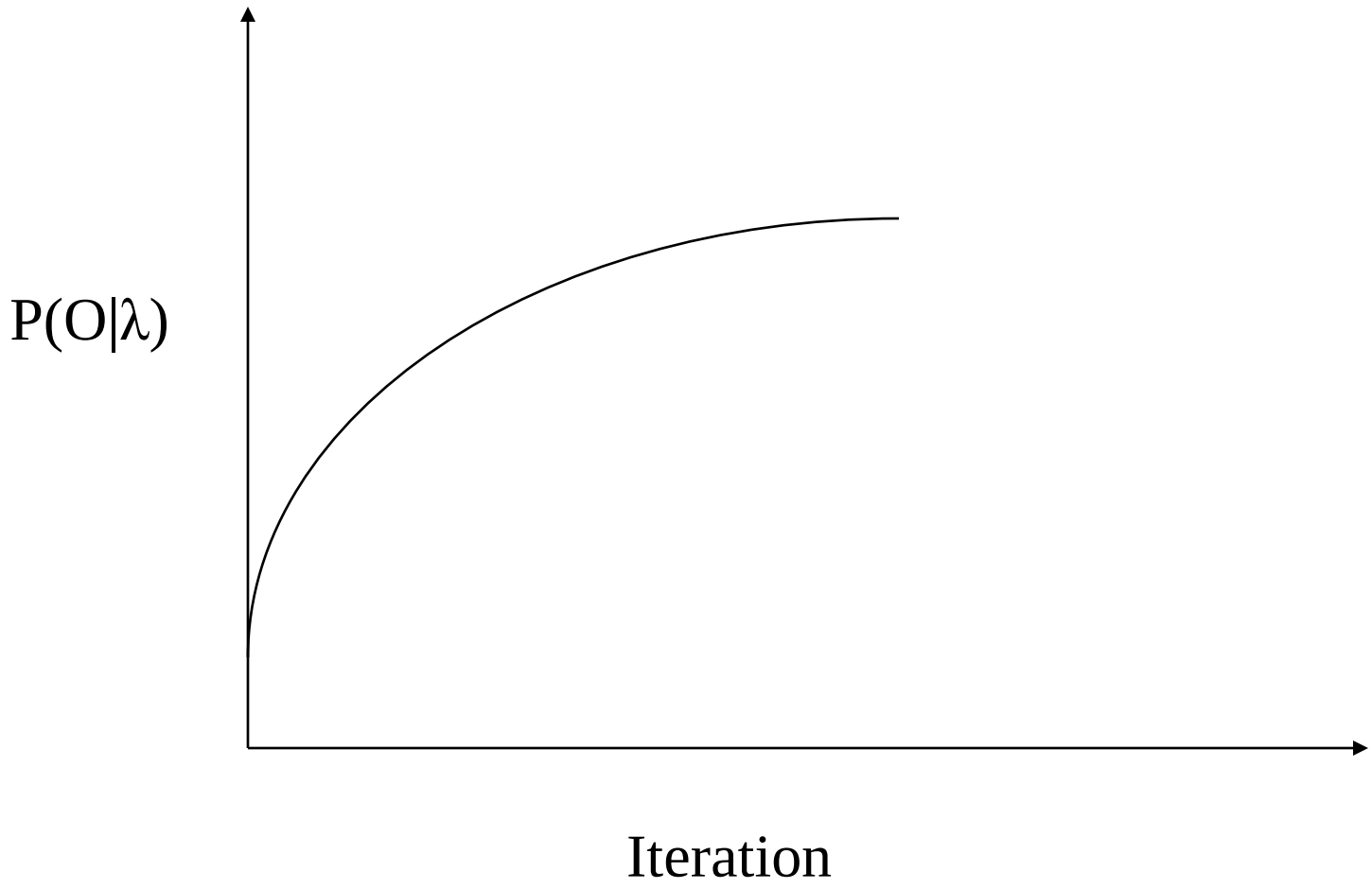
The final result of this estimation procedure is called a maximum likelihood estimate of the HMM. (local maxima)

Local Optimal Guaranteed

If we define the current model as $\lambda=(A,B,\pi)$ and use it to compute the new model $\bar{\lambda}=(A,B,\pi)$, it has been proven by Baum et al. that either 1) initial model λ defines a critical point, in which case $\bar{\lambda} = \lambda$; or 2) $\bar{\lambda}$ is more likely than λ in the sense that $P(O|\bar{\lambda}) > P(O|\lambda)$, i.e., we have found a new model $\bar{\lambda}$ from which the observation sequence is more likely to have been produced.



Likelihood monotonically crease per iteration until it converges to local maxima.



Likelihood monotonically increases per iteration until it converges to local maxima. It usually converges very fast in several iterations.

Another Way to Interpret the Re-estimation Formulas

The re-estimation formulas can be derived by maximizing (using standard constrained optimization techniques) Baum's auxiliary function (auxiliary variable Q):

$$Q(\lambda, \overline{\lambda}) = \sum_{Q} P(Q \mid O, \lambda) \log[P(O, Q) \mid \overline{\lambda})$$

It has been proven that the maximization of the auxiliary function leads to increased likelihood, i.e.

$$\max_{\overline{\lambda}}[Q(\lambda, \overline{\lambda})] \Rightarrow P(O|\overline{\lambda}) \geq P(O|\lambda)$$

The selection Q is problem-specific.

Relation to EM algorithm

- E (expectation) step is the computation of the auxiliary function $Q(\lambda, \overline{\lambda})$ (*averaging*).
- M (maximization) step is the maximization over $\overline{\lambda}$.
- Thus Baum-Welch re-estimation is essentially identical to the EM steps for this particular problem.

Insights

- The stochastic constraints of the HMM parameters, namely

$$\sum_{i=1}^{N} \overline{\pi}_i = 1 \qquad \sum_{j=1}^{N} \overline{a}_{ij} = 1, 1 \le i \le N \qquad \sum_{i=1}^{M} \overline{b}_j(k) = 1, 1 \le j \le N$$

are automatically satisfied at each iteration. By looking at the parameter estimation problem as *constrained optimization* of P(O| λ) subject to the above constraints, the techniques of of Lagrange Multipliers can be used to find the values of π_i, a_{ij}, and $b_j(k)$ which maximize P (use P to denote P(O| λ). The same results as Baum-Welch's formula's will be reached. Comments: since the entire problem can be set up as an optimization problem, *standard gradient techniques* can be used to solve for "optimal" value too.

Types of HMM

Ergodic model has property that every state can be reached from every other state.

Left-right model: model has the properties that as time increases the state index increases (or stay the same), i.e., the states proceeds from left to right. (model signals whose properties change over time (speech or biological sequence).

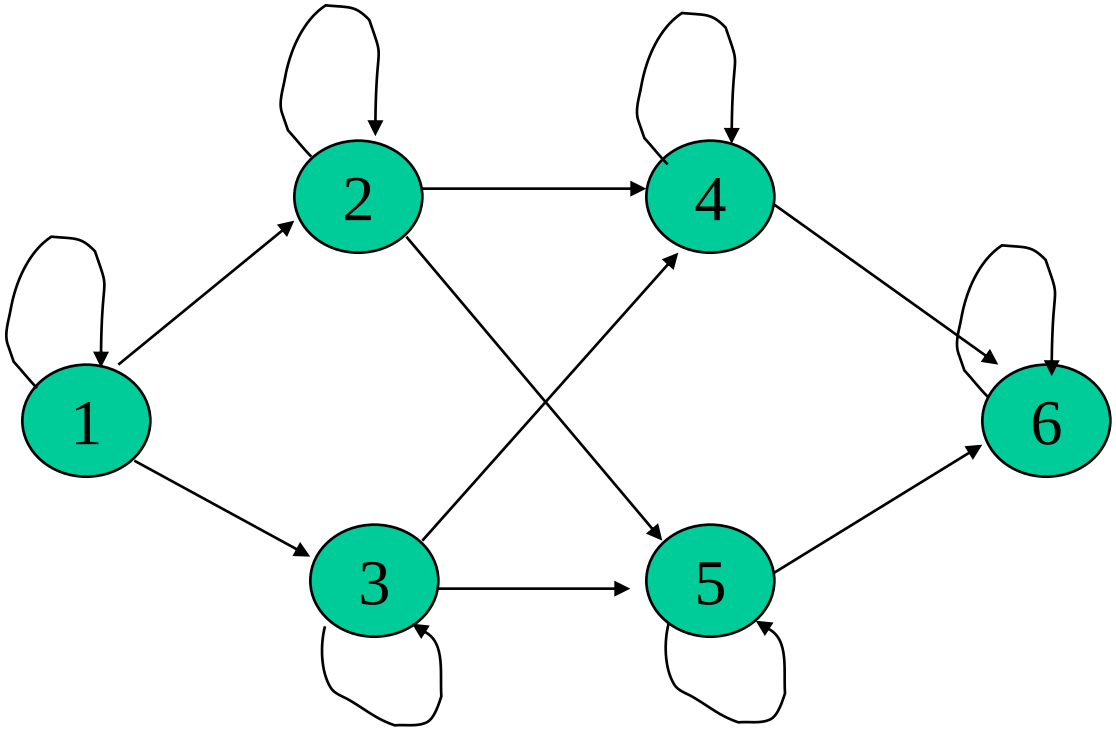
Property 1: $a_{ij} = 0, i > j.$

Property 2: $\pi_1 = 1, \pi_i = 0 (i \neq 1).$

Optional constraints: $a_{ij} = 0, j > i + \Delta$, make sure large change of state indices are not allowed.

For the last state in a left-right model, $a_{NN} = 1, a_{Ni} = 0, i < N.$

One Example of Left-Right HMM



Imposition of constraints of the left-right model essentially has no effect on the re-estimation procedure because any HMM parameter set to zero initially, will remain at zero.

HMM for Continuous Signals

- In order to use a continuous observation density, some restrictions have to be placed on the form of the model probability density function (pdf) to insure the parameters of the pdf can be re-estimated in a consistent way. Use a finite mixture of Gaussian distributions are used. (see the Rabiner's paper for more details).

Implementation Issue 1: Scaling

- To compute $a_t(i)$ and $b_t(i)$, Multiplication of a large number of terms (probability), value heads to 0 quickly, which exceed the precision range of any machine.
- The basic procedure is to multiply them by a scaling coefficient that is independent of i (i.e., it depends only on t). Logarithm cannot be used because of summation. But we can use $c_t = \frac{1}{\sum_{i=1}^{N} a_t(i)}$

  C_t will be stored for the time points when the scaling is performed. C_t is used for both $a_t(i)$ and $b_t(i)$. The scaling factor will be canceled out for parameter estimation.
- For Viterbi algorithm, use logarithm is ok.

Implementation Issue 2: Multiple Observation Sequence

- Denote a set of K observation sequences as $O = [O^{(1)},O^{(2)},\ldots,O^{(k)}]$. Assume the observed sequences are independent.
- The re-estimation of formulas for multiple sequences are modified by adding together the individual frequencies for each sequence.

- Adjust the parameter of model λ to maximize:

$$P(O \mid \lambda) = \prod_{k=1}^{K} P(O^{(k)} \mid \lambda) = \prod_{k=1}^{K} P_k$$

$$\overline{b_j(l)} = \frac{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{t=1, s.t. O_t = v_l}^{T_k} a_t^k(i)\beta_t^k(i)}{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{t=1}^{T_k} a_t^k(i)\beta_t^k(i)}$$

$$\overline{a_{ij}} = \frac{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{t=1}^{T_k - 1} a_t^k(i) a_{ij} b_j(O_{t+1}^k) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{t=1}^{T_k - 1} a_t^k(i)\beta_t^k(i)}$$

π_i is not re-estimated since $\pi_1 = 1$, $\pi_i = 0$, $i \neq 1$ for left-right HMM.

Initial Estimate of HMM Parameters

- Random or uniform initialization of π and A is appropriate.
- For emission probability matrix B, good estimate is helpful in discrete HMM, and essential in continuous HMM. Initially segment sequences into states and then compute empirical emission probability.
- Initialization (or using prior) is important when there is no sufficient data (pseudo counts, Dirichilet Prior).