Correlated And Time-Varying Events

Correlated And Time-Varying Events

- So far, events and statistics are
  - independent
  - time (space) invariant
- Many applications violate the above
  - speech signals
  - images (textures)
- Need a more sophisticated model

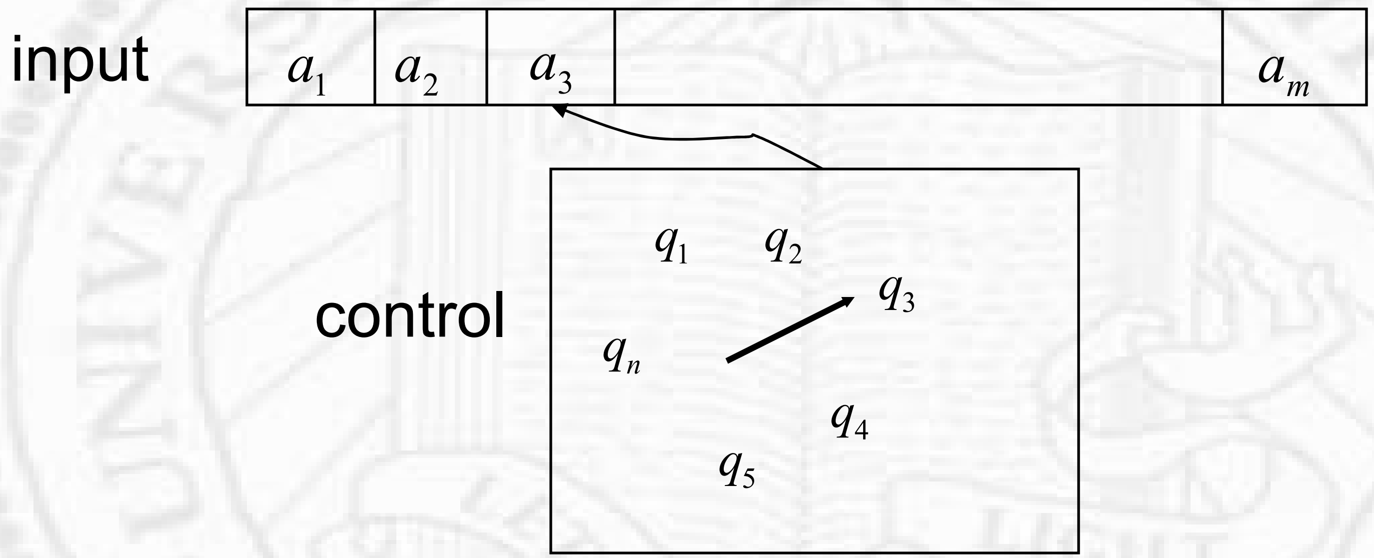
Possibilities

- This problem (and its variants) has been studied in many fields over a long period of time
- We will briefly discuss
  - Finite automata (computer science, syntactic pattern recognition)
  - Markov chain (mathematics, statistics)
  - Hidden Markov model (signal processing, speech processing)

Finite Automata

- As a language (or more generally, a *pattern*) ***recognition*** device
- A machine with a *fixed* number of states (memory)
- It parses a string (a *pattern*) and based on the current state and current input to decide on the next state
- Eventually, when the string (the *pattern*) is exhausted, the machine will halt and either accept (recognize) or reject (not recognize) the input

Finite Automata (cont.)



Finite Automata (cont.)

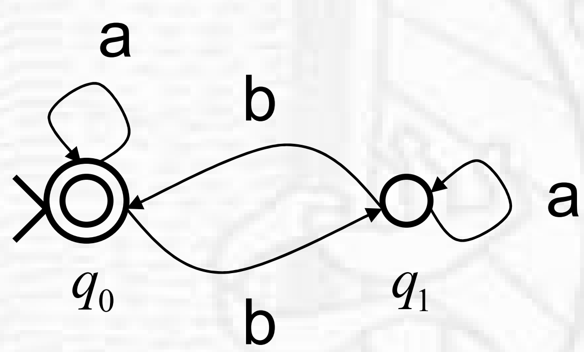
- A deterministic finite automata is a quintuple

$$M = (K, \Sigma, \delta, s, F)$$

  - K is a finite set of states
  - Σ is an alphabet
  - $s \in K$ is the initial state
  - $F \subset K$ is the set of final states
  - $\delta(K \times \Sigma) \rightarrow K$ transition function

Example

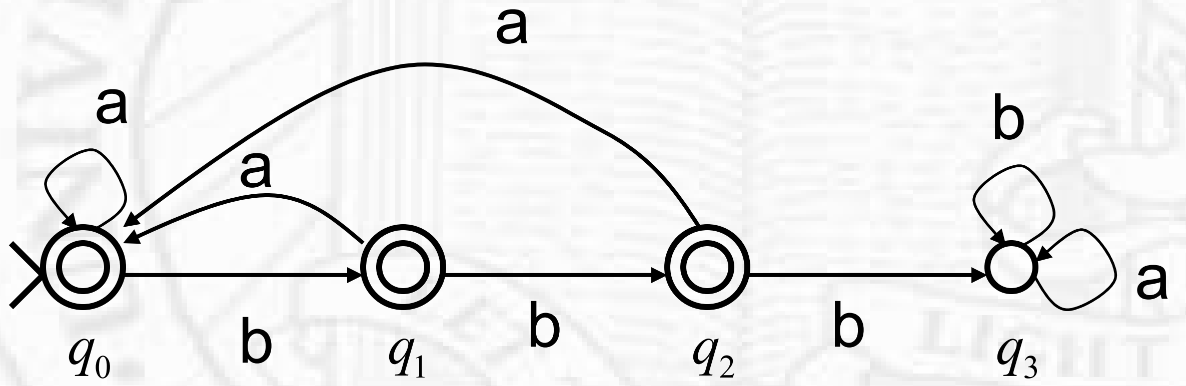
- A finite automata that accepts all strings in $\{a,b\}^*$ that have an even number of b's
  - alphabet: $\{a,b\}$
  - state: $\{q_o, q_1\}$
  - initial state: q_o
  - final states: $\{q_o\}$
  - transition rules



q	σ	$\delta(q,\sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

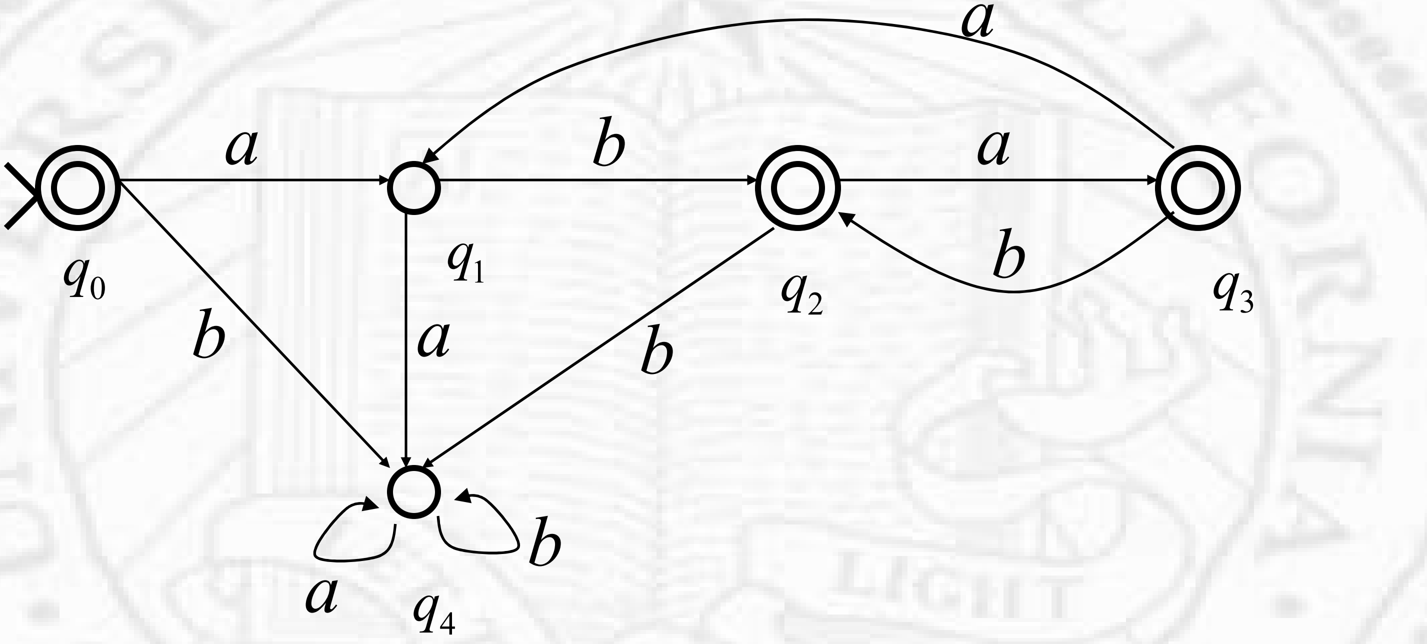
More Example

- A finite automata that accepts strings over {a,b}* that does not contain three consecutive b's



More Example

- A finite automata that accepts $L = (ab \cup aba)^*$



An Opposite Definition

- Grammar: as a language (or pattern) ***generation*** device
- A grammar is a quadruple
  - V an alphabet
  - Σ subset of V (terminals)
  - $R:(V-\Sigma)\rightarrow V^*$ rewriting rules
  - S start symbol

An Opposite Definition

- A language
  - all the strings (patterns) that can be generated by applying the rewriting rules from the start symbol

$$L(G) = \{w \mid w \in \Sigma^*, S \overset{*}{\Rightarrow} w\}$$

Regular Grammar

- The rewriting rule must be

$$R:(V-\Sigma)\rightarrow\Sigma^{*}((V-\Sigma)\cup\{e\})$$

At most one non-terminal on the right-hand side
Non-terminal, if present, must be the last symbol in the string

Example

$R:$

$S \to aMb$

$M \to A$

$M \to B \qquad \longrightarrow \qquad a(a^* \cup b^*)b$

$A \to aA$

$A \to e$

$B \to bB$

$B \to e$

Equivalency

- A type of automata (as a ***recognition*** device) can be made to recognize languages generated by a particular grammar (as a ***generation*** device)
- E.g., the class of languages accepted by finite automata is exactly the class of languages generated by regular grammars

Generalization

- Non-deterministic automata (allows multiple transitions out of a state)
- Pushdown automata (context free languages)
- Stochastic grammar
- The whole area of syntactic pattern recognition
- etc. etc.

String (Pattern) Grammar

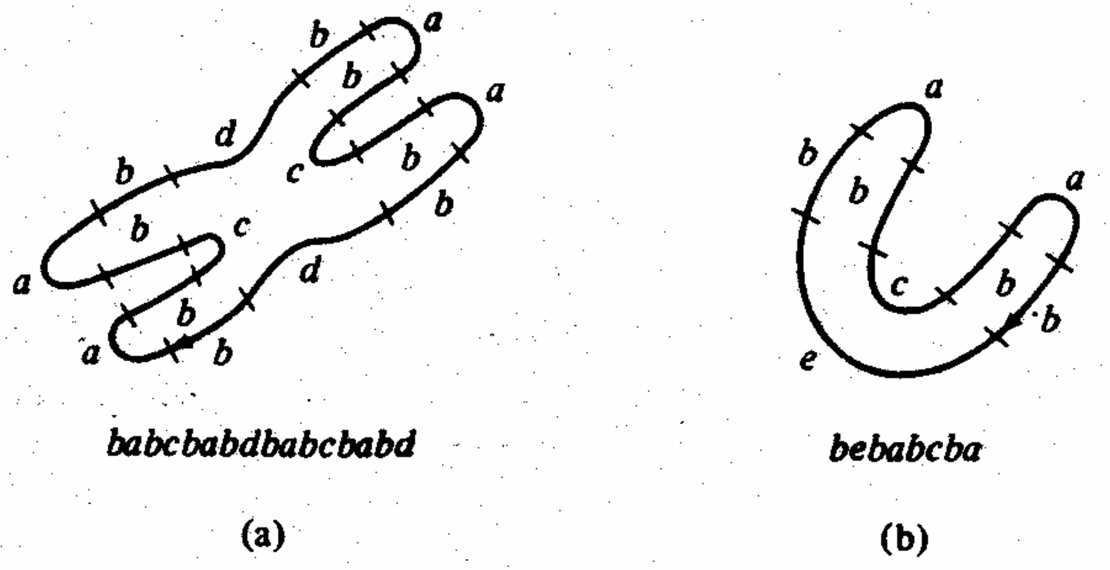
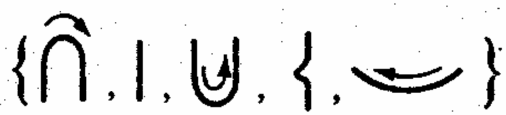


Figure 3.19. (a) Submedium chromosome; (b) telocentric chromosome.

$$G = (V_N, V_T, P, \{\langle\text{submedian chromosome}\rangle, \langle\text{telocentric chromosome}\rangle\})$$

where

$$V_N = \{\langle\text{submedian chromosome}\rangle, \langle\text{telocentric chromosome}\rangle, \langle\text{arm pair}\rangle,$$
$$\langle\text{left part}\rangle, \langle\text{right part}\rangle, \langle\text{arm}\rangle, \langle\text{side}\rangle, \langle\text{bottom}\rangle\}$$

$V_T =$ 

a *b* *c* *d* *e*

and P:

$$\langle\text{submedian chromosome}\rangle \longrightarrow \langle\text{arm pair}\rangle\,\langle\text{arm pair}\rangle$$
$$\langle\text{telocentric chromosome}\rangle \longrightarrow \langle\text{bottom}\rangle\,\langle\text{arm pair}\rangle$$
$$\langle\text{arm pair}\rangle \longrightarrow \langle\text{side}\rangle\,\langle\text{arm pair}\rangle$$

$$\langle \text{arm pair} \rangle \longrightarrow \langle \text{arm pair} \rangle \langle \text{side} \rangle$$
$$\langle \text{arm pair} \rangle \longrightarrow \langle \text{arm} \rangle \langle \text{right part} \rangle$$
$$\langle \text{arm pair} \rangle \longrightarrow \langle \text{left part} \rangle \langle \text{arm} \rangle$$
$$\langle \text{left part} \rangle \longrightarrow \langle \text{arm} \rangle\, c$$
$$\langle \text{right part} \rangle \longrightarrow c\, \langle \text{arm} \rangle$$
$$\langle \text{bottom} \rangle \longrightarrow b\, \langle \text{bottom} \rangle$$
$$\langle \text{bottom} \rangle \longrightarrow \langle \text{bottom} \rangle\, b$$
$$\langle \text{bottom} \rangle \longrightarrow e$$
$$\langle \text{side} \rangle \longrightarrow b\, \langle \text{side} \rangle$$
$$\langle \text{side} \rangle \longrightarrow \langle \text{side} \rangle\, b$$
$$\langle \text{side} \rangle \longrightarrow b$$
$$\langle \text{side} \rangle \longrightarrow d$$
$$\langle \text{arm} \rangle \longrightarrow b\, \langle \text{arm} \rangle$$
$$\langle \text{arm} \rangle \longrightarrow \langle \text{arm} \rangle\, b$$
$$\langle \text{arm} \rangle \longrightarrow a$$

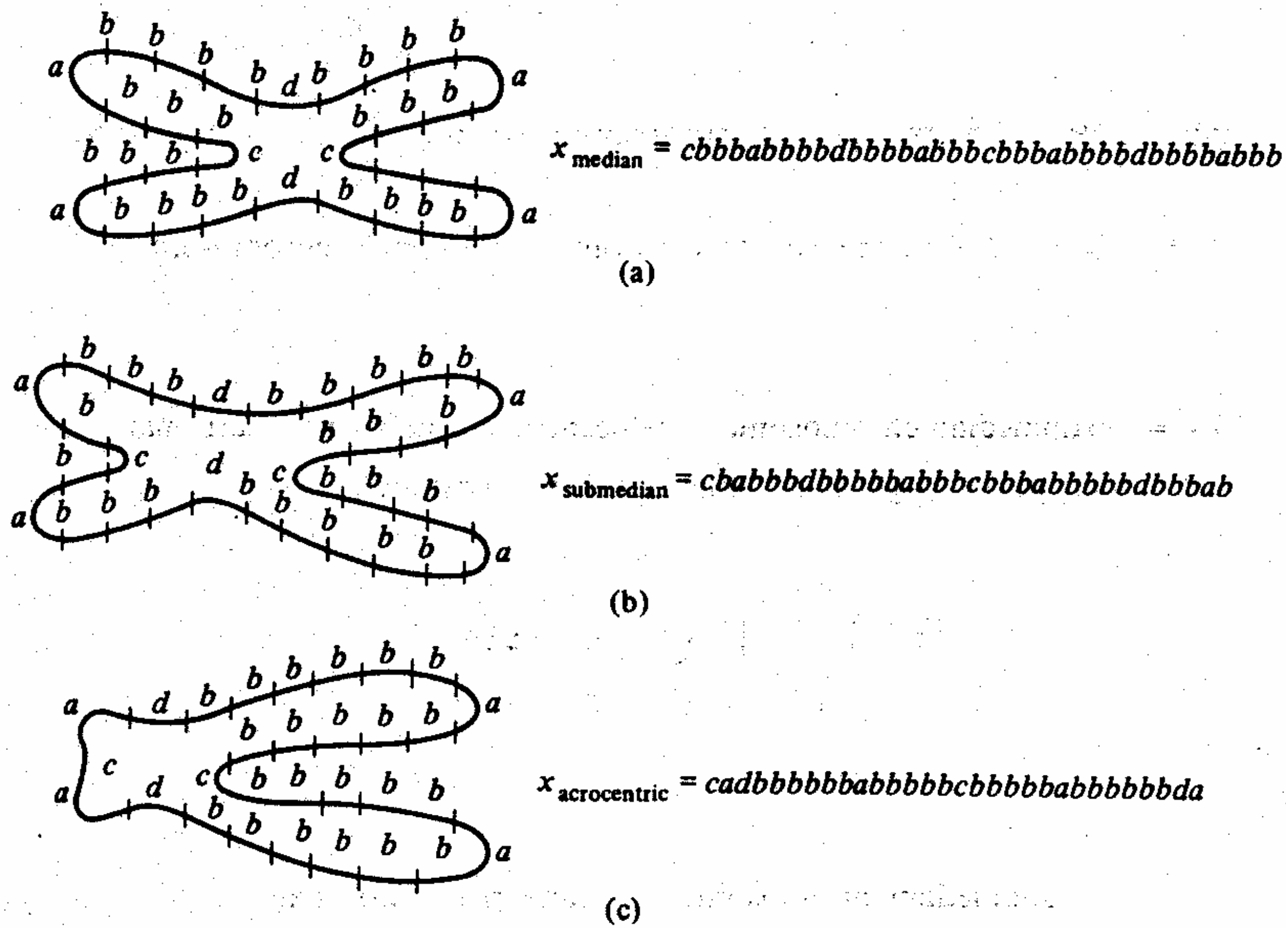


Figure 3.20. Three sample chromosomes: (a) median string representation; (b) submedian representation; (c) arcocentric string representation.

Tree Grammar

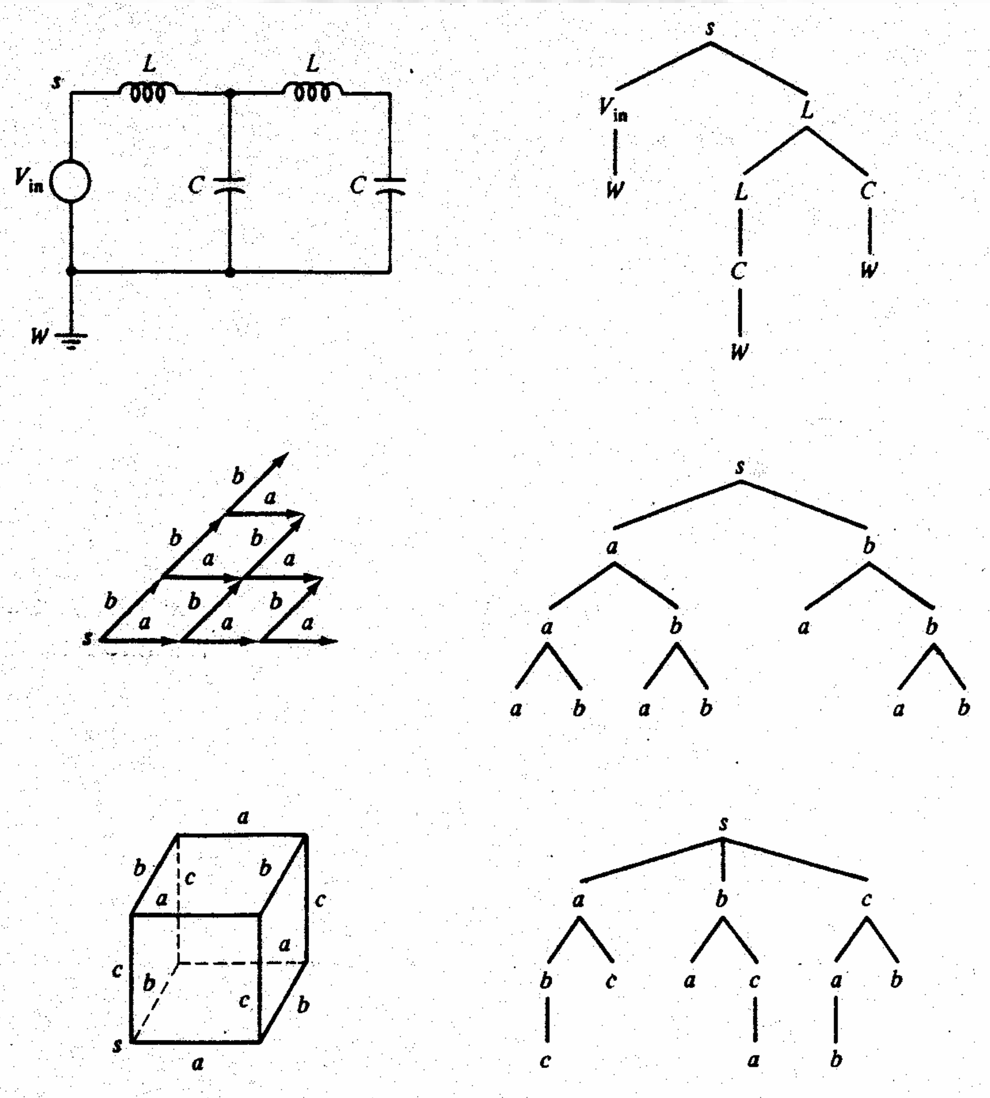


Figure 4.6. (a) Patterns; (b) corresponding tree representations.

Difficulty

- How to extract primitives reliably from images?
- How to parse the the extracted primitives?
- How to correct errors?

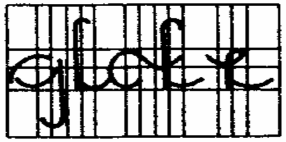


Figure 3.2. Cursive strokes of the word "globe."

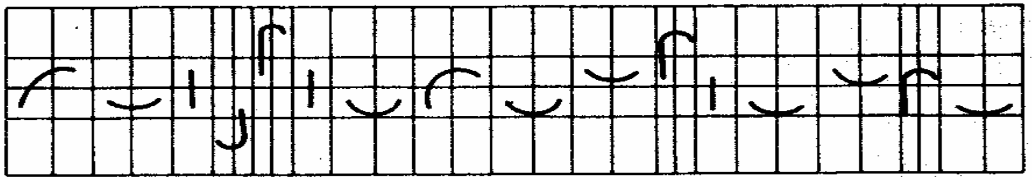
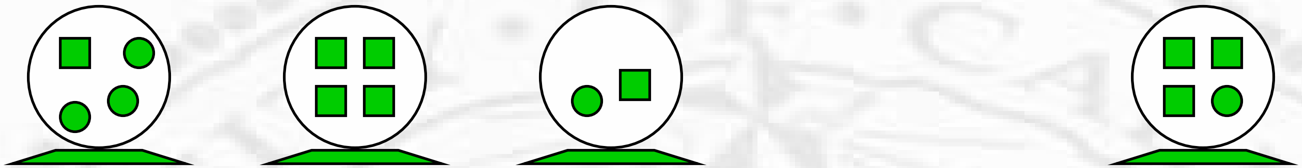


Figure 3.3. Stroke-sequence representation of the word "globe."

Hidden Markov Model

- *Internal states*: short-time, steady, well-behaved part of a signal or an image
- *State transition*: characterize how one state evolves into the next
- *Initial states*: the starting point of evolution
- "Hidden" implies that the states are usually not directly observable, but will influence the behavior of the model

An example - urn-and-ball-model

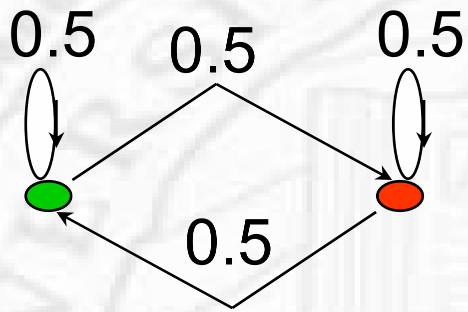


urn1	urn2	urn3	urnN
P(R)=.	P(R)=.	P(R)=.	P(R)=.
P(B)=.	P(B)=.	P(B)=.	P(B)=.
...	...	...	...

Time	1	2	3	...	T
urn (hidden) state	u1	u9	u5	...	u7
ball (observation)	R	B	Y	...	G

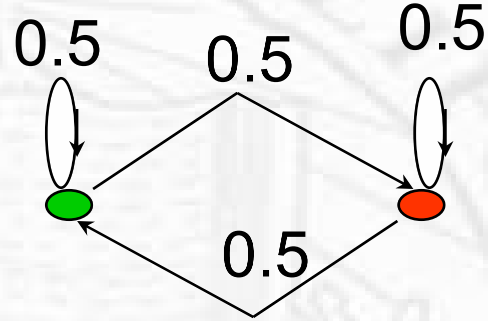
Another example - competing HMMs

2-fair coins



P(H)=0.5 P(H)=0.5
P(T)=0.5 P(T)=0.5

2-biased coins



P(H)=0.8 P(H)=0.2
P(T)=0.2 P(T)=0.8

Observation: HTTHTHH ….

1. Which model best describes the observation?
2. What are the most likely state transitions over time?

Terminology

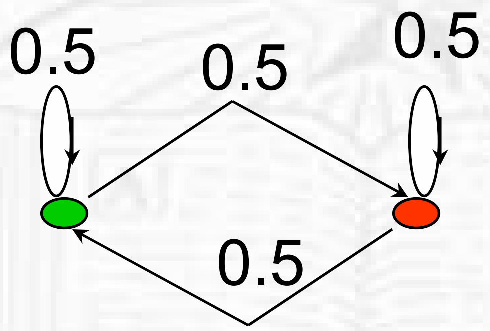
- T: length of the observation sequence
- N: number of states (urns) in the model
- M: number of observation symbols (colors)
- O: observations (colors) $\{O_1, O_2, \ldots, O_T\}$
- Q: states (urns) $\{q_1, q_2, \ldots, q_N\}$
- V: set of possible symbols $\{v_1, v_2, \ldots, v_M\}$
- A: state transition density $a_{ij} = p(q_j^{t+1} | q_i^t)$
- B: observation symbol density in a state $b_j(k) = p(v_k^t | q_j^t)$
- π : initial state distribution $\pi_i = P(q_i^1)$

Three Problems of HMMs

- *Evaluation*: Given an observation sequence O, and a model $\lambda = (A, B, \pi)$ how to compute $P(O|\lambda)$
- *Estimation*: Given an observation sequence O, and a model $\lambda = (A, B, \pi)$ how to choose a state sequence $I = i_1, i_2, \ldots, i_T$ which is optimal
- *Training*: How to adjust the model parameters to maximize $\lambda = (A, B, \pi)$ $P(O|\lambda)$

Example - Evaluation

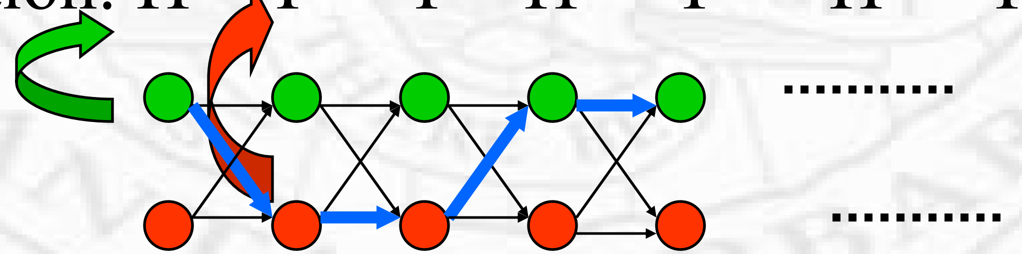
2-biased coins



P(H)=0.8 P(H)=0.2

P(T)=0.2 P(T)=0.8

Observation: H T T H T H H ….



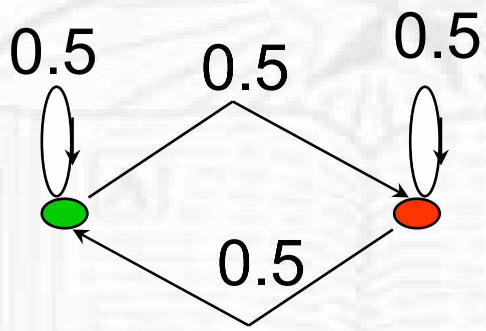
How good can the states explain the output?

Intuition: Given a Model …

- A good (or likely) sequence of states is one that
  - Each state can well explain the corresponding observation
  - Each transition has high likelihood to happen
- A bad (or unlikely) sequence of states does not have either (or both) properties
- A probabilistic framework
  - Enumerate all possible state transitions in a model to determine how good is a model

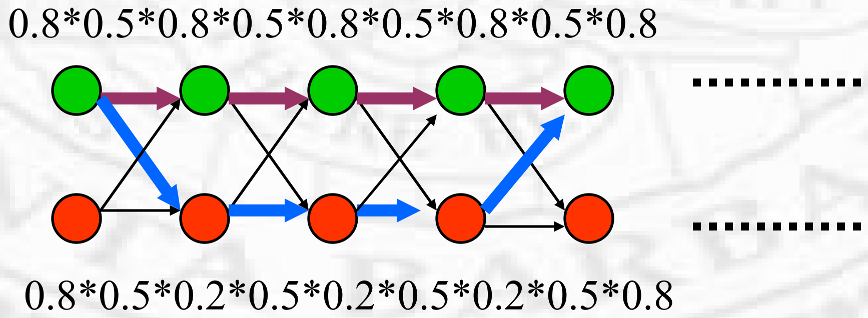
Example - Evaluation

2-biased coins



P(H)=0.8 P(H)=0.2
P(T)=0.2 P(T)=0.8

Observation: H H H H H H H ….



Evaluation

- Enumerates all possible state transitions in T steps
- For each possible state sequence, compute the possibility of observing O

$$P(O|\lambda)=\sum_{all I}P(O,I|\lambda)=\sum_{all I}p(O|I,\lambda)p(I|\lambda)$$

$$P(O|I,\lambda)=b_{i_1}(O_1)b_{i_2}(O_2)...b_{i_T}(O_T)$$

$$p(I|\lambda)=\pi_{i_1}a_{i_1 i_2}a_{i_2 i_3}...a_{i_{T-1}i_T}$$

$$P(O|\lambda)=\sum_{all I}\pi_{i_1}b_{i_1}(O_1)a_{i_1 i_2}b_{i_2}(O_2)a_{i_2 i_3}...a_{i_{T-1}i_T}b_{i_T}(O_T)$$

Graphical Interpretation

- Brute force method
  - Every leaf node requires a full path evaluation
  - Interior edges are traversed multiple times

Level, # of nodes

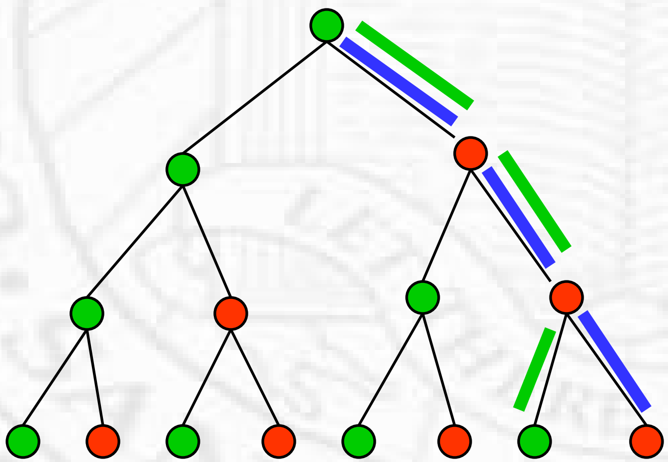
0, 1

1, 2

2, 4

3, 8

k, 2^k



$O(2^k \times (2k-1))$

- Dynamic Programming
  - Remember partial results
  - Interior edges are traversed only once
  - Can be made more efficient

Level, # of nodes

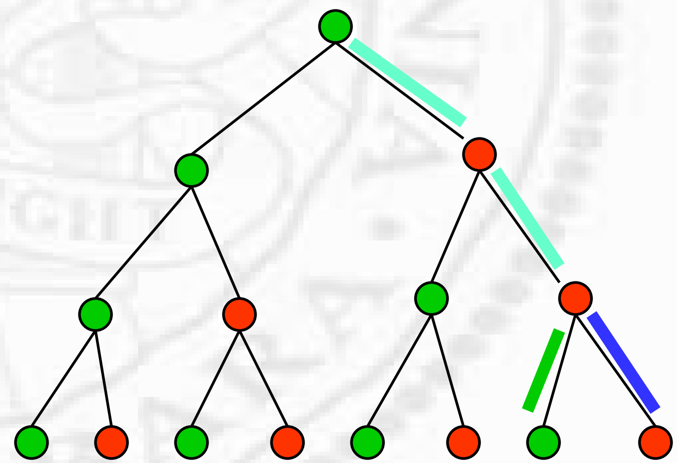
0, 1

1, 2

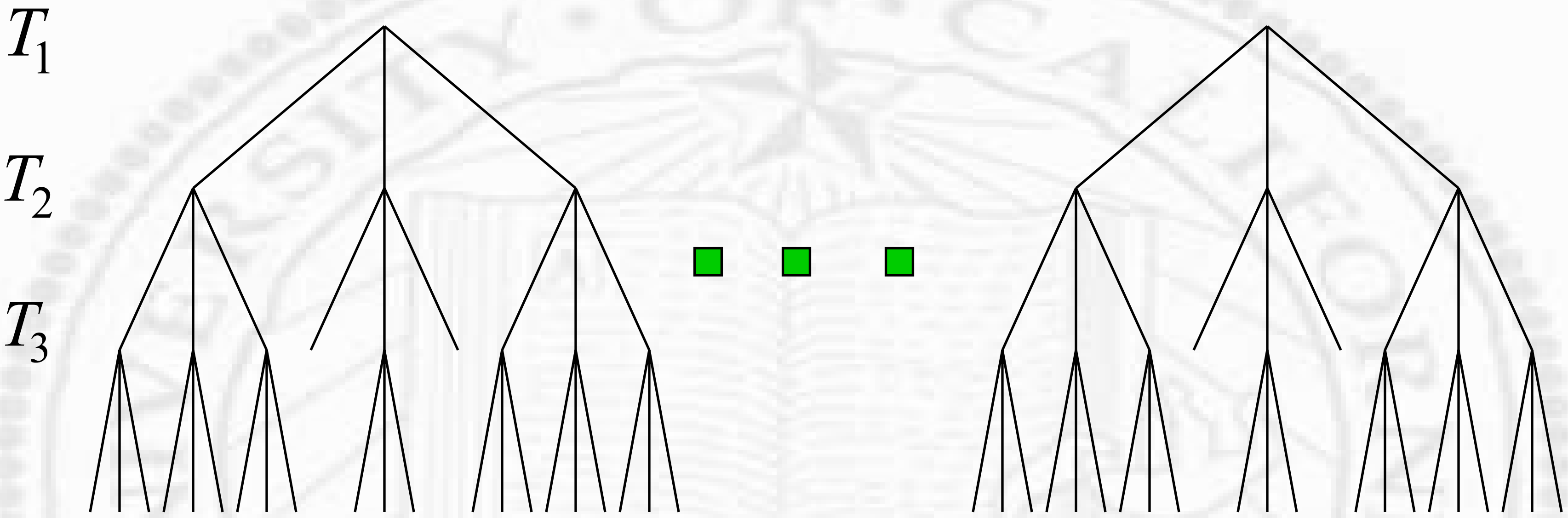
2, 4

3, 8

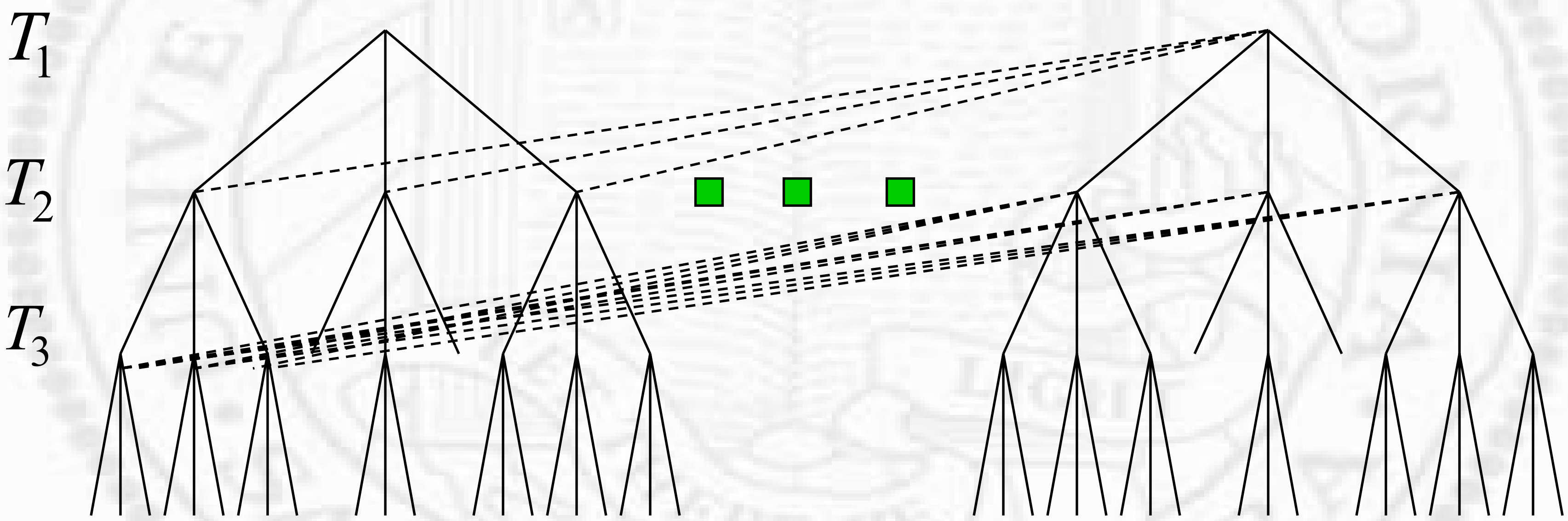
k, 2^k



- N states and T steps require computation $O(N^T)$

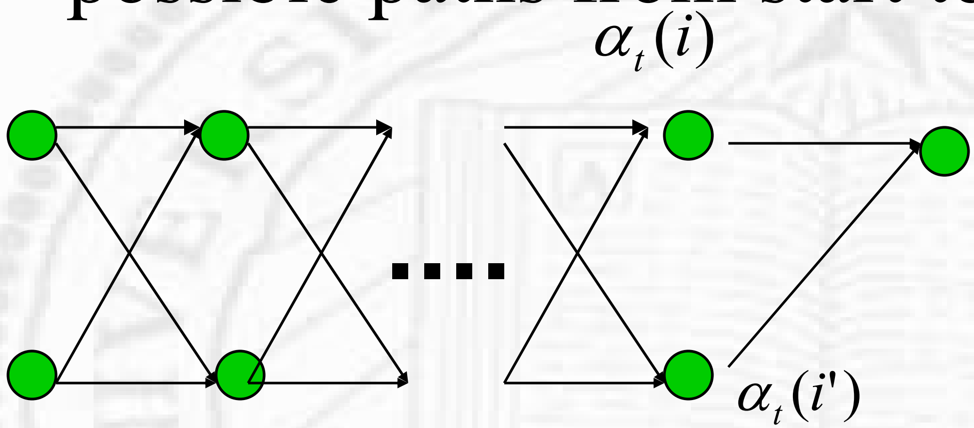


- Efficient forward -backward procedure (based on dynamic programming)
- Remember *all* costs



Intuition

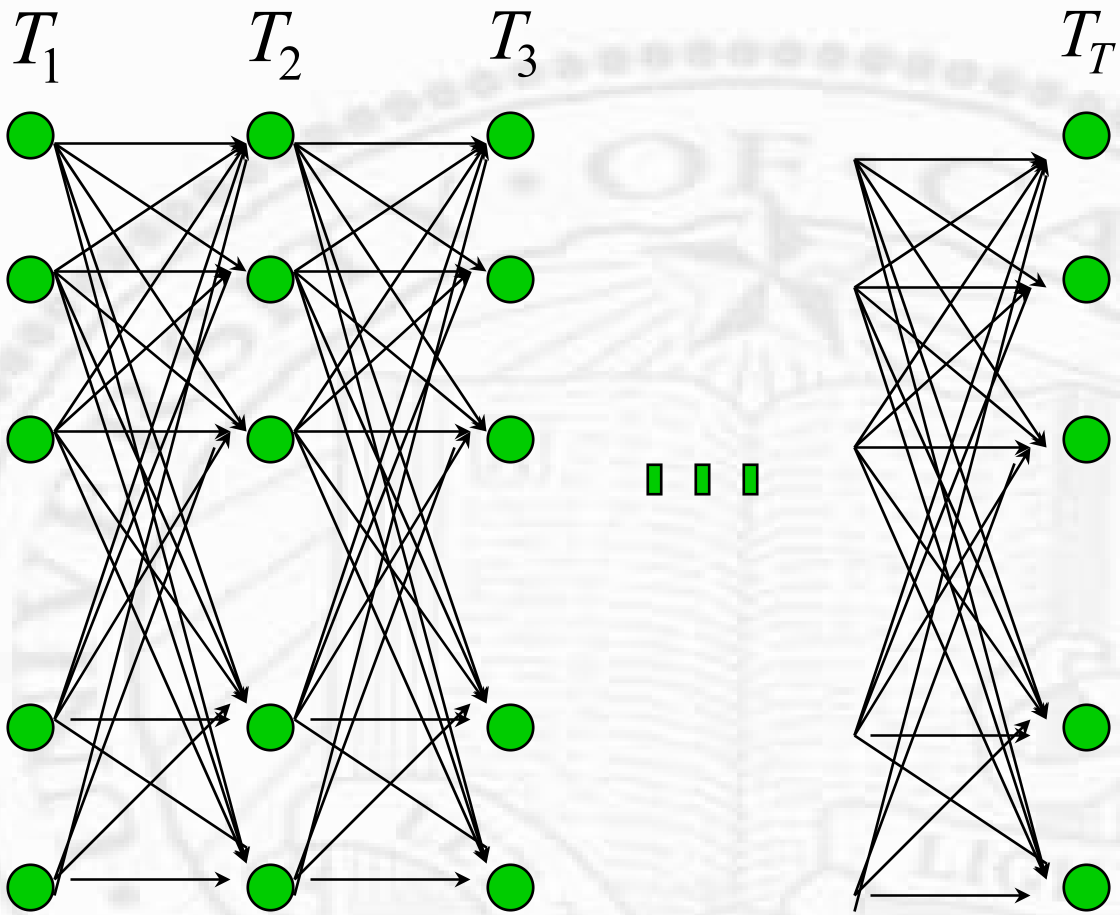
- At each node, the cost is the sum of all possible paths from start to that node



$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{n} a_t(i) a_{ij} \right] b_j(O_{t+1})$$

- To proceed one more stage (*t* to *t+1*), the total cost is
  - Total cost at stage *t*
  - Transition cost from *t* to *t+1*
  - Observation cost at stage *t+1*

Extending all paths at once!



N states per stage/Tstages

- Forward approach

$$\alpha_t(i) = p(O_1, O_2, \ldots, O_t, i_t = q_i | \lambda)$$

$$1.\ \alpha_1(i) = \pi_i b_i(O_1) \qquad 1 \le i \le N$$

$$2.\ for\ \ t = 1,2,\ldots,T-1 \qquad 1 \le j \le N$$

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

$$3.\ P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

- Backward approach

$$\beta_t(i) = p(O_{t+1}, O_{t+2}, \ldots, O_T | i_t = q_i, \lambda)$$

$$1.\ \beta_T(i) = 1 \qquad 1 \le i \le N$$

$$2.\ for\ \ t = T-1, T-2, \ldots, 1 \qquad 1 \le i \le N$$

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

$$3.\ P(O|\lambda) = \sum_{i=1}^{N} \beta_1(i)$$

Estimation #1

- Optimize each stage *individually*
- *Does not* take into consideration if a path can be built with those individual states
  - Usually fine, but *not* if a_{ij} is zero for some transition
- *Does not* take into consideration the transition costs between those states
- A localized, greedy approach

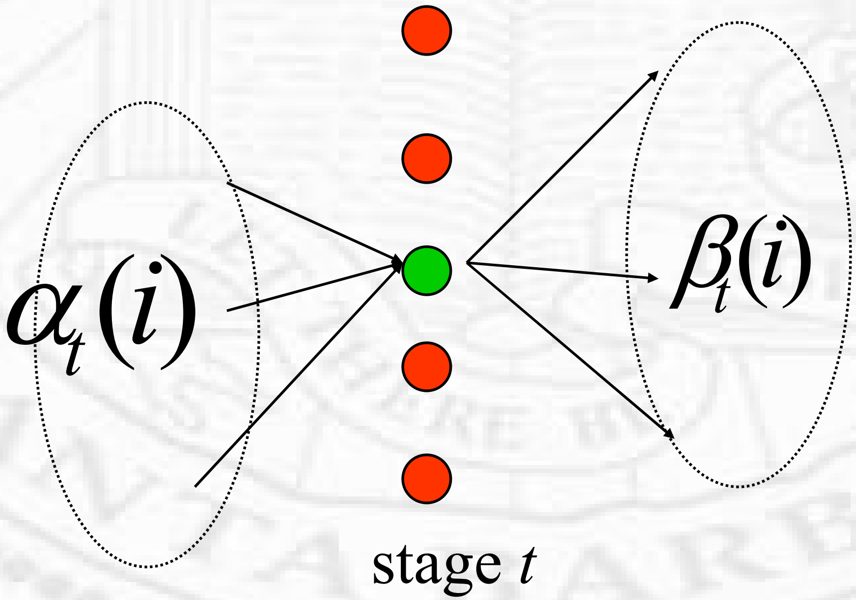
Estimation #1

$$\gamma_t(i) = p(q_t = i \mid O, \lambda) = \frac{p(O, q_t = i, \lambda)}{p(O, \lambda)} = \frac{p(O, q_t = i \mid \lambda)\, p(\lambda)}{p(O \mid \lambda)\, p(\lambda)} = \frac{p(O, q_t = i \mid \lambda)}{p(O \mid \lambda)}$$

$$= \frac{p(O, q_t = i \mid \lambda)}{\sum_{j=1}^{N} p(O, q_t = j \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

All possible ways through stage t via node i

All possible ways through stage t



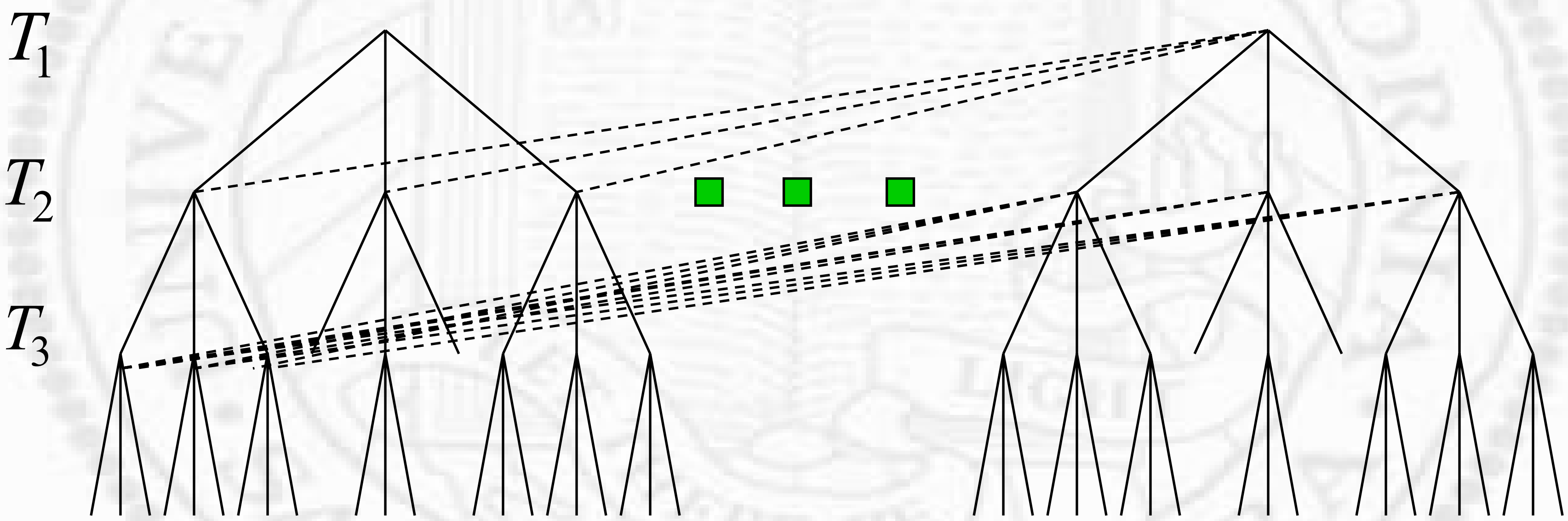
Estimation 2

- Find the best state sequence (path)
  - Viterbi algorithm: a dynamic programming solution for finding the shortest (best) path

$$1.\ \delta_1(i) = \pi_i b_i(O_1) \qquad 1 \le i \le N$$
$$\psi_1(i) = 0$$
$$2.\ \delta_t(j) = \max_{1\le i\le N}[\delta_{t-1}(i)a_{ij}]b_j(O_t) \qquad 1 \le j \le N$$
$$\psi_t(j) = \arg\max_{1\le i\le N}[\delta_{t-1}(i)a_{ij}] \qquad 2 \le t \le T$$
$$3.\ \text{cost} = \max_{1\le i\le N}[\delta_T(i)]$$

path through back tracking on ψ

- Efficient forward -backward procedure (based on dynamic programming)
- Remember the *best* cost instead of *all* cost



Graphical Illustration

Level, # of nodes

0, 1

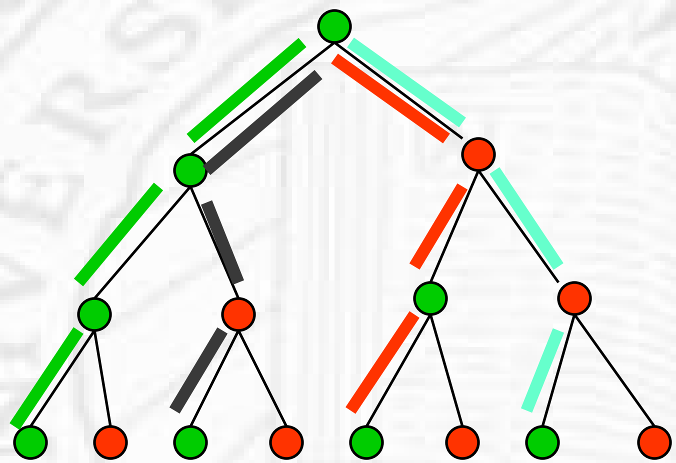
1, 2

2, 4

3, 8

k , 2^k

$O(2^k \times k)$



1,1,1,1 (green)
1,1,2,1 (black)
1,2,1,1 (red)
1,2,2,1 (cyan)

- All go from 1,x,x,1
- Only the best path needs to be remembered

Training

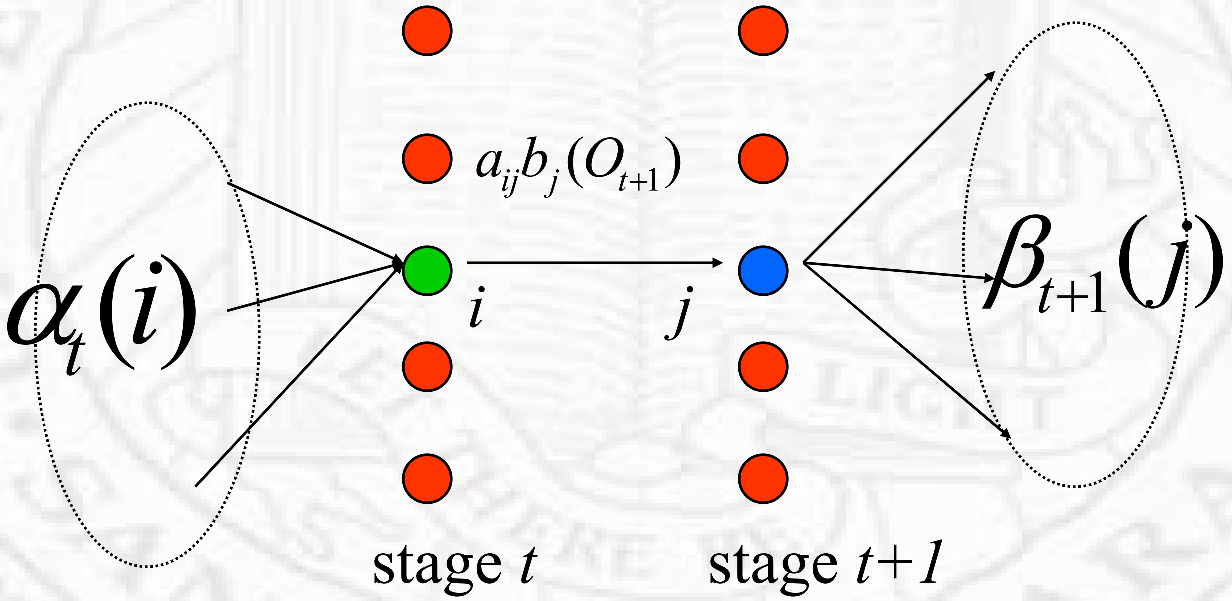
- Observations: All known or partially known
- Structures: known or not known
- Parameters: known or not known
- If we don't know the structure (hence, not parameters either), it is an arbitrarily hard problem
  - You better have some domain knowledge or some hypothesis of the structures, otherwise, you are doomed

Training

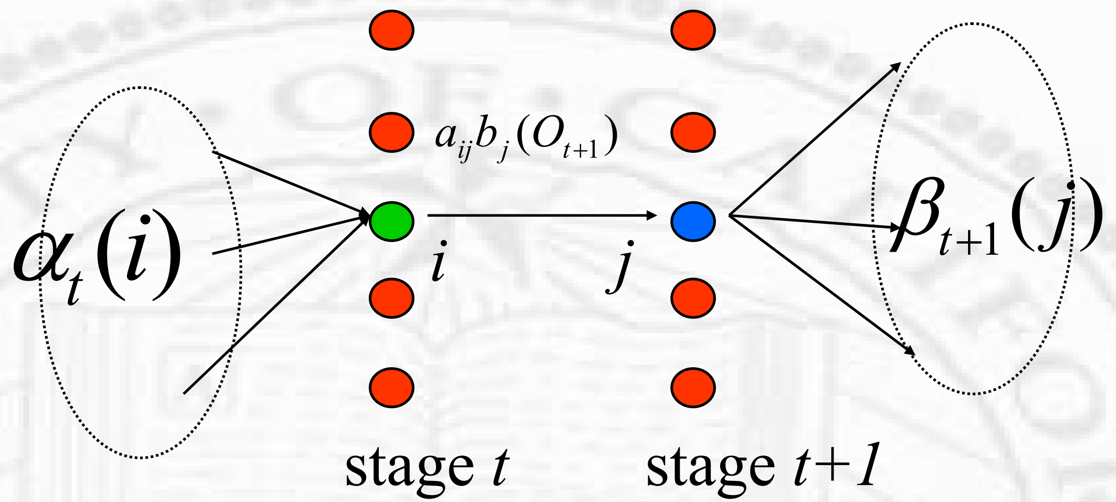
- A much more reasonable assumption
  - Observations are available for training
  - Network structure is known
  - Need to determine or fine tune parameters
    - Initial probability
    - Transition probability
    - Observation probability
  - Formulated as EM optimization

Training: Baum-Welch

- Probability of in state *i* at time *t* and state *j* at time *t+1*



Training: Baum-Welch



$$\xi_t(i,j) = p(i_t = q_i, i_{t+1} = q_j \mid O, \lambda)$$

$$= \frac{p(i_t = q_i, i_{t+1} = q_j, O, \lambda)}{P(O, \lambda)}$$

$$= \frac{p(i_t = q_i, i_{t+1} = q_j, O \mid \lambda)}{P(O \mid \lambda)}$$

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

More definitions

- $\gamma_t(i) = \sum_{j=1}^{n} \xi_t(i,j)$ Being in state *i* at time *t*
- $\sum_{t=1}^{T-1} \gamma_t(i)$ Expected number of transitions made from state q_i
- $\sum_{t=1}^{T-1} \xi_t(i,j)$ Expected number of transitions from state q_i to state q_j

Intuition

- π: expected frequency (# of times) in state i at time $t=1$
- a_{ij}: expected number of transition from i to j/expected transition from I
- $B_j(k)$: expected number of times in state j and observing k/expected number of times in state j

Baum-Welch procedure

- iterate on

$$\hat{\lambda} = (\hat{\pi}, \hat{a}_{ij}, \hat{b}_j(k))$$

$$\hat{\pi}_i = \gamma_1(i)$$

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

$$\hat{b}_i(k) = \frac{\sum_{\substack{t=1 \\ O_t = k}}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

EM Algorithms

Compute

$$\hat{\lambda} = (\hat{\pi}, \hat{a}_{ij}, \hat{b}_j(k))$$

Update

$$\xi_t(i,j) \quad \gamma_t(i)$$