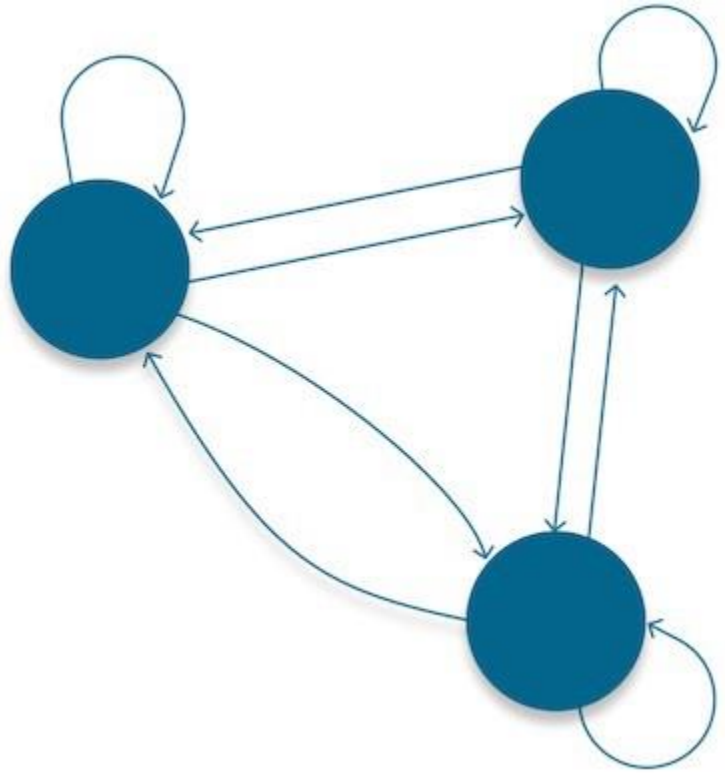
Hidden Markov models for sequential pattern classification

- Slides compiled by Sanghamitra De

Markov chain

- Andrei Markov created a way to describe how random, also called stochastic, systems or processes evolve over time.
- The system is modeled as a sequence of states and, as time goes by, it moves in between states with a specific probability.
- Since the states are connected, they form a *chain*.
- This way of modeling the world is called a *Markov Chain*.
- In Markov chains, as you move along the chain, the state where you are at any given time matters.
- The transitions between states are *conditioned*, or *dependent*, on the state you are in before the transition occurs.
- Putting all of these characteristics together, Markov was able to prove that, as long as you can reach all states in the chain, the probability of moving to a particular state will converge to a single steady value in the long run.

Examples of stochastic processes: growth of a bacterial population, an electrical current fluctuating due to thermal noise, or the movement of a gas molecule, Brownian motion.



Example of a Markov chain.

Markov chain

- A Markov chain is simplest type of Markov model, where all states are observable and probabilities converge over time.
- Hidden Markov Models are similar to Markov chains, but they have a few *hidden* states.
- Since they're hidden, you can't see them directly in the chain, only through the observation of another process that depends on it.
- Shannon used *Markov chains* to model the English language as a sequence of letters that have a certain degree of randomness and dependencies between each other.
- In the end, this Markov model was able to produce English-like text.

Markov chain

- Markov models:
  - ✓ Describe the world in a more realistic way,
  - ✓ Are a useful tool to make long-term predictions about a system or process.
- Since Markov models describe the behavior over time, you can use them to ask different questions about the future state of the system:
  - ✓ **How it evolves over time**: In what state is the system going to be in N time steps?
  - ✓ **Tracing possible sequences in the process**: When the system goes from *State A* to *State B* in N steps, what is the likelihood that it follows a specific path p?

Markov chain

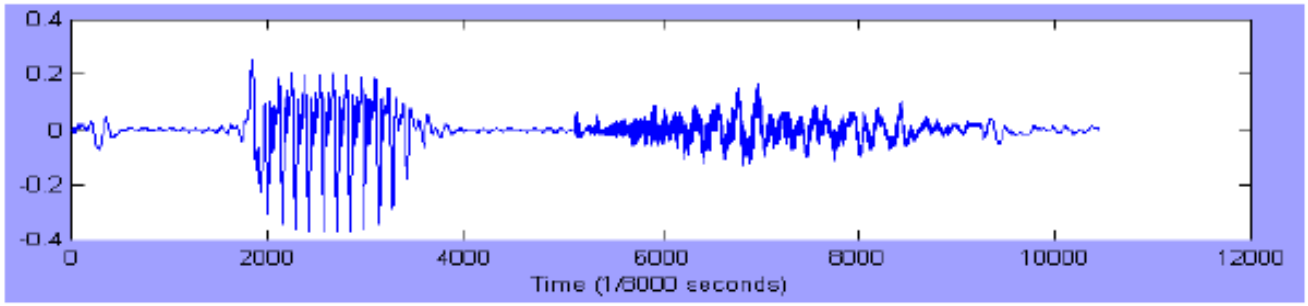
- A Markov chain has short-term memory, it only remembers where you are now and where you want to go next.
- This means the path you took to reach a particular state doesn’t impact the likelihood of moving to another state.
- The only thing that can influence the likelihood of going from one state to the other is the state you are currently in.
- For the states in the Markov Chain, one can encode the dependencies between states, using conditional probabilities.
- In the context of Markov models, these conditional probabilities are called **transition probabilities**.

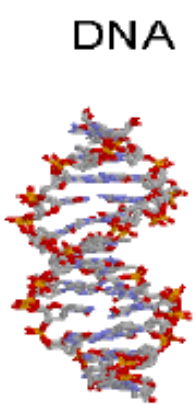
Markov chain

- That is, Transition probabilities describe the transition between states in the chain as a conditional probability.
- Put in mathematical notation, these probabilities can be represented as a **transition matrix**.

Sequential Data

- Data are sequentially generated according to time or index
- Spatial information along time or index

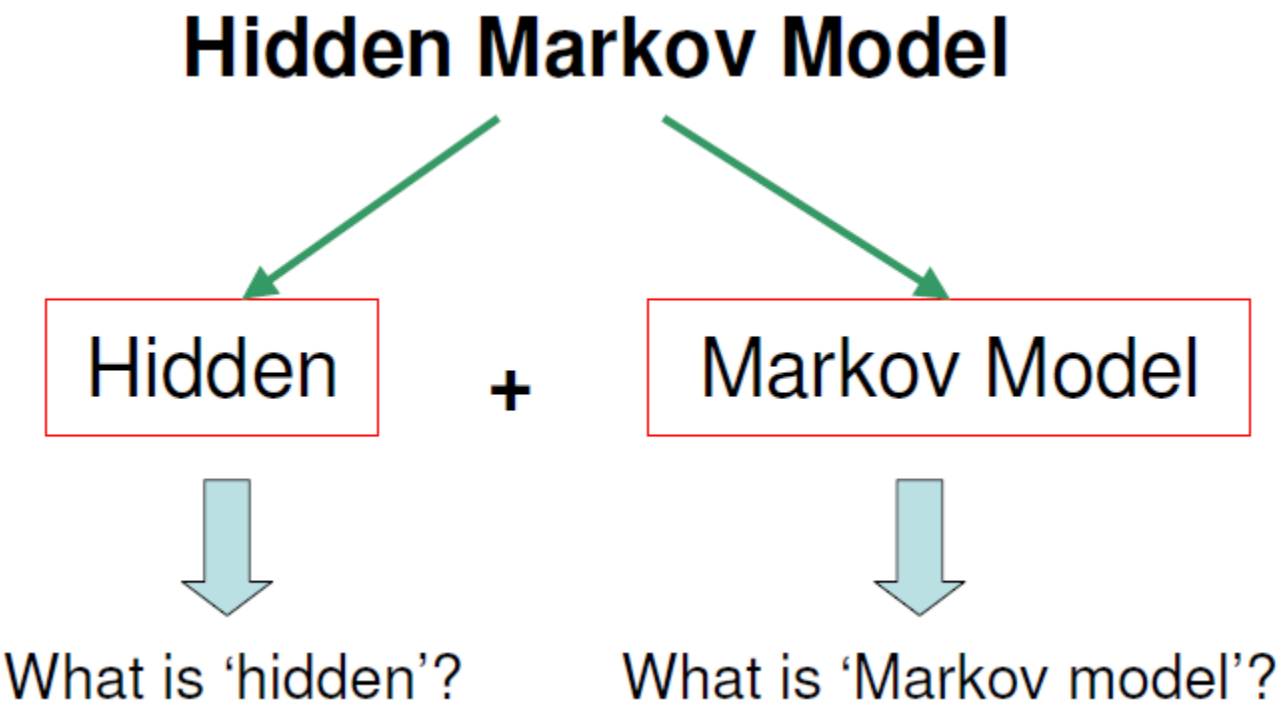




Hidden Markov Model

- HMMs can be applied in many fields where the goal is to recover a data sequence that is not immediately observable (but other data that depend on the sequence are).
- HMMs are used to uncover hidden spatial patterns and mine temporal and spatial data.

Hidden Markov Model



Successful Application Areas of HMM

- On-line handwriting recognition
- Speech recognition
- Gesture recognition
- Language modeling
- Motion video analysis and tracking
- Protein sequence/gene sequence alignment
- Stock price prediction
- And more …

Advantage of HMM on Sequential Data

- Natural model structure: doubly stochastic process
  - ✓ transition parameters model temporal variability
  - ✓ output distribution model spatial variability
- Efficient and good modeling tool for
  - ✓ sequences with temporal constraints
  - ✓ spatial variability along the sequence
  - ✓ real world complex processes
- Efficient evaluation, decoding and training algorithms
  - ✓ Mathematically strong
  - ✓ Computationally efficient
- Proven technology!
  - ✓ Successful stories in many applications

Markov Model

- Scenario
- Graphical representation
- Definition
- Sequence probability
- State probability

Markov Model: Scenario

- Classify a weather into three states
  - ✓ State 1: rain or snow
  - ✓ State 2: cloudy
  - ✓ State 3: sunny
- By carefully examining the weather of some city for a long time, we found following weather change pattern

		Tomorrow		
		Rain/snow	Cloudy	Sunny
Today	Rain/Snow	0.4	0.3	0.3
	Cloudy	0.2	0.6	0.2
	Sunny	0.1	0.1	0.8

- Assumption: tomorrow weather depends only on today one!

Markov Models

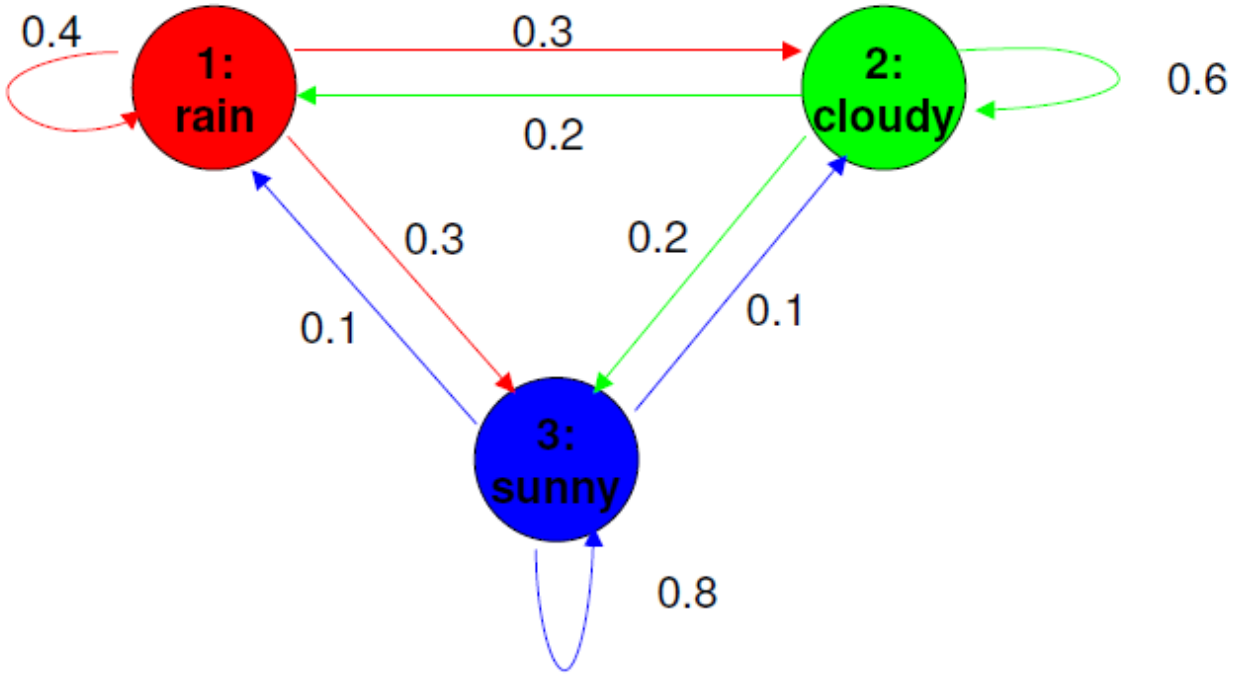
- A Markov model is a finite state machine with N distinct states begins at (Time t = 1) in initial state .
- It moves from current state to Next state according to the transition probabilities associated with the Current state
- This kind of system is called ***Finite or Discrete Markov model.***

NOTE:

- ✓ In a first order Markov model, the current event only depends on the immediately preceding event.
- ✓ Second order models use the two preceding events.
- ✓ Third order models use the three preceding, etc.

Markov Model: Graphical Representation

- Visual illustration with diagram



- Each state corresponds to one observation
- Sum of outgoing edge weights is one

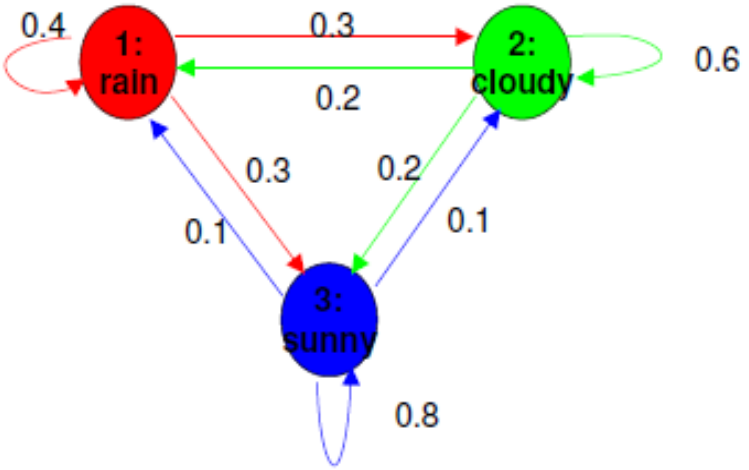
Markov Model: Definition

- Observable states

$$\{1,\ 2,\ \cdots,\ N\}$$

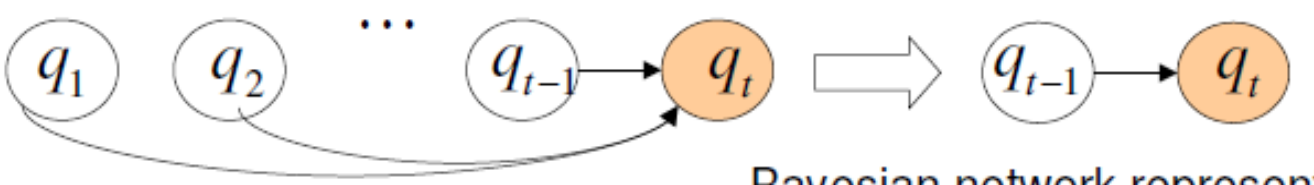
- Observed sequence

$$q_1, q_2, \cdots, q_T$$



- 1st order Markov assumption

$$P(q_t = j \mid q_{t-1} = i, q_{t-2} = k, \cdots) = P(q_t = j \mid q_{t-1} = i)$$



Bayesian network representation

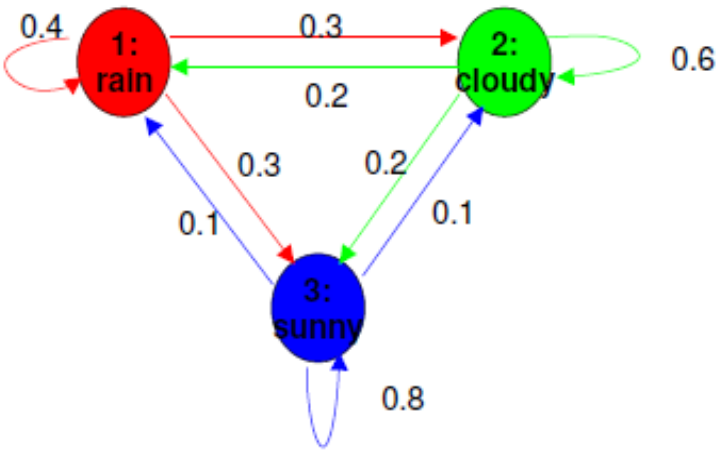
- Stationary

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

Markov Model: Definition (contd.)

- State transition matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N1} & a_{NN} & \cdots & a_{NN} \end{bmatrix}$$



  - Where

$$a_{ij} = P(q_t = j \mid q_{t-1} = i), \qquad 1 \le i, j \le N$$

  - With constraints

$$a_{ij} \ge 0, \qquad \sum_{j=1}^{N} a_{ij} = 1$$

- Initial state probability

$$\pi_i = P(q_1 = i), \qquad 1 \le i \le N$$

Markov Model: Sequence Probability

- Conditional probability

$$P(A,B) = P(A \mid B)P(B)$$

- Sequence probability of Markov model

$$P(q_1, q_2, \cdots, q_T)$$

Chain rule

$$= P(q_1)P(q_2 \mid q_1)\cdots P(q_{T-1} \mid q_1, \cdots, q_{T-2})P(q_T \mid q_1, \cdots, q_{T-1})$$

$$= P(q_1)P(q_2 \mid q_1)\cdots P(q_{T-1} \mid q_{T-2})P(q_T \mid q_{T-1})$$

1st order Markov assumption

Markov Model: Sequence Probability

- In a typical first-order Markov Model, the next state does only depend on the current state.
- The state transition probabilities in a second order Markov-Model do not only depend on the current state but also on the previous state

Markov Model: Sequence Probability

Consider four different states: A, B, C, D. Let's assume we are currently in state A. Then

$$P(X_{t+1} = B | X_t = A)$$

describes the probability, that in the next (time) step $(t+1)$ we transit to state B, given the information that the current state is A. Remark: X_t is a random variable and we say, that the collection of X_t over time is a stochastic process. Similarly there also exists $P(A|A), P(C|A), P(D|A)$. Obviously we require that

$$P(A|A) + P(B|A) + P(C|A) + P(D|A) = 1.$$

This exemplary construct is a typical first-order Markov Model. **The next state does only depend on the current state**. Formally:

$$P(X_{t+1} = B | X_t = A, X_{t-1} = B, \ldots, X_{t-n} = D) = P(X_{t+1} = B | X_t = A)$$

In contrast, **the state transition probabilities in a second order Markov-Model do not only depend on the current state but also on the previous state**. Hence with the singular knowledge of the current state, we can in general not compute the probabilities of the next state. Hence, transition probabilities are given by

$$P(X_{t+1} = A | X_t = A, X_{t-1} = B)$$

$$P(X_{t+1} = B | X_t = A, X_{t-1} = B)$$

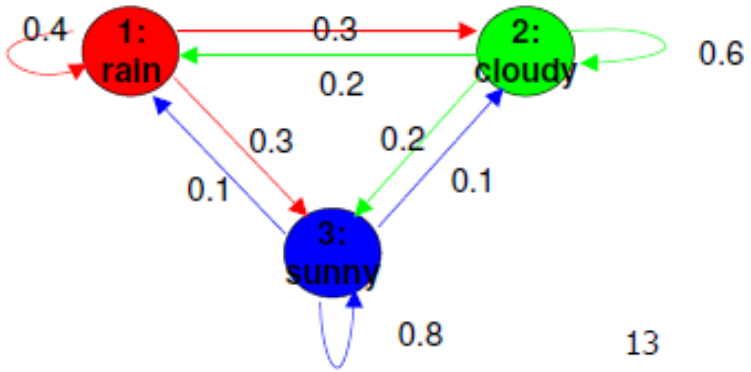
...

Markov Model: Sequence Probability (Contd.)

Question: What is the probability that the weather for the next 7 days will be "sun-sun-rain-rain-sun-cloudy-sun" when today is sunny?

$S_1 : rain, \quad S_2 : cloudy, \quad S_3 : sunny$

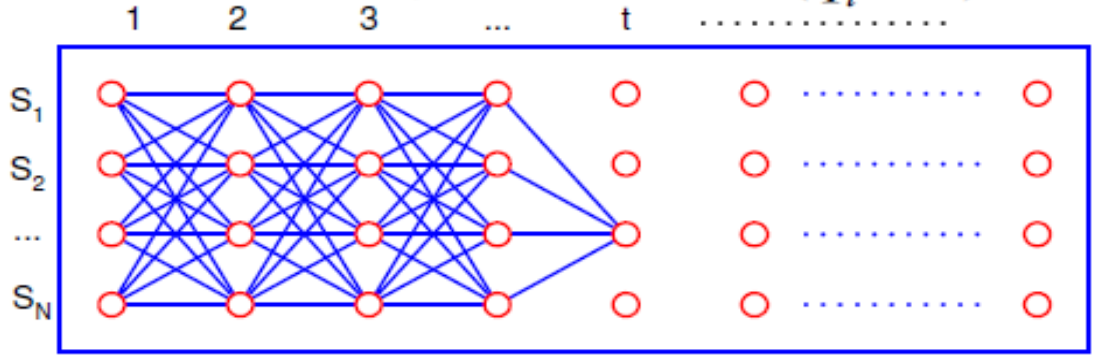
$$
\begin{aligned}
P(O \mid \text{model}) &= P(S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3 \mid \text{model}) \\
&= P(S_3) \cdot P(S_3 \mid S_3) \cdot P(S_3 \mid S_3) \cdot P(S_1 \mid S_3) \\
&\quad \cdot P(S_1 \mid S_1) P(S_3 \mid S_1) P(S_2 \mid S_3) P(S_3 \mid S_2) \\
&= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\
&= 1 \cdot (0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2) \\
&= 1.536 \times 10^{-4}
\end{aligned}
$$



13

Markov Model: State Probability

- State probability at time t : $P(q_t = i)$



- Simple but slow algorithm:
  - Probability of a path that ends to state *i* at time *t*:

$$Q_t(i) = (q_1, q_2, \cdots, q_t = i)$$

$$P(Q_t(i)) = \pi_{q_1} \prod_{k=2}^{t} P(q_k \mid q_{k-1})$$

  - Summation of probabilities of all the paths that ends to I at t

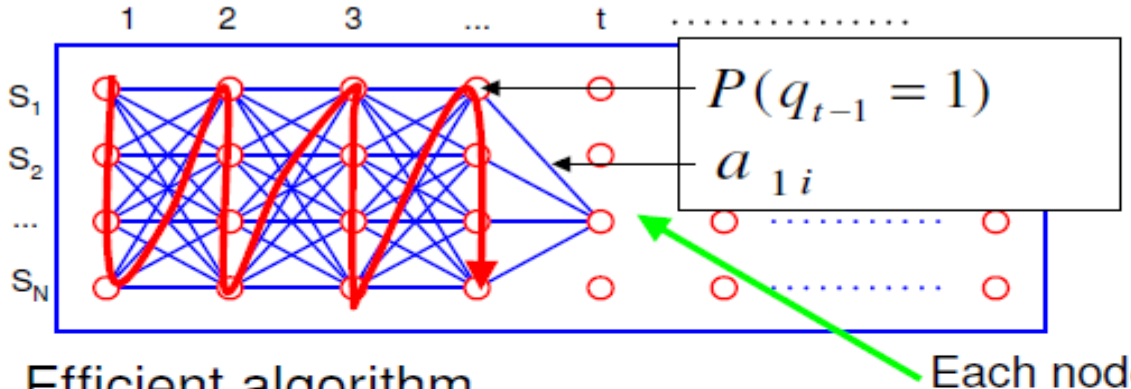
$$P(q_t = i) = \sum_{all\ Q_t(i)'s} P(Q_t(i))$$

Exponential time complexity:

$$O(N^t)$$

Markov Model: State Probability (Contd.)

- State probability at time t : $P(q_t = i)$



- Efficient algorithm
  - Recursive path probability calculation

$$P(q_t = i) = \sum_{j=1}^{N} P(q_{t-1} = j, q_t = i)$$

$$= \sum_{j=1}^{N} P(q_{t-1} = j) P(q_t = i \mid q_{t-1} = j)$$

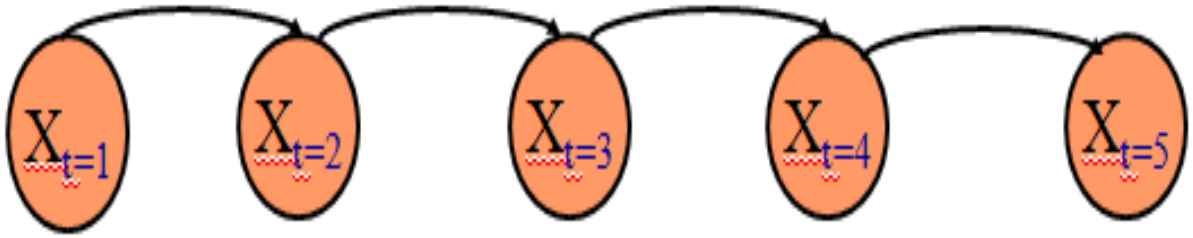
$$= \sum_{j=1}^{N} P(q_{t-1} = j) \cdot a_{ji}$$

Time complexity: $O(N^2 t)$

Markov Property

- Markov Property : The Current state of the system depends only on the previous state of the system

- The State of the system at Time [T+1] depends on the state of the system at time T.



Discrete Markov Model : Example

- A Discrete Markov Model with 5 states.
- Each a_{ij} represents the probability of moving from state ‘ i’ to state ’j’.

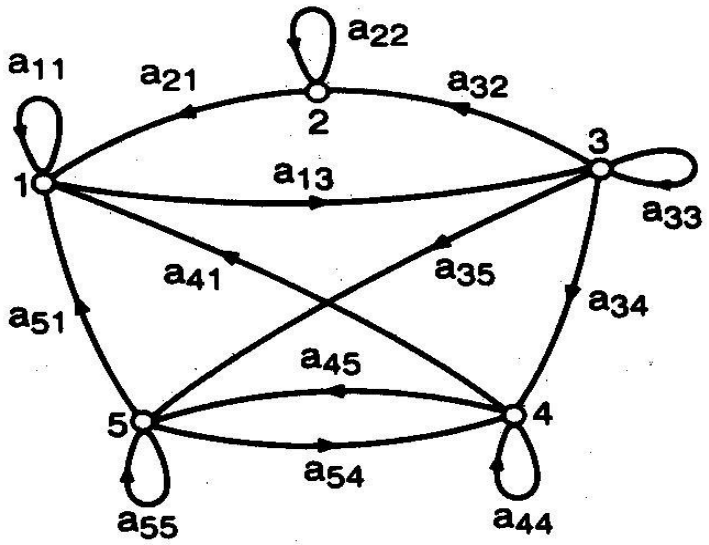


Figure 6.1 A Markov chain with five states (labeled 1 to 5) with selected state transitions.

Example

- The probability to start in a given state i is π_i.
- The Vector π represents the start probabilities.
- To define Markov model, the following probabilities have to be specified: transition probabilities $a_{ij} = P(S_i \mid S_j)$ and initial probabilities

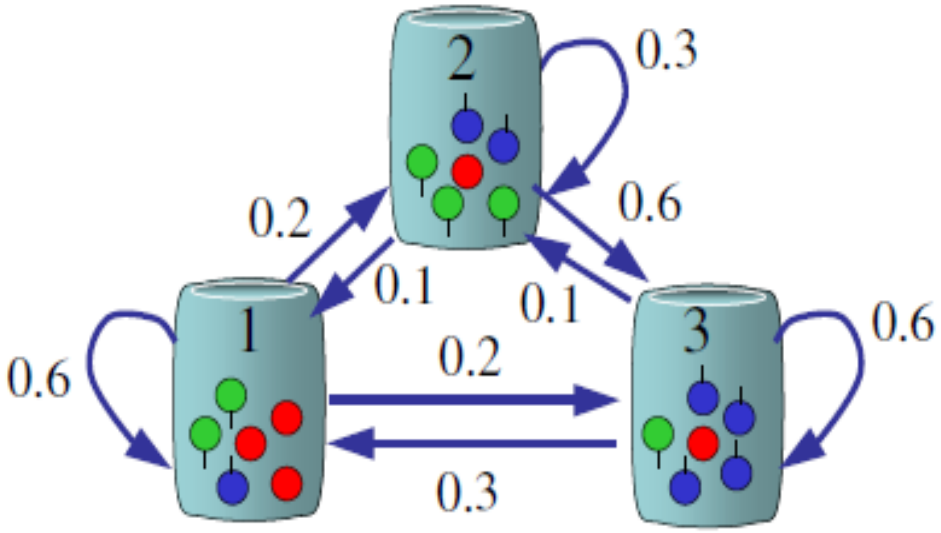
$$\pi_i = P(S_i)$$

Hidden Markov Models

- A Hidden Markov model is a statistical model in which the system being modeled is assumed to be markov process with unobserved hidden states.

- In Regular Markov models the state is clearly visible to others in which the state transition probabilities are the parameters only where as in HMM the state is not visible but the output is visible.

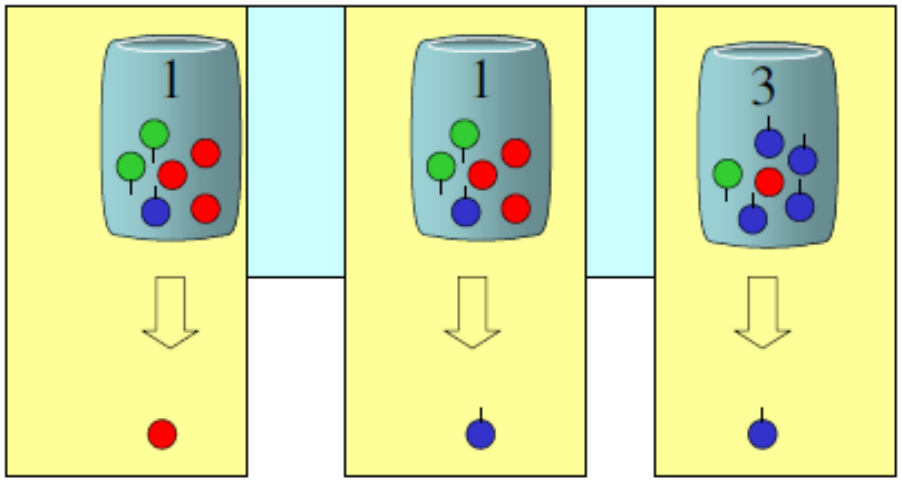
Hidden Markov Model: Example

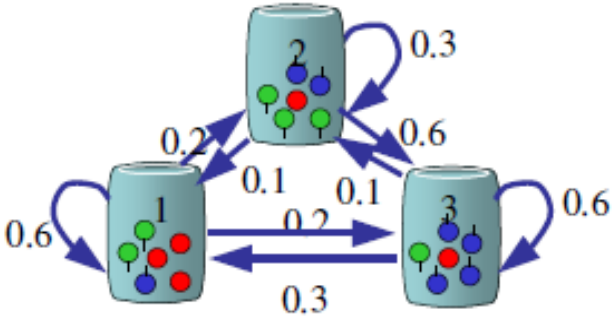


- N urns containing color balls
- M distinct colors
- Each urn contains different number of color balls

HMM: Generation Process

- Sequence generating algorithm
  - Step 1: Pick initial urn according to some random process
  - Step 2: Randomly pick a ball from the urn and then replace it
  - Step 3: Select another urn according to a random selection process
  - Step 4: Repeat steps 2 and 3



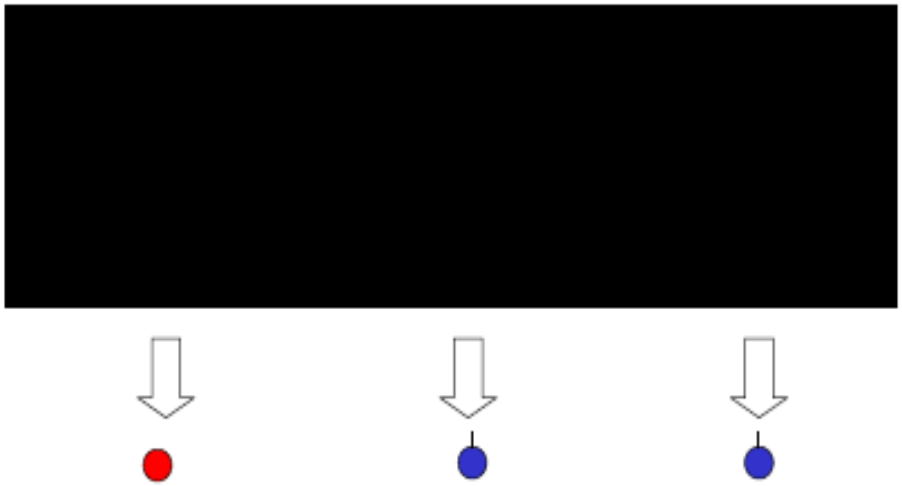


Markov process: $\{q(t)\}$

Output process: $\{f(x|q)\}$

HMM: Hidden Information

Now, what is hidden?



- We can just see the chosen balls
- We can't see which urn is selected at a time
- So, urn selection (state transition) information is hidden

HMM: Definition

Notation: $\lambda = (A, B, \Pi)$

(1) N: Number of states

(2) M: Number of symbols observable in states

$$V = \{v_1, \cdots, v_M\}$$

(3) A: State transition probability distribution

$$A = \{a_{ij}\}, \quad 1 \le i, j \le N$$

(4) B: Observation symbol probability distribution

$$B = \{b_i(v_k)\}, \quad 1 \le i \le N, 1 \le j \le M$$

(5) Π: Initial state distribution

$$\pi_i = P(q_1 = i), \quad 1 \le i \le N$$

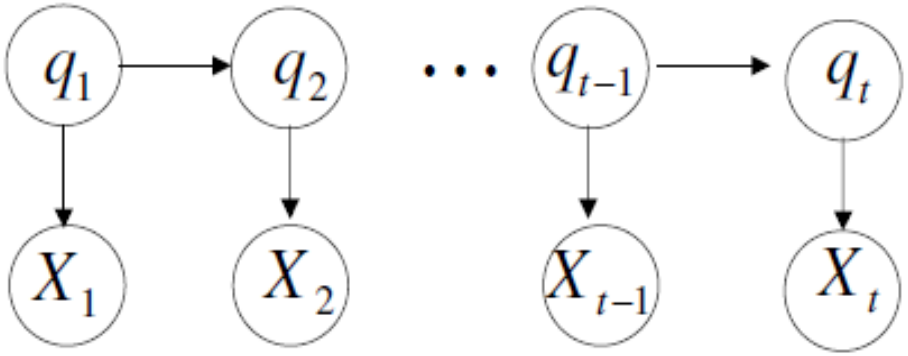
HMM: Dependency Structure

- 1-st order Markov assumption of transition

$$P(q_t \mid q_1, q_2, \cdots, q_{t-1}) = P(q_t \mid q_{t-1})$$

- Conditional independency of observation parameters

$$P(X_t \mid q_t, X_1, \cdots, X_{t-1}, q_1, \cdots, q_{t-1}) = P(X_t \mid q_t)$$



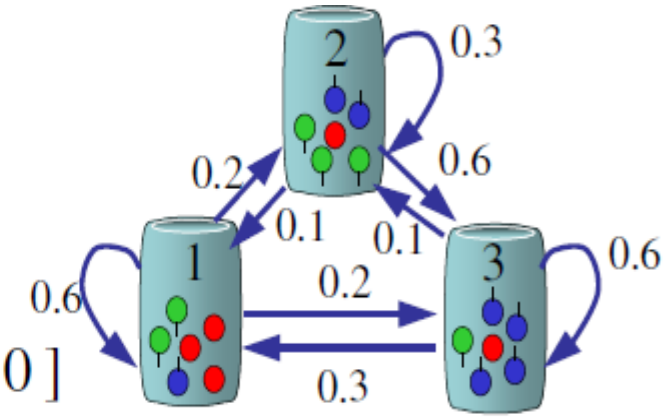
Bayesian network representation

HMM: Example Revisited

- # of states: N=3
- # of observation: M=3

  V = { R, G, B }
- Initial state distribution

$$\pi = \{ P(q_1 = i) \} = [1,0,0]$$



- State transition probability distribution

$$A = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.1 & 0.3 & 0.6 \\ 0.3 & 0.1 & 0.6 \end{bmatrix}$$

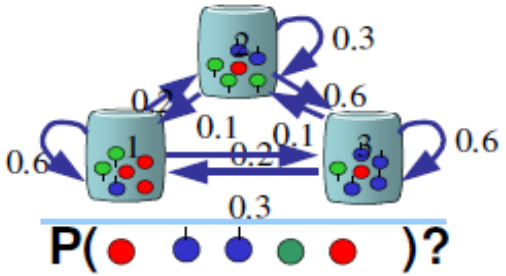
- Observation symbol probability distribution
-

$$B = \{b_i(v_k)\} = \begin{bmatrix} 3/6 & 2/6 & 1/6 \\ 1/6 & 3/6 & 2/6 \\ 1/6 & 1/6 & 4/6 \end{bmatrix}$$

HMM: Three Problems

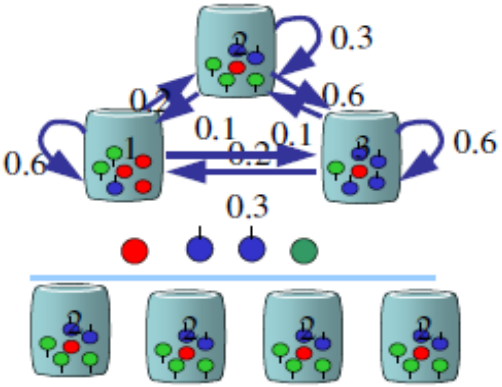
- What is the probability of generating an observation sequence?
  - Model evaluation

$$P(X = x_1, x_2, \cdots, x_T \mid \lambda) = ?$$



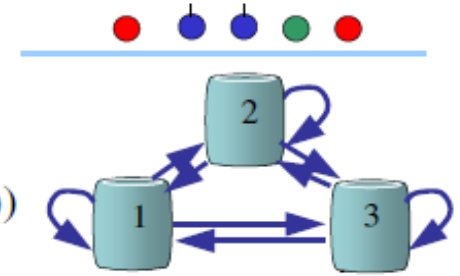
- Given observation, what is the most probable transition sequence?
  - Segmentation or path analysis

$$Q^* = \arg\max_{Q=(q_1,\cdots,q_T)} P(Q, X \mid \lambda)$$



- How do we estimate or optimize the parameters of an HMM?
  - Training problem

$$P(X \mid \lambda = (A, B, \pi)) < P(X \mid \lambda' = (A', B', \pi'))$$



HMM Description

- It consists of set of states : $S_1, S_2, S_3 \ldots\ldots S_n$.
- Process moves from One state to another state generating a sequence of states $S_{i1}, S_{i2}, \ldots S_{ik} \ldots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state

$$P(S_{ik} \mid S_{ik1}, S_{ik2}, \ldots\ldots\ldots S_{ik-1}) = P (S_{ik} \mid S_{ik-1})$$

- States are not visible, but each state randomly generates one of M observations (or visible states)

$$V = \{ v1, v2, v3 \ldots\ldots vk \ldots\ldots \}$$

HMM Essentials

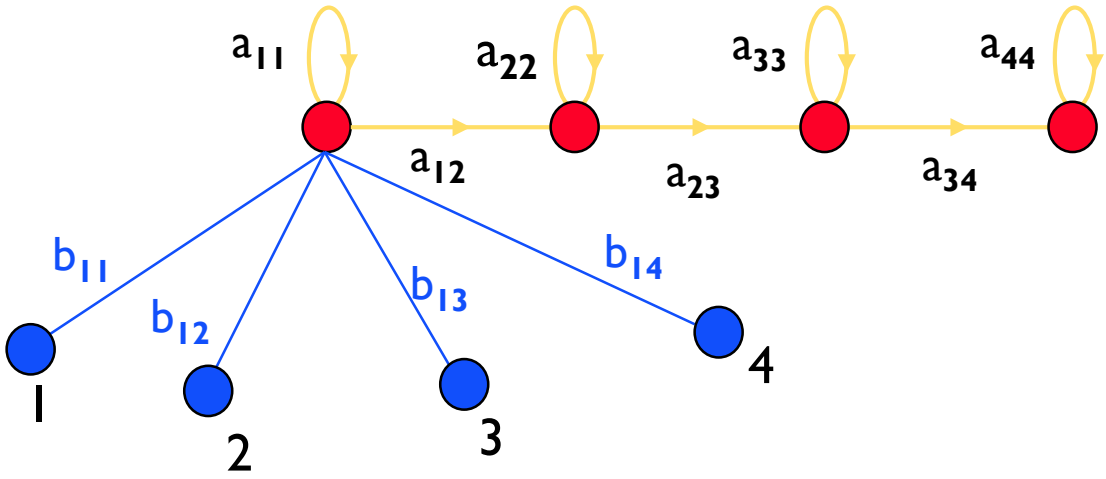
- To define hidden Markov model, the following probabilities have to be specified:
  - ✓ matrix of transition probabilities $A=(a_{ij})$, $a_{ij}= P(s_i | s_j)$
  - ✓ matrix of observation probabilities

    $B=(b_i(v_m))$, $b_i(v_m)= P(v_m | s_i)$ and a vector of initial probabilities $\pi=(\pi_i)$, $\pi_i = P(s_i)$.

- Model is represented by $M=(A, B, \pi)$.

Hidden markov models (Probabilistic finite state automata)

- The Scenarios where states cannot be directly observed.
- We need an extension i.e, Hidden Markov Models

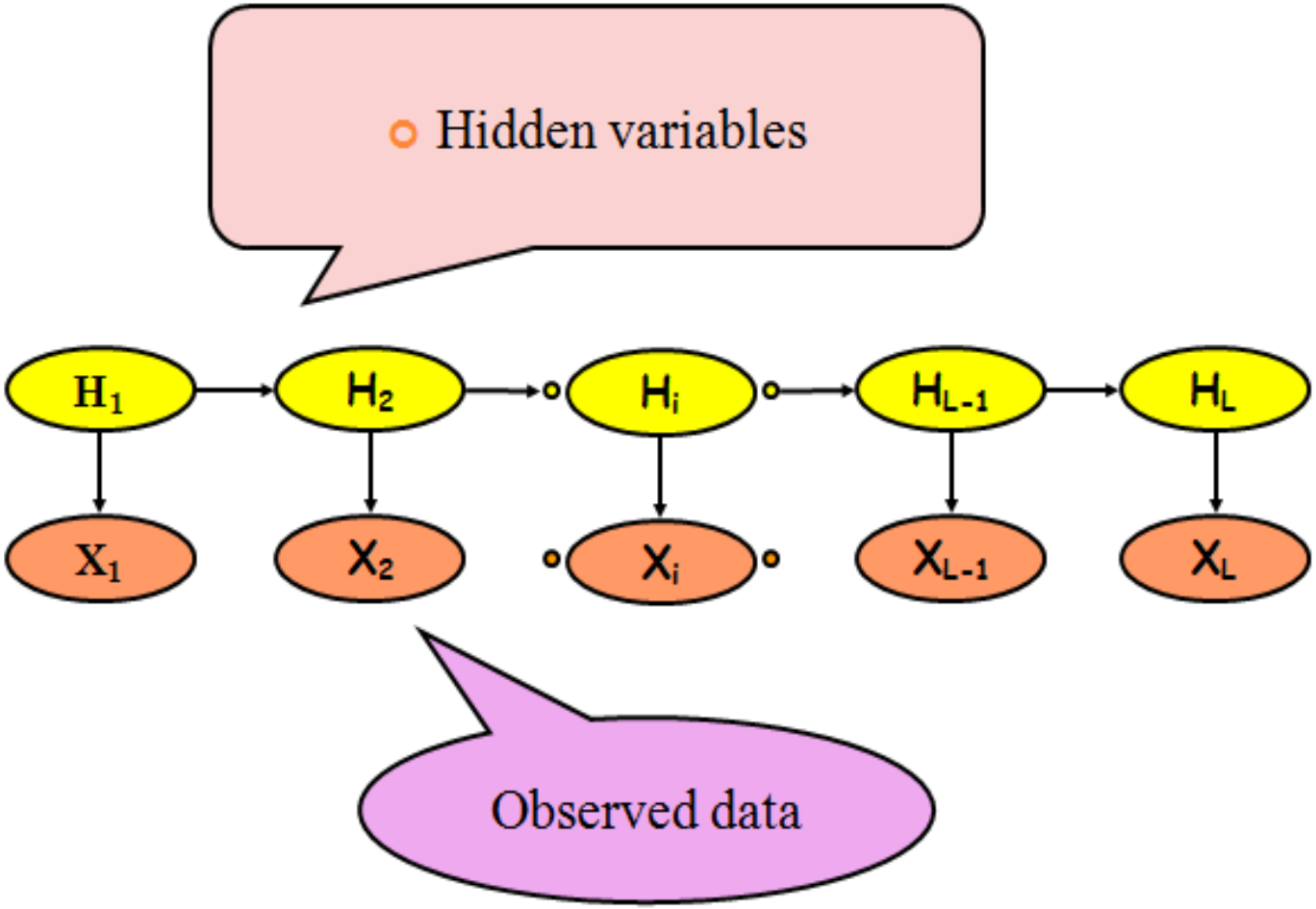


a_{ij} are state transition probabilities.

b_{ik} are observation (output) probabilities.

$b_{11} + b_{12} + b_{13} + b_{14} = 1,$
$b_{21} + b_{22} + b_{23} + b_{24} = 1.$

Hidden Markov Models - HMM



Hidden markov model recognition

- For a given model M = { A, B, p } and a given state sequence Q_1 Q_2 Q_3 … Q_L , the probability of an observation sequence O_1 O_2 O_3 … O_L is
  $P(O|Q,M) = bQ_1O_1\ bQ_2O_2\ bQ_3O_3 \cdots bQ_TO_T$

- For a given hidden Markov model M = { A, B, p}
  the probability of state sequence Q_1 Q_2 Q_3 Q_L
  is (the initial probability of Q_1 is taken to be pQ_1)
  $P(Q|M) = pQ_1\ aQ_1Q_2\ aQ_2Q_3\ aQ_3Q_4 \cdots aQ_{L-1}Q_L$

Hidden markov model recognition

- So for a given HMM, M the probability of an observed sequence $O_1O_2O_3 \ \ldots \ O_T$ is obtained by summing over all possible state sequences.

$$P(Q|M) = pQ_1 \ aQ_1Q_2 \ aQ_2Q_3 \ aQ_3Q_4 \cdots aQ_{T-1}Q_T$$

$$P(O|Q) = bQ_1O_1 \ bQ_2O_2 \ bQ_3O_3 \cdots bQ_TO_T$$

Main issues

- **Evaluation problem:** Given the HMM

  $M = \{ A, B, \pi \}$

  and observation sequence $O = o_1, o_2 \ldots\ldots o_k$,

  Calculate the probability that model M has generated sequence O.

- **Solution:** *Forward-Backward Algorithm*

Main issues

- **Learning Problem:** Given some training observation sequences O = o_1,o_2 ……o_k, and general structure of HMM (visible and hidden states) Determine HMM parameters that best fit the training data.
- **Solution:** *Baum-Welch Algorithm* which is an algorithm to find HMM parameters A, B, and Π with the maximum likelihood of generating the given symbol sequence in the observation vector

Main issues

- **Decoding problem:** Given the HMM M = { A, B, π } and observation sequence O = $o_1, o_2 \ldots\ldots o_k$,
  Calculate the most likely sequence of hidden states S_i that generated sequence O.

- **Solution:** *Viterbi Algorithm*

Solutions to evaluation problem

- **Evaluation problem**: For this problem We use an *Forward- Backward algorithm*
- This algorithm mainly consists of defining a forward or backward variable as the joint probability of partial state sequence such as:

  O = o1, o2, …..ok and the hidden state S_i at time k is $\alpha k(i) = p(o1\ o2\ o3 \ldots ok, Qk = S_i)$.
- The three states in this algorithm are *initialisation*, *forward recursion* and *termination*.

Forward Algorithm

- **Define forward variable $\alpha_t(i)$ as:**

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i|\lambda)$$

- **$\alpha_t(i)$ is the probability of observing the partial sequence $(o_1, o_2 \ldots, o_t)$ such that the state q_t is i.**

- **Induction:**

  1. **Initialization:** $\alpha_1(i) = \pi_i b_i(o_1)$

  2. **Induction:**

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

  3. **Termination:**

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

- **Complexity:** $O(N^2T)$.

Backward Algorithm

- **Define backward variable $\beta_t(i)$ as:**

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

- **$\beta_t(i)$ is the probability of observing the partial sequence $(o_{t+1}, o_{t+2} \ldots, o_T)$ such that the state q_t is i.**

- **Induction:**

  1. **Initialization:** $\beta_T(i) = 1$

  2. **Induction:**

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

Solutions to Learning problem

- The solution to this problem is to estimate parameters.
- The parameters that need to be estimated are *Transmission probabilities* and *emission probabilities*. Since they sum upto 1, only 2 transmission and 2 estimation parameters are to be found.
- More parameter estimation be done using *Baum-Welch algorithm.*

Solution to Learning Problem

- **Estimate $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$**
- **No analytic method because of complexity – iterative solution.**
- **Baum-Welch Algorithm:**
  1. **Let initial model be λ_0.**
  2. **Compute new λ based on λ_0 and observation O.**
  3. **If $\log P(O|\lambda) - \log P(O|\lambda_0) < DELTA$ stop.**
  4. **Else set $\lambda_0 \leftarrow \lambda$ and goto step 2.**

Solution to decoding problem

- Decoding problem solution: Viterbi Algorithm
- In this algorithm we go through the observations from start to end referring a state of hidden machine for each observation.
- We also record the values of *Overall Probability*, *Viterbi path* (sequence of states) and the *viterbi probability* (Probability of observed state sequences in viterbi path)
- The probability of possible step given its corresponding observation is ***probability of transmission*** times ***emission probability***.

Viterbi algorithm

- **Overall Probability**: Multiply each new probability with the old one and then add together.
- **Viterbi probability**: Take the highest next step probability and multiply with the next step viterbi probability.
- **Viterbi path**: Add the next step path to viterbi path.

Viterbi algorithm with example

- Let's say: A person basically does 3 activities: walk, clean and shop depending on the weather conditions.
- Possibility of weather conditions are 'Rainy' and 'sunny'.
- In this example weather condition states are **hidden** and we will know the weather condition by his/her activities.

Viterbi algorithm with example

- As we discussed in earlier slides for every hidden markov model (HMM) we need Transition probabilities and Emission probabilities.
- The transition probabilities are :

  P(R ---> R) (Rainy stays rainy) = 0.7

  P(R ---> S) (Rainy turns into Sunny) = 0.3

  P (S ---> S) (Sunny stays into sunny) = 0.6

  P (S----> R) (Sunny turns into rainy) = 0.4
- *Transition probabilities* give the state of next day (i.e. sunny or rainy)
- *Emission probabilities* give behavior/activity due to different state changes

Viterbi algorithm with example

- The Observations of her activities is

  If it is Rainy the behaviour is

  Walk = 0.1

  Clean = 0.5

  Shop = 0.4

- If it is Sunny the behaviour is

  Walk = 0.6

  Clean = 0.3

  Shop = 0.1

Viterbi algorithm with example

- If the observations are WCSW
- Then according to algorithm find the overall prob, vit Prob, vit_path.
- In vi_path you get the sequence of states which need to compare with the original states in order to know the accuracy
- Through many examples the accuracy varies between 80-90%

WCSW: Walk-Clean-Shop-Walk

Decoding: finding the most likely path

- Definition of decoding: Finding the most likely state sequence X that explains the observations, given this HMM's parameters.

$$\hat{X} = \underset{X_0 \ldots X_{T+1}}{\operatorname{argmax}} P(X|O,\mu) =$$

$$\underset{X_0 \ldots X_{T+1}}{\operatorname{argmax}} \prod_{t=0}^{T+1} P(O_t|X_t)P(X_t|X_{t-1})$$

- Search space of possible state sequences X is $O(N^T)$; too large for brute force search.

Viterbi is a Dynamic Programming Application

We can use Dynamic Programming if two conditions apply:

- Optimal substructure property
  - An optimal state sequence $X_0 \ldots X_j \ldots X_{T+1}$ contains inside it the sequence $X_0 \ldots X_j$, which is also optimal
- Overlapping subsolutions property
  - If both X_t and X_u are on the optimal path, with $u > t$, then the calculation of the probability for being in state X_t is part of each of the many calculations for being in state X_u.

Intuition behind Viterbi

- Here's how we can save ourselves a lot of time.
- Because of the Limited Horizon of the HMM, we don't need to keep a complete record of how we arrived at a certain state.
- For the first-order HMM, we only need to record one previous step.
- Just do the calculation of the probability of reaching each state once for each time step.
- Then memoise this probability in a Dynamic Programming table
- This reduces our effort to $O(N^2 T)$.
- This is for the first order HMM, which only has a memory of one previous state.

Viterbi: main data structure

- Memoisation is done using a *trellis*.
- A trellis is equivalent to a Dynamic Programming table.
- The trellis is $N \times (T + 1)$ in size, with states j as rows and time steps t as columns.
- Each cell j, t records the Viterbi probability $\delta_j(t)$, the probability of the optimal state sequence ending in state s_j at time t:

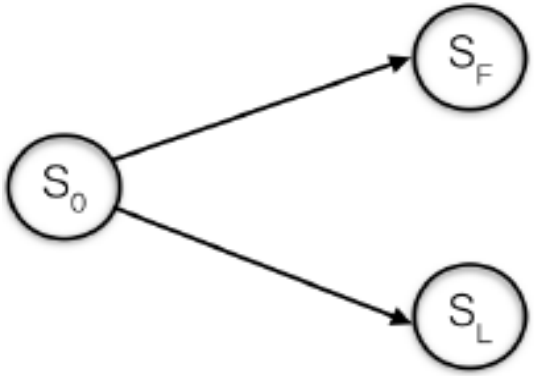
$$\delta_j(t) = \max_{x_0,\ldots,x_{t-1}} P(X_0 \ldots X_{t-1}, o_1 o_2 \ldots o_t, X_t = s_j | \mu)$$

Viterbi algorithm, initialisation

- The initial $\delta_j(1)$ concerns time step 1.
- It stores, for all states, the probability of moving to state s_j from the start state, and having emitted o_1.
- We therefore calculate it for each state s_j by multiplying transmission probability a_{0j} from the start state to s_j, with the emission probability for the first emission o_1.

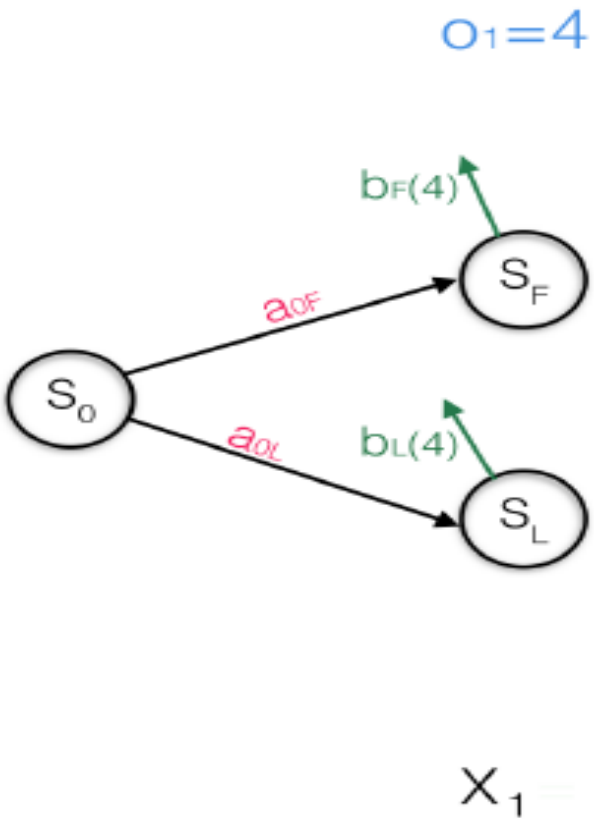
$$\delta_j(1) = a_{0j} b_j(o_1), 1 \leq j \leq N$$

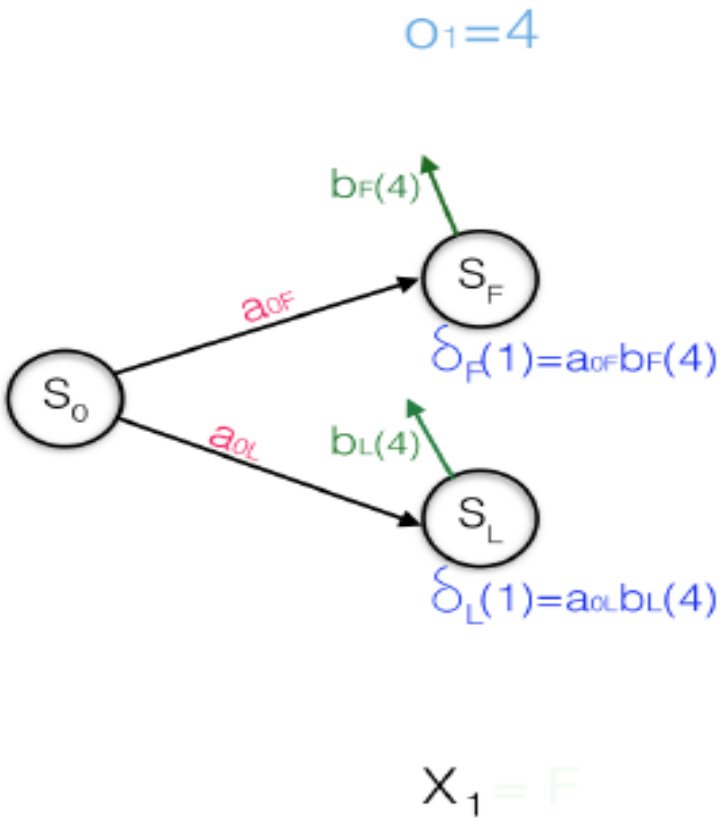
Viterbi algorithm, initialisation



$X_1 = F$

Viterbi algorithm, initialisation: observation is 4



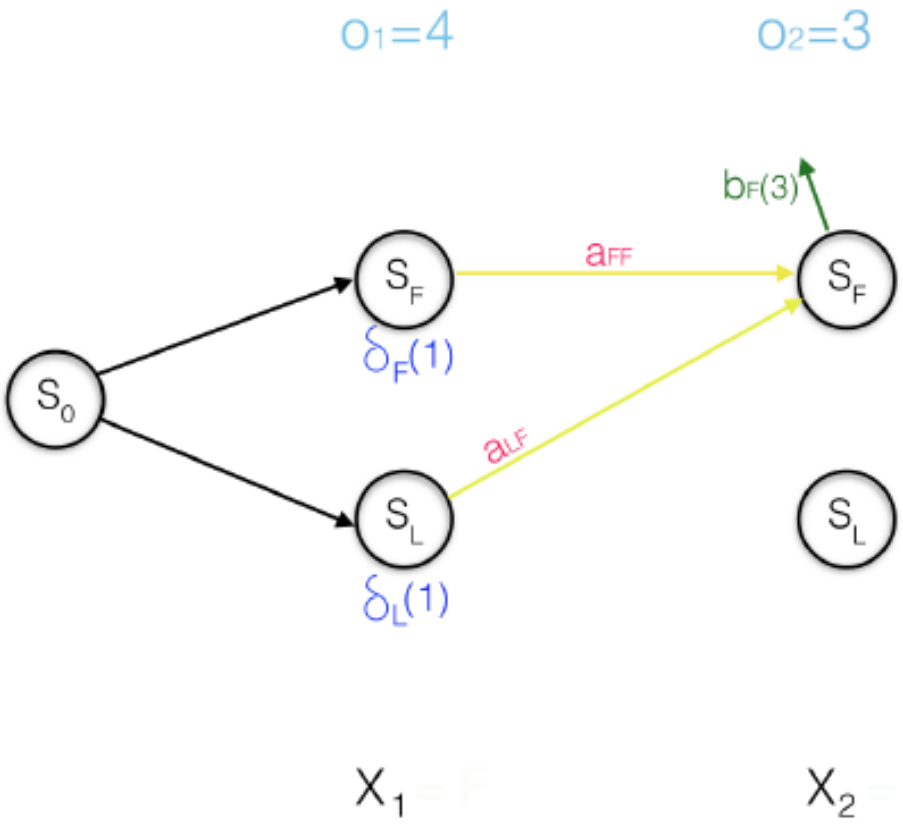


Viterbi algorithm, main step, observation is 3

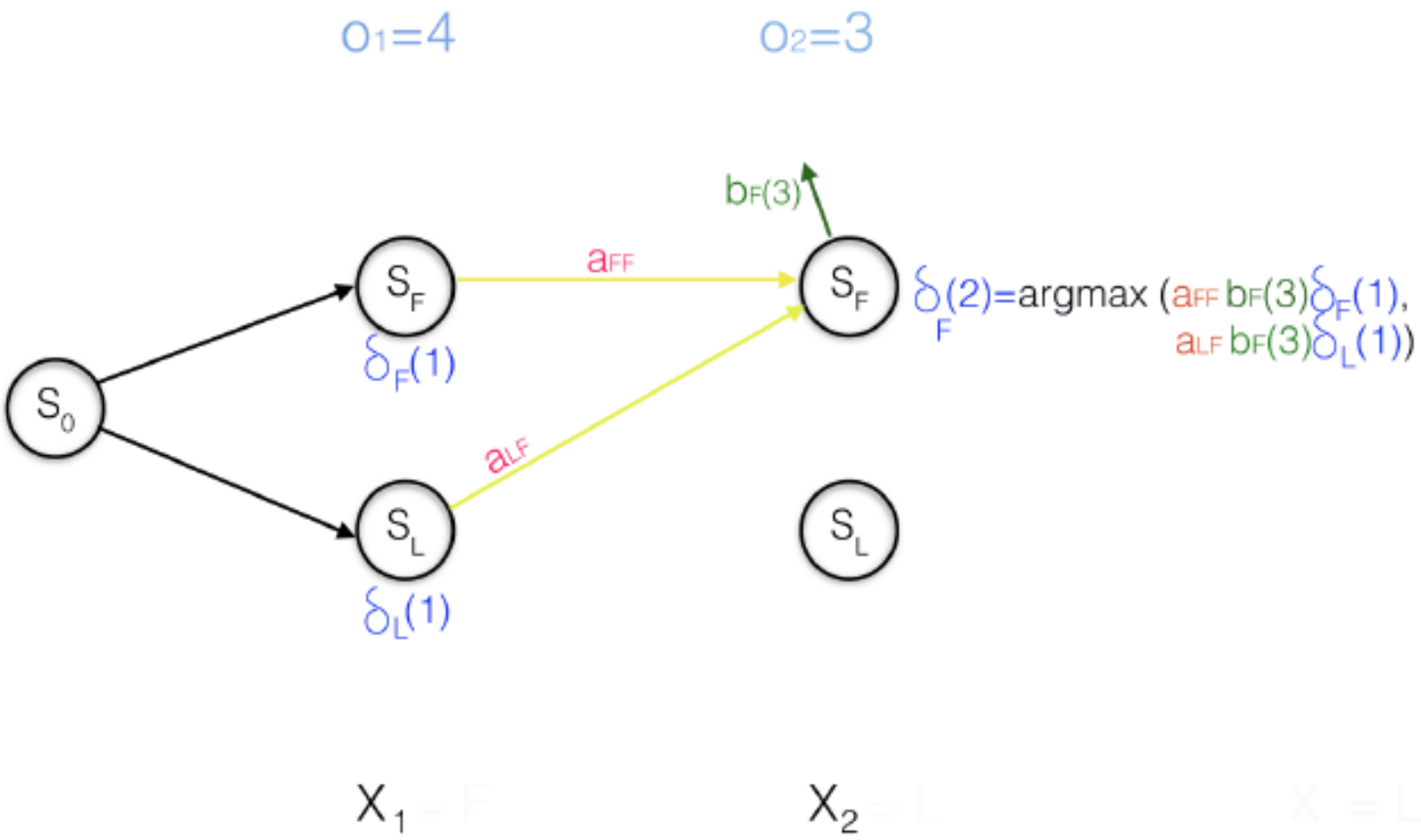
- $\delta_j(t)$ stores the probability of the best path ending in s_j at time step t.
- This probability is calculated by maximising over the best ways of transmitting into s_j for each s_i.
- This step comprises:
  - $\delta_i(t-1)$: the probability of being in state s_i at time $t-1$
  - a_{ij}: the transition probability from s_i to s_j
  - $b_i(o_t)$: the probability of emitting o_t from destination state s_j

$$\delta_j(t) = \max_{1 \le i \le N} \delta_i(t-1) \cdot a_{ij} \cdot b_j(o_t)$$

Viterbi algorithm, main step



Viterbi algorithm, main step



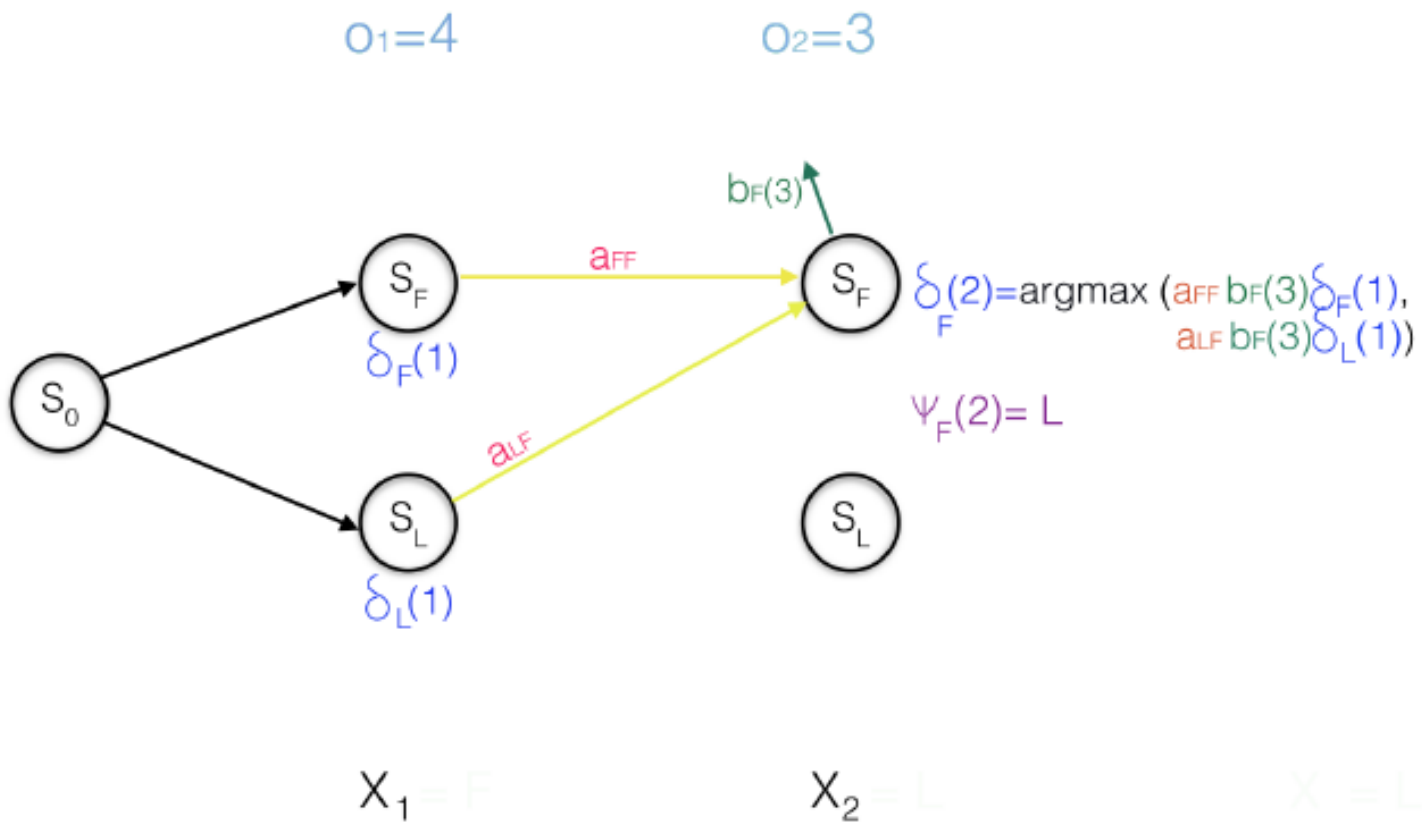
Viterbi algorithm, main step, ψ

- $\psi_j(t)$ is a helper variable that stores the $t-1$ state index i on the highest probability path.

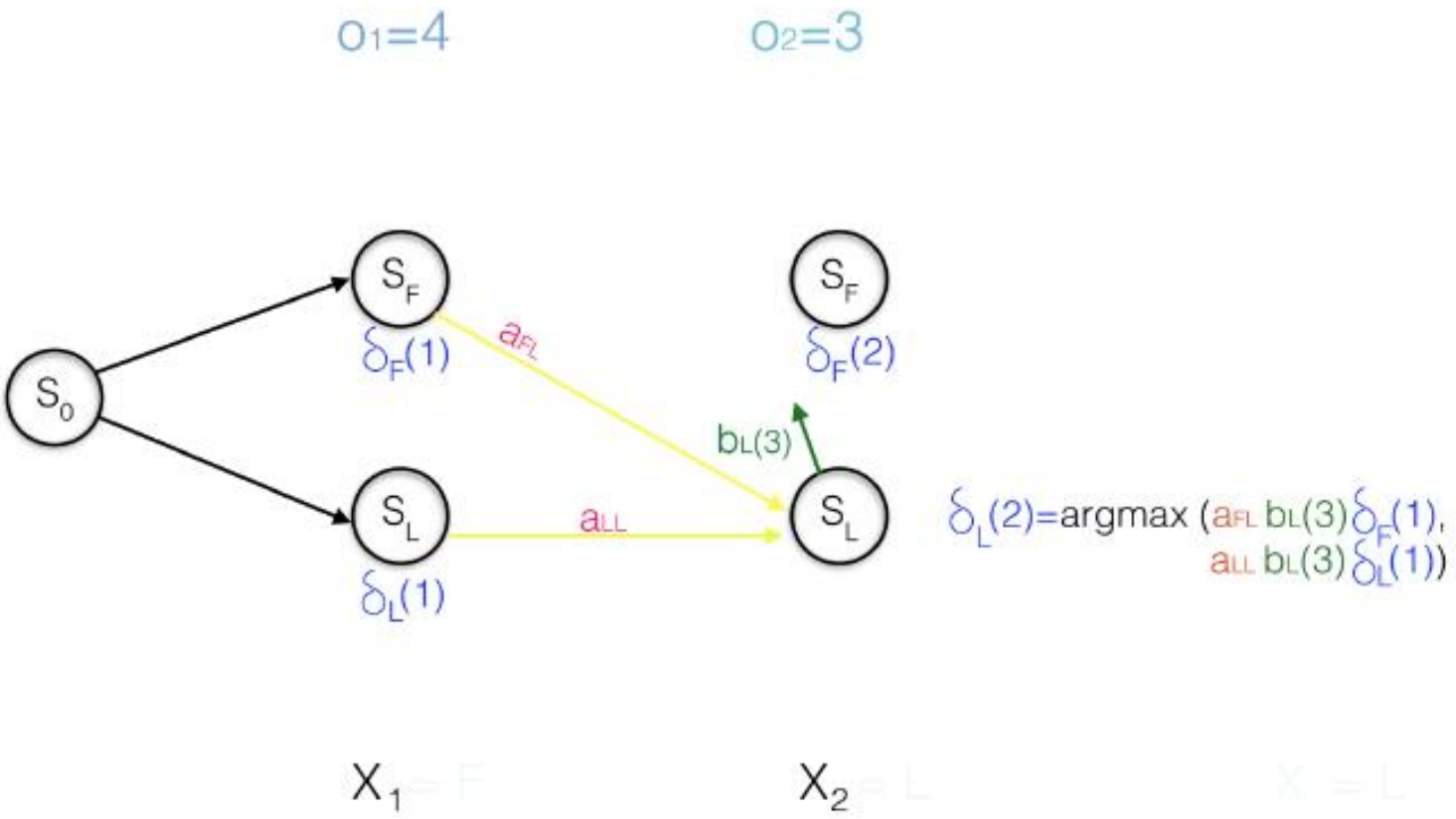
$$\psi_j(t) = \underset{1 \le i \le N}{\operatorname{argmax}}\, \delta_i(t-1) a_{ij} b_j(o_t)$$

- In the backtracing phase, we will use ψ to find the previous cell in the best path.

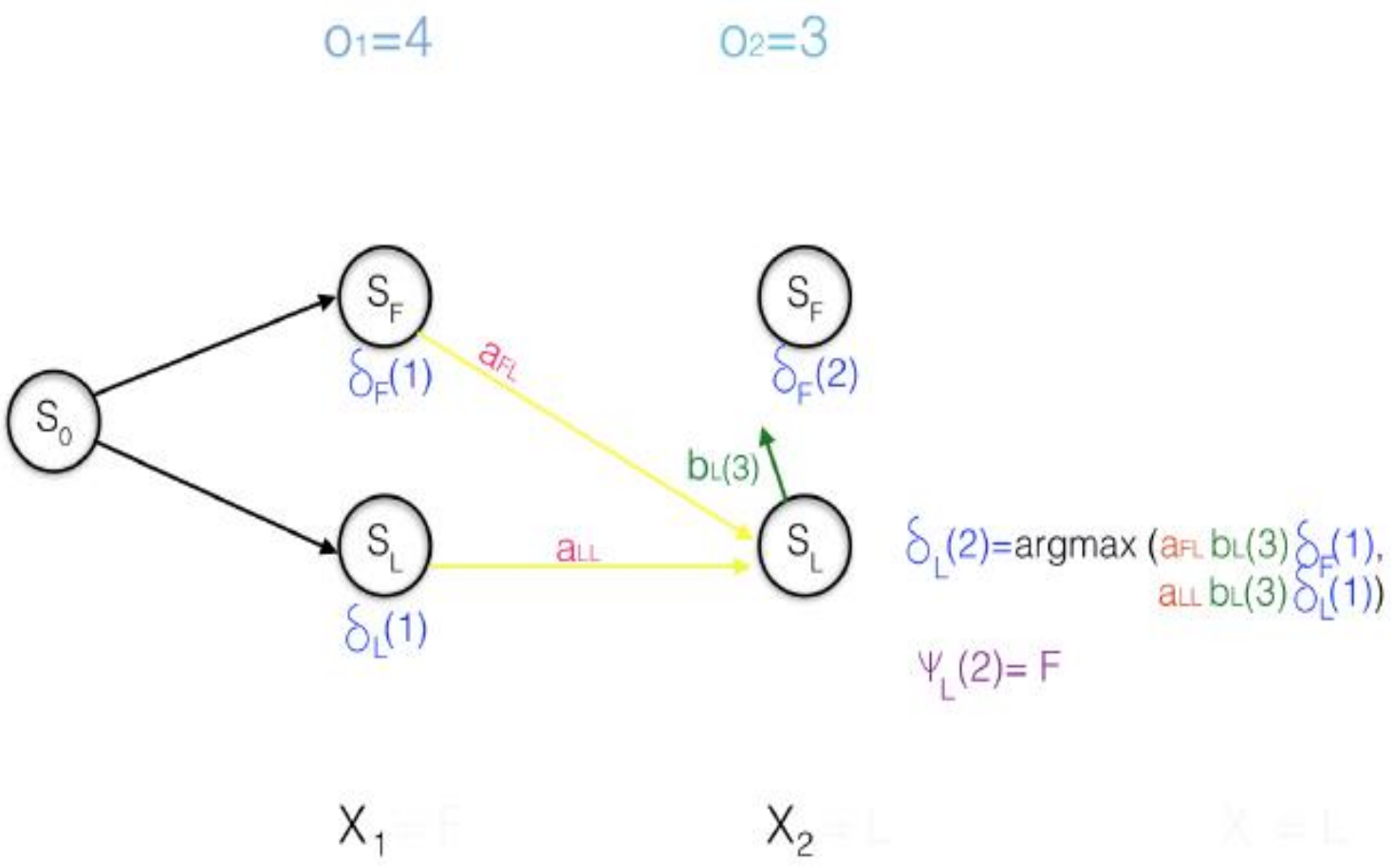
Viterbi algorithm, main step



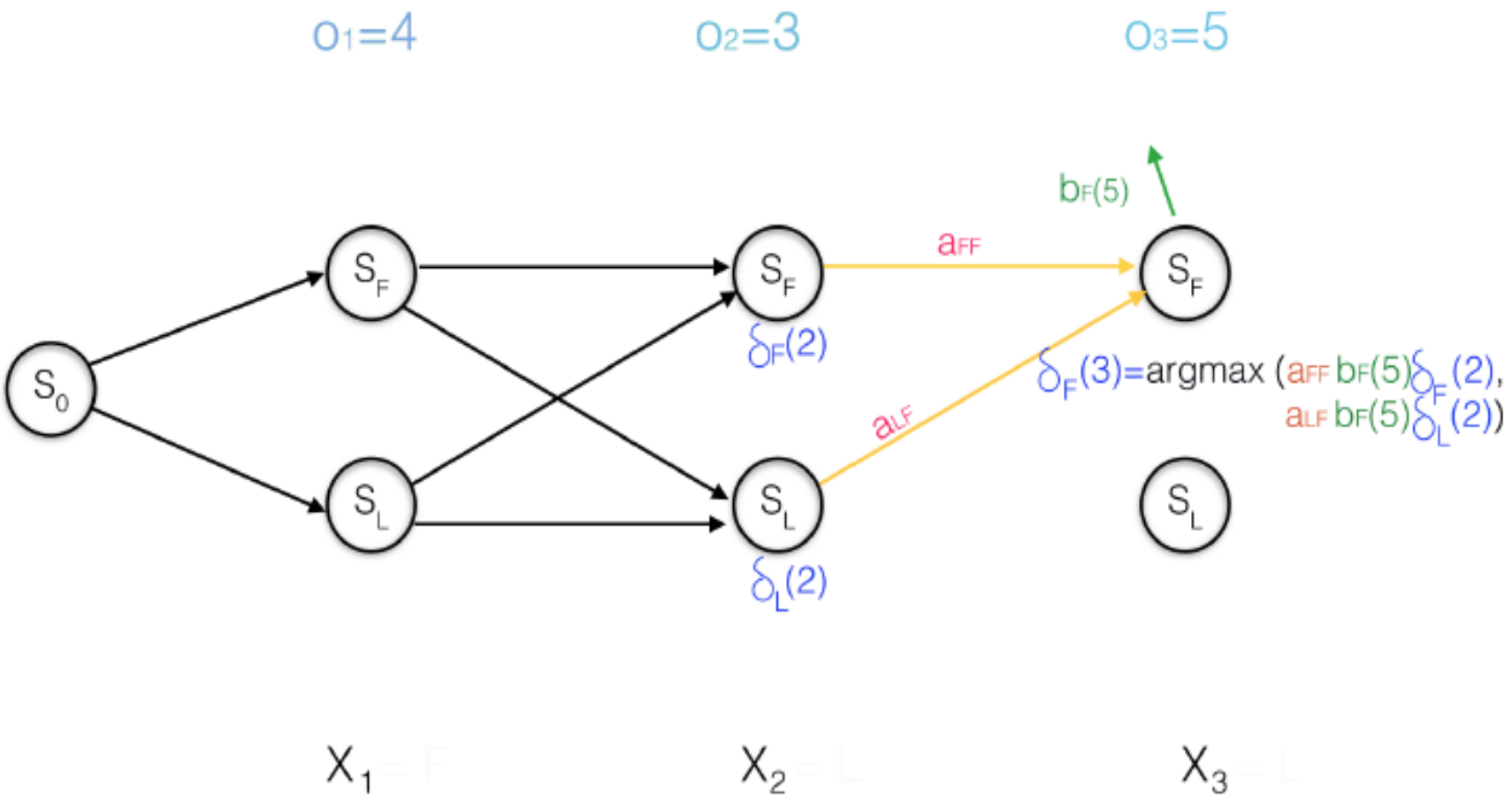
Viterbi algorithm, main step



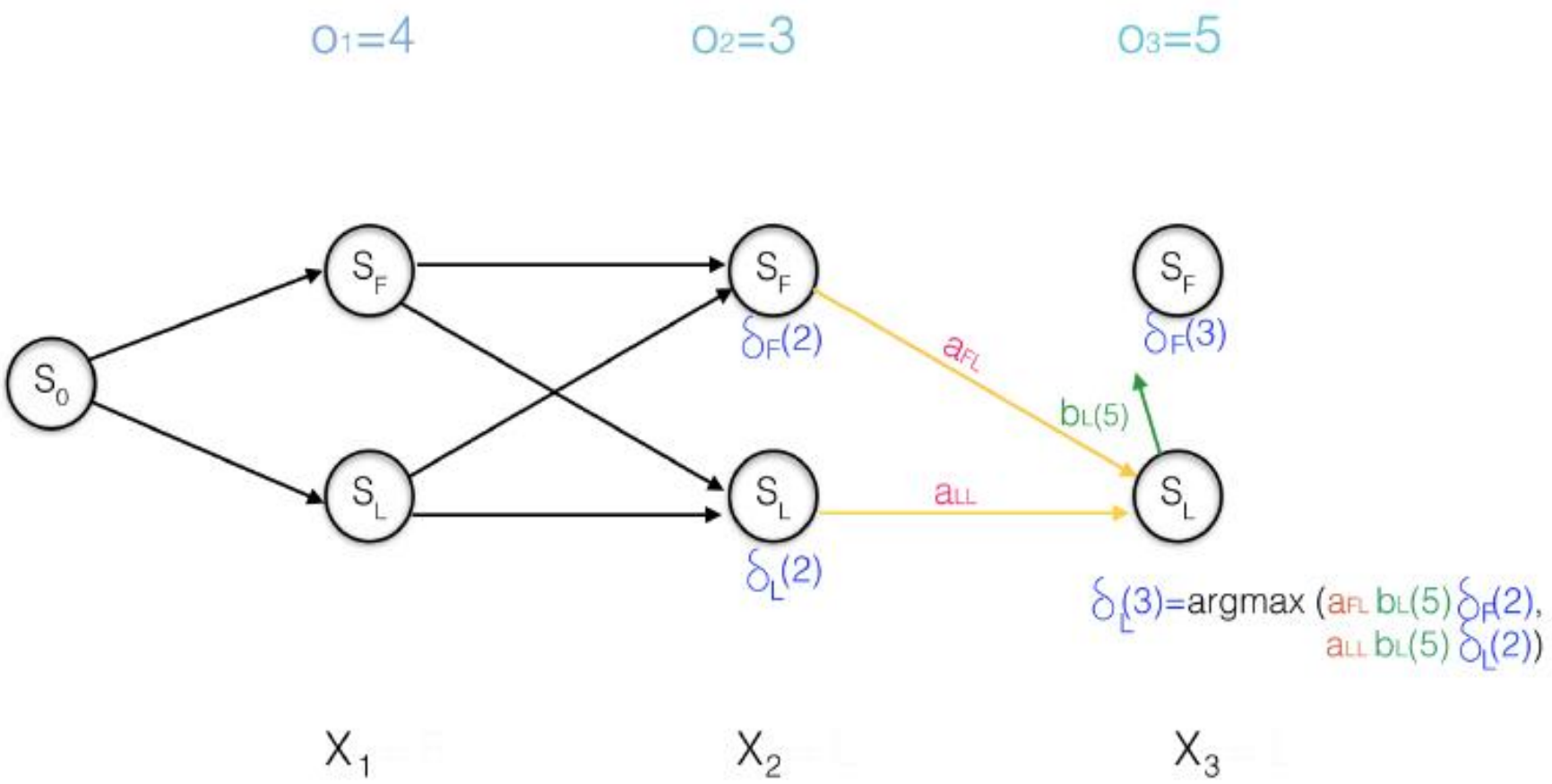
Viterbi algorithm, main step



Viterbi algorithm, main step, observation is 5



Viterbi algorithm, main step, observation is 5



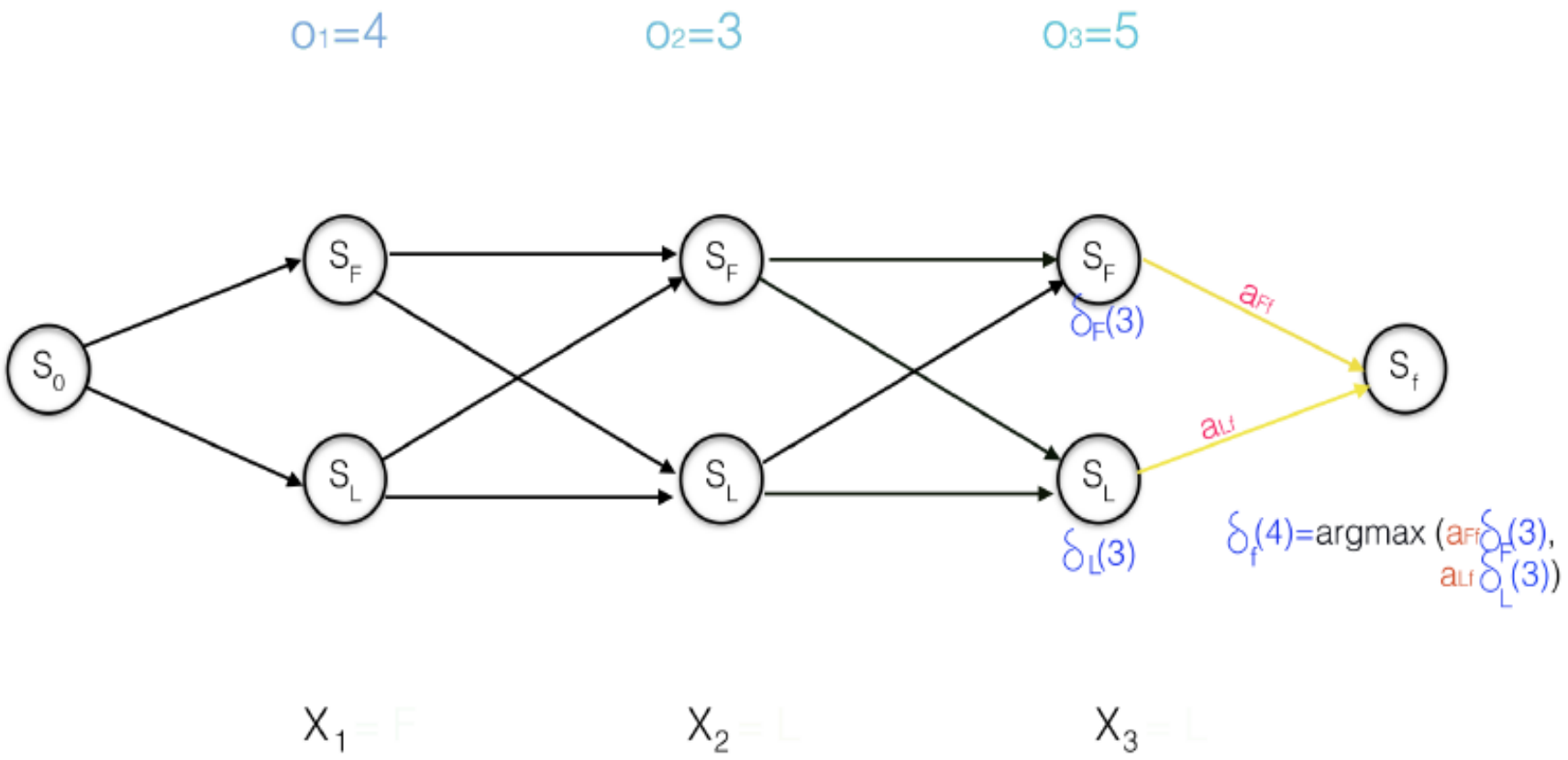
Viterbi algorithm, termination

- $\delta_f(T+1)$ is the probability of the entire state sequence up to point $T+1$ having been produced given the observation and the HMM's parameters.

$$P(X|O,\mu) = \delta_f(T+1) = \max_{1 \le i \le N} \delta_i \cdot (T) a_{if}$$

- It is calculated by maximising over the $\delta_i(T) \cdot a_{if}$, almost as per usual
- Not quite as per usual, because the final state s_f does not emit, so there is no $b_i(o_T)$ to consider.

Viterbi algorithm, termination



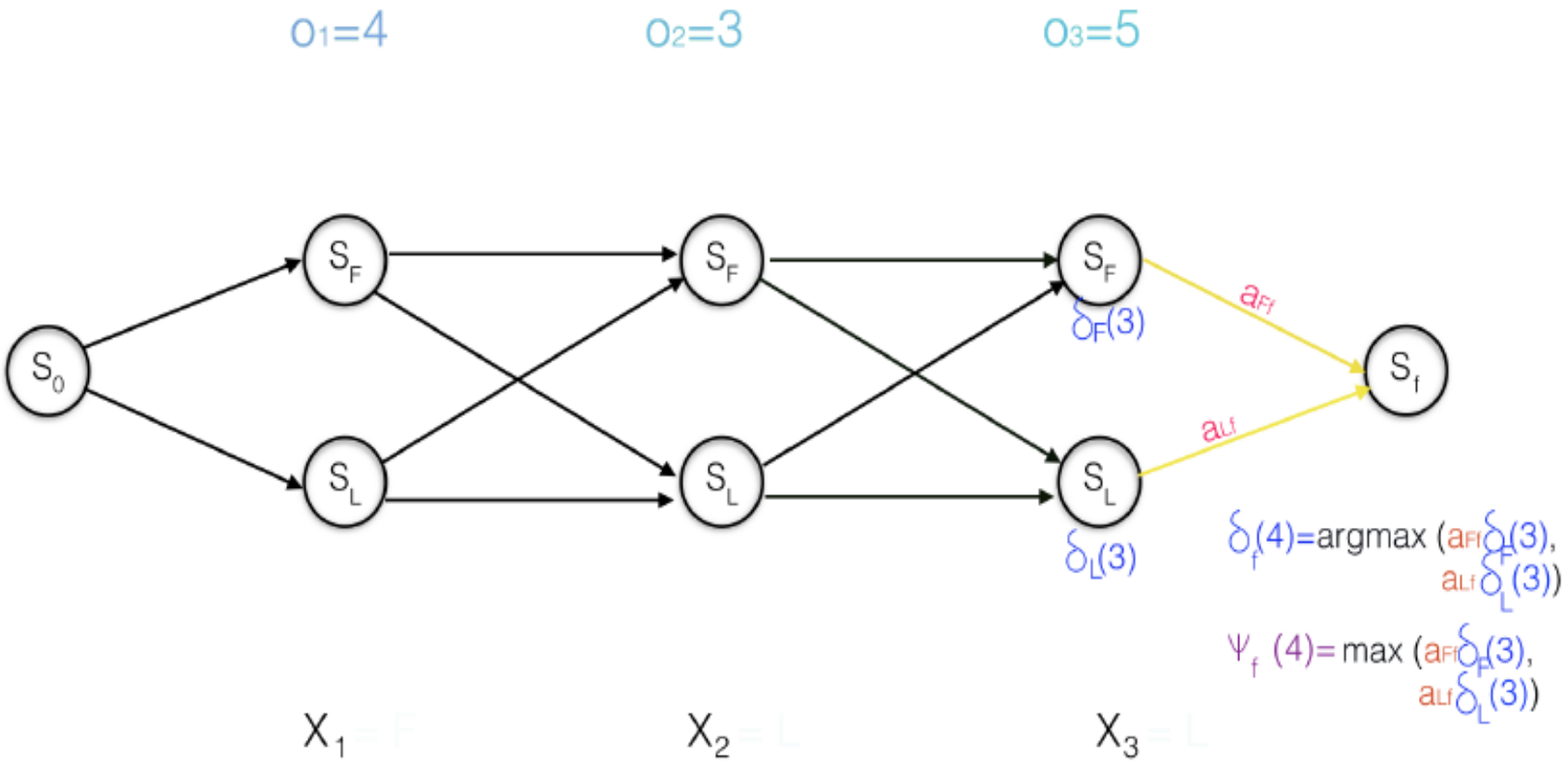
Viterbi algorithm, backtracking

- ψ_f is again calculated analogously to δ_f.

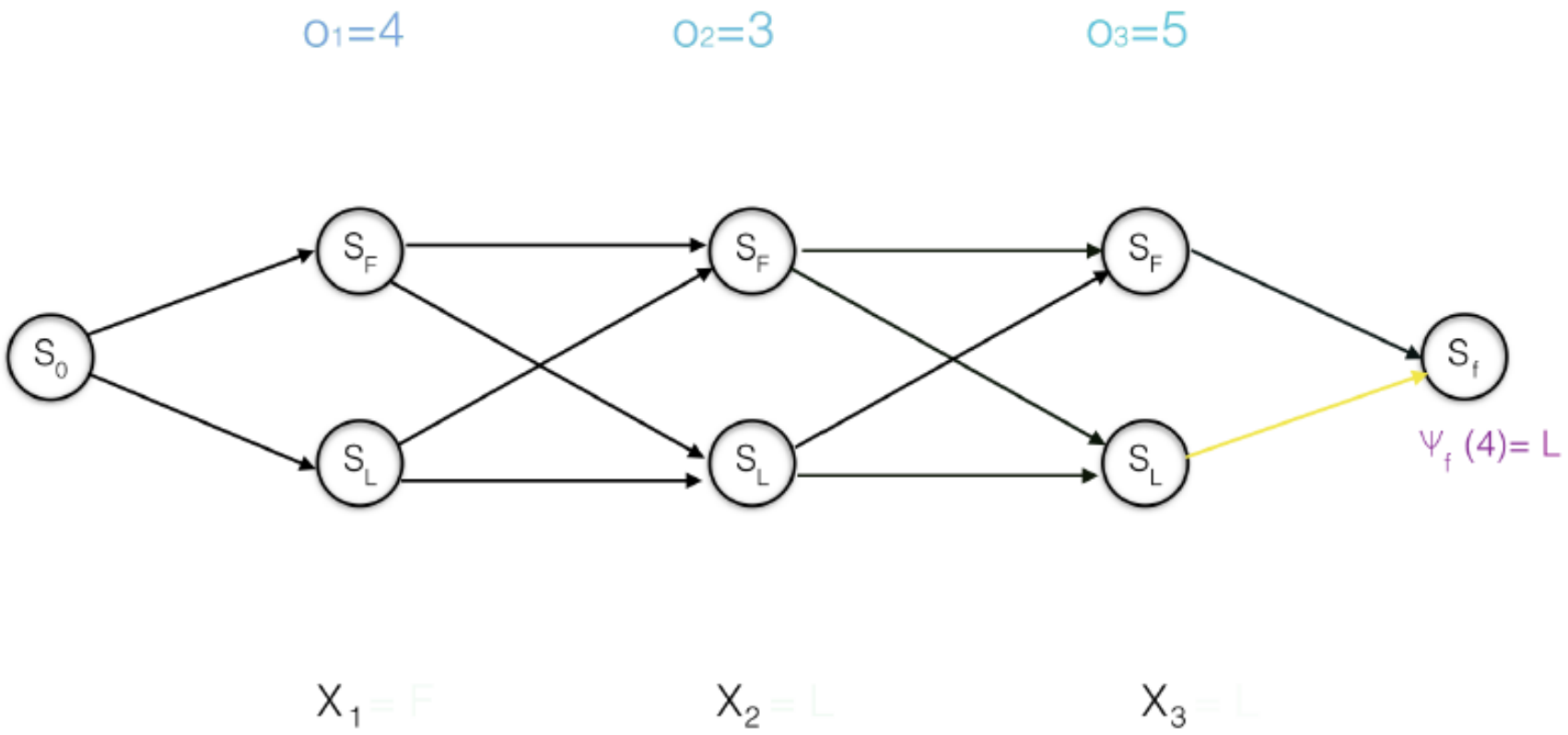
$$\psi_f(T+1) = \underset{1 \le i \le N}{\operatorname{argmax}}\, \delta_i(T) \cdot a_{if}$$

- It records X_T, the last state of the optimal state sequence.
- We will next go back to the cell concerned and look up **its** ψ to find the second-but-last state, and so on.

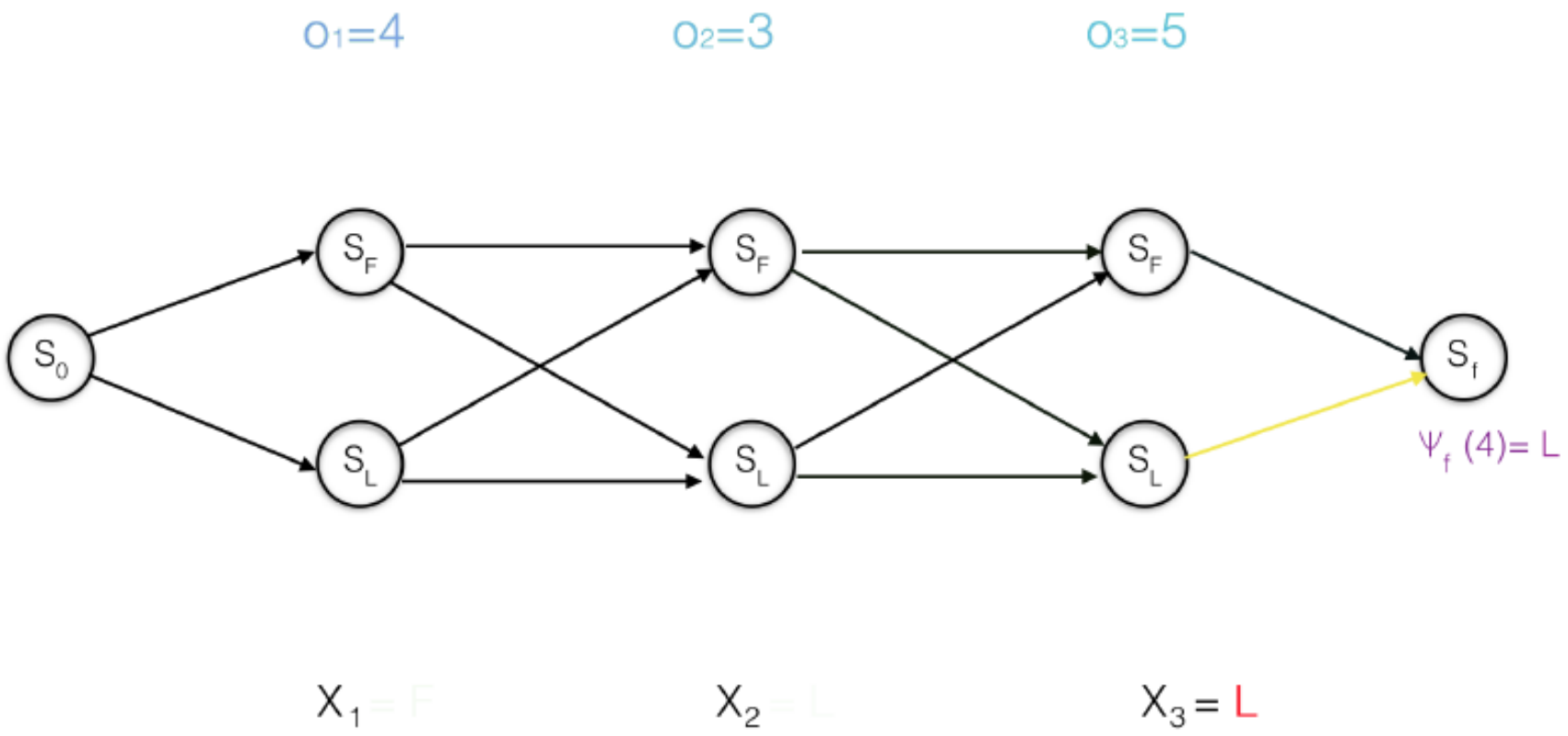
Viterbi algorithm, backtracking



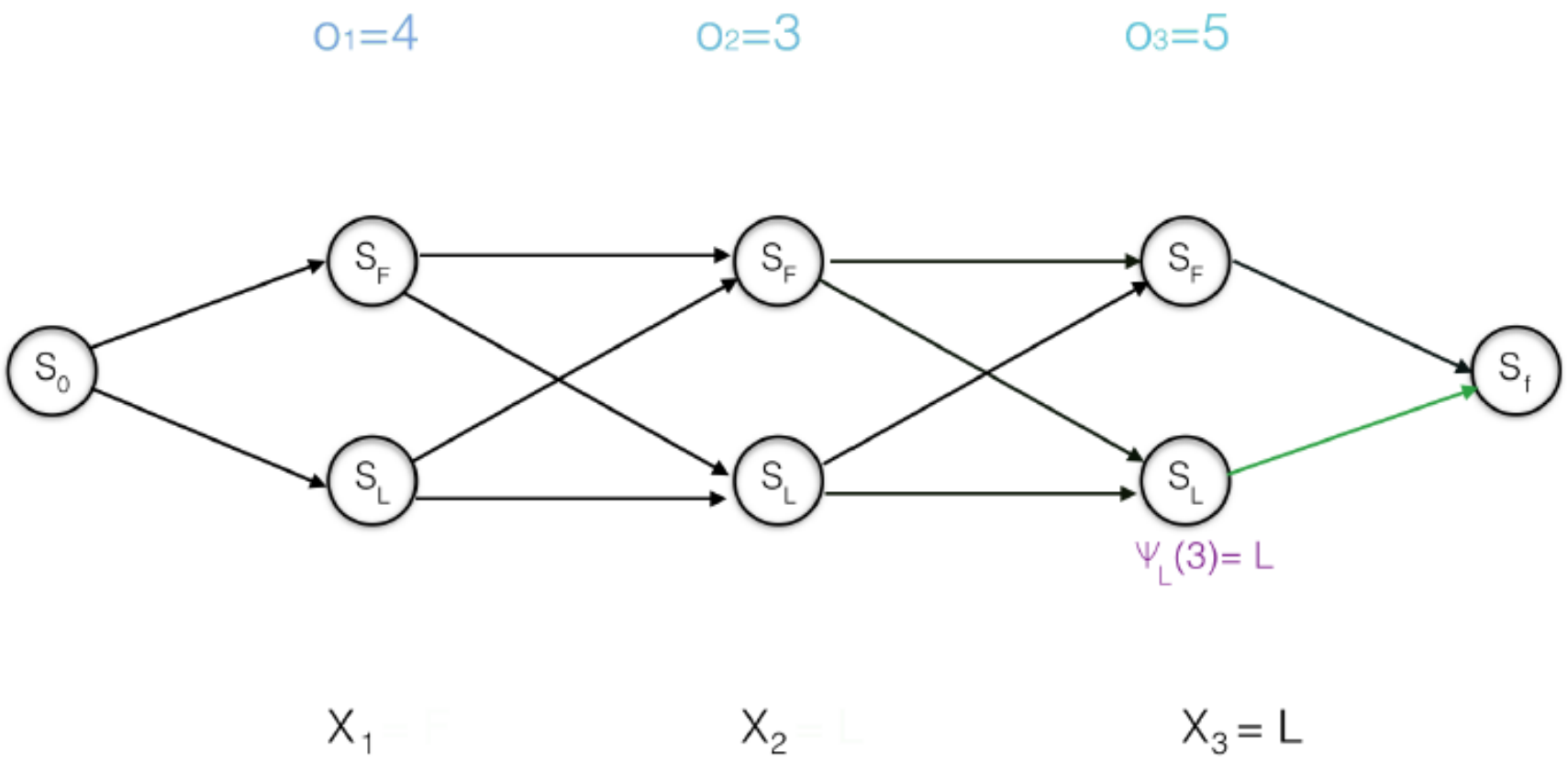
Viterbi algorithm, backtracking



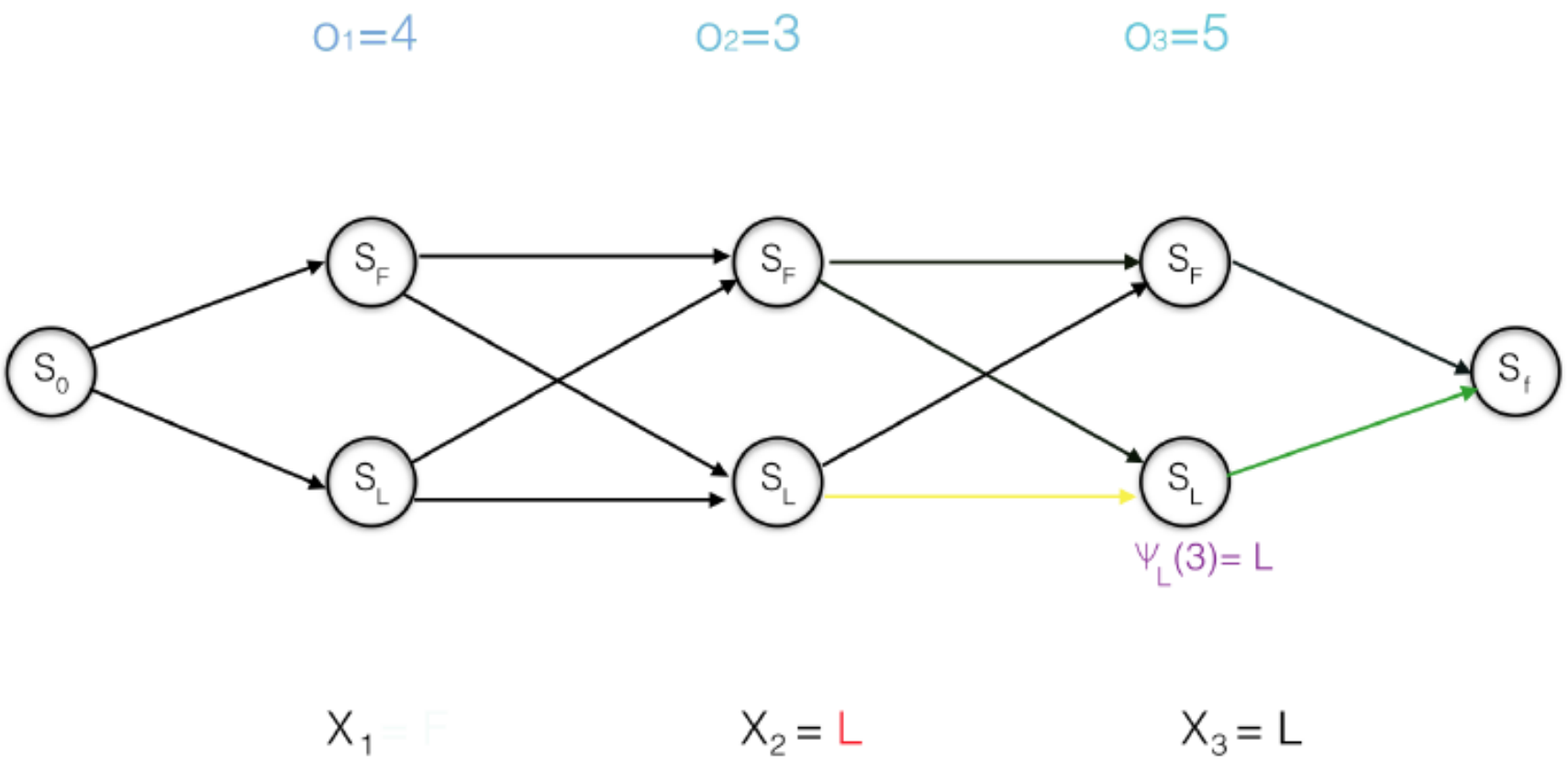
Viterbi algorithm, backtracking



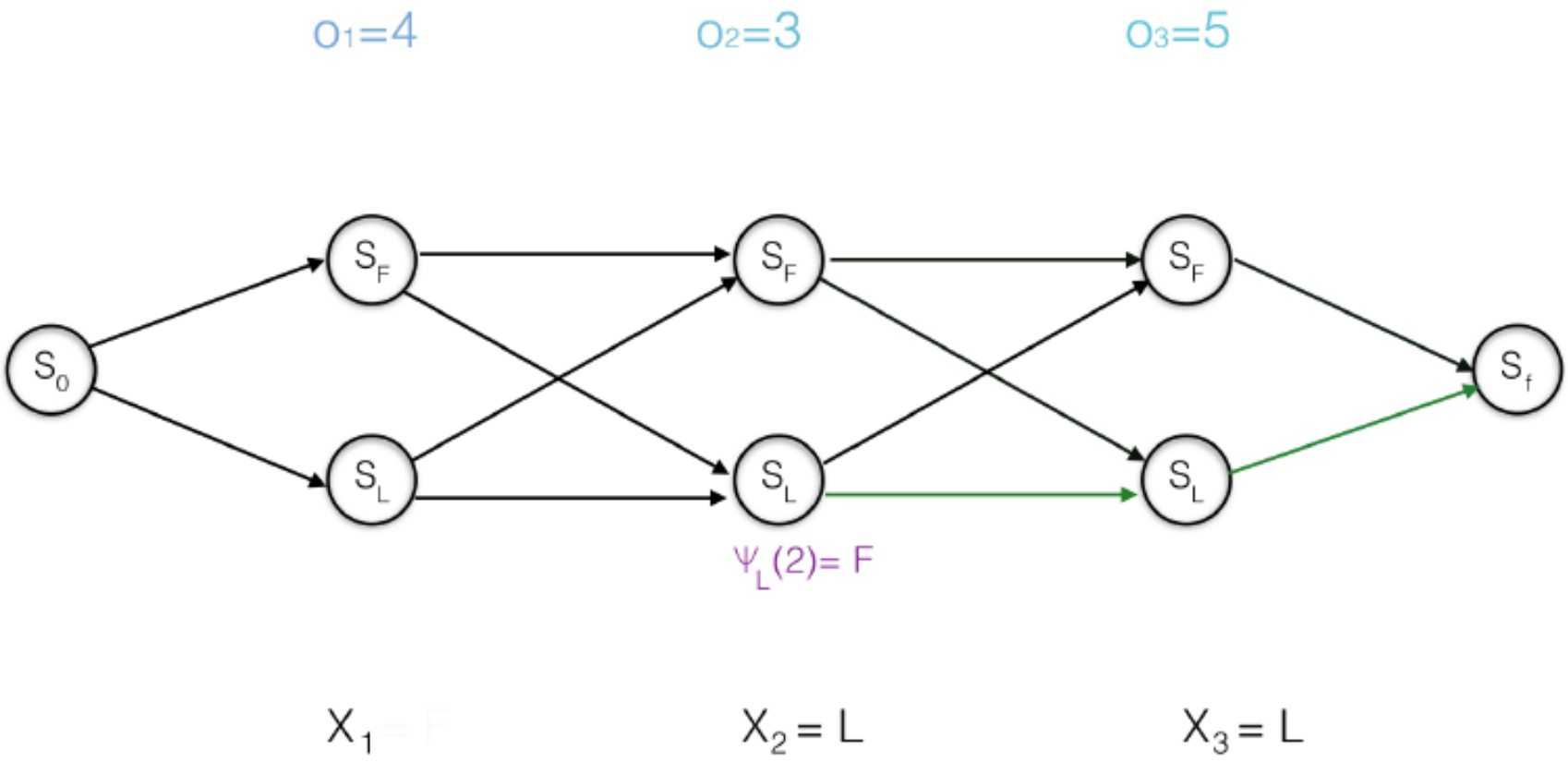
Viterbi algorithm, backtracking



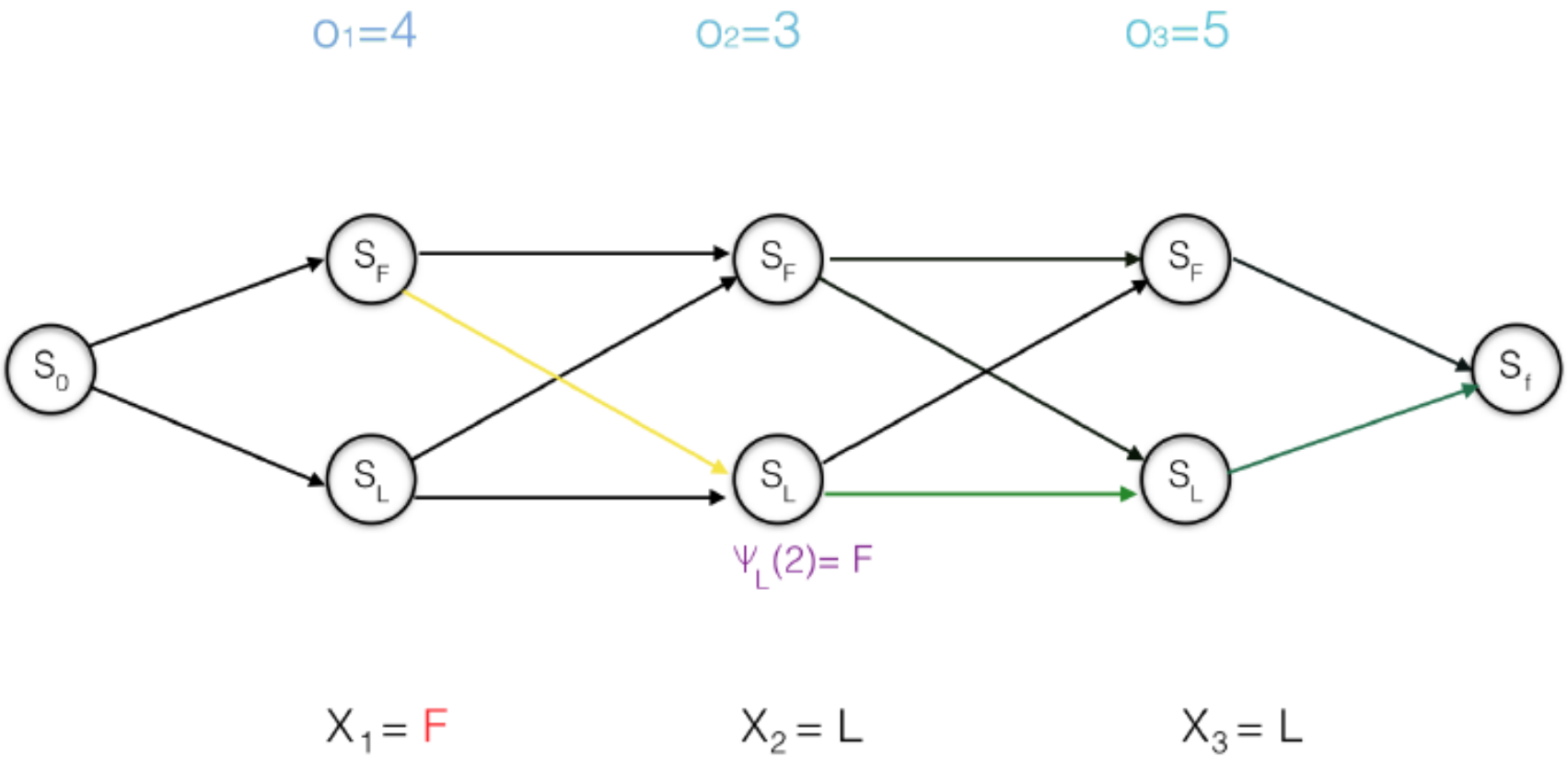
Viterbi algorithm, backtracking



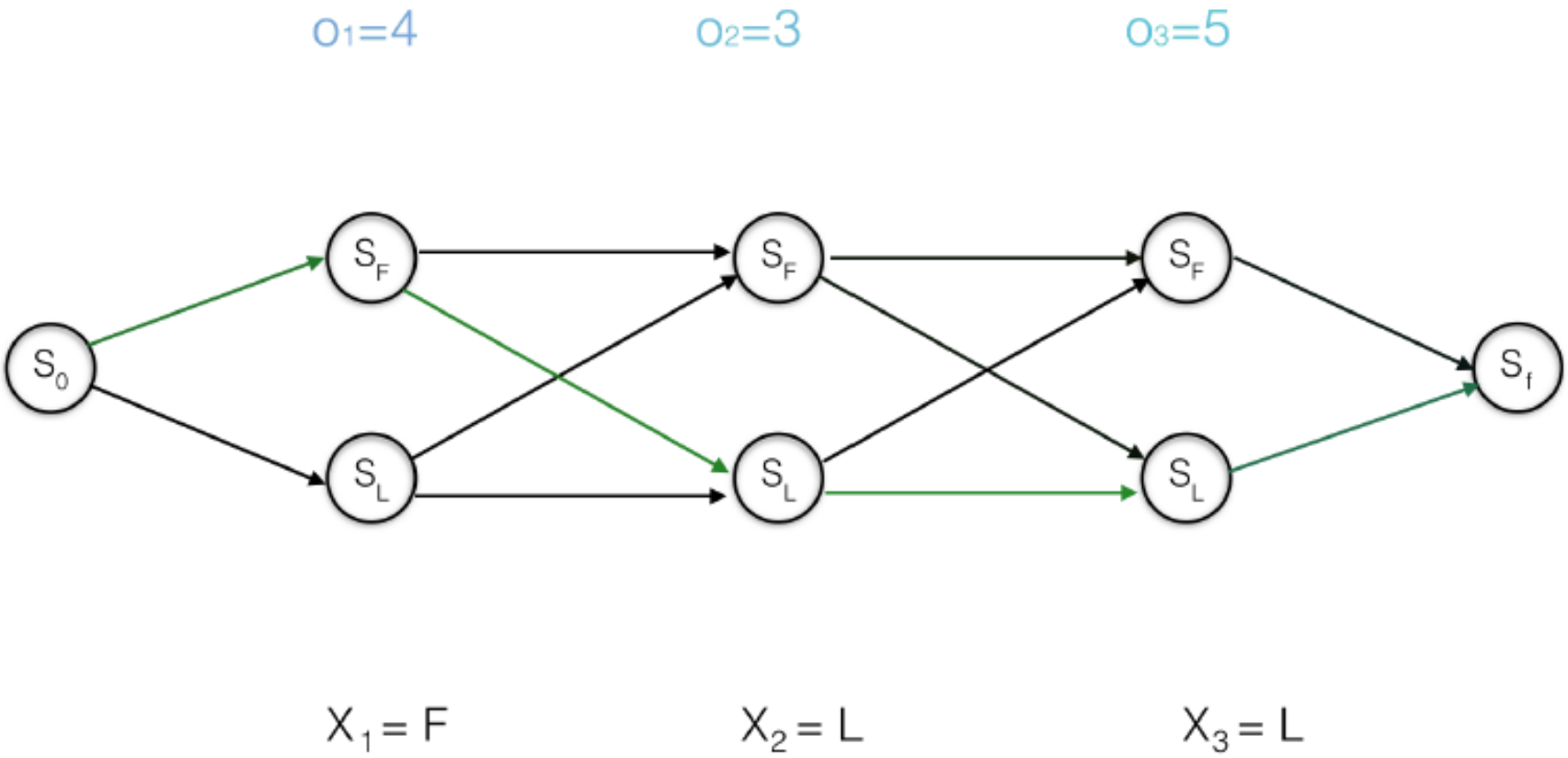
Viterbi algorithm, backtracking



Viterbi algorithm, backtracking



Viterbi algorithm, backtracking



Viterbi algorithm

- **Choose the most likely path**
- **Find the path** $(q_1, q_l, \ldots, q_T)$ **that maximizes the likelihood:**

$$P(q_1, q_2, \ldots, q_T | O, \lambda)$$

- **Solution by Dynamic Programming**
- **Define:**

$$\delta_l(i) = \max_{q_1, q_2, \ldots, q_{l-1}} P(q_1, q_2, \ldots, q_l = i, o_1, o_2, \ldots, o_l | \lambda)$$

- $\delta_l(i)$ **is the highest prob. path ending in state** i
- **By induction we have:**

$$\delta_{l+1}(j) = \max_i [\delta_l(i) a_{ij}] \cdot b_j(o_{l+1})$$

Viterbi Algorithm

- **Initialization:**

$$\delta_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N$$

$$\psi_1(i) = 0$$

- **Recursion:**

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

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}]$$

$$2 \leq t \leq T, 1 \leq j \leq N$$

- Termination:

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

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

- Path (state sequence) backtracking:

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

Types of HMM

- Continuous Density
- Ergodic
- State Duration

Continuous Density HMM

- Normally continuous density hidden Markov models (CDHMMs) use a mixture model like mixture-of-Gaussians as the distribution of the observations for a given state.
- This is, however, unnecessarily complicated for the HMM/NSSM hybrid so single Gaussians will be used as the observation model.

HMM/NSSM: Hidden Markov Model/Nonlinear State-Space Model

The model

- In the basic model:
  - ✓ The hidden state sequence is denoted by (M= M_1, … , M_T) and other parameters by Θ.
  - ✓ The observations (x(1), …., x(T)), given the corresponding hidden state, are assumed to be Gaussian with diagonal covariance matrix.
- Given the HMM state sequence M, the individual observations are assumed to be independent. Therefore the likelihood of the data can be written as:

$$P(\boldsymbol{X}|\boldsymbol{M},\boldsymbol{\theta}) = \prod_{t=1}^{T}\prod_{k=1}^{N} p(x_k(t)|M_t).$$

The model

- Because of the Markov property, the prior distribution of the probabilities of the hidden states can also be written in factorial form:

$$P(\boldsymbol{M}|\boldsymbol{\theta}) = p(M_1|\boldsymbol{\theta}) \prod_{t=1}^{T-1} p(M_{t+1}|M_t, \boldsymbol{\theta}).$$

- The factors of last 2 equations are defined to be:

$$p(x_k(t)|M_t = i) = N(x_k(t);\ m_k(i), \exp(2v_k(i)))$$

$$p(M_{t+1} = j|M_t = i, \boldsymbol{\theta}) = a_{ij}$$

$$p(M_1 = i|\boldsymbol{\theta}) = \pi_i.$$

The model

- The priors of all the parameters defined above are:

$$p(\mathbf{a}_{i,.}) = \text{Dirichlet}(\mathbf{a}_{i,.};\ \mathbf{u}_i^{(A)})$$

$$p(\boldsymbol{\pi}) = \text{Dirichlet}(\boldsymbol{\pi};\ \mathbf{u}^{(\pi)})$$

$$p(m_k(i)) = N(m_k(i);\ m_{m_k}, \exp(2v_{m_k}))$$

$$p(v_k(i)) = N(v_k(i);\ m_{v_k}, \exp(2v_{v_k})).$$

- These should be written as conditional distributions conditional to the parameters of the hyperprior but the conditioning variables have been dropped out to simplify the notation.

The model

- The parameters $\mu^{(\pi)}$ and $\mu_i^{(A)}$ of the Dirichlet priors are fixed.
- Their values should be chosen to reflect true prior knowledge on the possible initial states and transition probabilities of the chain.
- All the other parameters m_{mk}, v_{mk}, m_{vk} and v_{vk} have higher hierarchical priors. As the number of parameters in such priors grows, only the full structure of the hierarchical prior of m_{mk} is given. It is:

$$p(\boldsymbol{\pi}) = \text{Dirichlet}(\boldsymbol{\pi};\ \mathbf{u}^{(\pi)})$$

$$p(m_k(i)) = N(m_k(i);\ m_{m_k}, \exp(2v_{m_k}))$$

$$p(v_k(i)) = N(v_k(i);\ m_{v_k}, \exp(2v_{v_k})).$$

The model

- The hierarchical prior of for example m(i) can be summarized as follows:
  - ✓ The different components of m(i) have different priors whereas the vectors corresponding to different states of the HMM share a common prior, which is parameterised with m_{m_n}
  - ✓ The parameters m_{m_n} corresponding to different components of the original vector m(i) share a common prior parameterised with m_{m_m}.
  - ✓ The parameter m_{m_m} has a fixed noninformative prior.

- The set of model parameters Θ consists of all these parameters and all the parameters of the hierarchical priors.

The model

- In the hierarchical structure formulated above, the Gaussian prior for the mean m of a Gaussian is a conjugate prior. Thus the posterior will also be Gaussian.
- The parameterisation of the variance with

$$\sigma^2 = \exp(2v), \; v \sim N(\alpha, \beta)$$

  is somewhat less conventional. The conjugate prior for variance of a Gaussian is the inverse gamma distribution.
- Adding a new level of hierarchy for the parameters of such a distribution would, however, be significantly more difficult.

The model

- The present parameterisation allows adding similar layers of hierarchy for the parameters of the priors of m and v.
- In this parameterisation the posterior of v is not exactly Gaussian but it may be approximated with one.
- The exponential function will ensure that the variance will always be positive and the posterior will thus be closer to a Gaussian.

Resources

- Introduction to Hidden Markov Model and Its Application, 16 April 2005, by Dr. Sung-Jung Cho, Samsung Advanced Institute of Technology (SAIT)
- https://www.cs.helsinki.fi/u/ahonkela/dippa/node48.html#:~:text=Normally%20continuous%20density%20hidden%20Markov,used%20as%20the%20observation%20model.
- http://www.math.lsa.umich.edu/~dburns/547/HMM1.pdf
- https://towardsdatascience.com/markov-models-and-markov-chains-explained-in-real-life-probabilistic-workout-routine-65e47b5c9a73
- Viterbi Algorithm for HMM Decoding - Machine Learning and Real-world Data by Simone Teufel and Ann Copestake, Computer Laboratory, University of Cambridge, Lent 2017
- HMM- Tapas Kanungo, Centre for Automation Research, University of Maryland.