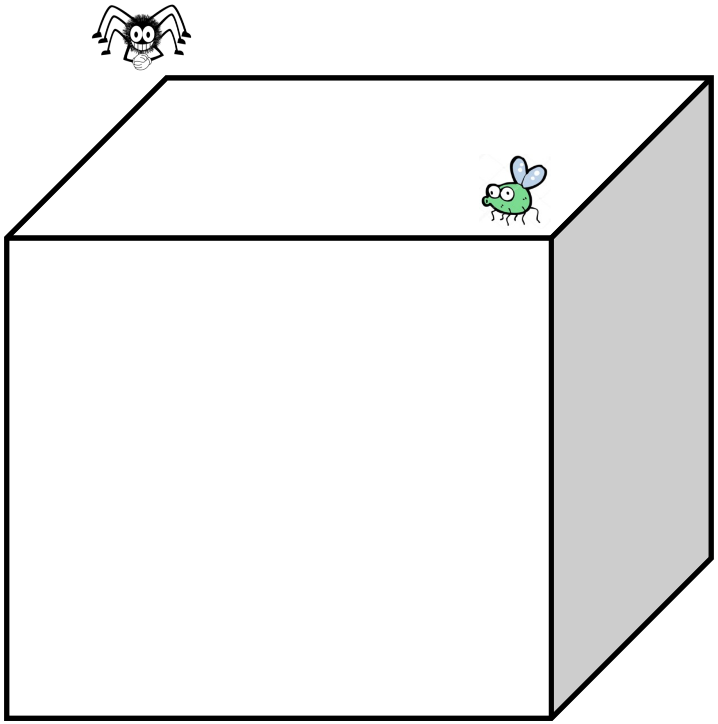
Machine Learning for Signal Processing

Hidden Markov Models

Bhiksha Raj

A quick intro to Markov Chains..



- The case of flider and spy..

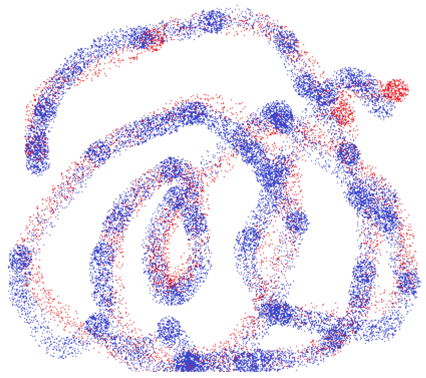
Prediction : a holy grail

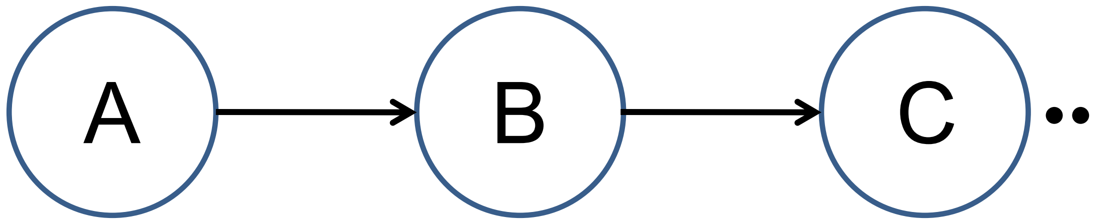
- Physical trajectories
  - Automobiles, rockets, heavenly bodies
- Natural phenomena
  - Weather
- Financial data
  - Stock market
- World affairs
  - Who is going to win the next election?
- Signals
  - Audio, video..

The wind and the target

- Aim: measure wind velocity accurately
  - For some important task
- Using a noisy wind speed sensor
  - E.g. arrows shot at a target
- Situation:
  - Wind speed at time t depends on speed at t-1
    - $S_t = S_{t-1} + \epsilon_t$
  - Arrow position at time t depends on wind speed at time t
    - $Y_t = AS_t + \gamma_t$
- Challenge: Given sequence of observation $Y_1, Y_2, \ldots, Y_t$
  - Estimate current wind speed S_t
  - Predict wind speed and arrow position at $t+1$: S_{t+1} and Y_{t+1}

A Common Trait

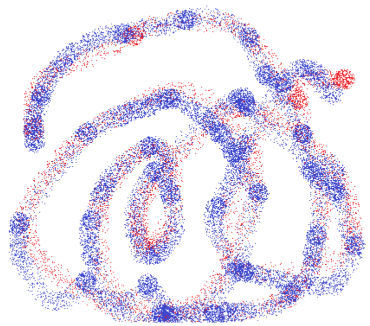




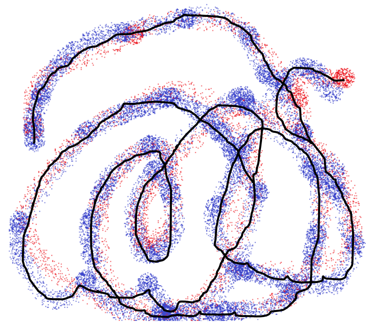
- ***Series data with trends***
- Stochastic functions of stochastic functions (of stochastic functions of …)
- An underlying process that progresses (seemingly) randomly
  - E.g. wind speed
  - E.g. Current position of a vehicle
  - E.g. current sentiment in stock market
- Random expressions of underlying process
  - E.g Wind speed sensor measurement
  - E.g. what you *see* from the vehicle
  - E.g. current stock prices of various stock

What a sensible agent must do

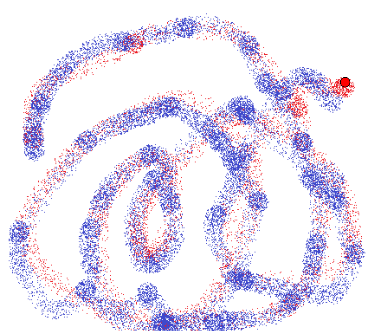
- *Learn* about the process
  - From whatever they know
    - E.g. learn the wind-speed function and the arrow-to-wind function
  - Basic requirement for other procedures



- *Track* underlying processes
  - Track the wind speed

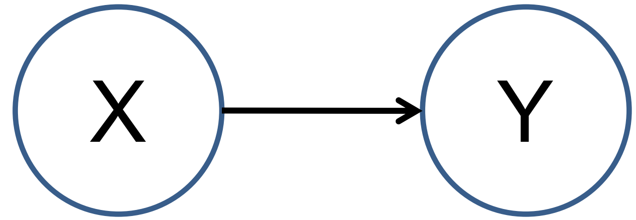


- Predict future values

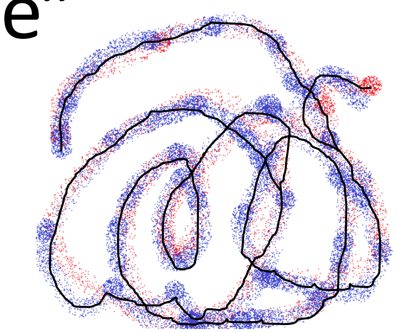


A Specific Form of Process..

- Doubly stochastic processes



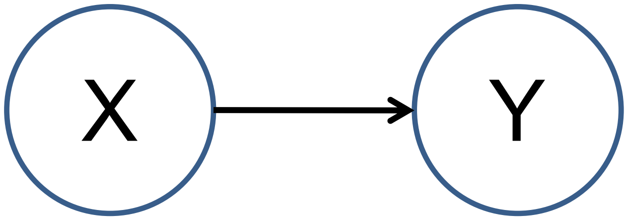
- One random process generates a "state" variable X
  - Random process X ⬅ P(X; Θ)



- Second-level process generates observations as a function of state X
- Random process Y ⬅ P(Y; f(X, Λ))

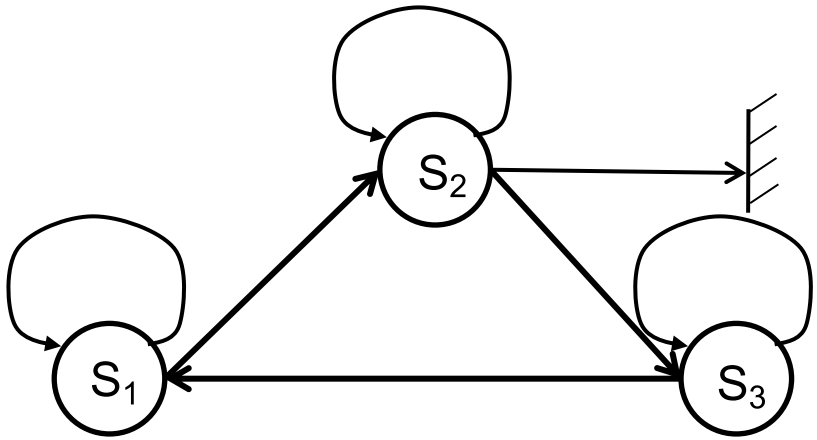
Doubly Stochastic Process *Model*

- Doubly stochastic processes are *models*
  - May not be a *true* representation of process underlying actual data



- First level variable may be a *quantifiable* variable
  - Position/state of vehicle
  - Second level variable is a stochastic function of position
- First level variable may *not* have meaning
  - “Sentiment” of a stock market
  - “Configuration” of vocal tract

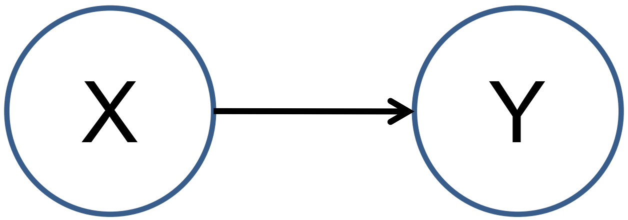
Markov Chain



- Process can go through a number of states
  – Random walk, Brownian motion..
- From each state, it can go to any other state with a probability
  – Which only depends on the current state
- Walk goes on forever
  – Or until it hits an “absorbing wall”
- Output of the process – a sequence of states the process went through

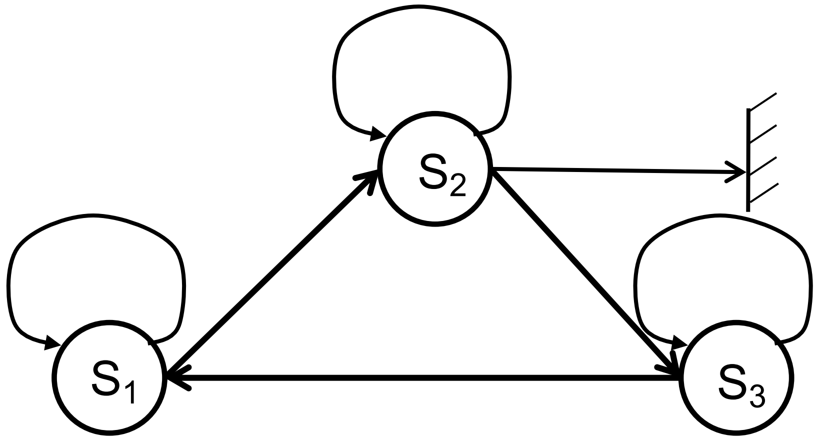
Stochastic Function of a Markov Chain

- First-level variable is ***usually*** abstract



- The first level variable assumed to be the output of a Markov Chain
- The second level variable is a random variable whose distribution is a function of the output of the Markov Chain
- Also called an HMM
- Another variant – stochastic function of Markov *process*
  - *Kalman Filtering..*

Stochastic Function of a Markov Chain



- Output:
  - $Y \leftarrow P(Y; f(S_i))$
    - Probability distribution is a function of the state

A little parable

You've been kidnapped

A little parable

You've been kidnapped

And blindfolded

A little parable

You've been kidnapped

And blindfolded

You can only *hear* the car
You must find your way back home from wherever they drop you off

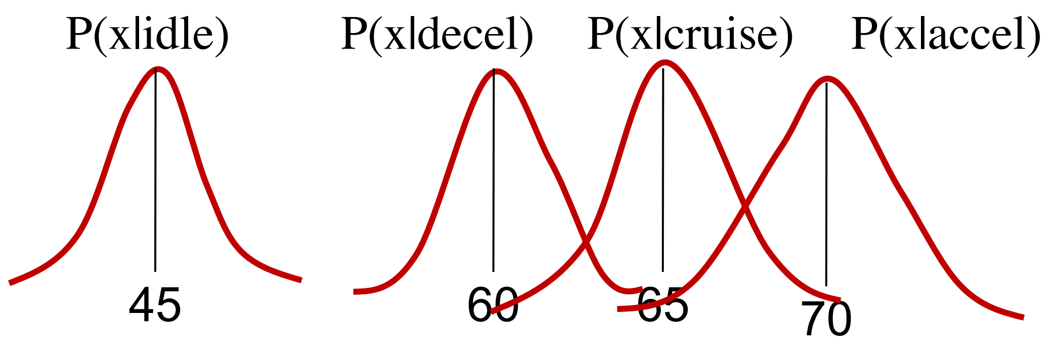
Kidnapped

- Determine automatically, by only *listening* to a running automobile, if it is:
  - Idling; or
  - Travelling at constant velocity; or
  - Accelerating; or
  - Decelerating
- You are super acoustically sensitive and can determine sound pressure level (SPL)
  - The SPL is measured once per second

What you know

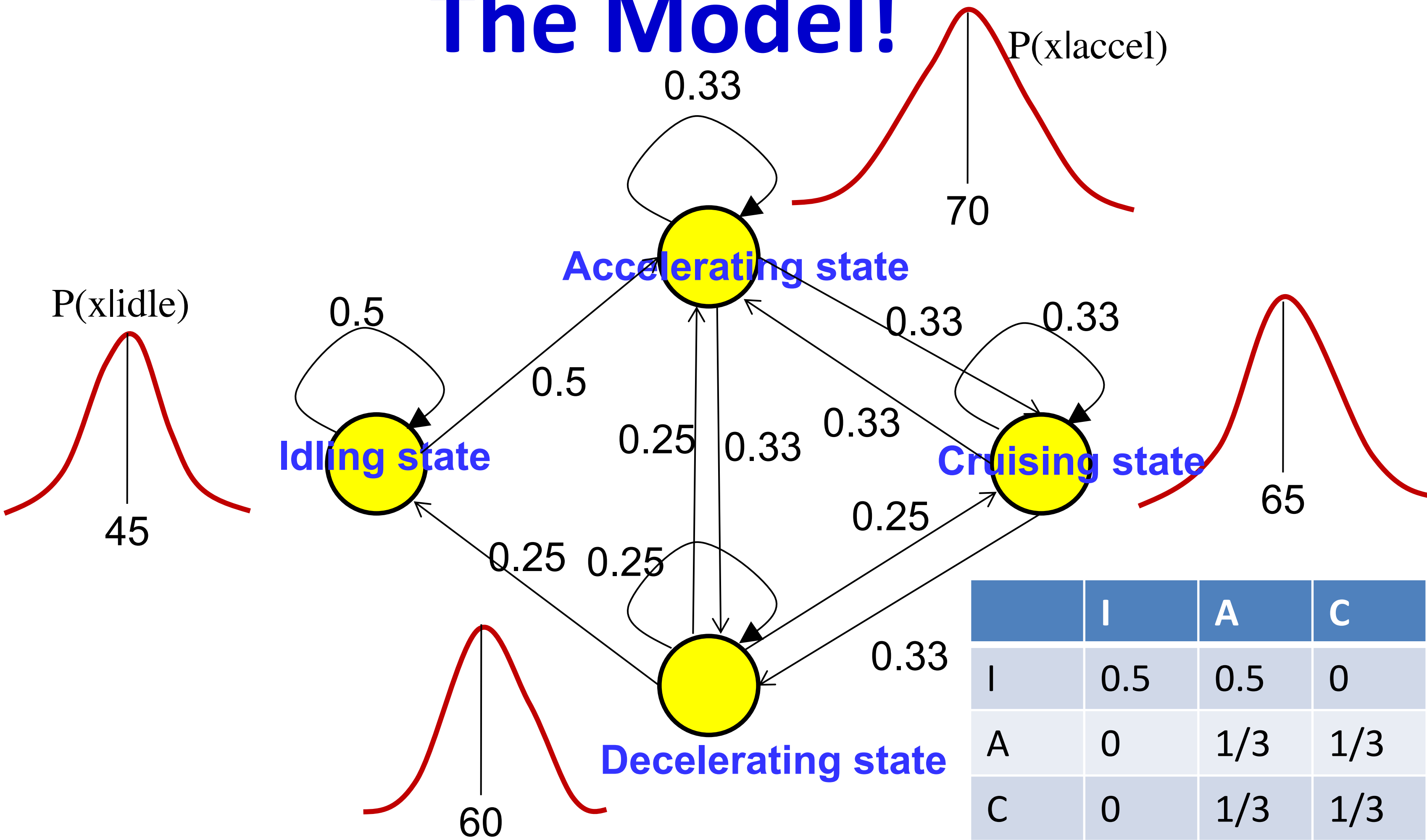
- An automobile that is at rest can accelerate, or continue to stay at rest
- An accelerating automobile can hit a steady-state velocity, continue to accelerate, or decelerate
- A decelerating automobile can continue to decelerate, come to rest, cruise, or accelerate
- An automobile at a steady-state velocity can stay in steady state, accelerate or decelerate

What else you know



- The probability distribution of the SPL of the sound is different in the various conditions
  - As shown in figure
    - In reality, depends on the car
- The distributions for the different conditions overlap
  - Simply knowing the current sound level is not enough to know the state of the car

The Model!



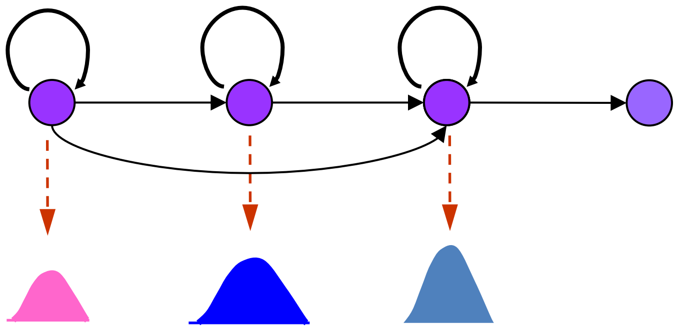
	I	A	C	D
I	0.5	0.5	0	0
A	0	1/3	1/3	1/3
C	0	1/3	1/3	1/3
D	0.25	0.25	0.25	0.25

- The state-space model
  - Assuming all transitions from a state are equally probable
- We will help you find your way back home in the next class

What is an HMM

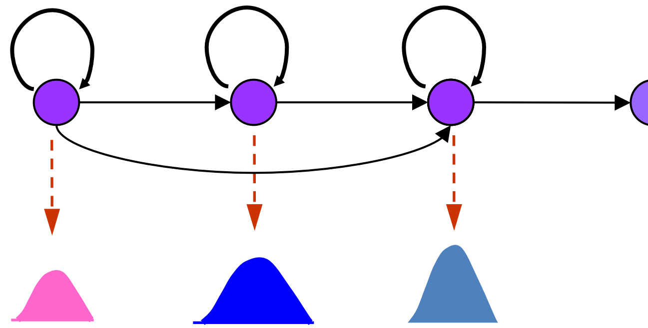
- The model assumes that the process can be in one of a number of states at any time instant
- The state of the process at any time instant depends only on the state at the previous instant (causality, Markovian)
- At each instant the process generates an observation from a probability distribution that is specific to the current state
- The generated observations are all that we get to see
  - the actual state of the process is not directly observable
    - Hence the qualifier hidden

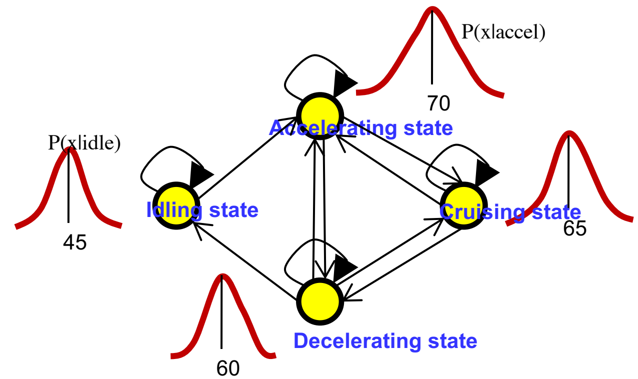
What is an HMM



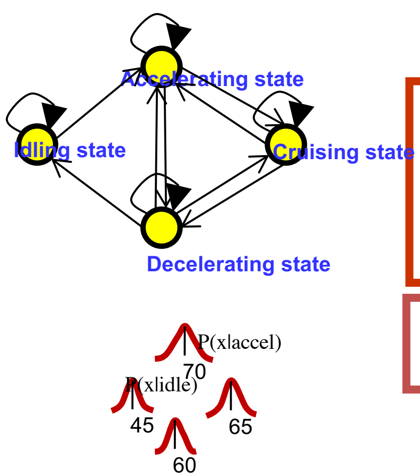
- “Probabilistic function of a markov chain”
- Models a dynamical system
- System goes through a number of states
  - Following a Markov chain model
- On arriving at any state it generates observations according to a state-specific probability distribution

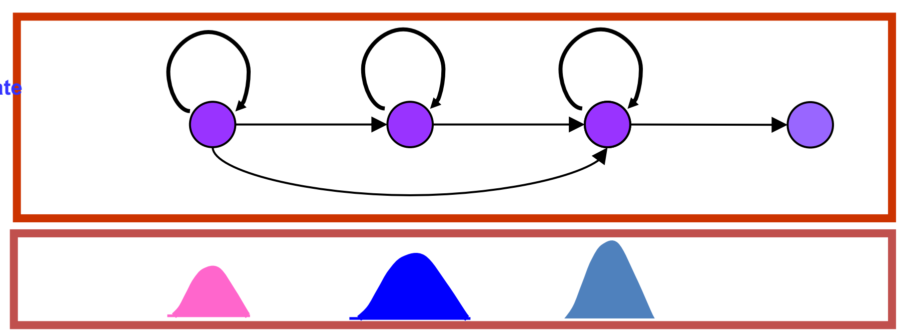
Hidden Markov Models





- A Hidden Markov Model consists of two components
  - A state/transition backbone that specifies how many states there are, and how they can follow one another
  - A set of probability distributions, one for each state, which specifies the distribution of all vectors in that state

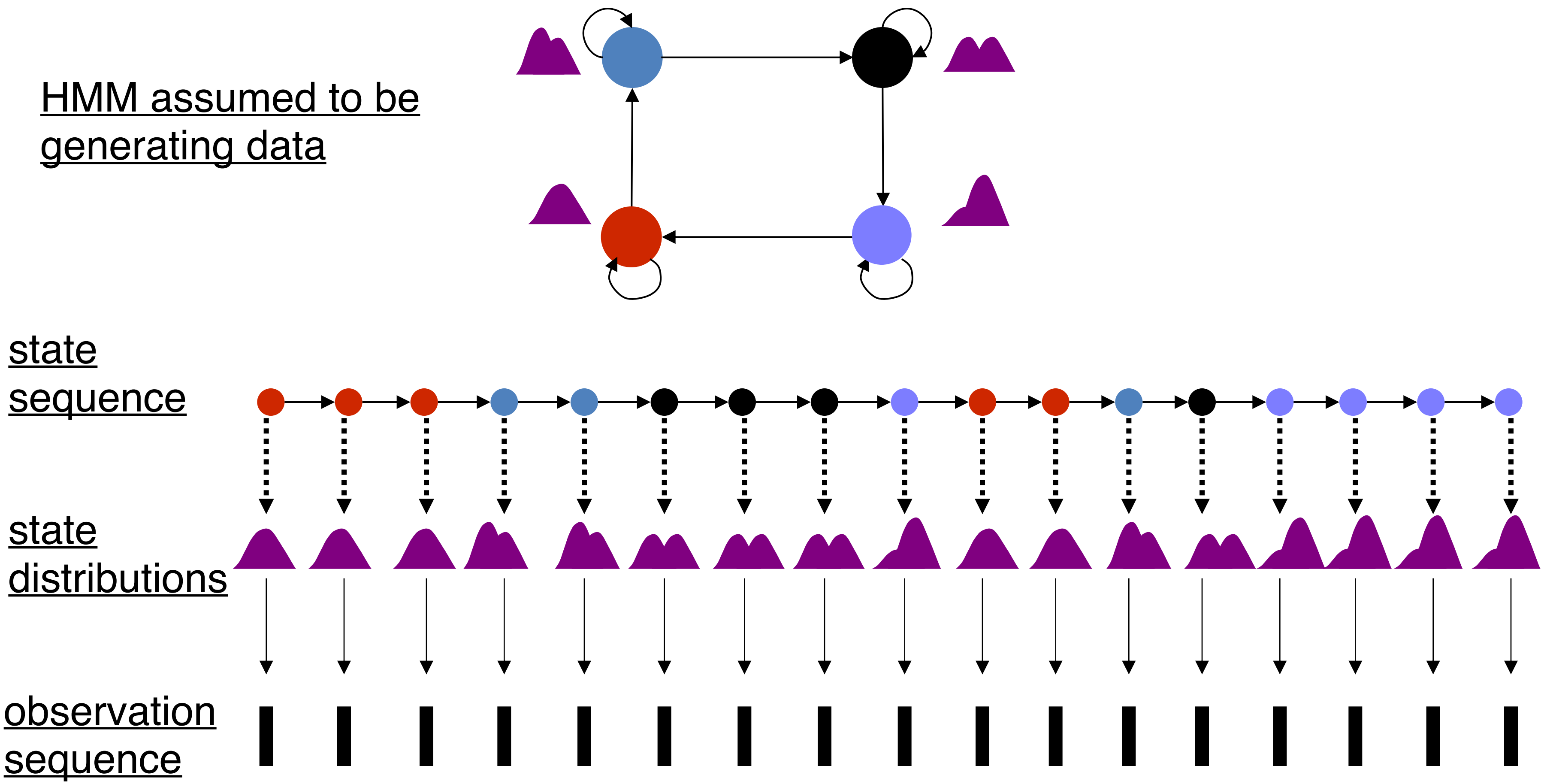




Markov chain

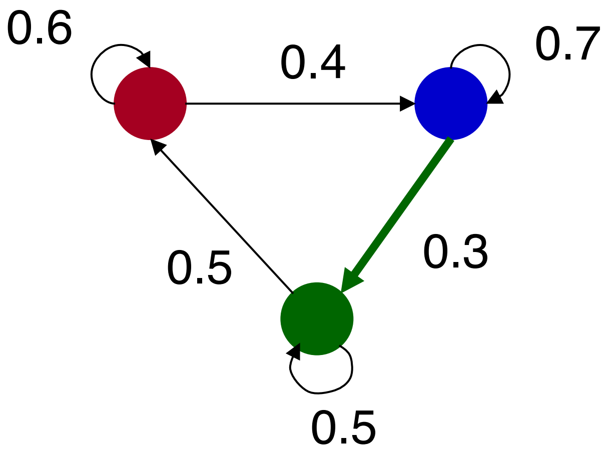
Data distributions

How an HMM models a process

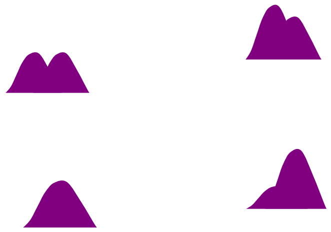


HMM Parameters

- The *topology* of the HMM
  - Number of states and allowed transitions
  - E.g. here we have 3 states and cannot go from the blue state to the red
- The transition probabilities
  - Often represented as a matrix as here
  - T_{ij} is the probability that when in state i, the process will move to j
- The probability π_i of beginning at any state s_i
  - The complete set is represented as π
- The *state output distributions*



$$T = \begin{pmatrix} .6 & .4 & 0 \\ 0 & .7 & .3 \\ .5 & 0 & .5 \end{pmatrix}$$



Three Basic HMM Problems

- What is the probability that it will generate a specific observation sequence

- Given an observation sequence, how do we determine which observation was generated from which state
  - The state segmentation problem

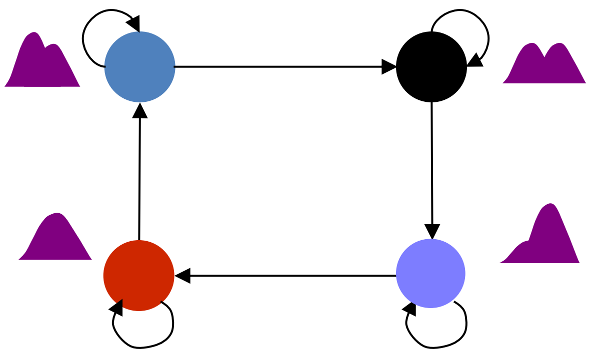
- How do we *learn* the parameters of the HMM from observation sequences

Computing the Probability of an Observation Sequence

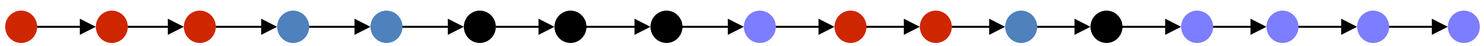
- Two aspects to producing the observation:
  - Progressing through a sequence of states
  - Producing observations from these states

Progressing through states

HMM assumed to be generating data



state sequence



- The process begins at some state (red) here
- From that state, it makes an allowed transition
  - To arrive at the same or any other state
- From that state it makes another allowed transition
  - And so on

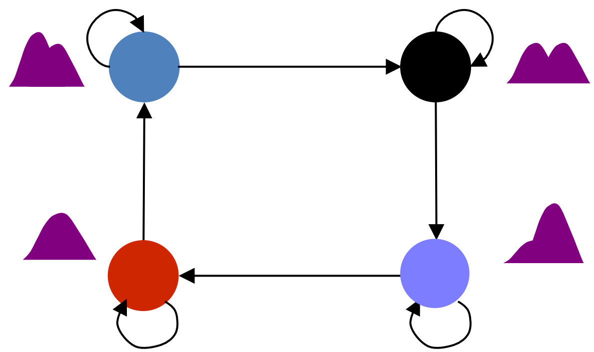
Probability that the HMM will follow a particular state sequence

$$P(s_1, s_2, s_3, \ldots) = P(s_1)P(s_2|s_1)P(s_3|s_2)\ldots$$

- $P(s_1)$ is the probability that the process will initially be in state s_1
- $P(s_j \mid s_i)$ is the transition probability of moving to state s_j at the next time instant when the system is currently in s_i
  - Also denoted by T_{ij} earlier

Generating Observations from States

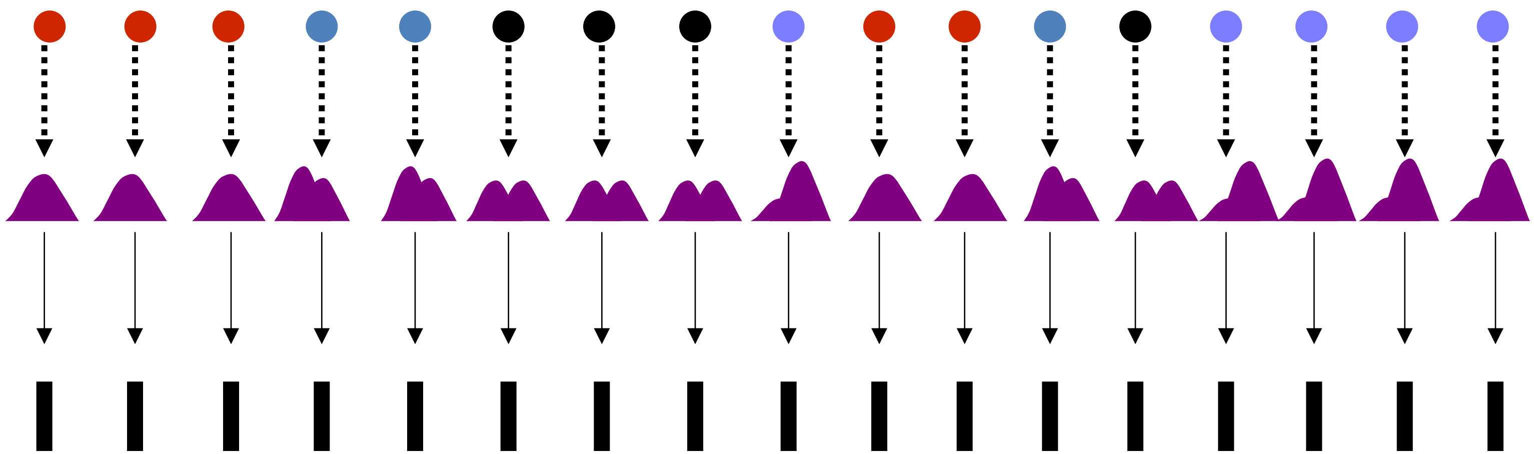
HMM assumed to be generating data



state sequence

state distributions

observation sequence



- At each time it generates an observation from the state it is in at that time

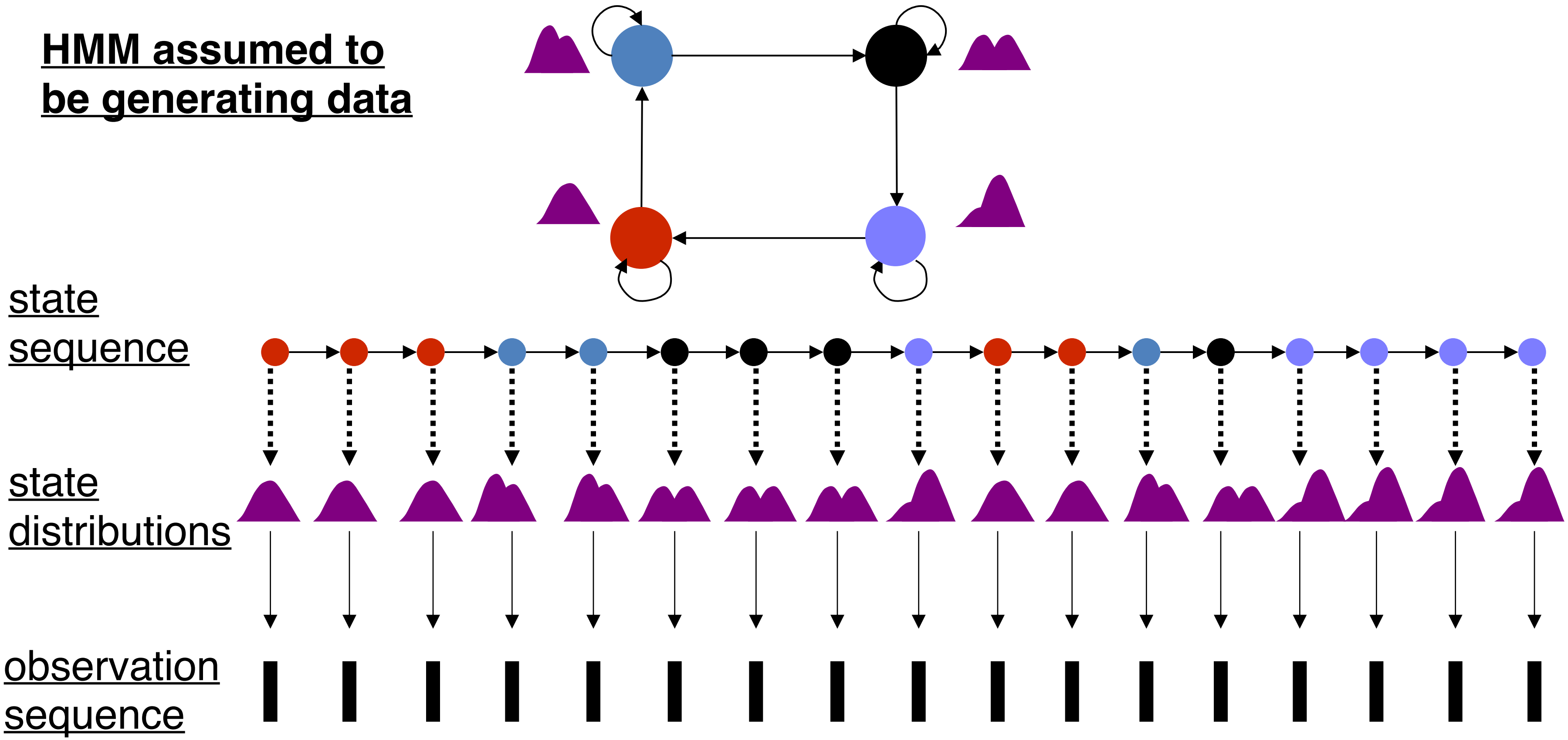
Probability that the HMM will generate a particular observation sequence given a state sequence (state sequence known)

$$P(o_1, o_2, o_3, \ldots | s_1, s_2, s_3, \ldots) = P(o_1|s_1)P(o_2|s_2)P(o_3|s_3)\ldots$$

Computed from the Gaussian or Gaussian mixture for state s_1

- $P(o_i \mid s_i)$ is the probability of generating observation o_i when the system is in state s_i

Proceeding through States and Producing Observations



- At each time it produces an observation and makes a transition

Probability that the HMM will generate a particular state sequence and from it, a particular observation sequence

$$P(o_1, o_2, o_3, \ldots, s_1, s_2, s_3, \ldots) =$$

$$P(o_1, o_2, o_3, \ldots | s_1, s_2, s_3, \ldots) P(s_1, s_2, s_3, \ldots) =$$

$$P(o_1|s_1) P(o_2|s_2) P(o_3|s_3) \ldots P(s_1) P(s_2|s_1) P(s_3|s_2) \ldots$$

Probability of Generating an Observation Sequence

- The precise state sequence is not known
- All possible state sequences must be considered

$$P(o_1, o_2, o_3, \ldots) = \sum_{\substack{all.possible \\ state.sequences}} P(o_1, o_2, o_3, \ldots, s_1, s_2, s_3, \ldots) =$$

$$\sum_{\substack{all.possible \\ state.sequences}} P(o_1|s_1) P(o_2|s_2) P(o_3|s_3) \ldots P(s_1) P(s_2|s_1) P(s_3|s_2) \ldots$$

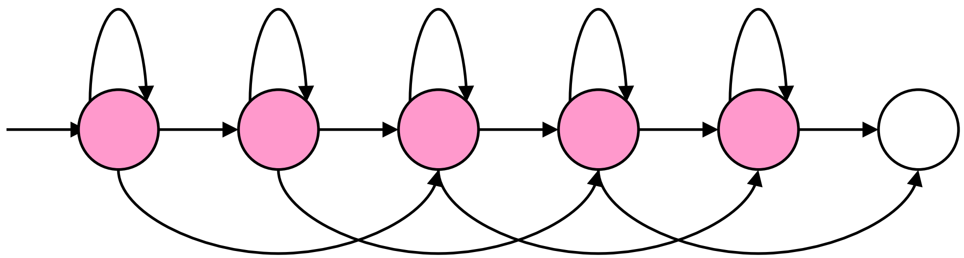
Computing it Efficiently

- Explicit summing over all state sequences is not tractable
  - A very large number of possible state sequences

- Instead we use the forward algorithm

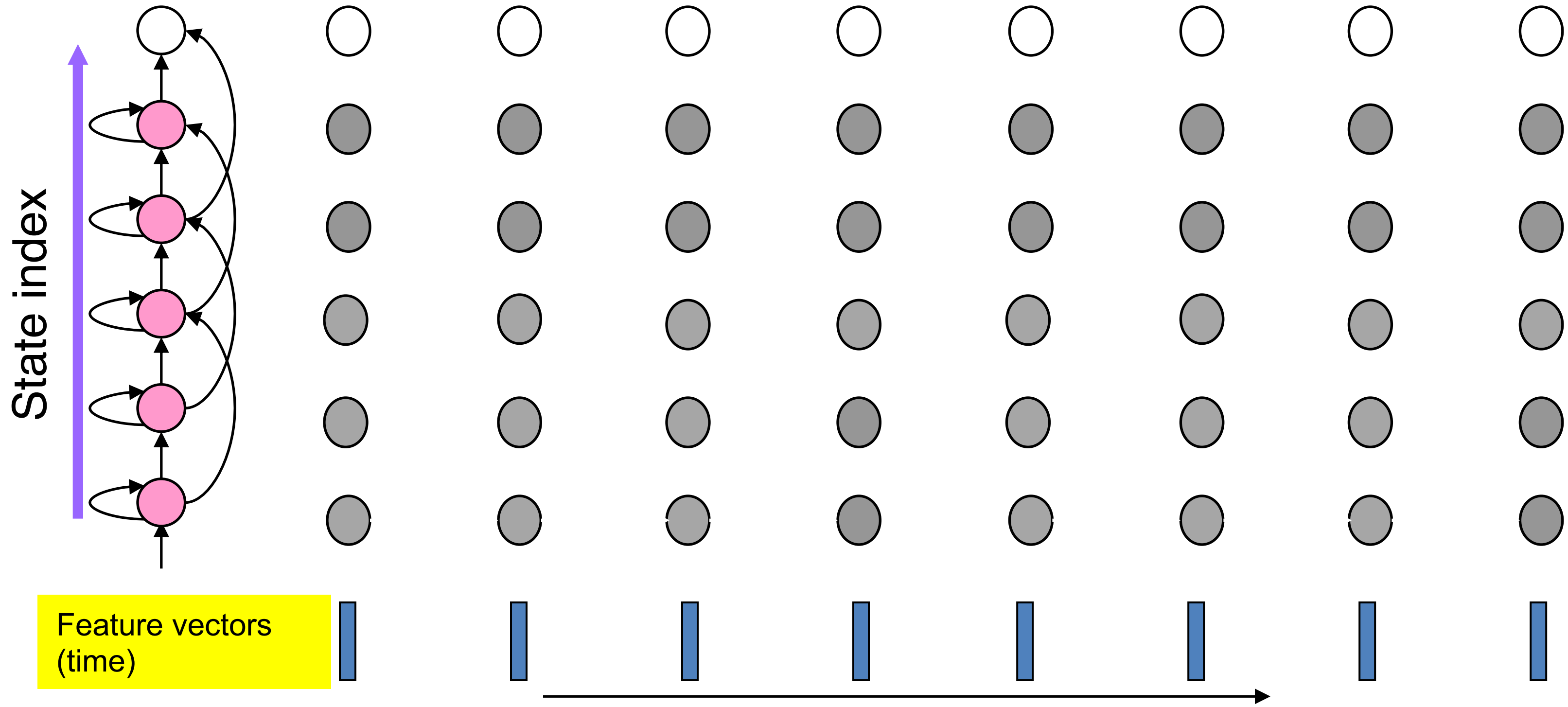
- A dynamic programming technique.

Illustrative Example



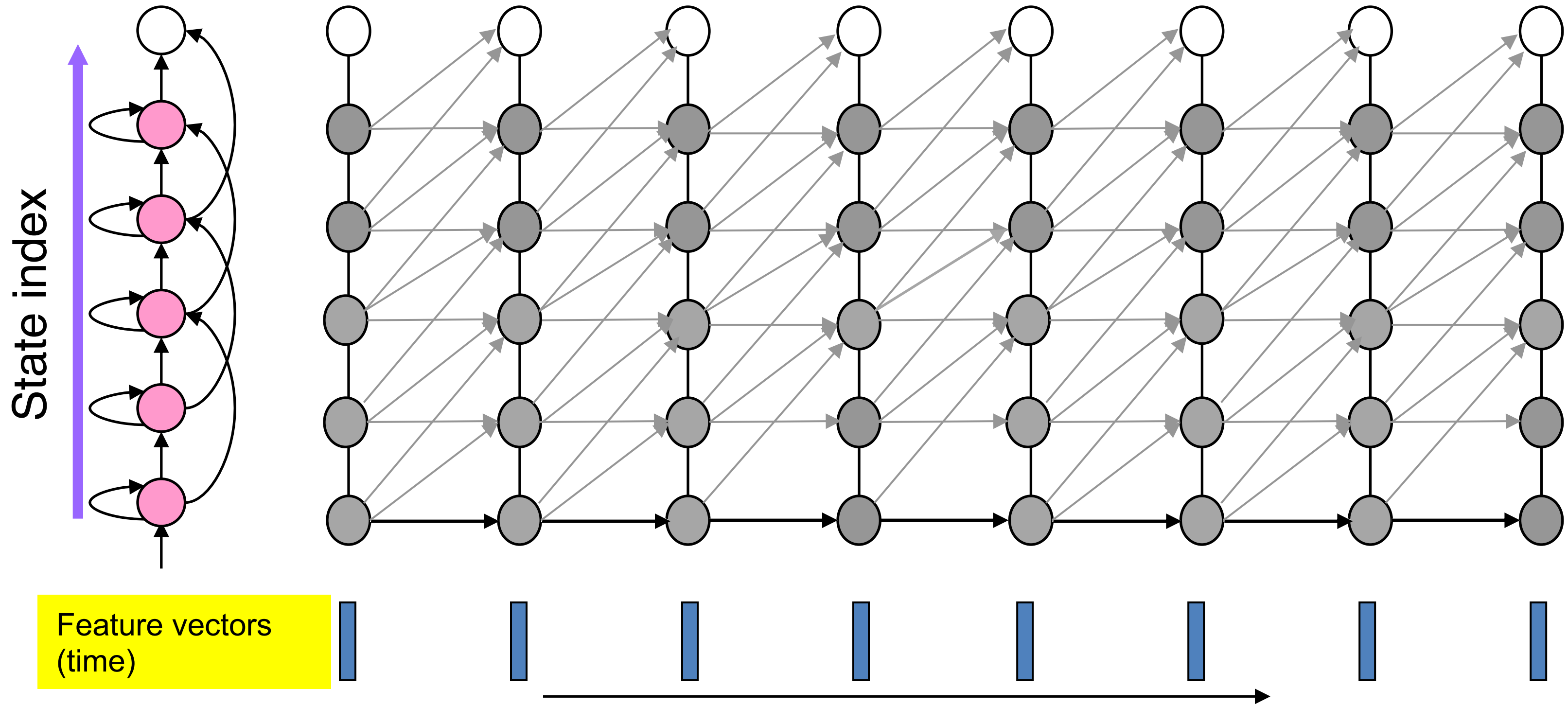
- Example: a generic HMM with 5 states and a "terminating state".
  - Left to right topology
    - $P(s_i)$ = 1 for state 1 and 0 for others
  - The arrows represent transition for which the probability is not 0

States and times…



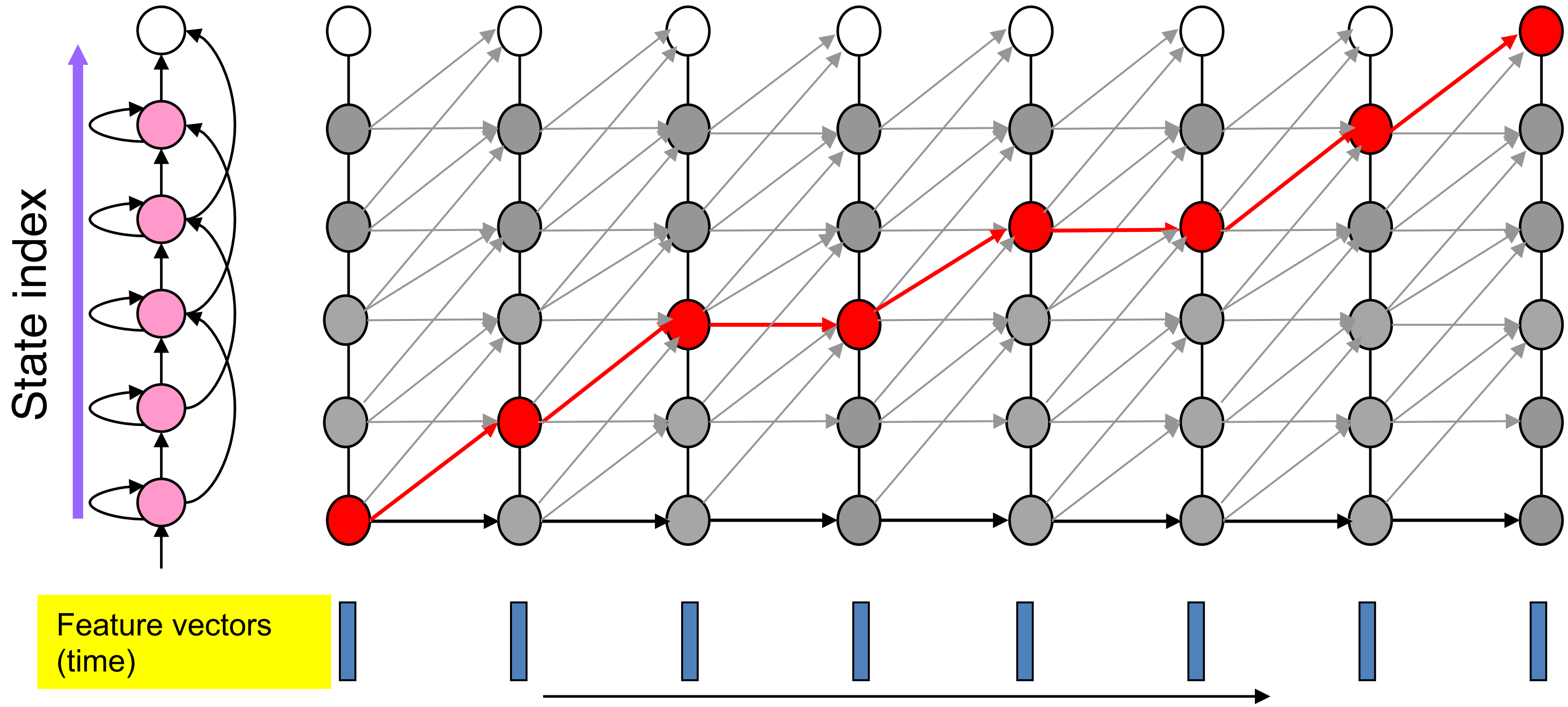
- The Y-axis represents HMM states, X axis represents observations
- Every node represents the event of a particular observation being generated from a particular state

The Trellis



- The trellis is a graphical representation of all possible paths through the HMM to produce a given observation
- Every edge in the graph represents a valid transition in the HMM over a single time step
  - Each edge carries the state transition probability between the source and destination states
- Every node represents the event of a particular observation being generated from a particular state
  - Each node for state s_t at time t carries the probability $P(O_t \mid s_t)$

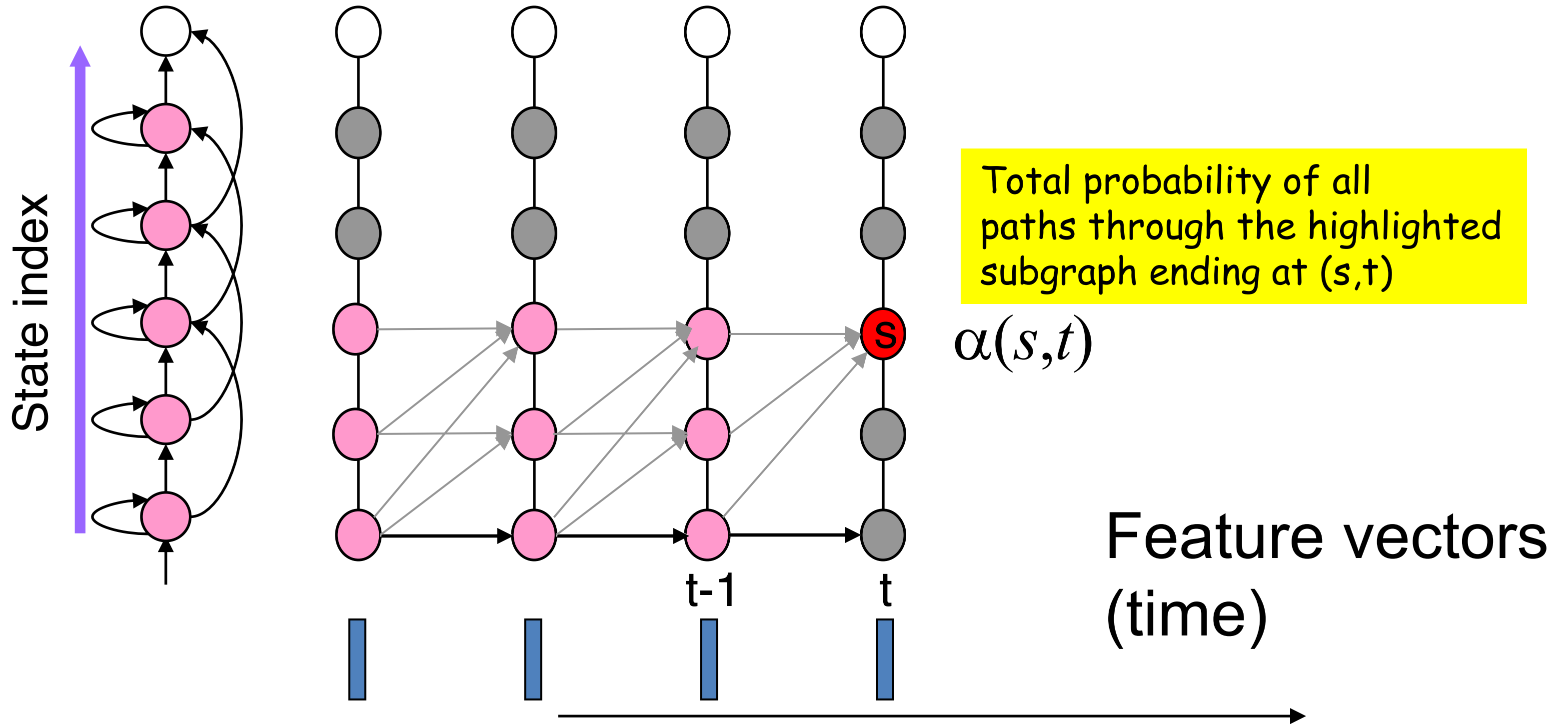
The Trellis



- Any path through the trellis is a sequence of states that the processes has traversed in generating the observations

- The probability of the path is the product of all the edge and node probabilities on the path
  - $P(s_0)P(O_0|s_0)\prod_{t=1} P(s_t|s_{t-1})P(O_t|s_t)$

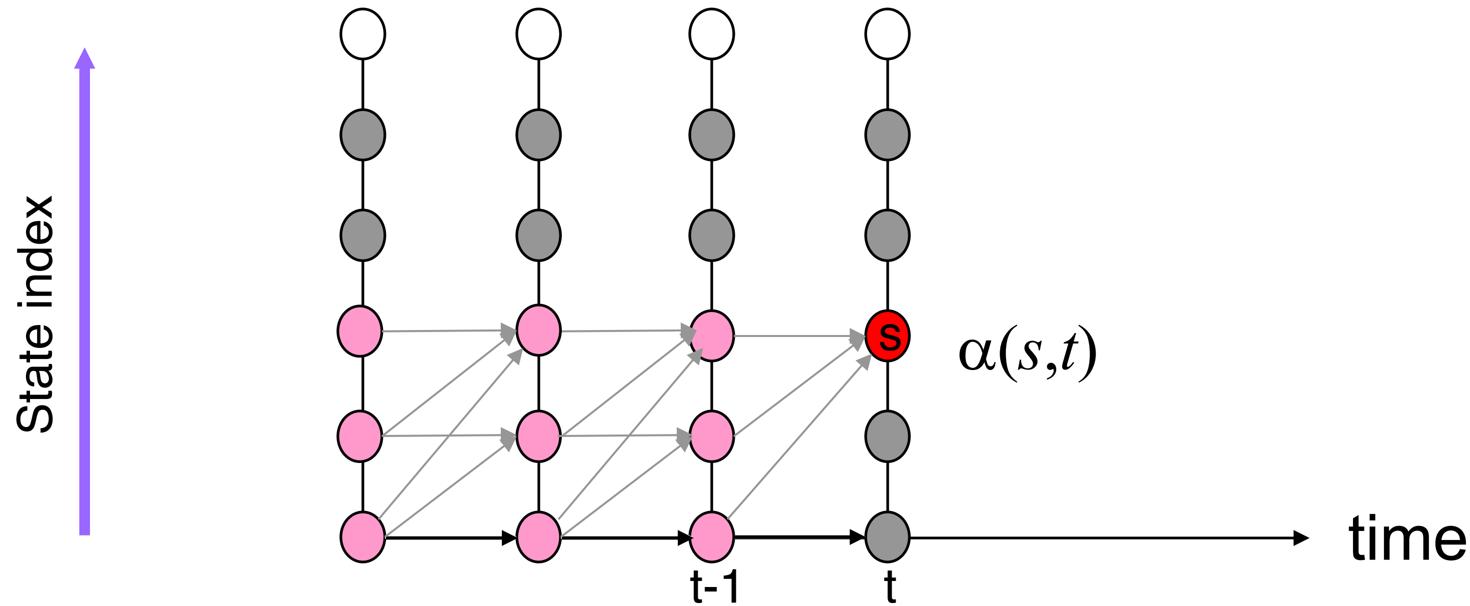
Diversion: The Trellis



- The trellis is a graphical representation of all possible paths through the HMM to produce a given observation
- The Y-axis represents HMM states, X axis represents observations
- Every edge in the graph represents a valid transition in the HMM over a single time step
- Every node represents the event of a particular observation being generated from a particular state

The Forward Algorithm

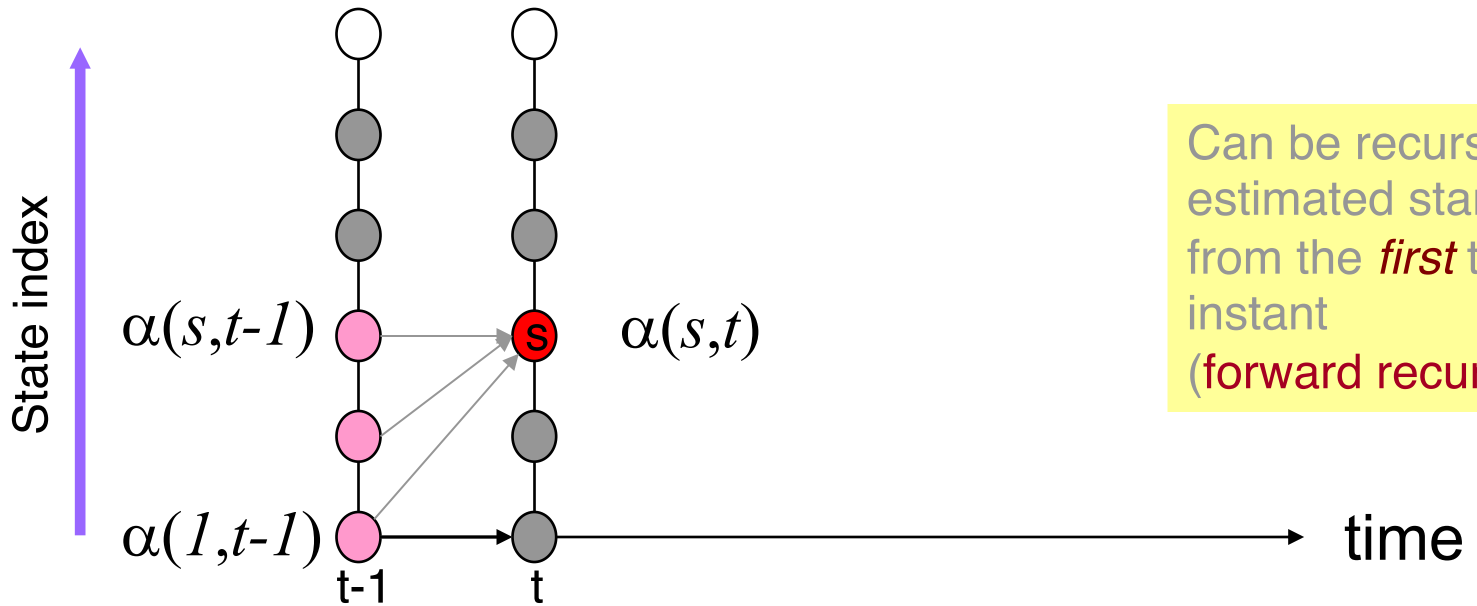
$$\alpha(s,t) = P(x_1, x_2, \ldots, x_t, state(t) = s)$$



- $\alpha(s,t)$ is the total probability of ALL state sequences that end at state *s* at time *t*, and all observations until x_t

The Forward Algorithm

$$\alpha(s,t) = P(x_1, x_2, \ldots, x_t, state(t) = s)$$



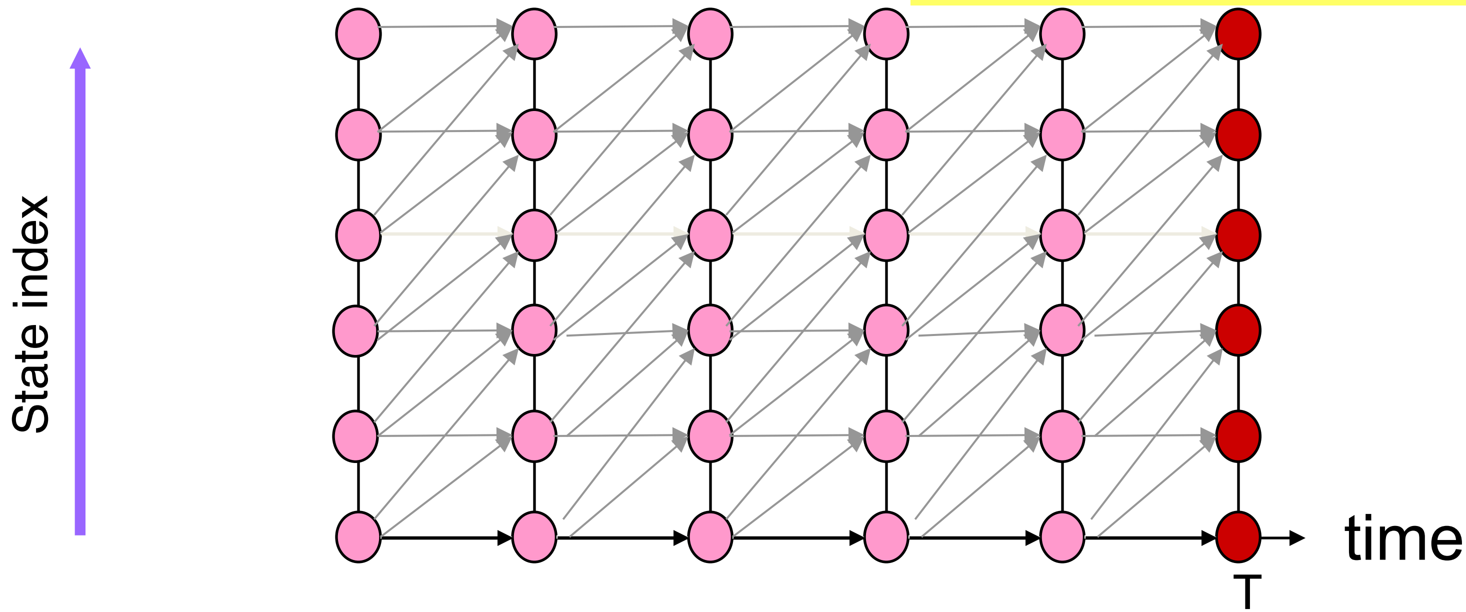
Can be recursively estimated starting from the *first* time instant (forward recursion)

$$\alpha(s,t) = \sum_{s'} \alpha(s', t-1) P(s \mid s') P(x_t \mid s)$$

- $\alpha(s,t)$ can be recursively computed in terms of $\alpha(s',t')$, the forward probabilities at time t-1

The Forward Algorithm

$$Totalprob = \sum_{s} \alpha(s,T)$$

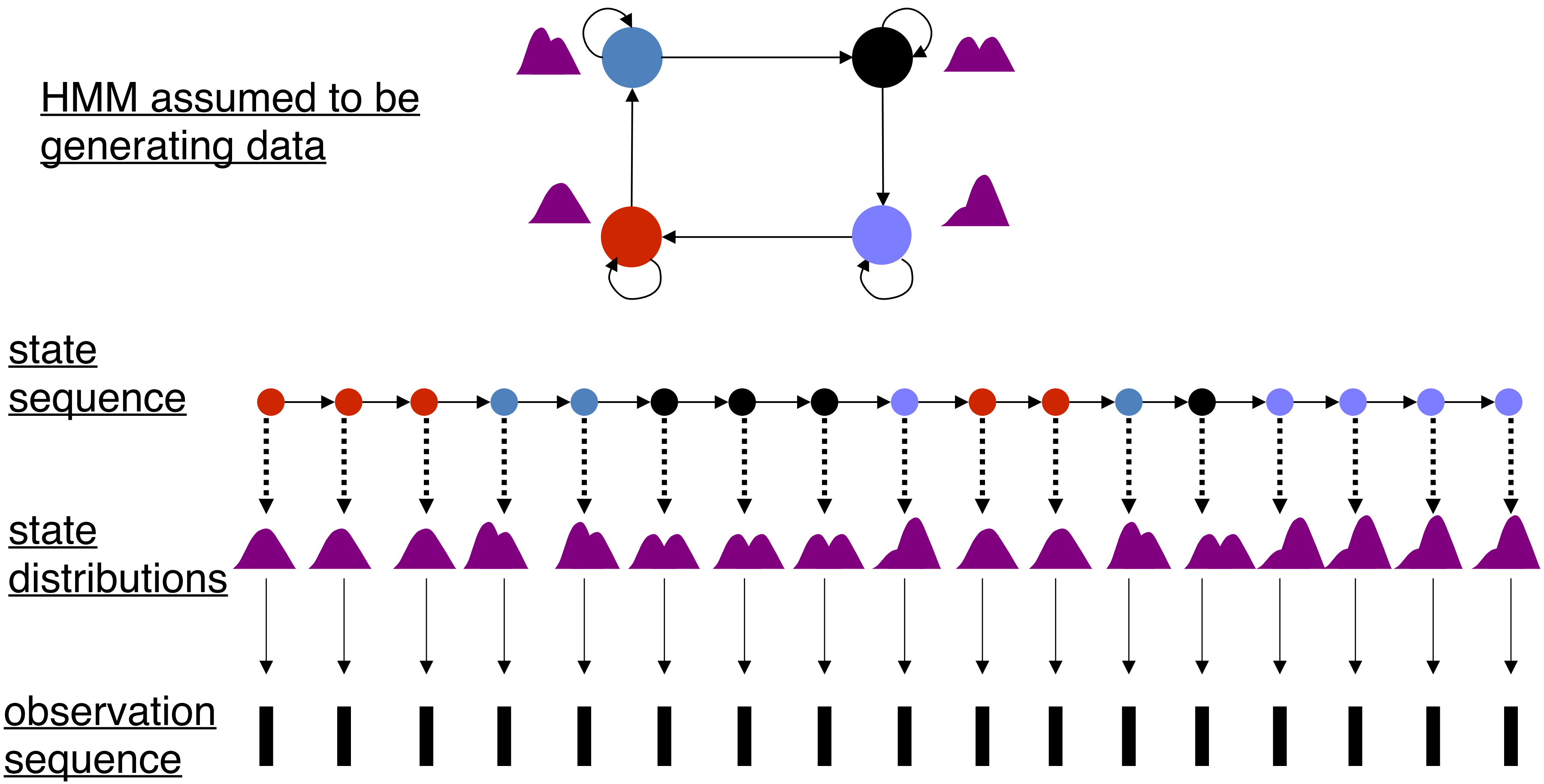


- In the final observation the alpha at each state gives the probability of all state sequences ending at that state
- General model: The total probability of the observation is the sum of the alpha values at all states

Problem 2: State segmentation

- Given only a sequence of observations, how do we determine which sequence of states was followed in producing it?

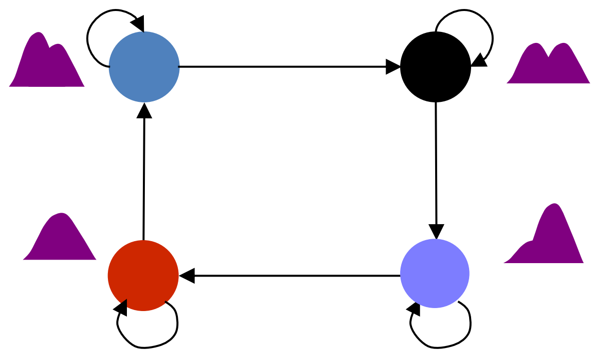
The HMM as a generator



- The process goes through a series of states and produces observations from them

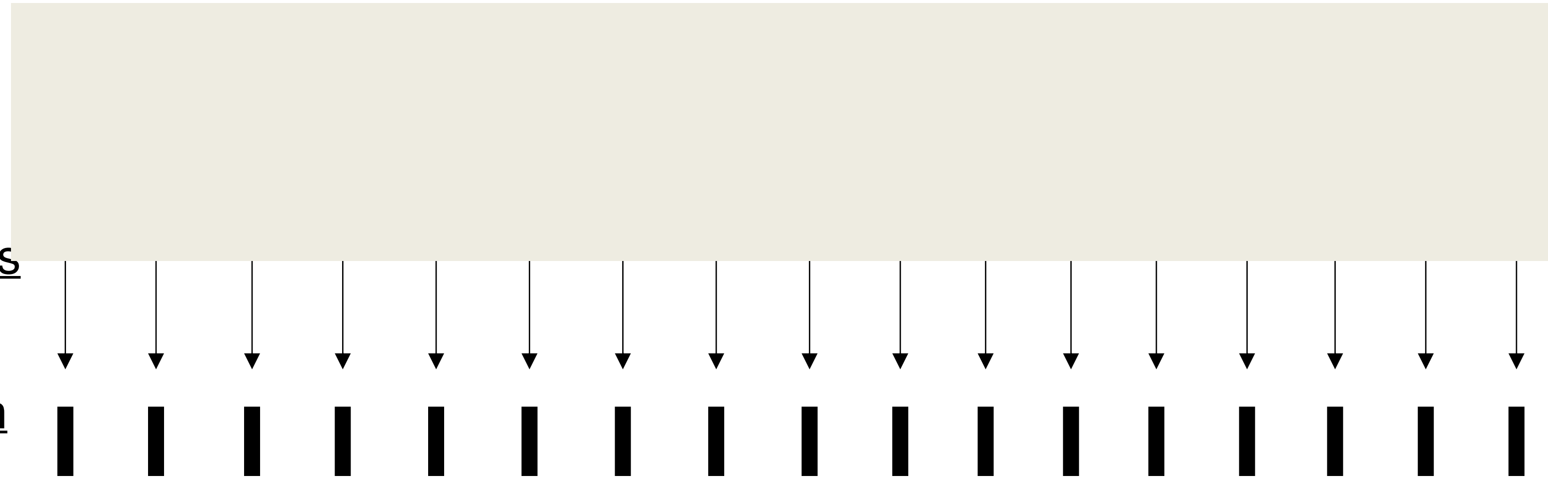
States are hidden

HMM assumed to be generating data



state sequence

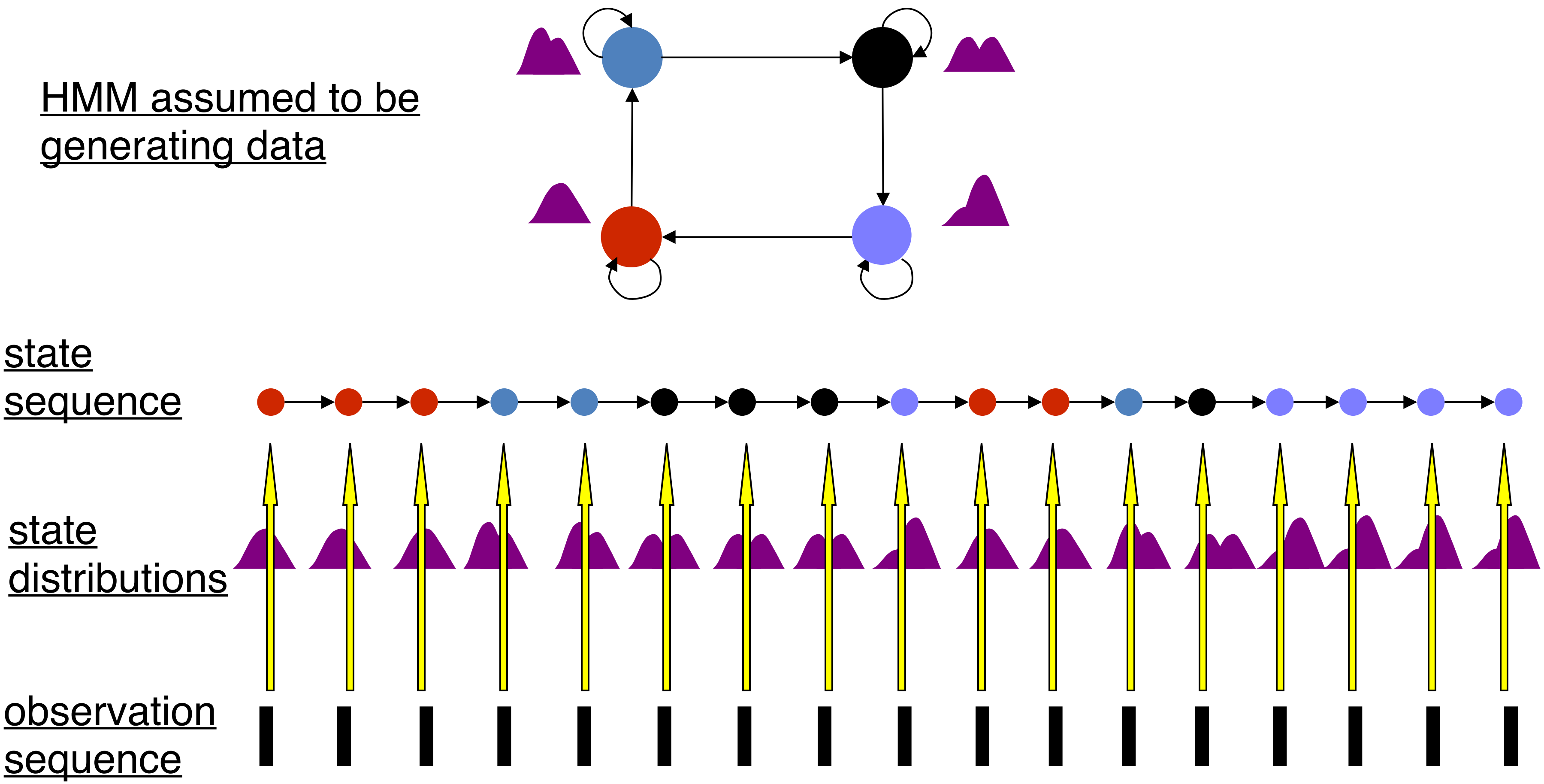
state distributions



observation sequence

- The observations do not reveal the underlying state

The state segmentation problem



- State segmentation: Estimate state sequence given observations

Estimating the State Sequence

- Many different state sequences are capable of producing the observation

- Solution: Identify the most *probable* state sequence
  - The state sequence for which the probability of progressing through that sequence and generating the observation sequence is maximum
  - i.e $P(o_1, o_2, o_3, \ldots, s_1, s_2, s_3, \ldots)$ is maximum

Estimating the state sequence

- Once again, exhaustive evaluation is impossibly expensive
- But once again a simple dynamic-programming solution is available

$$P(o_1, o_2, o_3, \ldots, s_1, s_2, s_3, \ldots) =$$

$$\underline{P(o_1|s_1)P(o_2|s_2)P(o_3|s_3)\ldots}\,\underline{P(s_1)P(s_2|s_1)P(s_3|s_2)\ldots}$$

- Needed:

$$\arg\max_{s_1, s_2, s_3, \ldots} P(o_1 \mid s_1)P(s_1)P(o_2 \mid s_2)P(s_2 \mid s_1)P(o_3 \mid s_3)P(s_3 \mid s_2)$$

Estimating the state sequence

- Once again, exhaustive evaluation is impossibly expensive
- But once again a simple dynamic-programming solution is available

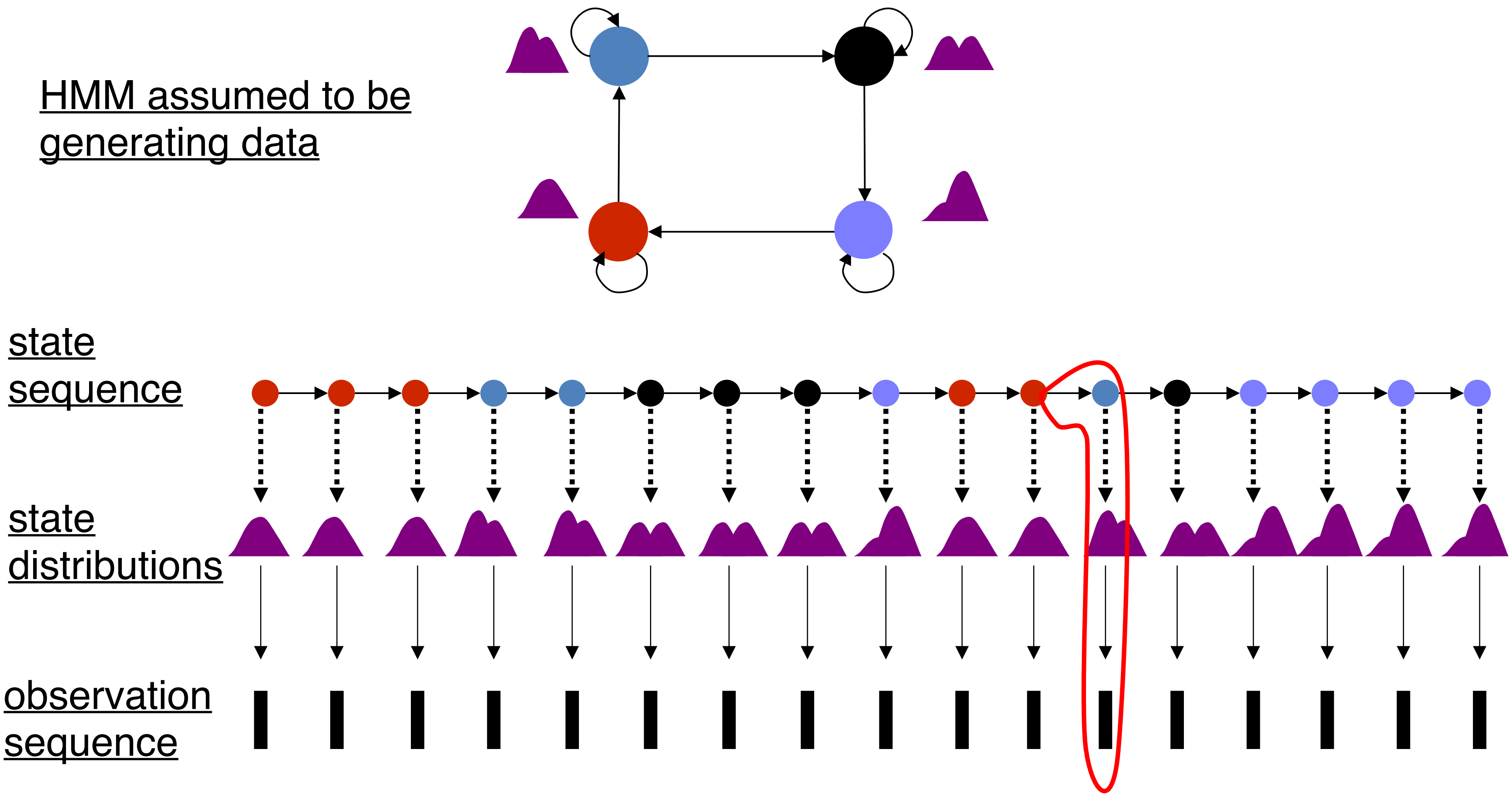
$$P(o_1, o_2, o_3, \ldots, s_1, s_2, s_3, \ldots) =$$

$$P(o_1|s_1)P(o_2|s_2)P(o_3|s_3)\ldots P(s_1)P(s_2|s_1)P(s_3|s_2)\ldots$$

- Needed:

$$\arg\max_{s_1, s_2, s_3, \ldots} P(o_1 \mid s_1)P(s_1)P(o_2 \mid s_2)P(s_2 \mid s_1)P(o_3 \mid s_3)P(s_3 \mid s_2)$$

The HMM as a generator



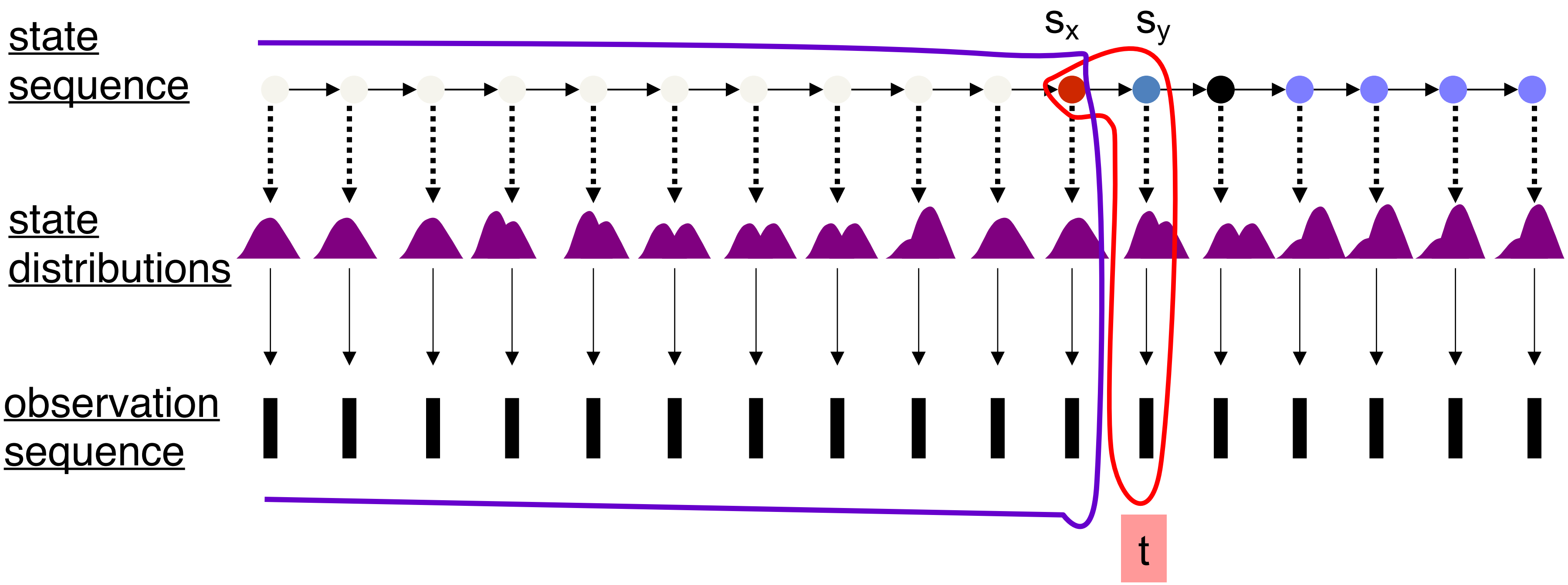
- Each enclosed term represents one forward transition and a subsequent emission

The state sequence

- The probability of a state sequence ?,?,?,?,s_x,s_y ending at time t, and producing all observations until o_t
  - $P(o_{1..t-1}, ?,?,?,?, s_x , o_t,s_y) = \underline{P(o_{1..t-1},?,?,?,?, s_x)}\, P(o_t|s_y)P(s_y|s_x)$

- The *best* state sequence that ends with s_x,s_y at t will have a probability equal to the probability of the best state sequence ending at t-1 at s_x times $P(o_t|s_y)P(s_y|s_x)$

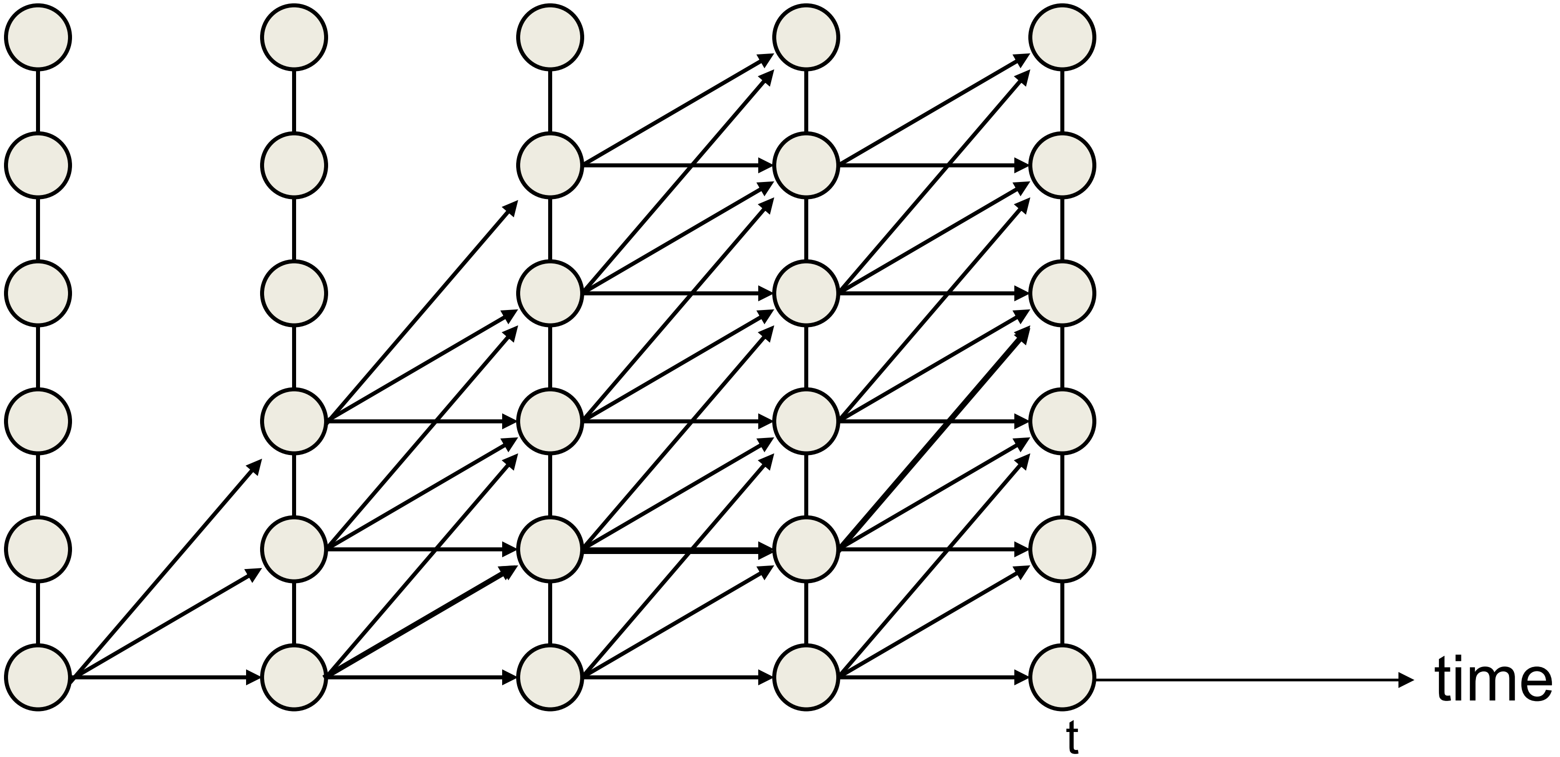
Extending the state sequence



- The probability of a state sequence ?,?,?,?,s_x,s_y ending at time *t* and producing observations until o_t
  - $P(o_{1..t-1}, o_t, ?,?,?,?, s_x, s_y) = P(o_{1..t-1}, ?,?,?,?, s_x)P(o_t|s_y)P(s_y|s_x)$

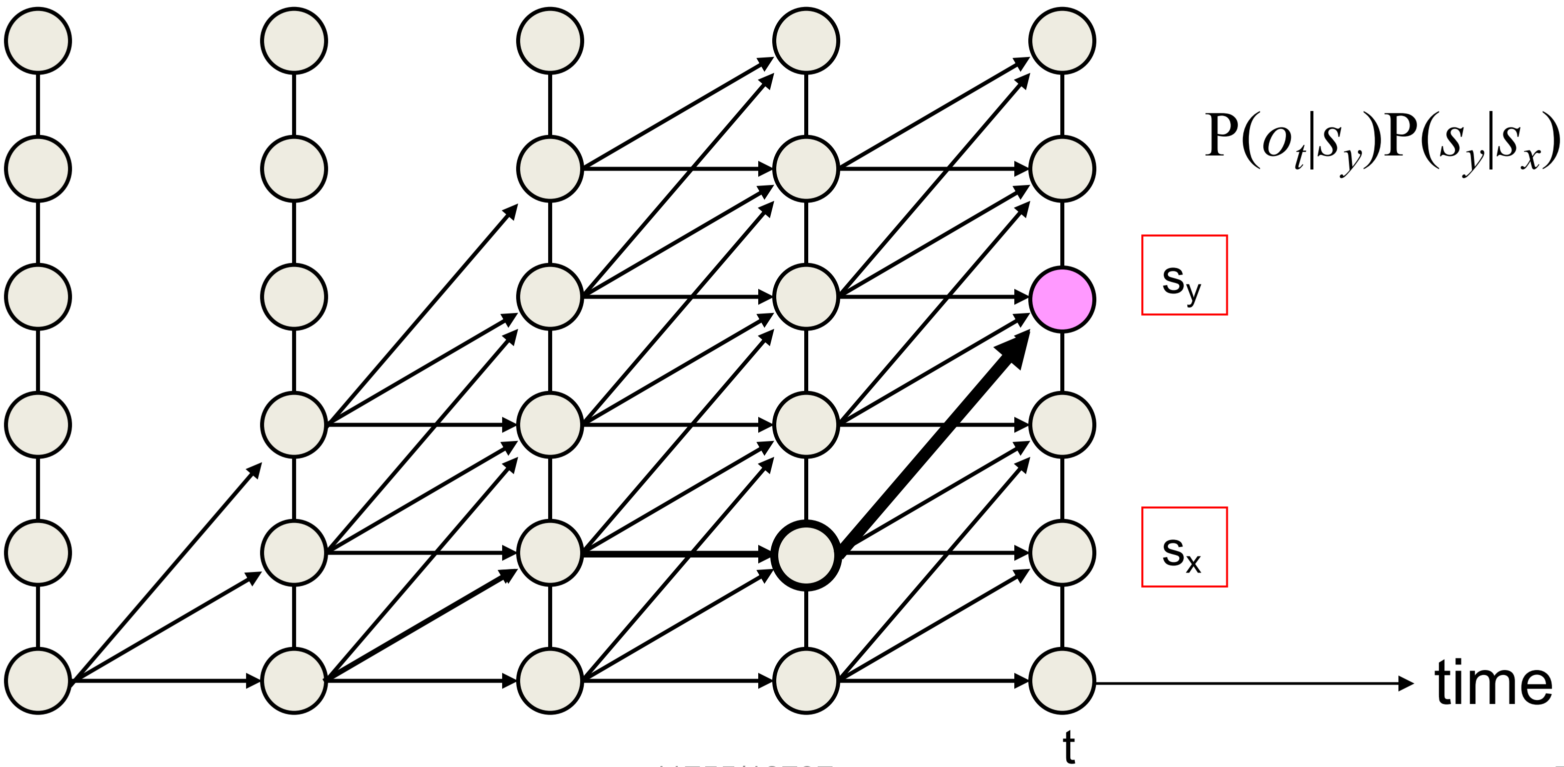
Trellis

- The graph below shows the set of all possible state sequences through this HMM in five time instants



The cost of extending a state sequence

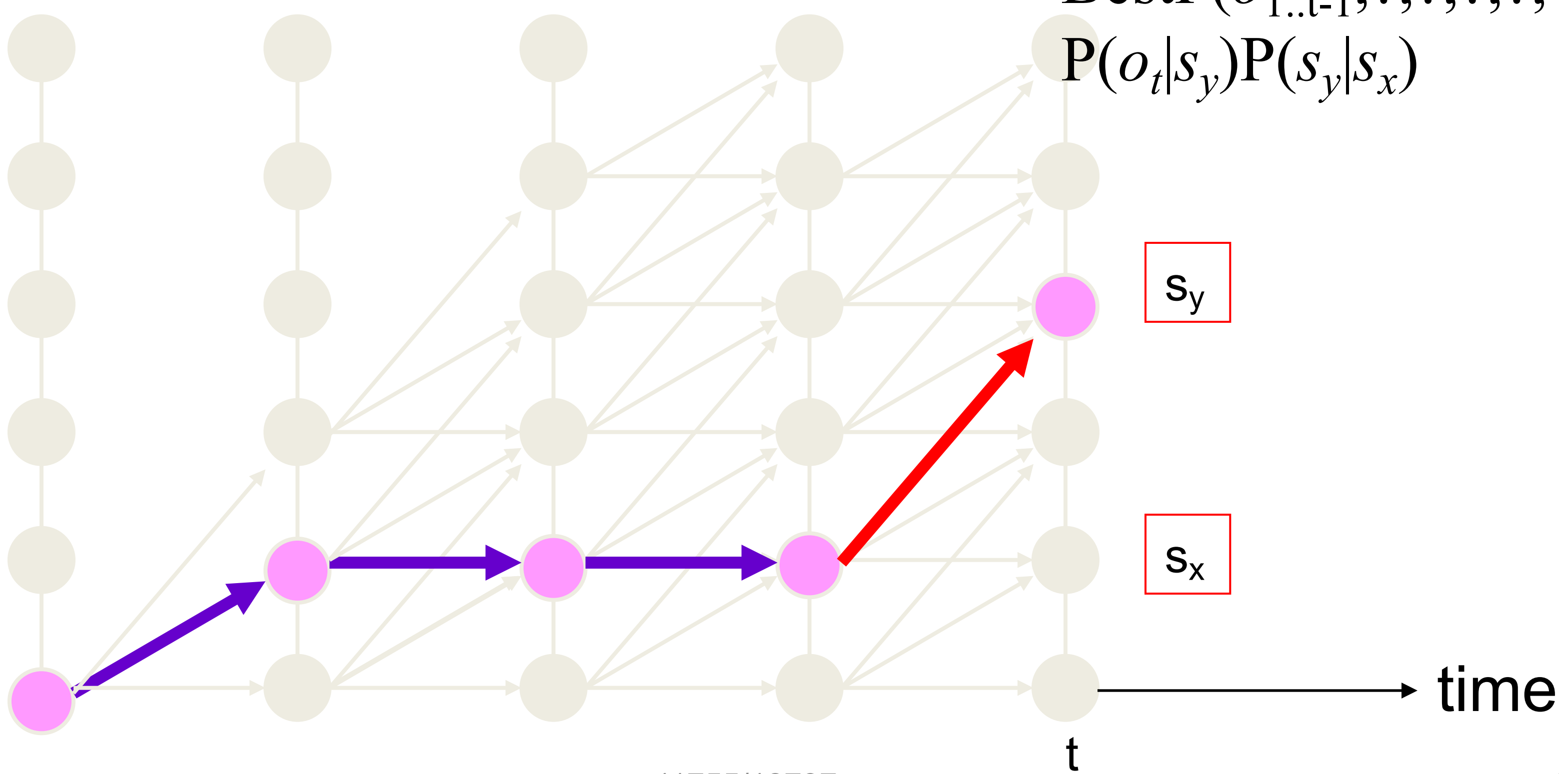
- The cost of *extending* a state sequence ending at s_x is only dependent on the transition from s_x to s_y, and the observation probability at s_y



The cost of extending a state sequence

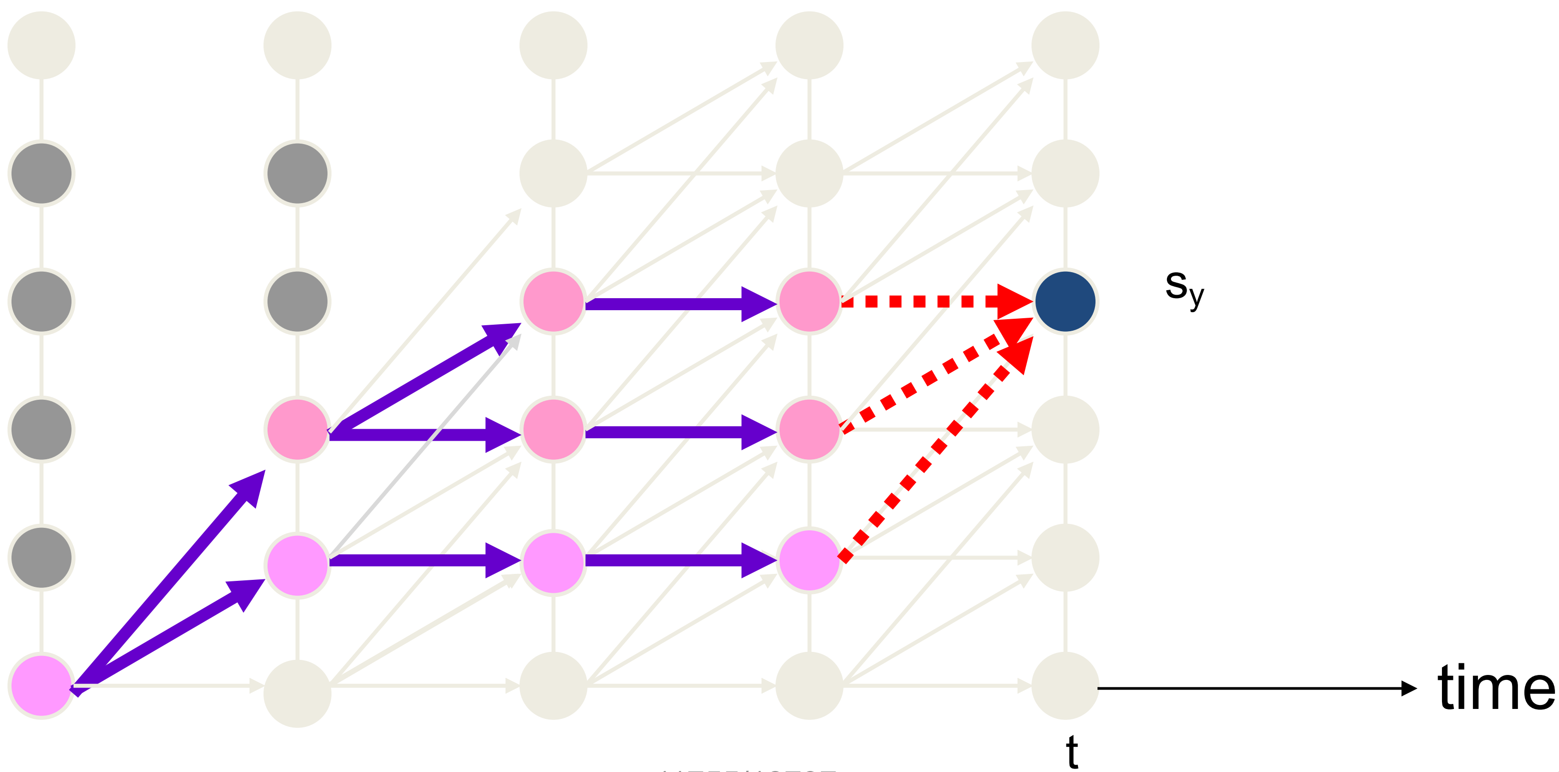
- The best path to s_y through s_x is simply an extension of the best path to s_x

$$\text{BestP}(o_{1..t-1},?,?,?,?,\ s_x)\ P(o_t|s_y)P(s_y|s_x)$$



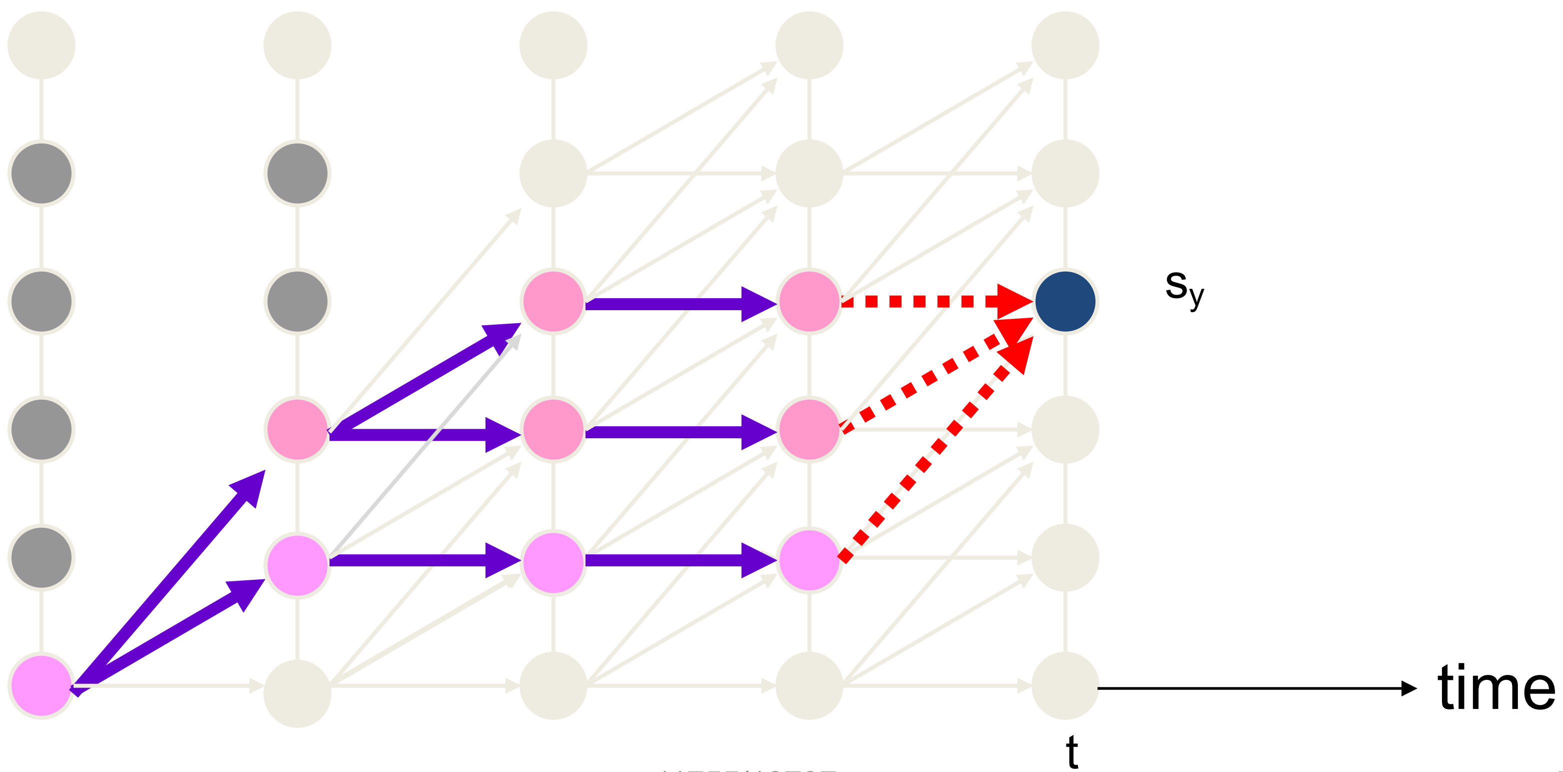
The Recursion

- The overall best path to s_y is an extension of the best path to one of the states at the previous time



The Recursion

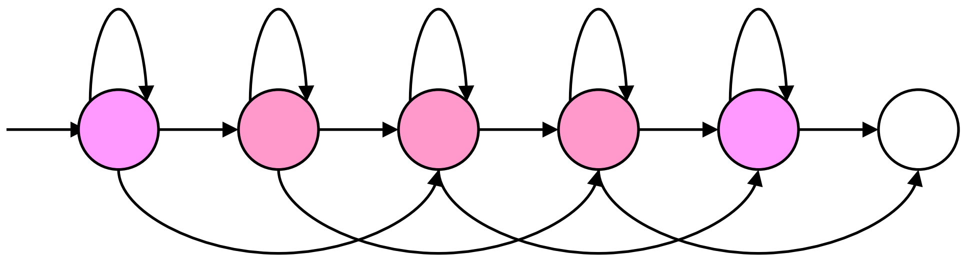
- Prob. of best path to s_y =
  Max$_{s_x}$ BestP($o_{1..t-1}$,?,?,?,?, s_x) $P(o_t|s_y)P(s_y|s_x)$

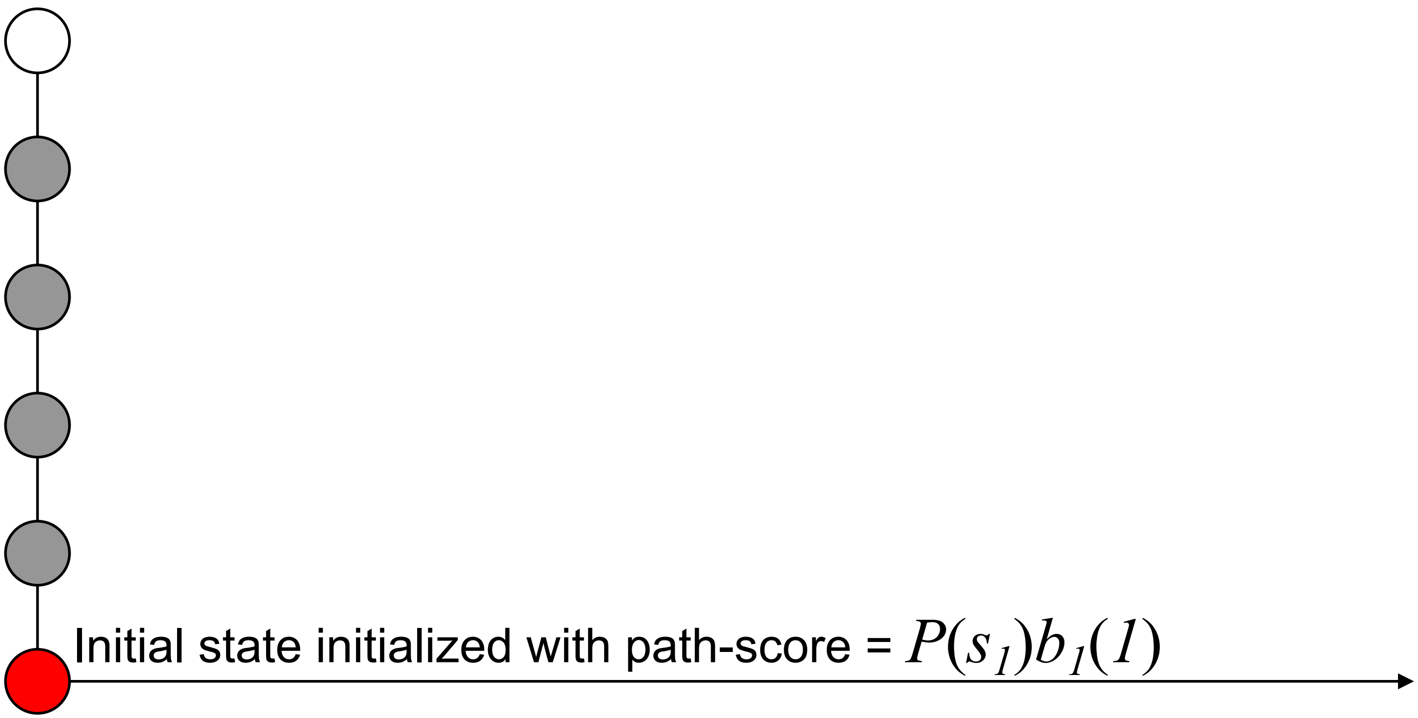


Finding the best state sequence

- The simple algorithm just presented is called the VITERBI algorithm in the literature
  - After A.J.Viterbi, who invented this dynamic programming algorithm for a completely different purpose: decoding error correction codes!

Viterbi Search (contd.)

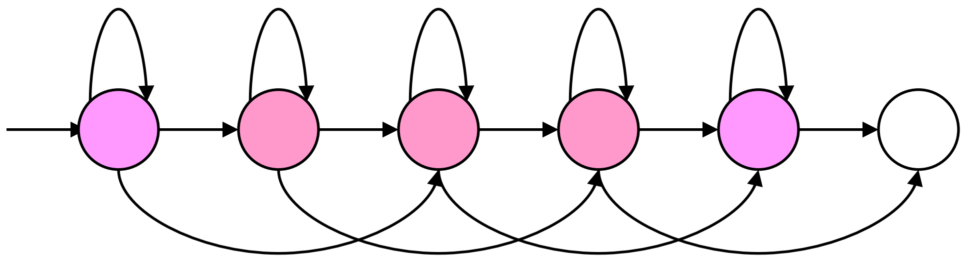


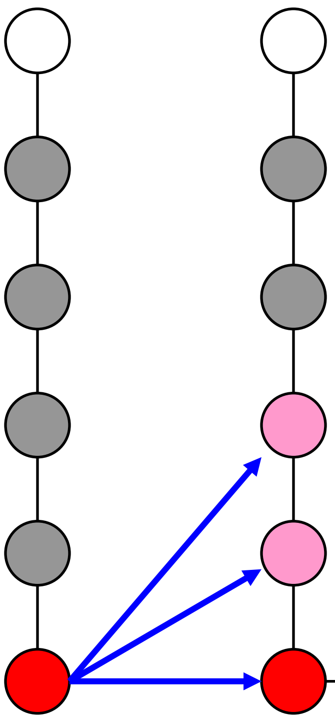


Initial state initialized with path-score = $P(s_1)b_1(1)$

In this example all other states have score 0 since $P(s_i) = 0$ for them

Viterbi Search (contd.)





- State with best path-score
- State with path-score < best
- State without a valid path-score

$$P_j(t) = \max_i \left[P_i(t\text{-}1)\ t_{ij}\ b_j(t)\right]$$

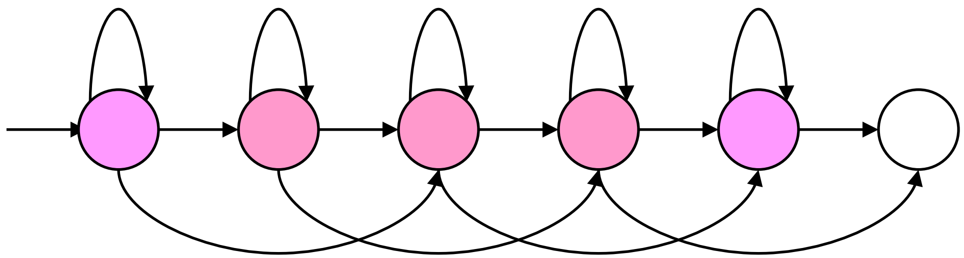
State transition probability, i to j

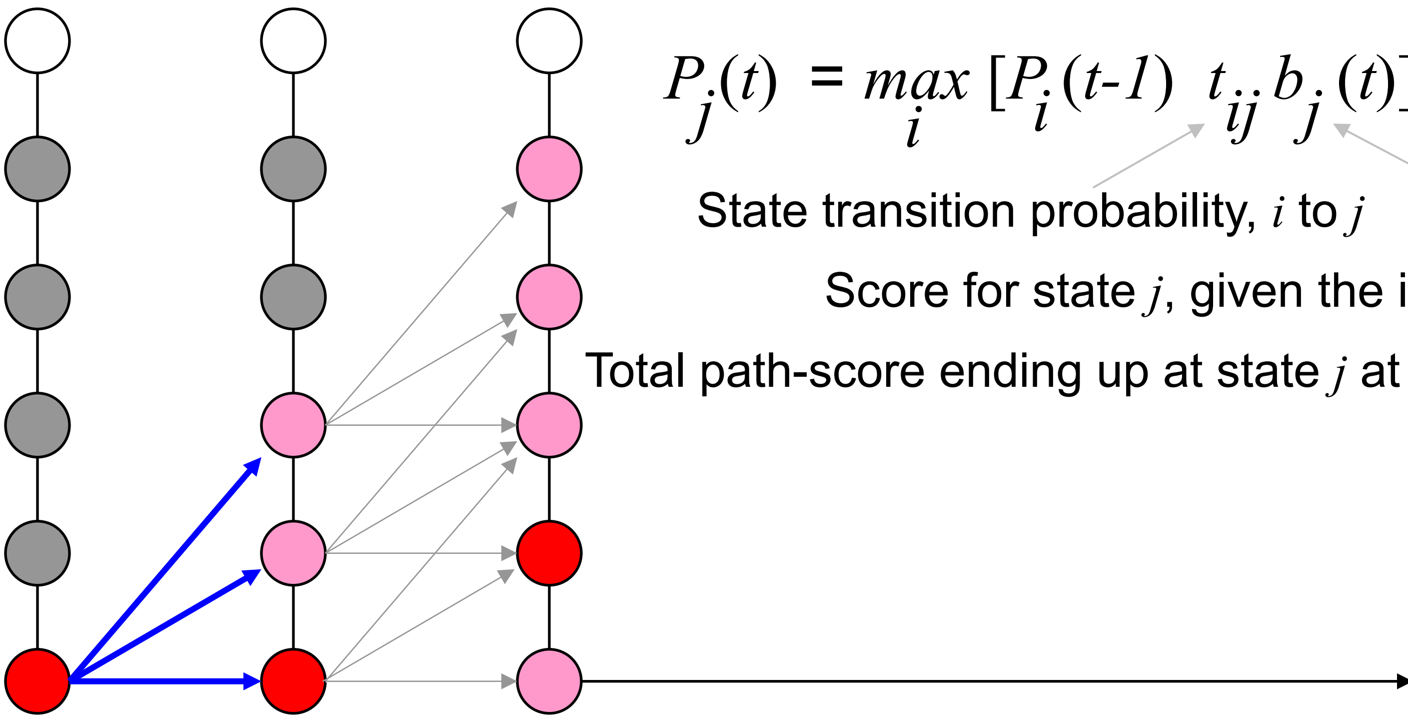
Score for state j, given the input at time t

Total path-score ending up at state j at time t

time

Viterbi Search (contd.)





$$P_j(t) = \max_i \, [P_i(t-1) \;\; t_{ij} \, b_j(t)]$$

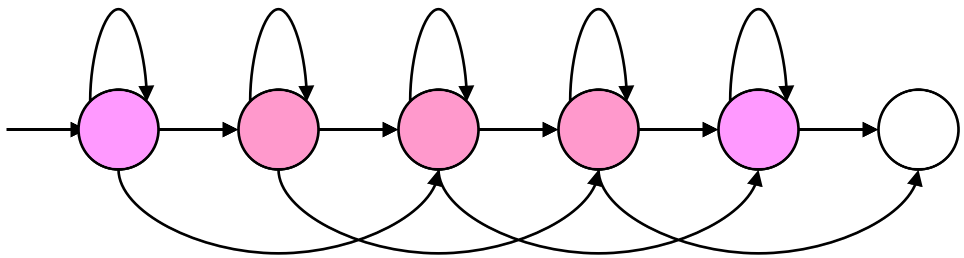
State transition probability, i to j

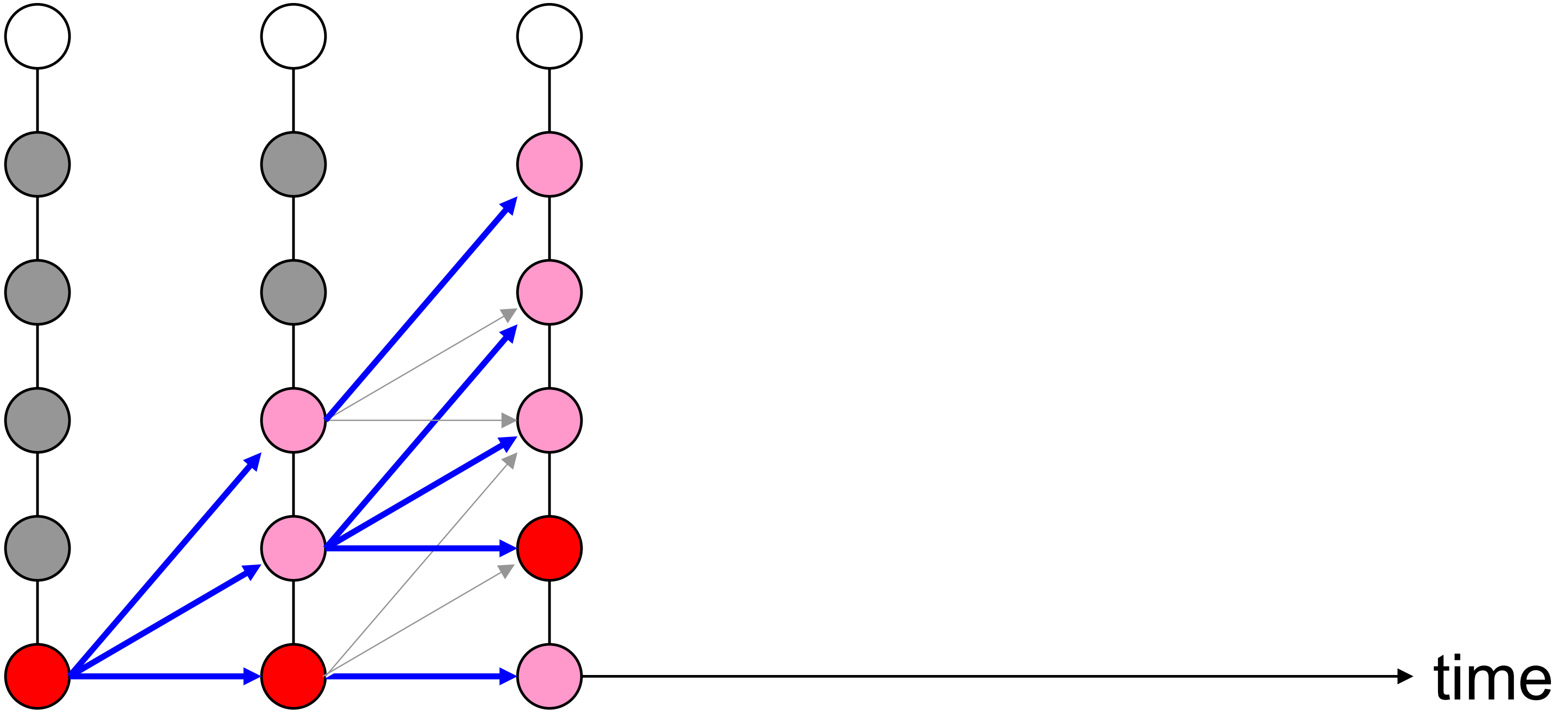
Score for state j, given the input at time t

Total path-score ending up at state j at time t

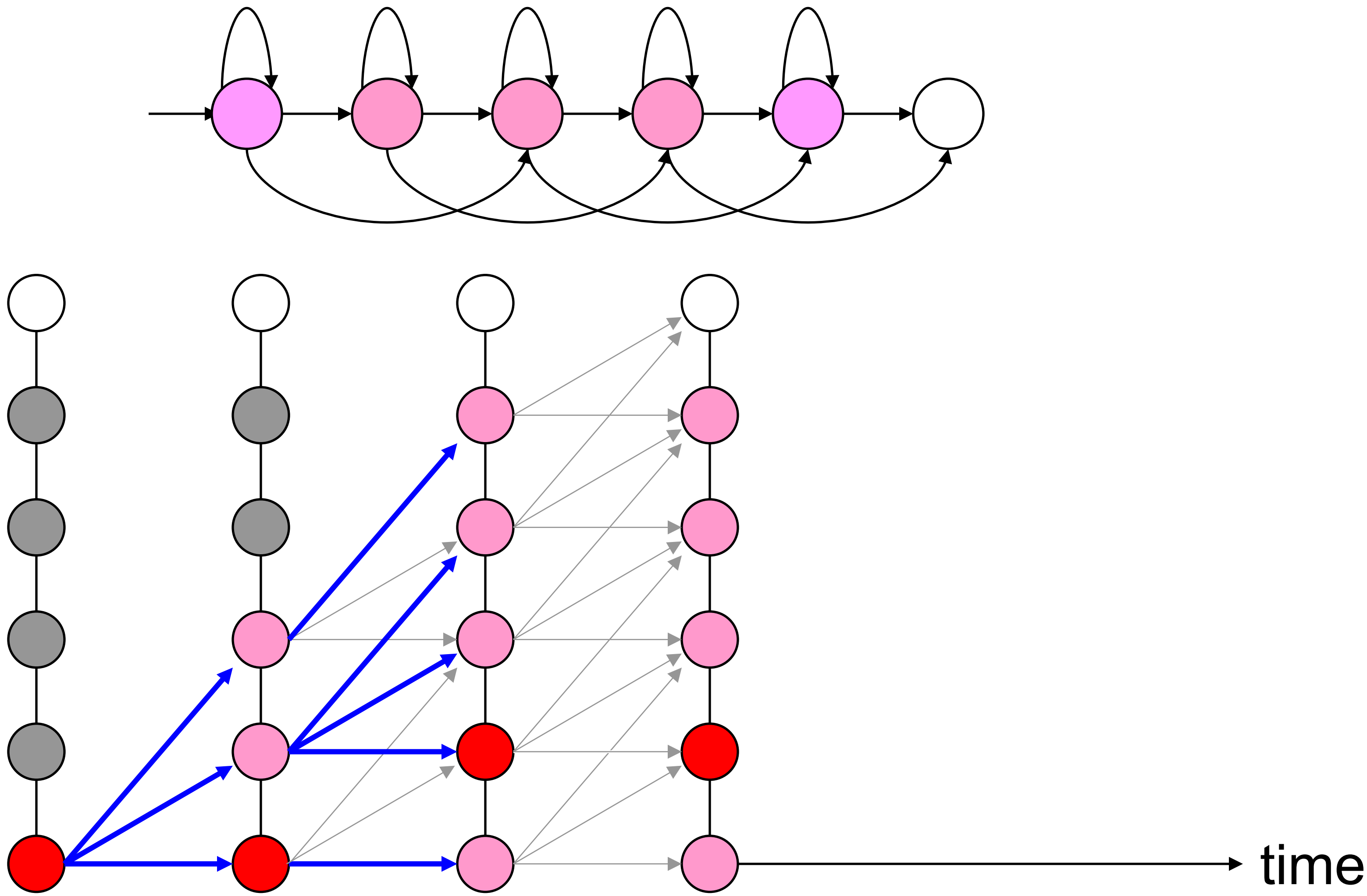
time

Viterbi Search (contd.)

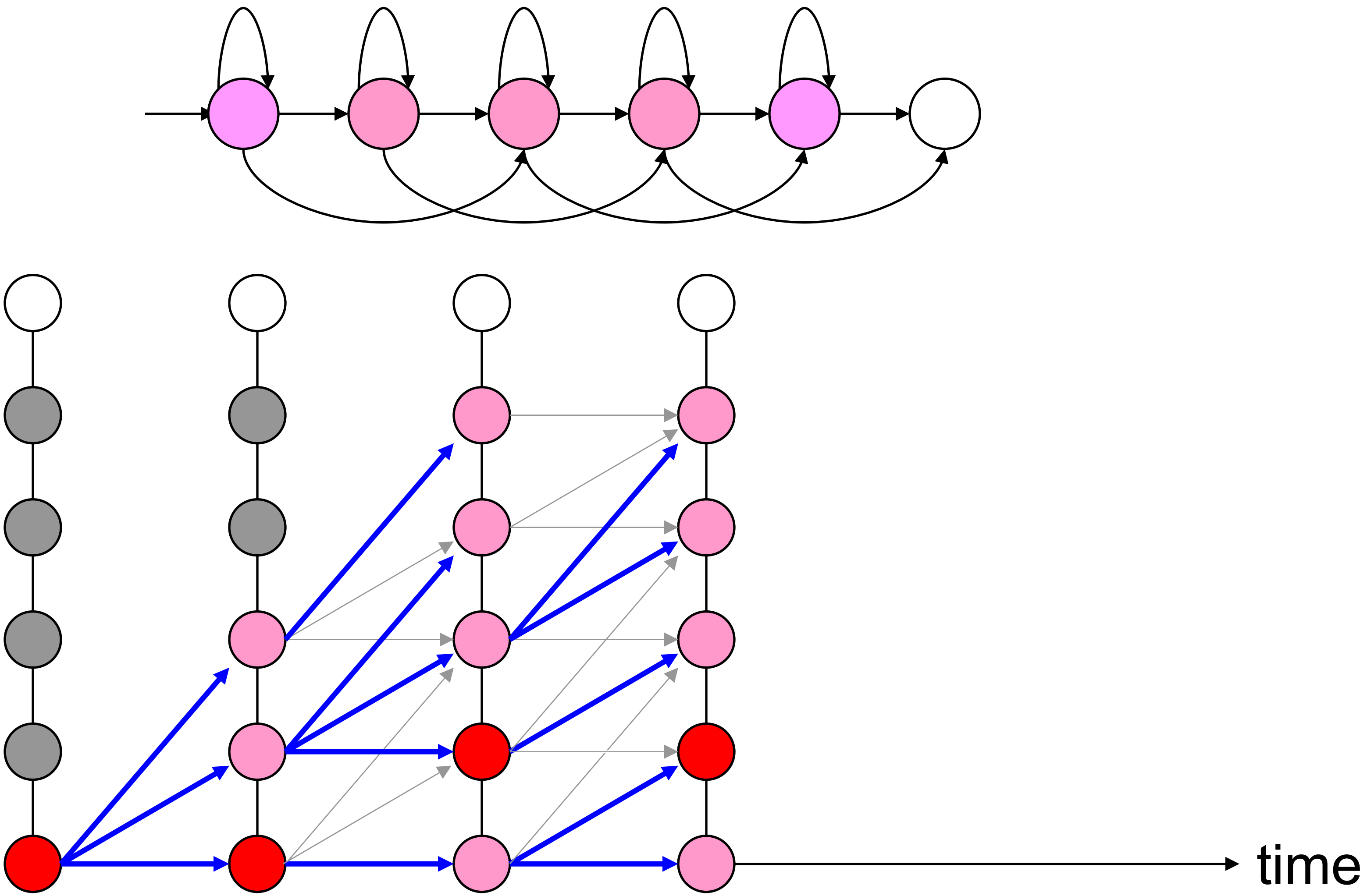




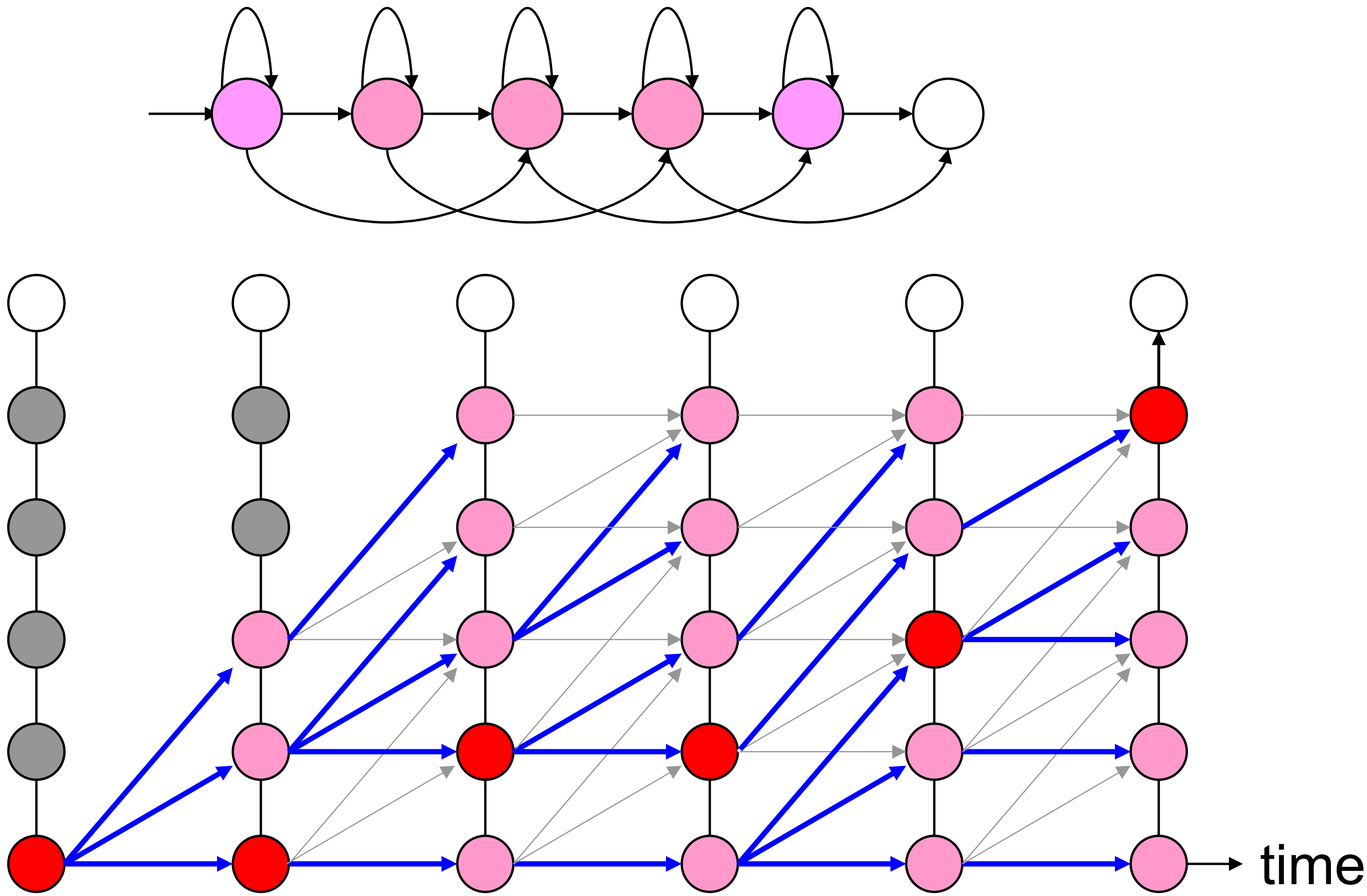
Viterbi Search (contd.)



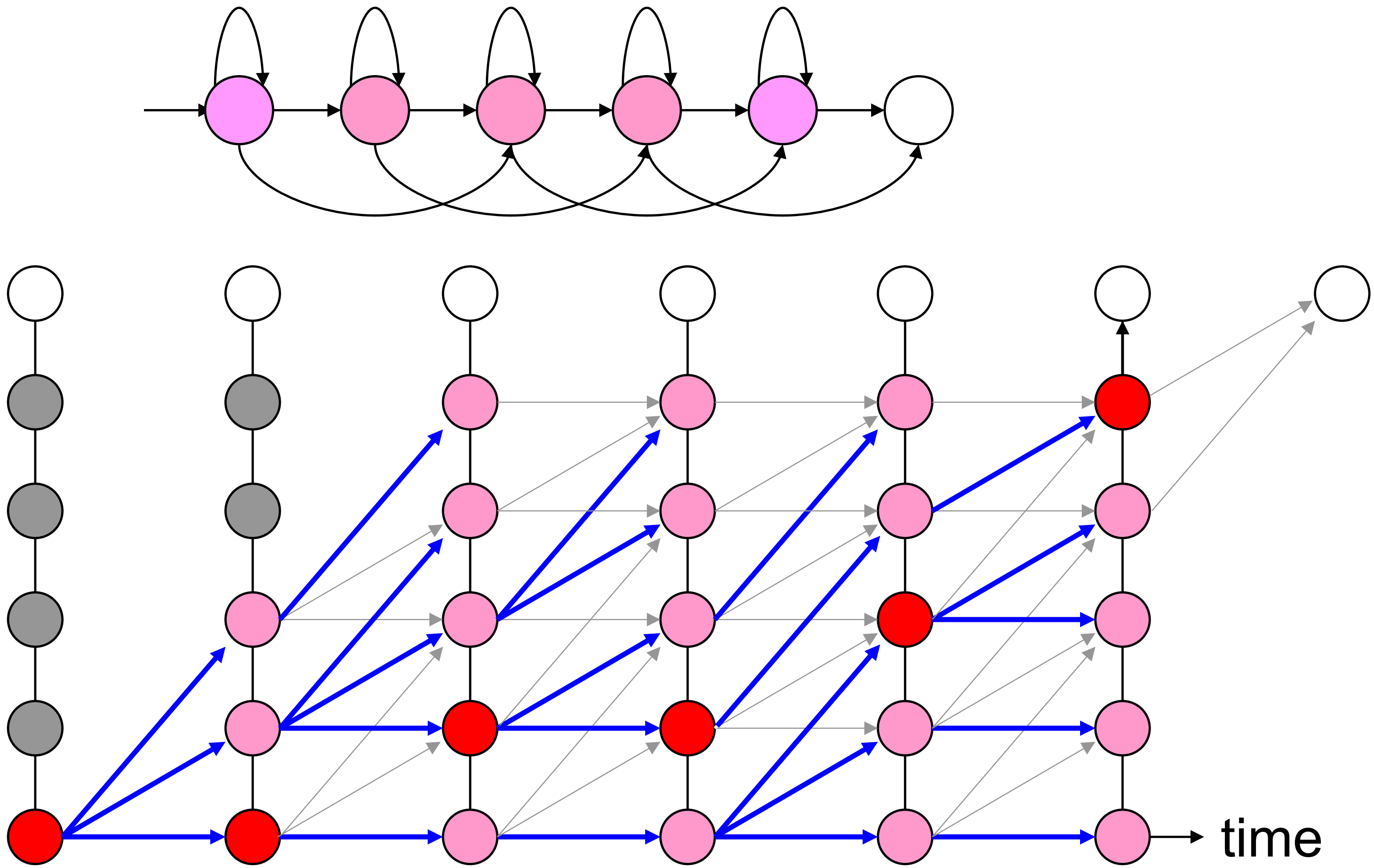
Viterbi Search (contd.)



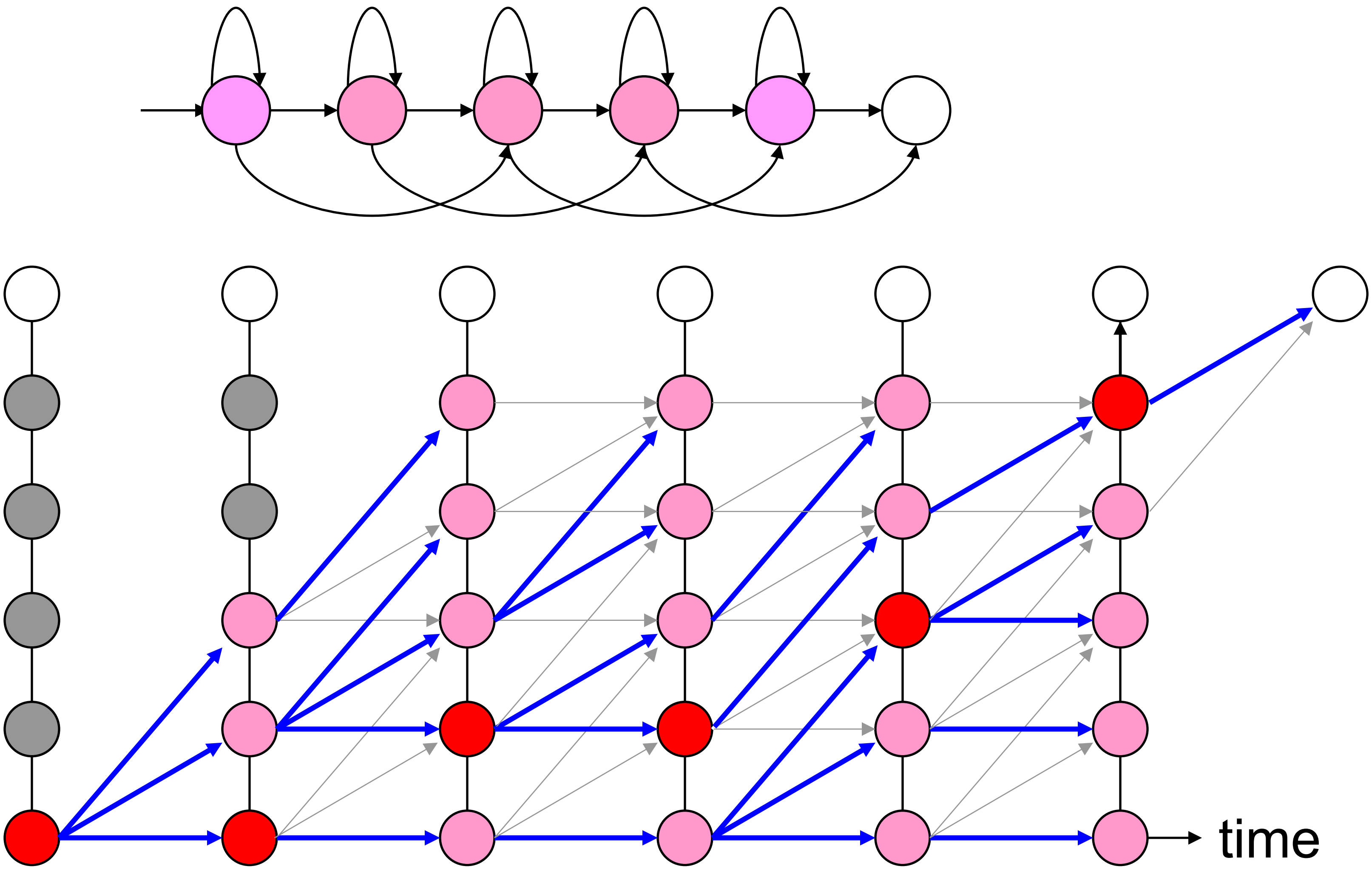
Viterbi Search (contd.)



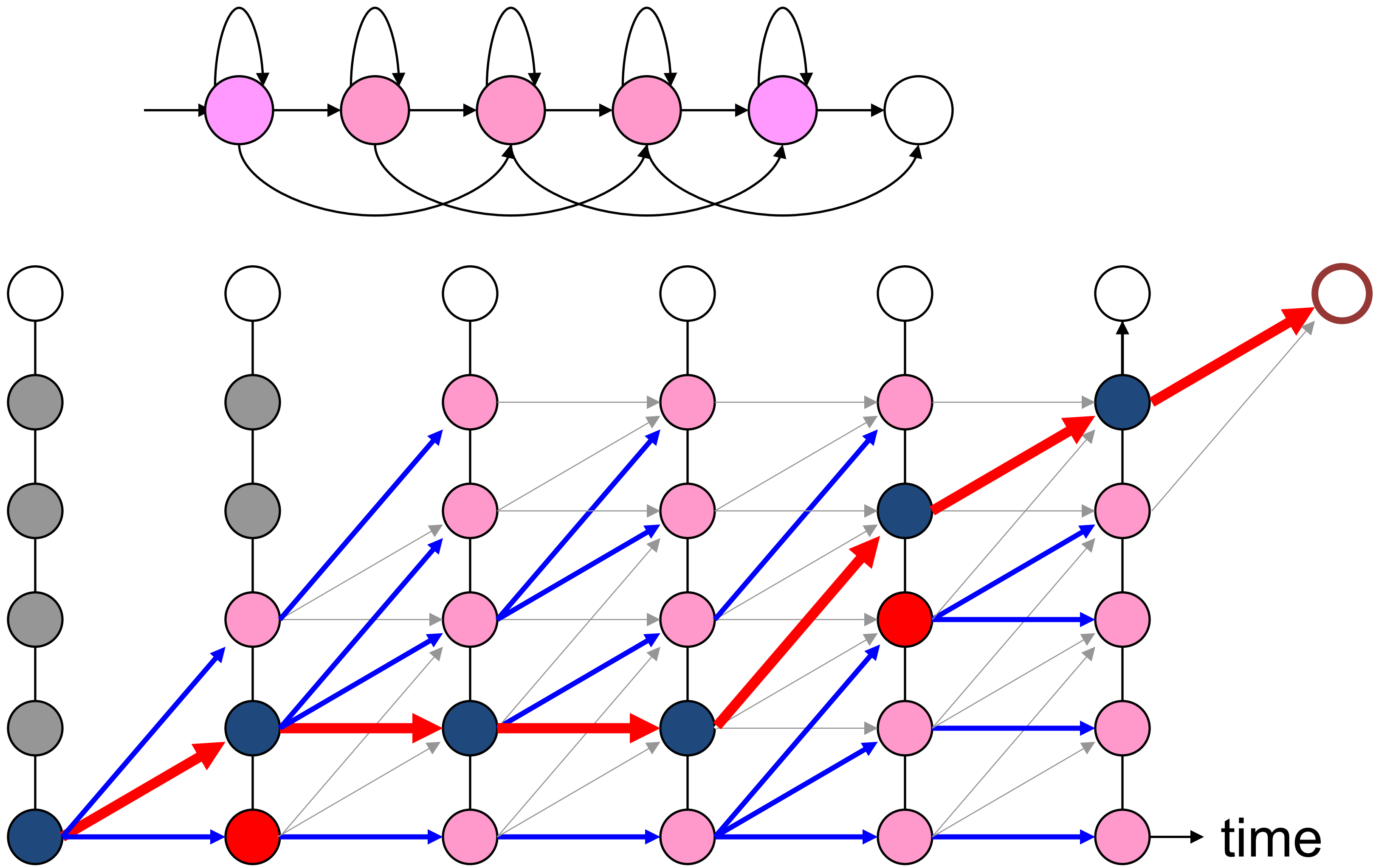
Viterbi Search (contd.)



Viterbi Search (contd.)

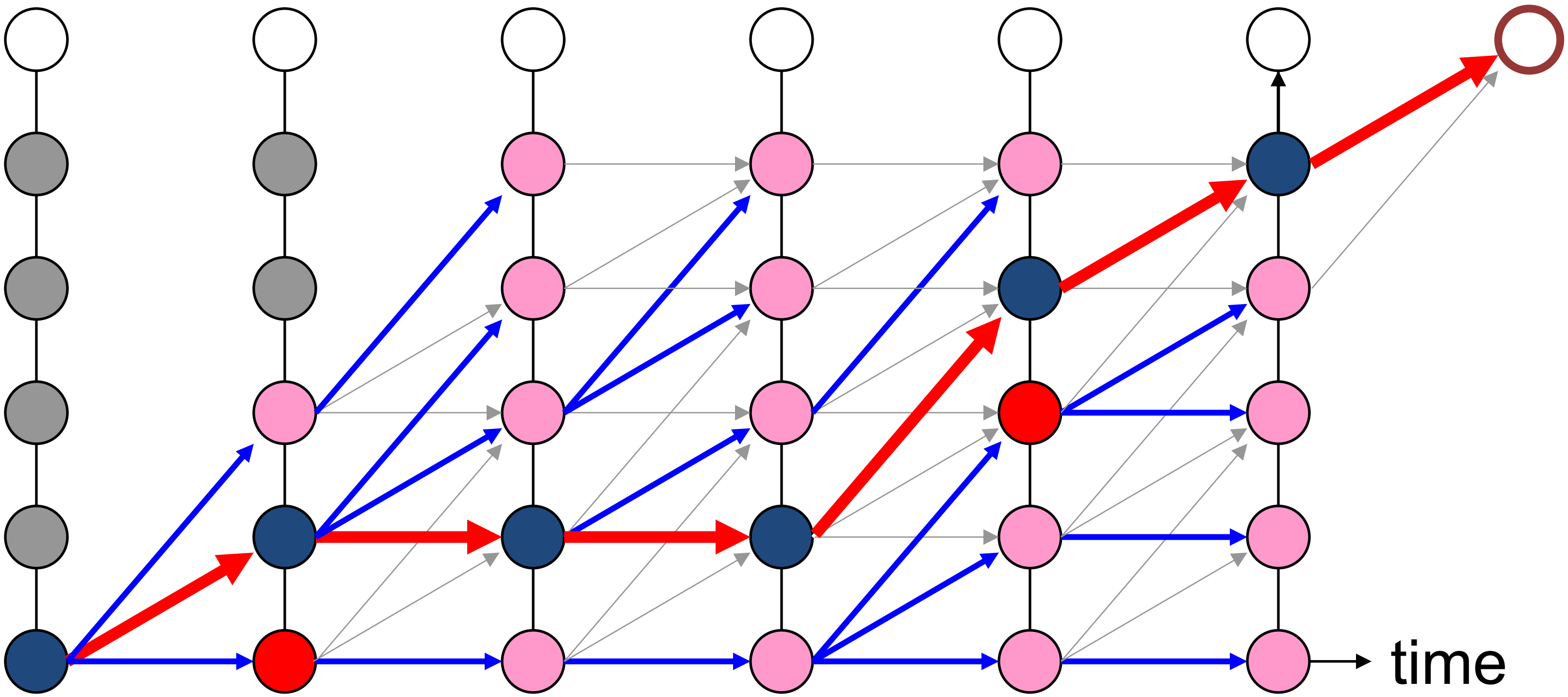


Viterbi Search (contd.)



Viterbi Search (contd.)

THE BEST STATE SEQUENCE IS THE ESTIMATE OF THE STATE SEQUENCE FOLLOWED IN GENERATING THE OBSERVATION



Problem3: Training HMM parameters

- We can compute the probability of an observation, and the best state sequence given an observation, using the HMM's parameters

- But where do the HMM parameters come from?

- They must be learned from a collection of observation sequences