HIDDEN MARKOV MODEL

Shingchern D. You

References

- https://web.stanford.edu/~jurafsky/slp3/9.pdf
- http://www.robots.ox.ac.uk/~vgg/rg/papers/hmm.pdf

Markov chain

- What is a Markov chain
- A Markov chain is "a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event“ – Wikipedia
- Think of a gambler: the possessed money can be viewed as a “state”
- The state changes over time

Markov chain

- Some notations
  - States: $S_1, \ldots, S_N$
  - In each time instance ($t = 1,2, \ldots, T$) a system changes to state q_t
- It is confusing about S_i and q_t
- Simple example
  - $S_1 = \text{hot}, S_2 = \text{cold}$ (N=2)
  - $q_t \in \{\text{hot}, \text{cold}\}$, $T = 100$

Markov chain

- For a special case of a **first-order Markov chain**, we have
$$P\left(q_t = S_j \middle| q_{t-1} = S_k, q_{t-2} = S_m, q_{t-2} = S_n, \ldots\right)$$
$$= P\left(q_t = S_j \middle| q_{t-1} = S_k\right)$$
- Probability from S_k to S_j is called state transition probability
- We further assume constant state transition probabilities (time independent)

Markov chain

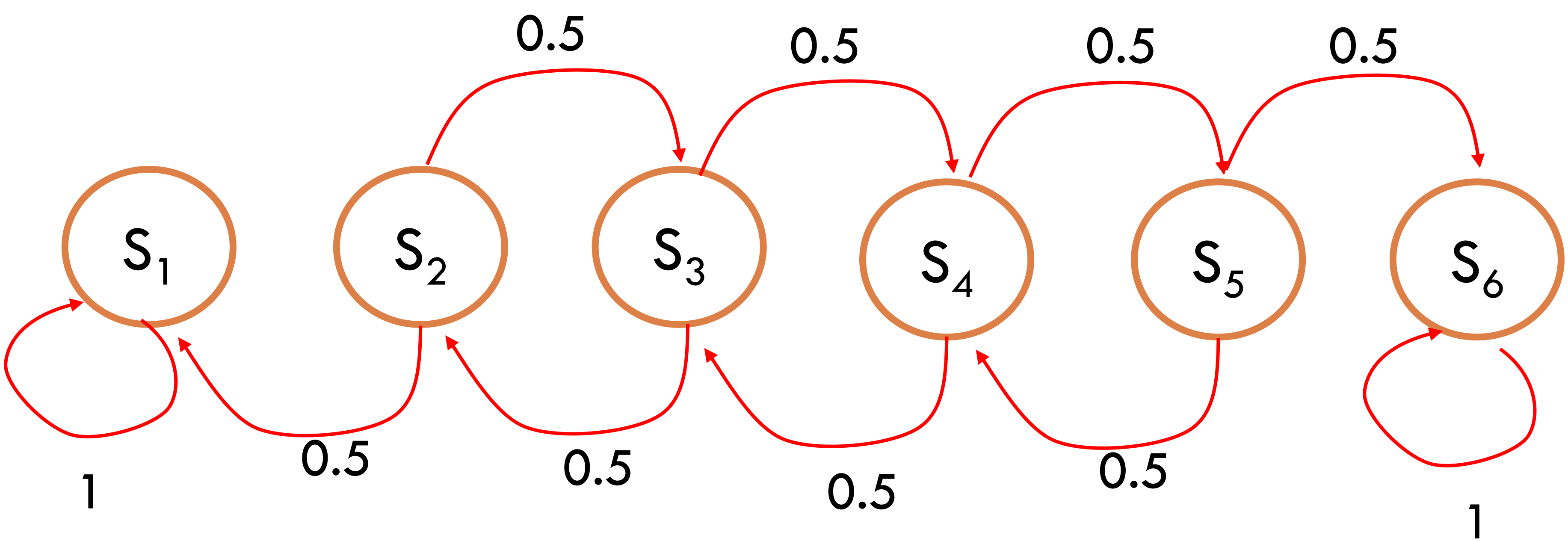
- We use gambler's ruin problem as an example
  - A gambler has 3 dollars before playing
  - Play a fair game of coin flipping : Probability (win) = Probability (lose) = 0.5
  - Every bet is one dollar
  - Stop playing if he has 5 dollars in hand or if he has no money to play

Markov chain

- States = amount of money gambler has in hand
- How many states do we have
  - \$ = 0, 1, …, 5 (6 states)
- It is a first-order Markov chain because next state depends only on present state (but not the past history)
- Transition probability is time independent (fixed for every bet)

Markov chain

- We can draw a simple graph to illustrate the Markov chain ($S_j = j - 1$ dollars in hand)



Markov chain

- From the figure we know $P(S_2|S_4) = 0$, but $P(S_5|S_4) = 0.5$
- Again, we confirm that the past history (how S_4 is arrived) does not affect the above conditional probability
- Exercise: Compute the probability the gambler will eventually broke (initially starting from S_4)

Markov chain

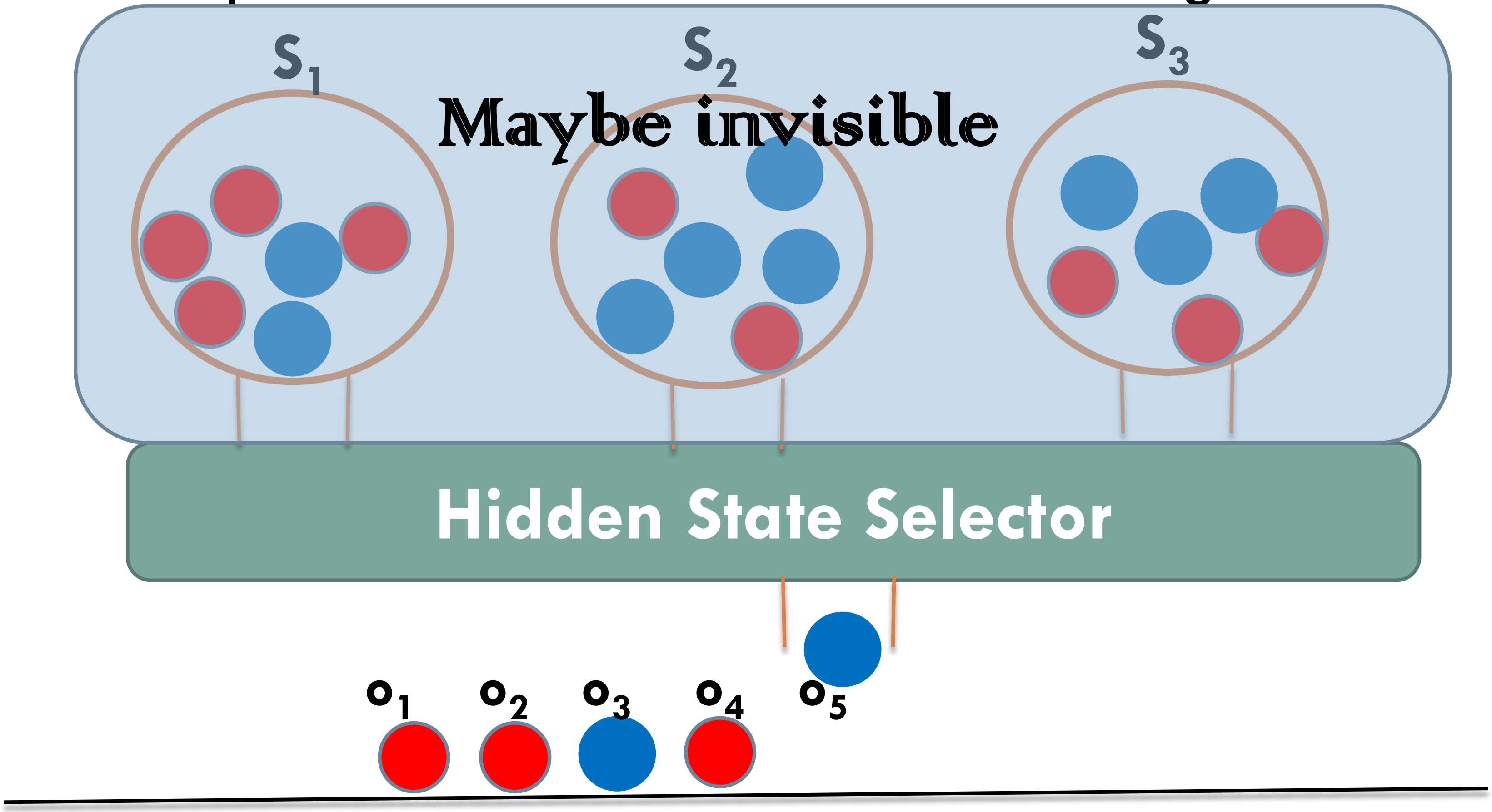
- FYI (For Your Information): If A has *a* dollars and B has *b* dollars to play the coin-tossing game till one side is broke, then $P(\text{A win}) = \frac{a}{a+b}$
- It says if one has more money, he is more likely to survive (win) even if playing a "fair" game
- If you are interested in **gambler's ruin problem**, you can find lots of materials on Internet

Hidden Markov Model

- So, what is hidden Markov model (HMM)
- We are unable to observe “states”
  - We need to “**guess**” the number of states
- Each state will emit one observation symbol (over multiple possible symbols)
- Each state has different probability to emit one particular symbol
  - We may not know exact emission probabilities

Hidden Markov Model

- Example: three urns with each urn having some balls



Hidden Markov Model

- A **discrete** hidden Markov model is characterized by the followings
  - N States $S = \{S_1, \dots, S_N\}$ and M observations V=$\{v_1, \dots, v_M\}$ (M = 2 in previous picture: red and blue)
  - State transition probability distribution A (matrix) with elements $P\left(q_{t+1} = S_j | q_t = S_i\right) = a_{ij}$
  - Observation symbol probability distribution B with elements $P(o_t = v_k | q_t = S_i) = b_{ik}$
  - Initial state distribution $\pi = [\pi_1, \dots, \pi_N]$

Hidden Markov Model

- To simplify the matters, we sometimes use
  - $Q = q_1 \dots q_T$ (a sequence of states)
  - O= $o_1 \dots o_T$ (a sequence of observations, $o_k \in \{v_1, \dots, v_M\}$)

Three basic problems in HMM

- **Evaluation problem**: To find the probability of the observation sequence $O = o_1 \dots o_T$ given the model $\lambda = (A, B, \pi)$
- **Decoding problem**: Given observation sequence $O = o_1 \dots o_T$ and model $\lambda = (A, B, \pi)$, how do we choose a corresponding state sequence $Q = q_1 \dots q_T$ which is optimal in some sense, i.e., best explains the observations

Three basic problems in HMM

- **Training (learning) problem**: Given the observation sequence $O = o_1 \dots o_T$, how do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(\lambda|O)$
- These problems are related to each other

Evaluation problem

- Previous picture with known A & B: O=RRBRB
- Brute force method: (enumerate all possibilities)

$$q_1 = S_1, q_2 = S_1, q_3 = S_1, q_4 = S_1, q_5 = S_1$$

$$P_1 = \pi_1 \left(\frac{4}{6}\right) a_{11} \left(\frac{4}{6}\right) a_{11} \left(\frac{2}{6}\right) a_{11} \left(\frac{4}{6}\right) a_{11} \left(\frac{2}{6}\right)$$

R **R** **B** **R** **B**

$$q_1 = S_1, q_2 = S_1, q_3 = S_1, q_4 = S_1, q_5 = S_2$$

$$P_2 = \pi_1 \left(\frac{4}{6}\right) a_{11} \left(\frac{4}{6}\right) a_{11} \left(\frac{2}{6}\right) a_{11} \left(\frac{4}{6}\right) a_{12} \left(\frac{4}{6}\right)$$

Evaluation problem

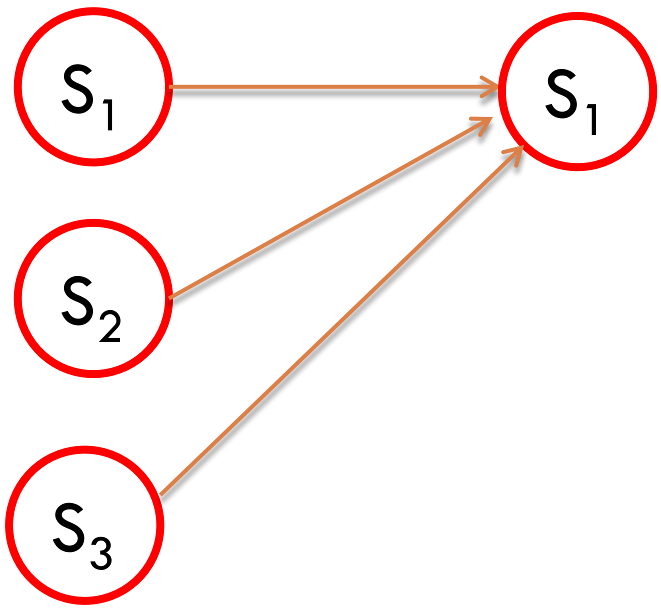
- Still many more possibilities not listed (how many?)
  - We need to add all of the possible probabilities
  - Computational burden is high ($2TN^T$ multiplicatoins)
- What can be done to reduce the computation
- Simplify the problem (T = 2) and $q_2 = S_1$
- We need to consider the followings: $q_1 = S_1$, $q_2 = S_1$, $q_1 = S_2$, $q_2 = S_1$, $q_1 = S_3$, $q_2 = S_1$

Evaluation problem

- Therefore, $P =$

$$\pi_1 b_{1R} a_{11} b_{1R} + \pi_2 b_{2R} a_{21} b_{1R} + \pi_3 b_{3R} a_{13} b_{1R}$$
$$= (\pi_1 b_{1R} a_{11} + \pi_2 b_{2R} a_{21} + \pi_3 b_{3R} a_{13})\ b_{1R}$$

- Graphical illustration



Evaluation problem

- Probability of $q_2 = S_1$ (@ t = 2) is solved by adding probabilities in t = 1
- Therefore, by adding probabilities for each state at time (t-1) multiplied with transition probabilities, we have the probability of one state at time t
- We can solve this problem efficiently by incrementally computing partial (probability) product terms

Evaluation problem

- The approach we used previously is called **dynamic programming:** a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions to next use -- Wikipedia
- Therefore, saving computation time at the expense of using larger storage space

Evaluation problem

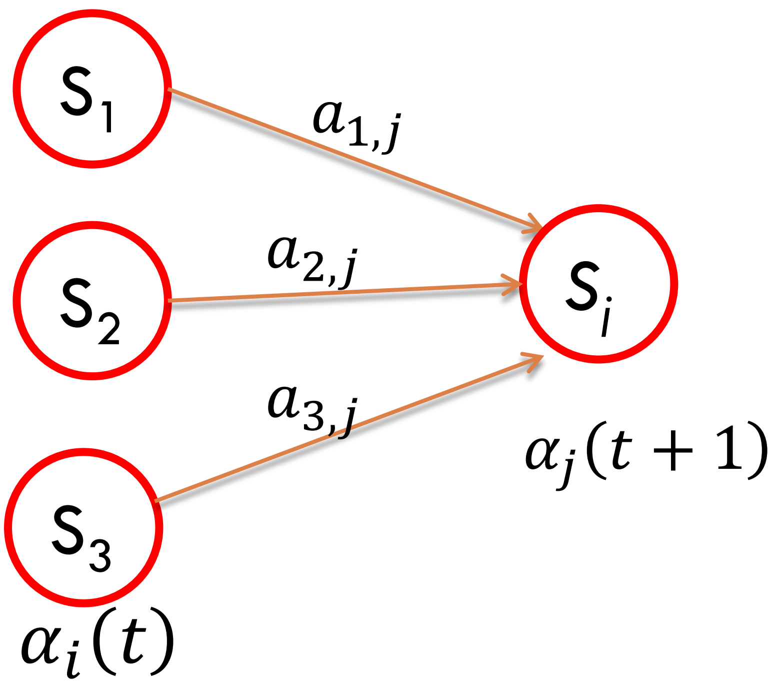
- Define forward variable $\alpha_j(t)$ as the probability of the partial observation sequence until time t with $q_t = S_j$
- Formal definition $\alpha_j(t) = P(o_1 \dots o_t, q_t = S_j | \lambda)$
- Final probability is: $P(O|\lambda) = \sum_{i=1}^{N} \alpha_i(T)$
  - $\alpha_i(T)$ is path probability terminated to state i at t = T
  - Above equation is **summing over all possible states** at time t = T

Evaluation problem

- Forward algorithm:
  - $\alpha_j(t=1) = \pi_j b_{jo_1}$
    - Recall b_{jo_1} is probability o_1 observed for S_j
    - o_1= R in previous picture
  - $\alpha_j(t+1) = \left(\sum_{i=1}^{N} a_{i,j}\alpha_i(t)\right) b_{jo_{(t+1)}}$
- Complexity: N^2T

Evaluation problem

- Why $\alpha_j(t+1) = \left(\sum_{i=1}^{N} a_{i,j}\alpha_i(t)\right) b_{j,o_{(t+1)}}$



Evaluation problem

- Define backward variable $\beta_j(t)$ as the probability of the partial observation sequence **from** time (t+1) to the end (i.e., t = T) with $q_t = S_i$
- Formal definition

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

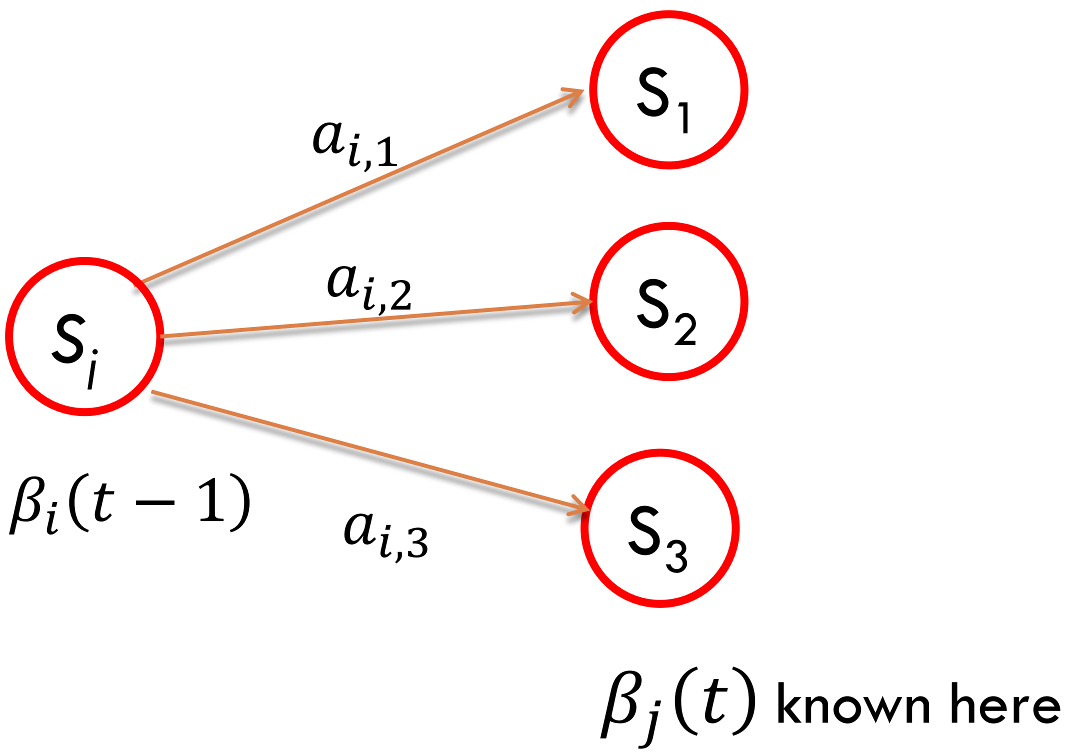
- Backward algorithm has the same complexity as forward algorithm
- For evaluation problem, either algorithm is OK

Evaluation problem

- Backward algorithm:
  - $\beta_i(T) = 1$ (arbitrarily defined, as we do not have o_{T+1})
  - $\beta_i(t-1) = \sum_{j=1}^{N} a_{ij} b_{j o_t} \beta_j(t)$
- If either forward or backward is OK in evaluation problem, why bother to mention backward algorithm
- It will be used in training problem

Evaluation problem

- Why $\beta_i(t-1) = \sum_{j=1}^{N} a_{ij} b_{j o_t} \beta_j(t)$



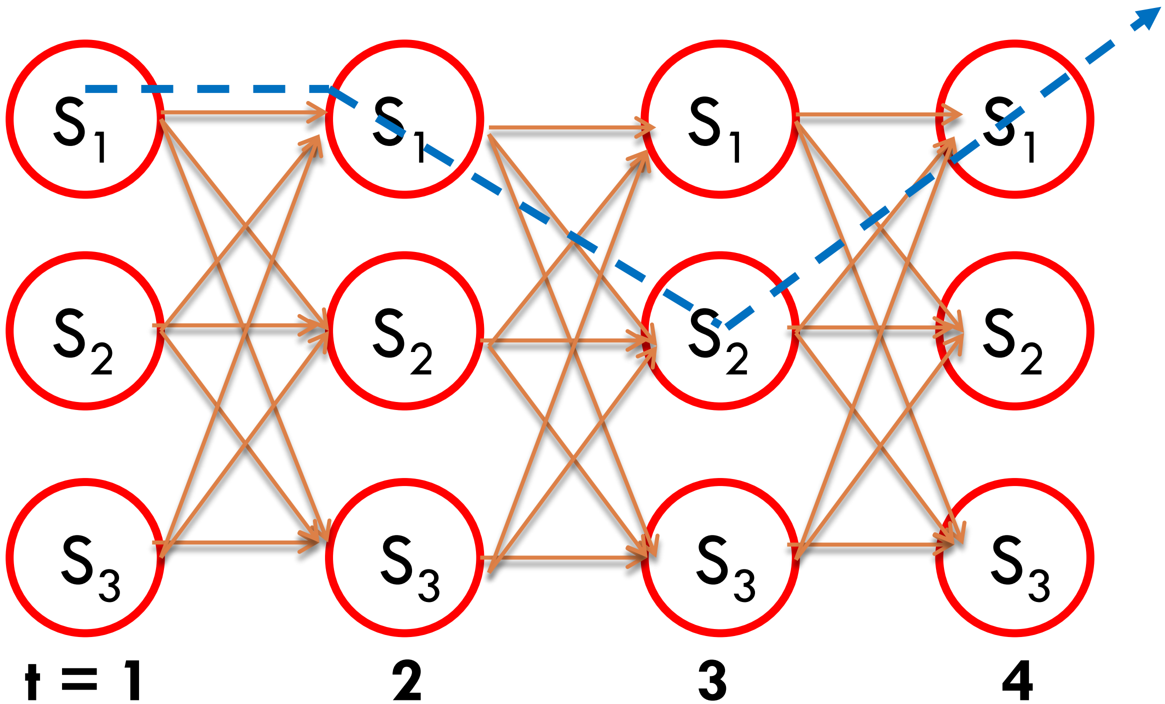
Evaluation problem

- Justification of $\beta_i(T) = 1$
- At t = T = 5, we just need to consider emit probabilities
- Therefore, no need to worry about $\beta_i(\cdot)$ for t = 5
- Numerical problem: $\beta_3(t = 4) = a_{31}\left(\frac{2}{6}\right)$

$$+a_{32}\left(\frac{4}{6}\right)+a_{33}\left(\frac{3}{6}\right)$$

Decoding problem

- Given $\lambda = (A, B, \pi)$ and a sequence of observations $O = o_1 \ldots o_T$, find the most probable sequence of states $Q = q_1 \ldots q_T$ (**Trellis diagram**)



Decoding problem

- Formally, we define the problem as $\max P(Q|O,\lambda) = \max P(Q,O|\lambda)$
- Brute force method: Find all possible paths and compute probability for each path
  - Not possible because too many paths to evaluate
- Keep only the best path in each time step to reduce complexity -- again dynamic programming
- Also known as **Viterbi decoder** in communications

Decoding problem

- Consider a simple numerical example:

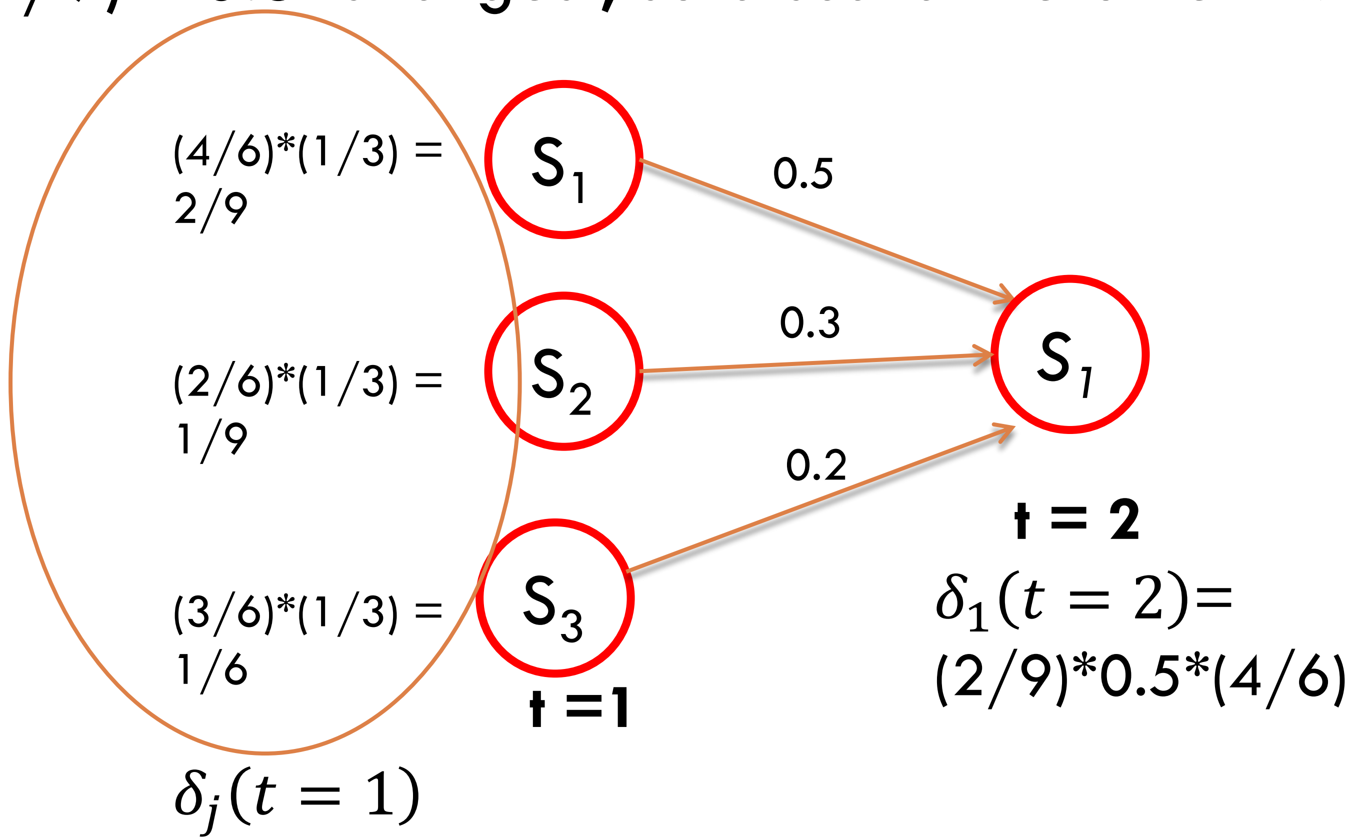
$$\pi = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

- At time t = 1, $o_1 = R$
- From previous picture, we have

$$P(o_1|S_1) = \frac{4}{6}, P(o_1|S_2) = \frac{2}{6}, P(o_1|S_3) = \frac{3}{6}$$

Decoding problem

- Assume $a_{11} = 0.5, a_{12} = 0.3, a_{13} = 0.2$. We know (2/9) * 0.5 is largest, so discard the other two paths



Decoding problem

- Use math to formally write down the idea

$$\delta_j(t) = \max_{q_1,\dots,q_{t-1}} P(q_1 \dots q_t = S_j, o_1 \dots o_t | \lambda)$$

- Decoding algorithm
  - Initial step: $\delta_j(t=1) = \pi_j b_{jo_1}$
  - $\delta_j(t+1) = \left(\max_i \delta_i(t) a_{ij}\right) b_{jo_{t+1}}$
- With backtracking (keeping the maximizing argument for each t and j) we find the optimal solution

Training problem

- Given the observation sequence $O = o_1 \dots o_T$, how do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(\lambda | O)$
  - we need to provide the number of states (not a learnable parameter)
- We are only able to find local optimal solution (not global optimal)

Training problem

- The algorithm can iteratively update model parameters, but need initial estimation of parameters to begin with
- Training algorithm
  - Forward-backward (or Baum-Welch)
  - A special expectation-maximization (EM) algorithm
  - Iteratively train transition probability (A) and emission probability (B)

Training problem

- Ignore emission probability at this moment
- Max likelihood estimation for A is

$$a_{ij} = \frac{C(i \to j)}{\sum_{q \in Q} C(i \to q)} = \frac{\#\text{ times } i \to j}{\#\text{ times } i \to \text{any state}}$$

- But we cannot observe state transition directly (recall what HMM stands for)
- We use $\hat{a}_{ij} = \frac{\text{expected \# transition } i \to j}{\text{expected \# transition from } i}$

Training problem

- Define $\xi_{ij}(t) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$
  - Meaning: Prob from S_i at time *t* to S_j at time (*t*+1) for given observation
  - Will explain how to compute it later
- Therefore, for a given time t $\hat{a}_{ij} = \frac{\xi_{ij}(t)}{\sum_{n=1}^{N} \xi_{in}(t)}$
- Summing over all time, we have

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \sum_{n=1}^{N} \xi_{in}(t)}$$

Training problem

- We can also compute emission matrix
- $\hat{b}_{jk} = \dfrac{\text{expected \# of times in state } j \text{ and observing } o_k}{\text{expecte \# of times in state } j}$
- We know $\text{expecte \# of times in state } j$ is equal to $\text{expected \# transition from } j$
- So, we can reuse previous equation, but summing to T instead of (T-1)

$$\text{expecte \# of times in state } j = \sum_{t=1}^{T} \sum_{n=1}^{N} \xi_{in}(t)$$

Training problem

- Therefore,

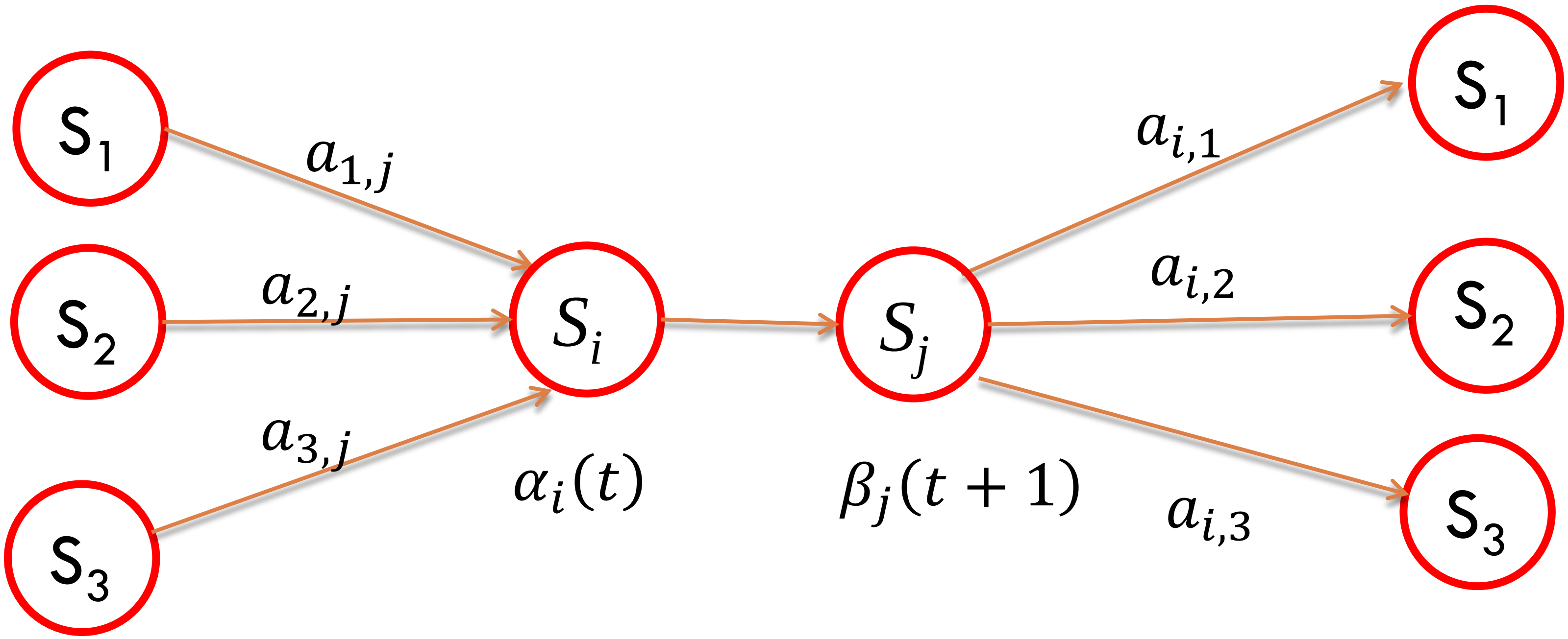
$$\hat{b}_{jk} = \frac{\sum_{t=1}^{T}\left(\sum_{n=1}^{N}\xi_{jn}(t)\right)\ell(o_t == v_k)}{\sum_{t=1}^{T}\sum_{n=1}^{N}\xi_{jn}(t)}$$

where $\ell(\exp) = \begin{cases} 1, \exp\text{ is true} \\ 0, \exp\text{ is false} \end{cases}$

- Sometimes, we define $\gamma_j(t) = \sum_{n=1}^{N}\xi_{jn}(t)$ to simplify the expression

Training problem

- How to find $\xi_{ij}(t)$
- $\xi_{ij}(t)$ = Prob. from S_i at time *t* to S_j at time (*t*+1)



Training problem

- Wek know $\xi_{ij}(t) = P\left(q_t = S_i, q_{t+1} = S_j \middle| O, \lambda\right) = \dfrac{\alpha_i(t)\beta_j(t+1)a_{ij}b_{jo_{t+1}}}{P(O|\lambda)}$

$$= \frac{\alpha_i(t)\beta_j(t+1)a_{ij}b_{jo_{t+1}}}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i(t)\beta_j(t+1)a_{ij}b_{jo_{t+1}}}$$

- Recall

$$\gamma_i(t) = \sum_{j=1}^{N}\xi_{ij}(t) = \sum_{j=1}^{N}\alpha_i(t)\beta_j(t+1)a_{ij}b_{jo_{t+1}}$$

Training problem

- However,

$$\sum_{j=1}^{N} \alpha_i(t)\beta_j(t+1)a_{ij}b_{jo_{t+1}} = \sum_{j=1}^{N} \alpha_i(t)\beta_j(t)$$

(Meaning: summing over all possible paths)

Therefore, we have the following equations for learning algorithm

Iterative training algorithm

$$\xi_{ij}(t) = \frac{\alpha_i(t)\beta_j(t+1)a_{ij}b_{jo_{t+1}}}{\sum_{n=1}^{N}\alpha_n(t)\beta_n(t)}$$

$$\gamma_i(t) = \sum_{j=1}^{N}\xi_{ij}(t)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_{ij}(t)}{\sum_{t=1}^{T-1}\gamma_i}$$

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

$$\hat{b}_{jk} = \frac{\sum_{t=1}^{T}\left(\gamma_j(t)\ell(o_t = v_k)\right)}{\sum_{t=1}^{T}\gamma_j}$$

Numerical example

- A good numerical example is available at

http://cs.jhu.edu/~jason/papers/eisner.hmm.xls

- You may want to download and carefully study it
- One word of caution: the author of the excel file made a small mistake in the computation
- Check out what is wrong (exercise)

Continuous HMM

- Can extend discrete HMM to continuous HMM
- Each state emits a continuous value
- Emission probability is modeled by GMM
- Therefore, in continuous case: (one state HMM) = GMM
- This part is Important … but do not have time to fully cover it

Using HMM

- To use HMM, we need to **train one model per class** data
- For example, to classify IRIS data, we need three models (individually trained) --- Note: it is only for explanation, **not** real case
- In reality, HMM is used only to deal with time-series data (data related to time)
- Example: previously we used HMM to classify audio segments with or without singing voices

Using HMM

- For classification, we feed the test sample to all trained HMM models
- The model provides highest likelihood (recall Viterbi decoder) is the winner
- The test sample is labeled as the class of the winner

Using continuous HMM

- Proper parameter initialization is **very** important
- Normalization sometimes is also very important
- How to determine the number of states and number of mixers also important
- Sometimes using canned programs may encounter error messages, such as "negative probability" or matrix not invertible (covariance, remember GMM?)
- Need to deal with these problems