Hidden Markov Models (HMMs)

Reading

More details can be found in

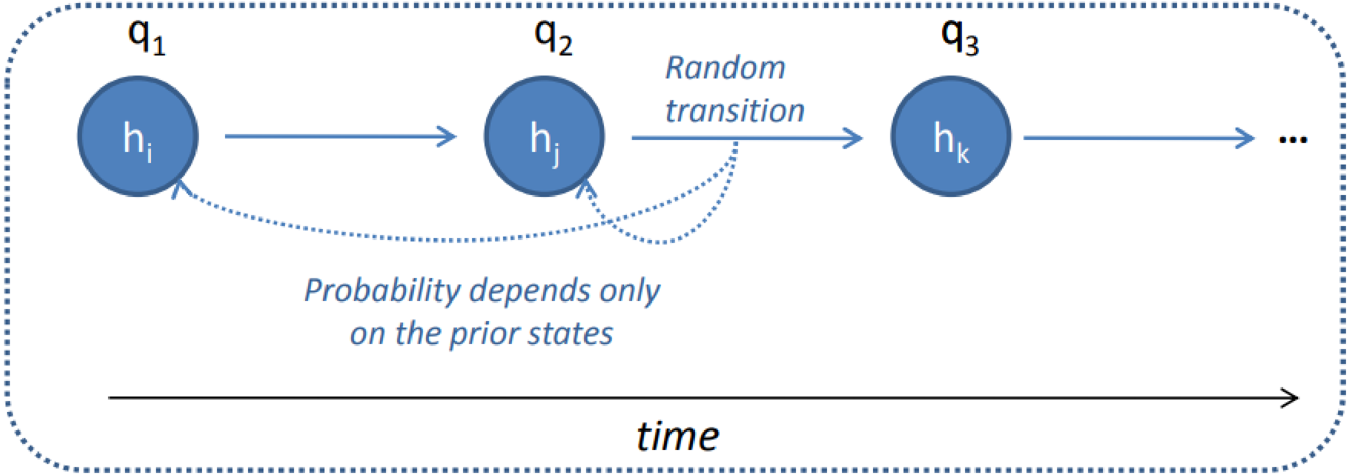
- Chapter 13 "Sequential Data" of Bishop's *Pattern Recognition and Machine Learning*

Applications

- Speech generation, speech recognition
- Financial forecasting: stock prices, currency exchange rates etc.
- DNA sequencing
- Weather prediction
- Video analytics
- Machine translation

Markov Chains

- Markov Chains model sequential processes.
- Consider a discrete random variable q with states $\{h_1, \ldots, h_n\}$.
- State of q changes randomly in discrete time steps.
- Transition probability depends only on the k previous states.
  - Markov Property



Markov Chain

Transition Probabilities

- Most simplest Markov chain:
  - Transition Probability depends only on the previous state (i.e. k=1):

$$P(q_t = h_i | q_{t-1}, \dots, q_1) = P(q_t = h_i | q_{t-1})$$

  - Transition Probability is time invariant:

$$P(q_t = h_i | q_{t-1}) = P(q_{t-1} = h_i | q_{t-2})$$

- In this case, the Markov chain is defined by:
  - An $(n \times n)$ Matrix *T* containing state change probabilities:

$$T_{ij} = P(q_t = h_i | q_{t-1} = h_j)$$

  - An n-dimensional vector π containing initial state probabilities:

$$\pi_i = P(q_1 = h_i)$$

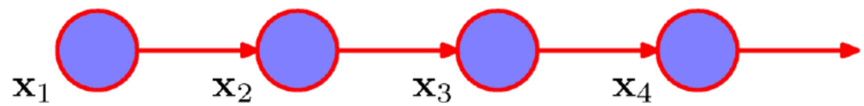
  - Since π and *K* contain probabilities, they have to be normalized:

$$\sum_{i=1}^{n} \pi_i = 1 \qquad \forall a : \sum_{i=1}^{n} K_{ai} = 1$$

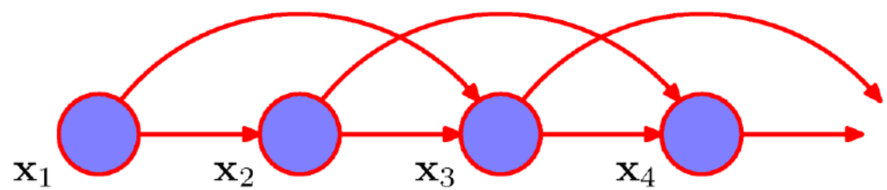
Nth Order Markov Chain

- Markov Assumption

1st order $$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_n | X_{n-1})$$



2nd order $$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_n | X_{n-1}, X_{n-2})$$



Prelude

- Given observations $(O_1, \dots O_n)$, hidden states $(S_1, \dots S_n)$, we want to define a model by providing structure to our joint distribution $p(O_1, \dots, On, S_1, \dots, S_n)$. We have

$$p(O_1, \dots, On, S_1, \dots, S_n) = p(S_1, \dots, S_n) p(O_1, \dots, O_n \mid S_1, \dots, S_n)$$

  using chain rule.

- Repeated application of chain rule to $p(S_1, \dots, S_n)$ yields

$$\begin{aligned}
p(S_1, \dots, S_n) &= p(S_1, \dots, S_{n-1}) p(S_n \mid S_1, \dots, S_{n-1}) \\
&= p(S_1, \dots, S_{n-2}) p(S_{n-1} \mid S_1, \dots, S_{n-2}) p(S_n \mid S_1, \dots, S_{n-1}) \\
&\vdots \\
&= p(S_1) p(S_2 \mid S_1) p(S_3 \mid S_1, S_2) \cdots p(S_n \mid S_1, \dots, S_{n-1}).
\end{aligned}$$

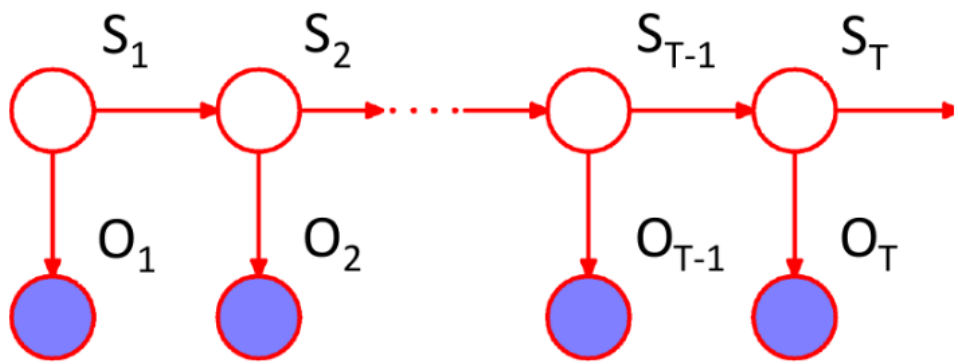
- Similarly for $p(O_1, \ldots, O_n \mid S_1, \ldots, S_n)$, we have

$$\begin{aligned} &p(O_1, \ldots, O_n \mid S_1, \ldots, S_n) \\ &\qquad = p(O_1 \mid S_1, \ldots, S_n) p(O_2 \mid O_1, S_1, \ldots, S_n) \cdots p(O_n \mid O_1, \ldots, O_{n-1}, S_1, \ldots, S_n). \end{aligned}$$

- So far we have enforced no assumptions; we will do so in the next few slides.

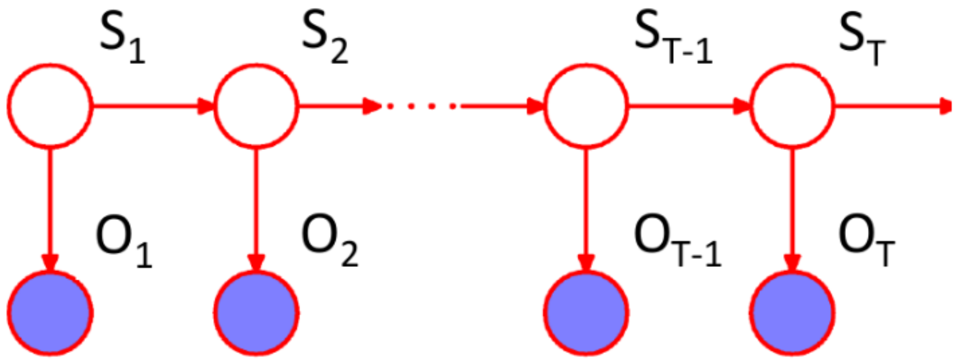
HMM definition

- Distributions that characterize sequential data with few parameters but are not limited by strong Markov assumptions.



Observation space	$O_t \in \{y_1, y_2, \ldots, y_K\}$
Hidden states	$S_t \in \{1, \ldots, I\}$

Joint Distribution Factorization



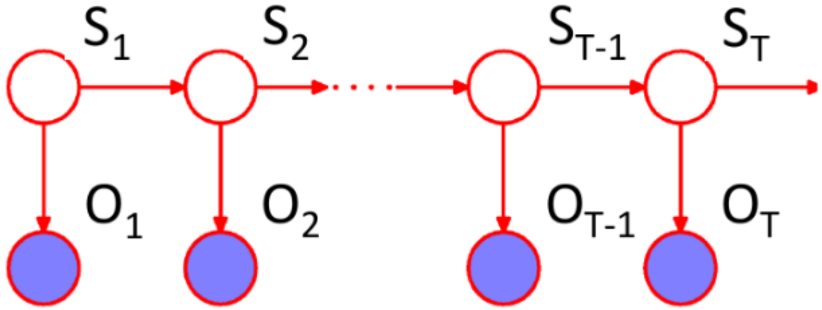
$$p(S_1, \ldots, S_T, O_1, \ldots, O_T) \quad = \quad \prod_{t=1}^{T} p(O_t|S_t) \prod_{t=1}^{T} p(S_t|S_{t-1})$$

1st order Markov assumption on hidden states $\{S_t\}$ t = 1, ..., T (can be extended to higher order).

Note: O_t depends on all previous observations $\{O_{t-1},...O_1\}$

Model Parameters

- Parameters – stationary/homogeneous markov model (independent of time t)



Initial probabilities

$p(S_1 = i) = \pi_i$

Transition probabilities

$p(S_t = j | S_{t-1} = i) = p_{ij}$

Emission probabilities

$p(O_t = y | S_t = i) = q_i^y$

$$p(\{S_t\}_{t=1}^T, \{O_t\}_{t=1}^T) = p(S_1) \prod_{t=2}^{T} p(S_t|S_{t-1}) \prod_{t=1}^{T} p(O_t|S_t)$$

HMM Example

- The Dishonest Casino

A casino has two die:

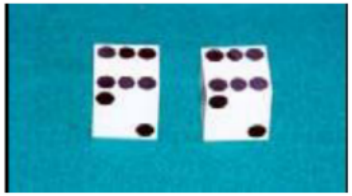
Fair dice

P(1) = P(2) = P(3) = P(5) = P(6) = 1/6

Loaded dice

P(1) = P(2) = P(3) = P(5) = 1/10
P(6) = ½

Casino player switches back-&-forth between fair and loaded die once every 20 turns



HMM Example

GIVEN: A sequence of rolls by the casino player

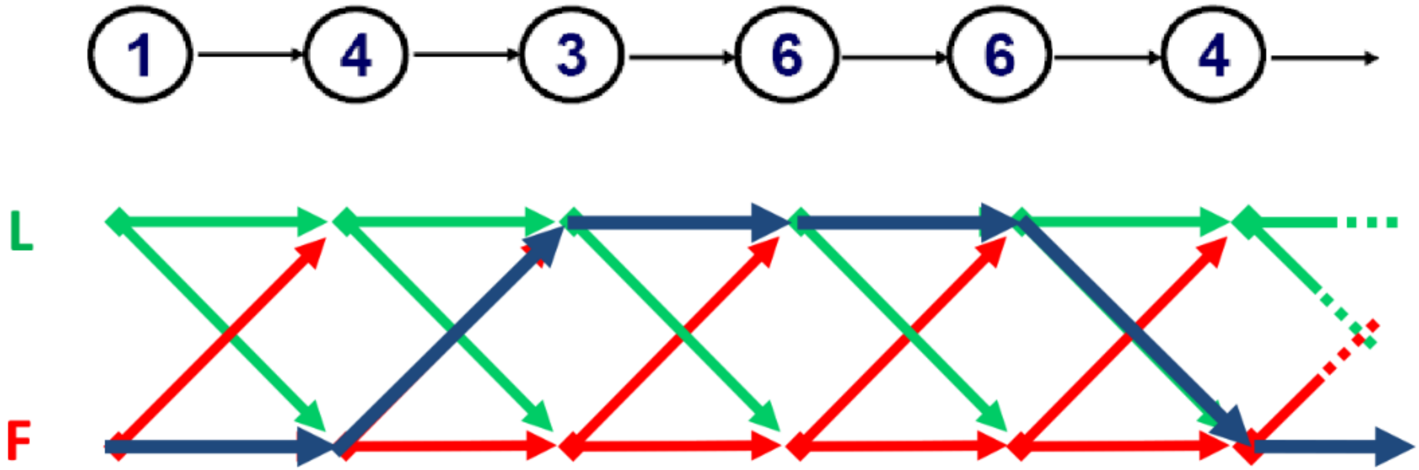
1245526462146146136136661664661636616366163616515615115146123562344

QUESTION

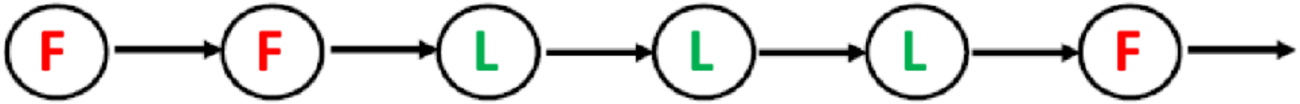
- How likely is this sequence, given our model of how the casino works?
  - This is the **EVALUATION** problem in HMMs
- What portion of the sequence was generated with the fair die, and what portion with the loaded die?
  - This is the **DECODING** question in HMMs
- How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?
  - This is the **LEARNING** question in HMMs

HMM Example

- Observed sequence: $\{O_t\}_{t=1}^{T}$

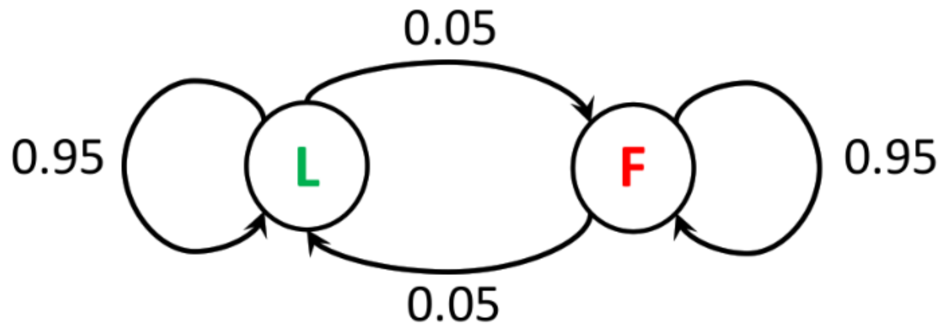


- Hidden sequence $\{S_t\}_{t=1}^{T}$ or segmentation):



HMM Example

- Switch between F and L once every 20 turns (1/20 = 0.05)



- HMM Parameters

Initial probs	$P(S_1 = L) = 0.5 = P(S_1 = F)$
Transition probs	$P(S_t = L/F \mid S_{t-1} = L/F) = 0.95$
	$P(S_t = F/L \mid S_{t-1} = L/F) = 0.05$
Emission probabilities	$P(O_t = y \mid S_t = F) = 1/6 \quad y = 1,2,3,4,5,6$
	$P(O_t = y \mid S_t = L) = 1/10 \quad y = 1,2,3,4,5$
	$= 1/2 \quad y = 6$

Three Main Learning Problems

- Evaluation – Given HMM parameters & observation seqn $\{O_t\}_{t=1}^T$

  find $p(\{O_t\}_{t=1}^T)$ prob of observed sequence

- Decoding – Given HMM parameters & observation seqn $\{O_t\}_{t=1}^T$

  find $\arg\max_{s_1,\ldots,s_T} p(\{S_t\}_{t=1}^T | \{O_t\}_{t=1}^T)$ most probable sequence of hidden states

- Learning – Given HMM with unknown parameters and $\{O_t\}_{t=1}^T$ observation sequence

  find $\arg\max_\theta p(\{O_t\}_{t=1}^T | \theta)$ parameters that maximize likelihood of observed data

Three Main Learning Problems

- **Evaluation** – What is the probability of the observed sequence? Forward Algorithm

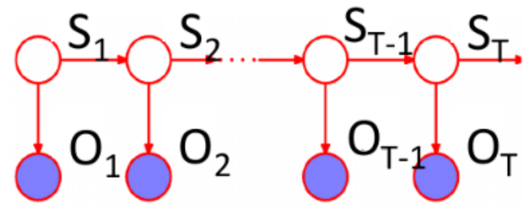
- **Decoding** – What is the probability that the third roll was loaded given the observed sequence? Forward-Backward Algorithm

  – What is the most likely die sequence given the observed sequence? Viterbi Algorithm

- **Learning** – Under what parameterization is the observed sequence most probable? Baum-Welch Algorithm (EM)

Evaluation: Forward Algorithm

- Given HMM parameters $p(S_1), p(S_t|S_{t-1}), p(O_t|S_t)$& observation sequence $\{O_t\}_{t=1}^T$



find probability of observed sequence

$$
\begin{aligned}
p(\{O_t\}_{t=1}^T) &= \sum_{S_1,\ldots,S_T} p(\{O_t\}_{t=1}^T, \{S_t\}_{t=1}^T) \\
&= \sum_{S_1,\ldots,S_T} p(S_1) \prod_{t=2}^{T} p(S_t|S_{t-1}) \prod_{t=1}^{T} p(O_t|S_t)
\end{aligned}
$$

requires summing over all possible hidden state values at all times – K^T exponential # terms!

Instead: $p(\{O_t\}_{t=1}^T) = \sum_k \underbrace{p(\{O_t\}_{t=1}^T, S_T = k)}_{\alpha_T^k}$

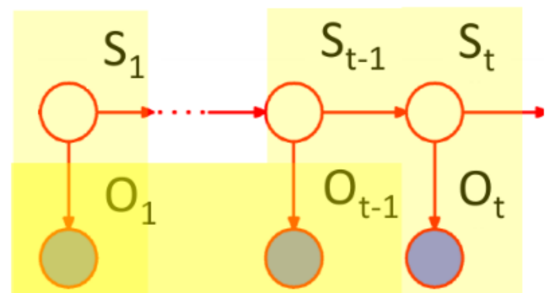
Compute recursively

Evaluation: Forward Algorithm

$$p(\{O_t\}_{t=1}^T) = \sum_k p(\{O_t\}_{t=1}^T, S_T = k) = \sum_k \alpha_T^k$$

Compute forward probability α_t^k recursively over t

$$\alpha_t^k := p(O_1, \ldots, O_t, S_t = k)$$



Introduce S_{t-1}

.

Chain rule

.

Markov assumption

.

$$= p(O_t | S_t = k) \sum_i \alpha_{t-1}^i p(S_t = k | S_{t-1} = i)$$

Evaluation: Forward Algorithm

Can compute α_t^k for all k, t using dynamic programming:

- Initialize: $\alpha_1^k = p(O_1 | S_1 = k)\, p(S_1 = k)$ for all k

- Iterate: for t = 2, ..., T

$$\alpha_t^k = p(O_t | S_t = k) \sum_i \alpha_{t-1}^i\, p(S_t = k | S_{t-1} = i) \qquad \text{for all k}$$

- Termination: $p(\{O_t\}_{t=1}^T) = \sum_k \alpha_T^k$

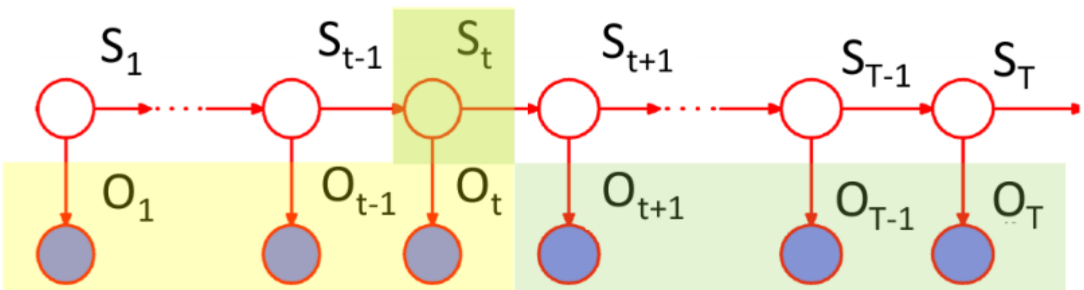
Decoding: Backward Algorithm

- Given HMM parameters $p(S_1), p(S_t|S_{t-1}), p(O_t|S_t)$& observation sequence $\{O_t\}_{t=1}^T$

  find probability that hidden state at time t was k $p(S_t = k|\{O_t\}_{t=1}^T)$

$$
\begin{aligned}
p(S_t = k, \{O_t\}_{t=1}^T) &= p(O_1, \ldots, O_t, S_t = k, O_{t+1}, \ldots, O_T) \\
&= \underbrace{p(O_1, \ldots, O_t, S_t = k)}_{\alpha_t^k}\underbrace{p(O_{t+1}, \ldots, O_T|S_t = k)}_{\beta_t^k}
\end{aligned}
$$

Compute recursively

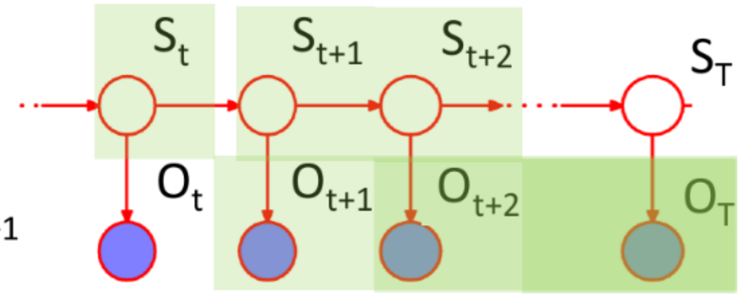


Decoding: Backward Algorithm

$$p(S_t = k, \{O_t\}_{t=1}^{T}) = p(O_1, \ldots, O_t, S_t = k)p(O_{t+1}, \ldots, O_T | S_t = k) = \alpha_t^k \beta_t^k$$

Compute forward probability β_t^k recursively over t

$$\beta_t^k \quad := \quad p(O_{t+1}, \ldots, O_T | S_t = k)$$



Introduce S_{t+1}

Chain rule

Markov assumption

$$= \quad \sum_i p(S_{t+1} = i | S_t = k) p(O_{t+1} | S_{t+1} = i) \beta_{t+1}^i$$

Decoding: Backward Algorithm

Can compute β_t^k for all k, t using dynamic programming:

- Initialize: $\beta_T^k = 1$ for all k

- Iterate: for t = T-1, …, 1

$$\beta_t^k = \sum_i p(S_{t+1} = i | S_t = k) p(O_{t+1} | S_{t+1} = i) \beta_{t+1}^i \quad \text{for all k}$$

- Termination: $p(S_t = k, \{O_t\}_{t=1}^T) = \alpha_t^k \beta_t^k$

$$p(S_t = k | \{O_t\}_{t=1}^T) = \frac{p(S_t = k, \{O_t\}_{t=1}^T)}{p(\{O_t\}_{t=1}^T)} = \frac{\alpha_t^k \beta_t^k}{\sum_i \alpha_t^i \beta_t^i}$$

Decoding: Viterbi Algorithm

- Most likely state assignment at time t

$$\arg\max_k p(S_t = k | \{O_t\}_{t=1}^T) = \arg\max_k \alpha_t^k \beta_t^k$$

E.g. Which die was most likely used by the casino in the third roll given the observed sequence?

- Most likely assignment of state sequence

$$\arg \max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T | \{O_t\}_{t=1}^T)$$

E.g. What was the most likely sequence of die rolls used by the casino given the observed sequence?

Not the same solution !

MLA of x?
MLA of (x,y)?

x	y	P(x,y)
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

Decoding: Viterbi Algorithm

- Given HMM parameters $p(S_1), p(S_t|S_{t-1}), p(O_t|S_t)$ & observation sequence $\{O_t\}_{t=1}^T$

  find most likely assignment of state sequence

$$\arg\max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T | \{O_t\}_{t=1}^T) = \arg\max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T, \{O_t\}_{t=1}^T)$$

$$= \arg\max_k \underbrace{\max_{\{S_t\}_{t=1}^{T-1}} p(S_T = k, \{S_t\}_{t=1}^{T-1}, \{O_t\}_{t=1}^T)}_{V_T^k}$$

Compute recursively

V_T^k - probability of most likely sequence of states ending at state $S_T = k$

Decoding: Viterbi Algorithm

$$\max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T, \{O_t\}_{t=1}^T) = \max_k V_T^k$$

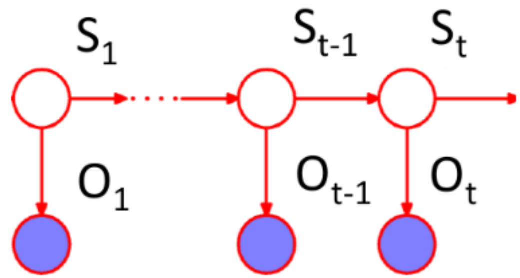
Compute probability V_t^k recursively over t

$$V_t^k \quad := \quad \max_{S_1,\ldots,S_{t-1}} p(S_t = k, S_1, \ldots, S_{t-1}, O_1, \ldots, O_t)$$

.
.
.

Bayes rule

Markov assumption



$$= \quad p(O_t | S_t = k) \max_i p(S_t = k | S_{t-1} = i) V_{t-1}^i$$

Decoding: Viterbi Algorithm

Can compute V_t^k for all k, t using dynamic programming:

- Initialize: $V_1^k = p(O_1 | S_1 = k) p(S_1 = k)$ for all k

- Iterate: for t = 2, …, T

$$V_t^k \quad = \quad p(O_t|S_t = k) \max_i p(S_t = k|S_{t-1} = i) V_{t-1}^i$$ for all k

- Termination: $\max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T, \{O_t\}_{t=1}^T) = \max_k V_T^k$

Traceback:

$$S_T^* = \arg\max_k V_T^k$$

$$S_{t-1}^* = \arg\max_i p(S_t^*|S_{t-1} = i) V_{t-1}^i$$

Learning: Baum-Welch (EM) Algorithm

- Given HMM with unknown parameters $\theta = \{\{\pi_i\}, \{p_{ij}\}, \{q_i^k\}\}$
  and observation sequence $\mathbf{O} = \{O_t\}_{t=1}^{T}$

  find parameters that maximize likelihood of observed data

  $$\arg\max_{\theta} p(\{O_t\}_{t=1}^{T} | \theta)$$

  But likelihood doesn't factorize since observations not i.i.d.

  hidden variables – state sequence $\{S_t\}_{t=1}^{T}$

  EM (Baum-Welch) Algorithm:

  E-step – Fix parameters, find expected state assignments

  M-step – Fix expected state assignments, update parameters

Learning: Baum-Welch (EM) Algorithm

- Start with random initialization of parameters
- **E-step** – Fix parameters, find expected state assignments

$$\gamma_i(t) = p(S_t = i | O, \theta) = \frac{\alpha_t^i \beta_t^i}{\sum_j \alpha_t^j \beta_t^j}$$

Forward-Backward algorithm

$$\xi_{ij}(t) = p(S_{t-1} = i, S_t = j | O, \theta)$$

$$= \frac{p(S_{t-1} = i | O, \theta) p(S_t = j, O_t, \dots, O_T | S_{t-1} = i, \theta)}{p(O_t, \dots, O_T | S_{t-1} = i, \theta)}$$

$$= \frac{\gamma_i(t-1)\ p_{ij}\ q_j^{O_t}\ \beta_t^j}{\beta_{t-1}^i}$$

Learning: Baum-Welch (EM) Algorithm

- Start with random initialization of parameters
- E-step:
  - $\gamma_i(t) = p_\theta(S_t = i \mid O) = \dfrac{\alpha_t^i \beta_t^i}{\sum_j \alpha_t^j \beta_t^j}$
  - $\xi_{ij}(t) = p_\theta(S_{t-1} = i, S_t = j \mid O) = \dfrac{\gamma_i(t-1) p_{ij} q_j^{O_t} \beta_t^j}{\beta_{t-1}^i}$
- M-step:
  - $\pi_i = \dfrac{\gamma_i(1)}{\sum_{j=1}^K \gamma_j(1)}$
  - $p_{ij} = \dfrac{\sum_{t=2}^T \xi_{ij}(t)}{\sum_{k=1}^K \sum_{t=2}^T \xi_{ik}(t)} = \dfrac{\sum_{t=2}^T \xi_{ij}(t)}{\sum_{t=2}^T \gamma_i(t)}$
  - $q_i^k = \dfrac{\sum_{t=1}^T \delta_{O_t=k} \gamma_i(t)}{\sum_{t=1}^T \gamma_i(t)}$