Hidden Markov Model

Statistical Machine Learning (ENGG*6600*02)

School of Engineering,
University of Guelph, ON, Canada

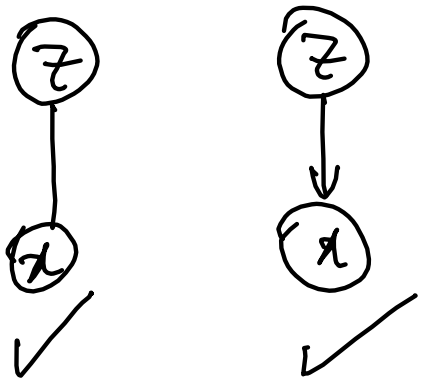
Course Instructor: Benyamin Ghojogh
Summer 2023

Probabilistic Graphical Models

Markov and Bayesian Networks

$P(x|z)$

- A **Probabilistic Graphical Model (PGM)** is a grpah-based representation of a complex distribution in the possibly high dimensional space [1].
- In other words, PGM is a combination of **graph theory** and **probability theory**.
- In a PGM, the random variables are represented by **nodes or vertices**. There exist **edges** between two variables which have interaction with one another in terms of probability. Different conditional probabilities can be represented by a PGM.
- There exist two types of PGM which are **Markov network** (also called **Markov random field**) and **Bayesian network** [1]. In the Markov network and Bayesian network, the edges of graph are **undirected** and **directed**, respectively.



The Markov Property

- Consider a **times series** of random variables $X_1, X_2, \ldots, X_n$. In general, the **joint probability** of these random variables can be written as:

$$\mathbb{P}(X_1, X_2, \ldots, X_n) = \mathbb{P}(X_1)\,\mathbb{P}(X_2 \mid X_1)\,\mathbb{P}(X_3 \mid X_2, X_1)\ldots\mathbb{P}(X_n \mid X_{n-1}, \ldots, X_2, X_1), \quad (1)$$

  according to **chain (or multiplication) rule** in probability.

- (The first order) **Markov property** is an assumption which states that in a time series of random variables $X_1, X_2, \ldots, X_n$, every random variable is merely **dependent on the latest previous random variable** and not the others. In other words:

$$\mathbb{P}(X_i \mid X_{i-1}, X_{i-2}, \ldots, X_2, X_1) = \mathbb{P}(X_i \mid X_{i-1}). \quad (2)$$

- Hence, with Markov property, the chain rule is simplied to:

$$\mathbb{P}(X_1, X_2, \ldots, X_n) = \mathbb{P}(X_1)\,\mathbb{P}(X_2 \mid X_1)\,\mathbb{P}(X_3 \mid X_2)\ldots\mathbb{P}(X_n \mid X_{n-1}). \quad (3)$$

- The Markov property can be of any order. For example, in a **second order Markov property**, a random variable is dependent on the latest and one-to-latest variables. Usually, the default Markov property is of order one.

- A ~~stochastic process~~ which has the Markov process is called a **Markovian process** (or **Markov process**).

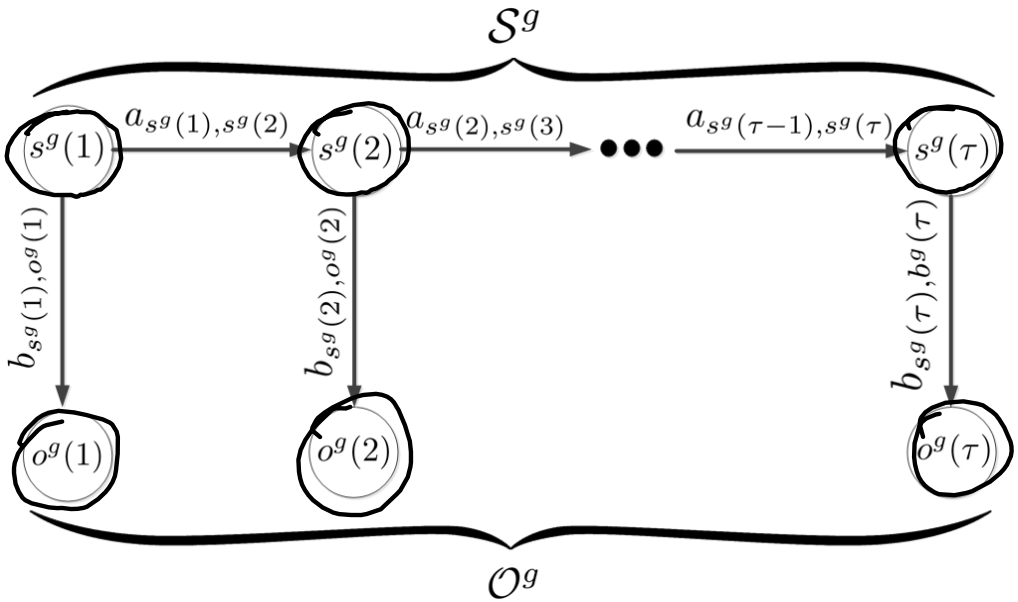
Discrete Time Markov Chain

- A **Markov chain** is a PGM which has Markov property.
- The Markov chain can be either **directed** or **undirected**.
- Usually, Markov chain is a **Bayesian network** where the edges are **directed**.
- It is important not to confuse **Markov chain** with **Markov network**.
- There are two types of Markov Chain which are **Discrete Time Markov Chain (DTMC)** [2] and **Continuous Time Markov Chain (CTMC)** [3]. As it is obvious from their names, in DTMC and CTMC, the time of transitions from a random variable to another one is and is not partitioned into **discrete slots**, respectively.
- If the variables in a DTMC are considered as states, the DTMC can be viewed as a **Finite-State Machine (FSM)** or a **Finite-State Automaton (FSA)** [4].

Hidden Markov Model (HMM)

Hidden Markov Model (HMM)

- **HMM** is a DTMC which contains a sequence of **hidden variables** (named **states**) in addition to a sequence of **emitted observation symbols** (**outputs**).



- We have an observation sequence of length τ which is the number of clock times, $t \in \{1, \ldots, \tau\}$. Let n and m denote the number of states and observation symbols, respectively.

Hidden Markov Model (HMM)

- We show the sets of states and possible observation symbols by $\mathcal{S} = \{s_1, \ldots, s_n\}$ and $\mathcal{O} = \{o_1, \ldots, o_m\}$, respectively.
- We show being in state s_i and in observation symbol o_i at time t by $s_i(t)$ and $o_i(t)$, respectively.
- Let $\mathbb{R}^{n \times n} \ni \boldsymbol{A} = [a_{i,j}]$ be the state **Transition Probability Matrix (TPM)**, where:

$$a_{i,j} := \mathbb{P}\big(s_j(t+1) \,|\, s_i(t)\big). \tag{4}$$

- We have:

$$\sum_{j=1}^{n} a_{i,j} = 1. \tag{5}$$

- The **Emission Probability Matrix (EPM)** is denoted by as $\mathbb{R}^{n \times m} \ni \boldsymbol{B} = [b_{i,j}]$ where:

$$b_{i,j} := \mathbb{P}\big(o_j(t) \,|\, s_i(t)\big), \tag{6}$$

  which is the probability of emission of the observation symbols from the states.
- We have:

$$\sum_{j=1}^{n} b_{i,j} = 1. \tag{7}$$

Hidden Markov Model (HMM)

- Let the initial state distribution be denoted by the vector $\mathbb{R}^n \ni \boldsymbol{\pi} = [\pi_1, \ldots, \pi_n]$ where:

$$\pi_i := \mathbb{P}\big(s_i(1)\big), \tag{8}$$

and:

$$\sum_{i=1}^{n} \pi_i = 1, \tag{9}$$

to satisfy the probability properties.

- An HMM model is denoted by the tuple $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$.
- Assume that a **sequence of states** is generated by the HMM according to the TPM. We denote this generated sequence of states by $\mathcal{S}^g := s^g(1), \ldots, s^g(\tau)$ where $s^g(t) \in \mathcal{S}, \forall t$.
- Likewise, a **sequence of outputs (observations)** is generated by the HMM according to EPM. We denote this generated sequence of output symbols by $\mathcal{O}^g := o^g(1), \ldots, o^g(\tau)$ where $o^g(t) \in \mathcal{O}, \forall t$.

Hidden Markov Model (HMM)

- We denote the probability of transition from state $s^g(t)$ to $s^g(t+1)$ by $a_{s^g(t),s^g(t+1)}$. So:

$$a_{s^g(t),s^g(t+1)} := \mathbb{P}(s^g(t+1) \mid s^g(t)). \tag{10}$$

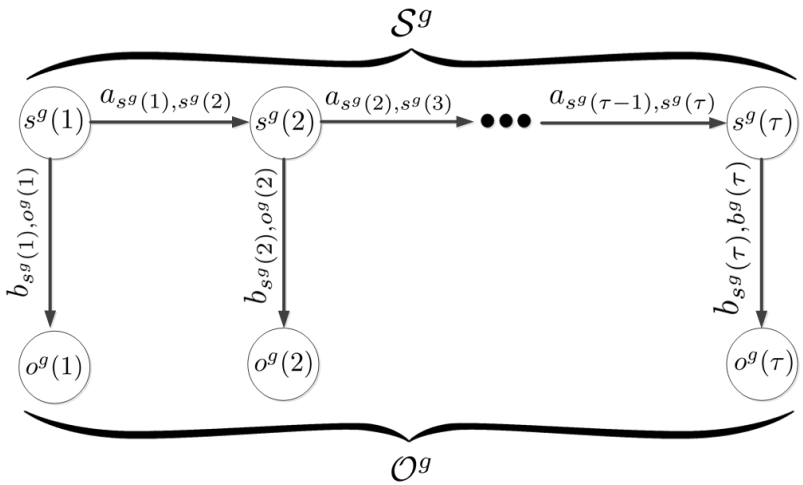
- Note that $s^g(t) \in \mathcal{S}$ and $s^g(t+1) \in \mathcal{S}$.
- We also denote:

$$\pi_{s^g(i)} := \mathbb{P}(s^g(i)). \tag{11}$$

- Likewise, we denote the probability of state $s^g(t)$ emitting the observation $o^g(t)$ by $b_{s^g(t),o^g(t)}$. So:

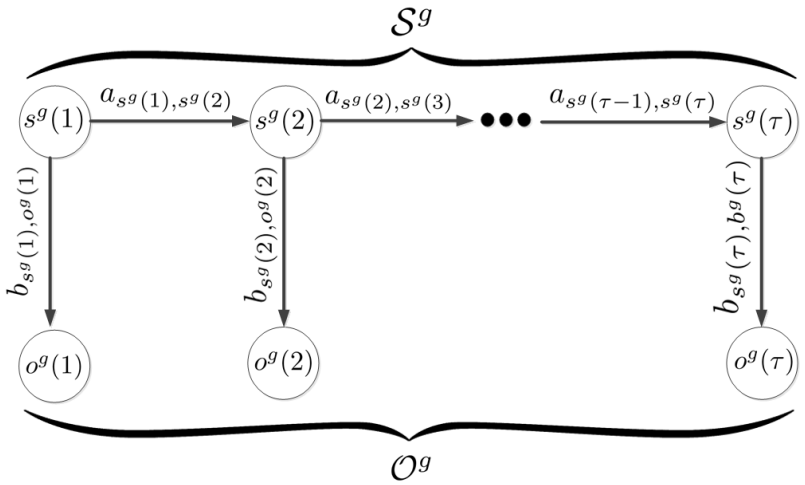
$$b_{s^g(t),o^g(t)} := \mathbb{P}(o^g(t) \mid s^g(t)). \tag{12}$$

- Note that $s^g(t) \in \mathcal{S}$ and $o^g(t) \in \mathcal{O}$.



Likelihood and Expectation Maximization in HMM

Likelihood



- According to the figure, the likelihood of occurrence of the state sequence $\mathcal{S}^g$ and the observation sequence $\mathcal{O}^g$ is [5]:

$$
\begin{aligned}
L = \mathbb{P}(\mathcal{S}^g, \mathcal{O}^g) &= \mathbb{P}\big(s^g(1)\big)\,\mathbb{P}(\text{next states} \,|\, \text{previous states})\,\mathbb{P}(\mathcal{O}^g \,|\, \mathcal{S}^g) \\
&= \mathbb{P}\big(s^g(1)\big) \prod_{t=1}^{\tau-1} \mathbb{P}(s^g(t+1) \,|\, s^g(t)) \prod_{t=1}^{\tau} \mathbb{P}(o^g(t) \,|\, s^g(t)) \\
&= \pi_{s^g(1)} \prod_{t=1}^{\tau-1} a_{s^g(t),s^g(t+1)} \prod_{t=1}^{\tau} b_{s^g(t),o^g(t)}. \qquad (13)
\end{aligned}
$$

Likelihood

- We found:

$$L = \pi_{s^g(1)} \prod_{t=1}^{\tau-1} a_{s^g(t),s^g(t+1)} \prod_{t=1}^{\tau} b_{s^g(t),o^g(t)}.$$

- The log-likelihood is:

$$\ell = \log(L) = \log \pi_{s^g(1)} + \sum_{t=1}^{\tau-1} \log(a_{s^g(t),s^g(t+1)}) + \sum_{t=1}^{\tau} \log(b_{s^g(t),o^g(t)}). \tag{14}$$

- Let $\mathbf{1}_i$ be a vector with entry one at index i, i.e., $\mathbf{1}_i := [0, 0, \ldots, 0, 1, 0 \ldots, 0]^\top$. Also, $\mathbf{1}_{s^g(1)}$ means the vector with entry one at the index of the first state in the sequence $\mathcal{S}^g$. For example, if there are three possible states and a sequence of length three, $s^g(1) = 2, s^g(2) = 1, s^g(3) = 3$, we have $\mathbf{1}_{s^g(1)} = [0, 1, 0]^\top$.
- The terms in this log-likelihood are:

$$\log \pi_{s^g(1)} = \mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi}, \tag{15}$$

$$a_{s^g(t-1),s^g(t)} = \prod_{i=1}^{n} \prod_{j=1}^{n} (a_{i,j})^{\mathbf{1}_i[i]\,\mathbf{1}_j[j]}, \quad \to 0 \text{ or } 1 \to \begin{cases} i = s^g(t-1) \\ j = s^g(t) \end{cases}$$

$$\implies \log(a_{s^g(t-1),s^g(t)}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}_i[i]\,\mathbf{1}_j[j] \log(a_{i,j}) = \mathbf{1}_{s^g(t-1)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t)}, \tag{16}$$

Likelihood

- Also:

$$b_{s^g(t),o^g(t)} = \prod_{i=1}^{n}\prod_{j=1}^{n}(b_{i,j})^{\mathbf{1}_i[i]\,\mathbf{1}_j[j]},$$

$$\implies \log(b_{s^g(t),o^g(t)}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{1}_i[i]\,\mathbf{1}_j[j]\log(b_{i,j}) = \mathbf{1}_{s^g(t)}^{\top}(\log \boldsymbol{B})\mathbf{1}_{o^g(t)}, \tag{17}$$

  where $\mathbf{1}_i[i] = \mathbf{1}_j[j] = 1$.

- Hence, we can write the log-likelihood:

$$\ell = \log \pi_{s^g(1)} + \sum_{t=1}^{\tau-1}\log(a_{s^g(t),s^g(t+1)}) + \sum_{t=1}^{\tau}\log(b_{s^g(t),o^g(t)}),$$

  as:

$$\ell = \mathbf{1}_{s^g(1)}^{\top}\log \boldsymbol{\pi} + \sum_{t=1}^{\tau-1}\mathbf{1}_{s^g(t-1)}^{\top}(\log \boldsymbol{A})\mathbf{1}_{s^g(t)} + \sum_{t=1}^{\tau}\mathbf{1}_{s^g(t)}^{\top}(\log \boldsymbol{B})\mathbf{1}_{o^g(t)}. \tag{18}$$

E-step in EM

- We found:

$$\ell = \mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi} + \sum_{t=1}^{\tau-1} \mathbf{1}_{s^g(t-1)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t)} + \sum_{t=1}^{\tau} \mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{B}) \mathbf{1}_{o^g(t)}.$$

- The missing variables in the log-likelihood are $\mathbf{1}_{s^g(1)}$, $\mathbf{1}_{s^g(t-1)}$, $\mathbf{1}_{s^g(t)}$, and $\mathbf{1}_{o^g(t)}$. The expectation of the log-likelihood with respect to the missing variables is:

$$\begin{aligned} Q(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}) &= \mathbb{E}(\ell) \\ &= \mathbb{E}(\mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi}) + \sum_{t=1}^{\tau-1} \mathbb{E}\big(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t+1)}\big) + \sum_{t=1}^{\tau} \mathbb{E}\big(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{B}) \mathbf{1}_{o^g(t)}\big). \end{aligned} \tag{19}$$

M-step in EM

- We maximize the $Q(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ with respect to the parameters π_i, $a_{i,j}$, and $b_{i,j}$:

$$\begin{aligned} &\underset{x}{\text{maximize}} && Q(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}) \\ &\text{subject to} && \sum_{i=1}^{n} \pi_i = 1, \\ & && \sum_{j=1}^{n} a_{i,j} = 1, \quad \forall i \in \{1, \ldots, n\}, \\ & && \sum_{j=1}^{n} b_{i,j} = 1, \quad \forall i \in \{1, \ldots, n\}, \end{aligned} \tag{20}$$

  where the constraints ensure that the probabilities in the initial states, the transition matrix, and the emission matrix add to one.

- The Lagrangian [6] for this optimization problem is:

$$\begin{aligned} \mathcal{L} = &\mathbb{E}(\mathbf{1}_{sg(1)}^{\top} \log \boldsymbol{\pi}) + \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{sg(t)}^{\top} (\log \boldsymbol{A}) \mathbf{1}_{sg(t+1)}) \\ &+ \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{sg(t)}^{\top} (\log \boldsymbol{B}) \mathbf{1}_{og(t)}) - \eta_1 (\sum_{i=1}^{n} \pi_i - 1) - \eta_2 (\sum_{j=1}^{n} a_{i,j} - 1) - \eta_3 (\sum_{j=1}^{n} b_{i,j} - 1), \end{aligned} \tag{21}$$

  where η_1, η_2, and η_3 are the Lagrange multipliers.

M-step in EM

- We had:

$$\mathcal{L} = \mathbb{E}(\mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi}) + \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t+1)})$$

$$+ \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{B}) \mathbf{1}_{o^g(t)}) - \eta_1 (\sum_{i=1}^{n} \pi_i - 1) - \eta_2 (\sum_{j=1}^{n} a_{i,j} - 1) - \eta_3 (\sum_{j=1}^{n} b_{i,j} - 1).$$

- The first term in the Lagrangian is simplified to $\mathbb{E}(\mathbf{1}_{s^g(1)}[1] \log \pi_1 + \cdots + \mathbf{1}_{s^g(1)}[\tau] \log \pi_\tau) = \mathbb{E}(\mathbf{1}_{s^g(1)}[1] \log \pi_1) + \cdots + \mathbb{E}(\mathbf{1}_{s^g(1)}[\tau] \log \pi_\tau)$; therefore, we have:

$$\frac{\partial \mathcal{L}}{\partial \pi_i} = \mathbb{E}(\mathbf{1}_{s^g(1)}[i]) - \eta_1 \pi_i \overset{\text{set}}{=} 0 \implies \pi_i = \frac{1}{\eta_1} \mathbb{E}(\mathbf{1}_{s^g(1)}[i]). \tag{22}$$

$$\sum_{i=1}^{n} \pi_i = 1 \overset{(22)}{\implies} \frac{1}{\eta_1} (\mathbb{E}(\mathbf{1}_{s^g(1)}[1]) + \cdots + \mathbb{E}(\mathbf{1}_{s^g(1)}[i]) + \cdots + \mathbb{E}(\mathbf{1}_{s^g(1)}[n]))$$

$$= \frac{1}{\eta_1} (0 + \cdots + 1 + \cdots + 0) \overset{\text{set}}{=} 1 \implies \eta_1 = 1. \tag{23}$$

$$\therefore \quad \pi_i = \mathbb{E}(\mathbf{1}_{s^g(1)}[i]). \tag{24}$$

$s^g(1)$-th element

M-step in EM

- We had: $\mathcal{L} = \mathbb{E}(\mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi}) + \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t+1)}) + \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{B}) \mathbf{1}_{o^g(t)}) - \eta_1(\sum_{i=1}^n \pi_i - 1) - \eta_2(\sum_{j=1}^n a_{i,j} - 1) - \eta_3(\sum_{j=1}^n b_{i,j} - 1).$
- Similarly, we have:

$$\frac{\partial \mathcal{L}}{\partial a_{i,j}} = \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \mathbf{1}_{s^g(t+1)}[j]) - \eta_2 a_{i,j} \overset{\text{set}}{=} 0 \implies a_{i,j} = \frac{1}{\eta_2} \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \mathbf{1}_{s^g(t+1)}[j]). \tag{25}$$

$$\sum_{j=1}^{n} a_{i,j} = 1 \overset{(25)}{\implies} \frac{1}{\eta_2} \sum_{j=1}^{n} \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \mathbf{1}_{s^g(t+1)}[j]) =$$

$$= \frac{1}{\eta_2} \sum_{t=1}^{\tau-1} \Big(\mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 0) + \cdots + \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 1) + \cdots + \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 0) \Big)$$

$s^g(t+1)$-th element

$$= \frac{1}{\eta_2} \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]) \overset{\text{set}}{=} 1 \implies \eta_2 = \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]). \tag{26}$$

$$\therefore \quad a_{i,j} = \frac{\sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \mathbf{1}_{s^g(t+1)}[j])}{\sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i])}. \tag{27}$$

M-step in EM

- We had: $\mathcal{L} = \mathbb{E}(\mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi}) + \sum_{t=1}^{\tau-1} \mathbb{E}\big(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t+1)}\big) + \sum_{t=1}^{\tau} \mathbb{E}\big(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{B}) \mathbf{1}_{o^g(t)}\big) - \eta_1(\sum_{i=1}^n \pi_i - 1) - \eta_2(\sum_{j=1}^n a_{i,j} - 1) - \eta_3(\sum_{j=1}^n b_{i,j} - 1).$
- Likewise, we have:

$$\frac{\partial \mathcal{L}}{\partial b_{i,j}} = \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \mathbf{1}_{o^g(t)}[j]) - \eta_3 \, b_{i,j} \overset{\text{set}}{=} 0 \implies b_{i,j} = \frac{1}{\eta_3} \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \, \mathbf{1}_{o^g(t)}[j]). \tag{28}$$

$$\sum_{j=1}^{n} b_{i,j} = 1 \overset{(28)}{\implies} \frac{1}{\eta_3} \sum_{j=1}^{n} \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \mathbf{1}_{o^g(t)}[j]) =$$

$$= \frac{1}{\eta_3} \sum_{t=1}^{\tau} \Big(\mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 0) + \cdots + \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 1) + \cdots + \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 0) \Big)$$

$o^g(t)$-th element

$$= \frac{1}{\eta_3} \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]) \overset{\text{set}}{=} 1$$

$$\implies \eta_3 = \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]). \tag{29}$$

$$\therefore \quad b_{i,j} = \frac{\sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \, \mathbf{1}_{o^g(t)}[j])}{\sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i])}. \tag{30}$$

Evaluation in HMM

Evaluation in HMM

- **Evaluation** in HMM means the following [7, 8]: Given the observation sequence $\mathcal{O}^g = o^g(1), \ldots, o^g(\tau)$ and the HMM model $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$, we want to compute $\mathbb{P}(\mathcal{O}^g \mid \lambda)$, i.e., the probability of the generated observation sequence.
- In summary:

$$\mathcal{O}^g, \text{given: } \lambda \Longrightarrow \mathbb{P}(\mathcal{O}^g \mid \lambda) = ? \tag{31}$$

- Note that $\mathbb{P}(\mathcal{O}^g \mid \lambda)$ can also be denoted by $\mathbb{P}(\mathcal{O}^g; \lambda)$. The $\mathbb{P}(\mathcal{O}^g \mid \lambda)$ is sometimes referred to as the **likelihood**.

Direct Calculation

- Assume that the state sequence $\mathcal{S}^g = s^g(1), \ldots, s^g(\tau)$ has caused the observation sequence $\mathcal{O}^g = o^g(1), \ldots, o^g(\tau)$. Hence, we have:

$$\mathbb{P}(\mathcal{O}^g \mid \mathcal{S}^g, \lambda) = b_{s^g(1),o^g(1)} b_{s^g(2),o^g(2)} \cdots b_{s^g(\tau),o^g(\tau)}. \tag{32}$$

- On the other hand, the probability of the state sequence $\mathcal{S}^g = s^g(1), \ldots, s^g(\tau)$ is:

$$\mathbb{P}(\mathcal{S}^g \mid \lambda) = \pi_{s^g(1)} a_{s^g(1),s^g(2)} \cdots a_{s^g(\tau-1),s^g(\tau)}. \tag{33}$$

- According to chain rule, we have:

$$\mathbb{P}(\mathcal{O}^g, \mathcal{S}^g \mid \lambda) = \mathbb{P}(\mathcal{O}^g \mid \mathcal{S}^g, \lambda)\, \mathbb{P}(\mathcal{S}^g \mid \lambda) \tag{34}$$

$$\pi_{s^g(1)} b_{s^g(1),o^g(1)} a_{s^g(1),s^g(2)} \cdots b_{s^g(\tau),o^g(\tau)} a_{s^g(\tau-1),s^g(\tau)}, \tag{35}$$

  which is the probability of occurrence of both the observation sequence $\mathcal{O}^g$ and state sequence $\mathcal{S}^g$.

- Any state sequence may have caused the observation sequence $\mathcal{O}^g$. Therefore, according to the law of total probability, we have:

$$\mathbb{P}(\mathcal{O}^g \mid \lambda) = \sum_{\forall \mathcal{S}^g} \mathbb{P}(\mathcal{O}^g, \mathcal{S}^g \mid \lambda) \overset{(34)}{=} \sum_{\forall \mathcal{S}^g} \mathbb{P}(\mathcal{O}^g \mid \mathcal{S}^g, \lambda)\, \mathbb{P}(\mathcal{S}^g \mid \lambda)$$

$$\overset{(35)}{=} \sum_{\forall s^g(1)} \sum_{\forall s^g(2)} \cdots \sum_{\forall s^g(\tau)} \pi_{s^g(1)} b_{s^g(1),o^g(1)} a_{s^g(1),s^g(2)} b_{s^g(\tau),o^g(\tau)} a_{s^g(\tau-1),s^g(\tau)}, \tag{36}$$

  which means that we start with the first state, then output the first observation, and then go to the next state. This procedure is **repeated** until the last state. The summations are over **all possible states in the state sequence**.

Direct Calculation

- We found:

$$\mathbb{P}(\mathcal{O}^g \mid \lambda) = \sum_{\forall s^g(1)} \sum_{\forall s^g(2)} \cdots \sum_{\forall s^g(\tau)} \pi_{s^g(1)} \, b_{s^g(1),o^g(1)} \, a_{s^g(1),s^g(2)} \, b_{s^g(\tau),o^g(\tau)} \, a_{s^g(\tau-1),s^g(\tau)}.$$

- The **time complexity** of this direct calculation of $\mathbb{P}(\mathcal{O}^g \mid \lambda)$ is in the order of $O(2\tau n^\tau)$ because at every time clock $t \in \{1, \ldots, \tau\}$, there are n possible states to go through [7].
- Because of n^τ, this is very inefficient especially for long sequences (large τ).

The Forward-Backward Procedure

- A more efficient algorithm for **evaluation** in HMM is **forward-backward procedure** [7].
- The forward-backward procedure includes two stages, i.e., **forward and backward belief propagation stages**.
- **The Forward Belief Propagation:**
- Similar to the belief propagation procedure, we define the **forward message** until time t as:

$$\alpha_i(t) := \mathbb{P}\left(o^g(1), o^g(2), \ldots, o^g(t), s^g(t) = s_i \mid \lambda\right), \tag{37}$$

which is the probability of partial observation sequence until time t and being in state s_i at time t.

- Algorithm 2 shows the **forward belief propagation** from state one to the state τ.

Input: $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$
$\alpha_i(1) = \pi_i \, b_{i,o^g(1)}, \quad \forall i \in \{1, \ldots, n\}$
for *state* j *from* 1 *to* n **do**
  for *time* t *from* 1 *to* $(\tau - 1)$ **do**
    $\alpha_j(t+1) = \left[\sum_{i=1}^{n} \alpha_i(t) \, a_{i,j}\right] b_{j,o^g(t+1)}$
$\mathbb{P}(\mathcal{O}^g \mid \lambda) = \sum_{i=1}^{n} \alpha_i(\tau)$
Return $\mathbb{P}(\mathcal{O}^g \mid \lambda), \forall i, \forall t : \alpha_i(t)$

Algorithm 2: The forward belief propagation in the forward-backward procedure

The Forward-Backward Procedure

Input: $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$
$\alpha_i(1) = \pi_i \, b_{i,o^g(1)}, \quad \forall i \in \{1, \ldots, n\}$
for *state* j *from* 1 *to* n **do**
  for *time* t *from* 1 *to* $(\tau - 1)$ **do**
    $\alpha_j(t+1) = \left[\sum_{i=1}^{n} \alpha_i(t) \, a_{i,j}\right] b_{j,o^g(t+1)}$
$\mathbb{P}(\mathcal{O}^g \mid \lambda) = \sum_{i=1}^{n} \alpha_i(\tau)$
Return $\mathbb{P}(\mathcal{O}^g \mid \lambda), \forall i, \forall t : \alpha_i(t)$

Algorithm 2: The forward belief propagation in the forward-backward procedure

- In this algorithm, $\alpha_i(t)$ is solved inductively. The initial forward message is:

$$\alpha_i(1) = \pi_i b_{i,o^g(1)}, \quad \forall i \in \{1, \ldots, n\}, \tag{38}$$

  which is the probability of occurrence of the initial state s_i and the observation symbol $o^g(1)$.
- The next forward messages are calculated as:

$$\alpha_j(t+1) = \left[\sum_{i=1}^{n} \alpha_i(t) a_{i,j}\right] b_{j,o^g(t+1)}, \tag{39}$$

  which is the probability of occurrence of observation sequence $o^g(1), \ldots, o^g(t)$, being in state s_i at time t, going to state j at time $t+1$, and the observation symbol $o^g(t+1)$.
- The summation is because, at time t, the state $s^g(t)$ can be any state so we should use the law of total probability.

The Forward-Backward Procedure

Input: $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$
$\alpha_i(1) = \pi_i \, b_{i,o^g(1)}, \quad \forall i \in \{1, \ldots, n\}$
for *state* j *from* 1 *to* n **do**
  for *time* t *from* 1 *to* $(\tau - 1)$ **do**
    $\alpha_j(t+1) = \left[\sum_{i=1}^{n} \alpha_i(t) \, a_{i,j} \right] b_{j,o^g(t+1)}$
$\mathbb{P}(\mathcal{O}^g \mid \lambda) = \sum_{i=1}^{n} \alpha_i(\tau)$
Return $\mathbb{P}(\mathcal{O}^g \mid \lambda), \forall i, \forall t : \alpha_i(t)$

Algorithm 2: The forward belief propagation in the forward-backward procedure

- Finally, using the law of total probability, we have:

$$\mathbb{P}(\mathcal{O}^g \mid \lambda) = \sum_{i=1}^{n} \mathbb{P}(\mathcal{O}^g, s^g(\tau) = s_i \mid \lambda) = \sum_{i=1}^{n} \alpha_i(\tau), \tag{40}$$

which is the desired probability in the evaluation for HMM. Hence, the forward belief propagation suffices for evaluation.

The Forward-Backward Procedure

- **The Backward Belief Propagation**:
- Again similar to the belief propagation procedure, we define the **backward message** since time τ to $t+1$ as:

$$\beta_i(t) := \mathbb{P}\big(o^g(t+1), o^g(t+2), \ldots, o^g(\tau) \,|\, s^g(t) = s_i, \lambda\big), \tag{41}$$

  which is the probability of partial observation sequence from $t+1$ until the end time τ given being in state s_i at time t.
- Algorithm 3 shows the backward belief propagation.

Input: $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$
$\beta_i(\tau) = 1, \quad \forall i \in \{1, \ldots, n\}$
for *state* i *from* 1 *to* n **do**
  for *time* t *from* $(\tau - 1)$ *to* 1 **do**
    $\beta_i(t) = \sum_{j=1}^{n} a_{i,j}\, b_{j,o^g(t+1)}$
Return $\forall i, \forall t : \beta_i(t)$

Algorithm 3: The backward belief propagation in the forward-backward procedure

The Forward-Backward Procedure

Input: $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$
$\beta_i(\tau) = 1, \quad \forall i \in \{1, \ldots, n\}$
for *state* i *from* 1 *to* n **do**
  for *time* t *from* $(\tau - 1)$ *to* 1 **do**
    $\beta_i(t) = \sum_{j=1}^{n} a_{i,j}\, b_{j,o^g(t+1)}$
Return $\forall i, \forall t : \beta_i(t)$

Algorithm 3: The backward belief propagation in the forward-backward procedure

- In this algorithm, the initial backward message is:

$$\beta_i(\tau) = 1, \quad \forall i \in \{1, \ldots, n\}. \tag{42}$$

- The next backward messages are calculated as:

$$\beta_i(t) = \sum_{j=1}^{n} a_{i,j}\, b_{j,o^g(t+1)}, \tag{43}$$

which is the probability of being in state s_i at time t, going to state j at time $t+1$, and the observation symbol $o^g(t+1)$.

- The summation is because, at time $t+1$, the state $s^g(t+1)$ can be any state so we should use the law of total probability.

The Forward-Backward Procedure

Input: $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$
$\beta_i(\tau) = 1, \quad \forall i \in \{1, \ldots, n\}$
for *state* i *from* 1 *to* n **do**
  for *time* t *from* $(\tau - 1)$ *to* 1 **do**
    $\beta_i(t) = \sum_{j=1}^{n} a_{i,j}\, b_{j,o^g(t+1)}$
Return $\forall i, \forall t : \beta_i(t)$

Algorithm 3: The backward belief propagation in the forward-backward procedure

- It is noteworthy that for very long sequences, the $\alpha_i(t)$ and $\beta(t)$ become extremely small, recursively (see Algorithms 2 and 3). Hence, some people normalize them at every iteration of algorithm [5]:

$$\alpha_i(t) \leftarrow \frac{\alpha_i(t)}{\sum_{j=1}^{\tau} \alpha_j(t)}, \tag{44}$$

$$\beta_i(t) \leftarrow \frac{\beta_i(t)}{\sum_{j=1}^{\tau} \beta_j(t)}, \tag{45}$$

in order to sum to one. However, note that if this normalization is done, we will have $\mathbb{P}(\mathcal{O}^g \mid \lambda) = 1$.

Estimation in HMM

Estimation in HMM

- **Estimation in HMM** means the following [7, 8]: Given the observation sequence $\mathcal{O}^g = o_1^g, \ldots, o_\tau^g$ and the HMM model $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$, we want to compute or estimate $\mathbb{P}(\mathcal{S}^g \mid \mathcal{O}^g, \lambda)$, i.e., the probability of a state sequence given an observation sequence.
- In summary:

$$\mathcal{S}^g, \text{given: } \mathcal{O}^g, \lambda \implies \mathbb{P}(\mathcal{S}^g \mid \mathcal{O}^g, \lambda) = ? \tag{46}$$

Greedy Approach

- Let the probability of being in state s_i at time t given the observation sequence $\mathcal{O}^g$ and the HMM model λ be denoted by:

$$\gamma_i(t) := \mathbb{P}\big(s^g(t) = s_i \mid \mathcal{O}^g, \lambda\big). \tag{47}$$

- We can say:

$$\gamma_i(t)\,\mathbb{P}(\mathcal{O}^g \mid \lambda) = \mathbb{P}\big(s^g(t) = s_i \mid \mathcal{O}^g, \lambda\big)\mathbb{P}(\mathcal{O}^g \mid \lambda)$$
$$\overset{(a)}{=} \mathbb{P}\big(s^g(t) = s_i, \mathcal{O}^g, \lambda\big) \overset{(b)}{=} \alpha_i(t)\,\beta_i(t)$$

$$\Longrightarrow \gamma_i(t) = \frac{\alpha_i(t)\,\beta_i(t)}{\mathbb{P}(\mathcal{O}^g \mid \lambda)} \tag{48}$$

$$= \frac{\alpha_i(t)\,\beta_i(t)}{\sum_{j=1}^{n} \alpha_j(t)\,\beta_j(t)}, \tag{49}$$

- where (a) is because of chain rule in probability and (b) is because:

$$\alpha_i(t)\,\beta_i(t) \overset{(37),(41)}{=} \mathbb{P}\big(o^g(1), o^g(2), \ldots, o^g(t), s^g(t) = s_i \mid \lambda\big)$$
$$\times \mathbb{P}\big(o^g(t+1), o^g(t+2), \ldots, o^g(\tau) \mid s^g(t) = s_i, \lambda\big)$$
$$= \mathbb{P}\big(o^g(1), o^g(2), \ldots, o^g(\tau), s^g(t) = s_i, \lambda\big) = \mathbb{P}\big(\mathcal{O}^g, s^g(t) = s_i, \lambda\big).$$

- The reason of (b) can also be interpreted in this way: it is because of the forward-backward procedure which states the **belief over a variable as product of the forward and backward messages**.

Greedy Approach

- In the **greedy approach**, at every time t, we select a state with **maximum probability of occurrence without considering the other states in the sequence**. Therefore, we have:

$$s^g(t) = \arg \max_{1 \leq i \leq n} \gamma_i(t), \quad \forall t \in \{1, \ldots, \tau\}, \tag{50}$$

The Viterbi Algorithm

- The greedy approach does **not optimize over the whole path** but **greedily chooses the best state at every time step**.
- Another approach is to find the **best state sequence which has the highest probability of occurrence**, i.e., maximizing $\mathbb{P}(\mathcal{O}^g, \mathcal{S}^g \mid \lambda)$ [7]. The **Viterbi algorithm** (1967) [9, 10] can be used to find this path of states [11].
- Different works, such as [12], have worked on using Viterbi algorithm for HMM.
- Algorithm 4 shows the Viterbi algorithm for estimation in HMM [7].

Input: $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$
// Initialization:
$\delta_i(1) = \pi_i \, b_{i,o^g(1)}, \quad \forall i \in \{1, \ldots, n\}$
$\psi_i(1) = 0, \quad \forall i \in \{1, \ldots, n\}$
// Recursion:
for *state* j *from* 1 *to* n **do**
  for *time* t *from* 2 *to* τ **do**
    $\delta_j(t) = \max_{1 \le i \le n} \left(\delta_i(t-1)\, a_{i,j}\right) b_{j,o^g(t)}$
    $\psi_j(t) = \arg\max_{1 \le i \le n} \left(\delta_i(t-1)\, a_{i,j}\right)$
// Termination:
$p^* = \max_{1 \le i \le n} \delta_i(\tau)$
$s^*(\tau) = \arg\max_{1 \le i \le n} \delta_i(\tau)$
// Backtracking:
for *time* t *from* $\tau - 1$ *to* 1 **do**
  $s^*(t) = \psi_{s^*(t+1)}(t+1)$
$\mathbb{P}(\mathcal{O}^g, \mathcal{S}^g \mid \lambda) = p^*$
Return $\mathbb{P}(\mathcal{O}^g, \mathcal{S}^g \mid \lambda), \mathcal{S}^g = s^*(1), \ldots, s^*(\tau)$

Algorithm 4: The Viterbi algorithm for estimation in HMM

The Viterbi Algorithm

- In this algorithm, we have variable $\delta_j(t)$:

$$\delta_j(t) = \max_{1 \le i \le n} \left[\delta_i(t-1)\, a_{i,j} \right] b_{j, o^g(t)}, \tag{51}$$

which is similar to $\alpha_j(t)$ defined in Eq. (39):

$$\alpha_j(t+1) = \left[\sum_{i=1}^{n} \alpha_i(t)\, a_{i,j} \right] b_{j, o^g(t+1)},$$

except that $\alpha_j(t)$ in the forward belief propagation uses **sum-product algorithm** [13] while $\delta_j(t)$ in the Viterbi algorithm uses **max-product algorithm** [14, 15].

The Viterbi Algorithm

- Similar to Eq. (38):

$$\alpha_i(1) = \pi_i \, b_{i,o^g(1)}, \quad \forall i \in \{1, \ldots, n\},$$

the initial $\delta_j(t)$ is:

$$\delta_i(1) = \pi_i \, b_{i,o^g(1)}, \quad \forall i \in \{1, \ldots, n\}. \tag{52}$$

- We define the index maximizing in Eq. (51):

$$\delta_j(t) = \max_{1 \le i \le n} \left[\delta_i(t-1) \, a_{i,j} \right] b_{j,o^g(t)},$$

as:

$$\psi_j(t) = \arg \max_{1 \le i \le n} \left(\delta_i(t-1) \, a_{i,j} \right). \tag{53}$$

- Then, a backward analysis is done starting from the end of state sequence:

$$p^* = \max_{1 \le i \le n} \delta_i(\tau), \tag{54}$$

$$s^*(\tau) = \arg \max_{1 \le i \le n} \delta_i(\tau), \tag{55}$$

- and the other states in the sequence are backtracked as:

$$s^*(t) = \psi_{s^*(t+1)}(t+1). \tag{56}$$

- The states $\mathcal{S}^g = s^*(1), s^*(2), \ldots, s^*(\tau)$ are the desired state sequence in the estimation.
- Therefore, the states in the state sequence are maximizing the forward belief propagation in a **max-product** setting.

The Viterbi Algorithm

- The probability of this path of states with **maximum probability of occurrence** is:

$$\mathbb{P}(\mathcal{O}^g, \mathcal{S}^g \mid \lambda) = p^*. \tag{57}$$

- Note that the Viterbi algorithm can be visualized using a **trellis structure** (see Appendix A in [16]).

Training the HMM

Training the HMM

- **Training HMM** means the following [7, 8]: Given the observation sequence $\mathcal{O}^g = o_1^g, \ldots, o_T^g$, we want to adjust the HMM model parameters $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ in order to maximize $\mathbb{P}(\mathcal{O}^g \mid \lambda)$.
- In summary:

$$\text{given: } \mathcal{O}^g, \mathcal{O}, \mathcal{S} \implies \lambda = \arg\max_{\lambda} \mathbb{P}(\mathcal{O}^g \mid \lambda). \tag{58}$$

The Baum-Welch Algorithm

- We can solve for Eq. (58):

$$\text{given: } \mathcal{O}^g, \mathcal{O}, \mathcal{S} \implies \lambda = \arg\max_{\lambda} \mathbb{P}(\mathcal{O}^g \mid \lambda),$$

  using **maximum likelihood estimation** using **Expectation Maximization (EM)**.
- The **Baum-Welch algorithm** (1970) [17] is the most well-known method for training HMM. It makes use of the EM results.
- We define the probability of occurrence of a path being in states s_i and s_j, respectively, at times t and $t+1$ by:

$$\xi_{i,j}(t) := \mathbb{P}\big(s^g(t) = s_i, s^g(t+1) = s_j \mid \mathcal{O}^g, \lambda\big). \tag{59}$$

- We can say:

$$\xi_{i,j}(t)\,\mathbb{P}(\mathcal{O}^g \mid \lambda) = \mathbb{P}\big(s^g(t) = s_i, s^g(t+1) = s_j \mid \mathcal{O}^g, \lambda\big)\mathbb{P}(\mathcal{O}^g \mid \lambda)$$

$$\overset{(a)}{=} \mathbb{P}\big(s^g(t) = s_i, s^g(t+1) = s_j, \mathcal{O}^g, \lambda\big) \overset{(b)}{=} \alpha_i(t)\, a_{i,j}\, b_{j,o^g(t+1)}\, \beta_j(t+1)$$

$$\implies \xi_{i,j}(t) = \frac{\alpha_i(t)\, a_{i,j}\, b_{j,o^g(t+1)}\, \beta_j(t+1)}{\mathbb{P}(\mathcal{O}^g \mid \lambda)} \tag{60}$$

$$= \frac{\alpha_i(t)\, a_{i,j}\, b_{j,o^g(t+1)}\, \beta_j(t+1)}{\sum_{r=1}^{n}\sum_{\ell=1}^{n} \alpha_r(t)\, a_{r,\ell}\, b_{\ell,o^g(t+1)}\, \beta_\ell(t+1)}, \tag{61}$$

where (a) is because of chain rule in probability and the reason of (b) is in the next slide.

The Baum-Welch Algorithm

- We found:

$$\xi_{i,j}(t)\,\mathbb{P}(\mathcal{O}^g \mid \lambda) = \mathbb{P}\big(s^g(t) = s_i, s^g(t+1) = s_j \mid \mathcal{O}^g, \lambda\big)\mathbb{P}(\mathcal{O}^g \mid \lambda)$$
$$\overset{(a)}{=} \mathbb{P}\big(s^g(t) = s_i, s^g(t+1) = s_j, \mathcal{O}^g, \lambda\big) \overset{(b)}{=} \alpha_i(t)\, a_{i,j}\, b_{j,o^g(t+1)}\, \beta_j(t+1)$$

- (b) is because of the following: According to Eqs. (37), (4), (6), and (41), we have:

$$\alpha_i(t) = \mathbb{P}(o^g(1), \ldots, o^g(t), s^g(t) = s_i \mid \lambda),$$
$$a_{i,j} = \mathbb{P}(s_j(t+1) \mid s_i(t)),$$
$$b_{j,o^g(t+1)} = \mathbb{P}(o^g(t+1) \mid s_j(t+1)),$$
$$\beta_j(t+1) = \mathbb{P}(o^g(t+2), \ldots, o^g(\tau) \mid s^g(t+1) = s_j, \lambda).$$

Therefore, we have:

$$\alpha_i(t)\, a_{i,j}(t)\, b_{j,o^g(t+1)}\, \beta_j(t+1) = \mathbb{P}\big(s^g(t) = s_i, s^g(t+1) = s_j, \mathcal{O}^g, \lambda\big).$$

- Note that we have $\sum_{i=1}^{n}\sum_{j=1}^{n}\xi_{i,j}(t) = 1$. Also, note that $\mathbb{P}(\mathcal{O}^g \mid \lambda)$ in Eq. (60) can be obtained from either the denominator of Eq. (61) or line 6 in Algorithm 2 (i.e., Eq. (40)):

$$\mathbb{P}(\mathcal{O}^g \mid \lambda) = \sum_{i=1}^{n}\mathbb{P}\big(\mathcal{O}^g, s^g(\tau) = s_i \mid \lambda\big) = \sum_{i=1}^{n}\alpha_i(\tau).$$

The Baum-Welch Algorithm

- In Eq. (60):

$$\xi_{i,j}(t) = \frac{\alpha_i(t)\, a_{i,j}\, b_{j,o^g(t+1)}\, \beta_j(t+1)}{\mathbb{P}(\mathcal{O}^g \mid \lambda)},$$

the terms $\alpha_i(t)$, $a_{i,j}(t)$, $b_{j,o^g(t+1)}$, and $\beta_j(t+1)$ stand for the probability of the first t observations ending in state s_i at time t, the probability of transitioning from state s_i (at time t) to state s_j (at time $t+1$), the probability of observing $o^g(t+1)$ from state s_j at time $t+1$, and the probability of the remainder of the observation sequence, respectively.

The Baum-Welch Algorithm

- Now, recall the Eqs. (24), (27), and (30) from EM algorithm for HMM:

$$\pi_i = \mathbb{E}(\mathbf{1}_{s^g(1)}[i]), \quad a_{i,j} = \frac{\sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\,\mathbf{1}_{s^g(t+1)}[j])}{\sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i])}, \quad b_{i,j} = \frac{\sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\,\mathbf{1}_{o^g(t)}[j])}{\sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i])}.$$

- On the other hand, recall Eqs. (47) and (59):

$$\gamma_i(t) = \mathbb{P}\big(s^g(t) = s_i \,|\, \mathcal{O}^g, \lambda\big), \quad \xi_{i,j}(t) = \mathbb{P}\big(s^g(t) = s_i, s^g(t+1) = s_j \,|\, \mathcal{O}^g, \lambda\big).$$

- Therefore, we can say:

$$\begin{cases} \mathbb{E}(\mathbf{1}_{s^g(1)}[i]) = \gamma_i(1), & (62) \\ \mathbb{E}(\mathbf{1}_{s^g(t)}[i]) = \gamma_i(t), & (63) \\ \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\,\mathbf{1}_{s^g(t+1)}[j]) = \xi_{i,j}(t), & (64) \\ \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\,\mathbf{1}_{o^g(t)}[j]) = \gamma_i(t), \text{ where } o^g(t) = j \text{ in } \gamma_i(t). & (65) \end{cases}$$

Hence:

$$\begin{cases} \pi_i = \gamma_i(1) \overset{(47)}{=} \mathbb{P}\big(s^g(1) = s_i \,|\, \mathcal{O}^g, \lambda\big), \quad \forall i \in \{1, \dots, n\}, & (66) \\ a_{i,j} = \dfrac{\sum_{t=1}^{\tau-1} \xi_{i,j}(t)}{\sum_{t=1}^{\tau-1} \gamma_i(t)}, \quad \forall i, j \in \{1, \dots, n\}, & (67) \\ b_{i,j} = \dfrac{\sum_{t=1,\, o^g(t)=j}^{\tau} \gamma_i(t)}{\sum_{t=1}^{\tau} \gamma_i(t)}, \quad \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\}. \end{cases}$$

The Baum-Welch Algorithm

- We found:

$$b_{i,j} = \frac{\sum_{t=1,\, o^g(t)=j}^{\tau} \gamma_i(t)}{\sum_{t=1}^{\tau} \gamma_i(t)}, \quad \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, m\}.$$

- With change of variable, we have:

$$b_{j,k} = \frac{\sum_{t=1,\, o^g(t)=k}^{\tau} \gamma_j(t)}{\sum_{t=1}^{\tau} \gamma_j(t)}, \quad \forall j \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\}. \tag{68}$$

The Baum-Welch Algorithm

- The algorithm is shown in **Algorithm 5** [7]. In this algorithm the initial probabilities of being in state s_i at time $t = 1$ is according to Eq. (66). Then the $a_{i,j}$ is then calculated using Eq. (67). According to counting in probability, it can also be interpreted as the ratio of the expected number of transitions from state s_i to s_j over the expected number of transitions out of state s_i. Finally, the $b_{j,k}$ is calculated using Eq. (68). Likewise, using counting in probability, the $b_{j,k}$ can be interpreted as the ratio of the expected number of times being in state s_j and seeing observation o_k over the expected number of times being in state s_j.

Input: $\gamma_i(t), \xi_{i,j}(t), \forall i, \forall j, \forall t$
$\pi_i = \gamma_i(1), \quad \forall i \in \{1, \ldots, n\}$
for *state* i *from* 1 *to* n **do**
  for *state* j *from* 1 *to* n **do**
    $a_{i,j} = \sum_{t=1}^{\tau-1} \xi_{i,j}(t) / \sum_{t=1}^{\tau-1} \gamma_i(t)$
for *state* j *from* 1 *to* n **do**
  for *observation* k *from* 1 *to* m **do**
    $b_{j,k} = \sum_{t=1,\, o^g(t)=k}^{\tau} \gamma_j(t) / \sum_{t=1}^{\tau} \gamma_j(t)$
// Normalization, for computer error corrections:
$\pi_i \leftarrow \pi_i / \sum_{j=1}^{n} \pi_j$
$a_{i,j} \leftarrow a_{i,j} / \sum_{\ell=1}^{n} a_{i,\ell}$
$b_{j,k} \leftarrow b_{j,k} / \sum_{\ell=1}^{m} b_{j,\ell}$
$\boldsymbol{\pi} = [\pi_1, \ldots, \pi_n]^\top, \boldsymbol{A} = [a_{i,j}], \boldsymbol{B} = [b_{j,k}], \quad \forall i, j \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\}$
Return $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$

Algorithm 5: The Baum-Welch algorithm for training the HMM

The Baum-Welch Algorithm

- For calculating Eq. (68):

$$b_{j,k} = \frac{\sum_{t=1,\, o^g(t)=k}^{\tau} \gamma_j(t)}{\sum_{t=1}^{\tau} \gamma_j(t)}, \quad \forall j \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\},$$

we use Eq. (49):

$$\gamma_i(t) = \frac{\alpha_i(t)\, \beta_i(t)}{\sum_{j=1}^{n} \alpha_j(t)\, \beta_j(t)},$$

where:

$$\alpha_j(t+1) = \Big[\sum_{i=1}^{n} \alpha_i(t)\, a_{i,j} \Big]\, b_{j,k}, \tag{69}$$

$$\beta_i(t) = \sum_{j=1}^{n} a_{i,j}\, b_{j,k}, \tag{70}$$

in line 5 in Algorithm 2 and in line 5 in Algorithm 3, respectively. We use the obtained $\alpha_i, \forall i, \forall t$ and $\beta_i(t), \forall i, \forall t$ for calculating Eq. (49).

Applications of HMM

Application in Speech Recognition

- Assume we have a dictionary of words consisting of $|\mathcal{W}|$ words.
- For every word indexed by $w \in \{1, \ldots, |\mathcal{W}|\}$, we have $|\mathcal{Q}_w|$ training instances spoken by one or several people. The training instances for a word are indexed by q where $q \in \{1, \ldots, |\mathcal{Q}_w|\}$.
- Every training instance is a sequence of observation symbols obtained from formants [18].
- We consider **an HMM model for every word** in the dictionary.
- **Training** the HMMs are as [7, 8]:
  1. For every word w_i, consider the training sequences, $\mathcal{O}_w^g$, indexed by q, i.e., $o_1^g, \ldots, o_{|\mathcal{Q}_w|}^g$.
  2. Train the HMM for the w-th word using Algorithm 6 to obtain λ_w.
- For an unknown **test** word with sequence $\mathcal{O}_t^g = o_1^g, \ldots, o_{|\mathcal{Q}_t|}^g$, we recognize the word as [7, 8]:
  1. Calculate $\mathbb{P}(\mathcal{O}_t^g \mid \lambda_w)$ for all $w \in \{1, \ldots, |\mathcal{W}|\}$ using the forward belief propagation, i.e., Algorithm 2 (see Eq. (40)).
  2. The test word is recognized as:

$$w^* = \arg\max_w \mathbb{P}(\mathcal{O}_t^g \mid \lambda_w). \tag{71}$$

So, the test word is recognized as the w-th word in the dictionary.

Application in Speech Recognition

- In **test phase for speech recognition**, usually, the **Viterbi algorithm** is used [8]. Hence, **another approach** for recognition of the test word $\mathcal{O}_t^g$ is:
  1. Calculate $\mathbb{P}(\mathcal{O}_t^g, \mathcal{S}_t^g \mid \lambda_w)$ for all $w \in \{1, \ldots, |\mathcal{W}|\}$ using the Viterbi algorithm, i.e., Algorithm 4 (see Eq. (57)).
  2. The test word is recognized as:

$$w^* = \arg\max_w \mathbb{P}(\mathcal{O}_t^g, \mathcal{S}_t^g \mid \lambda_w). \tag{72}$$

  So, the test word is recognized as the w-th word in the dictionary.

- Note that the words are pronounced with **different lengths (fast or slowly)** by different people. As **HMM is robust to different repetitions of states**, the recognition of words with different pacing is possible.

Application in Action Recognition

- In action recognition, every action can be seen as a **sequence of poses** where every pose may be **repeated** for several frames [19]. Hence, HMM can be used for action recognition [20].
- Assume we have a **set of actions** denoted by $\mathcal{W}$ where the actions are indexed by $w \in \{1, \ldots, |\mathcal{W}|\}$.
- We have $|\mathcal{Q}_w|$ training instances for every action where the training instances are indexed by $q \in \{1, \ldots, |\mathcal{Q}_w|\}$.
- Every training instance is a **sequence of observation symbols** where the symbols are the **poses**, e.g., sitting, standing, etc.
- **An HMM is trained for every action** [19, 21].
- Training and testing HMMs for action recognition is the same as training and test phases explained for speech recognition.
- The actions are performed with **different sequence lengths (fast or slowly)** by different people. As **HMM is robust to different repetitions of states**, the recognition of actions with different pacing is possible.

Application in Action Recognition

- In action recognition, we have a dataset of actions consisting of several defined poses [19, 21]. For example, if the dataset includes three actions **sit, stand, and turn**, the format of actions is as follows:
  - Action sit: $\underbrace{\text{stand}, \text{stand} \ldots, \text{stand}}_{\text{stand}}, \underbrace{\text{sit}, \text{sit} \ldots, \text{sit}}_{\text{sit}}$
  - Action stand: $\underbrace{\text{sit}, \text{sit} \ldots, \text{sit}}_{\text{sit}}, \underbrace{\text{stand}, \text{stand} \ldots, \text{stand}}_{\text{stand}}$
  - Action turn: $\underbrace{\text{stand}, \text{stand} \ldots, \text{stand}}_{\text{stand}}, \underbrace{\text{tilt}, \text{tilt} \ldots, \text{tilt}}_{\text{tilt}}$

  where the actions are modeled as **sequences of some poses, i.e., stand, sit, and tilt**.
- The actions can have **different lengths or pacing**.
- An example **training dataset** with its instances is shown in Table 1. In some sequences of dataset, there are some **noisy poses in the middle of sequences** of correct poses for making a difficult instance.
- An example **test dataset** is also shown in Table 2. The three test sequences are **different from the training sequences** to check the **generalizability** of the HMM models.
- Three HMM models can be trained for the three actions in this dataset and then, the test action sequences can be fed to the HMM models to be recognized.
- See our paper "**Fisherposes** for human action recognition using Kinect sensor data" (2017) [19].

Application in Action Recognition

Action	Sequence	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$	$t=11$	$t=12$
Sit	1	stand	stand	stand	sit	sit	sit	×	×	×	×	×	×
	2	stand	stand	sit	sit	sit	sit	sit	sit	×	×	×	×
	3	stand	stand	stand	stand	stand	sit	sit	×	×	×	×	×
	4	stand	stand	stand	stand	stand	sit	sit	sit	sit	sit	×	×
	5	stand	stand	stand	stand	stand	sit	sit	sit	sit	stand	sit	sit
Stand	1	sit	sit	sit	stand	stand	stand	×	×	×	×	×	×
	2	sit	sit	sit	sit	sit	sit	stand	stand	×	×	×	×
	3	sit	sit	stand	stand	stand	stand	stand	×	×	×	×	×
	4	sit	sit	sit	sit	stand	stand	stand	stand	stand	×	×	×
	5	sit	sit	sit	sit	stand	stand	stand	sit	stand	stand	stand	×
Turn	1	stand	stand	stand	tilt	tilt	tilt	×	×	×	×	×	×
	2	stand	stand	tilt	tilt	tilt	tilt	tilt	tilt	×	×	×	×
	3	stand	stand	stand	stand	stand	tilt	tilt	×	×	×	×	×
	4	stand	stand	stand	stand	tilt	tilt	tilt	tilt	tilt	tilt	×	×
	5	stand	stand	stand	stand	tilt	tilt	tilt	stand	tilt	tilt	tilt	×

Table 1. An example training dataset for action recognition

Action	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$	$t=11$
Sit	stand	stand	stand	sit	sit	sit	sit	×	×	×	×
Stand	sit	sit	sit	sit	stand	stand	stand	×	×	×	×
Turn	stand	stand	stand	stand	tilt	tilt	stand	stand	tilt	tilt	tilt

Table 2. An example test dataset for action recognition

Acknowledgment

- Some slides are based on our tutorial paper: "Hidden Markov Model: Tutorial" [22]
- For more information on HMM, refer to our tutorial paper [22].
- Some slides of this slide deck are inspired by teachings of Prof. Mehdi Molkaraie, Prof. Kevin Granville, and Prof. Mu Zhu at University of Waterloo, Department of Statistics.
- HMM in sklearn: `https://scikit-learn.sourceforge.net/stable/modules/hmm.html`

References

[1] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[2] S. M. Ross, *Introduction to probability models*. Academic press, 2014.

[3] G. F. Lawler, *Introduction to stochastic processes*. Chapman and Hall/CRC, 2018.

[4] T. L. Booth, *Sequential machines and automata theory*. New York, NY: Wiley, 1967.

[5] Z. Ghahramani, "An introduction to hidden Markov models and Bayesian networks," in *Hidden Markov models: applications in computer vision*, pp. 9–41, World Scientific, 2001.

[6] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[7] L. R. Rabiner and B.-H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[8] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[9] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

References (cont.)

[10] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[11] P. Blunsom, "Hidden Markov models," tech. rep., 2004.

[12] Y. He, "Extended Viterbi algorithm for second order hidden markov process," in *[1988 Proceedings] 9th International Conference on Pattern Recognition*, pp. 718–720, IEEE, 1988.

[13] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[14] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 736–744, 2001.

[15] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[16] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing*. Pearson Prentice Hall, 2019.

[17] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

References (cont.)

[18] I. R. Titze and D. W. Martin, *Principles of voice production*. Acoustical Society of America, 1998.

[19] B. Ghojogh, H. Mohammadzade, and M. Mokari, "Fisherposes for human action recognition using kinect sensor data," *IEEE Sensors Journal*, vol. 18, no. 4, pp. 1612–1627, 2017.

[20] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden Markov model," in *Proceedings 1992 IEEE Computer Society conference on computer vision and pattern recognition*, pp. 379–385, IEEE, 1992.

[21] M. Mokari, H. Mohammadzade, and B. Ghojogh, "Recognizing involuntary actions from 3d skeleton data using body states," *Scientia Iranica*, 2018.

[22] B. Ghojogh, F. Karray, and M. Crowley, "Hidden Markov model: Tutorial," 2019.