Hidden Markov Model: Tutorial

Benyamin Ghojogh BGHOJOGH@UWATERLOO.CA
Department of Electrical and Computer Engineering,
Machine Learning Laboratory, University of Waterloo, Waterloo, ON, Canada

Fakhri Karray KARRAY@UWATERLOO.CA
Department of Electrical and Computer Engineering,
Centre for Pattern Analysis and Machine Intelligence, University of Waterloo, Waterloo, ON, Canada

Mark Crowley MCROWLEY@UWATERLOO.CA
Department of Electrical and Computer Engineering,
Machine Learning Laboratory, University of Waterloo, Waterloo, ON, Canada

Abstract

This is a tutorial paper for Hidden Markov Model (HMM). First, we briefly review the background on Expectation Maximization (EM), Lagrange multiplier, factor graph, the sum-product algorithm, the max-product algorithm, and belief propagation by the forward-backward procedure. Then, we introduce probabilistic graphical models including Markov random field and Bayesian network. Markov property and Discrete Time Markov Chain (DTMC) are also introduced. We, then, explain likelihood estimation and EM in HMM in technical details. We explain evaluation in HMM where direct calculation and the forward-backward belief propagation are both explained. Afterwards, estimation in HMM is covered where both the greedy approach and the Viterbi algorithm are detailed. Then, we explain how to train HMM using EM and the Baum-Welch algorithm. We also explain how to use HMM in some applications such as speech and action recognition.

1. Introduction

Assume we have a time series of hidden random variables. We name the hidden random variables as states (Ghahramani, 2001). Every hidden random variable can generate (emit) an observation symbol or output. Therefore, we have a set of states and a set of observation symbols. A Hidden Markov Model (HMM) has some hidden states and some emitted observation symbols (Rabiner, 1989). HMM models the the probability density of a sequence of observed symbols (Roweis, 2003).

This paper is a technical tutorial for evaluation, estimation, and training in HMM. We also explain some applications for HMM. There exist many different applications for HMM. One of the most famous applications is in speech recognition (Rabiner, 1989; Gales et al., 2008) where an HMM is trained for every word in the speech (Rabiner & Juang, 1986). Another application of HMM is in action recognition (Yamato et al., 1992) because an action can be seen as a sequence or times series of poses (Ghojogh et al., 2017; Mokari et al., 2018). An interesting application of HMM, which is not trivial in the first glance, is in face recognition (Samaria, 1994). The face can be seen as a sequence of organs being modeled by HMM (Nefian & Hayes, 1998). Usage of HMM in DNA analysis is another application of HMM (Eddy, 2004).

The remainder of paper is as follows. In Section 2, we review some necessary background for the paper. Section 3 introduces probabilistic graphical models. Expectation maximization in HMM is explained in Section 4. afterwards, evaluation, estimation, and training in HMM are explain in Sections 5, 6, and 7, respectively. Some applications are introduced in more details in Section 8. Finally, Section 9 concludes the paper.

2. Background

Some background required for better understanding of the HMM algorithms is mentioned in this Section.

2.1. Expectation Maximization

This section is taken from our previous paper (Ghojogh et al., 2019). Sometimes, the data are not fully observ-

able. For example, the data are known to be whether zero or greater than zero. As an illustration, assume the data are collected for a particular disease but for convenience of the patients participated in the survey, the severity of the disease is not recorded but only the existence or non-existence of the disease is reported. So, the data are not giving us complete information as $X_i > 0$ is not obvious whether is $X_i = 2$ or $X_i = 1000$.

In this case, MLE cannot be directly applied as we do not have access to complete information and some data are missing. In this case, Expectation Maximization (EM) (Moon, 1996) is useful. The main idea of EM can be summarized in this short friendly conversation:

– What shall we do? The data are missing! The log-likelihood is not known completely so MLE cannot be used.
– Mmm, probably we can replace the missing data with something...
– Aha! Let us replace it with its mean.
– You are right! We can take the mean of log-likelihood over the possible values of the missing data. Then everything in the log-likelihood will be known, and then...
– And then we can do MLE!

Assume $D^{(obs)}$ and $D^{(miss)}$ denote the observed data (X_i's $= 0$ in the above example) and the missing data (X_i's > 0 in the above example). The EM algorithm includes two main steps, i.e., E-step and M-step.

Let Θ be the parameter which is the variable for likelihood estimation. In the E-step, the log-likelihood $\ell(\Theta)$, is taken expectation with respect to the missing data $D^{(miss)}$ in order to have a mean estimation of it. Let $Q(\Theta)$ denote the expectation of the likelihood with respect to $D^{(miss)}$:

$$Q(\Theta) := \mathbb{E}_{D^{(miss)}|D^{(obs)},\Theta}[\ell(\Theta)]. \tag{1}$$

Note that in the above expectation, the $D^{(obs)}$ and Θ are conditioned on, so they are treated as constants and not random variables.

In the M-step, the MLE approach is used where the log-likelihood is replaced with its expectation, i.e., $Q(\Theta)$; therefore:

$$\widehat{\Theta} = \arg\max_{\Theta} Q(\Theta). \tag{2}$$

These two steps are iteratively repeated until convergence of the estimated parameters $\widehat{\Theta}$.

2.2. Lagrange Multiplier

Suppose we have a multi-variate function $Q(\Theta_1, \ldots, \Theta_K)$ (called "objective function") and we want to maximize (or minimize) it. However, this optimization is constrained and its constraint is equality $P(\Theta_1, \ldots, \Theta_K) = c$ where c is a constant. So, the constrained optimization problem is:

$$\begin{aligned} \underset{\Theta_1,\ldots,\Theta_K}{\text{maximize}} \quad & Q(\Theta_1, \ldots, \Theta_K), \\ \text{subject to} \quad & P(\Theta_1, \ldots, \Theta_K) = c. \end{aligned} \tag{3}$$

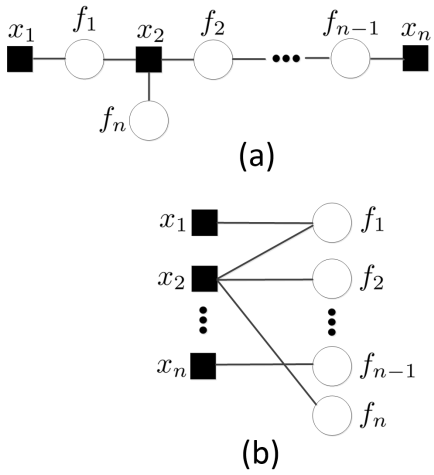

Figure 1. (a) An example factor graph, and (b) its representation as a bipartite graph.

For solving this problem, we can introduce a new variable η which is called "Lagrange multiplier". Also, a new function $\mathcal{L}(\Theta_1, \ldots, \Theta_K, \eta)$, called "Lagrangian" is introduced:

$$\begin{aligned} \mathcal{L}(\Theta_1, \ldots, \Theta_K, \eta) = & Q(\Theta_1, \ldots, \Theta_K) \\ & - \eta\big(P(\Theta_1, \ldots, \Theta_K) - c\big). \end{aligned} \tag{4}$$

Maximizing (or minimizing) this Lagrangian function gives us the solution to the optimization problem (Boyd & Vandenberghe, 2004):

$$\nabla_{\Theta_1,\ldots,\Theta_K,\eta} \mathcal{L} \overset{\text{set}}{=} 0, \tag{5}$$

which gives us:

$$\begin{aligned} \nabla_{\Theta_1,\ldots,\Theta_K} \mathcal{L} \overset{\text{set}}{=} 0 \implies & \nabla_{\Theta_1,\ldots,\Theta_K} Q = \eta \nabla_{\Theta_1,\ldots,\Theta_K} P, \\ \nabla_{\eta} \mathcal{L} \overset{\text{set}}{=} 0 \implies & P(\Theta_1, \ldots, \Theta_K) = c. \end{aligned}$$

2.3. Factor Graph, The Sum-Product and Max-Product Algorithms, and The Forward-Backward Procedure

2.3.1. Factor Graph

A factor graph is a bipartite graph where the two partitions of graph include functions (or factors) $f_j, \forall j$ and the variables $x_i, \forall i$ (Kschischang et al., 2001; Loeliger, 2004). Every factor is a function of two or more variables. The variables of a factor are connected by edges to it. Hence, we have a bipartite graph. Figure 1 shows an example factor graphs where the factor and variable nodes are represented by circles and squares, respectively.

2.3.2. The Sum-Product Algorithm

Assume we have a factor graph with some variable nodes and factor nodes. For this, one of the nodes is considered as

the root of tree. The result of algorithm does not depend on which node to be taken as the root (Bishop, 2006). Usually, the first variable or the last one are considered as the root. Then, some messages from the leaf (or leaves) are sent up on the tree to the root (Bishop, 2006). The messages can be initialized to any fixed constant values (see Eq. (8)). The message from the variable node x_i to its neighbor factor node f_j is:

$$m_{x_i \to f_j} = \prod_{f \in \mathcal{N}(x_i) \setminus f_j} m_{f \to x_i}, \tag{6}$$

where $f \in \mathcal{N}(x_i) \setminus f_j$ means all the neighbor factor nodes to the variable x_i except f_j. Also, $m_{x_i \to f_j}$ and $m_{f \to x_i}$ denote the message from the variable x_i to the factor f_j and the message from the factor f to the variable x_i, respectively. The message from a the factor node f_j to its neighbor variable x_i is:

$$m_{f_j \to x_i} = \sum_{\text{values of } x \in \mathcal{N}(f_j) \setminus x_i} f_j \prod_{x \in \mathcal{N}(f_j) \setminus x_i} m_{x \to f_j}. \tag{7}$$

The Eq. (6) includes both sum and product. This is why this procedure is named the sum-product algorithm (Kschischang et al., 2001; Bishop, 2006). In the factor graph, if the variable x_i has degree one, the message is:

$$m_{x_i \to f_j} = [1, 1, \ldots, 1]^\top \in \mathbb{R}^k, \tag{8}$$

where k is the number of possible values that the variables can get. Moreover, if the variable x_i has degree two connected to f_j and f_ℓ, the message is:

$$m_{x_i \to f_j} = m_{f_\ell \to x_i}. \tag{9}$$

A good example for better understanding of Eqs. (6) and (7) exists in chapter 8 in (Bishop, 2006).

For the exact convergence (and inference) of belief propagation, the graph should be a tree, i.e., should be cycle-free (Kschischang et al., 2001). If the factor graph has cycles, the inference is approximate where the algorithm is stopped after a while manually. The belief propagation in such graphs is called loopy belief propagation (Murphy et al., 1999; Bishop, 2006). Note that Markov chains are cycle-free so the exact belief propagation can be applied for them.

It is also noteworthy that as every variable which is connected to only two factor nodes merely passes the message through (see Eq. (9)), a new graphical model, named normal graph (Forney, 2001), was proposed which states every variable as an edge rather than a node (vertex). More details about factor graph and sum-product algorithm can be found in references (Kschischang et al., 2001; Loeliger, 2004) and chapter 8 of (Bishop, 2006). Also note that some alternatives to sum-product algorithm are min-product, max-product, and min-sum (Bishop, 2006).

```
Initialize the messages to [1, 1, ..., 1]^⊤
for time t from 1 to τ do
    Forward pass: Do sum-product algorithm
    Backward pass: Do sum-product algorithm
    belief = forward message × backward message
    if max(belief change) then
        break the loop
Return beliefs
```

Algorithm 1: The forward-backward algorithm using the sum-product sub-algorithm

2.3.3. The Max-Product Algorithm

The max-product algorithm (Weiss & Freeman, 2001; Pearl, 2014) is similar to the sum-product algorithm where the summation operator is replaced by the maximum operator. In this algorithm, the messages from the variable nodes to the factor nodes and vice versa are:

$$m_{x_i \to f_j} = \prod_{f \in \mathcal{N}(x_i) \setminus f_j} m_{f \to x_i}, \tag{10}$$

$$m_{f_j \to x_i} = \max_{\text{values of } x \in \mathcal{N}(f_j) \setminus x_i} f_j \prod_{x \in \mathcal{N}(f_j) \setminus x_i} m_{x \to f_j}. \tag{11}$$

2.3.4. Belief Propagation with Forward-Backward Procedure

In order to learn the beliefs over the variables and factor nodes, belief propagation can be applied using a forward-backward procedure (see chapter 8 in (Bishop, 2006)). The forward-backward algorithm using the sum-product sub-algorithm is shown in Algorithm 1. In the forward-backward procedure, the belief over a random variable x_i is the product of the forward and backward messages.

3. Probabilistic Graphical Models

3.1. Markov and Bayesian Networks

A Probabilistic Graphical Model (PGM) is a grpah-based representation of a complex distribution in the possibly high dimensional space (Koller & Friedman, 2009). 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 (Koller & Friedman, 2009). In the Markov network and Bayesian network, the edges of graph are undirected and directed, respectively.

3.2. 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:

$$\mathbb{P}(X_1, X_2, \ldots, X_n) = \mathbb{P}(X_1)\,\mathbb{P}(X_2 \,|\, X_1) \\ \mathbb{P}(X_3 \,|\, X_2, X_1) \ldots \mathbb{P}(X_n \,|\, X_{n-1}, \ldots, X_2, X_1), \tag{12}$$

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:

$$\mathbb{P}(X_i \,|\, X_{i-1}, X_{i-2}, \ldots, X_2, X_1) = \mathbb{P}(X_i \,|\, X_{i-1}). \tag{13}$$

Hence, with Markov property, the chain rule is simplified to:

$$\mathbb{P}(X_1, X_2, \ldots, X_n) \\ = \mathbb{P}(X_1)\,\mathbb{P}(X_2 \,|\, X_1)\,\mathbb{P}(X_3 \,|\, X_2) \ldots \mathbb{P}(X_n \,|\, X_{n-1}). \tag{14}$$

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 property is called a Markovian process (or Markov process).

3.3. 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) (Ross, 2014) and Continuous Time Markov Chain (CTMC) (Lawler, 2018). 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) (Booth, 1967). Also, note that the DTMC can be viewed as a Sub-Shifts of Finite Type in modeling dynamic systems (Brin & Stuck, 2002).

3.4. 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. 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:

$$a_{i,j} := \mathbb{P}\big(s_j(t+1) \,|\, s_i(t)\big). \tag{15}$$

We have:

$$\sum_{j=1}^{n} a_{i,j} = 1. \tag{16}$$

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

$$b_{i,j} := \mathbb{P}\big(o_j(t) \,|\, s_i(t)\big), \tag{17}$$

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

$$\sum_{j=1}^{n} b_{i,j} = 1. \tag{18}$$

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

$$\pi_i := \mathbb{P}\big(s_i(1)\big), \tag{19}$$

and:

$$\sum_{i=1}^{n} \pi_i = 1, \tag{20}$$

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 $o^g(t) \in \mathcal{O}, \forall t$.

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:

$$a_{s^g(t),s^g(t+1)} := \mathbb{P}(s^g(t+1) \,|\, s^g(t)). \tag{21}$$

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

$$\pi_{s^g(i)} := \mathbb{P}(s^g(i)). \tag{22}$$

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:

$$b_{s^g(t),o^g(t)} := \mathbb{P}(o^g(t) \,|\, s^g(t)). \tag{23}$$

Note that $s^g(t) \in \mathcal{S}$ and $o^g(t) \in \mathcal{O}$. Figure 2 depicts the structure of an HMM model.

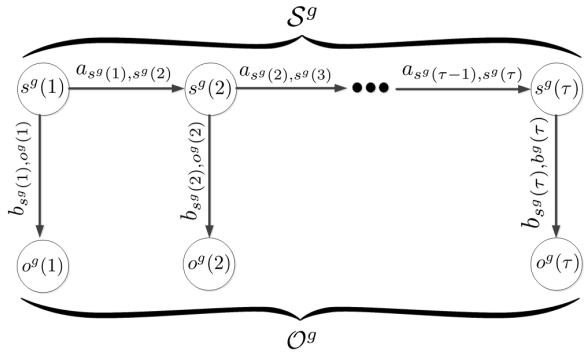

Figure 2. A Hidden Markov Model

4. Likelihood and Expectation Maximization in HMM

The EM algorithm can be used for analysis and training in HMM (Moon, 1996). In the following, we explain the details of EM for HMM.

4.1. Likelihood

According to Fig. 2, the likelihood of occurrence of the state sequence $\mathcal{S}^g$ and the observation sequence $\mathcal{O}^g$ is (Ghahramani, 2001):

$$\begin{aligned} L &= \mathbb{P}(\mathcal{S}^g, \mathcal{O}^g) \\ &= \mathbb{P}\big(s^g(1)\big)\, \mathbb{P}(\text{next states} \,|\, \text{previous states})\, \mathbb{P}(\mathcal{O}^g \,|\, \mathcal{S}^g) \\ &= \mathbb{P}\big(s^g(1)\big) \prod_{t=1}^{\tau-1} \mathbb{P}(s^g(t+1) \,|\, s^g(t)) \prod_{t=1}^{\tau} \mathbb{P}(o^g(t) \,|\, s^g(t)) \\ &= \pi_{s^g(1)} \prod_{t=1}^{\tau-1} a_{s^g(t),s^g(t+1)} \prod_{t=1}^{\tau} b_{s^g(t),o^g(t)}. \end{aligned} \tag{24}$$

The log-likelihood is:

$$\begin{aligned} \ell = \log(L) = \log \pi_{s^g(1)} &+ \sum_{t=1}^{\tau-1} \log(a_{s^g(t),s^g(t+1)}) \\ &+ \sum_{t=1}^{\tau} \log(b_{s^g(t),o^g(t)}). \end{aligned} \tag{25}$$

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:

$$\log \pi_{s^g(1)} = \mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi}, \tag{26}$$

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

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

$$= \mathbf{1}_{s^g(t-1)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t)}, \tag{27}$$

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

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

$$= \mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{B}) \mathbf{1}_{o^g(t)}, \tag{28}$$

where $\mathbf{1}_i[i] = \mathbf{1}_j[j] = 1$. Hence, we can write the log-likelihood as:

$$\begin{aligned} \ell = \mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi} &+ \sum_{t=1}^{\tau-1} \mathbf{1}_{s^g(t-1)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t)} \\ &+ \sum_{t=1}^{\tau} \mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{B}) \mathbf{1}_{o^g(t)}. \end{aligned} \tag{29}$$

4.2. E-step in EM

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:

$$\begin{aligned} Q(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}) &= \mathbb{E}(\ell) \\ &= \mathbb{E}(\mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi}) + \sum_{t=1}^{\tau-1} \mathbb{E}\big(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t+1)}\big) \\ &+ \sum_{t=1}^{\tau} \mathbb{E}\big(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{B}) \mathbf{1}_{o^g(t)}\big). \end{aligned} \tag{30}$$

4.3. 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 $b_{i,j}$:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & Q(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}) \\ \text{subject to} \quad & \sum_{i=1}^{n} \pi_i = 1, \\ & \sum_{j=1}^{n} a_{i,j} = 1, \quad \forall i \in \{1, \ldots, n\}, \\ & \sum_{j=1}^{n} b_{i,j} = 1, \quad \forall i \in \{1, \ldots, n\}, \end{aligned} \tag{31}$$

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

The Lagrangian (Boyd & Vandenberghe, 2004) for this optimization problem is:

$$\mathcal{L} = \mathbb{E}(\mathbf{1}_{s^g(1)}^\top \log \boldsymbol{\pi}) + \sum_{t=1}^{\tau-1} \mathbb{E}\big(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{A}) \mathbf{1}_{s^g(t+1)}\big) + \sum_{t=1}^{\tau} \mathbb{E}\big(\mathbf{1}_{s^g(t)}^\top (\log \boldsymbol{B}) \mathbf{1}_{o^g(t)}\big) - \eta_1 (\sum_{i=1}^{n} \pi_i - 1) - \eta_2 (\sum_{j=1}^{n} a_{i,j} - 1) - \eta_3 (\sum_{j=1}^{n} b_{i,j} - 1), \tag{32}$$

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

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:

$$\frac{\partial \mathcal{L}}{\partial \pi_i} = \mathbb{E}(\mathbf{1}_{s^g(1)}[i]) - \eta_1 \pi_i \stackrel{\text{set}}{=} 0$$
$$\implies \pi_i = \frac{1}{\eta_1} \mathbb{E}(\mathbf{1}_{s^g(1)}[i]). \tag{33}$$

$$\sum_{i=1}^{n} \pi_i = 1 \stackrel{(33)}{\implies} \frac{1}{\eta_1} \big(\mathbb{E}(\mathbf{1}_{s^g(1)}[1]) + \cdots + \mathbb{E}(\mathbf{1}_{s^g(1)}[i]) + \cdots + \mathbb{E}(\mathbf{1}_{s^g(1)}[n])\big)$$
$$= \frac{1}{\eta_1}(0 + \cdots + 1 + \cdots + 0) \stackrel{\text{set}}{=} 1 \implies \eta_1 = 1. \tag{34}$$
$$\therefore \quad \pi_i = \mathbb{E}(\mathbf{1}_{s^g(1)}[i]). \tag{35}$$

Similarly, we have:

$$\frac{\partial \mathcal{L}}{\partial a_{i,j}} = \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\, \mathbf{1}_{s^g(t+1)}[j]) - \eta_2\, a_{i,j} \stackrel{\text{set}}{=} 0$$
$$\implies a_{i,j} = \frac{1}{\eta_2} \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\, \mathbf{1}_{s^g(t+1)}[j]). \tag{36}$$

$$\sum_{j=1}^{n} a_{i,j} = 1 \stackrel{(36)}{\implies} \frac{1}{\eta_2} \sum_{j=1}^{n} \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\, \mathbf{1}_{s^g(t+1)}[j]) =$$
$$= \frac{1}{\eta_2} \sum_{t=1}^{\tau-1} \Big(\mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 0) + \ldots + \underbrace{\mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 1)}_{s^g(t+1)\text{-th element}} + \cdots + \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 0)\Big)$$
$$= \frac{1}{\eta_2} \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]) \stackrel{\text{set}}{=} 1 \implies \eta_2 = \sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]). \tag{37}$$

$$\therefore \quad a_{i,j} = \frac{\sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\, \mathbf{1}_{s^g(t+1)}[j])}{\sum_{t=1}^{\tau-1} \mathbb{E}(\mathbf{1}_{s^g(t)}[i])}. \tag{38}$$

Likewise, we have:

$$\frac{\partial \mathcal{L}}{\partial b_{i,j}} = \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\, \mathbf{1}_{o^g(t)}[j]) - \eta_3\, b_{i,j} \stackrel{\text{set}}{=} 0$$
$$\implies b_{i,j} = \frac{1}{\eta_3} \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\, \mathbf{1}_{o^g(t)}[j]). \tag{39}$$

$$\sum_{j=1}^{n} b_{i,j} = 1 \stackrel{(39)}{\implies} \frac{1}{\eta_3} \sum_{j=1}^{n} \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\, \mathbf{1}_{o^g(t)}[j]) =$$
$$= \frac{1}{\eta_3} \sum_{t=1}^{\tau} \Big(\mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 0) + \cdots + \underbrace{\mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 1)}_{o^g(t)\text{-th element}}$$
$$+ \cdots + \mathbb{E}(\mathbf{1}_{s^g(t)}[i] \times 0)\Big) = \frac{1}{\eta_3} \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]) \stackrel{\text{set}}{=} 1$$
$$\implies \eta_3 = \sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]). \tag{40}$$

$$\therefore \quad b_{i,j} = \frac{\sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i]\, \mathbf{1}_{o^g(t)}[j])}{\sum_{t=1}^{\tau} \mathbb{E}(\mathbf{1}_{s^g(t)}[i])}. \tag{41}$$

In Section 7.1, we will simplify the statements for π_i, $a_{i,j}$, and $b_{i,j}$.

5. Evaluation in HMM

Evaluation in HMM means the following (Rabiner & Juang, 1986; Rabiner, 1989): 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:

$$\mathcal{O}^g, \text{given: } \lambda \implies \mathbb{P}(\mathcal{O}^g \mid \lambda) = ? \tag{42}$$

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.

5.1. 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:

$$\mathbb{P}(\mathcal{O}^g \mid \mathcal{S}^g, \lambda) = b_{s^g(1), o^g(1)}\, b_{s^g(2), o^g(2)} \ldots b_{s^g(\tau), o^g(\tau)}. \tag{43}$$

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

$$\mathbb{P}(\mathcal{S}^g \mid \lambda) = \pi_{s^g(1)}\, a_{s^g(1), s^g(2)} \ldots a_{s^g(\tau-1), s^g(\tau)}. \tag{44}$$

According to chain rule, we have:

$$\mathbb{P}(\mathcal{O}^g, \mathcal{S}^g \,|\, \lambda) = \mathbb{P}(\mathcal{O}^g \,|\, \mathcal{S}^g, \lambda)\, \mathbb{P}(\mathcal{S}^g \,|\, \lambda) = \tag{45}$$

$$\pi_{s^g(1)}\, b_{s^g(1),o^g(1)}\, a_{s^g(1),s^g(2)} \,\dots\, b_{s^g(\tau),o^g(\tau)}\, a_{s^g(\tau-1),s^g(\tau)}, \tag{46}$$

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:

$$\begin{aligned}
\mathbb{P}(\mathcal{O}^g|\,\lambda) &= \sum_{\forall \mathcal{S}^g} \mathbb{P}(\mathcal{O}^g, \mathcal{S}^g \,|\, \lambda) \\
&\overset{(45)}{=} \sum_{\forall \mathcal{S}^g} \mathbb{P}(\mathcal{O}^g \,|\, \mathcal{S}^g, \lambda)\, \mathbb{P}(\mathcal{S}^g \,|\, \lambda) \\
&\overset{(46)}{=} \sum_{\forall s^g(1)} \sum_{\forall s^g(2)} \cdots \sum_{\forall s^g(\tau)} \pi_{s^g(1)}\, b_{s^g(1),o^g(1)}\, a_{s^g(1),s^g(2)} \\
&\qquad \dots\, b_{s^g(\tau),o^g(\tau)}\, a_{s^g(\tau-1),s^g(\tau)}, \qquad (47)
\end{aligned}$$

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.

The time complexity of this direct calculation of $\mathbb{P}(\mathcal{O}^g|\,\lambda)$ is in the order of $O(2\tau n^\tau)$ because at every time clock $t \in \{1, \dots, \tau\}$, there are n possible states to go through (Rabiner & Juang, 1986). Because of n^τ, this is very inefficient especially for long sequences (large τ).

5.2. The Forward-Backward Procedure

A more efficient algorithm for evaluation in HMM is forward-backward procedure (Rabiner & Juang, 1986). The forward-backward procedure includes two stages, i.e., forward and backward belief propagation stages.

5.2.1. The Forward Belief Propagation

Similar to the belief propagation procedure, we define the forward message until time t as:

$$\alpha_i(t) := \mathbb{P}\big(o^g(1), o^g(2), \dots, o^g(t), s^g(t) = s_i \,|\, \lambda\big), \tag{48}$$

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

$$\alpha_i(1) = \pi_i\, b_{i,o^g(1)}, \quad \forall i \in \{1, \dots, n\}, \tag{49}$$

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:

$$\alpha_j(t+1) = \Big[\sum_{i=1}^{n} \alpha_i(t)\, a_{i,j}\Big]\, b_{j,o^g(t+1)}, \tag{50}$$

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

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

which is the probability of occurrence of observation sequence $o^g(1), \dots, o^g(t)$, 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.

Proposition 1. *The Eq. (50) can be interpreted as the sum-product algorithm (see Section 2.3.2).*

Proof. The Algorithm 2 has iterations over states indexed by j. Also the Eq. (50) has sum over states index by i. Consider all states indexed by i and a specific state s_j (see Fig. 3). We can consider every two successive states as a factor node in the factor graph. Hence, a state indexed by i and the state s_j form a factor node which we denote by $f^{i,j}$. The observation symbol $o^g(t+1)$ emitted from s_j is considered as the variable node in the factor graph.

The message $\alpha_i(t)$ is the message received to the factor node $f^{i,j}$ so far. Therefore, in the sum-product algorithm (see Eq. (6)), we have:

$$m_{o^g(t) \to f^{i,j}} = \alpha_i(t). \tag{51}$$

The message $a_{i,j}\, b_{j,o^g(t+1)}$ is the message received from the factor nodes $f^{i,j}, \forall i$ to the variable node $o^g(t+1)$. Therefore, in the sum-product algorithm (see Eq. (7)), we have:

$$m_{f^{i,j} \to o^g(t+1)} = a_{i,j}\, b_{j,o^g(t+1)}. \tag{52}$$

Hence:

$$\begin{aligned}
m_{f \to o^g(t+1)} &= \sum_{i=1}^{n} m_{f^{i,j} \to o^g(t+1)}\, m_{o^g(t) \to f^{i,j}_{\text{next}}} \qquad (53) \\
&= \sum_{i=1}^{n} a_{i,j}\, b_{j,o^g(t+1)} \alpha_i(t) \\
&= \Big[\sum_{i=1}^{n} \alpha_i(t)\, a_{i,j}\Big]\, b_{j,o^g(t+1)}. \qquad (54)
\end{aligned}$$

where f is the set of all factor nodes, $\{f^{i,j}\}_{i=1}^{n}$, and $f^{i,j}_{\text{next}}$ is the factor $f^{i,j}$ in the next time slot.

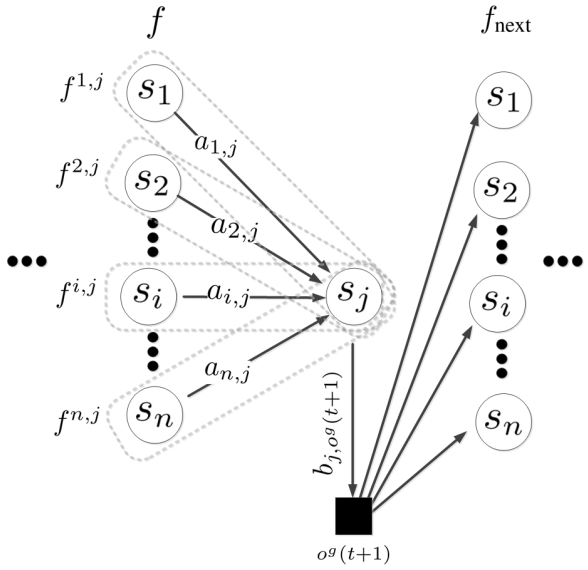

Figure 3. Modeling forward belief propagation for HMM as a sum-product algorithm in a factor graph.

Note that if we consider Fig. 3 for all states indexed by j, a lattice network is formed. □

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

$$\mathbb{P}(\mathcal{O}^g \,|\, \lambda) = \sum_{i=1}^{n} \mathbb{P}\big(\mathcal{O}^g, s^g(\tau) = s_i \,|\, \lambda\big) = \sum_{i=1}^{n} \alpha_i(\tau), \tag{55}$$

which is the desired probability in the evaluation for HMM. Hence, the forward belief propagation suffices for evaluation. In the following, we explain the backward belief propagation which is required for other sections of the paper.

5.2.2. The Backward Belief Propagation

Again similar to the belief propagation procedure, we define the backward message since time τ to $t+1$ as:

$$\beta_i(t) := \mathbb{P}\big(o^g(t+1), o^g(t+2), \ldots, o^g(\tau) \,|\, s^g(t) = s_i, \lambda\big), \tag{56}$$

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. In this algorithm, the initial backward message is:

$$\beta_i(\tau) = 1, \quad \forall i \in \{1, \ldots, n\}. \tag{57}$$

The next backward messages are calculated as:

$$\beta_i(t) = \sum_{j=1}^{n} a_{i,j}\, b_{j,o^g(t+1)}, \tag{58}$$

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

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.

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 (Ghahramani, 2001):

$$\alpha_i(t) \leftarrow \frac{\alpha_i(t)}{\sum_{j=1}^{\tau} \alpha_j(t)}, \tag{59}$$

$$\beta_i(t) \leftarrow \frac{\beta_i(t)}{\sum_{j=1}^{\tau} \beta_j(t)}, \tag{60}$$

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

5.2.3. Time Complexity

The time complexity of the forward belief propagation is in the order of $O(\tau n^2)$ because we have $O(n\tau)$ loops each of which includes summation over n states (Rabiner & Juang, 1986). This is much more efficient than the complexity of direct calculation of $\mathbb{P}(\mathcal{O}^g|\,\lambda)$ which was $O(2\tau n^\tau)$. Similarly, the time complexity of the backward belief propagation is in the order of $O(\tau n^2)$ (Rabiner & Juang, 1986).

6. Estimation in HMM

Estimation in HMM means the following (Rabiner & Juang, 1986; Rabiner, 1989): 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 $\mathbb{P}(\mathcal{S}^g \,|\, \mathcal{O}^g, \lambda)$, i.e., the probability of a state sequence given an observation sequence. In summary:

$$\mathcal{S}^g, \text{given: } \mathcal{O}^g, \lambda \implies \mathbb{P}(\mathcal{S}^g \,|\, \mathcal{O}^g, \lambda) = ? \tag{61}$$

6.1. 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:

$$\gamma_i(t) := \mathbb{P}\big(s^g(t) = s_i \,|\, \mathcal{O}^g, \lambda\big). \tag{62}$$

We can say:

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

$$\implies \gamma_i(t) = \frac{\alpha_i(t)\,\beta_i(t)}{\mathbb{P}(\mathcal{O}^g \mid \lambda)} \tag{63}$$
$$= \frac{\alpha_i(t)\,\beta_i(t)}{\sum_{j=1}^{n}\alpha_j(t)\,\beta_j(t)}, \tag{64}$$

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

$$\alpha_i(t)\,\beta_i(t) \overset{(48),(56)}{=}$$
$$= \mathbb{P}\big(o^g(1),o^g(2),\ldots,o^g(t),s^g(t)=s_i \mid \lambda\big)$$
$$\times\, \mathbb{P}\big(o^g(t+1),o^g(t+2),\ldots,o^g(\tau) \mid s^g(t)=s_i,\lambda\big)$$
$$= \mathbb{P}\big(o^g(1),o^g(2),\ldots,o^g(\tau),s^g(t)=s_i,\lambda\big)$$
$$= \mathbb{P}\big(\mathcal{O}^g,s^g(t)=s_i,\lambda\big).$$

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 (see Section 2.3.4). Note that we have $\sum_{i=1}^{n}\gamma_i(t)=1$. Also, note that $\mathbb{P}(\mathcal{O}^g \mid \lambda)$ in Eq. (63) can be obtained from either the denominator of Eq. (64) or line 6 in Algorithm 2 (i.e., Eq. (55)).

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:

$$s^g(t) = \arg\max_{1\le i\le n}\gamma_i(t), \quad \forall t\in\{1,\ldots,\tau\}, \tag{65}$$

6.2. 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 $\mathbb{P}(\mathcal{O}^g,\mathcal{S}^g \mid \lambda)$ (Rabiner & Juang, 1986). The Viterbi algorithm (Viterbi, 1967; Forney, 1973) can be used to find this path of states (Blunsom, 2004). Different works, such as (He, 1988), have worked on using Viterbi algorithm for HMM.

Algorithm 4 shows the Viterbi algorithm for estimation in HMM (Rabiner & Juang, 1986). In this algorithm, we have variable $\delta_j(t)$:

$$\delta_j(t) = \max_{1\le i\le n}\Big[\delta_i(t-1)\,a_{i,j}\Big]\,b_{j,o^g(t)}, \tag{66}$$

which is similar to $\alpha_j(t)$ defined in Eq. (50), except that $\alpha_j(t)$ in the forward belief propagation uses sum-product algorithm (Kschischang et al., 2001) while $\delta_j(t)$ in the Viterbi algorithm uses max-product algorithm (Weiss & Freeman, 2001; Pearl, 2014) (see Sections 2.3.2 and 2.3.3).

1 **Input:** $\lambda=(\boldsymbol{\pi},\boldsymbol{A},\boldsymbol{B})$
2 // Initialization:
3 $\delta_i(1)=\pi_i\,b_{i,o^g(1)}, \quad \forall i\in\{1,\ldots,n\}$
4 $\psi_i(1)=0, \quad \forall i\in\{1,\ldots,n\}$
5 // Recursion:
6 **for** *state* j *from* 1 *to* n **do**
7 **for** *time* t *from* 2 *to* τ **do**
8 $\delta_j(t)=\max_{1\le i\le n}\big(\delta_i(t-1)\,a_{i,j}\big)\,b_{j,o^g(t)}$
9 $\psi_j(t)=\arg\max_{1\le i\le n}\big(\delta_i(t-1)\,a_{i,j}\big)$
10 // Termination:
11 $p^*=\max_{1\le i\le n}\delta_i(\tau)$
12 $s^*(\tau)=\arg\max_{1\le i\le n}\delta_i(\tau)$
13 // Backtracking:
14 **for** *time* t *from* $\tau-1$ *to* 1 **do**
15 $s^*(t)=\psi_{s^*(t+1)}(t+1)$
16 $\mathbb{P}(\mathcal{O}^g,\mathcal{S}^g \mid \lambda)=p^*$
17 **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

Proposition 2. *The Eq. (66) can be interpreted as the max-product algorithm (see Section 2.3.3).*

Proof. The Algorithm 4 has iterations over states indexed by j. Also the Eq. (66) has maximum operator over states index by i. Consider all states indexed by i and a specific state s_j (see Fig. 4). The definitions of factor node and variable node in the factor graph are similar to the proof of Proposition 1.

The message $\delta_i(t-1)$ is the message received to the factor node $f^{i,j}$ so far. Therefore, in the max-product algorithm (see Eq. (10)), we have:

$$m_{o^g(t-1)\to f^{i,j}} = \delta_i(t-1). \tag{67}$$

The message $a_{i,j}\,b_{j,o^g(t)}$ is the message received from the factor nodes $f^{i,j},\forall i$ to the variable node $o^g(t)$. Therefore, in the max-product algorithm (see Eq. (11)), we have:

$$m_{f^{i,j}\to o^g(t)} = a_{i,j}\,b_{j,o^g(t)}. \tag{68}$$

Hence:

$$m_{f\to o^g(t)} = \max_{1\le i\le n} m_{f^{i,j}\to o^g(t)}\,m_{o^g(t)\to f^{i,j}_{\text{next}}} \tag{69}$$
$$= \max_{1\le i\le n} a_{i,j}\,b_{j,o^g(t)}\delta_i(t-1)$$
$$= \Big[\max_{1\le i\le n}\delta_i(t-1)\,a_{i,j}\Big]\,b_{j,o^g(t)}. \tag{70}$$

where f is the set of all factor nodes, $\{f^{i,j}\}_{i=1}^{n}$, and $f^{i,j}_{\text{next}}$ is the factor $f^{i,j}$ in the next time slot.

Note that if we consider Fig. 4 for all states indexed by j, a lattice network is formed which is common in Viterbi algorithm (see (Rabiner & Juang, 1986)). □

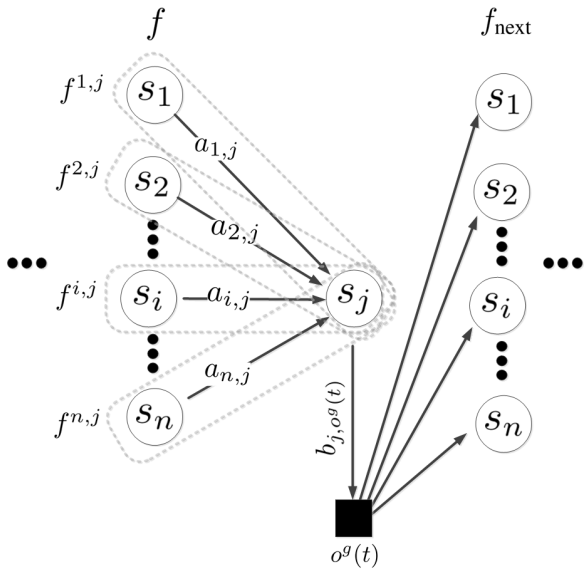

Figure 4. Modeling Viterbi algorithm for HMM as a max-product algorithm in a factor graph. This figure is very similar to Fig. 3 where the difference is in the index of time.

Similar to Eq. (49), the initial $\delta_j(t)$ is:

$$\delta_i(1) = \pi_i \, b_{i,o^g(1)}, \quad \forall i \in \{1, \dots, n\}. \tag{71}$$

We define the index maximizing in Eq. (66) as:

$$\psi_j(t) = \arg \max_{1 \le i \le n} \big(\delta_i(t-1) \, a_{i,j}\big). \tag{72}$$

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

$$p^* = \max_{1 \le i \le n} \delta_i(\tau), \tag{73}$$

$$s^*(\tau) = \arg \max_{1 \le i \le n} \delta_i(\tau), \tag{74}$$

and the other states in the sequence are backtracked as:

$$s^*(t) = \psi_{s^*(t+1)}(t+1). \tag{75}$$

The states $\mathcal{S}^g = s^*(1), s^*(2), \dots, s^*(\tau)$ 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 probability of this path of states with maximum probability of occurrence is:

$$\mathbb{P}(\mathcal{O}^g, \mathcal{S}^g \,|\, \lambda) = p^*. \tag{76}$$

Note that the Viterbi algorithm can be visualized using a trellis structure (see Appendix A in (Jurafsky & Martin, 2019)).

7. Training the HMM

Training HMM means the following (Rabiner & Juang, 1986; Rabiner, 1989): Given the observation sequence $\mathcal{O}^g = o_1^g, \dots, o_\tau^g$, we want to adjust the HMM model parameters $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ in order to maximize $\mathbb{P}(\mathcal{O}^g \,|\, \lambda)$. In summary:

$$\text{given: } \mathcal{O}^g, \mathcal{O}, \mathcal{S} \implies \lambda = \arg\max_{\lambda} \mathbb{P}(\mathcal{O}^g \,|\, \lambda). \tag{77}$$

7.1. The Baum-Welch Algorithm

We can solve for Eq. (77) using maximum likelihood estimation using Expectation Maximization (EM). The Baum-Welch algorithm (Baum et al., 1970) 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:

$$\xi_{i,j}(t) := \mathbb{P}\big(s^g(t) = s_i, s^g(t+1) = s_j \,|\, \mathcal{O}^g, \lambda\big). \tag{78}$$

We can say:

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

$$\implies \xi_{i,j}(t) = \frac{\alpha_i(t) \, a_{i,j} \, b_{j,o^g(t+1)} \, \beta_j(t+1)}{\mathbb{P}(\mathcal{O}^g \,|\, \lambda)} \tag{79}$$

$$= \frac{\alpha_i(t) \, a_{i,j} \, b_{j,o^g(t+1)} \, \beta_j(t+1)}{\sum_{r=1}^{n} \sum_{\ell=1}^{n} \alpha_r(t) \, a_{r,\ell} \, b_{\ell,o^g(t+1)} \, \beta_\ell(t+1)}, \tag{80}$$

where (a) is because of chain rule in probability and (b) is because of the following: According to Eqs. (48), (15), (17), and (56), we have:

$$\begin{aligned}
&\alpha_i(t) = \mathbb{P}(o^g(1), \dots, o^g(t), s^g(t) = s_i \,|\, \lambda), \\
&a_{i,j} = \mathbb{P}(s_j(t+1) \,|\, s_i(t)), \\
&b_{j,o^g(t+1)} = \mathbb{P}(o^g(t+1) \,|\, s_j(t+1)), \\
&\beta_j(t+1) = \mathbb{P}(o^g(t+2), \dots, o^g(\tau) \,|\, s^g(t+1) = s_j, \lambda).
\end{aligned}$$

Therefore, we have:

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

Note that we have $\sum_{i=1}^{n} \sum_{j=1}^{n} \xi_{i,j}(t) = 1$. Also, note that $\mathbb{P}(\mathcal{O}^g \,|\, \lambda)$ in Eq. (79) can be obtained from either the denominator of Eq. (80) or line 6 in Algorithm 2 (i.e., Eq. (55)).

In Eq. (79), 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.

Now, recall the Eqs. (35), (38), and (41) from EM algorithm for HMM. We write these equations again for convenience of the reader:

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

On the other hand, recall Eqs. (62) and (78) which are repeated here:

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

Therefore, we can say:

$$\mathbb{E}(\mathbf{1}_{s^g(1)}[i]) = \gamma_i(1), \tag{81}$$
$$\mathbb{E}(\mathbf{1}_{s^g(t)}[i]) = \gamma_i(t), \tag{82}$$
$$\mathbb{E}(\mathbf{1}_{s^g(t)}[i]\,\mathbf{1}_{s^g(t+1)}[j]) = \xi_{i,j}(t), \tag{83}$$
$$\mathbb{E}(\mathbf{1}_{s^g(t)}[i]\,\mathbf{1}_{o^g(t)}[j]) = \gamma_i(t), \text{ where } o^g(t) = j \text{ in } \gamma_i(t). \tag{84}$$

Hence:

$$\pi_i = \gamma_i(1) \overset{(62)}{=} \mathbb{P}\big(s^g(1) = s_i \,|\, \mathcal{O}^g, \lambda\big), \quad \forall i \in \{1, \dots, n\}, \tag{85}$$

$$a_{i,j} = \frac{\sum_{t=1}^{\tau-1} \xi_{i,j}(t)}{\sum_{t=1}^{\tau-1} \gamma_i(t)}, \quad \forall i, j \in \{1, \dots, n\}, \tag{86}$$

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

With change of variable, we have:

$$b_{j,k} = \frac{\sum_{t=1,\, o^g(t)=k}^{\tau} \gamma_j(t)}{\sum_{t=1}^{\tau} \gamma_j(t)}, \quad \forall j \in \{1, \dots, n\}, \; \forall k \in \{1, \dots, m\}. \tag{87}$$

The algorithm is shown in Algorithm 5 (Rabiner & Juang, 1986). In this algorithm the initial probabilities of being in state s_i at time $t = 1$ is according to Eq. (85). Then the $a_{i,j}$ is then calculated using Eq. (86). 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. (87). 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.

1 **Input:** $\gamma_i(t), \xi_{i,j}(t), \forall i, \forall j, \forall t$
2 $\pi_i = \gamma_i(1), \quad \forall i \in \{1, \dots, n\}$
3 **for** *state* i *from* 1 *to* n **do**
4 **for** *state* j *from* 1 *to* n **do**
5 $a_{i,j} = \sum_{t=1}^{\tau-1} \xi_{i,j}(t) / \sum_{t=1}^{\tau-1} \gamma_i(t)$
6 **for** *state* j *from* 1 *to* n **do**
7 **for** *observation* k *from* 1 *to* m **do**
8 $b_{j,k} = \sum_{t=1,\, o^g(t)=k}^{\tau} \gamma_j(t) / \sum_{t=1}^{\tau} \gamma_j(t)$
9 // Normalization, for computer error corrections:
10 $\pi_i \leftarrow \pi_i / \sum_{j=1}^{n} \pi_j$
11 $a_{i,j} \leftarrow a_{i,j} / \sum_{\ell=1}^{n} a_{i,\ell}$
12 $b_{j,k} \leftarrow b_{j,k} / \sum_{\ell=1}^{m} b_{j,\ell}$
13 $\boldsymbol{\pi} = [\pi_1, \dots, \pi_n]^\top, \boldsymbol{A} = [a_{i,j}], \boldsymbol{B} = [b_{j,k}], \quad \forall i, j \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\}$
14 **Return** $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$

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

Note that in the numerator of Eq. (87), we have $o^g(t) = k$ in $\gamma_j(t)$, i.e., according to Eq. (62), we have $\gamma_j(t) = \mathbb{P}\big(s^g(t) = s_j \,|\, o^g(1), \dots, o^g(t) = k, \dots, o^g(\tau), \lambda\big)$. This means that according to Eq. (63), we set $o^g(t) = k$ in Eqs. (50) and (58), or in Algorithms 2 and 3, for calculation of $\gamma_j(t)$ in Eq. (87). On the other hand, in line 8 in Algorithm 5, we are iterating over all the time slots $t \in \{1, \dots, \tau\}$. Hence, in the numerator of Eq. (87), we will have $\gamma_j(t) = \mathbb{P}\big(s^g(t) = s_j \,|\, o^g(1) = k, \dots, o^g(t) = k, \dots, o^g(\tau) = k, \lambda\big)$. Therefore:

- For calculating the numerator of Eq. (87), we use Eq. (64) where:

$$\alpha_j(t+1) = \Big[\sum_{i=1}^{n} \alpha_i(t)\, a_{i,j}\Big]\, b_{j,k}, \tag{88}$$

$$\beta_i(t) = \sum_{j=1}^{n} a_{i,j}\, b_{j,k}, \tag{89}$$

  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 the numerator of Eq. (64).

- For calculating the denominator of Eq. (87), we use Eq. (64) where Algorithms 2 and 3 are used for calculating $\alpha_i(t)$ and $\beta_i(t)$.

It is noteworthy that because of a possible error caused by computer, it is better to normalize the obtained HMM model parameters (see lines 10 to 12 in Algorithm 5) in order to make sure that Eqs. (20), (16), and (18) are satisfied.

7.2. Training the HMM

We know that Algorithm 5 requires $\gamma_i(t)$ and $\xi_{i,j}(t)$. According to Eqs. (63) and (79), the $\gamma_i(t)$ and $\xi_{i,j}(t)$ also require $\alpha_i(t)$ and $\beta_i(t)$. The $\alpha_i(t)$ and $\beta_i(t)$ can be computed using Algorithms (2) and (3), respectively. Moreover, note that in calculating $\alpha_i(t)$, $\beta(i)$, and $\xi_{i,j}(t)$, we need some π_i, $a_{i,j}$, and $b_{i,j}$ from $\boldsymbol{\pi}$, $\boldsymbol{A}$, and $\boldsymbol{B}$, respectively. Therefore, we need an initial $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ to compute another $\lambda' = (\boldsymbol{\pi}', \boldsymbol{A}', \boldsymbol{B}')$ at the end according to Algorithm 5. Thus, we should fine tune $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ iteratively. The obtained λ' from the previous λ satisfies $\mathbb{P}(\mathcal{O}^g \mid \lambda') \geq \mathbb{P}(\mathcal{O}^g \mid \lambda)$ (Rabiner & Juang, 1986) (see (Baum & Eagon, 1967) or (Baum et al., 1970) for proof) so we have progress in convergence (note that $\mathbb{P}(\mathcal{O}^g \mid \lambda)$ is obtained for each iteration in line 6 in Algorithm 6). If we have several training sequences $\mathcal{Q}$, indexed by $q \in \{1, \dots, |\mathcal{Q}|\}$, we use one of the sequences at the first iteration, the second sequence at the second iteration, and so on. We repeat this procedure until convergence which is reached when there is no significant change in λ. Algorithm 6 shows how to train an HMM.

Input: $\{\mathcal{O}^g = o_1^g, \dots, o_\tau^g\}_{q=1}^{|\mathcal{Q}|}$
Initialize $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ where $\sum_{i=1}^n \pi_i = 1$, $\sum_{j=1}^n a_{i,j} = 1$, $\sum_{j=1}^n b_{i,j} = 1$
while *Convergence* **do**
  for *each* q *from* 1 *to* $|\mathcal{Q}|$ **do**
    Consider the q-th training sequence
    $\mathbb{P}(\mathcal{O}^g \mid \lambda), \forall i, \forall t : \alpha_i(t) \leftarrow$ Do Algorithm 2 [input: λ]
    $\forall i, \forall t : \beta_i(t) \leftarrow$ Do Algorithm 3 [input: λ]
    $\forall i, \forall t : \gamma_i(t) \leftarrow$ Eq. (63) [input: α, β]
    $\forall i, \forall j, \forall t : \xi_{i,j}(t) \leftarrow$ Eq. (79) [input: α, β, λ]
    $\lambda \leftarrow$ Do Algorithm 5 [input: γ, ξ]
    if *change of* λ *is small* **then**
      Convergence $\leftarrow$ True
Return $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$

Algorithm 6: Training the HMM

8. Usage of HMM in Applications

In Section 1, we introduced some application of HMM in speech recognition (Rabiner & Juang, 1986; Rabiner, 1989; Gales et al., 2008), action recognition (Yamato et al., 1992; Ghojogh et al., 2017), and face recognition (Nefian & Hayes, 1998; Samaria, 1994). Here, we explain the applications in speech and action recognition in more details.

8.1. Application in Speech Recognition

Assume we have a dictionary of words consisting of $|\mathcal{W}|$ words. For every word indexed by $w \in \{1, \dots, |\mathcal{W}|\}$, we have $|\mathcal{Q}_w|$ training instances spoken by one or several people. The training instances for a word are indexed by q where $q \in \{1, \dots, |\mathcal{Q}_w|\}$. Every training instance is a sequence of observation symbols obtained from formants (Titze & Martin, 1998). We consider an HMM model for every word in the dictionary. Training the HMMs are as (Rabiner & Juang, 1986; Rabiner, 1989):

1. For every word w_i, consider the training sequences, $\mathcal{O}_w^g$, indexed by q, i.e., $o_1^g, \dots, o_{|\mathcal{Q}_w|}^g$.
2. Train the HMM for the w-th word using Algorithm 6 to obtain λ_w.

For an unknown test word with sequence $\mathcal{O}_t^g = o_1^g, \dots, o_{|\mathcal{Q}_t|}^g$, we recognize the word as (Rabiner & Juang, 1986; Rabiner, 1989):

1. Calculate $\mathbb{P}(\mathcal{O}_t^g \mid \lambda_w)$ for all $w \in \{1, \dots, |\mathcal{W}|\}$ using the forward belief propagation, i.e., Algorithm 2 (see Eq. (55)).
2. The test word is recognized as:
$$w^* = \arg\max_w \mathbb{P}(\mathcal{O}_t^g \mid \lambda_w). \tag{90}$$
So, the test word is recognized as the w-th word in the dictionary.

In test phase for speech recognition, usually, the Viterbi algorithm is used (Rabiner, 1989). 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 $w \in \{1, \dots, |\mathcal{W}|\}$ using the Viterbi algorithm, i.e., Algorithm 4 (see Eq. (76)).
2. The test word is recognized as:
$$w^* = \arg\max_w \mathbb{P}(\mathcal{O}_t^g, \mathcal{S}_t^g \mid \lambda_w). \tag{91}$$
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.

8.2. 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 (Ghojogh et al., 2017). Hence, HMM can be used for action recognition (Yamato et al., 1992).

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

Assume we have a set of actions denoted by $\mathcal{W}$ where the actions are indexed by $w \in \{1, \dots, |\mathcal{W}|\}$. We have $|\mathcal{Q}_w|$ training instances for every action where the training instances are indexed by $q \in \{1, \dots, |\mathcal{Q}_w|\}$. 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 (Ghojogh et al., 2017; Mokari et al., 2018). 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.

In action recognition, we have a dataset of actions consisting of several defined poses (Ghojogh et al., 2017; Mokari et al., 2018). For example, if the dataset includes three actions sit, stand, and turn, the format of actions is as follows:

- Action sit: $\underbrace{\text{stand}, \text{stand} \dots, \text{stand}}_{\text{stand}}, \underbrace{\text{sit}, \text{sit} \dots, \text{sit}}_{\text{sit}}$
- Action stand: $\underbrace{\text{sit}, \text{sit} \dots, \text{sit}}_{\text{sit}}, \underbrace{\text{stand}, \text{stand} \dots, \text{stand}}_{\text{stand}}$
- Action turn: $\underbrace{\text{stand}, \text{stand} \dots, \text{stand}}_{\text{stand}}, \underbrace{\text{tilt}, \text{tilt} \dots, \text{tilt}}_{\text{tilt}}$

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.

9. Conclusion

In this paper, we explained the theory of HMM for evaluation, estimation, and training. We started with some required background, i.e., EM, factor graphs, sum-product and max-product algorithms, forward-backward propagation, Markov and Bayesian networks, Markov property, and DTMC. We then introduced HMM and detailed EM in HMM. Evaluation in HMM was explained in both direct calculation and forward-backward procedure. We introduced estimation in HMM using the greedy approach and the Viterbi algorithm. Training the HMM was also covered using the Baum-Welch algorithm based on EM algorithm. We also introduced speech and action recognition as two popular applications of HMM.

Acknowledgment

The authors hugely thank Prof. Mehdi Molkaraie, Prof. Kevin Granville, and Prof. Mu Zhu whose courses partly covered the materials mentioned in this tutorial paper.

References

Baum, Leonard E and Eagon, John Alonzo. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, 1967.

Baum, Leonard E, Petrie, Ted, Soules, George, and Weiss, Norman. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The annals of mathematical statistics*, 41(1): 164–171, 1970.

Bishop, Christopher M. *Pattern recognition and machine learning*. Springer, 2006.

Blunsom, Phil. Hidden Markov models. Technical report, 2004.

Booth, Taylor L. *Sequential machines and automata theory*. New York, NY: Wiley, 1967.

Boyd, Stephen and Vandenberghe, Lieven. *Convex optimization*. Cambridge university press, 2004.

Brin, Michael and Stuck, Garrett. *Introduction to dynamical systems*. Cambridge university press, 2002.

Eddy, Sean R. What is a hidden Markov model? *Nature biotechnology*, 22(10):1315, 2004.

Forney, G David. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

Forney, G David. Codes on graphs: Normal realizations. *IEEE Transactions on Information Theory*, 47(2):520–548, 2001.

Gales, Mark, Young, Steve, et al. The application of hidden Markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304, 2008.

Ghahramani, Zoubin. An introduction to hidden Markov models and Bayesian networks. In *Hidden Markov models: applications in computer vision*, pp. 9–41. World Scientific, 2001.

Ghojogh, Benyamin, Mohammadzade, Hoda, and Mokari, Mozhgan. Fisherposes for human action recognition using kinect sensor data. *IEEE Sensors Journal*, 18(4): 1612–1627, 2017.

Ghojogh, Benyamin, Ghojogh, Aydin, Crowley, Mark, and Karray, Fakhri. Fitting a mixture distribution to data: tutorial. *arXiv preprint arXiv:1901.06708*, 2019.

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

Jurafsky, Daniel and Martin, James H. *Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing*. Pearson Prentice Hall, 2019.

Koller, Daphne and Friedman, Nir. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Kschischang, Frank R, Frey, Brendan J, and Loeliger, Hans-Andrea. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2): 498–519, 2001.

Lawler, Gregory F. *Introduction to stochastic processes*. Chapman and Hall/CRC, 2018.

Loeliger, H-A. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, 2004.

Mokari, Mozhgan, Mohammadzade, Hoda, and Ghojogh, Benyamin. Recognizing involuntary actions from 3d skeleton data using body states. *Scientia Iranica*, 2018.

Moon, Todd K. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

Murphy, Kevin P, Weiss, Yair, and Jordan, Michael I. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 467–475. Morgan Kaufmann Publishers Inc., 1999.

Nefian, Ara V and Hayes, Monson H. Hidden markov models for face recognition. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 5, pp. 2721–2724. IEEE, 1998.

Pearl, Judea. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

Rabiner, Lawrence R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Rabiner, Lawrence R and Juang, Biing-Hwang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

Ross, Sheldon M. *Introduction to probability models*. Academic press, 2014.

Roweis, Sam. Hidden Markov models. Technical report, University of Toronto, 2003.

Samaria, Ferdinando Silvestro. *Face recognition using hidden Markov models*. PhD thesis, University of Cambridge, Cambridge, UK, 1994.

Titze, Ingo R and Martin, Daniel W. *Principles of voice production*. Acoustical Society of America, 1998.

Viterbi, Andrew. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

Weiss, Yair and Freeman, William T. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001.

Yamato, Junji, Ohya, Jun, and Ishii, Kenichiro. 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.