# Hidden Markov Model (HMM)

Zhe Feng

November 22, 2020

## 1 Introduction

Hidden Markov Model (HMM) is one of the most commonly used family of models used in various areas such as signal processing and automatic speech recognition (ASR). It is a special case of Bayesian network, but is sufficiently generalised to describe many real-world problems. At a high level, it describes systems whose actual states are not observable to us (hence hidden), yet indirectly, it correlates to some observable results (and ccthus called observation). The states evolve over time with transition probability and thus the observations. A solid example would be in ASR, each word/character are hidden states, where observations are the voices. The key assumption of HMM is that the current state only dependent on the previous state, but not dependent on any states before the previous state. e.g. in ASR, it's equivalent of saying current word/character only dependent on previous word/character. Obviously this is not a realistic assumption, but practically it works pretty well. Besides, one could reduce the temporal order by increase the dimensionality (similar trick used in dynamic systems - one can always transfer a higher order differential equation system, to an first order differential equation by increase the dimension of the states, that's why you always say dynamic system is described as $\dot{x} = f(x)$ with $x$ being a state vector).

## 2 Problem description

### 2.1 Model form

In this article, we focus on simplest form of HMM, where there are 2 discrete variables:

- $x \in \{1, 2, \cdot, N\}$ the state, where 1, 2 etc. represent each state, e.g. 1 can be happy, 2 can be sad.
- $y \in \{1, 2, \cdot, M\}$ the observation where the value indicates the observation value (e.g. 1 can be raining, 2 can be sunshine etc.).

Note that here both $x$ and $y$ takes discrete value here, this is the standard setting for HMM, but generalisable to continuous values, it's just in those situation, the computation will become an issue for practical use. With the above variable defined, each HMM is usually described by a triple:

- $\pi = \{\pi_1, \pi_2, \cdots, \pi_N\}$: the initial probability distribution of the state, where $\pi_i = P(x_1 = i)$
- $A \in [0,1]^{N \times N}$: the state transition matrix, where each element $a_{i,j} = P(x = j|x = i)$
- $B \in [0,1]^{N \times M}$: the emission matrix, where each element $b_{i,j} = P(y = j|x = i)$

There's also one key assumption, which we will keep refer to:

**Assumption 1.** *HMM is based on the following two key assumptions:*

- *Current observation only depends on current hidden state:*

$$P(y_t|y_1, y_1, \cdots, y_{t-1}, y_{t+1}, \cdots y_T, x_{1:T}) = P(y_t|x_t) \tag{1}$$

- *Current hidden state only dependent on previous state:*

$$P(x_t|x_{1:t-1}) = P(x_t|x_{t-1}) \tag{2}$$

With this formulation, 3 fundamental problems of HMM are of interest:

1. **Decoding problem:** given a model $\{\pi, A, B\}$ and observations $\{y_t\}_{t=1:T}$ (here I used a shorthand notation: $\{y_t\}_{t=1:T} := \{y_1, y_1, \cdots, y_T\}$, we will also use $y_{1:T}$ in many cases to further simplify if not confusion will rise), what is the most likely state sequence $\{x_t\}_{t=1:T}$ (similarly, we will use $x_{1:T}$ as a short hand) is generated by the $k$th component?
2. **Learning problem:** given a set of $T$ observations $\{y_t\}_{t=1:T}$, what was the most likely model $\{\pi, A, B\}$.
3. **Evaluation problem:** given a model $\{\pi, A, B\}$, what is the probability of observing a particular sequence, i.e. find $P(y_1, y_1, \cdots, y_T|\pi, A, B)$.

The 3rd problem is in general of less interest in the area of machine learning, the first and the second problem is equivalent to prediction problem and learning (maybe with some variation) problem.

## 3 The decoding problem and the optimisation problem under the hood

In decoding problem, with the model $\{\pi, A, B\}$ and a set of observations $\{y_t\}_{t=1:T}$ known, we want to find what's the most likely hidden state sequence $\{x_t\}_{t=1:T}$. In a more formal way of stating this:

$$\max_{x_{1:T}} P(x_{1:T}|y_{1:T}) \tag{3}$$

here we use $x_{1:T}$ instead of $\{x_t\}_{t=1:T}$ to further simplify notation.

### 3.1 Solve the optimisation problem with dynamic programming

The optimisation problem (3) miximize conditional probability, which is a bit hard to solve (you can try to solve it directly). Instead, the above problem can be translated into a simpler problem. We know that

$$P(x_{1:T}|y_{1:T}) = \frac{P(x_{1:T}, y_{1:T})}{P(y_{1:T})} \tag{4}$$

and since $y_{1:T}$ is known, $P(y_{1:T})$ becomes a positive constant between 0 and 1 (strictly larger than 0, otherwise we will not be able to observe the sequence). This means that the optimisation problem (3) is equivalent to maximise the joint probability:

$$\max_{x_{1:T}} P(x_{1:T}, y_{1:T}) \tag{5}$$

The above problem has a nicer struture. When in conjunction with the assumptions of HMM in Assumption 1, this can be solved use dynamic programming.

To do that, we first try to break the jioint probability optimisation problem a bit further:

$$\max_{x_{1:T}} P(x_{1:T}, y_{1:T}) = \max_{x_{1:T}} P(x_T, y_T, x_{1:T-1}, y_{1:T-1}) \tag{6}$$
$$= \max_{x_{1:T}} P(x_T, y_T|x_{1:T-1}, y_{1:T-1})P(x_{1:T-1}, y_{1:T-1}) \tag{7}$$
$$= \max_{x_{1:T}} P(y_T|x_T, x_{1:T-1}, y_{1:T-1})P(x_T|x_{1:T-1}, y_{1:T-1})P(x_{1:T-1}, y_{1:T-1}) \tag{8}$$
$$= \max_{x_{1:T}} P(y_T|x_T)P(x_T|x_{T-1})P(x_{1:T-1}, y_{1:T-1}) \tag{9}$$
$$= \max_{x_T, x_{T-1}} P(y_T|x_T) \max_{x_{1:T-2}} P(x_T|x_{T-1})P(x_{1:T-1}, y_{1:T-1}) \tag{10}$$

Now we can see a nice recursive structure start to show. If we define a value function:

$$V_t(x_t) := \max_{x_{1:t-1}} P(x_t, x_{1:t-1}, y_{1:t}) \tag{11}$$

Note that the optimisation function does not optmise over $x_t$, instead it's a function of $x_t$, this facilitate the dynamic program's recursive structure. With the above, the optimisation problem (6) can be written as:

$$\max_{x_{1:T}} P(x_{1:T}, y_{1:T}) = \max_{x_T} \max_{x_{1:T-1}} P(x_T, x_{1:T-1}, y_{2:T}) = \max_{x_T} V_T(x_T) \tag{12}$$

Now we find the equivalence between our value function $V_T(x_T)$ and the maximisation problem for joint probability (6), we now need to establish the recursive sub-problems.

For any $t \in \{2, \cdots, T\}$, we can formulate the following structure:

$$\begin{align} V_t(x_t) &= \max_{x_{1:t-1}} P(x_t, x_{1:t-1}, y_{1:t}) \tag{13} \\ &= \max_{x_{1:t-1}} P(y_t|x_t)P(x_t|x_{t-1})P(x_{t-1}, x_{1:t-1}, y_{1:t-1}) \tag{14} \\ &= \max_{x_{t-1}} P(y_t|x_t)P(x_t|x_{t-1}) \max_{x_{1:t-2}} P(x_{t-1}, x_{1:t-1}, y_{1:t-1}) \tag{15} \\ &= \max_{x_{t-1}} P(y_t|x_t)P(x_t|x_{t-1})V_{t-1}(x_{t-1}) \tag{16} \end{align}$$

and when $t = 1$, since $x_1$ is generated by initial probability $\pi$ and thus no optimisation problem in this case:

$$V_1(x_1) = P(y_1|x_1)\pi(x_1) \tag{17}$$

So in summary, at a high level, the algorithm can be written as:

1. calculate first value function $V_1(x_1) = P(y_1|x_1)\pi(x_1)$
2. recursively, calculate the value function $V_t(x_t) = \max_{x_{t-1}} P(y_t|x_t)P(x_t|x_{t-1})V_{t-1}(x_{t-1})$ for $t \in \{2, \cdots, T-1\}$ and record $x_{t-1} = \arg\max_{x_{t-1}} P(y_t|x_t)P(x_t|x_{t-1})V_{t-1}(x_{t-1})$
3. calculate the final value function $\max_{x_T} V_T(x_T)$ and record $x_T = \arg\max_{x_T} V_T(x_T)$
4. the recorded state sequence $\{x_1, x_2, \cdots x_T\}$ maximise the joint probability $P(x_{1:T}, y_{1:T})$

Up to this step, the deduction is pretty general for any model satisfy HMM assumptions, e.g. it's applicable to Gaussian HMM as well.

The hurdle, however, is that in this general form, the problem $\max_{x_{t-1}} P(y_t|x_t)P(x_t|x_{t-1})V_{t-1}(x_{t-1})$ may not necessarily easy to solve. This is because variable $x_t$ is considered as an unknown constant in the optimisation problem and it prevents the problem to be solved numerically.

For this document we will focus on the discrete form of HMM (and the standard form), and recall the format of the mode is $\{\pi_0, A, B\}$ with

$$P(x = j|x = i) = a_{i,j},\ P(y = j|x = i) = b_{i,j}, \tag{18}$$

with that, the recursive optimisation problem can be siplified to

$$\begin{aligned}
V_1(x_1 = i) &:= V_{1,i} = P(y_1|x_1 = i)\pi_i \\
V_t(x_t = j) &:= V_{t,j} = \max_i P(y_t|x_t = j)P(x_t = j|x_{t-1} = i)V_{t-1}(x_{t-1}) \\
&= \max_i P(y_t|x_t = j)a_{i,j}V_{t-1,i}
\end{aligned}$$

and in this case, $V_t(x_t)$ become $V_{t,j}$ we can record $V_{t,j}$ for all possible value of $j$ for the calculation in next iteration.

1. calculate $V_{1,i} = P(y_1|x_1 = i)\pi_i$ for all $i$, and store them in an vector $\mathbf{V}_t := \begin{bmatrix} V_{t,1} \\ V_{t,2} \\ \vdots \\ V_{t,N} \end{bmatrix} \in \mathbb{R}^N$.
2. calculate $V_{t,j} = \max_i P(y_t|x_t = j)a_{i,j}V_{t-1,i}$ and $x_{t-1} = \arg\max_i P(y_t|x_t = j)a_{i,j}V_{t-1,i}$, we also record $\mathbf{V}_t$ for all $j$

As always, for computational purposes, matrix calculation is always better than loops. To facilitate the computation, assume at time $t$ we have $y_t = k$. Then

$$\begin{aligned}
\mathbf{V}_t = \begin{bmatrix} V_{t,1} \\ V_{t,2} \\ \vdots \\ V_{t,N} \end{bmatrix} &= \begin{bmatrix} \max_i b_{1,k}a_{i,1}V_{t-1,i} \\ \max_i b_{2,k}a_{i,2}V_{t-1,i} \\ \vdots \\ \max_i b_{N,k}a_{i,N}V_{t-1,i} \end{bmatrix} \\
&= \max_{\text{index of each row}} \begin{bmatrix} b_{1,k}a_{1,1}V_{t-1,1} & b_{1,k}a_{2,2}V_{t-1,2} & \cdots & b_{1,k}a_{N,1}V_{t-1,N} \\ b_{2,k}a_{1,2}V_{t-1,1} & b_{2,k}a_{2,2}V_{t-1,2} & \cdots & b_{2,k}a_{N,2}V_{t-1,N} \\ \vdots & \ddots & & \\ b_{N,k}a_{1,N}V_{t-1,1} & b_{N,k}a_{2,N}V_{t-1,2} & \cdots & b_{N,k}a_{N,N}V_{t-1,N} \end{bmatrix} \\
&= \max_{\text{index of each row}} [b_{1:N,k}, \cdots b_{1:N,k}]_N^\top \circ A^\top \circ [\mathbf{V}_{t-1}, \cdots \mathbf{V}_{t-1}]_N^\top
\end{aligned}$$

where $[b_{1:N,k}, \cdots b_{1:N,k}]_N \in \mathbb{R}^{N\times N}$ is the $k$th column of $B$ matrix, repeated $N$ times and $[\mathbf{V}_{t-1}, \cdots \mathbf{V}_{t-1}]_N \in \mathbb{R}^{N\times N}$ is the $\mathbf{V}_{t-1}$ repeated $N$ times and $\circ$ notes the element-wise product.

Also to facilitate the recovery of the optimal state sequence, we define the following quantity:

$$\begin{aligned}
\hat{x}_{t-1,i} &:= \arg\max_j V_{t-1}(x_t = i|x_{t-1} = j) \\
&= \arg\max_j P(y_t = k|x_t = i)P(x_t = i|x_{t-1} = j)V_{t-1}(x_{t-1} = j) \\
&= \arg\max_j b_{i,k}a_{j,i}V_{t-1,j} \forall t \in \{2, 3 \cdots, T\} \\
x_T^* &:= \arg\max_i V_T(x_T = i)
\end{aligned}$$

where $\hat{x}_{t-1,i}$ represents the $x_{t-1}$ that optimise $V_{t-1}(x_t = i)$, and $x_T^*$ is the optimal state at time $T$ that maximize $V_T$, becasue this is the terminal state so we know the best value (besides, there's no $x_{T+1}$). Since we know the optimal state $x_T = x_T^*$, we can trace back to deduce the optimal $x_{T-1}^*$:

$$x_{T-1}^* = \arg\max_j V_{t-1}(x_T = x_T^* | x_{T-1} = j) = \hat{x}_{T-1,x_T^*}$$

Hence we can then determine a backtracking algorithm with the following step for all $t$:

$$x_{t-1}^* = \hat{x}_{t-1,x_t^*},\ t \in \{T, T-1, \cdots, 2\}$$

Finally, we can write the programmable algorithm as follows:

1. calculate $\mathbf{V}_1 = B \circ \boldsymbol{\pi}$.
2. If $y_t = k$, then calculate $\mathbf{V}_t = \max_{\text{element of each row}}[b_{1:N,k}, \cdots b_{1:N,k}]_N^\top \circ A^\top \circ [\mathbf{V}_{t-1}, \cdots \mathbf{V}_{t-1}]_N^\top,\ t \in \{2, \cdots, T\}$.
   Also record $\hat{\mathbf{x}}_{t-1} = \arg\max_{\text{index of each row}}[b_{1:N,k}, \cdots b_{1:N,k}]_N^\top \circ A^\top \circ [\mathbf{V}_{t-1}, \cdots \mathbf{V}_{t-1}]_N^\top$
3. Compute optimal end state $x_T^* = \arg\max_i V_T(x_T = i)$
4. Backtrace the optimal sequence of $x_t$ by finding the $x_{t-1}$ that optimise the value function $V_{t,j}$ which is $x_{t-1}^* = \hat{x}_{t-1,x_t^*}$