Introduction to Hidden Markov Models

Alperen Degirmenci

This document contains derivations and algorithms for implementing Hidden Markov Models. The content presented here is a collection of my notes and personal insights from two seminal papers on HMMs by Rabiner in 1989 [2] and Ghahramani in 2001 [1], and also from Kevin Murphy's book [3]. This is an excerpt from my project report for the MIT 6.867 Machine Learning class taught in Fall 2014.

I. Hidden Markov Models (HMMs)

HMMs have been widely used in many applications, such as speech recognition, activity recognition from video, gene finding, gesture tracking. In this section, we will explain what HMMs are, how they are used for machine learning, their advantages and disadvantages, and how we implemented our own HMM algorithm.

A. Definition

A hidden Markov model is a tool for representing probability distributions over sequences of observations [1]. In this model, an observation X_t at time t is produced by a stochastic process, but the state Z_t of this process cannot be directly observed, i.e. it is *hidden* [2]. This hidden process is assumed to satisfy the Markov property, where state Z_t at time t depends only on the previous state, Z_{t-1} at time $t-1$. This is, in fact, called the first-order Markov model. The n^{th}-order Markov model depends on the n previous states. Fig. 1 shows a Bayesian network representing the first-order HMM, where the hidden states are shaded in gray.

We should note that even though we talk about "time" to indicate that observations occur at discrete "time steps", "time" could also refer to locations within a sequence [3].

The joint distribution of a sequence of states and observations for the first-order HMM can be written as,

$$P(Z_{1:N}, X_{1:N}) = P(Z_1)P(X_1|Z_1)\prod_{t=2}^{N} P(Z_t|Z_{t-1})P(X_t|Z_t) \tag{1}$$

where the notation $Z_{1:N}$ is used as a shorthand for $Z_1, \dots, Z_N$. Notice that Eq. 1 can be also written as,

$$P(X_{1:N}, Z_{1:N}) = P(Z_1)\prod_{t=2}^{N} P(Z_t|Z_{t-1})\prod_{t=1}^{N} P(X_t|Z_t) \tag{2}$$

which is same as the expression given in the lecture notes.

There are five elements that characterize a hidden Markov model:

The author is with the School of Engineering and Applied Sciences at Harvard University, Cambridge, MA 02138 USA. (adegirmenci@seas.harvard.edu). This document is an excerpt from a project report for the MIT 6.867 Machine Learning class taught in Fall 2014.

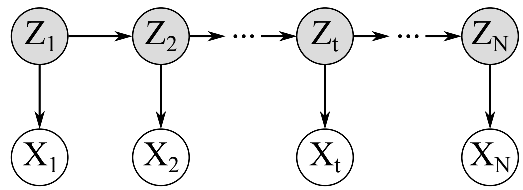

Fig. 1. A Bayesian network representing a first-order HMM. The hidden states are shaded in gray.

1) Number of states in the model, K: This is the number of states that the underlying hidden Markov process has. The states often have some relation to the phenomena being modeled. For example, if a HMM is being used for gesture recognition, each state may be a different gesture, or a part of the gesture. States are represented as integers $1, \dots, K$. We will encode the state Z_t at time t as a $K \times 1$ vector of binary numbers, where the only non-zero element is the k-th element (i.e. $Z_{tk} = 1$), corresponding to state $k \in K$ at time t. While this may seem contrived, it will later on help us in our computations. (Note that [2] uses N instead of K).

2) Number of distinct observations, Ω: Observations are represented as integers $1, \dots, \Omega$. We will encode the observation X_t at time t as a $\Omega \times 1$ vector of binary numbers, where the only non-zero element is the l-th element (i.e. $X_{tl} = 1$), corresponding to state $l \in \Omega$ at time t. While this may seem contrived, it will later on help us in our computations. (Note that [2] uses M instead of Ω, and [1] uses D. We decided to use Ω since this agrees with the lecture notes).

3) State transition model, $\mathbf{A}$: Also called the state transition probability distribution [2] or the transition matrix [3], this is a $K \times K$ matrix whose elements A_{ij} describe the probability of transitioning from state $Z_{t-1,i}$ to $Z_{t,j}$ in one time step where $i, j \in \{1, \dots, K\}$. This can be written as,

$$A_{ij} = P(Z_{t,j} = 1|Z_{t-1,i} = 1). \tag{3}$$

Each row of $\mathbf{A}$ sums to 1, $\sum_j A_{ij} = 1$, and therefore it is called a stochastic matrix. If any state can reach any other state in a single step (fully-connected), then $A_{ij} > 0$ for

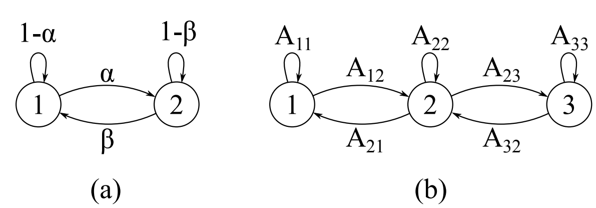

Fig. 2. A state transition diagram for (a) a 2-state, and (b) a 3-state ergodic Markov chain. For a chain to be ergodic, any state should be reachable from any other state in a finite amount of time.

all i, j; otherwise $\mathbf{A}$ will have some zero-valued elements. Fig. 2 shows two state transition diagrams for a 2-state and 3-state first-order Markov chain. For these diagrams, the state transition models are,

$$A_{(a)} = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}, A_{(b)} = \begin{bmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{bmatrix}. \quad (4)$$

The conditional probability can be written as

$$P(Z_t|Z_{t-1}) = \prod_{i=1}^{K}\prod_{j=1}^{K} A_{ij}^{Z_{t-1,i}Z_{t,j}}. \quad (5)$$

Taking the logarithm, we can write this as

$$\begin{aligned} logP(Z_t|Z_{t-1}) &= \sum_{i=1}^{K}\sum_{j=1}^{K} Z_{t-1,i}Z_{t,j}\log A_{ij} && (6)\\ &= Z_t^\top \log(\mathbf{A})Z_{t-1}. && (7)\end{aligned}$$

4) Observation model, $\mathbf{B}$*:* Also called the emission probabilities, $\mathbf{B}$ is a $\Omega \times K$ matrix whose elements B_{kj} describe the probability of making observation $X_{t,k}$ given state $Z_{t,j}$. This can be written as,

$$B_{kj} = P(X_t = k|Z_t = j). \quad (8)$$

The conditional probability can be written as

$$P(X_t|Z_t) = \prod_{j=1}^{K}\prod_{k=1}^{\Omega} B_{kj}^{Z_{t,j}X_{t,k}}. \quad (9)$$

Taking the logarithm, we can write this as

$$\begin{aligned} logP(X_t|Z_t) &= \sum_{j=1}^{K}\sum_{k=1}^{\Omega} Z_{t,j}X_{t,k}\log B_{kj} && (10)\\ &= X_t^\top \log(\mathbf{B})Z_t. && (11)\end{aligned}$$

5) Initial state distribution, π*:* This is a $K \times 1$ vector of probabilities $\pi_i = P(Z_{1i=1})$. The conditional probability can be written as,

$$P(Z_1|\pi) = \prod_{i=1}^{K} \pi_i^{Z_{1i}}. \quad (12)$$

Given these five parameters presented above, an HMM can be completely specified. In literature, this often gets abbreviated as

$$\lambda = (A, B, \pi). \quad (13)$$

B. Three Problems of Interest

In [2] Rabiner states that for the HMM to be useful in real-world applications, the following three problems must be solved:

- *Problem 1:* Given observations $X_1, \ldots, X_N$ and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(X_{1:N}|\lambda)$, the probability of the observations given the model? This is a part of the *exact inference* problem presented in the lecture notes, and can be solved using forward filtering.

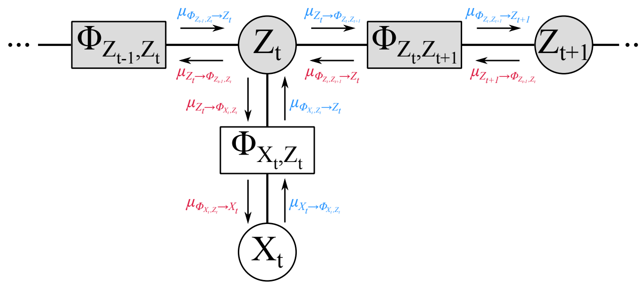

Fig. 3. Factor graph for a slice of the HMM at time t.

- *Problem 2:* Given observations $X_1, \ldots, X_N$ and the model λ, how do we find the "correct" hidden state sequence $Z_1, \ldots, Z_N$ that best "explains" the observations? This corresponds to finding the most probable sequence of hidden states from the lecture notes, and can be solved using the Viterbi algorithm. A related problem is calculating the probability of being in state Z_{tk} given the observations, $P(Z_t = k|X_{1:N})$, which can be calculated using the forward-backward algorithm.
- *Problem 3:* How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(X_{1:N}|\lambda)$? This corresponds to the learning problem presented in the lecture notes, and can be solved using the Expectation-Maximization (EM) algorithm (in the case of HMMs, this is called the Baum-Welch algorithm).

C. The Forward-Backward Algorithm

The forward-backward algorithm is a dynamic programming algorithm that makes use of message passing (belief propagation). It allows us to compute the filtered and smoothed marginals, which can be then used to perform inference, MAP estimation, sequence classification, anomaly detection, and model-based clustering. We will follow the derivation presented in Murphy [3].

1) The Forward Algorithm: In this part, we compute the filtered marginals, $P(Z_t|X_{1:T})$ using the *predict-update cycle*. The prediction step calculates the *one-step-ahead predictive density*,

$$\begin{aligned} &P(Z_t = j|X_{1:t-1}) = \cdots \\ &\qquad \sum_{i=1}^{K} = P(Z_t = j|Z_{t-1} = i)P(Z_{t-1} = i|X_{1:t-1}) \end{aligned} \quad (14)$$

which acts as the new prior for time t. In the update state, the observed data from time t is absorbed using Bayes rule:

$$\begin{aligned} \alpha_t(j) &\triangleq P(Z_t = j|X_{1:t}) \\ &= P(Z_t = j|X_t, X_{1:t-1}) \\ &= \frac{P(X_t|Z_t = j, \cancel{X_{1:t-1}})P(Z_t = j|X_{1:t-1})}{\sum_j P(X_t|Z_t = j, \cancel{X_{1:t-1}})P(Z_t = j|X_{1:t-1})} \\ &= \frac{1}{C_t}P(X_t|Z_t = j)P(Z_t = j|X_{1:t-1}) \end{aligned} \quad (15)$$

Algorithm 1 Forward algorithm

Input: $\boldsymbol{A}$, $\psi_{1:N}$, $\boldsymbol{\pi}$
$[\boldsymbol{\alpha}_1,\, C_1]$ = normalize($\psi_1 \odot \boldsymbol{\pi}$) ;
for $t = 2 : N$ **do**
    $[\boldsymbol{\alpha}_t,\, C_t]$ = normalize($\psi_t \odot (\boldsymbol{A}^\top \boldsymbol{\alpha}_{t-1})$) ;
Return $\boldsymbol{\alpha}_{1:N}$ and $\log P(X_{1:N}) = \sum_t \log C_t$
Sub: $[\boldsymbol{\alpha},\, C]$ = normalize($\boldsymbol{u}$): $C = \sum_j u_j$; $\alpha_j = u_j / C$;

Algorithm 2 Backward algorithm

Input: $\boldsymbol{A}$, $\psi_{1:N}$, $\boldsymbol{\alpha}$
$\beta_N = 1$;
for $t = N-1 : 1$ **do**
    β_t = normalize($\boldsymbol{A}(\psi_{t+1} \odot \beta_{t+1})$;
$\boldsymbol{\gamma}$ = normalize($\boldsymbol{\alpha} \odot \boldsymbol{\beta}$, 1)
Return $\boldsymbol{\gamma}_{1:N}$

where the observations $X_{1:t-1}$ cancel out because they are *d-separated* from X_t. C_t is the normalization constant (to avoid confusion, we used C_t as opposed to Z_t from [3]) given by,

$$C_t \triangleq P(X_t | X_{1:t-1}) = \sum_{j=1}^{K} = P(X_t | Z_t = j) P(Z_t = j | X_{1:t-1}). \tag{16}$$

The $K \times 1$ vector $\boldsymbol{\alpha}_t = P(Z_t | X_{1:T})$ is called the (filtered) *belief state* at time t.

In matrix notation, we can write the recursive update as:

$$\boldsymbol{\alpha}_t \propto \psi_t \odot \left(\boldsymbol{A}^\top \boldsymbol{\alpha}_{t-1} \right) \tag{17}$$

where $\psi_t = [\psi_{t1}, \psi_{t2}, \dots, \psi_{tK}] = \{P(X_t | Z_t = i)\}_{1 \le i \le K}$ is the local evidence at time t which can be calculated using Eq. 9, $\boldsymbol{A}$ is the transition matrix, and $\odot$ is the Hadamard product, representing elementwise vector multiplication. The pseudo-code in Algorithm 1 outlines the steps of the computation.

The $\log$ probability of the evidence can be computed as

$$\log P(X_{1:N} | \lambda) = \sum_{t=1}^{N} \log P(X_t | X_{1:t-1}) = \sum_{t=1}^{N} \log C_t \tag{18}$$

This, in fact, is the solution for *Problem 1* stated by Rabiner [2]. Working in the $\log$ domain allows us to avoid numerical underflow during computations.

2) The Forward-Backward Algorithm: Now that we have the filtered belief states $\boldsymbol{\alpha}$ from the forward messages, we can compute the backward messages to get the smoothed marginals:

$$\begin{aligned} P(Z_t = j | X_{1:N}) &\propto P(Z_t = j . X_{t+1:N} | X_{1:t}) \qquad (19) \\ &\propto P(Z_t = j | X_{1:t}) P(X_{t+1:N} | Z_t = j, \cancel{X_{1:t}}). \end{aligned}$$

which is the probability of being in state Z_{tj}. Given that the hidden state at time t is j, define the conditional likelihood of future evidence as

$$\beta_t(j) \triangleq P(X_{t+1:N} | Z_t = j). \tag{20}$$

Also define the desired smoothed posterior marginal as

$$\gamma_t(j) \triangleq P(Z_t = j | X_{1:N}). \tag{21}$$

Then we can rewrite Eq. 19 as

$$\gamma_t(j) \propto \alpha_t(j) \beta_t(j) \tag{22}$$

We can now compute the β's recursively from right to left:

$$\begin{aligned} \beta_{t-1}(i) &= P(X_{t:N} | Z_{t-1} = i) \\ &= \sum_j P(Z_t = j, X_t, X_{t+1:N} | Z_{t-1} = i) \\ &= \sum_j P(X_{t+1:N} | Z_t = j, \cancel{X_t}, \cancel{Z_{t-1} = j}) \\ &\qquad \cdots P(Z_t = j, X_t | Z_{t-1} = i) \\ &= \sum_j P(X_{t+1:N} | Z_t = j) P(X_t | Z_t = j, \cancel{Z_{t-1} = i}) \\ &\qquad \cdots P(Z_t = j | Z_{t-1} = i) \\ &= \sum_j \beta_t(j) \psi_t(j) \boldsymbol{A}(i, j) \end{aligned} \tag{23}$$

This can be written as

$$\beta_{t-1} = \boldsymbol{A} \left(\psi_t \odot \beta_t \right) \tag{24}$$

The base case for β_N is

$$\beta_N(i) = P(X_{N+1:N} | Z_N = i) = P(\emptyset | Z_N = i) = 1 \tag{25}$$

Finally, the smoothed posterior is then

$$\boldsymbol{\gamma}_i = \frac{\boldsymbol{\alpha_i} \odot \boldsymbol{\beta_i}}{\sum_j \left(\boldsymbol{\alpha}_i(j) \odot \boldsymbol{\beta}_i(j) \right)} \tag{26}$$

where the denominator ensures that each column of γ sums to 1 to ensure it is a stochastic matrix. The pseudo-code in Algorithm 2 outlines the steps of the computation.

D. The Viterbi Algorithm

In order to compute the most probable sequence of hidden states (*Problem 2*), we will use the Viterbi algorithm. This algorithm computes the shortest path through the trellis diagram of the HMM. The trellis diagram shows how each state in the model at one time step connects to the states in the next time step. In this section, we again follow the derivation presented in Murphy [3].

The Viterbi algorithm also has a forward and backward pass. In the forward pass, instead of the sum-product algorithm, we utilize the max-product algorithm. The backward pass recovers the most probable path through the trellis diagram using a traceback procedure, propagating the most likely *state* at time t back in time to recursively find the most likely *sequence* between times $1:t$. This can be expressed as,

$$\delta_t(j) \triangleq \max_{Z_1, \dots, Z_{t-1}} P(Z_{1:t-1}, Z_t = j | X_{1:t}). \tag{27}$$

This probability can be expressed as a combination of the transition from the previous state i at time $t-1$ and the most

Algorithm 3 Viterbi algorithm

Input: $X_{1:N}$, K, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\pi}$
Initialize: $\delta_1 = \boldsymbol{\pi} \odot B_{X_1}$, $a_1 = \mathbf{0}$;
for $t = 2 : N$ **do**
  for $j = 1 : K$ **do**
    $[a_t(j),\ \delta_t(j)] = \max_i(\log \delta_{t-1}(:) + \log \mathbf{A}_{ij} + \log B_{X_t}(j))$;
$Z_N^* = \arg\max(\delta_N)$;
for $t = N-1 : 1$ **do**
  $Z_t^* = a_{t+1} Z_{t+1}^*$;
Return $Z_{1:N}^*$

probable path leading to i,

$$\delta_t(j) = \max_{1 \le i \le K} \delta_{t-1}(i) \mathbf{A}_{ij} \mathbf{B}_{X_t}(j). \tag{28}$$

Here $B_{X_t}(j)$ is the emission probability of observation X_t given state j. We also need to keep track of the most likely previous state i,

$$a_t(j) = \arg\max_i \delta_{t-1}(i) \mathbf{A}_{ij} \mathbf{B}_{X_t}(j). \tag{29}$$

The initial probability is

$$\delta_1(j) = \pi_j \mathbf{B}_{X_1}(j). \tag{30}$$

The most probable final state is

$$Z_N^* = \arg\max_i \delta_N(i). \tag{31}$$

The most probable sequence can be computing using trace-back,

$$Z_t^* = a_{t+1} Z_{t+1}^*. \tag{32}$$

In order to avoid underflow, we can work in the log domain. This is one of the advantages of the Viterbi algorithm, since $\log \max = \max \log$; this is not possible with the forward-backward algorithm since $\log \sum \neq \sum \log$. Therefore

$$\log \delta_t(j) \triangleq \max_i \log \delta_{t-1}(i) + \log \mathbf{A}_{ij} + \log \mathbf{B}_{X_t}(j). \tag{33}$$

The pseudo-code in Algorithm 3 outlines the steps of the computation.

E. The Baum-Welch Algorithm

The Baum-Welch algorithm is in essence the Expectation-Maximization (EM) algorithm for HMMs. Given a sequence of observations $X_{1:N}$, we would like to find

$$\arg\max_\lambda P(X;\lambda) = \arg\max_\lambda \sum_Z P(X,Z;\lambda) \tag{34}$$

by doing maximum-likelihood estimation. Since summing over all possible Z is not possible in terms of computation time, we use EM to estimate the model parameters.

The algorithm requires us to have the forward and backward probabilities α, β calculated using the forward-backward algorithm. In this section we follow the derivation presented in Murphy [3] and the lecture notes.

Algorithm 4 Baum-Welch algorithm

Input: $X_{1:N}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$
for $t = 1 : N$ **do**
  $\gamma(:,t) = (\alpha(:,t) \odot \beta(:,t))./sum(\alpha(:,t) \odot \beta(:,t))$;
  $\xi(:,:,t) = ((\alpha(:,t) \odot A(t+1)) * (\beta(:,t+1) \odot B(X_{t+1}))^T)./sum(\alpha(:,t) \odot \beta(:,t))$;
$\hat{\pi} = \gamma(:,1)./sum(\gamma(:,1))$;
for $j = 1 : K$ **do**
  $\hat{A}(j,:) = sum(\xi(2:N,j,:),1)./sum(sum(\xi(2:N,j,:),1),2)$;
  $B(\hat{j},:) = \left(X(:,j)^T \boldsymbol{\gamma}\right)./sum(\boldsymbol{\gamma},1)$;
Return $\hat{\pi}$, $\hat{A}$, $\hat{B}$

1) E Step:

$$\begin{aligned} \gamma_{tk} &\triangleq P(Z_{tk} = 1 | X, \lambda^{old}) \\ &= \frac{\alpha_k(t)\beta_k(t)}{\sum_{j=1}^N \alpha_j(t)\beta_j(t)} \end{aligned} \tag{35}$$

$$\begin{aligned} \xi_{tjk} &\triangleq P(Z_{t-1,j} = 1, Z_{tk} = 1 | X, \lambda^{old}) \\ &= \frac{\alpha_j(t) A_{jk} \beta_k(t+1) B_k(X_{t+1})}{\sum_{i=1}^N \alpha_i(t)\beta_i(t)} \end{aligned} \tag{36}$$

2) M Step: The parameter estimation problem can be turned into a constrained optimization problem where $P(X_{1:N}|\lambda)$ is maximized, subject to the stochastic constraints of the HMM parameters [2]. The techniques of Lagrange multipliers can be then used to find the model parameters, yielding the following expressions:

$$\hat{\pi}_k = \frac{\mathbb{E}[N_k^1]}{N} = \frac{\gamma_{1k}}{\sum_{j=1}^K \gamma_{1j}} \tag{37}$$

$$\hat{A}_{jk} = \frac{\mathbb{E}[N_{jk}]}{\sum_{k'} \mathbb{E}[N_{jk}]} = \frac{\sum_{t=2}^N \xi_{tjk}}{\sum_{l=1}^K \sum_{t=2}^N \xi_{tjl}} \tag{38}$$

$$\hat{B}_{jl} = \frac{\mathbb{E}[M_{jl}]}{\mathbb{E}[N_j]} = \frac{\sum_{t=1}^N \gamma_{tl} X_{tj}}{\sum_{t=1}^N \gamma_{tl}} \tag{39}$$

$$\lambda^{new} = (\hat{A}, \hat{B}, \hat{\lambda}) \tag{40}$$

The pseudo-code in Algorithm 4 outlines the steps of the computation.

F. Limitations

A fully-connected transition diagram can lead to severe overfitting. [1] explains this by giving an example from computer vision, where objects are tracked in a sequence of images. In problems with large parameter spaces like this, the transition matrix ends up being very large. Unless there are lots of examples in the data set, or unless some *a priori* knowledge about the problem is used, then this leads to severe overfitting. A solution to this is to use other types of HMMs, such as factorial or hierarchical HMMs.

References

[1] Z. Ghahramani, "An Introduction to Hidden Markov Models and Bayesian Networks," International Journal of Pattern Recognition and Artificial Intelligence, vol. 15, no. 1, pp. 9–42, 2001.

[2] L. 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.
[3] K.P. Murphy, *Machine Learning: A Probabilistic Perspective*, Cambridge, MA: MIT Press, 2012.